Strings in Python

String Declaration

Strings in Python

- In python, strings can be created by enclosing the character or the sequence of characters in the quotes.
- Python allows us to use single quotes, double quotes, or triple quotes to create strings.

```
Example:
str1 = 'Hello Python'
str2 = "Hello Python"
str3 = '''Hello Python'''
```

- In python, strings are treated as the sequence of characters which means that python doesn't support the character data type instead a single character written as **'p'** is treated as the string of length 1.

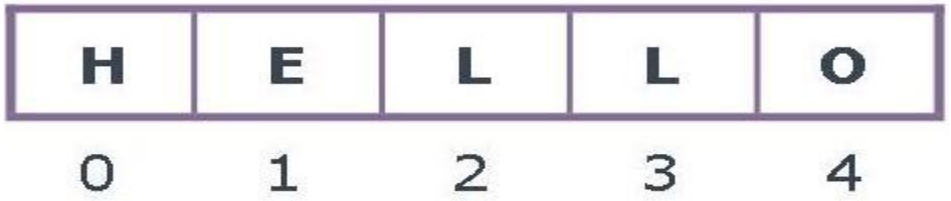
String Indexing

String Indexing in Python

- Like other programming languages, the indexing of the python strings starts from 0. **For example**, the string "HELLO" is indexed as given in the below figure.

```
str="HELLO"

str[0]=H
str[1]=E
str[4]=O
```



- Python allows negative indexing for its sequences.
- The index of -1 refers to the last item, -2 to the second last item and so on.

```
str[-1]=O
str[-2]=L
str[-4]=E
```

String Operators

String Operators in Python

+	It is known as concatenation operator used to join the strings.
*****	It is known as repetition operator. It concatenates the multiple copies of the same string.
[ ]	It is known as slice operator. It is used to access the sub-strings of a particular string.
[:]	It is known as range slice operator. It is used to access the characters from the specified range.
in	It is known as membership operator. It returns if a particular sub-string is present in the specified string.
not in	It is also a membership operator and does the exact reverse of in. It returns true if a particular substring is not present in the specified string.
r/R	It is used to specify the raw string. To define any string as a raw string, the character r or R is followed by the string. Such as "hello \n python".
%	It is used to perform string formatting. It makes use of the format specifies used in C programming like %d or %f to map their values in python.

String Operators in Python

cont…

Example: "stropdemo.py"

```
str1 = "Hello"
str2 = " World"
print(str1*3) # prints HelloHelloHello
print(str1+str2) # prints Hello world
print(str1[4]) # prints o
print(str1[2:4]) # prints ll
print('w' in str1) # prints false as w is not present in str1
print('Wo' not in str2) # prints false as Wo is present in str2.
print(r'Hello\n world') # prints Hello\n world as it is written
print("The string str1 : %s"%(str1)) # prints The string str : Hello
```

Output: **python** ifdemo.py

```
HelloHelloHello
Hello World
o
ll
False
False
Hello\n world
The string str1 : Hello
```

String Functions & Methods

String Functions & Methods in Python

- Python provides various in-built functions & methods that are used for string handling. Those are

  - len()
  - lower()
  - upper()
  - replace()
  - join()
  - split()
  - find()
  - index()
  - isalnum()
  - isdigit()
  - isnumeric()
  - islower()
  - isupper()

☞ len():

- In python, len() function returns length of the given string.

Syntax: `len(string)`

Example: **strlendemo.py**

```
str1="Python Language"
print(len(str1))
```

Output:

python strlendemo.py
15

String Functions & Methods in Python

Cont..

☞ lower ():

- In python, lower() method returns all characters of given string in lowercase.

Syntax:

```
str.lower()
```

Example: **strlowerdemo.py**

```
str1="PyTHOn"
print(str1.lower())
```

Output:

python strlowerdemo.py
python

☞ upper ():

- In python, upper() method returns all characters of given string in uppercase.

Syntax:

```
str.upper()
```

Example: **strupperdemo.py**

```
str="PyTHOn"
print(str.upper())
```

Output:

python strupperdemo.py
PYTHON

String Functions & Methods in Python

Cont..

☞ replace()

- In python, replace() method replaces the old sequence of characters with the new sequence.

Syntax: `str.replace(old, new[, count])`

Example: **strreplacedemo.py**

```
str = "Java is Object-Oriented Java"
str2 = str.replace("Java","Python")
print("Old String: \n",str)
print("New String: \n",str2)
str3 = str.replace("Java","Python",1)
print("\n Old String: \n",str)
print("New String: \n",str3)
```

Output: **python** strreplacedemo.py

```
Old String: Java is Object-Oriented and Java
New String: Python is Object-Oriented and Python
Old String: Java is Object-Oriented and Java
New String: Python is Object-Oriented and Java
```

String Functions & Methods in Python

Cont..

☞ split():

- In python, split() method splits the string into a comma separated list. The string splits according to the space if the delimiter is not provided.

Syntax: `str.split([sep="delimiter"])`

Example: **strsplitdemo.py**

```
str1 = "Python is a programming language"
str2 = str1.split()
print(str1);print(str2)
str1 = "Python,is,a,programming,language"
str2 = str1.split(sep=',')
print(str1);print(str2)
```

Output: **python** strsplitdemo.py

```
Java is a programming language
['Java', 'is', 'a', 'programming', 'language']
Java, is, a, programming, language
['Java', 'is', 'a', 'programming', 'language']
```

String Functions & Methods in Python

Cont..

☞ find():

- In python, find() method finds substring in the given string and returns index of the first match. It returns -1 if substring does not match.

Syntax: `str.find(sub[, start[,end]])`

Example: **strfinddemo.py**

```
str1 = "python is a programming language"
str2 = str1.find("is")
str3 = str1.find("java")
str4 = str1.find("p",5)
str5 = str1.find("i",5,25)
print(str2,str3,str4,str5)
```

Output:

python strfinddemo.py

7 -1 12 7

String Functions & Methods in Python

Cont..

☞ index():

- In python, index() method is same as the find() method except it returns error on failure. This method returns index of first occurred substring and an error if there is no match found.

Syntax: `str. index(sub[, start[,end]])`

Example: **strindexdemo.py**

```
str1 = "python is a programming language"
str2 = str1.index("is")
print(str2)
str3 = str1.index("p",5)
print(str3)
str4 = str1.index("i",5,25)
print(str4)
str5 = str1.index("java")
print(str5)
```

Output:

python strindexdemo.py
7
12
7
Substring not found

String Functions & Methods in Python

Cont..

☞ isalnum():

- In python, isalnum() method checks whether the all characters of the string is alphanumeric or not.
- A character which is either a letter or a number is known as alphanumeric. It does not allow special chars even spaces.

Syntax: `str.isalnum()`

Example: **straldemo.py**

```
str1 = "python"
str2 = "python123"
str3 = "12345"
str4 = "python@123"
str5 = "python 123"
print(str1. isalnum())
print(str2. isalnum())
print(str3. isalnum())
print(str4. isalnum())
print(str5. isalnum())
```

Output:

python straldemo.py
True
True
True
False
False

String Functions & Methods in Python

Cont..

☞ isdigit():

- In python, isdigit() method returns True if all the characters in the string are digits. It returns False if no character is digit in the string.

Syntax: `str.isdigit()`

Example: **strdigitdemo.py**

```
str1 = "12345"
str2 = "python123"
str3 = "123-45-78"
str4 = "IIIV"
str5 = "/u00B23" # 23
str6 = "/u00BD" # 1/2
print(str1.isdigit())
print(str2.isdigit())
print(str3.isdigit())
print(str4.isdigit())
print(str5.isdigit())
print(str6.isdigit())
```

Output:

python strdigitldemo.py
True
False
False
False
True
False

String Functions & Methods in Python

Cont..

☞ isnumeric():

- In python, isnumeric() method checks whether all the characters of the string are numeric characters or not. It returns True if all the characters are numeric, otherwise returns False.

Syntax: `str. isnumeric()`

Example: **strnumericdemo.py**

```
str1 = "12345"
str2 = "python123"
str3 = "123-45-78"
str4 = "IIIV"
str5 = "/u00B23" # 23
str6 = "/u00BD" # 1/2
print(str1.isnumeric())
print(str2.isnumeric())
print(str3.isnumeric())
print(str4.isnumeric())
print(str5.isnumeric())
print(str6.isnumeric())
```

Output:

python strnumericldemo.py
True
False
False
False
True
True

String Functions & Methods in Python

Cont..

☞ islower():

- In python, islower() method returns True if all characters in the string are in lowercase. It returns False if not in lowercase.

Syntax: `str.islower()`

Example: **strlowerdemo.py**

```
str1 = "python"
str2="PytHOn"
str3="python3.7.3"
print(str1.islower())
print(str2.islower())
print(str3.islower())
```

Output:

python strlowerldemo.py
True
False
True

String Functions & Methods in Python

Cont..

☞ isupper():

- In python string isupper() method returns True if all characters in the string are in uppercase. It returns False if not in uppercase.

Syntax: `str.isupper()`

Example: **strupperdemo.py**

```
str1 = "PYTHON"
str2="PytHOn"
str3="PYTHON 3.7.3"
print(str1.isupper())
print(str2.isupper())
print(str3.isupper())
```

Output:

python strupperldemo.py
True
False
True