Functions

1. Functions Basics

In [1]:
```
def greet():
    print("Hello World!")
```

In [2]:
```
greet()
```

```
Hello World!
```

In [3]:
```
greet()
greet()
greet()
```

```
Hello World!
Hello World!
Hello World!
```

In [4]:
```
def greet():
    print("Hello World!")
    return("!!!")

rc = greet()
print(rc)
```

```
Hello World!
!!!
```

In [7]:
```
def greetN(n):
    '''Prints "Hello World" n times" '''
    i = 0
    while(i < n ):
        print("Hello World!")
        i = i + 1
    return("!!!")

rc = greetN(5)
print(rc)
```

```
Hello World!
Hello World!
Hello World!
Hello World!
Hello World!
!!!
```

2. Parameter Passing

In [9]:
```
def sumNums( n1, n2, n3 ):
    sum = n1 + n2 + n3
    return sum


tot = sumNums( 10, 20, 30)
print("Total = ", tot )
```

```
Total =  60
```

In [16]:
```
def sumNums( n1, n2=20, n3=30 ):
    sum = n1 + n2 + n3
    return sum


tot = sumNums( 10, 50  )
print("Total = ", tot )
```

```
Total =  90
```

In [23]:
```
def sumVariNums( *args ):
    sum = 0
    for n in args:
        sum = sum + n
    return sum


tot = sumVariNums( 10, 20, 30, 40, 50 )
print("Total = ", tot )
```

```
Total =  150
```

In [27]:
```
def sumVariNums( *args ):
    sumList = []

    for n in args:
        sum = 0
        for i in n:
            sum = sum + i
        sumList.append(sum)
    return sumList


totList = sumVariNums( [1, 2, 3], [4, 5, 6, 10 ], [7, 8, 9, 10, 1
1], [10, -5, -3, -1, 100 ], [10, 20 ])

print("Totals List: ", totList )
```

```
Totals List:  [6, 25, 45, 101, 30]
```

3. More Parameters Passing

In [34]:
```
def fruitBasket( **kwargs ):
    count = 0
    for fruit, cnt in kwargs.items():
        printString = "{:20}{:10}".format(fruit, cnt)
        print( printString )
        count = count + cnt
    return count

tot = fruitBasket( apples=100, banana=144, pears=77, grapes=200, ma
ngoes=35 )
print("-" * 30 )
print( "{:20}{:10}".format("Total Fruits: ", tot) )
```

```
banana                144
grapes                200
apples                100
pears                  77
mangoes                35
------------------------------
Total Fruits:         556
```

In [35]:
```
def myFunc( n1, n2, n3, *args, **kwargs ):
    sum1 = sum2 = sum3 = 0
    sum1 = n1 + n2 + n3
    for n in args:
        sum2 = sum2 + n
    for k,v in kwargs.items():
        sum3 = sum3 + v

    sums = [ sum1, sum2, sum3 ]
    return sums

sums = myFunc( 10, 20, 30, 11, 22, 33, 44, one=100, two=200, three=
300 )
print ( sums )
```

```
[60, 110, 600]
```

4. Iterators

In [36]:
```
x = [ 1, 2, 3, 4, 5]
for i in x: print(i)
```

```
1
2
3
4
5
```

In [39]:
```
s = "Hello"
for c in s: print(c)
```

```
H
e
l
l
o
```

In [40]:
```
x
```

Out[40]:
```
[1, 2, 3, 4, 5]
```

In [41]:
```
I = iter( x )
```

In [42]:
```
type( I )
```

Out[42]:
```
list_iterator
```

In [43]:
```
I.__next__()
```

Out[43]:
```
1
```

In [44]:
```
I.__next__()
```

Out[44]:
```
2
```

In [45]:
```
I.__next__()
```

Out[45]:
```
3
```

In [46]:
```
I.__next__()
```

Out[46]:
```
4
```

In [47]:
```
I.__next__()
```

Out[47]:
```
5
```

In [48]:
```
I.__next__()
```

```
---------------------------------------------------------------------
---------
StopIteration                             Traceback (most recent c
all last)
<ipython-input-48-ae00d62724fd> in <module>()
----> 1 I.__next__()

StopIteration:
```

In [49]:
```
s
```

Out[49]:
```
'Hello'
```

In [50]:
```
S = iter( s )
```

In [51]:
```
S.__next__()
```

Out[51]:
```
'H'
```

In [52]:
```
S.__next__()
```

Out[52]:
```
'e'
```

In [53]:
```
S.__next__()
```

Out[53]:
```
'l'
```

In [54]:
```
S.__next__()
```

Out[54]:
```
'l'
```

In [55]:
```
S.__next__()
```

Out[55]:
```
'o'
```

In [56]:
```
S.__next__()
```

```
---------------------------------------------------------------------
---------
StopIteration                             Traceback (most recent c
all last)
<ipython-input-56-1aa93545d9a9> in <module>()
----> 1 S.__next__()

StopIteration:
```

5. Generator Functions

In [58]:
```
x = range( 10 )
for i in x: print(i, end=' ')
```

```
0 1 2 3 4 5 6 7 8 9
```

In [59]:
```
# y = range( start, stop, step )

y = range( 10, 100, 5  )
for i in y: print(i, end=' ')
```

```
10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90 95
```

In [60]:
```
type( y )
```

Out[60]:
```
range
```

In [61]:
```
def myRange(start, stop, step):
    i = start
    while( i < stop ):
        yield i
        i = i + step
```

In [62]:
```
R = myRange( 0, 10, 1)
```

In [63]:
```
type(R)
```

Out[63]:
```
generator
```

In [64]:
```
for i in R: print( i, end=' ')
```

```
0 1 2 3 4 5 6 7 8 9
```

Lambda Functions

In [20]:
```
def power2(x):
    return x ** 2
power2(5)
```

Out[20]:
```
25
```

In [21]:
```
lambda x : x ** 2
```

Out[21]:
```
<function __main__.<lambda>>
```

In [22]:
```
sq = lambda x : x ** 2
```

In [23]:
```
sq(5)
```

Out[23]:
```
25
```

In [24]:
```
type(sq)
```

Out[24]:
```
function
```

In [25]:
```
sum = lambda x, y, z : x+y+z
sum(1,2,3)
```

Out[25]:
```
6
```

In [26]:
```
isEven = lambda x : x % 2 == 0
isEven(6)
```

Out[26]:
```
True
```

In [27]:
```
isEven(7)
```

Out[27]:
```
False
```

```
In [28]: func1 = lambda x : x + 2 if x%2==0 else x+1
         func1(10)

Out[28]: 12

In [29]: func1(7)

Out[29]: 8

In [30]: A = [ 1, 2, 3, 'Apple', lambda n:n*4 ]

In [31]: A[4](5)

Out[31]: 20
```

1. map(fn, x)

2. reduce(fn, x)

3. filter(fn, x)

fn: Any function created with a def keyword or a Lambda function. Normally we use Lambda functions with these 3 functions.

x: Any container object like a String, List or a Tuple.

1. map(fn, x) executes 'fn' function on each item of the container object and produces a new container object.

2. reduce(fn, x) works differently. It applies 'fn' function continually to the sequence to produce a single value.

import functools for Python 3.x

reduce(lambda x,y: x+y, [47, 11, 42, 13]) produces 113

3. filter(fn, x) function filters out all the items of the list for which 'fn' function returns a True.

map() function code examples

In [8]:
```
C = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
C
```

Out[8]:
```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

In [10]:
```
fn = lambda x: x * 2
list( map(fn, C) )
```

Out[10]:
```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

In [11]:
```
fn = lambda x: x+2 if x%2==0 else x+1
list( map(fn, C) )
```

Out[11]:
```
[2, 4, 4, 6, 6, 8, 8, 10, 10, 12]
```

In [15]:
```
C = "hello"
fn = lambda s:s.upper()
list( map(fn, C) )
```

Out[15]:
```
['H', 'E', 'L', 'L', 'O']
```

In [17]:
```
C = "Python Programming if fun".split()
fn = lambda s:s.upper()
list( map(fn, C) )
```

Out[17]:
```
['PYTHON', 'PROGRAMMING', 'IF', 'FUN']
```

In [18]:
```
C = "Python Programming if fun".split()
fn = lambda s:len(s)
list( map(fn, C) )
```

Out[18]:
```
[6, 11, 2, 3]
```

In [19]:
```
x = [1,2,3]
y = [10,20,30]
fn = lambda a,b: a*b
list( map(fn, x,y) )
```

Out[19]:
```
[10, 40, 90]
```

reduce() function code examples

In [20]:
```
import functools
```

In [22]:
```
C = [ 10, 11, 22, 33, 5, 77, 22 ]
fn = lambda x,y: x if x>y else y
functools.reduce(fn, C)
```

Out[22]:
```
77
```

In [24]:
```
C = [ 10, 11, 22, 33, 5, 77, 22 ]
fn = lambda x,y: x+y
functools.reduce(fn, C)
```

Out[24]:
```
180
```

filter() function code examples

In [25]:
```
C = [ i for i in range(20)]
C
```

Out[25]:
```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
19]
```

In [27]:
```
fn = lambda x: x>10
list( filter(fn, C) )
```

Out[27]:
```
[11, 12, 13, 14, 15, 16, 17, 18, 19]
```

In [28]:
```
fn = lambda x: x%2
list( filter(fn, C) )
```

Out[28]:
```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]
```

In [29]:
```
fn = lambda x: x%2==0
list( filter(fn, C) )
```

Out[29]:
```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

In [32]:
```
import os
os.environ['PATH']
```

Out[32]:
```
'/usr/local/mysql/bin/:/Users/krishnayamarthy/anaconda/bin:/Librar
y/Frameworks/Python.framework/Versions/3.4/bin:/Library/Framework
s/Python.framework/Versions/3.4/bin:/Library/Frameworks/Python.fra
mework/Versions/3.4/bin:/Library/Frameworks/Python.framework/Versi
ons/3.4/bin:/Library/Frameworks/Python.framework/Versions/3.4/bi
n:/usr/local/bin:/usr/bin:/bin:/usr/sbin:/sbin'
```

In [ ]:
```

```