Functions

In [1]:
```
# Function Definition
def greetings(name, greeting):
    print(greeting + ' ', name, sep = '')
```

In [2]:
```
# Function call
greetings('John', 'Hello')
```

```
Hello! John
```

Positional Argument

In [3]:
```
greetings('Hello', 'John')
```

```
John! Hello
```

Keyword Arguments

In [4]:
```
greetings(greeting= 'Hello', name = 'John')
```

```
Hello! John
```

Parameters with Default value

In [5]:
```
def greetings(name, greeting):
    print(greeting + ' ', name, sep = '')
```

In [6]:
```
greetings('John')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-6-e93223bc8154> in <module>
----> 1 greetings('John')

TypeError: greetings() missing 1 required positional argument: 'greeting'
```

In [7]:
```
def greetings(name, greeting = 'Hello'):
    print(greeting + ' ', name, sep = '')
```

In [8]:
```
greetings('John')
```

```
Hello! John
```

In [9]:
```
greetings('John', 'Hi')
```

```
Hi! John
```

Arbitrary Arguments

Non Keyword Arbitrary Argument

In [10]:
```
# Function defined with non keyword variable no. of arguments
def total(*args):
    tot = 0
    for n in args:
        tot += n
    print(tot)
```

In [11]:
```
total(23, 67)
```

```
90
```

In [12]:
```
total(90, 100, 10, 30, 40)
```

```
270
```

Keyword Arbitrary Argument

In [13]:
```
# Function defined with variable no. keyword variable no. of arguments
def information(**kwargs):
    for key in kwargs.keys():
        print(key, ":", kwargs[key])
```

In [14]:
```
information(name = 'John Davis', age = 34)
```

```
name : John Davis
age : 34
```

In [15]:
```
information(name = 'John Davis', age = 34, city = 'London', country  = 'UK')
```

```
name : John Davis
age : 34
city : London
country : UK
```

return statement

In [16]:
```
def net_amount(Amount, tax= 18, discount = 10):
    disc = (Amount * discount)/100
    tax_amnt = (Amount - disc) * tax/100
    net = (Amount - disc) + tax_amnt
    print(round(net,2))
```

In [18]:
```
amnt = float(input('Please enter the bill amount :'))
tax = float(input('Please enter tax percentage :'))
discount = float(input('Please enter discount percentage :'))
```

In [19]:
```
return_value = net_amount(amnt, tax, discount)
```

```
4632.5
```

In [20]:
```
print(return_value)
```

```
None
```

In [21]:
```
def net_amount(Amount, tax= 18, discount = 10):
    disc = (Amount * discount)/100
    tax_amnt = (Amount - disc) * tax/100
    net = (Amount - disc) + tax_amnt
    return net
```

In [22]:
```
amnt = float(input('Please enter the bill amount :'))
tax = float(input('Please enter tax percentage :'))
discount = float(input('Please enter discount percentage :'))
```

In [23]:
```
return_value = net_amount(amnt, tax, discount)
```

In [24]:
```
print('Total Amount to be paid : ', return_value)
```

```
Total Amount to be paid :  4632.5
```

Scope of Variables

global variable

In [25]:
```
var = 10
print(var)
```

```
10
```

local variables

In [26]:
```
def foo(inp):
    # local variable
    lv = 5
    print('input parameter : ', inp)
    print('local value : ', lv)
```

In [27]:
```
foo(45)
```

```
input parameter :  45
local value :  20
```

calling local variable outside function

In [28]:
```
print(inp)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-28-cb8c237a903d> in <module>
----> 1 print(inp)

NameError: name 'inp' is not defined
```

calling global variable inside a function

In [29]:
```
def foo2():
    square = var ** 2
    return square
```

In [30]:
```
foo2()
```

Out[30]:
```
100
```

local and global variable with same name

In [31]:
```
var = 10 # global variable
```

In [32]:
```
def foo3():
    var = 50 # local variable
    print('inside function var =', var)
```

In [33]:
```
foo3()
print('outside function var =', var)
```

```
inside function var  = 50
outside function var = 10
```

creating or manipulating global variable inside a function

In [34]:
```
var = 15
print('before function call var =', var)

def foo4():
    global var
    var *= 5
    print('inside function var  =', var)

foo4()
print('outside function var =', var)
```

```
before function call var = 15
inside function var  = 75
outside function var = 75
```

Generator Functions

In [35]:
```
def myGenerator():
    n = 1
    print('First Iteration')
    yield 'Number 1 : n = ' + str(n)

    n += 2
    print('Second Iteration')
    yield 'Number 2 : n = ' + str(n)

    n += 3
    print('Third Iteration')
    yield 'Number 3 : n = ' + str(n)
```

generator call : way 1

In [36]:
```
gen = myGenerator()
```

In [37]:
```
gen
```

Out[37]:
```
<generator object myGenerator at 0x7fa2f7979940>
```

In [38]:
```
print(next(gen))
```

```
First Iteration
Number 1 : n = 1
```

In [39]:
```
print(next(gen))
```

```
Second Iteration
Number 2 : n = 3
```

In [40]:
```
print(next(gen))
```

```
Third Iteration
Number 3 : n = 6
```

the next function is called untill all values are yielded.

generator call : way 2

In [41]:
```
gen = myGenerator()
for value in gen :
    print(value, '\n')
```

```
First Iteration
Number 1 : n = 1

Second Iteration
Number 2 : n = 3

Third Iteration
Number 3 : n = 6
```

built in functions

In [42]:
```
print(dir(__builtins__))
```

```
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'BlockingIOError', 'BrokenPipeError', 'BufferError', 'BytesWarning', 'ChildProcessError', 'ConnectionAbortedError', 'ConnectionError', 'ConnectionRefusedError', 'ConnectionResetError', 'DeprecationWarning', 'EOFError', 'Ellipsis', 'EnvironmentError', 'Exception', 'False', 'FileExistsError', 'FileNotFoundError', 'FloatingPointError', 'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError', 'ImportWarning', 'IndentationError', 'IndexError', 'InterruptedError', 'IsADirectoryError', 'KeyError', 'KeyboardInterrupt', 'LookupError', 'MemoryError', 'ModuleNotFoundError', 'NameError', 'None', 'NotADirectoryError', 'NotImplemented', 'NotImplementedError', 'OSError', 'OverflowError', 'PendingDeprecationWarning', 'PermissionError', 'ProcessLookupError', 'RecursionError', 'ReferenceError', 'ResourceWarning', 'RuntimeError', 'RuntimeWarning', 'StopAsyncIteration', 'StopIteration', 'SyntaxError', 'SyntaxWarning', 'SystemError', 'SystemExit', 'TabError', 'TimeoutError', 'True', 'TypeError', 'UnboundLocalError', 'UnicodeDecodeError', 'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError', 'UnicodeWarning', 'UserWarning', 'ValueError', 'Warning', 'ZeroDivisionError', '__IPYTHON__', '__build_class__', '__debug__', '__doc__', '__import__', '__loader__', '__name__', '__package__', '__spec__', 'abs', 'all', 'any', 'ascii', 'bin', 'bool', 'bytearray', 'bytes', 'callable', 'chr', 'classmethod', 'compile', 'complex', 'copyright', 'credits', 'delattr', 'dict', 'dir', 'display', 'divmod', 'enumerate', 'eval', 'exec', 'filter', 'float', 'format', 'frozenset', 'get_ipython', 'getattr', 'globals', 'hasattr', 'hash', 'help', 'hex', 'id', 'input', 'int', 'isinstance', 'issubclass', 'iter', 'len', 'license', 'list', 'locals', 'map', 'max', 'memoryview', 'min', 'next', 'object', 'oct', 'open', 'ord', 'pow', 'print', 'property', 'range', 'repr', 'reversed', 'round', 'set', 'setattr', 'slice', 'sorted', 'staticmethod', 'str', 'sum', 'super', 'tuple', 'type', 'vars', 'zip']
```

map function

In [43]:
```
temp = [32.0, 65.3, 35.6, 40.0, 44.7, 21.5]
for T in temp:
    F = (1.8*T) + 32
    print('{:f} in degree C is equivalent to {:6.1f} in degree F.'.format(T,F))
```

```
22.0 in degree C is equivalent to   71.6 in degree F.
65.3 in degree C is equivalent to  149.5 in degree F.
35.6 in degree C is equivalent to   96.1 in degree F.
40.0 in degree C is equivalent to  104.0 in degree F.
44.7 in degree C is equivalent to  112.5 in degree F.
21.5 in degree C is equivalent to   70.7 in degree F.
```

In [44]:
```
def Fahrenhite(T):
    return round((1.8*T) + 32, 1)
```

In [45]:
```
results = list(map(Fahrenhite, temp))
print('Temp in deg C ', temp)
print('Temp in deg F', results)

Temp in deg C  [22.0, 65.3, 35.6, 40.0, 44.7, 21.5]
Temp in deg F  [71.6, 113.5, 95.7, 104.0, 112.5, 70.7]
```

using lambda function

In [46]:
```
results = list(map(lambda T: round(((1.8*T) + 32), 1), temp))
print('Temp in deg C ', temp)
print('Temp in deg F ', results)

Temp in deg C  [22.0, 65.3, 35.6, 40.0, 44.7, 21.5]
Temp in deg F  [71.6, 113.5, 95.7, 104.0, 112.5, 70.7]
```

filter function

In [47]:
```
numbers = [32, 65, 62, 21, 55, 91, 88, 17 ]
odd_numbers = []
for n in numbers:
    if n % 2 :
        odd_numbers.append(n)
print(odd_numbers)

[65, 21, 55, 91, 17]
```

In [48]:
```
numbers = [32, 65, 62, 21, 55, 91, 88, 17 ]
odd_numbers = list(filter ( lambda n : n % 2, numbers))
print(odd_numbers)

[65, 21, 55, 91, 17]
```

reduce function

In [49]:
```
numbers = [2,4,6,8,10,12]
tot = 0
for n in numbers:
    tot += n
print('Total :', tot)

Total : 42
```

In [50]:
```
from _functools import reduce
numbers = [2,4,6,8,10,12]
result = reduce(lambda x , y : x + y, numbers)
print('Total = ', result)

Total =  42
```

In [ ]:
```

```