104

PYTHON PROGRAMS FOR BEGINNERS

CMR Institute of Technology

#132, ITPL Main Road, Bengaluru-560037, India

https://www.cmrit.ac.in/

INDEX

96	Class Circle	Difficult	251
97	Add Two Time Objects	Difficult	253
98	Linear Search	Easy	254
99	Binary Search	Difficult	255
100	Bubble Sort	Difficult	257
101	Selection Sort	Difficult	258
102	Insertion Sort	Difficult	259
103	Merge Sort	Difficult	260
104	Quick Sort	Difficult	261

0.All Formats

This chapter does not contain any programs.

This chapter summarizes several frequently used operations and functions in Python programming.

If you are already familiar with Python, you may skip this chapter.

1 How to input a string in Jupyter Notebook

[1]:
```
#By default, input() function takes string data type

#Letters,Space,Numbers,Special Characters - all are considered as string

s=input("Enter a string : ")
```

```
Enter a string : raveesh hegde
```

2 How to input a string in online coding tests

[2]:
```
s=input()

#In online coding tests, do not type any statement inside input() function

#In online coding tests, input data will be taken automatically, you need not␣
↪enter any data manually
```

```
raveesh hegde
```

3 How to check the data type of s

[3]:
```
print(type(s))
```

```
<class 'str'>
```

4 How to input an integer in Jupyter Notebook

[4]:
```
n=int(input("Enter an integer"))

#By defualt, input() function takes string data type

#If you want an integer, convert string to integer using int() function
```

```
Enter an integer2
```

5 How to input an integer in online coding tests

[5]:
```
n=int(input())

#In online coding tests, do not type any statement inside input() function

#In online coding tests, input data will be taken automatically, you need not␣
↪enter any data manually
```

```
2
```

6 How to check the data type of n

[6]:
```
print(type(n))
```

```
<class 'int'>
```

7 How to input a float in Jupyter Notebook

[7]:
```
x=float(input("Enter a float : "))

#By defualt, input() function takes string data type

#If you want a float, convert string to float using float() function
```

```
Enter a float : 2.3
```

8 How to input a float in online coding tests

[8]:
```
x=float(input())

#In online coding tests, do not type any statement inside input() function

#In online coding tests, input data will be taken automatically, you need not␣
↪enter any data manually
```

2.3

9 How to input multiple strings in Jupyter Notebook

Format 1 : Multiple strings in a single line separated by space

[9]:
```
list1=input("Enter multiple strings separated by space : ")

#At this point, list1 is a single string. Even space is considered as a string.
```

```
Enter multiple strings separated by space : raveesh hegde cmrit ece 123
```

[10]:
```
#Now, we split the string and create a list

list1=list1.split()
```

[11]:
```
#To view the list
print(list1)

# Now, we have a list of strings
```

```
['raveesh', 'hegde', 'cmrit', 'ece', '123']
```

10 How to input multiple strings in online coding tests

Format 1 : Multiple strings in a single line separated by space

For Ex. :

raveesh hegde cmrit ece 123 !@#$%^&*()

[12]:
```
#In online coding tests, do not type any statement inside input() function

#In online coding tests, input data will be taken automatically, you need not enter any data manually

list1=input() #Enter multiple strings separated by space

list1=list1.split() #After split(), we get a list
```

```
raveesh hegde cmrit ece 123
```

11 How to input multiple strings in online coding tests

Format 2 : Multiple strings in a single line separated by comma

For Ex. :

raveesh,hegde,cmrit,ece,123 !@#$%^&*()

[13]:
```
#In online coding tests, do not type any statement inside input() function

#In online coding tests, input data will be taken automatically, you need not enter any data manually

list1=input() #Enter multiple strings separated by comma

list1=list1.split(",")   #After split(), we get a list
```

raveesh,hegde,cmrit,ece,123

12 How to input multiple strings in online coding tests

Format 3 : Multiple strings entered one below the other (Press enter key after entering every string)

For Ex. :

raveesh

hegde

cmrit

ece

123

!@#$%^&*()

[15]:
```
n=int(input()) #Take the length of the list as input

list1=[] #Start with an empty list

for i in range(n):
    list1.append(input())

#In online coding tests, input data will be taken automatically, you need not enter any data manually
```

5
raveesh
hegde
cmrit
ece
123

13 How to input multiple strings in online coding tests

Format 4 : Multiple strings within square brackets

For Ex. : ['raveesh','hegde','cmrit','ece']

[16]:
```
n=int(input()) #Take the length of the list as input

list1=[] #Start with an empty list

for i in range(n):
    list1.append(input()) #Press 'Enter' key after typing every element

#In online coding tests, input data will be taken automatically, you need not enter any data manually
```

```
5
raveesh
hegde
cmrit
ece
123
```

14 How to input multiple integers in online coding tests

Format 1 : Multiple integers in a single line separated by space

For Ex. :

2 4 6 8

[17]:
```
list1=input() #Enter multiple integers separated by space

list1=list1.split() #After split(), we get a list

list1=[int(i) for i in list1] #Convert every element of the list to integer

#In online coding tests, input data will be taken automatically, you need not enter any data manually
```

```
2 4 6 8
```

15 How to input multiple integers in online coding tests

Format 2 : Multiple integers in a single line separated by comma

For Ex. :

2,4,6,8

[18]:
```
list1=input() #Enter multiple integers separated by comma

list1=list1.split(",") #After split(), we get a list
```

```
list1=[int(i) for i in list1] #Convert every element of the list to integer

#In online coding tests, input data will be taken automatically, you need not enter any data manually
```

2,4,6,8

16 How to input multiple integers in online coding tests

Format 3 : Multiple integers entered one below the other (Press enter key after entering every integer)

For Ex. :

2

4

6

8

[19]:
```
n=int(input()) #Take the length of the list as input

list1=[] #Start with an empty list

for i in range(n):
    list1.append(int(input()))

#In online coding tests, input data will be taken automatically, you need not enter any data manually
```

```
4
2
4
6
8
```

17 How to input multiple integers in online coding tests

Format 4 : Multiple integers within square brackets

For Ex. : [2,4,6,8]

[20]:
```
n=int(input()) #Take the length of the list as input

list1=[] #Start with an empty list

for i in range(n):
    list1.append(int(input())) #Press 'Enter' key after typing every element
```

```
#In online coding tests, input data will be taken automatically, you need not␣
↪enter any data manually
```

```
4
2
4
6
8
```

18 How to print elements of a list in online coding tests

Format 1 : Elements inside square brackets, separated by space

For Ex. :

[2,4,6,8]

[21]:
```
print(list1)
```

```
[2, 4, 6, 8]
```

19 How to print elements of a list in online coding tests

Format 2 : Elements in a single line separated by space

For Ex. :

2 4 6 8

[22]:
```
print(*list1)
```

```
2 4 6 8
```

20 How to print elements of a list in online coding tests

Format 3 : Elements in a single line separated by comma, but NO comma after the last element

For Ex. :

2,4,6,8

[23]:
```
print(*list1,sep=",")
```

```
2,4,6,8
```

21 How to print elements of a list in online coding tests

Format 3 : Elements in a single line separated by comma and comma after the last element

For Ex. :

2,4,6,8,

[24]:
```
for i in list1:
    print(i,end=",")
```

2,4,6,8,

22 How to print elements of a list in online coding tests

Format 4 : Elements one below the other

For Ex. :

2

4

6

8

[25]:
```
print(*list1,sep="\n")
```

2
4
6
8

23 How to print elements of a list in online coding tests

Format 4 : Elements one below the other and comma after every element

For Ex. :

2,

4,

6,

8,

[26]:
```
for i in list1:
    print("{},".format(i))
```

2,
4,
6,
8,

24 Important Note :

There are multiple ways to input and output data in Python.

Which method should we follow?

In online coding tests, along with questions, sample test cases will also be given.

We have to follow the method which matches the format of sample input and output test cases.

25 Variables

1. In Python, there is no need to declare a variable. We can directly assign a value to it.
2. A variable name must start with a letter or the underscore character
3. A variable name cannot start with a number
4. A variable name can only contain alpha-numeric characters and underscores (A to z, 0 to 9, and _)
5. Variable names are case-sensitive (age, Age and AGE are three different variables)
6. A variable name cannot be any of the Python keywords.
7. Type of a variable can change during the course of a program

[27]:
```
x=5

y=5.0

z="Raveesh"

a=3>2
```

26 Operators

Following are the built-in operators in python

1. Arithmetic operators
2. Assignment operators
3. Comparison operators
4. Logical operators
5. Identity operators
6. Membership operators
7. Bitwise operators

27 Arithmetic Operators

[28]:
```
x=2

y=3

# Addition

z=x+y

print(z)
```

```
5
```

[29]:
```
# Subtraction

z=x-y

print(z)
```

```
-1
```

[30]:
```
# Multiplication

z=x*y

print(z)
```

```
6
```

[31]:
```
# Division

z=x/y

print(z)
```

```
0.6666666666666666
```

[32]:
```
# Modulus Operator

# This gives the remainder

z=x%y

print(z)
```

```
2
```

[33]:
```
# Floor division

# This gives the quotient

z=x//y

print(z)
```

```
0
```

[34]:
```
# Exponentiation

z=x**y #x to the power y

print(z)
```

```
8
```

28 Assignment Operators

[35]:
```
x=2

print(x)
```

```
2
```

[36]:
```
x+=3 # This means x=x+3

print(x)
```

```
5
```

[37]:
```
# Similarly -=, *=, /=, %=, //=, **= all are valid
```

29 Comparison Operators

[38]:
```
x=2

y=3
```

[39]:
```
# Check whether x is equal to y

x==y

#This will result in either True or False
```

[39] : False

[40] :
```
# Check whether x is not equal to y

x!=y

#This will result in either True or False
```

[40] : True

[41] :
```
# Check whether x is greater than y

x>y

#This will result in either True or False
```

[41] : False

[42] :
```
# Check whether x is less than y

x<y

#This will result in either True or False
```

[42] : True

[43] :
```
# Check whether x is greater than or equal to y

x>=y

#This will result in either True or False
```

[43] : False

[44] :
```
# Check whether x is less than or equal to y

x<=y

#This will result in either True or False
```

[44] : True

30 Logical Operators : and, or

[45] :
```
print(x)

print(y)
```

```
2
3
```

[46] :
```
# and operator : returns true if both conditions are met

x>0 and y<3
```

[46] : False

[47] :
```
# or operator : returns true if at least one condition is met

x>0 or y<3
```

[47] : True

[48] :
```
# not operator : reverse the result

not(x>0 or y<3)
```

[48] : False

31 Identity Operators - is, is not

[49] :
```
x = ["apples", "oranges"]

print(x)

y = ["apples", "oranges"]

print(y)

z = x

print(z)
```

```
['apples', 'oranges']
['apples', 'oranges']
['apples', 'oranges']
```

[50] :
```
print(x is z)

# Returns True because z is the same object as x
```

True

[51] :
```
print(x is y)

# Returns False because x is not the same object as y, even though they have the␣
↪same content
```

False

[52] :
```
print(x == y)

# Returns True because content of x is same as content of y
```

True

[53] :
```
print(x is not z)

# Returns False because z is the same object as x
```

False

[54] :
```
print(x is not y)

# Returns True because x is not the same object as y, even though they have the
↪same content
```

True

32 Membership Operators - in, not in

[55] :
```
x=[2,4,6,8]

print(x)
```

[2, 4, 6, 8]

[56] :
```
2 in x

#This will result in either True or False
```

[56] : True

[57] :
```
3 in x

#This will result in either True or False
```

[57] : False

[58] :
```
3 not in x

#This will result in either True or False
```

[58] : True

33 Bitwise Operators

[59] :
```
x=2

print(x)

y=4

print(y)
```

```
2
4
```

[60] :
```
# Bitwise and : Set each bit to 1 if both bits are 1

z=x&y

print(z)
```

```
0
```

[61] :
```
# Bitwise or : Set each bit to 1 if at least one bit is 1

z=x|y

print(z)
```

```
6
```

[62] :
```
# Bitwise xor : Set each bit to 1 if both bits are different

z=x^y

print(z)
```

```
6
```

[63] :
```
# Bitwise not : Reverse each bit of the number

# Numbers are represented in 2's complement form

z=~x

print(z)
```

```
-3
```

[64] :
```
# Zero fill left shift the bits, fill zeros in the end

z=x<<1
```

```
print(z)
```

4

[65] :
```
# Signed right shift

z=x>>1

print(z)
```

1

34 Conditional Statements

[66] :
```
x=2

if x>0:
    print("The number is positive")
elif x<0:
    print("The number is negative")
else:
    print("The number is 0")
```

The number is positive

35 Loops

35.1 For Loop

[67] :
```
x=[2,4,6,8]

for i in x:
    print(i)
```

2
4
6
8

[68] :
```
for i in range(0,len(x)): #This will loop from 0 to length(x)-1, not up to␣
↪length(x)
    print(x[i])
```

2
4
6
8

35.2 While Loop

[69] :
```
i=0

while(i<len(x)):
    print(x[i])
    i=i+1
```

```
2
4
6
8
```

35.3 Break, Continue, Pass

1. 'break' will end the loop before all iterations are done
2. 'continue' will make the loop go to next iteration
3. 'pass' is used when there is nothing to do

For loop cannot be empty. Functions cannot be empty. But, for some reason, if you want to keep them empty, you can use 'pass' command inside them to avoid error.

[70] :
```
x=[2,4,6,8]

for i in x:
    print(i)
else:
    print("Done printing")

#'else' is executed when the loop is finished

#'else' will not be executed if for loop is stopped by 'break' statement
```

```
2
4
6
8
Done printing
```

36 Data Types

Following are the built-in data types in Python

1. Numeric Types: int, float, complex
2. Text Type: str
3. Boolean Type: bool
4. Sequence Types: list, tuple, range

5. Mapping Type: dict
6. Set Types: set, frozenset
7. Binary Types: bytes, bytearray, memoryview
8. None Type: NoneType

We can use print() function to display the value of a variable.

We can use type() function to display the type of a variable.

We can use id() function to display the address of a variable.

37 Integer Data Type

[71] :
```
x=5

print(x)

type(x)
```

5

[71]: int

38 Float Data Type

[72]:
```
x=5.0

print(x)

type(x)
```

5.0

[72]: float

39 Complex Data Type

[73]:
```
# Complex Data Type

x=2+3j

print(x)

type(x)
```

(2+3j)

[73] : complex

[74] :
```
print(x.real)
```

2.0

[75] :
```
print(x.imag)
```

3.0

[76] :
```
print(x.conjugate())
```

(2-3j)

[77] :
```
print(abs(x))
```

3.6055512754639896

[78] :
```
print(angle(x))
```

NameError Traceback (most recent call last)

<ipython-input-78-d583ba68d5ae> in <module>
----> 1 print(angle(x))

NameError: name 'angle' is not defined

[79] :
```
import cmath

print(cmath.phase(x))
```

0.982793723247329

[80] :
```
import cmath

print(cmath.polar(x))#Gives magnitude and phase
```

(3.6055512754639896, 0.982793723247329)

[81] :
```
import cmath

print(cmath.rect(3.6055512754639896, 0.982793723247329))#Gives magnitude and
↪phase
```

(2.0000000000000004+3.0000000000000004j)

40 String Data Type

[82] :
```
x="Raveesh"

print(x)

type(x)
```

Raveesh

[82] :
```
str
```

[83] :
```
# We can concatenate two strings using +

x="Raveesh"

y=" Hegde"

print(x+y)

#String doesnt have append() method

#We have to use '+' operator to append a string with another string
```

Raveesh Hegde

[84] :
```
print(x[0])
```

R

[85] :
```
# Strings are immutable

# That means, we cannot overwrite a value

x[0]=A
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)

<ipython-input-85-ae43f5e15f4e> in <module>
      3 # That means, we cannot overwrite a value
      4
----> 5 x[0]=A


NameError: name 'A' is not defined
```

[86] :
```
#Strings are iterable

for i in x:
    print(i)
```

```
R
a
v
e
e
s
h
```

[87] :
```
#String slicing

print(x[2:]) #Index 2 to last index
```

```
veesh
```

[88] :
```
#Reversing a string

print(x[-1::-1]) #First '-1' indicates last element. Last '-1' indicates step␣
↪size
```

```
hseevaR
```

[89] :
```
#Appending a value to a string

x.append(c)

print(x)
```

```
---------------------------------------------------------------------------

AttributeError                            Traceback (most recent call last)

<ipython-input-89-aa52899887e1> in <module>
      1 #Appending a value to a string
      2
----> 3 x.append(c)
      4
      5 print(x)


AttributeError: 'str' object has no attribute 'append'
```

[90] :
```
x+'c'
```

[90] : 'Raveeshc'

[91] :
```
#To get ASCII value of a letter

asc=ord(' ')

print(asc)
```

32

[92] :
```
#To search for a letter/String

x="Raveesh Hegde CMRIT ECE"

x.find("CMR")#Returns the index of first occurence of string
             #Return -1 if value is not found
```

[92]: 14

[93] :
```
#To search for a letter/String

x="Raveesh Hegde CMRIT ECE"

x.index("CMR")#Returns the index of first occurence of string
              #Raise an exception if value is not found
```

[93]: 14

[94] :
```
#To search for a letter/String

x="Raveesh Hegde CMRIT CMRIT ECE"

x.rfind("CMR")#Returns the index of last occurence of string
              #Return -1 if value is not found
```

[94]: 20

[95] :
```
#To search for a letter/String

x="Raveesh Hegde CMRIT CMRIT ECE"

x.rindex("CMR")#Returns the index of last occurence of string
               #Raise an exception if value is not found
```

[95]: 20

[96] :
```
#To replace a Letter/String
```

```
x="Raveesh Hegde ECE"
y=x.replace("ECE","CMRIT")
print(y)
```

Raveesh Hegde CMRIT

[97] :
```
#To split a string
x="Raveesh Hegde ECE"
y=x.split() #Split the string
print(y)
```

['Raveesh', 'Hegde', 'ECE']

[98] :
```
#To split a string
x="Raveesh, Hegde ECE"
y=x.split(",") #Split the string
print(y)
```

['Raveesh', ' Hegde ECE']

[99] :
```
#To make the first letter capital
x="raveesh hegde ece"
y=x.capitalize()
print(y)
```

Raveesh hegde ece

[100] :
```
#To capitalize each word
x="raveesh hegde ece"
y=x.title()
print(y)
```

Raveesh Hegde Ece

[101] :
```
#To convert to upper case
x="raveesh hegde ece"
y=x.upper()
print(y)
```
RAVEESH HEGDE ECE

[102] :
```
#To convert to lower case
x="RAVEESH HEGDE ECE"
y=x.lower()
print(y)
```
raveesh hegde ece

[103] :
```
#To swap case
x="Raveesh Hegde ECE"
y=x.swapcase()
print(y)
```
rAVEESH hEGDE ece

[104] :
```
#To remove spaces at the beginning and at the end of the string
x="  Raveesh Hegde ECE "
y=x.strip()
print(y)
```
Raveesh Hegde ECE

[105] :
```
#To join the elements of an iterable with '#'
myTuple = ("John", "Peter", "Vicky")
x = "#".join(myTuple)
print(x)
type(x)
```

John#Peter#Vicky

[105]: str

[106] :
```
#To join the elements of an iterable with ' space '

myTuple = ("Rama", "Shyama", "Bhama")

x  = " ".join(myTuple)

print(x)
```

Rama Shyama Bhama

[107] :
```
#To check whether a string ends with a specific character

x="Raveesh Hegde ECE$"

y=x.endswith("$")

print(y)
```

True

Some more string methods

1. isalnum() Returns True if all characters in the string are alphanumeric
2. isalpha() Returns True if all characters in the string are in the alphabet
3. isdecimal() Returns True if all characters in the string are decimals
4. isdigit() Returns True if all characters in the string are digits
5. isidentifier() Returns True if the string is an identifier
6. islower() Returns True if all characters in the string are lower case
7. isnumeric() Returns True if all characters in the string are numeric
8. isprintable() Returns True if all characters in the string are printable
9. isspace() Returns True if all characters in the string are whitespaces
10. istitle() Returns True if the string follows the rules of a title
11. isupper() Returns True if all characters in the string are upper case

41 Boolean Data Type

[108] :
```
x=3>2

print(x)
```

```
type(x)
```

True

[108]: bool

[109] :
```
y=True

print(y)

type(y)
```

True

[109]: bool

[110] :
```
z=False

print(z)

type(z)
```

False

[110]: bool

42 Collection of Elements :

Python has 4 built-in data types for collection of elements - List, Tuple, Set, Dictionary

List is ordered, indexed, mutable, heterogeneous, allows duplicates

Tuple is ordered, indexed, immutable, heterogeneous, allows duplicates

Set is unordered, unindexed, immutable, heterogeneous, with NO duplicates

Dictionary is unordered, indexed, mutable, heterogeneous, allows duplicate values, but NOT duplicate keys

43 List Data Type

[111] :
```
# List is an ordered, mutable, hetergoeneous collection of items

# List allows duplicates

# Use square brackets to declare a list

x=[2, 3.0, "Raveesh", 2>3]

print(x)
```

```
type(x)
```

[2, 3.0, 'Raveesh', False]

[111] : list

[112] :
```
x[0] #First element
```

[112]: 2

[113] :
```
x[-1] #Last element
```

[113] : False

[114] :
```
# We can overwrite an element of the list

x[0]=4

print(x)
```

[4, 3.0, 'Raveesh', False]

[115] :
```
#To find number of elements in the list

len(x)
```

[115]: 4

[116] :
```
# List is iterable
print("The elements of the list are")
for i in x:
    print(i)
```

The elements of the list are
4
3.0
Raveesh
False

[117] :
```
# To append an element to the list

y=[1,3,5,7]

x.append(y) #The entire list 'y' is appended to 'x' as one element

print(x) #Note that original list 'x' itself is updated
```

[4, 3.0, 'Raveesh', False, [1, 3, 5, 7]]

[118] :
```
# To append elements of another list

z=[2,4,6,8]

x.extend(z) #The elements of z are appended to x as individual elements

print(x) #Note that original list 'x' itself is updated
```

```
[4, 3.0, 'Raveesh', False, [1, 3, 5, 7], 2, 4, 6, 8]
```

[119] :
```
# To append elements of one list to another list and create a new list

p=[2,4,6,8]

q=[1,3,5,7]

r=p+q

print(r)
```

```
[2, 4, 6, 8, 1, 3, 5, 7]
```

[120] :
```
#To create a reference to the list

a=x #Now 'a' is same as x
    #Whatever changes we make in a will reflect in x also

print("List a is")
print(a)

print("List x is ")
print(x)

#Overwrite an element of a
a[0]=100 #This will not only overwrite a[0], but also x[0]

print("New list a is")
print(a)

print("New list x is")
print(x)
```

```
List a is
[4, 3.0, 'Raveesh', False, [1, 3, 5, 7], 2, 4, 6, 8]
List x is
[4, 3.0, 'Raveesh', False, [1, 3, 5, 7], 2, 4, 6, 8]
New list a is
[100, 3.0, 'Raveesh', False, [1, 3, 5, 7], 2, 4, 6, 8]
New list x is
```

```
[100, 3.0, 'Raveesh', False, [1, 3, 5, 7], 2, 4, 6, 8]
```

[121] :
```
#To copy a list

b=x.copy()

print(b)
```

```
[100, 3.0, 'Raveesh', False, [1, 3, 5, 7], 2, 4, 6, 8]
```

[122] :
```
#To clear the elements of the list
b.clear()

print(b)
```

```
[]
```

[123] :
```
#To find the number of occurances of a specific element

a.count(100) #In list 'a', how many times '100' has appeared
```

[123]:
```
1
```

[124] :
```
#To find the index of a value
a.index(100) #Returns the index of the value '100'
             #If the value is repeated, only the first index will be ouput
             #Returns an error if the value is not found

a.find(100) #Returns the index of the value '100'
             #If the value is repeated, only the first index will be ouput
             #Returns -1 if the value is not found
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)

<ipython-input-124-b216379d57ef> in <module>
      4             #Returns an error if the value is not found
      5
----> 6 a.find(100) #Returns the index of the value '100'
      7             #If the value is repeated, only the first index will be
↪ouput
      8             #Returns -1 if the value is not found


AttributeError: 'list' object has no attribute 'find'
```

[125] :
```
#To insert a value at a specific position

x=[2,4,6,8]

print("The list x is",x)

x.insert(2,"New") #Insert at THIRD position

print("The new list is",x)
```

```
The list x is [2, 4, 6, 8]
The new list is [2, 4, 'New', 6, 8]
```

[126] :
```
#To remove an element from the list

print("The original list is",x)

x.remove("New") #Remove the element "New"

print("The new list is",x)
```

```
The original list is [2, 4, 'New', 6, 8]
The new list is [2, 4, 6, 8]
```

[127] :
```
#To remove an element from a specific position in the list

print("The original list is",x)

x.pop(3) #Remove the element in 4th position

print("The new list is",x)
```

```
The original list is [2, 4, 6, 8]
The new list is [2, 4, 6]
```

[128] :
```
#To reverse the list

print("The original list is",x)

x.reverse() #This will reverse the original list itself. It does NOT create a␣
↪new list.

print("The new list is",x)
```

```
The original list is [2, 4, 6]
The new list is [6, 4, 2]
```

[129] :
```
#To sort the list
```

```
print("The original list is",x)

x.sort() #This will sort the original list itself. It does NOT create a new list.

print("The new list is",x)

#If you want to create a NEW list which contains the sorted elements, then use␣
 ↪sorted() function

y=sorted(x) #This will leave the original list unchanged

print("The new list is",y)
```

```
The original list is [6, 4, 2]
The new list is [2, 4, 6]
The new list is [2, 4, 6]
```

[130] :
```
#Sorting the list in assending order

list1.sort()

print(list1)

#Note that orginal list itself is arranged in assending order

#A new list is not created
```

```
[2, 4, 6, 8]
```

[131] :
```
#Max and min elements

print(max(list1))

print(min(list1))
```

```
8
2
```

[132] :
```
#Sorting the list in descending order

list1.sort(reverse=True)

print(list1)

#Note that orginal list itself is arranged in assending order

#A new list is not created
```

```
[8, 6, 4, 2]
```

43.1 List Comprehension

List comprehension offers a shorter syntax when you want to create a new list based on the values of an existing list.

Example:

Based on a list of fruits, you want a new list, containing only the fruits with the letter “a” in the name.

Without list comprehension you will have to write a for statement with a conditional test inside:

[133] :
```
# Without List Comprehension

fruits = ["apple", "banana", "cherry", "kiwi", "mango"]
newlist = []

for x in fruits:
  if "a" in x:
    newlist.append(x)

print(newlist)
```

['apple', 'banana', 'mango']

[134] :
```
# With List Comprehension

fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

newlist = [x for x in fruits if "a" in x]

print(newlist)
```

['apple', 'banana', 'mango']

43.2 General Syntax for List Comprehension

newlist = [expression for item in iterable if condition == True]

Here, for, in, if, True are Keywords

44 Tuple Data Type

[135] :
```
# Tuple is an ordered, immutable, hetergoeneous collection of items

# Tuple allows duplicates

# Use round brackets to declare a tuple

x=(2, 3.0, "Raveesh", 2>3)
```

```
print(x)

type(x)
```

(2, 3.0, 'Raveesh', False)

[135] : tuple

[136] :
```
x[0] #First element
```

[136]: 2

[137] :
```
x[-1] #last element
```

[137] : False

[138] :
```
#length of 'x'
len(x)
```

[138]: 4

[139] :
```
#Tuple is immutable. So, we cannot overwrite a value

x[0]=4

print(x)
```

TypeError Traceback (most recent call last)

<ipython-input-139-1c4571ce6d83> in <module>
1 #Tuple is immutable. So, we cannot overwrite a value
2
----> 3 x[0]=4
4
5 print(x)

TypeError: 'tuple' object does not support item assignment

[140] :
```
#Tuple is immutable. So, we cannot append any value.
x.append(10)
```

```
AttributeError                            Traceback (most recent call last)

<ipython-input-140-99fae3912164> in <module>
      1 #Tuple is immutable. So, we cannot append any value.
----> 2 x.append(10)

AttributeError: 'tuple' object has no attribute 'append'
```

[141] :
```
#To count the frequency of a value

x=(2,4,4,6,8)

x.count(4) #To count how many times '4' has occured
```

[141]: 2

[142] :
```
#To find the index of a value
x.index(4) #Only the index of first occurance
```

[142]: 1

45 Range Data Type

[143] :
```
#From 0 to 4

x=range(5)

print(x)

type(x)
```

```
range(0, 5)
```

[143] : range

[144] :
```
len(x)
```

[144]: 5

46 Dictionary Data Type

[145] :
```
# Dictionary is an ordered, mutable, hetergoeneous collection of items having keys and values

# Dictionary does not allow duplicates

# Use curly brackets to declare a dictionary

x={"Name":"Virat", "Matches":350, "Centuries":50}

print(x)

type(x)
```

{'Name': 'Virat', 'Matches': 350, 'Centuries': 50}

[145] : dict

[146] :
```
x["Name"]
```

[146] : 'Virat'

[147] :
```
x["Matches"]
```

[147]: 350

[148] :
```
x["Centuries"]
```

[148]: 50

[149] :
```
#Length of the dictionary
len(x)
```

[149]: 3

[150] :
```
#To get all the keys of the dictionary
y=x.keys()
print(y)
```

dict_keys(['Name', 'Matches', 'Centuries'])

[151] :
```
#To get all the values of the dictionary
z=x.values()
print(z)
```

dict_values(['Virat', 350, 50])

[152] :
```
#To get key:value pair of the dictionary

print("Current dictionary is",x)

c=x.items()

print(c)
```

```
Current dictionary is {'Name': 'Virat', 'Matches': 350, 'Centuries': 50}
dict_items([('Name', 'Virat'), ('Matches', 350), ('Centuries', 50)])
```

[153] :
```
#To get value of a key
p=x.get('Name')

print(p)

print(x['Name'])
```

```
Virat
Virat
```

[154] :
```
#To get value of a key
p=x.get('Name1') #If the key is not present, return 'none'

print(p)

print(x['Name1'])#If the key is not present, throws an error
```

```
None
```

```
---------------------------------------------------------------------------

KeyError                                  Traceback (most recent call last)

<ipython-input-154-87cdffbf3b28> in <module>
  4 print(p)
  5
----> 6 print(x['Name1'])#If the key is not present, throws an error


KeyError: 'Name1'
```

[155] :
```
#We can overwrite the values of the dictionary

print("Current dictionary is",x)

x["Centuries"]=51
```

```
print("New dictionary is",x)
```

Current dictionary is {'Name': 'Virat', 'Matches': 350, 'Centuries': 50}
New dictionary is {'Name': 'Virat', 'Matches': 350, 'Centuries': 51}

[156] :
```
#To append a key:value pair to the dictionary

x.update({"Runs":10000})

print("Updated dictionary is",x)
```

Updated dictionary is {'Name': 'Virat', 'Matches': 350, 'Centuries': 51, 'Runs': 10000}

[157] :
```
#To remove the element with the specifid key

x.pop("Runs") #Remove the element with key "Runs"

print("Updated dictionary is",x)
```

Updated dictionary is {'Name': 'Virat', 'Matches': 350, 'Centuries': 51}

[158] :
```
#To remove the last item in the dictionary

x.popitem()

print("Updated dictionary is",x)
```

Updated dictionary is {'Name': 'Virat', 'Matches': 350}

[159] :
```
#To make a copy of the dictionary

a=x.copy()

print(a)
```

{'Name': 'Virat', 'Matches': 350}

[160] :
```
#To clear the contents of the dictionary

a.clear()

print("New dictionary is",a)
```

New dictionary is {}

[161] :
```
#To return a dictionary with the specified keys and value
p = ('key1', 'key2', 'key3')
```

```
q = 0

new_dict = dict.fromkeys(p,q)


print(new_dict)
```

```
{'key1': 0, 'key2': 0, 'key3': 0}
```

[162] :
```
print("Present dictionary is",x)

 c = x.setdefault("Runs", 10000) #If "Runs" key is available, do not do anything
                                 #If "Runs" key is not available, create it ans␣
 ↪set it to 10000

print("New dictionary is",x)

print("Value of c is ",c)
```

```
Present dictionary is {'Name': 'Virat', 'Matches': 350}
New dictionary is {'Name': 'Virat', 'Matches': 350, 'Runs': 10000}
Value of c is    10000
```

47 Set Data Type

[163] :
```
# Set is an unordered, mutable, hetergoeneous collection of items

# Set does not allow duplicates

# Repetitions will be automatically removed by a set

# Use curly brackets to declare a set

# Everytime you print a set, you may get different order

x={"Virat", "Virat", 350, 50}

print(x)

type(x)
```

```
{50, 'Virat', 350}
```

[163] :
```
set
```

[164] :
```
#Set is unordered. So, indexing does not work.
set[0]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)

<ipython-input-164-a1a636ee0185> in <module>
      1 #Set is unordered. So, indexing does not work.
----> 2 set[0]

TypeError: 'type' object is not subscriptable
```

[165] :
```
x.append(2)
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)

<ipython-input-165-ef5ccb6ef609> in <module>
----> 1 x.append(2)

AttributeError: 'set' object has no attribute 'append'
```

[166] :
```
x.add(2) #Instead of append, use 'add()' method

print(x)
```

```
{2, 50, 'Virat', 350}
```

[167] :
```
#Union of Sets using | operator

x={1,3,5,7}

y={2,4,6,8}

z=x|y

print(z)
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

[168] :
```
#Union of Sets using union() method
x={1,3,5,7}

y={2,4,6,8}
```

```
z=x.union(y)
print(z)
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

[169] :
```
#Difference between union using these two methods :
# '|' operator takes only 'set' objects
# union() method takes any iterable object such as list, string, dictionary
```

[170] :
```
#Intersection of sets using & operator
x={1,3,5,7}
y={2,4,6,8}
z=x&y
print(z)
```

```
set()
```

[171] :
```
#Intersection of Sets using intersection method
x={1,3,5,7}
y={2,4,6,8}
z=x.union(y)
print(z)
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

[172] :
```
#update() method does the union of two sets and updates the first set

x={1,3,5,7}
y={2,4,6,8}
x.update(y)
print(x)
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
```

[173]:
```
#intersection_update() method does the intersection of two sets and updates the first set


x={1,3,5,7}

y={2,4,6,8}

x.intersection_update(y)

print(x)
```

```
set()
```

[174]:
```
#Difference between two sets

x={1,3,5,7,2,4}

y={2,4,6,8}

z = x.difference(y) #This will give elements present in x, which are not present in y

print(z)
```

```
{1, 3, 5, 7}
```

[175]:
```
#Find the difference and update x

x={1,3,5,7,2,4}

y={2,4,6,8}

x.difference_update(y)

print(x)
```

```
{1, 3, 5, 7}
```

[176]:
```
#Symmetric difference : Elements that are not present in both sets

x={1,3,5,7,2,4}

y={2,4,6,8}

z=x.symmetric_difference(y)

print(z)
```

```
{1, 3, 5, 6, 7, 8}
```

[177]:
```
#Symmetric difference update
x={1,3,5,7,2,4}
y={2,4,6,8}
x.symmetric_difference_update(y)
print(x)
```

```
{1, 3, 5, 6, 7, 8}
```

[178]:
```
#To remove a random element from the list and give the removed element
#Original set will be updated
x={1,3,5,7,2,4}
y=x.pop()
print(y)
print(x)
```

```
1
{2, 3, 4, 5, 7}
```

[179]:
```
#To delete a specified element from the list
#Dont raise an error if the specified element doesnt exist
#Update the original set
x={1,3,5,7,2,4}
x.discard(10)
print(x)
```

```
{1, 2, 3, 4, 5, 7}
```

[180]:
```
#To delete a specified element from the list
#Raise an error if the specified element doesnt exist
#Update the original set
```

```
x={1,3,5,7,2,4}

x.remove(10)

print(x)
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)

<ipython-input-180-137cb818148b> in <module>
      7 x={1,3,5,7,2,4}
      8
----> 9 x.remove(10)
     10
     11 print(x)


KeyError: 10
```

[181] :
```
#Check whether sets are disjoint

x={1,3,5,7,2,4}

y={2,4,6,8}

z=x.isdisjoint(y)

print(z)
```

```
False
```

[182] :
```
#Check whether x is a subset of y

x={2,4}

y={2,4,6,8}

z1 =x.issubset(y)

z2=y.issubset(x)

print(z1)

print(z2)
```

```
True
```

False

[183]:
```
#Check whether y is a superset of x

x={2,4}

y={2,4,6,8}

z=y.issuperset(x)

print(z)
```

True

[184]:
```
#To copy a set

x={2,4,6,8}

y=x.copy()

print(y)
```

{8, 2, 4, 6}

[185]:
```
#To remove all elements of a set

x={2,4,6,8}

x.clear()

print(x)
```

set()

[186]:
```
#To remove duplicates from a list

list1=[2,2,4,4,4,6,6,7,7,8,8]

set1=set(list1) #This will convert list into set. Set will not allow duplicates

list2=list(set1) #This will convert set into list

print("List 1 is - ",list1)

print("List 2 is - ",list2)
```

List 1 is - [2, 2, 4, 4, 4, 6, 6, 7, 7, 8, 8]
List 2 is - [2, 4, 6, 7, 8]

[187] :
```
#To get the frequency of each element of the list

list1=[2,2,4,4,4,6,8,8,8,8]

set1=set(list1) #'set' will remove all duplicates

for i in set1:
    print("Count of {} is {}".format(i,list1.count(i)))
```

```
Count of 8 is 4
Count of 2 is 2
Count of 4 is 3
Count of 6 is 1
```

48 Frozenset Data Type

[188] :
```
#This is the immutable version of set data type

x = frozenset({"apple", "banana", "cherry"})

print(x)

type(x)
```

```
frozenset({'banana', 'cherry', 'apple'})
```

[188]:
```
frozenset
```

49 Bytes Data Type

[189] :
```
x = b"Hello"

print(x)

type(x)
```

```
b'Hello'
```

[189]:
```
bytes
```

50 Byte Array Data Type

[190] :
```
x = bytearray(5)

print(x)
```

```
type(x)
```

```
bytearray(b'\x00\x00\x00\x00\x00')
```

[190]: bytearray

51 Memoryview Data Type

[191] :
```
x = memoryview(bytes(5))

print(x)

type(x)
```

```
<memory at 0x019693D8>
```

[191]: memoryview

52 None Data Type

[192] :
```
x = None

print(x)

type(x)
```

```
None
```

[192]: NoneType

53 Type Conversion

[193] :
```
x=1
print(type(x))

y=1.9
print(type(y))

z=1+2j
print(type(z))
```

```
<class 'int'>
<class 'float'>
<class 'complex'>
```

[194] :
```
#We can convert int to float

x=1

a=float(x)

print(a)
```

1.0

[195] :
```
#We can convert float to int

b=2.9

y=int(b)

print(y)
```

2

[196] :
```
c=str(z)

print(c)
```

(1+2j)

[197] :
```
# We cant convert complex to any other data type

c=float(z)

print(c)
```

TypeError Traceback (most recent call last)

<ipython-input-197-359add14c132> in <module>
1 # We cant convert complex to any other data type
2
----> 3 c=float(z)
4
5 print(c)

TypeError: can't convert complex to float

[198] :
```
# We cant convert string to float

x="raveesh"

y=float(x)

print(y)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)

<ipython-input-198-d3a83e4767bb> in <module>
      3 x="raveesh"
      4
----> 5 y=float(x)
      6
      7 print(y)

ValueError: could not convert string to float: 'raveesh'
```

54 Operators

Following are the built-in operators in python

1. Arithmetic operators
2. Assignment operators
3. Comparison operators
4. Logical operators
5. Identity operators
6. Membership operators
7. Bitwise operators

55 Arithmetic Operators

[199] :
```
x=2

print(x)

y=3
```

```
print(y)
```

```
2
3
```

[200] :
```
# Addition

z=x+y

print(z)
```

```
5
```

[201] :
```
# Subtraction

z=x-y

print(z)
```

```
-1
```

[202] :
```
# Multiplication

z=x*y

print(z)
```

```
6
```

[203] :
```
# Division

z=x/y

print(z)
```

```
0.6666666666666666
```

[204] :
```
# Modulus
# This gives the reminder

z=x%y

print(z)
```

```
2
```

[205] :
```
# Floor division
# This gives the quotient

z=x//y
```

```
print(z)
```

0

[206] :
```
# Exponentiation

z=x**y

print(z)
```

8

56 Assignment Operators

[207] :
```
x=2

print(x)
```

2

[208] :
```
x+=3 # This means x=x+3

print(x)
```

5

[209] :
```
# Similarly -=, *=, /=, %=, //=, **= all are valid
```

57 Comparison Operators

[210] :
```
x=2

y=3
```

[211] :
```
# Check whether x is equal to y

x==y
```

[211] : False

[212] :
```
# Check whether x is not equal to y

x!=y
```

[212] : True

[213] :
```
# Check whether x is greater than y

x>y
```

[213] : False

[214] :
```
# Check whether x is less than y

x<y
```

[214] : True

[215] :
```
# Check whether x is greater than or equal to y

x>=y
```

[215] : False

[216] :
```
# Check whether x is less than or equal to y

x<=y
```

[216] : True

58 Logical Operators : and, or

[217] :
```
print(x)

print(y)
```

```
2
3
```

[218] :
```
# and operator : returns true if both conditions are met

x>0 and y<3
```

[218] : False

[219] :
```
# or operator : returns true if at least one condition is met

x>0 or y<3
```

[219] : True

[220] :
```
# not operator : reverse the result

not(x>0 or y<3)
```

[220] : False

59 Identity Operators - is, is not

[221] :
```
x = ["apples", "oranges"]

print(x)

y = ["apples", "oranges"]

print(y)

z = x

print(z)
```

```
['apples', 'oranges']
['apples', 'oranges']
['apples', 'oranges']
```

[222] :
```
print(x is z)

# Returns True because z is the same object as x
```

True

[223] :
```
print(x is y)

# Returns False because x is not the same object as y, even though they have the
↪same content
```

False

[224] :
```
print(x == y)

# Returns True because content of x is same as content of y
```

True

[225] :
```
print(x is not z)

# Returns False because z is the same object as x
```

False

[226] :
```
print(x is not y)

# Returns True because x is not the same object as y, even though they have the
↪same content
```

True

60 Membership Operators - in, not in

[227] :
```
x=[2,4,6,8]

print(x)
```

[2, 4, 6, 8]

[228] :
```
2 in x
```

[228] : True

[229] :
```
3 in x
```

[229] : False

[230] :
```
3 not in x
```

[230] : True

61 Bitwise Operators

[231] :
```
x=2

print(x)

y=4

print(y)
```

2
4

[232] :
```
# Bitwise and : Set each bit to 1 if both bits are 1

z=x&y

print(z)
```

0

[233] :
```
# Bitwise or : Set each bit to 1 if at least one bit is 1

z=x|y
```

```
print(z)
```

6

[234] :
```
# Bitwise xor : Set each bit to 1 if both bits are different

z=x^y

print(z)
```

6

[235] :
```
# Bitwise not : Reverse each bit of the number

# Numbers are represented in 2's complement form

z=~x

print(z)
```

-3

[236] :
```
# Zero fill left shift the bits, fill zeros in the end

z=x<<1

print(z)
```

4

[237] :
```
# Signed right shift

z=x>>1

print(z)
```

1

62 Functions

[238] :
```
# Defining a function

def my_function(x,y):
    z1=x+y
    z2=x-y
    return z1,z2

# Calling a function
```

```
a1,a2=my_function(2,3)

print("The sum and differences are")

print(a1,a2)
```

The sum and differences are
5 -1

63 Lambda Function

1. A lambda function is a small anonymous function. (It does not have any name).
2. A lambda function can take any number of arguments, but can only have one expression.
3. Lambda helps you use a function only once, and hence, avoids cluttering up the code with function definitions.
4. In short, Python's lambda keyword lets you define a function in a single line of code and use it immediately.
5. Syntax : lambda arguments : expression

[239] :
```
x = lambda a, b, c : a + b + c

print(x(2,4,6))
```

12

64 Classes and Objects

1. Class is a blueprint which defines some properties and behaviors.
2. An object is an instance of a class which has those properties and behaviours attached.
3. A class is not allocated memory when it is defined.
4. An object is allocated memory when it is created.
5. Class is a logical entity whereas objects are physical entities.
6. Classes provide a means of bundling data and functionality together.

[240] :
```
# Lets define a class by name 'Cricketer'

# It has properties like Name, Matches,Runs, Wickets, Average and functions like␣
↪Avg

# These properties are called Atributes and functions are called Methods

# We have to create a class using the keyword 'class'
```

```
class Cricketer:
    def _init_(self,arg1,arg2,arg3,arg4):
        self.Name=arg1
        self.Matches=arg2
        self.Runs=arg3
        self.Wickets=arg4
        # Self is a keyword here

    def Avg(self,arg3, arg2):
        self.Average=arg3/arg2

#Now, lets create an object 'Virat'

Virat=Cricketer("Virat Kohli", 200, 10000, 20)

print("The Name of the Cricketer is",Virat.Name)

Virat.Avg(10000,200)

print("{}'s Average Is {}".format(Virat.Name,Virat.Average))
```

```
The Name of the Cricketer is Virat Kohli
Virat Kohli's Average Is 50.0
```

[241] :
```
# Now, lets create an object 'Rohit'

Rohit=Cricketer("Rohit Sharma", 150,5000, 1)

print("The Name of the Cricketer is",Rohit.Name)

Rohit.Avg(5000,150)

print("{}'s Average Is {}".format(Rohit.Name,Rohit.Average))
```

```
The Name of the Cricketer is Rohit Sharma
Rohit Sharma's Average Is 33.333333333333336
```

65 Inheritance

1. Inheritance allows us to define a class that inherits all the methods and properties from another class.
2. Parent class is the class being inherited from, also called base class.
3. Child class is the class that inherits from another class, also called derived class.

[242] :
```
# We have already created a class 'Cricketer'
```

```
# Now, lets create a child class 'Player' which inherits all properties and␣
 ↪methods of parent class 'Cricketer'

class Player(Cricketer):
  pass
```

66 Polymorphism

1. Poly means many.
2. Polymorphism refers to methods/functions/operators with the same name that can be executed on many objects or classes.
3. For ex., len() function can be used with list, tuple, set, dictionary type of objects

[243] :
```
x=[2,4,6,8] # x is a list

print(len(x))

y=(1,3,5) # y is a tuple

print(len(y))
```

```
4
3
```

67 File Handling

Prerequisite : Make sure that the text file and the program file are in same folder.

The key function for working with files in Python is the open() function.

The open() function takes two parameters; filename and mode.

There are four different methods (modes) for opening a file:

“r” - Read - Default value. Opens a file for reading, error if the file does not exist

“a” - Append - Opens a file for appending, creates the file if it does not exist

“w” - Write - Opens a file for writing, creates the file if it does not exist

“x” - Create - Creates the specified file, returns an error if the file exists

In addition you can specify if the file should be handled as binary or text mode

“t” - Text - Default value. Text mode

“b” - Binary - Binary mode (e.g. images)

67.1 Reading an Existing Text File

[244] :
```
f = open("My Text File.txt")

f.read()

# Following is the content of the text file
```

[244] : 'This is the new sentence'

67.2 Closing the Text File

[245] :
```
# To close the file

f.close()

# In some cases, due to buffering, changes made to a file may not show until we close the file.

# So, we should close the files after all the operations are done.
```

67.3 Appending to the Contents of a Text File

[246] :
```
# To write to a file ( To append to the existing content)

f = open("My Text File.txt","a") # This opens the file for appending

# By default, file opens in read only mode.

# So, to append to the contents of a file, we have to specify 'a' in open() function

f.write("This is the new sentence")

f.close()

f = open("My Text File.txt")

f.read()
```

[246]: 'This is the new sentenceThis is the new sentence'

67.4 Overwriting the Contents of a Text File

[247]:
```
# To write to a file ( To overwrite the existing content)

f = open("My Text File.txt","w") # This opens the file for overwriting

# By default, file opens in read only mode.

# So, to overwrite the contents of a file, we have to specify 'w' in open()␣
↪function

f.write("This is the new sentence")

f.close()

f = open("My Text File.txt")

f.read()
```

[247] :
```
'This is the new sentence'
```

67.5 Creating a New Text File

[248] :
```
# To create a new text file

f1 = open("My New Text File.txt", "x")

# Returns an error if the file already exists
```

```
---------------------------------------------------------------------------

FileExistsError                           Traceback (most recent call last)

<ipython-input-248-897abeac6a66> in <module>
      1 # To create a new text file
      2
----> 3 f1 = open("My New Text File.txt", "x")
      4
      5 # Returns an error if the file already exists


FileExistsError: [Errno 17] File exists: 'My New Text File.txt'
```

67.6 Deleting a Text File

[249] :
```
import os
os.remove("To Delete.txt")
```

```
---------------------------------------------------------------------------
FileNotFoundError                         Traceback (most recent call last)

<ipython-input-249-d7e6c1af39b7> in <module>
      1 import os
----> 2 os.remove("To Delete.txt")

FileNotFoundError: [WinError 2] The system cannot find the file specified:␣
↪'To Delete.txt'
```

68 Exception Handling

[250] :
```
# Suppose a variable or a function is used in program, but not defined.

try:
  print(a) #Suppose 'a' is not defined
except:
  print("Variable 'a' not found. An exception occurred")
```

1.0

[251] :
```
try:
  print("Hello")
except:
  print("Something went wrong")
else: #This is executed if except condition is not executed
  print("Nothing went wrong")
```

Hello
Nothing went wrong

[252] :
```
try:
  print(a)
except:
  print("Variable a is not defined")
finally:#This is executed if except condition executed
  print("The 'try except' is finished")
```

1.0

The 'try except' is finished

[253] :
```
# Suppose a variable or a function is used in program, but not defined.

# Then, Name Error occurs.

try:
  print(a)
except NameError:
  print("Variable a is not defined")
except:
  print("Something else went wrong")
```

1.0

[254] :
```
# Name error also occurs when a method (function) is used before importing the module

x = 5.5

try :
    print(math.ceil(x))
except :
    print("Math module not imported")
```

Math module not imported

[255] :
```
x = -1

if x < 0:
  raise Exception("Sorry, no numbers below zero")
```

```
---------------------------------------------------------------------------
Exception                                 Traceback (most recent call last)

<ipython-input-255-2edc57024fbc> in <module>
  2
  3 if x < 0:
----> 4   raise Exception("Sorry, no numbers below zero")


Exception: Sorry, no numbers below zero
```

[256] :
```
x = "hello"

if not type(x) is int:
```

```
  raise TypeError("Only integers are allowed")
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)

<ipython-input-256-bc91768a6271> in <module>
      2
      3 if not type(x) is int:
----> 4     raise TypeError("Only integers are allowed")

TypeError: Only integers are allowed
```

69 Modules

1. Module is a library of functions
2. To create a module, save the functions in a file with extension .py

[257]:
```
# Module Creation Example

# Save the file as MyModule.py

def greeting(name):
  print("Hello, " + name)
```

[258]:
```
# To use the module in a program, import the module using 'import' keyword

import mymodule

mymodule.greeting("Jonathan")
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)

<ipython-input-258-16f087db6b2f> in <module>
      1 # To use the module in a program, import the module using 'import'␣
↪keyword
      2
----> 3 import mymodule
      4
      5 mymodule.greeting("Jonathan")
```

```
ModuleNotFoundError: No module named 'mymodule'
```

[259] :
```
# Use dir() function to list out all functions present in a module

y=dir(mymodule)

print(y)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)

<ipython-input-259-0a4b962921db> in <module>
      1 # Use dir() function to list out all functions present in a module
      2
----> 3 y=dir(mymodule)
      4
      5 print(y)

NameError: name 'mymodule' is not defined
```

70 Datetime Module

[260] :
```
# To Display Current Date and Time

import datetime

x = datetime.datetime.now() # Here, datetime is a function inside datetime module

print(x)
```

```
2023-11-08 13:22:46.552733
```

[261] :
```
# Use dir() function to list out all funtions present in a module

y=dir(datetime)

print(y)
```

```
['MAXYEAR', 'MINYEAR', '_builtins_', '_cached_', '_doc_', '_file_',
'_loader_', '_name_', '_package_', '_spec_', 'date', 'datetime',
'datetime_CAPI', 'sys', 'time', 'timedelta', 'timezone', 'tzinfo']
```

[262] :
```
import datetime

x = datetime.datetime.now()

print(x.year)
```

2023

[263] :
```
# Present Day in Full

print(x.strftime("%A"))
```

Wednesday

[264] :
```
# Present Day in Short

print(x.strftime("%a"))
```

Wed

[265] :
```
# Weekday number : 0:Sunday to 6:Saturday

print(x.strftime("%w"))
```

3

[266] :
```
# Day of the Month

print(x.strftime("%d"))
```

08

[267] :
```
# Month Name in Short

print(x.strftime("%b"))
```

Nov

[268] :
```
# Month Name in Full

print(x.strftime("%B"))
```

November

[269] :
```
# Month as a Number 1 to 12

print(x.strftime("%m"))
```

11

[270] :
```
# Year Short Version

print(x.strftime("%y"))
```

23

[271] :
```
# Year Full Version

print(x.strftime("%Y"))
```

2023

[272] :
```
# Hour 0 to 23

print(x.strftime("%H"))
```

13

[273] :
```
# Hour 0 to 12

print(x.strftime("%I"))
```

01

[274] :
```
# AM/PM

print(x.strftime("%p"))
```

PM

[275] :
```
# Minute 0 to 59

print(x.strftime("%M"))
```

22

[276] :
```
# Second 0 to 59

print(x.strftime("%S"))
```

48

[277] :
```
# Time Zone

print(x.strftime("%Z"))
```

[278] :
```
# Day Number of the Year 001 to 365

print(x.strftime("%j"))
```

312

[279]:
```
# Week Number of the Year 00 to 53

print(x.strftime("%U"))
```

45

[280]:
```
# Local version of date and time

print(x.strftime("%c"))
```

Wed Nov 8 13:22:48 2023

[281]:
```
# Century

print(x.strftime("%C"))
```

20

[282]:
```
# Local version of date

print(x.strftime("%x"))
```

11/08/23

[283]:
```
# Local version of time

print(x.strftime("%X"))
```

13:22:48

71 Built-In Math Functions

[284]:
```
x = min(5, 10, 25)

y = max(5, 10, 25)

print(x)

print(y)
```

5
25

[285]:
```
x = abs(-7.25)

print(x)
```

7.25

[286] :
```
x=3+4j

print(abs(x))
```

5.0

[287] :
```
x = pow(4, 3)

print(x)
```

64

72 Math Module

[288] :
```
# Square Root

import math

x = math.sqrt(64)

print(x)
```

8.0

[289] :
```
# Round-Off Functions

import math

x = math.ceil(1.4) # returns next nearest integer

y = math.floor(1.4) # returns previous nearest integer

z=math.trunc(1.9) # returns integer part

print(x) # returns 2

print(y)

print(z)
```

2
1
1

[290] :
```
# Math Constants

import math
```

```
x = math.pi

print(x)

y=math.e

print(y)

z=math.inf

print(z)

a=math.nan

print(a)

b=math.tau

print(b)
```

```
3.141592653589793
2.718281828459045
inf
nan
6.283185307179586
```

73 Math Functions

1. math.acos() Returns the arc cosine of a number
2. math.acosh() Returns the inverse hyperbolic cosine of a number
3. math.asin() Returns the arc sine of a number
4. math.asinh() Returns the inverse hyperbolic sine of a number
5. math.atan() Returns the arc tangent of a number in radians
6. math.atan2() Returns the arc tangent of y/x in radians
7. math.atanh() Returns the inverse hyperbolic tangent of a number
8. math.ceil() Rounds a number up to the nearest integer
9. math.comb() Returns the number of ways to choose k items from n items without repetition and order
10. math.copysign() Returns a float consisting of the value of the first parameter and the sign of the second parameter
11. math.cos() Returns the cosine of a number
12. math.cosh() Returns the hyperbolic cosine of a number

13. math.degrees() Converts an angle from radians to degrees
14. math.dist() Returns the Euclidean distance between two points (p and q), where p and q are the coordinates of that point
15. math.erf() Returns the error function of a number
16. math.erfc() Returns the complementary error function of a number
17. math.exp() Returns E raised to the power of x
18. math.expm1() Returns Ex - 1
19. math.fabs() Returns the absolute value of a number
20. math.factorial() Returns the factorial of a number
21. math.floor() Rounds a number down to the nearest integer
22. math.fmod() Returns the remainder of x/y
23. math.frexp() Returns the mantissa and the exponent, of a specified number
24. math.fsum() Returns the sum of all items in any iterable (tuples, arrays, lists, etc.)
25. math.gamma() Returns the gamma function at x
26. math.gcd() Returns the greatest common divisor of two integers
27. math.hypot() Returns the Euclidean norm
28. math.isclose() Checks whether two values are close to each other, or not
29. math.isfinite() Checks whether a number is finite or not
30. math.isinf() Checks whether a number is infinite or not
31. math.isnan() Checks whether a value is NaN (not a number) or not
32. math.isqrt() Rounds a square root number downwards to the nearest integer
33. math.ldexp() Returns the inverse of math.frexp() which is x * (2**i) of the given numbers x and i
34. math.lgamma() Returns the log gamma value of x
35. math.log() Returns the natural logarithm of a number, or the logarithm of number to base
36. math.log10() Returns the base-10 logarithm of x
37. math.log1p() Returns the natural logarithm of 1+x
38. math.log2() Returns the base-2 logarithm of x
39. math.perm() Returns the number of ways to choose k items from n items with order and without repetition
40. math.pow() Returns the value of x to the power of y
41. math.prod() Returns the product of all the elements in an iterable
42. math.radians() Converts a degree value into radians

43. math.remainder() Returns the closest value that can make numerator completely divisible by the denominator
44. math.sin() Returns the sine of a number
45. math.sinh() Returns the hyperbolic sine of a number
46. math.tan() Returns the tangent of a number
47. math.tanh() Returns the hyperbolic tangent of a number

74 Complex Math Module

[291] :
```
import cmath
y=dir(cmath)
print(y)
```

['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atanh', 'cos', 'cosh', 'e', 'exp', 'inf', 'infj', 'isclose', 'isfinite', 'isinf', 'isnan', 'log', 'log10', 'nan', 'nanj', 'phase', 'pi', 'polar', 'rect', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau']

74.1 CMath Functions

1. cmath.acos(x) Returns the arc cosine value of x
2. cmath.acosh(x) Returns the hyperbolic arc cosine of x
3. cmath.asin(x) Returns the arc sine of x
4. cmath.asinh(x) Returns the hyperbolic arc sine of x
5. cmath.atan(x) Returns the arc tangent value of x
6. cmath.atanh(x) Returns the hyperbolic arctangent value of x
7. cmath.cos(x) Returns the cosine of x
8. cmath.cosh(x) Returns the hyperbolic cosine of x
9. cmath.exp(x) Returns the value of Ex, where E is Euler's number (approximately 2.718281. . .), and x is the number passed to it
10. cmath.isclose() Checks whether two values are close, or not
11. cmath.isfinite(x) Checks whether x is a finite number
12. cmath.isinf(x) Check whether x is a positive or negative infinty
13. cmath.isnan(x) Checks whether x is NaN (not a number)
14. cmath.log(x[, base]) Returns the logarithm of x to the base
15. cmath.log10(x) Returns the base-10 logarithm of x
16. cmath.phase() Return the phase of a complex number

17. cmath.polar() Convert a complex number to polar coordinates
18. cmath.rect() Convert polar coordinates to rectangular form
19. cmath.sin(x) Returns the sine of x
20. cmath.sinh(x) Returns the hyperbolic sine of x
21. cmath.sqrt(x) Returns the square root of x
22. cmath.tan(x) Returns the tangent of x
23. cmath.tanh(x) Returns the hyperbolic tangent of x

75 Random Module

[292] :
```
import random

x=random.random() #Generate a random float between 0 and 1

print(x)
```

0.7300469140482612

[293] :
```
x=random.randint(10,20) #Generate a random integer between 10 and 20 including␣
↪10 and 20

print(x)
```

16

[294] :
```
l=[2,4,6,8]

x=random.choice(l) #Select an element from the list randomly

print(x)
```

2

[295] :
```
x=random.randrange(10,100,5) #Make a list from 10 to 100 in step of 5 and select␣
↪one value randomly

print(x)
```

10

[296] :
```
#Use dir() function to list all the methods in a module

import random

dir(random)
```

[296] : ['BPF',
 'LOG4',
 'NV_MAGICCONST',
 'RECIP_BPF',
 'Random',
 'SG_MAGICCONST',
 'SystemRandom',
 'TWOPI',
 '_BuiltinMethodType',
 '_MethodType',
 '_Sequence',
 '_Set',
 '__all__',
 '__builtins__',
 '__cached__',
 '__doc__',
 '__file__',
 '__loader__',
 '__name__',
 '__package__',
 '__spec__',
 '_acos',
 '_bisect',
 '_ceil',
 '_cos',
 '_e',
 '_exp',
 '_inst',
 '_itertools',
 '_log',
 '_os',
 '_pi',
 '_random',
 '_sha512',
 '_sin',
 '_sqrt',
 '_test',
 '_test_generator',
 '_urandom',
 '_warn',
 'betavariate',
 'choice',
 'choices',
 'expovariate',
 'gammavariate',
 'gauss',
 'getrandbits',

```
 'getstate',
 'lognormvariate',
 'normalvariate',
 'paretovariate',
 'randint',
 'random',
 'randrange',
 'sample',
 'seed',
 'setstate',
 'shuffle',
 'triangular',
 'uniform',
 'vonmisesvariate',
 'weibullvariate']
```

[297] :
```
help(random.uniform)
```

```
Help on method uniform in module random:

uniform(a, b) method of random.Random instance
    Get a random number in the range [a, b) or [a, b] depending on rounding.
```

76 Numpy Module

1. Numpy is a python library
2. Numpy is short form for Numerical Python
3. Numpy is used to work with arrays

[298] :
```
# To create a numpy 1D array

import numpy as np

arr = np.array([1, 2, 3, 4, 5]) # To np.array function, we can either pass a␣
↪list or tuple

print(arr)

print(type(arr))
```

```
[1 2 3 4 5]
<class 'numpy.ndarray'>
```

[299] :
```
# To create a numpy 2D array

import numpy as np
```

```
arr = np.array([[2,4,6,8],[1,3,5,7]]) # 2 Square Brackets Indicate 2D Array

print(arr)

print(type(arr)) # To Check the Data Type of the Array

print(np.ndim(arr)) # To check the dimension of the array

print(arr.shape) # To get number of rows and columns

print(arr[1,2]) # Print the element present in 2nd Row, 3rd Column
```

```
[[2 4 6 8]
 [1 3 5 7]]
<class 'numpy.ndarray'>
2
(2, 4)
5
```

[300] :
```
# Array Slicing

import numpy as np

arr = np.array([1, 2, 3, 4, 5]) # To np.array function, we can either pass a␣
↪list or tuple

print(arr[1:4]) #Include the first index, exclude the last index
```

```
[2 3 4]
```

[301] :
```
# To Get the Data Type of the Array

import numpy as np

arr = np.array([1, 2, 3, 4])

print(arr.dtype)
```

```
int32
```

[302] :
```
# To Convert 1D Array into 2D Array

import numpy as np

arr = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])

newarr = arr.reshape(4, 3)
```

```
print(newarr)
```

```
[[ 1  2  3]
 [ 4  5  6]
 [ 7  8  9]
 [10 11 12]]
```

[303] :
```
# To Convert 2D Array into 1D Array

newarr2=newarr.flatten() #Flatten Row Wise

print(newarr2)


newarr3=newarr.flatten('F') #Flatten Column Wise

print(newarr3)
```

```
[ 1  2  3  4  5  6  7  8  9 10 11 12]
[ 1  4  7 10  2  5  8 11  3  6  9 12]
```

[304] :
```
# To Print Array Elements

import numpy as np

arr = np.array([[2,4,6,8],[1,3,5,7]]) # 2 Square Brackets Indicate 2D Array

for x in arr:
    print(x)
```

```
[2 4 6 8]
[1 3 5 7]
```

[305] :
```
# To Print Array Elements

import numpy as np

arr = np.array([[2,4,6,8],[1,3,5,7]]) # 2 Square Brackets Indicate 2D Array

for x in arr:
    for y in x:
        print(y)
```

```
2
4
6
8
1
```

```
3
5
7
```

[306]:
```
# Array Concatenation

import numpy as np

arr1 = np.array([1, 2, 3])

arr2 = np.array([4, 5, 6])

arr = np.concatenate((arr1, arr2))

print(arr)
```

```
[1 2 3 4 5 6]
```

[307]:
```
# Sort the elements of an Array

import numpy as np

arr = np.array([3, 2, 0, 1])

print(np.sort(arr))
```

```
[0 1 2 3]
```

[308]:
```
# Filtering an array : Retain a Few, Reject a Few

import numpy as np

arr = np.array([41, 42, 43, 44])

x = arr[[True, False, True, False]]

print(x)
```

```
[41 43]
```

77 Matrix Operations

[309]:
```
import numpy as np

x=[2,4,6,8]

y=[1,3,5,7]
```

```
print(x)

print(y)
```

```
[2, 4, 6, 8]
[1, 3, 5, 7]
```

[310] :
```
# Matrix Concatenation

z=x+y

print(z)
```

```
[2, 4, 6, 8, 1, 3, 5, 7]
```

[311] :
```
# Matrix Addition

z=np.add(x,y)

print(z)
```

```
[ 3  7 11 15]
```

[312] :
```
# Matrix Subtraction

z=np.subtract(x,y)

print(z)
```

```
[1 1 1 1]
```

[313] :
```
# Element by Element Multiplication

z=np.multiply(x,y)

print(z)
```

```
[ 2 12 30 56]
```

[314] :
```
#Element by Element Division

z=np.divide(x,y)

print(z)
```

```
[2.         1.33333333 1.2        1.14285714]
```

[315] :
```
# Matrix Multiplication

z=np.matmul(x,y) # x into y transpose
```

```
print(z)
```

100

[316]:
```
# Transpose of a Matrix

x=[[2,4,6,8],[1,3,5,7]]

z=np.transpose(x)

print(z)
```

```
[[2  1]
 [4  3]
 [6  5]
 [8  7]]
```

[317]:
```
# Inverse of a Matrix

x=[[2,4],[1,3]] # x must be a square matrix to find inverse

print(x)

print(np.linalg.inv(x))
```

```
[[2, 4], [1, 3]]
[[ 1.5 -2. ]
 [-0.5  1. ]]
```

[318]:
```
# Determinant of a Matrix

print(np.linalg.det(x))
```

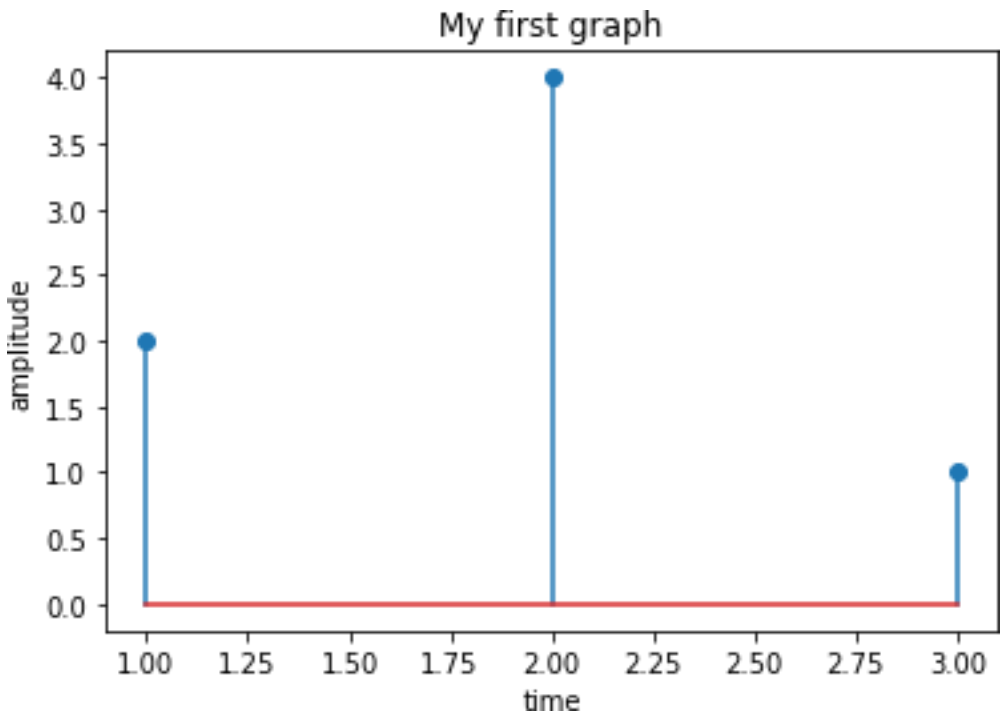
2.0

78 Graphs

[319]:
```
#First Import the necessary packages and modules

import matplotlib.pyplot as plt

# x axis values

x = [1,2,3]

# corresponding y axis values
```

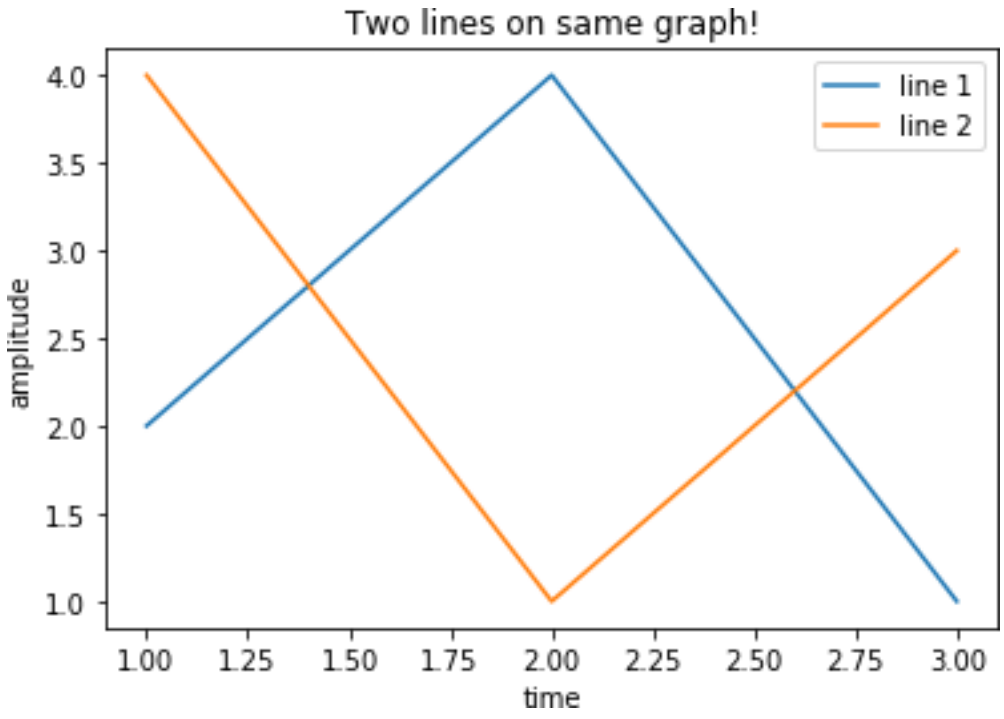
```
y = [2,4,1]

# plotting the points

plt.stem(x, y) # plt.stem is for discrete graphs, plt.plot for continuous graphs

# naming the x axis

plt.xlabel('time')

# naming the y axis

plt.ylabel('amplitude')

# giving a title to my graph

plt.title('My first graph')

# function to show the plot

plt.show()
```

```
C:\Users\cmrit\anaconda3\lib\site-packages\ipykernel_launcher.py:15:
UserWarning: In Matplotlib 3.3 individual lines on a stem plot will be added as
a LineCollection instead of individual lines. This significantly improves the
performance of a stem plot. To remove this warning and switch to the new
behaviour, set the "use_line_collection" keyword argument to True.
  from ipykernel import kernelapp as app
```



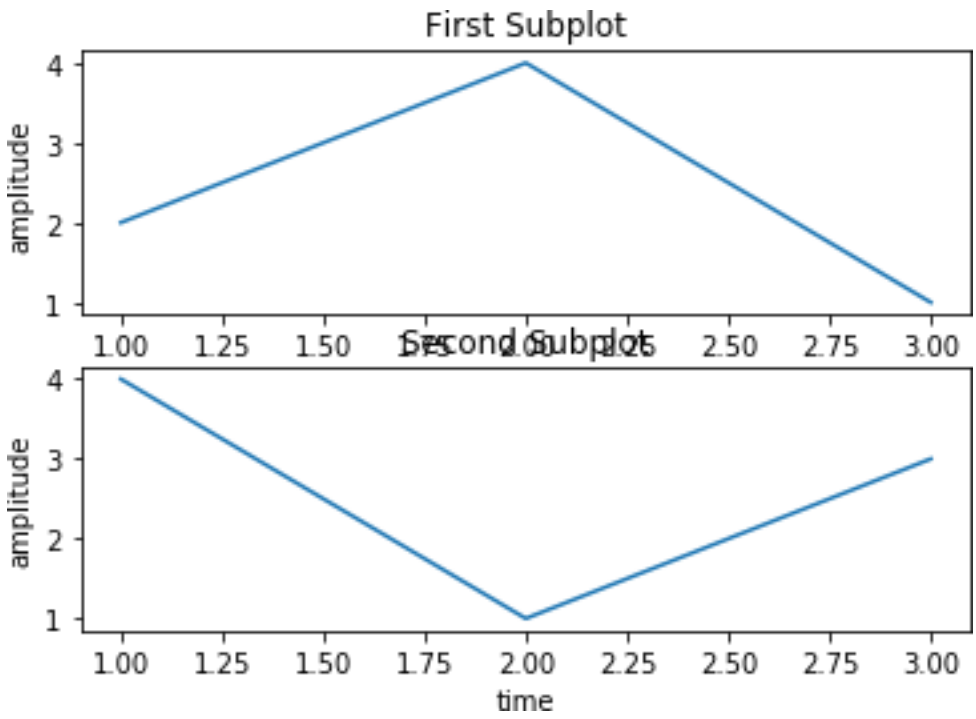
[320]:
```
#How to plot multiple graphs

import matplotlib.pyplot as plt

# line 1 points

x1 = [1,2,3]

y1 = [2,4,1]

# plotting the line 1 points

plt.plot(x1, y1, label = "line 1")

# line 2 points

x2 = [1,2,3]

y2 = [4,1,3]

# plotting the line 2 points

plt.plot(x2, y2, label = "line 2")
```

```
# naming the x axis
plt.xlabel('time')
# naming the y axis
plt.ylabel('amplitude')
# giving a title to my graph
plt.title('Two lines on same graph!')
# show a legend on the plot
plt.legend()
# function to show the plot
plt.show()
```



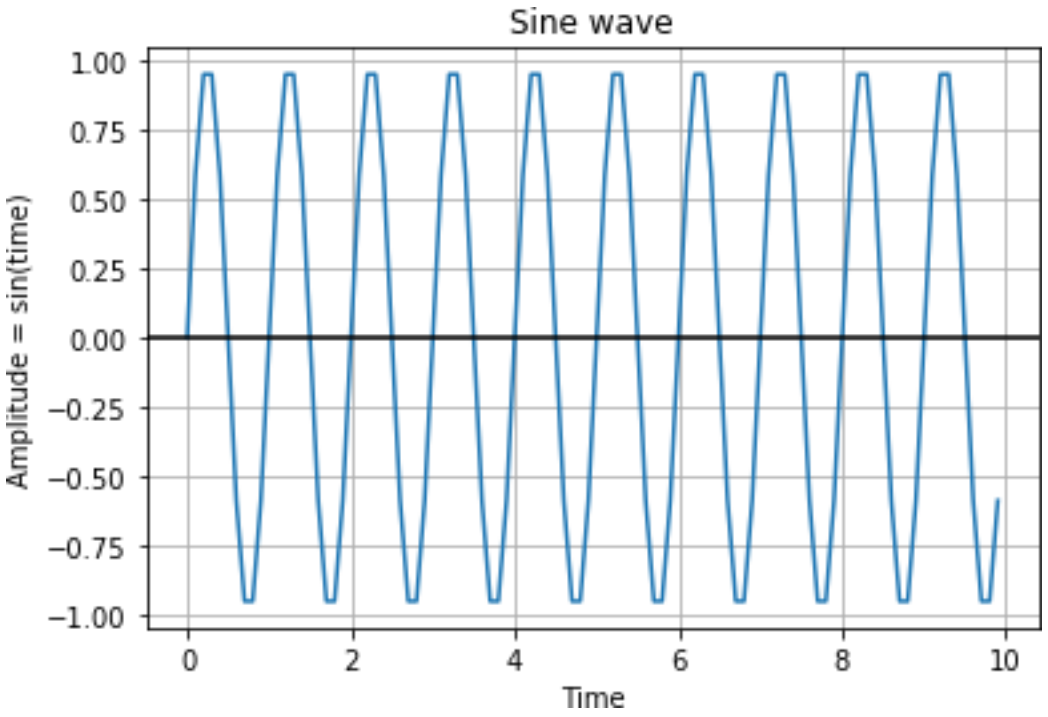
[321] :
```
#Subplot
```

```
import matplotlib.pyplot as plt

x1 = [1,2,3]

y1 = [2,4,1]

x2 = [1,2,3]

y2 = [4,1,3]

plt.subplot(2, 1, 1)

plt.plot(x1, y1, label = "line 1")

# naming the x axis

plt.xlabel('time')

# naming the y axis

plt.ylabel('amplitude')

# giving a title to my graph

plt.title('First Subplot')

plt.subplot(2, 1, 2)

plt.plot(x2, y2, label = "line 2")

# naming the x axis

plt.xlabel('time')

# naming the y axis

plt.ylabel('amplitude')

# giving a title to my graph

plt.title('Second Subplot')

plt.show()
```



[322]:
```
# Plotting sin wave

import numpy as np

import matplotlib.pyplot as plot

# Get x values of the sine wave

time=np.arange(0, 10, 0.1)

#Amplitude of the sine wave is sine of a variable like time

amplitude = np.sin(2*np.pi*time)

# Plot a sine wave using time and amplitude obtained for the sine wave

plot.plot(time, amplitude)

# Give a title for the sine wave plot

plot.title('Sine wave')

# Give x axis label for the sine wave plot
```

```
plot.xlabel('Time')

# Give y axis label for the sine wave plot

plot.ylabel('Amplitude = sin(time)')

plot.grid(True, which='both')

plot.axhline(y=0, color='k')

# Display the sine wave

plot.show()
```



1.Student Data

1 Write a Python program to read the student details like Name, USN, and Marks in three subjects. Display the student details, total marks and percentage with suitable messages.

[1]:
```
name = input("Enter your name: ") #By default, input() function takes string data type

usn = input("Enter your USN: ") #By default, input() function takes string data type

sub01 = int(input("Enter your marks out of 100 in Subject 1: ")) #If you want integer data, use int() function

sub02 = int(input("Enter your marks out of 100 in Subject 2:")) #If you want integer data, use int() function

sub03 = int(input("Enter your marks out of 100 in Subject 3: ")) #If you want integer data, use int() function
```

```
Enter your name: raveesh
Enter your USN: 1cr22ec000
Enter your marks out of 100 in Subject 1: 35
Enter your marks out of 100 in Subject 2:37
Enter your marks out of 100 in Subject 3: 38
```

[2]:
```
total = sub01+sub02+sub03

avg = (total)/3
```

[3]:
```
print("Name: ", name)

print("USN: ", usn)

print("Subject 01 Marks: ", sub01)
```

```
print("Subject 02 Marks: ", sub02)

print("Subject 03 Marks: ", sub03)

print("Total Marks: ", total)

print("Percentage = {:.3f}".format(avg)) #{:.3f} indicates that 3 decimal values
 ↪will be present in the output float
```

```
Name: raveesh
USN: 1cr22ec000
Subject 01 Marks: 35
Subject 02 Marks: 37
Subject 03 Marks: 38
Total Marks: 110
Percentage = 36.667
```

2.Celsius to Fahrenheit

1 Write a Python program to convert temperature in celsius to fahrenheit

[1]:
```
celsius=float(input("Enter  celsius  value  "))

#By default, input() function takes string data type.

#If you want to take float values from user, use float() function
```

```
Enter  celsius  value  37.6
```

[2]:
```
#Formula to convert celsius into fahrenheit

fahrenheit=celsius*1.8+32
```

[3]:
```
print("{:.2f} degree Celsius is equal to {:.4f} degree Fahrenheit".
↪format(celsius,fahrenheit))
```

```
37.60 degree Celsius is equal to 99.6800 degree Fahrenheit
```

3.Simple Interest

1 Write a Python function to calculate simple interest

[1]:
```
#Function definition

def simple_interest(p,t,r):
    si = (p * t * r)/100
    print('The Simple Interest is', si)
```

[2]:
```
#Main program

P = int(input("Enter the principal amount :"))

T = float(input("Enter the time duration in years:"))

R = float(input("Enter the rate of interest :"))
```

```
Enter the principal amount :10000
Enter the time duration in years:5
Enter the rate of interest :8.5
```

[3]:
```
#Calling the function

simple_interest(P,T,R)
```

```
The Simple Interest is 4250.0
```

4.Compound Interest

1 Write a Python function to calculate compound interest

[1]:
```
import math #math is a Python library for math operations
```

[2]:
```
#Function definition

def compound_interest(P,T,R):
    Final_Amount = P * (math.pow((1 + (R/100)),T))
    Compound_Interest = Final_Amount - P
    return Final_Amount,Compound_Interest
```

[3]:
```
#Main program

P=int(input("Enter Principal Amount: "))

T=float(input("Enter Time Duration in Years: "))

R=float(input("Enter Rate of Interest: "))
```

```
Enter Principal Amount: 100000
Enter Time Duration in Years: 5
Enter Rate of Interest: 8.5
```

[4]:
```
#Calling the function

fin_amount,comp_int=compound_interest(P,T,R)

print("Final Amount is {:.2f} and Compound Interest is {:.3f}".
 ↪format(fin_amount,comp_int))
```

```
Final Amount is 150365.67 and Compound Interest is 50365.669
```

5.Largest of 3 Numbers

1 Write a Python program to find the largest of 3 numbers

[1]:
```
n1=int(input("Enter the first number"))

n2=int(input("Enter the second number"))

n3=int(input("Enter the third number"))
```

```
Enter the first number10
Enter the second number10
Enter the third number20
```

[2]:
```
if n1>=n2 and n1>=n3:
    print("The largest number is ",n1)

elif n2>=n1 and n2>=n3:
    print("The largest number is ",n2)

else:
    print("The largest number is ",n3)
```

```
The largest number is  20
```

6.Best 2 Test Marks

1 Write a Python program to find the best 2 test marks out of 3 tests and find the average of them.

[1]:
```
test1marks=int(input("Enter test 1 marks"))

test2marks=int(input("Enter test 2 marks"))

test3marks=int(input("Enter test 3 marks"))
```

```
Enter test 1 marks30
Enter test 2 marks30
Enter test 3 marks35
```

[2]:
```
if test1marks>=test2marks and test1marks>=test3marks:
    if test2marks>=test3marks:
        total=test1marks+test2marks
    else:
        total=test1marks+test3marks

elif test2marks>=test1marks and test2marks>=test3marks:
    if test1marks>=test3marks:
        total=test2marks+test1marks
    else:
        total=test2marks+test3marks

else:
    if test1marks>=test2marks:
        total=test3marks+test1marks
    else:
        total=test3marks+test2marks
average=total/2
```

[4]:
```
print("Total marks of best 2 test marks is",total)

print("Average of best 2 test marks is",average)
```

```
Total marks of best 2 test marks is 65
Average of best 2 test marks is 32.5
```

7.Number of Likes, Dislikes

1 Two people made a list of movies they like or dislike.

Find the number of movies they both like or dislike.

Input is two strings containing 1s and 0s, 1 stands for like, 0 stands for dislike.

[1]:
```
#Function definition

def count_like_dislike (A, B):
    out=0
    l=min(len(A),len(B))
    for i in range(l):
        if A[i]==B[i]:
            out=out+1
    return  out
```

[2]:
```
A=input("Enter list1")

B=input("Enter  list2")

out= count_like_dislike(A, B)

print (out)
```

```
Enter list1101011
Enter  list21010
4
```

8.Even Odd

1 Write a Python program to count of number of even numbers and odd numbers from the given list.

Give distinct numbers as input values.

Sample Input :

5

1 2 3 4 5

Description:The first line of input consists of a single integer N denoting the total number of input

The second line of input consists of N space separated integers, denoting the individual elements in the input

Sample Output :

2 3

Output Description:Output a single line with the count of even numbers and odd numbers from N, separated by a single space.

[1]:
```
n=int(input()) #Enter the number of elements
list1=input() #Enter the elements separated by space
list1=list1.split() #After split(), we get a list
```

```
list1=[int(i) for i in list1] #Convert every element to integer
```

```
5
1 2 3 4 5
```

[2]:
```
even=0

odd=0

for i in list1:
    if i%2==0:
        even=even+1
    else:
        odd=odd+1
```

[3]:
```
print(even,odd)
```

```
2 3
```

2 Given a list of numbers, find the sum of odd and even numbers.

Skip the numbers divisible by 3.

Sample Input : 2 3 4 5 6 7

Sample Output : 6,12

[4]:
```
list1=input() #Enter the values separated by space
```

```
2 3 4 5 6 7
```

[5]:
```
list1=list1.split() #After split(), we get a list

list1=[int(i) for i in list1] #Convert every element to integer
```

[6]:
```
even_sum=0
odd_sum=0
for i in list1:
    if i%2==0 and i%3!=0:
        even_sum=even_sum+i
    elif i%2!=0 and i%3!=0:
        odd_sum=odd_sum+i
```

[7]:
```
print("{},{}".format(even_sum,odd_sum))
```

```
6,12
```

9.Reverse the Array and Find the Sum

1 Given an array A of N numbers (integers), you have to write a program which prints the sum of the elements of array A with the corresponding elements of the reverse of array A.
If array A has elements [1,2,3], then reverse of the array A will be [3,2,1] and the resultant array should be [4,4,4].

Input Format: The first line of the input contains a number N repre- senting the number of elements in array A.

The second line of the input contains N numbers separated by a space. (after the last elements, there is no space)

Output Format: Print the resultant array elements separated by a space. (no space after the last element)

Example: Input:

4

2 5 3 1

Output :

3 8 8 3

[1]:
```
len1=int(input()) #Enter the length of the list
```

```
4
```

[2]:
```
list1=input() #Enter the elements of the list separated by space
```

```
2 5 3 1
```

[3]:
```
list1=list1.split() #After split(), we get a list

list1=[int(i) for i in list1] #Convert every element of the list to integer
```

[4]:
```
list2=list1[::-1] #To reverse the list

list3=[] #Start with an empty list

for i in range(len(list1)):
    sum1=list1[i]+list2[i]
    list3.append(sum1)
```

[5]:
```
print(*list3) #To print the elements separated by space
```

```
3 8 8 3
```

10.Type of Triangle

1 Write a Python program to identify the type of triangle given its sides.

[1]:
```
x = int(input("enter side 1 : "))
y = int(input("enter side 2 : "))
z = int(input("enter side 3 : "))
```

```
enter side 1 : 2
enter side 2 : 2
enter side 3 : 3
```

[2]:
```
if x == y == z:
    print("Equilateral Triangle")
elif x==y or y==z or z==x:
    print("Isosceles  Triangle")
else:
    print("Scalene Triangle")
```

```
isosceles triangle
```

11.Parity Bit

1 Input a binary number.

If there are odd number of ones, then parity bit must be 1.

If there are even number of ones, then parity bit must be 0.

[1]:
```
n=input() #Enter a sequence of 0s and 1s
          #Note that 'n' is a string, NOT an integer
```

```
101011
```

[2]:
```
l=n.count("1") #Count the number of 1s
```

[3]:
```
if l%2==0:
    parity="0"
else:
    parity="1"
```

[4]:
```
n=n+parity #Here '+' does concatenation of strings, NOT arithmetic addition
```

[5]:
```
print(n)
```

```
1010110
```

12.Odd Factors of a Number

1 Write a Python program to print the odd factors of a number.

Input Format : 10

Output Format : 1 5

[1]:
```
num=int(input()) #Enter a number
```

```
10
```

[2]:
```
if num<1:
    print("Number must be greater than 0")
else:
    list1=[] #Start with an empty list
    for i in range(1,num+1):
        if num%i==0 and i%2!=0:
            list1.append(i)
```

[3]:
```
#See the output format given

#We have to print the values separated by space

print(*list1)
```

```
1 5
```

[4]:
```
#If we have to print the values separated by comma, but no comma after the last␣
↪element

print(*list1,sep=",")
```

```
1,5
```

[5]:
```
#If we have to print the values separated by comma and there should be a comma␣
↪even after the last element

for i in list1:
```

```
    print(i,end=",")
```

1,5,

[6]:
```
#If we have to print the values one below the other

print(*list1,sep="\n")
```

1
5

[7]:
```
#If we have to print the values one below the other and there should be a comma␣
↪after every element

for i in list1:
    print("{},".format(i))
```

1,
5,

[8]:
```
#If we have to print the values as a list

print(list1)
```

[1, 5]

2 Important Note :

There are multiple ways to input and output data in Python.

Which method should we follow?

In online coding tests, along with questions, sample test cases will also be given.

We have to follow the method which matches the format of sample input and output test cases.

13.Rule Based Product

1 Write a python program to find the product of a list of numbers based on the following rules.

If there is NO 5, display the product of all the numbers.

If one of the integers is 5, display the product of numbers to the right of 5 excluding 5 and any subsequent 5's

If there are NO numbers to the right of 5, display -1

Input :

5

1 5 6 7 8

The first line contains the length of the list

The second line contains the list of space-separated numbers.

Output :

336

[1]:
```
n = int(input()) #Enter the length of the list
```

```
5
```

[2]:
```
#There are several ways to input multiple elements of a list

#But the input format indicates that elements are separated by space
```

```
#So use the following method to take input

  list1 =input() #Enter multiple integers separated by space. At this point, list1␣
   ↪is NOT a list, but a string. Because, input() function takes string by default.

list1=list1.split() #After split(), we get a list. But the list elements are␣
 ↪still NOT integers, but strings.

list1=[int(i) for i in list1] #Convert every element of the list to integer␣
 ↪using int() function. Now we get a list of integers

#Refer to 'Program 0-All Formats' to learn other methods of inputting the␣
 ↪elements of a list
```

```
1 5 6 7 8
```

[3]:
```
product = 1 #Variable to store the rule-based product

if 5 not in list1:
    for i in list1:
        product=product*i
else:
    index1 = list1.index(5)
    if index1 == len(list1)-1:
        product = -1
    else:
        for i in list1[index1+1:]:
            if i !=5:
                product=product*i

print(product) # to print the product
```

```
336
```

14.Spy Number

1 Write a Python program to check whether a given number is Spy number or not.

A number is said to be a Spy number if the sum of all the digits is equal to the product of all digits.

For Ex. 1412 is a Spy number

Bacause,

Product of all digits = 1 * 4 * 1 * 2 = 8

Sum of all digits = 1+4+1+2 = 8

[1]:
```
n=int(input()) #Enter a number
```

```
1412
```

[2]:
```
#Note that integer cannot be indexed, but string can be indexed

#So, convert 'int' to 'str' and access the elements

n=str(n) #To convert 'int' to 'str'
```

[3]:
```
prod=1

sum1=0
```

[4]:
```
for i in n:
    prod=prod*int(i) #int() function converts 'str' to 'int'
    sum1=sum1+int(i) #int() function converts 'str' to 'int'
```

[5]:
```
if prod==sum1:
    print("Spy Number")
else:
    print("Not a Spy Number")
```

```
Spy Number
```

15.Neon Number

1 Write a Python Program to check whether the accepted number is a Neon number or not.

A number is said to be Neon number if the sum of the digits of the square of the number is equal to the number itself.

If the number is a Neon number print True, otherwise print False

Sample Input : 9
Sample Output : True

Explanation : 9^2=81
8+1=9

[1]:
```
num1=int(input()) #Enter a number
```

```
9
```

[2]:
```
num2=num1**2 #Square the number

num2=str(num2) #Integer cannot be indexed. So convert 'int' to 'str'. String can be indexed.
```

[3]:
```
sum1=0
for i in num2:
    sum1=sum1+int(i)
```

[4]:
```
if sum1==num1:
    print("True")
else:
    print("False")
```

```
True
```

16.Armstrong Number

1 Write a Python program to check whether a number is Armstrong number or not.

Armstrong Number means a number which is equal to sum of its digits raised to total number of digits

For Ex. : 153

There are 3 digits in 153.

1^3 + 5^3 + 3^3 = 153

So 153 is an Armstrong number

Sample Input : 153

Sample Output : 153 is an Armstrong number

[1]:
```
n=int(input()) #Enter a number
```

```
153
```

[2]:
```
# To find the total number of digits

#Convert 'int' to 'str' and then use len() function

count1=len(str(n))
```

[3]:
```
sum1=0

for i in str(n):
    sum1=sum1+int(i)**count1 #Here '**' is used to compute power
```

[4]:
```
if n==sum1:
    print("{} is an Armstrong number".format(n))
else:
    print("{} is not an Armstrong number".format(n))
```

```
153 is an Armstrong number
```

17.GCD

1 Write a Python program to find the GCD of the given two numbers

Sample Input : 16 20
Sample Output : 4

[1]:
```
#There are several ways to input multiple elements

#But the input format indicates that elements are separated by space

#So use the following method to take input

  list1 =input() #Enter multiple integers separated by space. At this point, list1␣
   ↪is NOT a list, but a string. Because, input() function takes string by default.

list1=list1.split() #After split(), we get a list. But the list elements are␣
 ↪still NOT integers, but strings.

list1=[int(i) for i in list1] #Convert every element of the list to integer␣
 ↪using int() function. Now we get a list of integers

#Refer to 'Program 0-All Formats' to learn other methods of inputting the␣
 ↪elements of a list

num1=list1[0] #First element of list1

num2=list1[1] #Second element of list2
```

```
16 20
```

[2]:
```
n=min(num1,num2) #Minimum of num1 and num2
```

[3]:
```
for i in range(n,0,-1):
    if num1%i==0 and num2%i==0:
        print(i)
        break
```

```
4
```

18. LCM

1 Write a Python program to find the LCM of two numbers.
Sample Input : 4,7
Sample Output : 28

[1]:
```
#There are several ways to input multiple elements

#But the input format indicates that elements are separated by comma

#So use the following method to take input

  list1 =input() #Enter multiple integers separated by space. At this point, list1␣
   ↪is NOT a list, but a string. Because, input() function takes string by default.

list1=list1.split(",") #After split(), we get a list. But the list elements are␣
 ↪still NOT integers, but strings.

list1=[int(i) for i in list1] #Convert every element of the list to integer␣
 ↪using int() function. Now we get a list of integers

num1=list1[0] #First element of list1

num2=list1[1] #Second element of list2
```

```
4,7
```

[2]:
```
n=max(num1,num2) #Maximum of 'num1' and 'num2'
```

[3]:
```
for i in range(n,num1*num2+1): #value of 'i' ranges from 'n' to 'num1*num2', NOT␣
 ↪upto "num1*num2+1"
    if i%num1==0 and i%num2==0: #if 'i' is divisible by both 'num1' and 'num2',␣
 ↪thats the LCM
        print(i)
        break
```

```
28
```

19.Perfect Square

1 Write a Python program to accept a number and display whether it is a perfect square or not.
Sample Input 1 : 25
Sample Output 1 : is a perfect square
Sample Input 2 : 7
Sample Output 2 : is not a perfect square

[1]:
```
num1=int(input()) #Enter a number
```

```
25
```

[2]:
```
num2=num1**0.5 #Square root of num1

num2=int(num2) #Convert num2 to int

num3=num2**2 #Square num2

if num3==num1:
    print("is a perfect square")
else:
    print("is not a perfect square")
```

```
is a perfect square
```

20.Sum of Squares of First n Natural Numbers

1 Write a Python program to find the sum of squares of first 'n' natural numbers.

Sample Input : 5

Sample Output : 55

[1]:
```
n=int(input()) #Enter the value of n
```

```
5
```

[2]:
```
sum1=0

for i in range(1,n+1): #Here, 'i' takes values from 1 to 'n', NOT up to 'n+1'
    sum1=sum1+i**2

print(sum1)
```

```
55
```

21.Factorial of a Number

1 Write a Python program to find the factorial of a given number.

Sample Input : 5

Sample Output : 120

[1]:
```
n=int(input()) #Enter a number
```

```
5
```

[2]:
```
fact=1

for i in range(n,0,-1): #Here, value of i ranges from n to 1, NOT up to 0.
    fact=fact*i
```

[3]:
```
print(fact)
```

```
120
```

2 Write a Python program to check whether a given number is Krishnamurthy number or not.

Print True if the number is Krishnamurthy number, else print False

A Krishnamurthy number is a number which is equal to the sum of the factorial of its digits

For example, 145 = 1! + 4! + 5!

[4]:
```
#Function to calculate factorial of a number

def factorial(n):
    fact=1
    for i in range(n,0,-1): #Here, value of i ranges from n to 1, NOT up to 0.
        fact=fact*i
    return fact
```

[8]:
```
#Function to check whether a number is Krishnamurthy number or not

def Krishnamurthy(n) :
    sum1=0
    n=str(n) #'int' cannot be indexed. So, convert 'int' to 'str'

    for i in n:
        sum1=sum1+factorial(int(i))
    if sum1==int(n):
        return True
    else:
        return False
```

[9]:
```
n=int(input()) #Enter a number
```

```
145
```

[10]:
```
if Krishnamurthy(n):
    print("True")
else:
    print("False")
```

```
True
```

3 Write a Python program to print Krishnamurthy numbers between 1 and 200001

[11]:
```
for i in range(1,200001):
    if (Krishnamurthy(i)):
        print(i)
```

```
1
2
145
40585
```

22.Palindrome

1 Develop a python program to check whether a given number is palindrome or not.

Print True if the number is Palindrome, otherwise print False

For Ex. 25352 is a palindrome

Because whether you read the digits from left to right or from right to left, you will get the same number

[1]:
```
num=int(input()) #Enter a number
```

```
25352
```

[2]:
```
num1=str(num) #'int' cannot be indexed. So, convert 'int' to 'str'

num2=num1[::-1] #This will read the string in reverse order

if num1==num2:
    print("True")
else:
    print("False")
```

```
True
```

23.Tens Place Ones Place

1 Write a Python Program to form an integer that has the Number of Digits at Ten's Place and the Least Significant Digit of the Entered Integer at One's Place.
If the input is of a single digit, print the result as 0.

Sample Input : 426
Sample Output : 36
Description : There are 3 digits in 426 and the least significant digit is 6. So output should be 36

[1]:
```
num=int(input()) #Enter a number
```

```
426
```

[2]:
```
num=str(num) #'int' cannot be indexed. So, convert 'int' to 'str'.

if len(num)==1: #Note that len() function works with 'str', NOT with 'int'
    result=0
else:
    lsd=num[-1] #num[-1] means last element of the string
                #Note that lsd is a string, NOT an integer

    count=len(num) #Note that count is an integer, NOT a string

    count=str(count) #Now count is a string

    result=count+lsd #Note that count and lsd are strings. So '+' does␣
↪concatenation, NOT arithmetic addition

    result=int(result) #Now result is an 'int'

print(result)
```

```
36
```

24.Leap Year

1 Given a year, determine whether it is a leap year or not.

If it is a leap year, return the Boolean True, otherwise return False.

A leap year is exactly divisible by 4 except for century years (years ending with 00).

The century year is a leap year only if it is perfectly divisible by 400.

[1]:
```
def isleap(x):
    if  x%4==0:
        if  x%400==0:
            return True
        elif x%100==0:
            return  False
        else:
            return  True
    else:
        return False
```

[2]:
```
year = int(input()) #Enter an year
```

200

[3]:
```
print(isleap(year))
```

False

2 Write a Python program to To Generate Leap Years in an Interval.

Sample Input :
200
300

Sample Output :
4 208 212 216 220 224 228 232 236 240 244 248 252 256 260 264 268 272 276 280 284 288 292 296

[4]:
```
start = int(input()) #Enter the start year

end = int(input()) #Enter the end year
```

```
200
300
```

[5]:
```
leap=[]#Start with an empty list

for i in range (start,end+1):
    if isleap(i):
        leap.append(i)

print(*leap) #To print the values separated by space
```

```
204 208 212 216 220 224 228 232 236 240 244 248 252 256 260 264 268 272 276 280
284 288 292 296
```

3 Write a Python program to generate the Next 15 Leap Years Starting From a Given Year.

Sample Input : 200

Sample Output : 204,208,212,216,220,224,228,232,236,240,244,248,252,256,260

[6]:
```
year=int(input()) #Enter the starting year
```

```
200
```

[7]:
```
leap=[] #Start with an empty list

count=0

while count<15:
```

```
    if isleap(year):
        leap.append(year)
        count=count+1
    year=year+1

print(*leap,sep=",") #To print the values separated by comma
```

204,208,212,216,220,224,228,232,236,240,244,248,252,256,260

25.Prime Number

1 Write a Python program to check whether a number is prime or not.

Print True if the number is prime. Otherwise, print False.

[1]:
```
def is_prime(num):
    if num<2:
        return False
    elif num==2:
        return True
    else:
        for i in range(2,num): #Here 'i' will vary from 2 to 'num-1', NOT up to 'num'
            if num%i==0:
                return False
        else:
            return True
```

[2]:
```
number=int(input()) #Enter a number
```

5

[3]:
```
print(is_prime(number))
```

True

2 Write a Python program to print all Prime numbers in an Interval.
Sample Input :

1

10

Sample Output:

2

3

5

7

[4]:
```
lower_limit=int(input()) #Enter the lower limit

upper_limit=int(input()) #Enter the upper limit
```

```
1
10
```

[5]:
```
prime=[] #Start with an empty list

for i in range(lower_limit,upper_limit+1): #Here,'i' will vary from␣
 ↪'lower_limit' to 'upper_limit', NOT up to 'upper_limit+1'
    if is_prime(i):
        prime.append(i)

print(*prime,sep="\n") #To print the elements of the list one below the other
```

```
2
3
5
7
```

3 Write a Python program to generate N prime numbers starting from the given number.

Sample Input :

10

5

Input Description : First number gives the starting point. Second number gives how many prime numbers to generate.

Sample Output :

[11, 13, 17, 19, 23]

[6]:
```
start=int(input()) #Enter the starting number

N=int(input()) #Enter how many prime numbers to generate
```

```
10
5
```

[7]:
```
count=0

prime=[] #Start with an empty list

while count<N:
    if is_prime(start):
        prime.append(start)
        count=count+1
    start=start+1
```

[8]:
```
print(prime)
```

```
[11, 13, 17, 19, 23]
```

26.Sum of Consecutive Primes

1 Some prime numbers are equal to the sum of other consecutive prime numbers.

For example 5 = 2 + 3, 17 = 2 + 3 + 5 + 7, 41 = 2 + 3 + 5 + 7 + 11 + 13.

Write a python program to find out the prime numbers that satisfy this property up to a given number.

Sample Input : 50

Sample Output :

5

17

41

[1]:
```
def is_prime(num):
    if num<2:
        return False
    elif num==2:
        return True
    else:
        for i in range(2,num): #Here 'i' will vary from 2 to 'num-1', NOT up to 'num'
            if num%i==0:
                return False
        else:
            return True
```

[2]:
```
num=int(input()) #Enter a number
```

```
50
```

[3]:
```
list1=[] #Start with an empty list

for i in range(2,num+1):
    if is_prime(i):
        list1.append(i)
print(list1)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]
```

[4]:
```
for i in range(1,len(list1)):
    sum=0
    for k in list1[0:i]:
        sum=sum+k
        if sum==list1[i]:
            print(list1[i])
            break
```

```
5
17
41
```

27.Prime Product

1 A positive integer m is a prime product if it can be written as p×q, where p and q are both primes.
Write a Python function primeproduct(m) that takes an integer m as input and returns True if m is a prime product and False otherwise.

[1]:
```
def is_prime(num):
    if num<2:
        return False
    elif num==2:
        return True
    else:
        for i in range(2,num): #Here 'i' will vary from 2 to 'num-1', NOT up to 'num'
            if num%i==0:
                return False
        else:
            return True
```

[2]:
```
num=int(input()) #Enter a number
```

```
10
```

[3]:
```
#To find all the prime factors of the number

list1=[] #Start with an empty list

for i in range(1,num+1):
    if num%i==0 and is_prime(i):
        list1.append(i)
```

[4]:
```
flag=0 #flag=0 indicates that the number cannot be expressed as a product of prime numbers

for i in range(0,len(list1)):
    for j in range(0,len(list1)):
        if list1[i]*list1[j]==num:
```

```
            flag=1 #flag=1 indicates that the number can be expressed as a␣
 ↪product of prime numbers
            break
if  flag==1:
    print(True)
else:
    print(False)
```

```
True
```

[5]:
```
#If you want to display the prime factors

flag=0 #flag=0 indicates that the number cannot be expressed as a product of␣
 ↪prime numbers

list2=[] #Start with an empty list

for i in range(0,len(list1)):
    for j in range(0,len(list1)):
        if list1[i]*list1[j]==num:
            flag=1 #flag=1 indicates that the number can be expressed as a␣
 ↪product of prime numbers
            if list1[i] not in list2:
                list2.append(list1[i])
            if list1[j] not in list2:
                list2.append(list1[j])
            break
print(list2)
```

```
[2, 5]
```

28.Product of Unique Prime Factors

1 Write a Python program to find the product of unique prime factors of a number.

Sample Input : 20

Sample Output :
2 5
10

Output Description:
The First line consists of the Prime Factors of 20 ie, 2 and 5 8.
The second line consists of the product of the prime factors : 2X5 =10

[1]:
```
n=int(input()) #Enter a number
```

```
Enter a number20
```

[2]:
```
def isprime(n):
    if n<2:
        return False
    elif n==2:
        return True
    else:
        for i in range(2,n):
            if n%i==0:
                return False
        return True
```

[3]:
```
l=[]
for i in range(2,n+1):
    if n%i==0 and isprime(i):
        l.append(i)

print(*l) #*l prints elements of the list 'l' seperated by space
```

```
2 5
```

[4]:
```
prod=1
for i in l:
    prod=prod*i

print(prod)
```

```
10
```

29.List of Unique Numbers in a List

1 Write a python program which takes an input of 'N' numbers and display the numbers which are unique(not repeated).

If no unique numbers are available in the list print "NULL".

Input Description:List of 'N' numbers and the first integer specifies the value of 'N', the size of the list, the rest of the numbers are elements of the list.

Output Description:A list of numbers which are unique.

Input :

6

10 12 45 12 10 65

Output : [45,65]

[1]:
```
n=int(input()) #Enter the length of the list
list1=input() #Enter the list of numbers separated by space
```

```
6
10 12 45 12 10 65
```

[2]:
```
list1=list1.split() #After splitting, we get a list with every element as 'str'
list1=[int(i) for i in list1] #Convert every element to 'int'
```

[3]:
```
list2=[] #start with an empty list

for i in list1:
    if list1.count(i)==1:
        list2.append(i)

if len(list2)==0:
    print("NULL")
else:
    print(list2)
```

```
[45, 65]
```

30.Fibonacci Series

1 Write a Python program to print the first n Fibonacci numbers.

Sample Input : 5

Sample Output :

0

1

1

2

3

[1]:
```
n=int(input()) #Enter how many Fibonacci numbers to be printed
```

```
5
```

[2]:
```
if n==1:
    fib=[0]
elif n==2:
    fib=[0,1]
elif n>2:
    fib=[0,1]
    for i in range (2,n):
        present=fib[i-1]+fib[i-2]
        fib.append(present)
```

[3]:
```
print(*fib,sep="\n") #Print values one below the other
```

```
0
1
```

1
2
3

2 Write a Python program to print Fibonacci numbers up to 'n'.

Sample Input : 5

Sample Output :

0

1

1

2

3

5

[4]:
```
n=int(input()) #Enter up to what number Fibonacci numbers to be printed
```

5

[5]:
```
if n==0:
    fib=[0]
elif n==1:
    fib=[0,1,1]
elif  n>=2:
    fib=[0,1,1]
    present=fib[-1]+fib[-2]
    while(present<=n):
        fib.append(present)
        present=fib[-1]+fib[-2]

print(*fib,sep="\n")
```

0
1
1
2

3
5

3 Write a Python program to Print the nth Fibonacci Number.

Sample Input :

5

Sample Output :

3

[6]:
```
n=int(input()) #Enter the value of 'n'
```

5

[7]:
```
fib=[] #Start with an empty list

if n==1:
    print(0)
elif n==2:
    print(1)
else:
    fib=[0,1]
    count=2
    while(count<n):
        present=fib[-1]+fib[-2]
        fib.append(present)
        count=count+1
    print(present)
```

3

4 Write a Python program to to check whether a number is Fibonacci or Not.

Print True if the number is Fibonacci, else print False

Sample Input : 8

Sample Output : True

[8]:
```
n=int(input()) #Enter a number
```

8

[9]:
```
if n<0:
    print(False)
elif n==0 or n==1:
    print(True)
else:
    fib=[0,1]
    present=fib[-1]+fib[-2]
    flag=0
    while(present<=n):
        if present==n:
            flag=1
            fib.append(present)
            break
        else:
            fib.append(present)
            present=fib[-1]+fib[-2]

    if flag:
        print(True)
    else:
        print(False)
```

True

31.ASCII Value

1 Write a Python program to display ASCII value of a character

[1]:
```
x=input("Enter a character : ")
```

```
Enter a character : a
```

[2]:
```
try:
    y=ord(x)
    print("ASCII Value of {} Is".format(x))
    print(y)
except: #If you give invalid input
    print("Invalid Input")
```

```
ASCII Value of a Is
97
```

[3]:
```
#To get character from ASCII value

chr(97)
```

[3] : 'a'

32.Caeser Cipher

1 Write a Python program to encode a message using Caesar cipher.

Sample Input : XYz

3

Sample Output : ABc

Note :ASCII Values

A to Z : 65 to 90

a to z : 97 to 122

[1]:
```
message = input() #Enter a string

shift = int(input()) #Enter the value of shift
```

```
XYz
3
```

[2]:
```
result="" #Start with an empty string
for i in message:
        new=ord(i)+shift
        if(i.isupper()):
            if new<=90:
                code=chr(new) #This is a character
            else:
                code=chr(65+new%91)
        elif(i.islower()):
            if new<=122:
                code=chr(new) #This is a character
```

```
            else:
                code=chr(97+new%123)

        result=result+code #Here, '+' does concatenation of strings, NOT addition of integers

print(result)
```

ABc

33.Run Length Encoding

1 Write a program to read the string of characters in which same character may be repeated consecutively like "XXXyZZZZZ".

Output the string in the form in which the count of the each character appears followed by character for example .

Input:

XXXyZZZZZ

Output:

3X1y5Z

[1]:
```
#Function definition

def encode(message):
    encoded_message ="" #Start with an empty string

    i = 0

    while (i <len(message)):
        count = 1
        ch = message[i]
        j = i
        while (j < len(message)-1):
            if (message[j] == message[j+1]):
                count = count+1
                j = j+1
            else:
                break
        encoded_message=encoded_message + str(count) + ch #Here, '+' does␣
↪concatenation of strings.
```

```
            i = j+1

    return encoded_message
```

[2]:
```
msg=input() #Enter a string

coded_msg=encode(msg)

print(coded_msg)
```

```
XXXyZZZZZ
3X1y5Z
```

34.Pythagorean Triplets

1 Write a Python program to generate Pythagorean triplets within a given limit

[1]:
```
limit=int(input()) #Enter upper limit
```

```
20
```

[2]:
```
c=0
m=2
while(c<limit):
    for n in range(1,m):
        a=m*m-n*n
        b=2*m*n
        c=m*m+n*n
        if(c>limit):
            break
        print(a,b,c)
    m=m+1
```

```
3 4 5
8 6 10
5 12 13
15 8 17
12 16 20
```

35.Senior Citizen

1 Write a Python program to read the name and year of birth of a person.

Display whether the person is a senior citizen or not.

[1]:
```
import datetime #'datetime' is a Python library

date = datetime.date.today()    # Get today's date.

# You can also use: date = datetime.datetime.now()

current_yr = date.year              # Extract year value from the date.

# You can also use: current_yr = date.strftime("%Y")
```

[2]:
```
yob = int(input("Enter the year of birth: "))

age =  current_yr - yob
```

```
Enter the year of birth: 1990
```

[3]:
```
if( age > 60 ):
      print("You are a Senior Citizen")
else:
      print("You are not a Senior Citizen")
```

```
You are not a Senior Citizen
```

36.Mean, Variance and Standard Deviation

1 Write a Python program to compute Mean, Variance and Standard Deviation

[1]:
```
import statistics #'statistics' is a Python library

n = int(input()) #Enter the number of values

list1 = [] #Start with an empty list

#Enter the values one below the other

for i in range (n):
        list1.append(int(input())) #Take int values and append to list1
```

```
Enter the number of values3
Enter the values
10
20
30
```

[2]:
```
mean=statistics.mean(list1)

variance=statistics.pvariance(list1)

std_dev=statistics.pstdev(list1)
```

[3]:
```
print("Mean is %.2f"%(mean))

print("Variance is %.2f "%(variance))

print("Standard deviation is %.2f "%(std_dev))
```

```
Mean is 20.00
Variance is 66.67
Standard deviation is 8.16
```

37.Largest Even Number

1 Write a Python program to input a string which is a mixture of letters and integers and special characters.

Find the largest even number from the available digit after removing the duplicates.

If an even number is not formed then return-1.

[1]:
```
str1=input() #Enter a string
```

cmrit@1234

[2]:
```
num="" #start with an empty string
for i in str1:
    if i.isdigit():
        num=num+i #Here '+' does concatenation of strings, NOT arithmetic addition
```

[3]:
```
if len(num)==0: #If there are no digits
    print(-1)

else:
    num=sorted(num,reverse=True)

    flag=0 #flag=0 indicates that even number is not possible

    for i in range(len(num)):
        if int(num[len(num)-1-i])%2==0:
            smallest_even = num[len(num)-1-i]
            num.remove(smallest_even)
            num.append(smallest_even)
            flag=1 #flag=1 indicates that even number is possible
            break

    if flag==1:
```

```
        ans = "".join(num)
        print(str(ans))
    else:
        print(-1)
```

4312

38.Binary to Decimal

1 Develop a Python program to convert binary number to decimal

[1]:
```
num=input() #Input a binary number
```

```
1011
```

[2]:
```
decimal=0

power=0

num=num[::-1] #Reverse the string

for i in num:
    decimal=decimal+int(i)*(2**power)
    power=power+1
```

[3]:
```
print(decimal)
```

```
11
```

39.Decimal to Binary

1 Write a Python program to convert decimal number to binary

[1]:
```
def dec_to_bin(num):
    i=0
    binary="" #Start with an empty string

    while(num>0):
        bit=num%2
        binary=str(bit)+binary #Here '+' does concatenation, NOT addidtion of integers
        num=num//2 #This will return the quotient
    return binary
```

[2]:
```
num=int(input()) #Input a decimal number
```

```
11
```

[3]:
```
print(dec_to_bin(num))
```

```
1011
```

40.Decimal to Hexadecimal

1 Write a Python program to convert decimal number to hexadecimal.

[1]:
```
#Program to convert decimal to hexadecimal
num=int(input())
print(hex(num))
```

```
23
0x17
```

[2]:
```
#Program to convert hexadecimal to decimal
num=input()
decnum=int(num,base=16)
print(decnum)
```

```
0X17
23
```

41.Roman to Decimal

1 Write a Python program to convert Roman number to integer

[1]:
```
def roman_to_int(roman):
    d={'I': 1,'V': 5,'X': 10,'L': 50,'C': 100,'D': 500,'M': 1000} #Form a␣
↪dictionary in which keys are Roman numbers and values are decimal numbers
    deci=0
    for i in range(len(roman)):
        prev=d[roman[i-1]]
        current=d[roman[i]]
        if current>prev and i>0:
            deci=deci+current-2*prev
        else:
            deci=deci+current
    return(deci)
```

[2]:
```
r=input()
dec=roman_to_int(r)
print(dec)
```

```
XXIV
24
```

42.Guess the Number

1 Write a Python program to simulate 'Guess the Number' Game

[1]:
```
import random #random() is a Python module

SecretNumber = random.randint(1, 20)
```

[2]:
```
print('I am thinking of a number between 1 and 20.')
```

I am thinking of a number between 1 and 20.

[3]:
```
# Ask the player to guess 6 times.

for i in range(6):
    print('Take a guess.')
    guess = int(input())

    if guess < SecretNumber:
        print('Your guess is low.')
    elif guess > SecretNumber:
        print('Your guess is high.')
    else:
        break  # This condition is when the guess is correct
```

Take a guess.
8
Your guess is high.
Take a guess.
3
Your guess is high.
Take a guess.
1
Your guess is low.
Take a guess.
2

[4] :
```
if guess == SecretNumber:
    print("Good job! You guessed my number in ",i,"attempts")
else:
```

```
    print('No. The number I was thinking of was ',secretNumber)
```

Good job! You guessed my number in 3 attempts

43.Number Pattern

1 Pattern 1

[4]:
```
rows = 6
# if you want user to enter a number, uncomment the below line
# rows = int(input('Enter the number of rows'))
# outer loop
for i in range(rows):
    # nested loop
    for j in range(i):
        # display number
        print(i, end=' ')
    # new line after each row
    print('')
```

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

2 Pattern 2

[5] :
```
rows = 5
b = 0
# reverse for loop from 5 to 0
for i in range(rows, 0, -1):
    b += 1
    for j in range(1, i + 1):
        print(b, end=' ')
    print('\r')
```

```
1 1 1 1 1
2 2 2 2
3 3 3
4 4
5
```

3 Pattern 3

[6] :
```
rows = 5
num = rows
# reverse for loop
for i in range(rows, 0, -1):
    for j in range(0, i):
        print(num, end=' ')
    print("\r")
```

```
5 5 5 5 5
5 5 5 5
5 5 5
5 5
5
```

4 Pattern 4

[7] :
```
rows = 5
for i in range(rows, 0, -1):
    for j in range(0, i + 1):
        print(j, end=' ')
    print("\r")
```

```
0 1 2 3 4 5
0 1 2 3 4
0 1 2 3
0 1 2
0 1
```

5 Pattern 5

[8] :
```
rows = 5
i = 1
while i <= rows:
    j = 1
    while j <= i:
        print((i * 2 - 1), end=" ")
        j = j + 1
    i = i + 1
    print('')
```

```
1
3 3
5 5 5
7 7 7 7
9 9 9 9 9
```

6 Pattern 6

[9] :
```
rows = 5
# reverse loop
for i in range(rows, 0, -1):
    num = i
    for j in range(0, i):
        print(num, end=' ')
    print("\r")
```

```
5 5 5 5 5
4 4 4 4
3 3 3
2 2
1
```

7 Pattern 7

[10] :
```
rows = 6
for i in range(1, rows):
    for j in range(i, 0, -1):
        print(j, end=' ')
    print("")
```

```
1
2 1
3 2 1
4 3 2 1
5 4 3 2 1
```

8 Pattern 8

[13]:
```
rows = 5
for i in range(0, rows + 1):
    for j in range(rows - i, 0, -1):
        print(j, end=' ')
    print()
```

```
5 4 3 2 1
4 3 2 1
3 2 1
2 1
1
```

9 Pattern 9

[14]:
```
rows = 5
for i in range(1, rows + 1):
    for j in range(1, rows + 1):
        if j <= i:
            print(i, end=' ')
        else:
            print(j, end=' ')
    print()
```

```
1 2 3 4 5
2 2 3 4 5
3 3 3 4 5
4 4 4 4 5
5 5 5 5 5
```

10 Pattern 10

[15]:
```
rows = 8
# rows = int(input("Enter the number of rows "))
for i in range(1, rows + 1):
    for j in range(1, i + 1):
        # multiplication current column and row
        square = i * j
        print(i * j, end='')
    print()
```

```
1
2  4
3  6  9
4  8  12  16
5  10  15  20  25
6  12  18  24  30  36
7  14  21  28  35  42  49
8  16  24  32  40  48  56  64
```

44.Star Pattern

1 Pattern 1

[1]:
```
# number of rows
rows = 5
for i in range(0, rows):
    # nested loop for each column
    for j in range(0, i + 1):
        # print star
        print("*", end=' ')
    # new line after each row
    print("\r")
```

```
*
* *
* * *
* * * *
* * * * *
```

[4]:
```
rows = 5
for j in range(1, rows+1):
    print("* " * j)
```

```
*
* *
* * *
* * * *
* * * * *
```

2 Pattern 2

[3]:
```
# number of rows
rows = 5
k = 2 * rows - 2
for i in range(0, rows):
    # process each column
    for j in range(0, k):
        # print space in pyramid
        print(end=" ")
```

```
    k = k - 2
    for j in range(0, i + 1):
        # display star
        print("* ", end="")
    print("")
```

```
        *
      * *
    * * *
  * * * *
* * * * *
```

3 Pattern 3

[5]:
```
rows = 5
for i in range(rows + 1, 0, -1):
    # nested reverse loop
    for j in range(0, i - 1):
        # display star
        print("*", end=' ')
    print(" ")
```

```
* * * * *
* * * *
* * *
* *
*
```

4 Pattern 4

[6]:
```
rows = 5
k = 2 * rows - 2
for i in range(rows, -1, -1):
    for j in range(k, 0, -1):
        print(end=" ")
    k = k + 1
    for j in range(0, i + 1):
        print("*", end=" ")
    print("")
```

```
      * * * * * *
       * * * * *
        * * * *
         * * *
          * *
           *
```

5 Pattern 5

[7]:
```
rows = 5
i = rows
while i >= 1:
    j = rows
    while j > i:
        # display space
        print(' ', end=' ')
        j -= 1
    k = 1
    while k <= i:
        print('*', end=' ')
        k += 1
    print()
    i -= 1
```

```
* * * * *
  * * * *
    * * *
      * *
        *
```

6 Pattern 6

[8]:
```
print("Print equilateral triangle Pyramid using asterisk symbol ")
# printing full Triangle pyramid using stars
size = 7
m = (2 * size) - 2
for i in range(0, size):
    for j in range(0, m):
        print(end=" ")
    # decrementing m after each loop
    m = m - 1
    for j in range(0, i + 1):
        print("* ", end=' ')
    print(" ")
```

```
Print equilateral triangle Pyramid using asterisk symbol
            *
          *   *
        *   *   *
      *   *   *   *
    *   *   *   *   *
  *   *   *   *   *   *
*   *   *   *   *   *   *
```

7 Pattern 7

[9]:
```
rows = 6
for i in range(0, rows):
    for j in range(0, i + 1):
        print("*", end=' ')
    print(" ")

print(" ")

for i in range(rows + 1, 0, -1):
    for j in range(0, i - 1):
        print("*", end=' ')
    print(" ")
```

```
*
* *
* * *
* * * *
* * * * *
* * * * * *

* * * * * *
* * * * *
* * * *
* * *
* *
*
```

8 Pattern 8

[10]:
```
rows = 5
for i in range(0, rows):
    for j in range(0, i + 1):
        print("*", end=' ')
    print("\r")

for i in range(rows, 0, -1):
    for j in range(0, i - 1):
        print("*", end=' ')
    print("\r")
```

```
*
* *
* * *
* * * *
* * * * *
```

```
* * * *
* * *
* *
*
```

9 Pattern 9

[12]:
```
rows = 5
i = 1
while i <= rows:
    j = i
    while j < rows:
        # display space
        print(' ', end=' ')
        j += 1
    k = 1
    while k <= i:
        print('*', end=' ')
        k += 1
    print()
    i += 1

i = rows
while i >= 1:
    j = i
    while j <= rows:
        print(' ', end=' ')
        j += 1
    k = 1
    while k < i:
        print('*', end=' ')
        k += 1
    print('')
    i -= 1
```

```
        *
      * *
    * * *
  * * * *
* * * * *
  * * * *
    * * *
      * *
        *
```

10 Pattern 10

[13]:
```
rows = 5
i = 0
while i <= rows - 1:
    j = 0
    while j < i:
        # display space
        print('', end=' ')
        j += 1
    k = i
    while k <= rows - 1:
        print('*', end=' ')
        k += 1
    print()
    i += 1

i = rows - 1
while i >= 0:
    j = 0
    while j < i:
        print('', end=' ')
        j += 1
    k = i
    while k <= rows - 1:
        print('*', end=' ')
        k += 1
    print('')
    i -= 1
```

```
* * * * *
 * * * *
  * * *
   * *
    *
    *
   * *
  * * *
 * * * *
* * * * *
```

11 Pattern 11

[14]:
```
rows = 5
k = 2 * rows - 2
for i in range(0, rows):
    for j in range(0, k):
```

```
        print(end=" ")
    k = k - 1
    for j in range(0, i + 1):
        print("* ", end="")
    print("")

k = rows - 2

for i in range(rows, -1, -1):
    for j in range(k, 0, -1):
        print(end=" ")
    k = k + 1
    for j in range(0, i + 1):
        print("* ", end="")
    print("")
```

```
        *
       * *
      * * *
     * * * *
    * * * * *
   * * * * * *
    * * * * *
     * * * *
      * * *
       * *
        *
```

12 Pattern 12

[15]:
```
rows = 5
i = 1
while i <= rows:
    j = rows
    while j > i:
        # display space
        print(' ', end=' ')
        j -= 1
    print('*', end=' ')
    k = 1
    while k < 2 * (i - 1):
        print(' ', end=' ')
        k += 1
    if i == 1:
        print()
    else:
        print('*')
```

```
    i += 1

i = rows - 1
while i >= 1:
    j  =  rows
    while j > i:
        print(' ', end=' ')
        j -= 1
    print('*', end=' ')
    k = 1
    while k <= 2 * (i - 1):
        print(' ', end=' ')
        k += 1
    if i == 1:
        print()
    else:
        print('*')
    i -= 1
```

```
          *
        *   *
      *       *
    *           *
  *               *
*                   *
  *               *
    *           *
      *       *
        *   *
          *
```

[ ]:

45.Bank Transaction

1 Write a python program that computes the net amount of a bank account based on a transaction log from console input. The transaction log format is shown below.

Input Format :

4

D 300

D 300

W 200

D 100

Output Format :

B 500

[1]:
```
n=int(input())
x=[]
for i in range(n):
    x.append(input())
```

```
3
D 100
D 200
W 100
```

[2]:
```
balance=0

for i in x:
    i=i.split()
    if i[0]=="D" or i[0]=="d":
        balance=balance+int(i[1])
    if i[0]=="W" or i[0]=="w":
        balance=balance-int(i[1])
```

[3]:
```
print("B",balance)
```

```
B 200
```

46.Permutations and Combinations

1 Write a Python program to find the Permutations and Combinations of the given list

[1]:
```
import itertools #itertools is a Python module
```

[7]:
```
val = [1, 2]
```

[8]:
```
#Permutation without replacement
perm_set = itertools.permutations(val)
print(perm_set)
```

```
<itertools.permutations object at 0x0452C540>
```

[9]:
```
for i in perm_set:
    print(i)
```

```
(1, 2)
(2, 1)
```

[12]:
```
#Permutations of a string
str1 = "AB"

perm_set = itertools.permutations(str1)
for i in perm_set:
    print(i)
```

```
('A', 'B')
('B', 'A')
```

[13]:
```
#Permutation of required length
val = [1, 2, 3]

perm_set = itertools.permutations(val,2)

for i in perm_set:
    print(i)
```

```
(1, 2)
(1, 3)
```

```
(2, 1)
(2, 3)
(3, 1)
(3, 2)
```

[14]:
```
#Combination without replacement
com_set = itertools.combinations(val, 2)

for i in com_set:
    print(i)
```

```
(1, 2)
(1, 3)
(2, 3)
```

[15]:
```
#Combination with replacement
com_set = itertools.combinations_with_replacement(val, 2)

for i in com_set:
    print(i)
```

```
(1, 1)
(1, 2)
(1, 3)
(2, 2)
(2, 3)
(3, 3)
```

2 To calculate Binomial Coefficients

2.1 npr=n!/(n-r)!

2.2 ncr=npr/r!

[20]:
```
import math

n=int(input("Enter the value of n"))

r=int(input("Enter the value of r"))
```

```
Enter the value of n5
Enter the value of r3
```

[21]:
```
npr=math.factorial(n)/math.factorial(n-r)
```

[22]:
```
npr
```

[22]: 60.0

[23]:
```
ncr=math.factorial(n)/math.factorial(n-r)/math.factorial(r)
```

[24]:
```
ncr
```

[24]: 10.0

47.Transpose of a Matrix

1 Write a Python program to find the transpose of a matrix

[11] :
```
X = [[1, 3], [4,5], [7, 9],[6,7]]

result = [[X[j][i] for j in range (len (X))] for i in range (len (X[0]))]

for r in result:
        print (r)
```

```
[1, 4, 7, 6]
[3, 5, 9, 7]
```

[13]:
```
import numpy #numpy is a Python library used to work with arrays

Y=numpy.transpose(X)

Y
```

[13]:
```
array([[1, 4, 7, 6],
       [3, 5, 9, 7]])
```

[14]:
```
type(Y)
```

[14]:
```
numpy.ndarray
```

48.Shuffle Two Lists

1 Write a function shuffle(l1,l2) that takes as input two lists, 11 and l2, and returns a list consisting of the first element in l1, then the first element in l2, then the second element in l1, then the second element in l2, and so on.

If the two lists are not of equal length, the remaining elements of the longer list are appended at the end of the shuffled output.

[1]:
```
list1=input() #Enter list1 elements separated by space

list1=[int(i) for i in list1.split()] #Split the list and convert every element
↪to integer
```

```
2 4 6 8
```

[2]:
```
list2=input() #Enter list2 elements separated by space

list2=[int(i) for i in list2.split()] #Split the list and convert every element
↪to integer
```

```
1 3 5
```

[3]:
```
list3=[] #Start with an empty list

for i in range(0,min(len(list1),len(list2))):
    list3.append(list1[i])
    list3.append(list2[i])
```

[4]:
```
if len(list1)>len(list2):
    list3.extend(list1[len(list2):])
elif len(list2)>len(list1):
    list3.extend(list2[len(list1):])

list3
```

[4]: [2, 1, 4, 3, 6, 5, 8]

49.Fully Dividing Sequence

1 A fully dividing sequence is a sequence a1,a2,. . . ,aN where ai divides aj whenever i < j.

For example, 3,15,60,720 is a fully dividing sequence.

Given a sequence of integers your aim is to find the length of the longest fully dividing subsequence of this sequence.

[1]:
```
n = int(input()) #Enter the length of the list

L = list() #Start with an empty list

best_stored = list() #Start with an empty list

for x in range(n):
  L.append(int(input()))
  best_stored.append(0)
```

```
5
2
4
6
8
16
```

[2]:
```
bestvals =list() #Start with an empty list

best_stored[0] = 1

for i in range(n):
  maxval = 1
  for j in range(i):
    if L[i] % L[j] == 0:
      maxval = max(maxval,(best_stored[j])+1)
  best_stored[i] = maxval
```

[3]:
```
print(max(best_stored),end="")
```

4

50.Histogram

1 Write a Python function histogram(l) that takes as input a list of integers with repetitions and returns a list of pairs as follows.

For each number n that appears in l, there should be exactly one pair (n,r) in the list returned by the function, where r is the number of repetitions of n in l.

The final list should be sorted in ascending order by r, the number of repetitions. For numbers that occur with the same number of repetitions, arrange the pairs in ascending order of the value of the number.

[1]:
```
def histogram(list1):
    d={} #Start with an empty dictionary
    for i in list1:
        if i in d.keys():
            d[i] +=1
        else:
            d[i] = 1
    ans = []
    for i in d.keys():
        number_of_times = d[i]
        ans.append((i, number_of_times))

    ans = sorted(ans, key=lambda x : x[0])
    ans = sorted(ans, key=lambda x : x[1])
    return ans
    #print(ans)
```

[2]:
```
x=[2,2,2,3,3,4,4,4,5]

histogram(x)
```

[2]:
```
[(5, 1), (3, 2), (2, 3), (4, 3)]
```

51.Expanding List

1 Write a function expanding(l) that takes as input a list of integer l and returns True if the absolute difference between each adjacent pair of elements strictly increases.

[1]:
```
def expanding(list1):
        diff= list1[1]-list1[0]
        for i in range(2,len(list1)):
            if abs(list1[i]-list1[i-1])>diff:
                diff=abs(list1[i]-list1[i-1])
            else:
                return False
        return True
```

[2]:
```
x=[2,3,5,8,12] #Initialize a list

expanding(x) #Call the function
```

[2] : True

52.Number of Occurrences

1 Write a Python program to count the number of occurrences of each digit/character in the input number/string

[1]:
```
str1 = input("Enter a string/number")
```

```
Enter a string/number1122233412
```

[2]:
```
str2=set(str1) #Converting 'string' data type to 'set' data type removes all duplicates

for i in str2:
    count1=str1.count(i)
    print("The frequency of digit/string ", i , " = ",count1)

#Space is also a character
```

```
The frequency of digit/string 3 = 2
The frequency of digit/string 2 = 4
The frequency of digit/string 4 = 1
The frequency of digit/string 1 = 3
```

53.Voting Machine

1 Voting was held in a class to elect the monitor.

You are given a list of names.

Write a Python program to find the winner.

If there is a tie, display the name which comes first in alphabetical order.

Sample Input : raveesh raveesh mahesh mahesh

Sample Output : mahesh,2

[1]:
```
names=input("Enter names separated by space ")
```

```
Enter names separated by space raveesh raveesh mahesh mahesh mahesh raveesh
```

[2]:
```
names=names.split() #split() function separates the elementys by space and gives a list
```

[3] :
```
names
```

[3] : ['raveesh', 'raveesh', 'mahesh', 'mahesh', 'mahesh', 'raveesh']

[4]:
```
keys=set(names) #set() function eliminates duplicates and gives only unique elements
```

[5]:
```
keys
```

[5] : {'mahesh', 'raveesh'}

[6] :
```
dict1={} #Start with an empty list
for i in keys:
    dict1[i]=names.count(i)
```

```
dict1 #dict1 contains name and vote
```

[6] : {'raveesh': 3, 'mahesh': 3}

[7] :
```
max_vote=max(dict1.values())
```

[8] :
```
max_vote
```

[8]: 3

[9] :
```
list1=[] #Start with an empty list
for i in dict1.keys():
    if dict1[i]==max_vote: #If a name has maximum votes, then append to the list
        list1.append(i)

list1
```

[9] : ['raveesh', 'mahesh']

[10] :
```
list2=sorted(list1) #Sort the list in alphabetical order
list2
```

[10] : ['mahesh', 'raveesh']

[11] :
```
print(list2[0],dict1[list2[0]]) #Display the name with maximum votes
```

mahesh 3

[12] :
```
#If you want to print name and vote separated by comma
print("{},{}".format(list2[0],dict1[list2[0]]))
```

mahesh,3

54. Vehicle Parking

1 The vehicles are parked in a parking lot in a (Row X Column) manner.

The status of the parking space is represented by 1(available)/ 0(Not available).

The status (0/1) of a parking space is represented as the element of the matrix.

Find the row in the parking lot which has the most available space to park the vehicle. The Row index starts from 1.

[1]:
```
r=int(input("Enter the number of rows"))

c=int(input("Enter the number of columns"))
```

```
Enter the number of rows3
Enter the number of columns4
```

[2]:
```
#Take input from the user separated by enter/newline
p=[]
for i in range(r*c):
    p.append(input())
```

```
1
1
0
0
1
0
0
0
0
0
```

```
0
0
```

[3]:
```
empty_space=0 #Assume that there are no emptly spaces

for i in range(r):
    list1=(p[i*c:(i+1)*c]) #Make a list row wise
    count1=list1.count('0') #Count the number of 0s
    if count1>empty_space:
        req_row=i+1
        empty_space=count1

print(req_row)
```

```
3
```

55.Remove Spaces and Find Length

1 Given a string s, remove any spaces if any and output the length of the string and whether it is even or odd.

[1]:
```
str1= input("Input a string")
```

```
Input a stringraveesh hegde cmrit ece
```

[2]:
```
str2 = str1.split() #Split the string and make a list

str3 = "".join(str2) #Join all the elements of the list
```

[3]:
```
if len(str3)%2==0:
    print(len(str3),"even")
else:
    print(len(str3),"odd")
```

```
20 even
```

56. Words Longer Than n

1 Write a Python program to input a sentence and make a list of words that are longer than 'n'. Take n and sentence as input.

[1]:
```
sentence=input("Enter a sentence")

n=int(input("Enter the minimum word length"))
```

```
Enter a sentenceBengaluru is a beautiful city
Enter the minimum word length3
```

[2]:
```
list1=sentence.split(" ") #Wherever space is there, sentence will be split and a␣
↪list will be formed

print(list1)
```

```
['Bengaluru', 'is', 'a', 'beautiful', 'city']
```

[3]:
```
list2=[] #Start with an empty array

for i in list1:
    if len(i)>n:
        list2.append(i)

print(list2)
```

```
['Bengaluru', 'beautiful', 'city']
```

57.Common and Uncommon Words

1 Write a Python program to separate common and uncommon words from two strings¶

[1]:
```
str1=input("Enter the first string - ")

str2=input("Enter the second string - ")
```

```
Enter the first string - raveesh hegde cmrit
Enter the second string - hegde cmrit bengaluru
```

[2]:
```
str1=str1.split()

str2=str2.split()
```

[3]:
```
common=[]

uncommon=[]

for i in str1:
    if i in str2:
        common.append(i)

print("Common words are ",common)

for i in str1:
    if i not in str2:
        uncommon.append(i)

for i in str2:
    if i not in str1:
        uncommon.append(i)

print("Uncommon words are", uncommon)
```

```
Common words are ['hegde', 'cmrit']
Uncommon words are ['raveesh', 'bengaluru']
```

58.Duplicate Character

1 Write a Python Program to remove all the duplicate characters in a given string

[1]:
```
string1=input() #Enter a string
```

raveesh hegde

[2]:
```
string2="" #Start with an empty string

for i in string1:
    if i not in string2:
        string2=string2+i

print("The new string is")

print(string2)
```

The new string is
ravesh gd

2 Write a Python Program to find all the duplicate characters in a given string, if no duplicates are present, then print "There are no duplicate characters in the string."

[3]:
```
name = input("Enter a string")
```

Enter a stringraveesh hegde

[4] :
```
l="" #Start with an empty string

for i in name:
    if name.count(i)>1 and i not in l:
        l=l+i

if len(l)>0:
    print(*l,sep=",")
```

```
else:
    print("There are no duplicate characters in the string.")
```

e,h

3 Write a Python program to remove duplicate words from a sentence

[5] :
```
sentence1=input("Enter a sentence")
```

Enter a sentenceraveesh hegde hegde cmrit ece ece

[6] :
```
sentence2=sentence1.split(" ") #Wherever space is there, sentence will be split and a list will be formed

print(sentence2)
```

['raveesh', 'hegde', 'hegde', 'cmrit', 'ece', 'ece']

[7] :
```
sentence3="" #Start with an empty sentence

for i in sentence2:
    if i not in sentence3:
        sentence3=sentence3+" "+i

print(sentence3)
```

raveesh hegde cmrit ece

4 Write a Python program that takes a sentence of words as input.

Removing all duplicate words and print the remaining words sorting them alphanumerically (based on ASCII value).

[8] :
```
st=input(" enter sentance").split()
```

enter sentanceraveesh hegde hegde cmrit ece ece

[9] :
```
st1=[]
c=0
for x in st:
    if x not in st1:
        st1.append(x)
print(sorted(st1))
```

['cmrit', 'ece', 'hegde', 'raveesh']

59.Remove Adjacent Duplicates

1 You are given a number with duplicate digits your task is to remove the immediate duplicate digits and print the result.

Sample Input : 1331

Sample Output : 131

[1]:
```
def remove_adjacent_dup(str1):
    result="" #Start with an empty string
    for i in range(len(str1)-1):
        if str1[i]!=str1[i+1]: #If present digit is not equal to next digit
            result=result+str1[i]
    result=result+str1[-1] #Concatenate last digit
    return int(result) #Convert the result into integer and return it
```

[2]:
```
num=input("Enter a number")

result=remove_adjacent_dup(num)

print(result)
```

```
Enter a number133322
132
```

60.First Non Repeating Character

1 You are given a string 'S' consisting of lowercase Letters.

Find the first non repeating character in S.

If you find all the characters are repeating print the answer as -1.

[1]:
```
str1=input() #Enter a string
```

engineering

[2]:
```
for i in str1:
    if str1.count(i)==1:
        print(i)
        break
else:
    print(-1)
```

r

61.Anagram and Heterogram

1 Write a Python program to group all anagrams present in a sentence together in a list.

Anagram means words containing same letters

For ex. 'listen' and 'silent' are anagrams.

[1]:
```
#Function definition

def isanagram(x,y):
    x=sorted(x.lower())
    y=sorted(y.lower())
    if x==y:
        return True
    else:
        return False
```

[2]:
```
#Main Program

list1=input() #Enter the words separated by space

list1=list1.split() #After split(), we get a list
```

listen cmrit ece silent

[3]:
```
list2=[] #Start with an empty list

for i in list1:
    for j in list1:
        if i!=j and isanagram(i,j):
            if i not in list2:
                list2.append(i)
            if j not in list2:
                list2.append(j)

if  len(list2)==0:
```

```
    print("No Anagrams")
else:
    print(*list2)
```

```
listen  silent
```

2 Write a Python program to check whether a given string is Heterogram or not.

A heterogram is a word, phrase, or sentence in which no letter of the alphabet occurs more than once.

For ex. 'the big dwarf only jumps'.

[4] :
```
str1=input() #Enter a string

str1=str1.replace(' ','') #Remove all spaces
```

```
the big dwarf only jumps
```

[5] :
```
str2=set(str1) # 'set' will remove all duplicates

if len(str1)==len(str2):
    print(True)
else:
    print(False)
```

```
True
```

62.Remove Repeated Words

Write a Python program to remove the repeated words of length greater than or equal to 5.

[1]:
```
str1=input() #Enter a sentence

str1=str1.split() #After split(), we get a list
```

raveesh raveesh ece ece

[2]:
```
list1=[] #This is an empty list

str3="" #This is an empty string

for i in str1:
    if i not in list1:
            list1.append(i) #This is to append the word to the list
            str3=str3+" "+i #This is to concatenate the word to the string
    else:
        if len(i)<5:
            list1.append(i) #This is to append the word to the list
            str3=str3+" "+i #This is to concatenate the word to the string

print(str3)
```

raveesh ece ece

63.Remove a Digit

1 Write a Python program to take List of numbers as elements and remove occurence of all a specific digit from every element and then return the resultant list.

Take size of the list, digit to be removed, and the list of elements as input.

Sample Input :

3

2

432 52 123

Sample Output :

[43, 5, 13]

[1]:
```
len1=int(input()) #Enter the number of elements
num=input() #enter the digit to replace
list1=input() #Enter the numbers separated by space
```

```
3
2
432 52 123
```

[2]:
```
list1=list1.split() #After split(), we get a list
```

[3]:
```
list2=[] #Start with an empty list
for i in list1:
    new=i.replace(num,"") #Replace the number with empty string

    list2.append(new) #Append the number to list2
```

[4]:
```
#Convert list elements into int

list2=[int(i) for i in list2 if i!=""]

list2
```

[4]: [43, 5, 13]

64.String Similarity

1 Write a Python program to find similarity between two strings.

For ex.

Input :

Python Exercises

Python Exercises

Output: 1

Input :

Python Exercises

Python Exercise

Output : 0.9696969696969697

[1]:
```
str1=input() #Enter first string

str2=input()  #Enter  second  string
```

```
Python Excercises
Python  Excercise
```

[2]:
```
str1=str1.lower() #Convert all letters to lower case

str2=str2.lower() #Convert all letters to lower case
```

[3]:
```
len1=len(str1)

len2=len(str2)

L=min(len1,len2)
```

[4]:
```
similar=0

for i in range(L):
    if str1[i]==str2[i]:
        similar=similar+1

print(similar*2/(len1+len2))
```

0.9696969696969697

1.1 Second Method

[5]:
```
str1=input() #Enter first string

str2=input()  #Enter  second  string
```

Python Excercises
Python Excercise

[6] :
```
str1=str1.lower() #Convert all letters to lower case

str2=str2.lower() #Convert all letters to lower case
```

[7] :
```
a=str1

b=str2
```

[8] :
```
import difflib

c=difflib.SequenceMatcher(a=a,b=b)
```

[9] :
```
print(c.ratio())
```

0.9696969696969697

65.Shadow Sentences

1 Shadow sentences are sentences where every word is the same length and order but without any of the same letters.

Write a python program that accepts two sentences that may or may not be shadows of each other.

The function should return True if they are and False if they aren't.

[1]:
```
s1=input() #Enter the first sentence

s2=input() #Enter the second sentence
```

```
raveesh hegde cmrit ece
pbcddqf abcfi xyzab pqr
```

[2]:
```
s1=s1.split() #After split(), we get a list

s2=s2.split() #After split(), we get a list
```

[3]:
```
flag=1
if len(s1)!=len(s2):
    flag=0
else:
    for i in range(len(s1)):
        if len(s1[i])!=len(s2[i]):
            flag=0
            break
        else:
            for j in range(len(s1[i])):
                for k in range(len(s1[i])):
                    if s1[i][j]==s2[i][k]:
                        flag=0
                        break

if flag==1:
```

```
    print(True)
else:
    print(False)
```

True

66.Vowels, Consonents, Digits

1 Write a Python program to count the number of vowels, consonents, digits, spaces, special characters in a string

[1]:
```
str1=input() #Enter a string
```

```
raveesh hegde @ CMRIT ECE 123
```

[2]:
```
spaces=[]

space_count=0

numbers=[]

numbers_count=0

upper_cases=[]

upper_cases_count=0

lower_cases=[]

lower_cases_count=0

special_char=[]

special_char_count=0

vowels=[]

vowels_count=0

consonents=[]

consonents_count=0
```

[3]:
```
for i in str1:
    if i==" ":
```

```
        space_count=space_count+1

    elif i.isupper():
        upper_cases.append(i)
        upper_cases_count=upper_cases_count+1
        if i in "AEIOU":
            vowels.append(i)
            vowels_count=vowels_count+1
        else:
            consonents.append(i)
            consonents_count=consonents_count+1

    elif i.islower():
        lower_cases.append(i)
        lower_cases_count=lower_cases_count+1
        if i in "aeiou":
            vowels.append(i)
            vowels_count=vowels_count+1
        else:
            consonents.append(i)
            consonents_count=consonents_count+1

    elif  i.isdigit():
        numbers.append(i)
        numbers_count=numbers_count+1

    else:
        special_char.append(i)
        special_char_count=special_char_count+1
```

[4]:
```
print("The number of spaces is ",space_count)

print("The upper case count is",upper_cases_count)

print("The lower case count is",lower_cases_count)

print("Vowels count is",vowels_count)

print("Consonent count is",consonents_count)

print("The digits count is",numbers_count)

print("The special char count is",special_char_count)
```

```
The number of spaces is 5
The upper case count is 8
The lower case count is 12
Vowels count is 8
```

Consonent count is 12
The digits count is 3
The special char count is 1

67.To Remove Vowels and Consonants from a String

1 Write a Python Program to remove all consonants from a string without affecting the spaces present in the string.

Sample Input : cmrit

Sample Output : i

[1]:
```
str1=input() #Input a string
```

```
cmrit ece
```

[2]:
```
str2="" #Start with an empty string

for i in str1:
    if i in "aeiouAEIOU" or i in " ":
        str2=str2+i #Here 'str2' and 'i' are strings. So, '+' does␣
↪concatenation, NOT arithmetic addition.
```

[3]:
```
print(str2)
```

```
i ee
```

2 Write a Python Program to remove all vowels from a string without affecting spaces. If there are no consonants, print -1.

Sample Input 1 : cmrit

Sample Output 1 : cmrt

Sample Input 2 : aaa eee iii

Sample Output 2 : -1

[4]:
```
str1=input() #Input a string
```

```
cmrit ece
```

[5]:
```
str2="" #Start with an empty string

for i in str1:
    if i not in "aeiouAEIOU":
        str2=str2+i #Here 'str2' and 'i' are strings. Hence, '+' does␣
↪concatenation, NOT arithmetic addition.
```

[6]:
```
if str2==" "*len(str2): #Here '*' does repetion of space, NOT arithmetic␣
↪multiplication.
                        #This step is to check whether all elements of str2 are␣
↪spaces.
  print("-1")
else:
  print(str2)
```

```
cmrt c
```

68.Highest Number of Vowels

1 Write a Python program to find the word in a sentence which has the most number vowels

[1]:
```
x=input() #Enter a string
```

raveesh hegde cmrit ece

[2]:
```
x=x.split() #After split(), we get a list
```

[3]:
```
count1=0

for i in x:
    count2=0
    for j in i:
        if j in "aeiouAEIOU":
            count2=count2+1
    if count2>count1:
        count1=count2
        req_word=i
```

[4]:
```
print(req_word)
```

raveesh

69.Substring

1 Write a Python program to check if a substring is present in a given string

[1]:
```
str1 =input() #Enter a string

str2 =input() #Enter the substring
```

```
raveesh hegde cmrit ece
ece
```

[2]:
```
if (str1.find(str2) == -1): #find() function returns -1 if str2 is not found in␣
↪str1
    print("Sub string is not present")
else:
    print("Substring is present")
```

```
Substring is present
```

70.Reverse the Words

1 Write Python program to reverse the words in a sentence.

Sample Input : raveesh hegde cmrit ece

Sample Output : ece cmrit hegde raveesh

[1]:
```
str1=input() #Enter a string
```

raveesh hegde cmrit

[2]:
```
str2 = str1.split() #After split(), we get a list

str3=str2[::-1] #Read the list in reverse order

output = " ".join(str3) #Join the elements of the list using space

print(output)
```

cmrit hegde raveesh

71.Reverse a Name and Capitalize Each Word

1 Sample Input : Raveesh Hegde

Sample Output : Edgeh Hseevar

[1]:
```
str1=input() #Enter a string
```

```
Raveesh Hegde
```

[2]:
```
str2=str1[::-1] #Reverse the string
```

[3]:
```
print(str2.title())
```

```
Edgeh Hseevar
```

72.Car Engine Number

1 A car company numbers the car's engine numbers based on the car's number plate. The engine number is the sum of all the integers present on the car's number plate. The issuing authority has hired you in order to provide the engine numbers to the cars. Your task is to develop an algorithm which takes input as in the form of string (Number Plate) and gives back the car's engine number in the form of an integer.

Sample Input: HR-05-AA-2119

Sample Output: 18

[1]:
```
n="0123456789" #This is a string

s=input() #Enter car number
```

```
ka-53-mj2038
```

[2]:
```
sum=0
for i in s:
    if i in n:
        sum=sum+int(i)
print(sum)
```

```
21
```

73.Replace a Character

1 Write a Python program to replace a specific character from a string

[1]:
```
str1=input() #Enter a string

a=input() #Which character should be replaced

b=input() #With what
```

```
raveesh
r
k
```

[2]:
```
str2="" #Start with an empty string

for i in str1:
    if i==a:
        str2=str2+b #Here '+' does concatenation of strings, NOT arithmetic addition
    else:
        str2=str2+i #Here '+' does concatenation of strings, NOT arithmetic addition
```

[3]:
```
print(str2) #This is the new string
```

```
kaveesh
```

74.Remove Duplicates and Sort Alphanumerically

1 Write a program that takes a sentence of words as input.

Remove all duplicate words and print the remaining words sorting them alphanumerically.

[1]:
```
str1=input() #Enter a Sentence

str1=str1.split() #After split(), we get a list
```

raveesh hegde cmrit hegde ece raveesh

[2]:
```
str2=set(str1) #Converting 'list' to 'set' will eliminate all duplicates

str3=list(str2) #Set cannot be indexed, but list can be indexed. So, convert
↪'str2' to list

str4=sorted(str3) #Sort the list alphanumerically

str5=" ".join(str4) #Join all the elements of str4 and give a single string

print(str5)
```

cmrit ece hegde raveesh

75.Fill With Stars

1 Write a Python program to take a line of text as input and Find the number of even length words in that line.

If those words length is less than 10, pad them in the right with the character '*' up to the length 10 .

Display the words after padding.

Sample Input : Hi How are you? Going to Bangalore

Sample Output :

Hi********

you?******

to********

[1]:
```
str1=input() #Enter a sentence

str1=str1.split() #After split(), we get a list
```

Hi How are you? Going to Bangalore

[2]:
```
for i in str1:
    if len(i)%2==0 and len(i)<10:
        print(i.ljust(10,'*'))
```

Hi********
you?******
to********

76.Wonderful String

1 Write a Python program to check whether a string is wonderful or not. A string is a wonderful string if it is made of exactly 3 different characters.

For ex. “aabbcc” is a wonderful string

Display “Wonderful” if the string is wonderful, else display -1.

[2]:
```
str1=input() #Enter a string
```

aabbbcccc

[3]:
```
if len(set(str1))==3:
    print("Wonderful")
else:
    print("-1")
```

Wonderful

77.Name, Company from Email Id

1 Write a Python program to input email addresses and print the user name and company names

Sample Input: peter@yahoo.com john@google.com xyz@accenture.com

Sample Ouput :

peter yahoo

john google

xyz accenture

[1]:
```
str1=input() #Enter email ids separated by space

str1=str1.split() #After split(), we get a list
```

```
peter@yahoo.com john@google.com xyz@accenture.com
```

[2]:
```
for i in str1:
    index1=i.find("@")
    index2=i.find(".")
    name=i[0:index1]
    company=i[index1+1:index2]
    print(name,company)
```

```
peter yahoo
john google
xyz accenture
```

2 Method 2 : Using Regular Expressions

[3]:
```
import re #Import Regular Expressions module
```

[4]:
```
str1 = input()#Enter email IDs separated by space

str1=str1.split() #After split(), we get a list
```

```
peter@yahoo.com john@google.com xyz@accenture.com
```

[5]:
```
sp = "(\w+)@((\w+)+.(\w{2,3}?).(\w{2,3})?)" #Form a search pattern
for i in str1:
  r2 = re.match(sp,i)
  print(r2.group(1)+" "+r2.group(3))
```

```
peter yahoo
john google
xyz accenture
```

78.Extract Email IDs from a Sentence

1 Given a sentence which consists of email ids, write a Python Program to extract all the email ids in the sentence.

Sample Input : I am John. My mail is jhon.b@gmail.com. My friend is Paul.c@yahoo.ac.in

Sample Output :

jhon.b@gmail.com

Paul.c@yahoo.ac.in

[1]:
```
import re #Import regular expressions module

txt=input() #Enter a sentence
```

I am John. My mail is jhon.b@gmail.com. My friend is Paul.c@yahoo.ac.in

[2]:
```
emailRegex = re.compile(r'''(
 [a-zA-Z0-9._%+-]+ # username
 @                 # @ symbol
 [a-zA-Z0-9.-]+    # domain name
  (\.[a-zA-Z]{2,4})    # dot-something
  (\.[a-zA-Z]{2,4})?
  )''', re.VERBOSE)
```

[3]:
```
list1=[] #Start with an empty list

list1=emailRegex.findall(txt) #Find all the matches

for i in list1:
    print(i[0])
```

jhon.b@gmail.com
Paul.c@yahoo.ac.in

79.Start with A, end with Z

1 You are given a list of words.

Find all the words that start with “A” or “a” and end with “Z” or “z”

Sample Input : [‘abcdz’, ‘xyzef’, ‘abafz’, ‘asdfgz’,‘1221z’]

Sample Output:[‘bcd’,’baf’]

[1]:
```
n=int(input()) #Enter the length of the list

list1=[] #Start with an empty list

for i in range(n):
    list1.append(input()) #Press "Enter" key after entering every element
```

```
4
asdfz
aghjjjz
az
ArhtZ
```

[2]:
```
list2=[] #Start with an empty list

for i in list1:
    if len(i)==5 and i[0] in "Aa" and i[-1] in "Zz":
        list2.append(i[1:4]) #Pick 2nd,3rd,4th characters

print(list2)
```

```
['sdf', 'rht']
```

80.Start with Hello, End with a Digit

1 Write a Python program that reads n strings as input.

Display all the strings that start with the word 'Hello' and end with the word of digits 0 - 9.

First input the number of strings and then the strings

Sample Input:

4

Hello How you 9

Helo How is

How are you99

Hello 999

Sample Output:

Hello How you 9

Hello 999

[1]:
```
n=int(input()) #Enter the number of strings
list1=[] #Start with an empty list
```

```
for i in range(n):
    list1.append(input())

#I have used this method of input, because sample inputs are given one below the␣
 ↪other
```

```
4
Hello How you 9
Helo  How is
How are you99
Hello  999
```

[2]:
```
list2=[] #Start with an empty list

for i in list1:
    if i[0:5]=="Hello" and i[-1].isdigit():
        list2.append(i)
```

[3]:
```
#To print the elements one below the other

print(*list2,sep="\n")
```

```
Hello How you 9
Hello  999
```

2 Method 2 : Using Regular Expressions

[4]:
```
import re #Import regular expression module
#Regular expression is a sequence of characters that forms a search pattern
```

[5]:
```
n=int(input("Enter the number of words"))
```

```
Enter the number of words4
```

[6]:
```
s=[]

i=0

while i<n:
  s1=input("Enter the next word")

  s.append(s1)

  i=i+1
```

```
Enter the next wordHello How you 9
Enter the next wordHelo How is
```

Enter the next wordHow are you99
Enter the next wordHello 999

[7] :
```
for i in s:

  if re.search("^Hello",i): #'^' indicates 'starts with'

    if re.search("[0-9]$",i): #'$' indicates 'ends with'

      print(i)
```

Hello How you 9
Hello 999

3 Method 3 : Using Regular Expressions

[8] :
```
import re
n=int(input())
i=0
lst=[]
while i<n:
    item=input()
    lst=lst+ [item]
    i=i+1
pattRegex = re.compile(r'^Hello .*\d+$')
for i in lst:
    if pattRegex.search(i)!=None:
        print(i)
```

4
Hello How you 9
Helo How is
How are you99
Hello 999
Hello How you 9
Hello 999

81.Parenthesis Matching

1 Write a Python program to check whether parenthesis are matched in a given string.

Return True or False depending on whether parentheis are matched or not.

[1]:
```
#Function Definition

def par_match(s):
    count=0
    for i in s:
        if i == ')':
            count=count-1
            if count<0:
                return False
        elif i=='(':
            count=count+1
    if count==0:
        return True
    else:
        return False
```

[2]:
```
str1=input() #Enter a string

print(par_match(str1)) #Call the function and print 'True' or 'False'
```

```
)(
False
```

82. Valid USN

1 **A University Serial Number (USN) contains 1 digit region code, followed by 2 character college code, 2 digit year, 2 character branch code, 3 digit roll number.**

Write a Python program to take 'n' strings and print all the strings which are USNs.

Sample Input :

4 (This is the number of input strings)

askdf

1CR20IS003

1CR19EC112

2CE18EC110

Sample Output :

1CR20IS003

1CR19EC112

2CE18EC110

[1]:
```
n=int(input()) #Enter the number of strings
```

```
4
```

[2]:
```
#Input strings are given one below the other.

#So, use the following method to take input

x=[]
for i in range(n):
    x.append(input()) #Press 'Enter' key after typing every input
```

```
askdf
1CR20IS003
1CR19EC112
2CE18EC110
```

[3]:
```
#Function Definition

def isusn(x):
    if len(x)==10:
        cond1=i[0].isdigit() #Returns true if all string elements are digits
        cond2=i[1:3].isalpha() #Returns true if all string elements are␣
↪alphabetic
        cond3=i[3:5].isdigit() #Returns true if all string elements are digits
        cond4=i[5:7].isalpha() #Returns true if all string elements are␣
↪alphabetic
        cond5=i[7:10].isdigit() #Returns true if all string elements are digits
        if cond1 and cond2 and cond3 and cond4 and cond5:
            return True
    return False
```

[4]:
```
list2=[] #Start with an empty list

for i in x:
    if isusn(i):
        list2.append(i)
```

[5]:
```
if len(list2)==0:
    print(None)
else:
    print(*list2,sep="\n") #To print the elements one below the other
```

```
1CR20IS003
1CR19EC112
2CE18EC110
```

2 Method 2 : Using Regular Expressions

[6]:
```
import re #Import Regular Expressions module

n=int(input()) #Enter the numbers of strings

list1=[] #Start with an empty list

for i in range(n):
    list1.append(input()) #Press 'Enter' key after typing every input
```

```
4
askdf
1CR20IS003
1CR19EC112
2CE18EC110
```

[7]:
```
sp = re.compile(r'\d{1}\w{2}\d{2}\w{2}\d{3}') #Form a search pattern. \d for␣
 ↪digit, \w for letter

list2=[] #Start with an empty list

for i in list1:
    if sp.search(i)!=None:
        list2.append(i)
```

[8]:
```
if len(list2)==0:
    print(None) #If there are NO valid USNs
else:
    print(*list2,sep="\n") #To print the elements one below the other
```

```
1CR20IS003
1CR19EC112
2CE18EC110
```

83.Valid Phone Number

1 Write a Python program to check the validity of a phone number.

Valid : 123-4563-7891

Invalid : 123-4563+7891

Print True if it is valid. Otherwise print False

[1]:
```
def is_phonenum(num):
    list1=num.split("-")
    if len(list1)==3:
        n1=list1[0]
        n2=list1[1]
        n3=list1[2]
        condition1=len(n1)==3 and n1.isdigit()
        condition2=len(n2)==4 and n2.isdigit()
        condition3=len(n3)==4  and  n3.isdigit()
        if condition1 and condition2 and condition3:
            return True
    return False
```

[2]:
```
phone_number=input() #Enter a phone number
```

```
123-4563-7891
```

[3]:
```
if is_phonenum(phone_number):
    print(True)
else:
    print(False)
```

```
True
```

2 Given a string containing phone number, write a python program to identify the same and print the Phone Number.

A phone number follows the following format :XXX-XXXX-XXXX

If no Phone number is available in the string print "Nil"

Sample Input - 1 : My Phone number is 080-2847-4463

Sample Output - 1: 080-2852-4477

Sample Input - 2: The country code for India is +91

Sample Output - 2 : Nil

[4]:
```
def is_phonenum(num):
    list1=num.split("-")
    if len(list1)==3:
        n1=list1[0]
        n2=list1[1]
        n3=list1[2]
        condition1=len(n1)==3 and n1.isdigit()
        condition2=len(n2)==4 and n2.isdigit()
        condition3=len(n3)==4  and  n3.isdigit()
        if condition1 and condition2 and condition3:
            return True
    return False
```

[5]:
```
str1=input() #Enter a sentence
```

```
My Phone number is 080-2847-4463
```

[6]:
```
list1=str1.split() #After split(), we get a list
```

[7]:
```
for i in list1:
    if is_phonenum(i):
        print(i)
        break
else:
    print("Nil")
```

```
080-2847-4463
```

3 Method 2 : Using Regular Expressions

[8]:
```
import re #Import Regular Expressions module

list1=input() #Enter a sentence

list1=list1.split() #After split(), we get a list
```

```
My Phone number is 080-2847-4463
```

[9] :
```
sp = re.compile(r'\b\d{3}-\d{4}-\d{4}\b') #Form a search pattern

flag=0 #flag=0 indicates thet valid phone number is not present

for i in list1:
    if sp.search(i)!=None:
        print(i)
        flag=1
if flag==0:
    print("Nil")
```

```
080-2847-4463
```

84.Valid PAN

1 Write a Python program to check the validity of a PAN.

PAN must have 10 characters, first 5 elements must be letters, next 4 elements must be digits, last element must be a letter

Sample Input : HXTPS2142R

Sample Output : Valid PAN

[1]:
```
str1=input() #Enter a string
```

```
BDDQS6617D
```

[2]:
```
flag=0

if len(str1)!=10:
    flag=0
else:
    if str1[0:5].isalpha() and str1[5:9].isdigit() and str1[9].isalpha():
        flag=1

if flag==1:
    print("Valid PAN")
else:
    print("Invalid PAN")
```

```
Valid PAN
```

2 Method 2 : Using Regular Expressions

[3]:
```
import re #Import Regular Expressions module

str1= input() #Enter a string
```

```
BDDQS6617D
```

[4]:
```
sp = re.compile("[A-Z]{5}[0-9]{4}[A-Z]{1}") #Form a search pattern

if sp.search(str1)!=None:
    print("Valid PAN")
else:
    print("Invalid PAN")
```

```
Valid  PAN
```

85.Valid Date

1 Write a Python program to take dates as string input and convert a date of yyyy-mm-dd format to dd-mm-yyyy format.

If the date is in dd-mm-yyyy format do not change it.

Sample Input: 2016-01-20 2020-05-25 12-10-2008 01-12-2007 1999-12-21

Sample Output: 20-01-2016 25-05-2020 12-10-2008 01-12-2007 21-12-1999

[1]:
```
str1=input() #Enter dates separated by space

str1=str1.split() #After split(), we get a list
```

```
2016-01-20 2020-05-25 12-10-2008 01-12-2007 1999-12-21
```

[2]:
```
list1=[] #Start with an empty list

for i in str1:
    str2=i.split("-") #After split(), we get a list

    if(len(str2[0])==4):
        str2.reverse()
        str3="-".join(str2)
        list1.append(str3)
    else:
        list1.append(i)
```

[3]:
```
print(*list1) #We use this method of printing because sample output shows that␣
↪output values should be printed in a sinle line separated by space
```

```
20-01-2016 25-05-2020 12-10-2008 01-12-2007 21-12-1999
```

2 Method 2 : Using Regular Expressions

[4]:
```
import re #Import Regular Expressions module

str1=input() ##Enter dates separated by space
```

```
2016-01-20 2020-05-25 12-10-2008 01-12-2007 1999-12-21
```

[5]:
```
sp = re.compile(r'(\d{4})-(\d{1,2})-(\d{1,2})') #Form a search pattern

valid_dates=sp.sub('\\3-\\2-\\1',str1) #sub() method is to substitute one
↪pattern with another pattern

print(valid_dates)
```

```
20-01-2016 25-05-2020 12-10-2008 01-12-2007 21-12-1999
```

86.Valid Password

1 Write a Python program to check the validity of password input by users.

Following are the criteria for checking the password:

At least 2 letter between [a-z]

At least 2 number between [0-9]

At least 1 letter between [A-Z]

At least 1 character from @

Minimum length of transaction password: 8

Maximum length of transaction password: 15

Your program should accept a list of passwords and check them according to the above criteria.

Passwords that match the criteria are to be printed.

Sample Input :

5

ABd12t34@1

a F1ttt

2w3EEEgw3*

2We3345

ab34#ullo*12AakQ

Sample Output :

ABd12t34@1

[1]:
```
n=int(input()) #Enter the length of the list
```

```
5
```

[2]:
```
list1=[] #Start with an empty list

for i in range(n):
    list1.append(input()) #Press 'Enter' key after typing every string
```

```
ABd12t34@1
a F1ttt
2w3EEEgw3*
2We3345
ab34#ullo*12AakQ
```

[3]:
```
def valid(x):
    if len(x)<8 or len(x)>15:
        return(False)
    else:
        low=0
        up=0
        d=0
        s=0
        for i in x:
            if i.islower():
                low=low+1
            elif i.isupper():
                up=up+1
            elif i.isdigit():
                d=d+1
            elif i in "$#@":
                s=s+1
        if low>1 and up>0 and d>1 and s>0:
            return(True)
        else:
            return(False)
```

[4]:
```
l=0
for i in list1:
    if valid(i):
        print(i)
        l=l+1

if l==0:
    print("None")
```

```
ABd12t34@1
```

2 Method 2 : Using Regular Expressions

[5]:
```
n=int(input()) #Enter the length of the list
```

```
5
```

[6]:
```
list1=[] #Start with an empty list

for i in range(n):
    list1.append(input()) #Press 'Enter' key after typing every string
```

```
ABd12t34@1
a F1ttt
2w3EEEgw3*
2We3345
ab34#ullo*12AakQ
```

[7]:
```
import re #Import Regular Expressions module

list2=[] #Start with an empty list

for i in list1:
    cond1=len(i)>=8 and len(i)<=15 #There should be at least 8 and at most 15 letters in the password

    cond2=len(re.findall('\d',i))>=2 #There should be at least 2 digits

    cond3=len(re.findall('[a-z]',i))>=2 #There should be at least 2 lower case letters

    cond4=len(re.findall('[A-Z]',i))>=1 #There should be at least 1 upper case letter

    cond5=len(re.findall('[$#@]',i))>=1 #There should be at least 1 special character

    if cond1 and cond2 and cond3 and cond4 and cond5:
        list2.append(i)


if len(list2) == 0: #If there is no valid password in the given list
    print(None)
else:
    print(*list2,sep="\n") #To print the valid passwords one below the other
```

```
ABd12t34@1
```

87.Valid Vehicle Number

1 Write a Python program to find the valid vehicle numbers from the given list of numbers.

Sample Input :

4

KA55NK2040

AB68XY3049

55KANK2040

AB6830XY49

Sample Output :

KA55NK2040

AB68XY3049

[1]:
```
n=int(input()) #Enter the length of the list

list1=[] #Start with an empty list

for i in range(n):
    list1.append(input()) #Press 'Enter' key after typing every element
```

```
4
KA55NK2040
AB68XY3049
55KANK2040
```

```
AB6830XY49
```

[2]:
```
list2=[] #Start with an empty list

for i in list1:
    cond1=len(i)==10 #There should be 10 letters in the vehicle number

    cond2=i[0:2].isalpha() #First 2 elements should be alphabetic

    cond3=i[2:4].isdigit() #Next 2 elements should be digits

    cond4=i[4:6].isalpha() #Next 2 elements should be alphabetic

    cond5=i[6:10].isdigit() #Next 4 elements should be digits

    if cond1 and cond2 and cond3 and cond4 and cond5:
        list2.append(i)
```

[3]:
```
if len(list2)==0:
    print(None) #If there are NO valid vehivle numbers
else:
    print(*list2,sep="\n") #To print the elements one below the other
```

```
KA55NK2040
AB68XY3049
```

2 Method 2 : Using Regular Expressions

[4]:
```
n=int(input()) #Enter the length of the list

list1=[] #Start with an empty list

for i in range(n):
    list1.append(input()) #Press 'Enter' key after typing every element
```

```
4
KA55NK2040
AB68XY3049
55KANK2040
AB6830XY49
```

[6]:
```
import re #Import reguler expressions module

sp = re.compile(r'\w{2}\d{2}\w{2}\d{4}') #Form a search pattern. \d for digit,␣
 ↪\w for letter. There should be 2 letters,2 digits,2 letters and 4 digits

list2=[] #Start with an empty list
```

```
for i in list1:
    if sp.search(i)!=None:
        list2.append(i)
```

[7]:
```
if len(list2)==0:
    print(None) #If there are NO valid vehicle numbers
else:
    print(*list2,sep="\n") #To print the elements one below the other
```

```
KA55NK2040
AB68XY3049
```

88.Valid IP Address

1 Write a Python program to print valid IP addresses from the given list.
A valid IP address contains
Four numbers ranging from 0 to 255 separated by dot
A number cannot start with 0 unless 0 is the only digit in that number

Sample Input :

4 (This number gives the length of the list)

192.168.0.1

292.168.0.1

192.168.02.1

192.16.0.1

Sample Output :

192.168.0.1

192.16.0.1

Print 'None' if there are no valid IP addresses

[1]:
```
n=int(input()) #Enter the number of inputs
list1=[] #Start with an empty list
```

```
for i in range(n):
    list1.append(input()) #Press 'Enter' key after typing every element
```

```
4
192.168.0.1
292.168.0.1
192.168.02.1
192.16.0.1
```

[2]:
```
def valid_ip(str1):
    list2=str1.split(".") #After split(), we get a list
    if len(list2)==4:
        for i in list2:
            if i.isdigit()==False or int(i)<0 or int(i)>255 or (i[0]=="0" and
 ↪len(i)!=1):
                return False
        else:
            return True
    else:
        return False
```

[3]:
```
list3=[] #Start with an empty list

for i in list1:
    if valid_ip(i):
        list3.append(i)
```

[4]:
```
if len(list3)==0:
    print(None) #If there are no valid IP addresses
else:
    print(*list3,sep="\n") #To print the elements one below the other
```

```
192.168.0.1
192.16.0.1
```

89.Valid URL

1 Take n URL strings as input, write a Python program to display the proper URLS, and also display count of the URLs that start with 'https' and ends with 'com'

Input :
6
https://www.yahoo.com
http://blog.hubspot.com/marketing/Index.html
http://www.cmrit.ac.in
https://www.karnatakacareers.in
ftp://internet.address.edu/file/path/file.txt
abc:/xxx.abc.vvv.in

Output:
https://www.yahoo.com
http://blog.hubspot.com/marketing/Index.html
http://www.cmrit.ac.in
https://www.karnatakacareers.in
ftp://internet.address.edu/file/path/file.txt

1

[1]:
```
import re #Import REgular Expressions module

n=int(input()) #Enter the number of URLs
```

```
6
```

[2]:
```
#Enter URLs one below the other
list1=[]
```

```
for i in range(n):
    list1.append(input())
```

```
https://www.yahoo.com
http://blog.hubspot.com/marketing/Index.html
http://www.cmrit.ac.in
https://www.karnatakacareers.in
ftp://internet.address.edu/file/path/file.txt
abc:/xxx.abc.vvv.in
```

[3]:
```
sp1 = re.compile(r'http(s)?|ftp://(\w)+(\.)(\w)+(\.)(\w){2,4}(/\w+){0,}((/\w+)\.(html|txt)?)') #Form a search pattern
```

[4]:
```
list2=[] #Start with an empty list

for i in list1:
    if sp1.search(i)!=None:
        list2.append(i)
```

[5]:
```
print(*list2,sep="\n") #To print the elements one below the other
```

```
https://www.yahoo.com
http://blog.hubspot.com/marketing/Index.html
http://www.cmrit.ac.in
https://www.karnatakacareers.in
ftp://internet.address.edu/file/path/file.txt
```

[6]:
```
sp2=re.compile(r'^https(.)+com$') #Form a search pattern to find the URLs that start with 'https' and end with 'com'

count=0
for i in list2:
    if sp2.search(i)!=None:
        count+=1

print(count)
```

```
1
```

90.Exception Handling

1 Write a Python program which takes 2 parameters a, b and prints the value of c where c=a/b.

Raise an exception if value of b is 0

[1]:
```
a = int(input()) #Enter the value of a

b = int(input()) #Enter the value of b

try:
    c  = a/b
    print(c)
except:
    print("Error : Division by Zero")
```

```
2
0
Error : Division by Zero
```

91.Top 10 Words

1 Write a Python program to print 10 most frequently appearing words in a text file.

[1]:
```
my_file = open('My Text File.txt') # Open the file in read mode
```

[2]:
```
sentence_list=my_file.readlines() # Return all lines of the file, as a list␣
↪where each line is an item in the list object
```

[3]:
```
word_list = [] #Start with an empty list. This list will eventually contain all␣
↪words in the text file

for i in sentence_list:
  word_list = word_list+i.split() #split() function separates words
```

[4]:
```
word_dict = {} #Start with an empty dictionary. This list will eventually␣
↪contain all words in the text file and their frequency

word_set=set(word_list) #set() will remove all duplicate items. Resulting␣
↪elements will form the keys for the dictionary.

for i in word_set:
  word_dict[i] = word_list.count(i) # Creating a dictionary with word and count.
```

[5]:
```
#sorted_keys = sorted(word_dict,key=word_dict.get, reverse=True) #This will␣
↪arrange the keys of the dictionary in reverse order of their frequency
sorted_keys = sorted(word_dict,reverse=True) #This will arrange the keys of the␣
↪dictionary in reverse order of their frequency

count=0
for i in sorted_keys: # 'i' will give the keys in the dictionary
  print(" Frequency of word '", i,"'=", word_dict[i])
  count+=1
  if(count == 10):
    break
```

```
Frequency of word ' written '= 1
Frequency of word ' world.[4][5][6] '= 1
```

Frequency of word " with '= 1
Frequency of word " was '= 3
Frequency of word " to '= 1
Frequency of word " the '= 14
Frequency of word " that '= 1
Frequency of word " than '= 1
Frequency of word " supreme '= 1
Frequency of word ' supremacy, '= 1

92.Sort a Text File

1 Write a program to sort the contents of a text file and write the sorted contents into a separate text file.

[1]:
```
fp = open("My Text File.txt", "r")
```

[2]:
```
word_lst = []
for i in fp:
  word_lst+=i.split()
  # append all the words to the list -> lst
```

[3]:
```
word_lst.sort()
  # Sorts all the words alphabetical order.
str = " ".join(word_lst)
  # Joins all words by the delimited " " = space.
```

[4]:
```
fp = open("textw.txt", "w")
  # textw.txt is the file to which the sorted content to be written.
  # Open the file to be written in write mode.
fp.write(str)
  # Write the content.
fp.close()
  # Close the file pointer.
```

93.ZIP a Folder

1 Write a program to backing Up a given Folder (Folder in a current working directory) into a ZIP File by using relevant modules and suitable methods.

[1]:
```
import shutil
import os.path

# Creating the ZIP file
#archived = shutil.make_archive('E:/Zipped file', 'zip', 'E:/Folder to be zipped')
archived = shutil.make_archive('Zipped Folder', 'zip', 'To Zip Folder')

if os.path.exists('Zipped Folder.zip'):
    print(archived)
else:
    print("ZIP file not created")
```

```
C:\Users\cmrit\Desktop\Python Programs\Final Programs\Zipped Folder.zip
```

94.Class Complex

1 Define a function which takes TWO objects representing complex numbers and returns new complex number with a addition of two complex numbers.

Define a suitable class 'Complex' to represent the complex number.

Develop a program to read N (N >=2) complex numbers and to com- pute the addition of N complex numbers.

[1]:
```
# User Defined class
class Complex:
    # Constructor to accept real and imaginary part
    def __init__(self, tempReal, tempImaginary):
        self.real = tempReal
        self.imaginary = tempImaginary

    # Defining addComplex() method for adding two complex number
    def addComplex(self, C1, C2):

        # creating temporary object of class Complex
        temp=Complex(0, 0)
        # adding real part of complex numbers
        temp.real = C1.real + C2.real
        # adding Imaginary part of complex numbers
        temp.imaginary = C1.imaginary + C2.imaginary
        # returning the sum
        return temp
```

[2]:
```
a  = input("Enter real and imaginary part of a complex number A : ")
# Complex number as real an imaginary part.
# Enter the real part and imaginary part as a list separated by space
# Both can be read as string and then convert into a list of real and imaginary␣
 ↪to compute.
```

```
a = a.split()
areal = int(a[0])
aimaginary = int(a[1])
```

```
Enter real and imaginary part of a complex number A : 2 4
```

[3]:
```
b = input("Enter real and imaginary part of a complex number B : ")
b = b.split()
breal = int(b[0])
bimaginary = int(b[1])
```

```
Enter real and imaginary part of a complex number B : 1 3
```

[4]:
```
complexR = Complex(0,0)
complexA = Complex(areal, aimaginary)
complexB = Complex(breal, bimaginary)
complexR =complexR.addComplex(complexA, complexB)
```

[5]:
```
print("Sum of Complex Number A and B = %d + i%d"%(complexR.real, complexR.
↪imaginary))
```

```
Sum of Complex Number A and B = 3 + i7
```

95.Class Student

1 Write a program that uses class Student which prompts the user to enter marks in three subjects and calculates total marks, percentage and displays the score card details.

[1]:
```
# User defined class
class Student:
    # Method to update the basic details of the student
    def __init__(self):
        self.rollno=input("Enter Roll Number : ")
        self.name = input("Enter Student Name : ")
        self.marks = []
    # setter method to read marks.
    def setMarks(self):
        self.marks.append(int(input("Enter Physics Marks : ")))
        self.marks.append(int(input("Enter Chemistry Marks : ")))
        self.marks.append(int(input("Enter Math Marks : ")))
    # getter method to display the score card.
    def display(self):
        print("Roll Number\t\t: ", self.rollno)
        print("Student Name\t\t: ", self.name)
        print("Physics Marks\t\t: ", self.marks[0])
        print("Chemistry Marks\t\t: ", self.marks[1])
        print("Physics Marks\t\t: ", self.marks[2])
        total=sum(self.marks)
        avg=total/3
        print("Total marks = %0.2f / 300 \t Average Score = %0.2f"%(total, avg))
```

[2]:
```
a = Student()
print("\nEnter your scores out of 100 in Physics, Chemistry and Mathematics.\n")
a.setMarks()
print("\n---------SCORE CARD ---------- \n")
a.display()
```

```
Enter Roll Number : 000
Enter Student Name : raveesh hegde

Enter your scores out of 100 in Physics, Chemistry and Mathematics.
```

Enter Physics Marks : 35
Enter Chemistry Marks : 36
Enter Math Marks : 37

---------SCORE CARD ---------

Roll Number : 000
Student Name : raveesh hegde
Physics Marks : 35
Chemistry Marks : 36
Physics Marks : 37
Total marks = 108.00 / 300 Average Score = 36.00

96.Class Circle

1 Write a Python program to create a class representing a Circle. Include methods to calculate its area and perimeter.

Sample Input: 4

Sample Output:

Area 50.27

Perimeter: 25.13

[1]:
```
import math
class Circle:
    def __init__(self, radius):
        self.radius = radius

    def calculate_circle_area(self):
        return math.pi * self.radius**2

    def calculate_circle_perimeter(self):
        return 2 * math.pi * self.radius
```

[2]:
```
radius = float(input()) #Input the radius of the circle

circle = Circle(radius)

area = circle.calculate_circle_area()

perimeter = circle.calculate_circle_perimeter()
```

```
4
```

[3]:
```
print("Area: {:.2f}".format(area))

print("Perimeter: {:.2f}".format(perimeter))
```

Area: 50.27
Perimeter: 25.13

97.Add Two Time Objects

1 Write a Python program to add the two time objects and display

[1]:
```
class t:
  def _init_(self,hh,mm,ss):
    self.hh=hh
    self.mm=mm
    self.ss=ss
  def addition(self,h,m,s):
    if self.ss>59 or s>59:
      print("invalid input")
    else:
      self.ss=self.ss+s
      if self.ss>60:
        self.mm=self.mm+1
        self.ss=self.ss-60
      if self.mm>59 or m>59:
        print("invalid input")
      else:
        self.mm=self.mm+m
        if self.mm>60:
          self.hh=self.hh+1
          self.mm=self.mm-60
        self.hh=self.hh+h
        print(str(self.hh)+":"+str(self.mm)+":"+str(self.ss))
```

[4]:
```
a,b,c=input().split()
x,y,z=input().split()
obj=t(int(a),int(b),int(c))
obj.addition(int(x),int(y),int(z))
```

```
10 20 30
1 2 3
11:22:33
```

98.LInear Search

1 Write a Python program to find the index of an element using linear search.

Return the index of the element.

Return -1 if the element is not found.

[1]:
```
x=input() #Enter space separated values

x=x.split() #After split(), we get a list

x=[int(i) for i in x] #Convert every element of 'x' to 'int'
```

```
2 4 6 8
```

[2]:
```
a=int(input()) #Enter the value to be found
```

```
6
```

[3]:
```
flag=0 #flag=0 indicates that element is not found

for i in range(len(x)):
    if x[i]==a:
        print(i)
        flag=1 #flag=1 indicates that element is found

if  flag==0:
    print(-1) #Print -1 if the element is not found
```

```
2
```

99.Binary Search

1 Write a Python program to find the index of an element using binary search.

Return the index of the element.

Return -1 if the element is not found

[1]:
```
def binary_search(list1, x):
    low = 0
    high = len(list1) - 1

    while low <= high:

        mid = (high + low) // 2

        if list1[mid]==x:
            return mid

        if list1[mid] < x:
            low = mid + 1

        elif arr[mid] > x:
            high = mid - 1

   # If we reach here, then the element was not present
    return -1
```

[2]:
```
list1=input() #Enter space separated values

list1=list1.split() #After split(), we get a list

list1=[int(i) for i in list1] #Convert every element of 'x' to 'int'

x=int(input()) #Enter the value to be found
```

```
2 4 6 8 10
```

8

[3]:
```
# Function call
result = binary_search(list1, x)
```

[4]:
```
if result != -1:
        print("Element is present at index", str(result))
else:
        print("Element is not present in array")
```

Element is present at index 3

100. Bubble Sort

1 Write a Python program to sort an array using Bubble Sort technique.

[1]:
```
# Python3 program for Bubble Sort Algorithm Implementation
def bubbleSort(x):
    n = len(x)
    for i in range(n):
        for j in range(0, n - i - 1):
            # Range of the array is from 0 to n-i-1
            # Swap the elements if the element found
            #is greater than the adjacent element
            if x[j] > x[j + 1]:
                x[j], x[j + 1] = x[j + 1], x[j] #Exchange x[j] and x[j+1]
```

[2]:
```
# Example to test the above code
arr = [ 2, 1, 10, 23 ]

bubbleSort(arr)

print("Sorted array is:")
for i in range(len(arr)):
    print("%d" % arr[i])
```

```
Sorted array is:
1
2
10
23
```

101.Selection Sort

1 Write a Python program to sort an array using Selection Sort technique.

[1]:
```
# Selection Sort algorithm in Python
def selectionSort(array, size):
    for s in range(size):
        min_idx = s
        for i in range(s + 1, size):
        # For sorting in descending order
        # for minimum element in each loop
            if array[i] < array[min_idx]:
                min_idx = i

        # Arranging min at the correct position
        (array[s], array[min_idx]) = (array[min_idx], array[s])
```

[2]:
```
# Driver code
data = [ 7, 2, 1, 6 ]
size = len(data)
selectionSort(data, size)

print('Sorted Array in Ascending Order is :')
print(data)
```

```
Sorted Array in Ascending Order is :
[1, 2, 6, 7]
```

102.Insertion Sort

1 Write a Python program to sort an array using Insertion Sort technique.

[1]:
```
# Creating a function for insertion sort algorithm
def insertion_sort(list1):
        # Outer loop to traverse on len(list1)
        for i in range(1, len(list1)):
            a = list1[i]
            # Move elements of list1[0 to i-1],
            # which are greater to one position
            # ahead of their current position
            j = i - 1
            while j >= 0 and a < list1[j]:
                list1[j + 1] = list1[j]
                j -= 1
                list1[j + 1] = a
        return list1
```

[2]:
```
# Driver code
list1 = [ 7, 2, 1, 6 ]
print("The unsorted list is:", list1)
print("The sorted new list is:", insertion_sort(list1))
```

```
The unsorted list is: [7, 2, 1, 6]
The sorted new list is: [1, 2, 6, 7]
```

103.Merge Sort

1 Write a Python program to sort an array using Merge Sort technique.

[1]:
```
def merge_sort(unsorted_list):
    if len(unsorted_list) <= 1:
        return unsorted_list
# Find the middle point and devide it
    middle = len(unsorted_list) // 2
    left_list = unsorted_list[:middle]
    right_list = unsorted_list[middle:]

    left_list = merge_sort(left_list)
    right_list = merge_sort(right_list)
    return list(merge(left_list, right_list))

# Merge the sorted halves
def merge(left_half,right_half):
    res = []
    while len(left_half) != 0 and len(right_half) != 0:
        if left_half[0] < right_half[0]:
            res.append(left_half[0])
            left_half.remove(left_half[0])
        else:
            res.append(right_half[0])
            right_half.remove(right_half[0])
    if len(left_half) == 0:
        res = res + right_half
    else:
        res = res + left_half
    return res
unsorted_list = [64, 34, 25, 12, 22, 11, 90]
print(merge_sort(unsorted_list))
```

```
[11, 12, 22, 25, 34, 64, 90]
```

104.Quick Sort

1 Write a Python program to sort an array using Quick Sort technique.

[1]:
```
# Python program for implementation of Quicksort Sort

# This implementation utilizes pivot as the last element in the nums list
# It has a pointer to keep track of the elements smaller than the pivot
# At the very end of partition() function, the pointer is swapped with the pivot
# to come up with a "sorted" nums relative to the pivot


# Function to find the partition position
def partition(array, low, high):

    # choose the rightmost element as pivot
    pivot = array[high]

    # pointer for greater element
    i = low - 1

    # traverse through all elements
    # compare each element with pivot
    for j in range(low, high):
        if array[j] <= pivot:

            # If element smaller than pivot is found
            # swap it with the greater element pointed by i
                i = i + 1

                # Swapping element at i with element at j
                (array[i], array[j]) = (array[j], array[i])

    # Swap the pivot element with the greater element specified by i
    (array[i + 1], array[high]) = (array[high], array[i + 1])

    # Return the position from where partition is done
    return i + 1
```

```
# function to perform quicksort


def quickSort(array, low, high):
    if low < high:

        # Find pivot element such that
        # element smaller than pivot are on the left
        # element greater than pivot are on the right
        pi = partition(array, low, high)

        # Recursive call on the left of pivot
        quickSort(array, low, pi - 1)

        # Recursive call on the right of pivot
        quickSort(array, pi + 1, high)


data = [1, 7, 4, 1, 10, 9, -2]
print("Unsorted Array")
print(data)

size = len(data)

quickSort(data, 0, size - 1)

print('Sorted Array in Ascending Order:')
print(data)
```

```
Unsorted Array
[1, 7, 4, 1, 10, 9, -2]
Sorted Array in Ascending Order:
[-2, 1, 1, 4, 7, 9, 10]
```