FUNCTIONS

The def keyword.

The ***def*** keyword is used to define a function in python. The defined function can then be called and python will execute it.

```
>>> def avgOfACollection(collectionOfNumbers):
        total=0
        howManyNumbers=len(collectionOfNumbers)
        for i in collectionOfNumbers:
                total+=i
        average=total/howManyNumbers
        return average

>>> avgOfACollection(range(10))
4.5
>>>
>>> avgOfACollection([12,56,25,12,76,201])
63.666666666666664
>>>
```

A function is a first class object and is **assignable to a namespace**.

```
>>> avg=avgOfACollection
>>> avg
<function avgOfACollection at 0x03B46C90>
>>> avg(range(20,30))
24.5
>>>
```

A function can also be **passed as an argument to a function**.

```
>>> def findAvg(average,collection):
        avg=average(collection)
        return avg

>>> collection=range(100)
>>> avg=findAvg(avgOfACollection,collection)
>>> avg
49.5
>>>
```

A function can also be **returned by a function.**

```
>>> def findAvg(average,collection):
        return average(collection)

>>> collection=range(100)
>>> avg=findAvg(avgOfACollection,collection)
>>> avg
49.5
>>>
```

A function that is created with a ***def*** keyword is called a **named function**. A function can be called using the name that references the function, so, it is a **callable**. This can be verified by noting the __call__ in the contents of the function object.

```
>>> type(findAvg)
<class 'function'>
>>>
>>>dir(findAvg)
['__annotations__', '__call__', '__class__', '__closure__', '__code__', '__defaults__',
'__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__get__',
'__getattribute__', '__globals__', '__gt__', '__hash__', '__init__', '__init_subclass__',
'__kwdefaults__', '__le__', '__lt__', '__module__', '__name__', '__ne__', '__new__',
'__qualname__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__',
'__str__', '__subclasshook__']
>>>
>>>
```

Note the similar properties between the contents of the function and the base object of python. A Function is an object.

```
>>> dir(object)
['__class__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__',
'__getattribute__', '__gt__', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__',
'__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__',
'__str__', '__subclasshook__']
>>>
>>> dir(findAvg)
['__annotations__', '__call__', '__class__', '__closure__', '__code__', '__defaults__',
'__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__get__',
'__getattribute__', '__globals__', '__gt__', '__hash__', '__init__', '__init_subclass__',
'__kwdefaults__', '__le__', '__lt__', '__module__', '__name__', '__ne__', '__new__',
'__qualname__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__', '__sizeof__',
'__str__', '__subclasshook__']
>>>
```

DOCSTRING OF FUNCTION

The first string after the function header is called the docstring and is short for documentation string. It is used to explain in brief, what a function does. Although optional, documentation is a good programming practice.

```
>>> def somefunction(x):
        '''this function takes an argument "x" and
        replicates it 5 times. The function returns nothing'''
        print(x*5)


>>> somefunction('$')
$$$$$
>>> somefunction(7)
35
>>>
```

THE RETURN STATEMENT

Function always returns a value. The return statement is used to exit a function and go back to the place from where it was called. This statement can contain expression which gets evaluated and the value is returned. If there is no expression in the statement or the return statement itself is not present inside a function, then the function will return the None object.

```
>>> def avgOfACollection(collectionOfNumbers):
        total=0
        howManyNumbers=len(collectionOfNumbers)
        for i in collectionOfNumbers:
                total+=i
        average=total/howManyNumbers
        return average
>>>
>>> def findAvg(average,collection):
        return average(collection)
>>>
>>>
>>> def somefunction(x):
        '''this function takes an argument "x" and
        prints it 5 times.The function returns nothing'''
        print(x*5)
```

FUNCTIONS

EXAMPLES:

1.

```
>>> def letter_count(text, letters='abcd'):
        count = 0
        for char in text:
                if char in letters:
                        count += 1
        return count

>>>
>>> letter_count('M Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900
32 bit (Intel)] on win32 Type "copyright", "credits" or "license()" for more information.')
10
>>>
>>> letter_count('e')
0
>>>
>>> letter_count('M Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900
32 bit (Intel)] on win32 Type "copyright", "credits" or "license()" for more information.',
'e')
9
>>>
>>> letter_count('M Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:04:45) [MSC v.1900
32 bit (Intel)] on win32 Type "copyright", "credits" or "license()" for more information.', '
')
22
>>>
```

2.

```
>>> def multiplyFunction(num1, num2):
        "This function multiply the parameters and return the result"
        result = num1 * num2
        return result

>>>
>>> value1 = int(input("Enter a value :"))
Enter a value :12
>>>
>>> value2 = int(input("Enter a value :"))
Enter a value :9
>>>
>>> res = multiplyFunction(value1, value2)
>>> print(res)
108
```

ASSIGNMENT

WRITE A SINGLE FUNCTION THAT FINDS THE AVERAGE OF THE NUMBERS IN ANY OF ALL THE COLLECTIONS GIVEN BELOW.
Consider that it will be possible that some of the objects in the collections below may be used to get numeric objects and used in the average.….

```
coll1=range(100)

coll2=range(0,100,3)

coll3=[2,5,8,13,23,45,76]

coll4=[12,’123’,24,’234’,35,’345’]

coll5=[[12,’123’,’abcd’,24,’234’,’def’,35,’345’,b’a’,b’A’]

coll6=[(11,21,’31’,b’1’),101,202.45,b’abcd’,’123.45’]
```