This file contains list of all functions and their descriptions with example.

LIST OF FUNCTIONS

- **Date_module.py(date manipulation functions)**
    - dateinfo()
    - weekday()
    - totimestamp()
    - todate()
    - to_utc()
    - from_utc()
    - c_time()
    - addtime()
    - removetime()
- **Date_module(DateFormatCode).py(date format manipulation function)**
    - Str_f_time()

- **Math_module.py(number manipulation function)**
    - Pow()
    - Sqrt()
    - Dist()
    - Factorial()
    - Combination()
    - Permutation()
    - Complex()
    - Real()
    - Imag()
    - Abs()
    - Compare()
    - Gcd()
    - Sum()
    - Radians()
    - Degrees()
    - CheckPrime()
    - Floor()
    - Ceil()
    - Bin()
    - Oct()
    - Hex()

- **String_module.py(string manipulation functions)**
    - uppercase()
    - lowercase()
    - legth()
    - strcapitalize()
    - strcount()
    - strfind()
    - strcenter()
    - stralphanum()

- strjoin()
- strsplit()
- strtitle()
- strreplace()
- strformat()
- strisdigit()
- strisalpha()
- strstrip()
- strpartition()

- **Tuple_operation.py(tuple manipulation functions)**
  - index()
  - count()

- **Array_module.py(array manipulation functions)**
  - Append()
  - Pop()
  - Remove()
  - Insert()
  - Count()
  - Extend()
  - ToList()
  - FromList()
  - Index()
  - Reverse()
  - FromUnicode()
  - ToUnicode()

- **List_operation.py(list manipulation functions)**
  - append()
  - insert()
  - remove()
  - pop()
  - reverse()
  - len()
  - count()
  - copy()
  - index()

ALL FUNCTION'S DESCRIPTION WITH EXAMPLES

//Date_module.py(date manipulation functions)//

#] dateinfo(dateobj,param):

Returns individual entities on date like year,month,day,hour,min,sec,msec.Needs two parameters i.e. datetime or date object and entity name.

Eg.

```
y=datetime.datetime(2020, 7, 1,0,0,1,1)
dateinfo(y,"year")
o/p: 2020
```

#] weekday(dateobj):

Returns day of the week,for Monday returns 0,for tuesday returns 1,respectively.Needs one parameter ie. Datetime object.

Eg.

```
y=datetime.datetime(2020, 7, 1)
weekday(y)
o/p: 2
```

#] totimestamp(dateobj):

Returns timestamp.Needs one parameter ie. Datetime object.

Eg.

```
y=datetime.datetime(2020, 12, 30, 23, 0, 1,1)
totimestamp(y)
o/p: 1609455601.000001
```

#] todate(timestamp):

Returns datetime object.Needs one parameter ie. Timestamp obtained from totimestamp() function.

Eg.

```
todate(1609455601.000001)
o/p: datetime.datetime(2020, 12, 30, 23, 0, 1, 1)
```

#] to_utc(dateobj):

Returns datetime object converted to utc standard.Needs one parameter ie.date time object

Eg.

```
y=datetime.datetime(2020, 1, 1, 5, 29, 9, 976549)
to_date(y)
o/p: datetime.datetime(2019, 12, 31, 23, 59, 9)
```

#] from_utc(dateobj):

Returns datetime object converted from utc standard date to normal date.Needs one parameter ie.datetime object

Eg.

```
y=datetime.datetime(2019, 12, 31, 23, 59, 9)
from_utc(y)
O/p: datetime.datetime(2020, 1, 1, 5, 29, 9)
```

#] c_time(dateobj):

Returns date string in well formatted form.Needs one parameter ie.datetime object.It works same as in built ctime() function.

Eg.

```
y=datetime.datetime.now()
c_time(y)
o/p: 'Thu Jul  16 20:42:24 2020'
```

#] addtime(dateobj,years=0,months=0,days=0,hours=0,minutes=0,seconds=0):

Returns datetime object with added time. Needs many parameters ie. Datetime object and others para meters to add in date like years, months, days, hours, minutes, seconds. We can send any no. of parameters as required.

Eg.

```
y=datetime.datetime(2020, 2, 29, 0, 0, 0)
addtime(y,hours=24)
O/p: datetime.datetime(2020, 3, 1, 0, 0)
```

#] removetime(dateobj,years=0,months=0,days=0,hours=0,minutes=0,seconds=0):

Returns datetime object with removed time.Needs many parameters ie. Datetime object and others parameters to add in date like years,months,days,hours.minutes,seconds.We can send any no. of parameters as required.

Eg.

```
y=datetime.datetime(2020, 3, 1, 0, 0, 1)
removetime(y,minutes=1)
O/p: datetime.datetime(2020, 2, 29, 23, 59, 1)
```

//Date_module(DateFormatCode).py(date format manipulation function)//

#] str_f_time(dateobj,string):

Returns a date string in a formatted manner.Needs two parameters ie. Datetime object and string to specify the format for the date.Works same as strftime function.

Eg.

```
date=datetime.datetime(2018, 2, 8,10,22 ,21,100)

str_f_time(date,"%c \n%X %x \n %A %j")
o/p:
Thu Feb  8 10:22:21 2018
10:22:21 02/08/18
Thursday 039

str_f_time(date,"%w     %d \t %B %p")
o/p:
4     8        February AM

str_f_time(date,"%I %m %H %Y ")
o/p: 10 02 10 2018
```

// Math_module.py(number manipulation function)//

#] Pow(base,exponent):

Return result of base to the power exponent.

Eg :

Pow(4,5)
o/p :
1024

#] Sqrt(num):

Return the square root of num.Num can be either int or float.

Eg:

Sqrt(16)
o/p :
4.0

#] Dist(Point1,Point2):

Return the Euclidean distance between two points.Both points must have same number of dimensios(List size must be same).

Eg:

Dist([2,4,5,1,4],[4,1,5,1,4])
o/p:
3.6055512754639896

#] Factorial(num):

Return the factorial of num.Factorial function not valid for negative value.

Eg:

Factorial(8)
o/p:
40320

#] Combination(n,k):

Return number of ways to choose “k” items from “n” items, where order of items doesn’t matter.

Eg:

Combination(23,14)
o/p:
817190

#] Permutation (n,k):

Return number of ways to arrange “k” items from “n” items in a particular order.

Eg:

Permutation(21,10)
o/p:
1279935820800

#] Complex(real,img):

Return complex form of real and imaginary number in a+bj format (str).

Eg:

Complex(5,-4.58)

o/p:

5-4.58j

#] Real(complexNumber):

Return float type value of real number from complex number.

Eg:

complexNum=Complex(5,-4.58)

Real(complexNum)

o/p:

5.0

#] Imag(complexNumber):

Return float type value of imaginary number from complex number.

Eg:

complexNum=Complex(5,-4.58)

Imag(complexNum):

o/p:

-4.58

#] Abs(num):

Return absolute value of num or expression.

Eg:

Abs(-56.454)

o/p:

56.454

#] Compare(num1,num2):

Return 0 if both values are equal ,1 if first value is greater than second else -1.

Eg:

Compare(43.22,43.22)

o/p:

0

#] Gcd(num1,num2):

Return greatest common divisor of two numbers.

Eg:

Gcd(70,28)

o/p:

14

#] Sum(iterableObject):

Return sum of iterable object.

Eg:

Sum([2,3,5,3,6.5])

o/p: 19.5

#] Radians(degreeValue):

Convert degree to radian.

Eg:

Radians(60)

o/p:

1.0471975511965976

#] Degrees(radianValue):

Convert radian to degree.

Eg:

Degrees(70)

o/p:

4010.7045659157625

#] CheckPrime(num):

Return Boolean value (True) if number is prime else return False.

Eg:

CheckPrime(37)

o/p:

True

#] Floor(num):

Return greatest integer less than or equal to given number.

Eg:

Floor(5.3)

o/p:

5

#] Ceil(num):

Return smallest integer greater than or equal to given number.

Eg:

Ceil(22.5)

o/p:

23

#] Bin(num):

Convert integer value into binary string.

Eg:

Bin(58)

o/p:

111010

#] Oct(num):

Convert integer value into octal string.

Eg:

Oct(95)

o/p:

137

#] Hex(num):

Convert integer value into Hexadecimal string.

Eg:

Hex(69754)

o/p:

1107A

//String_module.py(string manipulation functions)//

#] uppercase(string):

Returns a string in which all letters are converted to uppercase letters, requires one parameter ie. a string.

Eg.

uppercase("xyz")

o/p: XYZ

#] lowercase(string):

Returns a string in which all letters are converted to lowercase letters, requires one parameter ie. a string.

Eg.

uppercase("XyZ")

o/p: xyz

#] length(string):

Returns length of string,requires one parameter ie.a string

Eg.

length("abcdef")

o/p: 6

#] strcapitalize(string):

Returns first letter of string converted to Uppercase,requires one parameter ie.a string

Eg.

strcapitalize("hello WorlD")

o/p:Hello world

#] strcount(string,word,start=None,end=None):

Returns count of occurrences of word in a string, requires total 4 parameters ie. a base string, a word to search in string .Other two are optional, those are start, end to mention the indexes in which we want to perform search operation.

Eg.

strcount("abba","a",0,2)

o/p: 1

#] strfind(string,word,start=None,end=None):

Returns index of first occurrence of word in a string, requires total 4 parameters ie. a base string, a word to search in string .Other two are optional, those are start, end to mention the indexes in which we want to perform search operation.

Eg.

strfind("hi there","t")

o/p: 3

#] strcenter(string,width=1,fillchar=" "):

Returns centered string, requires 3 parameters those are a base string to perform operations on, width to decide the string of that width to return ,and one optional parameter ie.fillchar to decide character to center the string(it must be single character).

Eg.

strcenter("aad",24,"X")
o/p: 'XXXXXXXXXXaadXXXXXXXXXX'

#] strisalphanum(string):

Returns True if string is alphanumeric else False, requires one parameter ie. a string.

Eg.

strisalphanum("a1ad0akjsc")
o/p: True

#] strjoin(string,iterable):

Returns a string by joined to all the elements of an iterable (such as list, string and tuple),requires 2 parameters ie. a string and a iterable

Egs.

strjoin("100","abc")
o/p: 'a100b100c'

strjoin("xyz",("100","200","300"))
o/p: '100xyz200xyz300'

strjoin("",("100","200","300"))
o/p: '100200300'

#] strsplit(string,seperator=None,maxsplit=-1):

Breaks up a string at the specified separator and returns a list of strings(we can specify max. splits to perform),requires 3 paramters ie. a string ,a separator for separating string which is a space by default and a maxsplit to specify no of splits we want.

Eg.

strsplit("hi there,my name is xyz","e",4)
o/p: ['hi th', 'r', ',my nam', ' is xyz']

strsplit("hi there,my name is xyz",)
o/p: ['hi', 'there,my', 'name', 'is', 'xyz']

#] strtitle(string):

Returns a string with first letter of each word capitalized, requires only one parameter ie. a string.

Eg.

strtitle("hello worlD")
o/p: 'Hello World'

#] strreplace(string,oldstring,newstring,count=-1)):

Returns a copy of the string where all occurrences of a substring is replaced with another substring.(we can specify count which decides how many replacements needs to perform, requires 4 parameters those are a string to perform operation on, an oldstring ,an newstring to replace ,and count to determine no of replacements.

Eg.

strreplace("hello xyz,xyz is from xyz town ","xyz","abc",2)
o/p: 'hello abc,abc is from xyz town '

strreplace("hello xyz,xyz is from xyz town ","xyz","100")
o/p: 'hello 100,100 is from 100 town '

#] strformat(string,*args,**kwargs):

Returns the given string into formatted manner. It works as format() function, requires more than 1 arguments just like format() function.

Eg.

strformat("this is py{}.A very {key1} {key2}","thon",key2="language",key1="popular")
o/p: 'this is python.A very popular language'

#] strisdigit(string):

Returns True if all chars in string are digits else False, requires one parameter ie. a string.

Eg.

strisdigit("11892123")
o/p: True

strisdigit("11892abc123")
o/p: False

#] strisalpha(string):

Returns True if all chars in string are alphabets else False, requires one parameter ie. a string.

Eg.

strisalpha("wq")
o/p: True

strisalpha("wq1")
o/p: False

#] strstrip(string,chars=" "):

Returns a copy of the string by removing both the leading and the trailing characters (based on the string argument passed).Requires two parameters ie. a string and chars to strip.

Eg.

strstrip(" xoxo xyz e xoxo "," oxe")
o/p: 'yz'

strstrip("android is awesome","an")
o/p: 'droid is awesome'

strstrip(" android is awesome ")
o/p: 'android is awesome'

#] strpartition(string,seperator):

The strpartition() method splits the string at the first occurrence of the argument string and returns a tuple containing the part the before separator, argument string and the part after the separator. Requires two parameters ie. a string and separator to separate string.

Eg.

```
strpartition("xyz abc xyz","y")
o/p: ['x', 'y', 'z abc xyz']
```

//Tuple_operation.py(tuple manipulation functions)//

#] index(tuple_name ,index):

Searches the tuple for a specified value and returns the position of where it was found

Eg:

```
tuple_name=("I","N","D","I","A")
print (index(tuple_name, "A"))
o/p:
 4
```

#] count(tuple_name,value):

Returns the number of times a specified value occurs in a tuple

Eg:

```
tuple_name=("I","N","D","I","A")
print(count(tuple_name,"I"))
o/p:
2
```

//Array_manipulation.py(array manipulation function)//

#] Append(Array,value):

Append given value in Array and return updated array.

Eg :

```
Array=array.array('i',[3,54,6,2,3,5])
Append(Array,90)
o/p :
array('i', [3, 54, 6, 2, 3, 5, 90])
```

#] Pop(Array,index=None):

Remove the array element by index and return updated array.

Default index is set to -1.

Eg :

```
Array=array.array('i',[3,54,6,2,3,5])
Pop(Array,3)
o/p :
array('i', [3, 54, 6, 3, 5])
```

#] Remove(Array,value):

Remove() remove first occurence of given value from array and returns updated array.

Eg :

```
Array=array.array('i',[3,54,6,2,3,5])
Remove(Array,3)
o/p :
array('i', [54, 6, 2, 3, 5])
```

#] Insert(Array,index,value):

Insert new value at a given index in array and returns update array.

Eg :

```
Array=array.array('i',[3,54,6,2,3,5])
Insert(Array,3,100)
o/p :
array('i', [3, 54, 6, 100, 2, 3, 5])
```

#] Count(Array,value)

Returns the positive int number of frequency of given value in Array.

Eg :

```
Array=array.array('i',[3,54,6,2,3,5])
Count(Array,3)
o/p :
2
```

#] Extend(Array,NewArray):

Append NewArray in existing array and returns updated array.

Eg :

```
Array=array.array('i',[3,54,6,2,3,5])
NewArray=array.array('i',[100,200,300])
Extend(Array,NewArray)
o/p :
array('i', [3, 54, 6, 2, 3, 5, 100, 200, 300])
```

#] ToList(Array):

Convert Array to list and return list.

Eg :

```
Array=array.array('i',[3,54,6,2,3,5])
ToList(Array)
o/p :
[3, 54, 6, 2, 3, 5]
```

#] FromList(Array,list):

Append list items in array and returns updated array.

Eg :

```
Array=array.array('i',[3,54,6,2,3,5])
list=[500,600,700]
FromList(Array,list)
o/p :
array('i', [3, 54, 6, 2, 3, 5, 500, 600, 700])
```

#] Index(Array,value):

Return index of first occurence of given value.

Eg :

```
Array=array.array('i',[3,54,6,2,3,5])
Index(Array,54)
o/p :
1
```

#] Reverse(Array):

Reverse the order of values in array and return updated array.

Eg :

```
Array=array.array('i',[3,54,6,2,3,5])
Reverse(Array)
o/p :
array('i', [5, 3, 2, 6, 54, 3])
```

#] FromUnicode(Array,str):

Extends the array with given string and return updated array.
The typecode must be 'u' (unicode).

Eg :

```
string=" International pvt ltd"
Array=array.array('u','Xpanxion')
FromUnicode(Array,string)
o/p :
array('u', 'Xpanxion International pvt ltd')
```

#] ToUnicode(Array):

Returns the converted unicode string from unicode array.

Eg :

```
Array=array.array('u','Xpanxion')
ToUnicode(Array)
o/p:
"Xpanxion"
```

//List_operation.py(list manipulation functions)//

#] append(list_name, z):

The append() method appends a passed obj into the existing list and updates it

Eg:

```
list_name=[1,2,3,3,7]
list_name=append(list_name,4)
print (list_name)
o/p:
[1, 2, 3, 3, 7, 4]
```

#] insert(list_name,value,index):

The insert() method inserts object into list at offset index.

This method does not return any value but it inserts the given element at the given index.

Eg:

```
list_name=[1,2,3,3,7,4]
list_name = insert( list_name, 5, 1)
print (list_name)
o/p:
[1, 5, 2, 3, 3, 7, 4]
```

#] remove(list_name,x):

This method does not return any value but removes the given object from the list.

Eg:

```
list_name= [1, 5, 2, 3, 3, 7, 4]
list_name= remove(list_name, 5)
print (list_name)
o/p:
[1, 2, 3, 3, 7, 4]
```

#] pop(list_name,index):

The pop() method removes object from the list from given index.

Eg:

```
list_name= [1, 5, 2, 3, 3, 7, 4]
list_name= pop(list_name, 3)
print (list_name)
o/p:
[1, 2, 3, 7, 4]
```

#] reverse(list_name):

The reverse() method reverses objects of list in place.

Eg:

```
list_name= [1, 2, 3, 7, 4]
list_name= reverse(list_name)
print(list_name)
o/p:
[4, 7, 3, 2, 1]
```

#] len(list_name):

The len() method returns the number of elements in the list.

Eg:

```
list_name= [1, 2, 3, 7, 4]
print(len(list_name))
o/p:
5
```

#] count(list_name ,x):

The count() method returns count of how many times obj occurs in list.

Eg:

```
list_name= [4, 7, 3, 2, 1]
print(count(list_name,3))
o/p:
1
```

#] copy(list_name,new_list):

The copy() method copies all objects of one list to new list.

Eg:

```
list_name=[4, 7, 3, 2, 1]
print(copy(list_name, new_list))
o/p:
[4, 7, 3, 2, 1]
```

#] index(list_name,value):

The index() method returns the lowest index in list that obj appears.

Eg:

```
list_name= [4, 7, 3, 2, 1]
print (index(list_name, 3))
o/p:
2
```