

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)
C:\Users\ALPIKA~1.GUP\AppData\Local\Temp\ipykernel_5748\263682724.py 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 [19]: amnt = float(input('Please enter the bill amount '))
tax = float(input('Please enter tax percentage :'))
discount = float(input('Please enter discount percentage :'))

Please enter the bill amount 455
Please enter tax percentage :43
Please enter discount percentage :10
```

```
In [20]: return_value = net_amount(amnt, tax, discount)

585.59
```

```
In [21]: print(return_value)

None
```

```
In [22]: 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 [24]: amnt = float(input('Please enter the bill amount '))
tax = float(input('Please enter tax percentage :'))
discount = float(input('Please enter discount percentage :'))

Please enter the bill amount 566
Please enter tax percentage :10
Please enter discount percentage :3
```

```
In [25]: return_value = net_amount(amnt, tax, discount)

Total Amount to be paid : 603.922
```

Scope of Variables

global variable

```
In [27]: var = 10
print(var)

10
```

local variables

```
In [28]: def foo(inp):
        # local variable
        lc = 30
        print('input parameter : ', inp)
        print('local value : ', lc)
```

```
In [29]: foo(45)

input parameter : 45
local value : 30
```

calling local variable outside function

```
In [30]: print(inp)

-----
NameError                                 Traceback (most recent call last)
C:\Users\ALPIKA~1.GUP\AppData\Local\Temp\ipykernel_5748\3636225408.py in <module>
----> 1 print(inp)

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

calling global variable inside a function

```
In [31]: def foo2():
        square = var ** 2
        return square
```

```
In [32]: foo2()

100
```

```
Out[32]: 100
```

local and global variable with same name

```
In [33]: var = 10 # global variable
```

```
In [34]: def foo3():
        var = 50 # local variable
        print('inside function var =', var)
```

```
In [35]: foo3()
print('outside function var =', var)

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

creating or manipulating global variable inside a function

```
In [36]: 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 [37]: 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 [38]: gen = myGenerator()

gen

<generator object myGenerator at 0x00000234D3EFFEB0>
```

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

First Iteration
Number 1 : n = 1
```

```
In [41]: print(next(gen))

Second Iteration
Number 2 : n = 3
```

```
In [42]: print(next(gen))

Third Iteration
Number 3 : n = 6
```

the next function is called untill all values are yielded.

generator call : way 2

```
In [43]: 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 [44]: 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', 'WindowsError', 'ZeroDivisionError', ' _IPYTHON_', ' _build_class_', ' _debug_', ' _doc_', ' _im_port_', ' _loader_', ' _name_', ' _package_', ' _spec_', ' _abs_', ' all_', ' any_', ' ascii_', ' bin_', ' bool_', ' bre akpoint', ' bytearray', ' bytes', ' callable', ' chr', ' classmethod', ' compile', ' complex', ' copyright', ' credits', ' delattr', ' dict', ' dir', ' display', ' divmod', ' enumerate', ' eval', ' exec', ' execfile', ' filter', ' float', ' for mat', ' frozenset', ' get_ipython', ' getattr', ' globals', ' enumerate', ' hasattr', ' hash', ' help', ' hex', ' id', ' input', ' int', ' isinstance', ' isinstance', ' iter', ' len', ' license', ' list', ' locals', ' map', ' max', ' memoryview', ' min', ' nex t', ' object', ' oct', ' open', ' ord', ' pow', ' print', ' property', ' range', ' repr', ' reversed', ' round', ' runfil e', ' set', ' setattr', ' slice', ' sorted', ' staticmethod', ' str', ' sum', ' super', ' tuple', ' type', ' vars', ' zip']
```

map function

```
In [45]: temp = [22.0, 45.3, 35.4, 40.0, 44.7, 21.5]
for T in temp:
    f = ((9/5) * T) + 32
    print('{:5.1f} 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.
45.3 in degree C is equivalent to 113.5 in degree F.
35.4 in degree C is equivalent to 95.7 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 [46]: def Fahrenhite(T):
        return round(((9/5) * T) + 32, 1)
```

```
In [47]: results = list(map(Fahrenhite, temp))
print('Temp in deg C ', temp)
print('Temp in deg F ', results)
```

```
Temp in deg C [22.0, 45.3, 35.4, 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 [48]: results = list(map(lambda T: round(((9/5) * T) + 32, 1), temp))
print('Temp in deg C ', temp)
print('Temp in deg F ', results)
```

```
Temp in deg C [22.0, 45.3, 35.4, 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 [49]: numbers = [32, 45, 62, 21, 55, 91, 88, 17 ]
odd_numbers = []
for n in numbers:
    if n % 2 :
        odd_numbers.append(n)
print(odd_numbers)
```

```
[45, 21, 55, 91, 17]
```

```
In [50]: numbers = [32, 45, 62, 21, 55, 91, 88, 17 ]
odd_numbers = list(filter( lambda n : n % 2, numbers))
print(odd_numbers)
```

```
[45, 21, 55, 91, 17]
```

reduce function

```
In [51]: numbers = [2,4,6,8,10,12]
tot = 0
for n in numbers:
    tot += n
print('Total :', tot)
```

```
Total : 42
```

```
In [52]: from functools import reduce
numbers = [2,4,6,8,10,12]
result = reduce(lambda x , y : x + y, numbers)
print('Total = ', result)
```

```
Total = 42
```