Functions:

What is a function:

A fuction is a block of code use to perform specific action. It provides code reusability.

Creating a function:

In [10]:
```
#Creating a simple print function
def print_(x):
    print (x)

#'def' is a keyword that is used to define a function. This is followed by t
#The input parameters or arguments are placed inside this parenthesis.
#The first statement of a function can be optional : the document string or
#A function may return some value or may not return anything.
```

Calling a function:

In [11]:
```
#For calling a function, we use function name and pass the required argument
#above function we will use:
print_('hello world')
```

```
hello world
```

Difference between print and return

A print statement is just used to give status of the code while debugging. It just prints the message that user defines. we cannot use the print value later on. Its just a message.

A return statment in function is used to return the value after the function is excuted and this value can be used in the other operations or functions. A function may or may not return the value. for eg:

In [3]:
```
def add_(a,b):
    return (a+b)
c = add_(3,4)
c + 10
```

Out[3]:
```
17
```

In [3]:
```
def add_(a,b):
    print (a+b)
c = add_(3,4)
c + 10 #A function with no return, returns none. Therefore we cannot use the
```

```
7

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-3-3ee293e7e983> in <module>
      2     print (a+b)
      3 c = add_(3,4)
----> 4 c + 10 #A function with no return, returns none.

TypeError: unsupported operand type(s) for +: 'NoneType' and 'int'
```

Difference between Global and Local Variables:

In [ ]:
```
#Will try to understand this difference by looking at the below example.
```

A local variable is defining the variable within the function itself. This local varible will not be accessed by any other function outside it . Whereas the global variable is defining the variable globally so that it can be accessed by all the functions. Will create two functions and define the same variable locally.

In [13]:
```
#Defining Locally: 'b' is the variable that is defined in both the function
#hence while appending values in 'b' in one function wont change the value o
def func_A():
    b = []
    for i in range (0,5):
        b.append(i)
    print (b)

def func_B():
    b = []
    for i in range (10,15):
        b.append(i)
    print (b)

func_A()
func_B()
```

```
[0, 1, 2, 3, 4]
[10, 11, 12, 13, 14]
```

In [14]:
```
#Defining globally: 'c' is the variable that is defined globally and hence a
#modifies the value of 'c'. Will create the same function as above but decla
#the function
c = []
def func_X():
    for i in range (0,5):
        c.append(i)
    print (c)

def func_Y():
    for i in range (10,15):
        c.append(i)
    print (c)

func_X()
func_Y()
```

```
[0, 1, 2, 3, 4]
[0, 1, 2, 3, 4, 10, 11, 12, 13, 14]
```

Function Arguments:

Arguments types can be : 1) Required arguments 2) Keyword arguments 3) Default arguments 4) Variable length arguments

Required arguments:

Required arguments are the arguments that are required by the function for its execution else it will throw an error even if one of the argument is missing. This is also called as positional arguments as the arguments are needed to be passed in correct order. for eg:

In [21]:
```
def info_R(x,y):
    return ('my name is {} and my age is {}'.format(x,y))
info_R('mike',25)
```

Out[21]:
```
'my name is mike and my age is 25'
```

Keyword arguments:

In case of keyword argument the arguments are identified by the parameter name. So this allows us to place arguments in any order as interpreter is going to use keywords to identify the match.

In [23]:
```
def info_K(x,y):
    return ('my name is {} and my age is {}'.format(x,y))
info_K(y = 25,x = 'mike')
```

Out[23]:
```
'my name is mike and my age is 25'
```

Default arguments:

Default arguments assumes a default value if some value is not provided during the function call for that argument. for eg:

In [26]:
```
def info_D(x,y=30):
    return ('my name is {} and my age is {}'.format(x,y))
info_D(x = 'mike')
```

Out[26]:
```
'my name is mike and my age is 30'
```

Variable Length arguments:

When we dont know the actual number of arguments then we use variable length arguments. This is defined by placing an astric * before the variable name.

In [1]:
```
def info_V(*x):
    for i in x:
        if i < 20:
            print (i,' is less than 20')
        else:
            print (i,' is greater than 20')
info_V(10,12,19,25,26,13,16)
```

```
10  is less than 20
12  is less than 20
19  is less than 20
25  is greater than 20
26  is greater than 20
13  is less than 20
16  is less than 20
```

Lambda Function:

Lambda function are defined by writing lambda. It takes any number of arguments but return a single value after evaluating an expression. They have their own local space of variables also they can access variables that are defined globally.

In [2]:
```
lam = lambda x,y,z : x/y*z
lam (20,25,100)
```

Out[2]:
```
80.0
```

Map operator with functions:

While executing a function, when a function reaches the 'return' keyword it terminates and executes the next line after that.for eg

In [10]:
```
def loop_():
    for i in range (0,5):
        return (i)
x = loop_()
print (x)
print ('continue from here')
#Above I have created a for loop, when I call this function, the function on
#in x and printed.
```

```
0
continue from here
```

so in order to run this function over a iterator like list, array etc. we use 'map' operator as:

In [8]:
```
def rev (x):
    rev = 1/int(x)
    return rev
reversal = [1,2,3,4,5]
y = []
for i in map(rev,reversal):
    y.append(i)
print (y)
#Above we have created a function that that reverses the given parameter, th
#function over an iterator.
```

```
[1.0, 0.5, 0.3333333333333333, 0.25, 0.2]
```

In [12]:
```
#Another example:
def strip_ (x):
    s = x.strip()
    return s
strip = ['  abc','  pqr   ','xyz      ']
z = []
for i in map(strip_, strip):
    z.append(i)
print (z)
#Above we have created a function that takes the uncleaned data and removes
#in new list z. we have used the 'map' operator on an iterator
```

```
['abc', 'pqr', 'xyz']
```

Generators in function:

The difference between return and yield, while a return statement terminates a function entirely, yield statement pauses the function saving all its states and later continues from there on successive calls.

In [17]:
```
def gen():
    for i in range(0,5):
        yield (i)

for j in gen():
    print (j)
#When function call reaches yield, it pauses there and returns the call back
```

```
0
1
2
3
4
```

Filter in function:

If we want to filter out a iterator based on some condition. It genrally tests each element of a sequence whether it satisfies the condition or not and returns the elements that satisfies the condition

In [29]:
```
def filt_(x):
    vowel = ['a','e','i','o','u']
    if x in vowel:
        return True
    else:
        return False
y= ['r', 'e', 'f', 'r', 'i', 'g', 'e', 'r', 'a', 't', 'o', 'r']
filtered = filter (filt_,y)
for i in filtered:
    print (i)
#In the above function the given iterator is being filtered based on the con
#or not
```

```
e
i
e
a
o
```

Practise Exercise:

In [38]:
```
# create a function that prints the string given as an argument
def str_(x):
    print (x)
str_('demo function')
```

```
demo function
```

In [45]:
```
#create a function that takes two argument and returns the product.
def prod_(x=0,y=1):
    return (x*y)
prod_(8*9)
```

Out[45]: 72

In [37]:
```
# Create a function that prints a string passed as an argument
def num_():
    x = int(input('plz enter a no. '))
    if x % 2 == 0:
        print (x,' is even')
    else:
        print (x,' is odd')
num_()
```

```
plz enter a no. 16
16  is even
```

In [32]:
```
# Create a function that takes sum of all elements passed in the argument:
def sum_(*x):
    total = 0
    for i in x:
        total += i
    return total

sum_(10,15,20,25,30)

```

Out[32]: 100

In [39]:
```
# Create a function that squares all the numbers in the iterator (apply map
s = [1,2,3,4,5,6]

def sqr_(x):
    return x**2

z = []
for i in map (sqr_,s):
    z.append(i)
print (z)
```

```
[1, 4, 9, 16, 25, 36]
```

In [41]:
```
# Create a function that filters out all the lower case letters and returns
t = ['a','B','C','d','e']

def filt_(x):

    if x.isupper():
        return True
    else:
        return False

for i in filter(filt_,t):
    print (i)

```

```
B
C
```

Practise Questions:

```
1. Create a function that takes variable arguments and return the maximum.
2. Create a function that calculates the factorial of the number given the user as input.
3. Create a function that check whether the number entered by the user is prime or not
4. Create a function that calculates the percentage of the given list of marks out of a total of 50. perc = [20,26,30,36,42,46,50]
5. Create a function that determine all factors of the given input by the user and returns the list of factors. For eg: if user enters 6 then [1,2,3,6]
6. Create a function that takes user input as  string and calculates all the upper case, lower case letters.
7. Create a function that takes user input as number and determines whether number is prime or not.
8. Create a function that takes variable arguments as number and return only a list of unique number for eg: (if user enters (1,1,2,2,2,3,3,4,4,5,6) then output will be [1,2,3,4,5,6])
9. Create a function that takes input as number from user and determines whether the number is perfect or not. For a perfect number sum of all its factors except the number is equals the number itself for eg: (1+2+3 = 6)
10. Create a function that cleans unwanted spaces from the list given as:
    s = ['  abc','  pqr   ','xyz      ']
11. Create a function that returns only the even number less than 20 from the given list : [2,8,16,7,13,21,24,6,100]

```

Answers:

In [1]:
```
def mx_ (*x):
    return (max(x))
mx_(3,10,4)

```

Out[1]:
```
10
```

In [18]:
```
def factors():
    x = int (input ('enter the number: '))
    factors = []
    for i in range (1,x+1):
        if x%i == 0:
            factors.append(i)
    return (factors)
factors()
```

```
enter the number: 6
```

Out[18]:
```
[1, 2, 3, 6]
```

In [4]:
```
def UL():
    x = input('enter the string ')
    upper_counter = 0
    lower_counter = 0
    for i in x:
        if i.isupper():
            upper_counter += 1
        else:
            lower_counter += 1
    return (upper_counter , lower_counter)
UL()
```

```
enter the string AAAk TT jkljdf JJ
```

Out[4]:
```
(7, 10)
```

In [11]:
```
def prime ():
    x = int(input('please enter a number '))
    flag = 1
    for i in range(2,x):
        if x%i==0:
            flag = 0
    if flag == 1:
        print (x,' is prime number')
    else:
        print (x,' is not a prime number')
prime ()
```

```
please enter a number 19
19  is prime number
```

In [16]:
```
def perfect ():
    x = int (input('enter a number: '))
    factors = []
    sum = 0
    for i in range (1,x):
        if x%i == 0:
            factors.append(i)
    for i in factors:
        sum += i
    if (sum == x):
        print (x,' is a perfect number')
    else:
        print (x, 'is not a perfect number')
perfect()

```

```
enter a number: 28
28  is a perfect number
```

In [1]:
```
def strip_ (x):
    s = x.strip()
    return s
strip = ['  abc','  pqr   ','xyz      ']
z = []
for i in map(strip_, strip):
    z.append(i)
print (z)
```

```
['abc', 'pqr', 'xyz']
```

In [5]:
```
def fac_(x):
    fac = 1
    for i in range(1,x+1):
        fac *= i
    return fac
fac_(6)
```

Out[5]:
```
720
```

In [8]:
```
def even (x):
    if x in range (0,20,2):
        return x

filt = [2,8,16,7,13,21,24,6,100]

num_lt20 = []
for i in filter (even,filt):
    num_lt20.append(i)
print (num_lt20)

```

```
[2, 8, 16, 6]
```

In [14]:
```
x = []
def uniq (*y):
    for i in y:
        if i not in x:
            x.append(i)
    return (x)

uniq (1,1,2,2,2,3,3,4,5,6,6)
```

Out[14]:
```
[1, 2, 3, 4, 5, 6]
```

In [ ]:
```
perc = [20,26,30,36,42,46,50]
def perc_ (x,y=50):
    return x/y*100

perc_cal = []
for i in map (perc_,perc):
    perc_cal.append(i)
print (perc_cal)

```