Data Types in Python

=>The purpose of Data Types is that "To allocate sufficient amount of memory space in main memory for storing the Literals or values ".

I. Fundamental Category Data Types.

1. int (42)
2. float (10.22)
3. bool (True/False)
4. complex (3+4j)
5. str ("Python")

II. Advance Data Types or Data structure or container.

1. list [1,2,3]
2. tuple (1,2,3)
3. set {1,2,3}
4. dictionary {1:"audi",2:"benz"}
5. range range(1,10)

1. int

=>'int' is one of the pre-defined class and treated as Fundamental Data Types.

=>The purpose of int data type is that " To store Integral values or Integer data or whole numbers (Numbers without decimal values).".

Examples: stno, empno, adno, htno, acno

=>int data type can also used for representing or storing Different Types of Number Systems

Examples:

```
Python Instructions                          Output
-----------------------------------   --------------------
>>> a=10
>>> print(a)----------------------------------10
>>> type(a)---------------------------------- <class 'int'>
>>> id(a)------------------------------------ 2005677310480
>>> print(a,type(a),id(a))-------------------10  <class 'int'>
                                                2005677310480
>>> a=12
```

```
>>> b=34
>>> c=a+b
>>> print(a, type(a))-----------------------12 <class 'int'>
>>> print(b, type(b))-----------------------34 <class 'int'>
>>> print(c, type(c))-----------------------46 <class 'int'>

>>> int=45   # Here   int is not a keyword and more over all
    class names can be used as Variable Names
>>> print(int, type(int))-------------------45 <class 'int'>
```

==========x==========

2. float data type

=>'float' is one of the pre-defined class and treated as Fundamental data type.
=>The purpose of float data type is that " To store Floating values or Number with decimal places".
Example: Percentage, empcomm, Tax...etc.

=>This data also supports Scientific Notation and general format is MANTISA e EXPONENT and whose equal floating point value = MANTISA x 10 to the power of EXPONENT.
=>The advantage of Scientific Notation is that to minimize the memory space.
=>This data type never supports Binary, Octal and Hexa Decimal Number System.

Examples:

```
>>> a=1.2
>>> print(a,type(a))------------------1.2 <class 'float'>
>>> a=0.999
>>> print(a,type(a))------------------0.999 <class 'float'>
>>> a=10
>>> b=1.2
>>> c=a+b
>>> print(a,type(a))------------------10 <class 'int'>
>>> print(b,type(b))------------------1.2 <class 'float'>
>>> print(c,type(c))------------------11.2 <class 'float'>
```

```
>>> a=3e2
>>> print(a)--------------------------300.0
>>> b=10e-2
>>> print(b,type(b))------------------0.1 <class 'float'>
```

```
>>> a=0.00000000000000000000000000000001
>>> print(a,type(a))----------------1e-37 <class 'float'>
-------------------------------------------------------------------
>>> a=0b1010.0b1010---------SyntaxError: invalid decimal literal
>>> a=0o12.0o22-------------SyntaxError: invalid decimal literal
>>> a=0x23.0x45-------------SyntaxError: invalid decimal literal

================================x================================
====================================================================
                              3. bool
====================================================================
=>'bool' is one of the pre-defined class and treated as
  Fundamental Data Type.
=>The purpose of bool data type is that "To store True and False
  Values (Logical Values). "
=>True and False are called Key Words and they are treated as
  Values for bool data type.
=>Internally, the True is treated as 1 and False is treated as
  0.
----------------------
Examples:
----------------------
>>> a=True
>>> b=False
>>> print(a,type(a))-----------------True <class 'bool'>
>>> print(b,type(b))-----------------False <class 'bool'>
--------------------------------------------------------------------
----------------------
>>> a=true---------------NameError: name 'true' is not defined.
>>> True=10-------------SyntaxError: cannot assign to True
--------------------------------------------------------------------
----------------------
>>> a=True
>>> b=False
>>> print(a+b)----------------------------1
>>> print(a+a)----------------------------2
>>> print(2+True+False)-------------------3
>>> print(0b1010*True)--------------------10
>>> print(0b1010+True)--------------------11
>>> print(0b1010*False+True)--------------1
>>> print(0b1010+True*1.2)----------------11.2

================================x================================
```

4. complex

=>'complex' one of the pre-defined class and treated as Fundamental Data Types

=>The purpose of complex data type is that to store Complex Values.

=>The general format of complex values is given bellow.

a+bj OR a-bj

=>Here 'a' is called REAL Part

=>Here 'b' is called IMAGINARY Part

=>Here 'j' represent sqrt(-1)

=>Internally the Real and Imaginary Parts are treated as floating point values.

=>To retrieve real and imaginary part of Complex Number, we use two pre-defined attributes and they present in complex object. They are

a) real----> complexobj.real---->Gives real Part of Complex Object

b) imag---> complexobj.imag---->Gives Imaginary Part of Complex Object

=>On complex data, we can perform Addition, Subtraction, Multiplication etc.

Examples

```
>>> a=2+3j
>>> print(a,type(a))---------------(2+3j) <class 'complex'>
>>> b=2-4j
>>> print(b,type(b))---------------(2-4j) <class 'complex'>
>>> c=-2-4j
>>> print(c,type(c))---------------(-2-4j) <class 'complex'>
```

```
>>> a=1.2+3.4j
>>> print(a,type(a))---------------(1.2+3.4j) <class 'complex'>
>>> b=-1.3-4.5j
>>> print(b,type(b))---------------(-1.3-4.5j) <class 'complex'>
>>> c=2+3.4j
>>> print(c,type(c))---------------(2+3.4j) <class 'complex'>
```

```
>>> a=4j
>>> print(a,type(a))---------------4j <class 'complex'>
>>> b=-4.5j
>>> print(a,type(a))---------------4j <class 'complex'>
>>> print(b,type(a))---------------(-0-4.5j) <class 'complex'>
```

```
>>> a=5.6j
>>> print(a,type(a))---------5.6j <class 'complex'>
>>> print(a.real)------------0.0
>>> print(a.imag)------------5.6
>>> print(a.imaginary)-------AttributeError: 'complex'
                              object has no attribute 'imaginary'
```

```
>>> a=2+3j
>>> b=2+4j
>>> print(a,type(a))-------------(2+3j) <class 'complex'>
>>> print(b,type(b))-------------(2+4j) <class 'complex'>
>>> print(a+b)-------------------(4+7j)
>>> print(a-b)-------------------1j
>>> print(a*b)-------------------(-8+14j)
```

```
>>> print((2+3j).real)--------------2.0
>>> print((2+3j).imag)--------------3.0
```

```
>>> a=True+Falsej  ---------------NameError: name 'Falsej' is
                                   not defined'False'?
>>> a=0b1010+4j
>>> print(a,type(a))-------------------(10+4j) <class 'complex'>
```

==========x==========

==========

1. str

==========

=>The purpose of str data type is that "To store String data or text data or Alphanumeric data or numeric data or any type data within double Quotes or single quotes or triple double quotes and triple single quotes."

=>Def. of str:

=>str is a collection of Characters or Alphanumeric data or numeric data or any type data enclosed within double Quotes or single quotes or triple double quotes and triple single quotes."

=>Types of Str data

=>In Python Programming, we have two types of Str Data. They are

1. Single Line String Data
2. Multi Line String Data

---

1. Single Line String Data:

```
=>Syntax1:-       varname=" Single Line String Data "
                                  (OR)
=>Syntax2:-       varname=' Single Line String Data '
```

=>With the help double Quotes (" ") and single Quotes (' ') we can store single line str data only but not possible to store multi line string data.

---

2 Multi Line String Data:

```
=>Syntax1:-       varname=" " " String Data1
                                  String Data2
                                  ----------------
                                  String data-n " " "

                                  (OR)
=>Syntax2:-      varname=' ' ' String Data1
                                  String Data2
                                  ----------------
                                  String data-n ' ' '
```

=>With the help triple double Quotes (" " " " " ") and Triple single Quotes (' ' ' ' ' ') we can store single line str data multi-line string data.

```
>>> s1="Python Programming"
>>> print(s1,type(s1))----------Python Programming <class 'str'>
>>> s2='Java Programming'
>>> print(s2,type(s2))----------Java Programming <class 'str'>
>>> addr1="Guido Van Rossum-----SyntaxError: unterminated string
                                literal (detected at line 1)
>>> addr1='Guido Van Rossum-----SyntaxError: unterminated
                              string literal (detected at line 1)
>>> addr1="""Guido Van Rossum
             ... FNO:3-4, Red Sea Side
             ... Python Software Foundation
             ... Nether Lands-56"""
>>> print(addr1,type(addr1))
                          Guido Van Rossum
                          FNO:3-4, Red Sea Side
                          Python Software Foundation
                          Nether Lands-56 <class 'str'>
```

```
>>> addr2='''James Gosling
             ... HNO:13-45, Hill Side
             ... Sun Mocro System INC
             ... USA-567'''
>>> print(addr2,type(addr2))
                  James Gosling
                  HNO:13-45, Hill Side
                  Sun Mocro System INC
                  USA-567 <class 'str'>
-------------------------------------------------------------
>>> s3="""Python Programming"""
>>> print(s3,type(s3))--------Python Programming <class 'str'>
>>> s4='''Data Scienece '''
>>> print(s4,type(s4))-----------Data Scienece  <class 'str'>
------------------------
>>> c1="A"
>>> print(c1,type(c1))----------A <class 'str'>
>>> c2='A'
>>> print(c2,type(c2))----------A <class 'str'>
>>> c3="""A"""
>>> print(c3,type(c3))----------A <class 'str'>
>>> c4='''A'''
>>> print(c4,type(c4))----------A <class 'str'>
>>> s5="Python3.10.5"
>>> print(s5,type(s5))----------Python3.10.5 <class 'str'>
>>> s6="ABCDabcbd45678#$%^&*_kvr"
>>> print(s6,type(s6))----------ABCDabcbd45678#$%^&*_kvr <class
                                                        'str'>

================================x================================
```

==========

=>Length of string (len()) :

==========

=>The len() function returns the number of items in an object.
=>When the object is a string, the len() function returns the number of characters in the string.
=>len() is not applicable for int, float, bool, complex data Types.
=>If we assign int, float, bool, complex data types in len(), then we get TypeError.

=>Syntax : - len(object)

=>Here "object" must be a sequence or a collection.

=>Examples

```
>>>mylist = "Hello"
>>>x = len(mylist)
>>>print(x)----------------5

>>>mylist = "Hello World"
>>>x = len(mylist)
>>>print(x)----------------11

>>>a=10
>>>len(a)----------------- TypeError: object of type 'int' has no len()
```

=>Indexing

=>The process of obtaining single character from given str object is called Indexing.

=>Syntax:- strobj[Index]

=>Here Index can be either +ve or -ve.
=>If we enter valid Index then we get a Character from str obj.
=>If we enter invalid Index then we get IndexError.

```
-------------------------------------------------------------------
=>Examples:
-------------------------------------------------------------------
>>> s="PYTHON"
>>> print(s,type(s))--------------PYTHON <class 'str'>
>>> print(s[0])------------P
>>> print(s[2])------------T
>>> print(s[4])------------O
>>> print(s[5])------------N
>>> print(s[-1])-----------N
>>> print(s[-6])-----------P
>>> print(s[-2])-----------O
>>> print(s[-3])-----------H
>>> print(s[-30])----------IndexError: string index out of range
>>> print(s[2])------------T
>>> print(s[12])-----------IndexError: string index out of range
-------------------------------------------------------------------
>>> "Java"[2]--------------'v'
>>> "Java"[-1]-------------'a'
>>> "Java"[-3]-------------'a'
>>> "Java"[-4]-------------'J'
>>> "Java"[-4+2]-----------'v'
>>> "Java"[True]-----------'a'
>>> "Java"[False]----------'J'
>>> "1234"[10-8]-----------'3'
>>> ""[3]-------------IndexError: string index out of range
```

```
-------------------------------------------------------------------
=>Slicing Operations
-------------------------------------------------------------------
=>The Process of obtaining range of characters or sub string
  from given string is called Slicing.
=>We can perform slicing Operations with 5 Syntaxes. They are
-------------------------------------------------------------------
-------------------------------------------------------------------
Syntax1:      1)strobj[BEGIN INDEX:END INDEX]
-------------------------------------------------------------------
=>This syntax obtains range of Characters from strobj from BEGIN
  INDEX to END INDEX-1 provided BEGIN INDEX<END INDEX otherwise
  we never get any output. (space or ' ' is a result)
------------------
Examples:
------------------
>>> s="PYTHON"
>>> print(s,type(s))---------------PYTHON <class 'str'>
>>> s[0:4]------------------------'PYTH'
```

```
>>> s[3:6]-------------------------'HON'
>>> s[2:6]-------------------------'THON'
>>> s[4:6]-------------------------'ON'
>>> s[3:2]-------------------------  ''
>>> s[-6:-3]-----------------------'PYT'
>>> s[-4:-1]-----------------------'THO'
>>> s[-3:-1]-----------------------'HO'
>>> s[-5:-1]-----------------------'YTHO'
>>> s[2:-1]------------------------'THO'
>>> s[0:-2]------------------------'PYTH'
>>> s[2:-2]------------------------'TH'
>>> s[20:-20]----------------------''
>>> s[0:-1]------------------------'PYTHO'
>>> s[0:25]------------------------'PYTHON'
```

---

2) strobj[Beg Index:]

=>In This syntax, we are specifying Begin Index and not specifying End Index.
=>If we don't specify End Index then EndIndex taken by PVM as len(strobj)-1 OR EndIndex Taken by PVM as Last Character Index

---

=>Examples:

```
>>> s="PYTHON"
>>> s[2:]---------------'THON'
>>> s[4:]---------------'ON'
>>> s[-2:]--------------'ON'
>>> s[-6:]--------------'PYTHON'
>>> s[0:]---------------'PYTHON'
>>> s-------------------'PYTHON'
>>> s[-4:]--------------'THON'
```

---

3) strobj[: EndIndex]

=>In This syntax, we are not specifying Begin Index and specifying End Index.
=>If we don't specify Begin Index then BegIndex taken by PVM as First Character Index i.e. either 0 or -(len(strobj))

=>Examples:

```
>>> s="PYTHON"
```

```
>>> print(s)--------------------PYTHON
>>> s[:4]-----------------------'PYTH'
>>> s[:-4]----------------------'PY'
>>> s[:-2]----------------------'PYTH'
>>> s[:6]-----------------------'PYTHON'
>>> s[:2]-----------------------'PY'
```

---

4) strobj[:]

=>In This syntax we are not specifying Both BeginIndex and EndIndex

=>If we don't specify Both BeginIndex and EndIndex then PVM Takes BegIndex as First Character Index i.e. either 0 or -(len(strobj)) and EndIndex as Last Character Index OR len(strobj)-1

=>Examples:

```
>>> s="PYTHON"
>>> print(s)--------------------------------PYTHON
>>> s[:]-----------------------------------'PYTHON'
>>> s[0:]----------------------------------'PYTHON'
>>> s[:6]----------------------------------'PYTHON'
>>> s[-6:]---------------------------------'PYTHON'
>>> s[0:6]---------------------------------'PYTHON'
>>> s[0:1]---------------------------------'P'
>>> s[-6:-5]-------------------------------'P'
```

NOTE: All the above Syntaxes are extracting the data from str object in forward direction by maintain 1 as step value.

---

5) strobj[BegIndex:EndIndex:STEP]

Rule1: Here BegIndex and EndIndex and STEP can be either +ve or -ve

Rule2: If the value of STEP is +VE then PVM Takes the characters from str obj from BegiIndex to EndIndex-1 in FORWARD Direction provided BeginIndex<EndIndex.

Rule3: If the value of STEP is -VE then PVM Takes the characters from str obj from BegiIndex to EndIndex+1 in BACKWARD Direction provided BeginIndex>EndIndex.

Rule4: In Forward Direction if endIndex is 0 then we never get any output.

Rule5: In Backward Direction if endIndex is -1 then we never get any output.

=>Example:

```
>>> s="PYTHON"
>>> print(s)----------------------PYTHON
>>> s[::]------------------------'PYTHON'
>>> s[::2]-----------------------'PTO'
>>> s[::3]-----------------------'PH'
>>> s[0:6:1]---------------------'PYTHON'
>>> s[2:4:2]---------------------'T'
>>> s[-6:-1:2]-------------------'PTO'
>>> s[2:5:2]---------------------'TO'
>>> s[-6:-2:2]-------------------'PT'

>>> s="PYTHON"
>>> print(s)----------------------PYTHON
>>> s[2:5:2]---------------------'TO'
>>> s[-6:-2:2]-------------------'PT'
-------------------------
>>> s="PYTHON"
>>> s[2:4:-1]--------------------'   '  No output
>>> s[4:2:-1]--------------------'OH'
>>> s[1:5:-1]--------------------  '      ' No output
>>> s[5:1:-1]--------------------'NOHT'
>>> s[5:1:-2]--------------------'NH'
>>> s[::-2]----------------------'NHY'
>>> s[5::-2]---------------------'NHY'
>>> s[::-1]----------------------'NOHTYP'
>>> s[-6:-1:-1]---------------   '     '
>>> s[-1:-6:-1]--------------- 'NOHTY'
>>> s[-1:-7:-1]---------------  'NOHTYP'
>>> s[2:0:1]------------------ '    '  No Output  (Rule-4)
>>> s[2:-1:-1]----------------'    '  No Output (Rule-5)
```

MiSc Examples:

```
>>> s="PYTHON PROGRAMMING"
>>> s[::-3]----------------------'GMRRNT'
>>> s[::-2]----------------------'GIMROPNHY'
>>> s[::-1]----------------------'GNIMMARGORP NOHTYP'
>>> res=s[0]+s[5]+s[11]
>>> print(res)----------------------PNR
```

```
>>> res=s[0:3]+s[7:10]
>>> print(res)----------------------PYTPRO
>>> "racecar"[::-1]-----------------'racecar'
>>> "racecar"[::-1][::2]------------'rccr'
>>> "racecar"[::-1][::2][::3]-----------'rr'
```

```
  strobj[Beg:End]
  strobj[Beg: ]
  strobj[ : End]
  strobj[ : ]
  strobj[Beg:End:Step]
```

=>capitalize()

=>This Function is used for capitalizing the first letter First word of a given Sentence only.

=>Syntax: strobj.capitalize()

(OR)

strobj=strobj.capitalize()

Examples:

```
>>> s="python"
>>> print(s,type(s))-----------------python <class 'str'>
>>> s.capitalize()-------------------'Python'

>>> s="python is an oop lang"
>>> print(s,type(s))-----------------python is an oop lang
                                          <class 'str'>

>>> s.capitalize()-------------------'Python is an oop lang'
```

```
>>> s="python"
>>> print(s,type(s))-----------------python <class 'str'>
>>> s.capitalize()-------------------'Python'
>>> print(s,type(s))-----------------python <class 'str'>
>>> s=s.capitalize()
>>> print(s,type(s))-----------------Python <class 'str'>
```

```
-----------------------------------------------------------------
-----------------------------------------------------------------
=>title():
-----------------------------------------------------------------
-----------------------------------------------------------------
=>This is used for obtaining Title Case of a Given Sentence (OR)
  Making all words First Letters are capital.

=>Syntax:          s.title()
                   (OR)
                   s=s.title()

------------------
Examples:
------------------
>>> s="python"
>>> print(s,type(s))---------python <class 'str'>
>>> s.capitalize()-----------'Python'
>>> s.title()----------------'Python'
-----------------------------------------------------------
>>> s="python is an oop lang"
>>> print(s,type(s))---------python is an oop lang <class 'str'>
>>> s.capitalize()-----------'Python is an oop lang'
>>> s.title()----------------'Python Is An Oop Lang'
>>> print(s)----------------- python is an oop lang
>>> s=s.title()
>>> print(s)----------------- Python Is An Oop Lang
-----------------------------------------------------------------
-----------------------------------------------------------------
=>index()
-----------------------------------------------------------------
-----------------------------------------------------------------
=>This Function obtains Index of the specified Value
=>If the specified value does not exist then we get ValueError

=>Syntax:      strobj.index(Value)
=>Syntax:      indexvalue=strobj.index(value)

Examples:
-----------------
>>> s="python"
>>> s.index("p")----------------0
>>> s.index("y")----------------1
>>> s.index("o")----------------4
```

```
>>> s.index("n")----------------5
>>> s.index("K")----------------ValueError: substring not found
NOTE:
------------
=>enumerate() is one the general function, which is used for
  finding Index and Value of any Iterable object.

>>> for i,v in  enumerate(s):
...     print("Index:{} and Value:{}".format(i,v))

---------------
OUTPUT
---------------
                    Index:0 and Value:p
                    Index:1 and Value:y
                    Index:2 and Value:t
                    Index:3 and Value:h
                    Index:4 and Value:o
                    Index:5 and Value:n
-----------------------------------------------
>>> lst=[10,"Rossum",23.45,True]
>>> for i,v in  enumerate(lst):
...     print("Index:{} and Value:{}".format(i,v))

---------------
OUTPUT
---------------
                    Index:0 and Value:10
                    Index:1 and Value:Rossum
                    Index:2 and Value:23.45
                    Index:3 and Value:True
-----------------------------------------------------------------
-----------------------------------------------------------------
=>upper()
-----------------------------------------------------------------
-----------------------------------------------------------------
=>It is used for converting any type of Str Data into Upper
  Case.
=>Syntax:-      strobj.upper()
                    OR
                strobj=strobj.upper()
-----------------
Examples:
-----------------
>>> s="python"
```

```
>>> print(s)--------------python
>>> s.upper()-------------'PYTHON'
>>> s="python is an oop lang"
>>> print(s)----------------python is an oop lang
>>> s.upper()--------------'PYTHON IS AN OOP LANG'
>>> s="Python IS an OOP lang"
>>> print(s)----------------Python IS an OOP lang
>>> s.upper()---------------'PYTHON IS  AN OOP LANG'
>>> s="AbCdEf"
>>> print(s)----------------AbCdEf
>>> s.upper()---------------'ABCDEF'
>>> s="PYTHON"
>>> print(s)----------------PYTHON
>>> s.upper()---------------'PYTHON'
>>> s="123"
>>> print(s)----------------123
>>> s.upper()---------------'123'
```

=>lower()

=>It is used for converting any type of Str Data into lower Case.

=>Syntax:- strobj.lower()

OR

strobj=strobj.lower()

Examples:

```
>>> s="Data Science"
>>> print(s)-------------Data Science
>>> s.lower()------------'data science'
>>> s="python"
>>> print(s)-------------python
>>> s.lower()------------'python'
>>> s="PYTHON"
>>> print(s)-------------PYTHON
>>> s.lower()------------'python'
>>> s="PYThon"
>>> print(s)-------------PYThon
>>> s.lower()------------'python'
```


=>isupper()

=>This Function returns True provided the given str object data
  is purely Upper Case otherwise it returns False.

=>Syntax: strobj.isupper()

Examples:

```
>>> s="PYTHON"
>>> s.isupper()-----------True
>>> s="python"
>>> s.isupper()-----------False
>>> s="Python"
>>> s.isupper()-----------False
>>> s="PYThon"
>>> s.isupper()-----------False
>>> s="123"
>>> s.isupper()-----------False
>>> s="%$#^&@"
>>> s.isupper()-----------False
```


=>islower()

=>This Function returns True provided the given str object data
  is purely lower Case otherwise it returns False.

=>Syntax: strobj.islower()

Examples:

```
>>> s="pythopn"
>>> s.islower()------------True
>>> s="pythOn"
>>> s.islower()------------False
>>> s="PYTHON"
>>> s.islower()------------False
>>> s="123"
>>> s.islower()------------False
```


=>isalpha()

=>This Function returns True provided str object contains Purely
  Alphabets otherwise returns False.

=>Syntax: strobj.isalpha()

Examples:

```
>>> s="Ambition"
>>> s.isalpha()--------------True
>>> s="Ambition123"
>>> s.isalpha()--------------False
>>> s="1234"
>>> s.isalpha()--------------False
>>> s="    "
>>> s.isalpha()--------------False
>>> s="#$%^@"
>>> s.isalpha()--------------False
>>> s="AaBbZz"
>>> s.isalpha()--------------True
```


=>isdigit()

=>This Function returns True provided given str object contains
purely digits otherwise returns False

=>Syntax: strobj.isdigit()

Examples:

```
>>> s="python"
>>> s.isdigit()----------------False
>>> s="python123"
>>> s.isdigit()----------------False
>>> s="123"
>>> s.isdigit()----------------True
```

```
>>> s="123 456"
>>> s.isdigit()-----------------False
>>> s="1_2_3"
>>> s.isdigit()-----------------False
>>> s="123KV"
>>> s.isdigit()-----------------False
```

=>isalnum()

=>This Function returns True provided str object contains either Alpabets OR Numerics or Alpha-Numerics only otherwise It returns False.

=>Syntax: strobj. isalphanum()

=>Examples:

```
>>> s="python310"
>>> s.isalnum()-----------------True
>>> s="python"
>>> s.isalnum()-----------------True
>>> s="310"
>>> s.isalnum()-----------------True
>>> s="$python310"
>>> s.isalnum()-----------------False
>>> s="python 310"
>>> s.isalnum()-----------------False
>>> s="$python3.10"
>>> s.isalnum()-----------------False
>>> s="python3.10"
>>> s.isalnum()-----------------False
```

=>isspace()

=>This Function returns True provided str obj contains purely space otherwise it returns False.

=>Syntax: strobj.isspace()

```
----------------------
Examples:
----------------------
>>> s="  "
>>> s.isspace()-------------------------True
>>> s=""
>>> s.isspace()-------------------------False
>>> s="python Prog"
>>> s.isspace()-------------------------False
>>> s="Prasana Laxmi"
>>> s.isspace()-------------------------False
>>> s.isalpha()-------------------------False
>>> s.isalpha() or s.isspace()-----------False
```

=>split()

=>This Function is used for splitting the given str object data into different words base specified delimter (- _ # % ^ ^ , ; .... etc)

=>The default delimter is space

=>The Function Returns Splitting data in the form of list object

```
=>Syntax:    strobj.split("Delimter")
                    (OR)
             strobj.split()
                    (OR)
             listobj=  strobj.split("Delimter")
                    (OR)
             listobj=strobj.split()
```

```
----------------
Examples:
----------------
>>> s="Python is an oop lang"
>>> print(s)---------------Python is an oop lang
>>> s.split()--------------['Python', 'is', 'an', 'oop', 'lang']
>>> len(s.split())---------5
>>> x=s.split()
>>> print(x,type(x))-------['Python', 'is', 'an', 'oop', 'lang']
<class 'list'>
>>> len(x)-----------------5
>>> s="12-09-2022"
>>> print(s)---------------12-09-2022
>>> s.split("-")-----------['12', '09', '2022']
```

```
>>> s="12-09-2022"
>>> dob=s.split("-")
>>> print(dob,type(dob))---------['12', '09', '2022']
                                  <class 'list'>
>>> print("Day",dob[0])----------Day 12
>>> print("Month ",dob[1])-------Month  09
>>> print("Year ",dob[2])--------Year  2022
-----------------------------------------------------
>>> s="Apple#Banana#kiwi/Guava"
>>> words=s.split("#")
>>> print(words)-----------['Apple', 'Banana', 'kiwi/Guava']
>>> words=s.split("/")
>>> print(words)-----------['Apple#Banana#kiwi', 'Guava']
```

=>join():

=>This Function is used for combining or joining list of values from any Iterable object

=>Syntax: strobj.join(Iterableobject)

Examples:

```
>>> lst=["HYD","BANG","AP","DELHI"]
>>> print(lst,type(lst))----------['HYD', 'BANG', 'AP', 'DELHI']
                                  <class 'list'>
>>> s=""
>>> s.join(lst)-------------------'HYDBANGAPDELHI'
>>> s=" "
>>> s.join(lst)-------------------'HYD BANG AP DELHI'
-----------------------------------------------
>>> t=("Rossum", "is", "Father", "of" ,"Python")
>>> print(t,type(t))---('Rossum', 'is', 'Father', of', 'Python')
                       <class 'tuple'>
>>> k=" "
>>> k.join(t)----------'Rossum is Father of Python'
>>> t=("Rossum","is", "Father", "of" ,"Python")
>>> k=" "
>>> k.join(t)----------'Rossum is Father of Python'
```

==========x==========

Examples:

```
#Wapp which will convert Upper letters into Lower letters and
 Lower letters into Upper letters wise versa.
#Sample.py
s="AmBiTiOn"
print("Given Data:",s)
for ch in s:
      if (ch.isupper()):
            lc=ch.lower()
            print(lc,end="")
      elif(ch.islower()):
            uc=ch.upper()
            print(uc,end="")
```

==========x==========