## Functions

```
def condition_checker(f):
  if f%2 == 0:
    print("abc")
    print(123)
    print(1+2+3)
    print("ab"+"bc"+"ca")
  else:
    print("cba")
    print(321)
    print(3-2-1)
    print("ab"*5)
```

```
condition_checker(10)
```

```
abc
123
6
abbcca
```

```
condition_checker(15)
```

```
cba
321
0
ababababab
```

```
def intro_to_python():
  pass

def teach_datatypes():
  pass

def teach_functions():
  pass

def teach_python():
  a = intro_to_python()
  b = teach_datatypes()
  c = teach_functions()
```

```
# defining a function

def python_function():
  print("Welcome to Functions class")
  l = [100,200,300,400]
  return l

# calling a function

func_return = python_function()
print(func_return)
print(type(func_return))
```

```
Welcome to Functions class
[100, 200, 300, 400]
<class 'list'>
```

```
# passing the arguments to the functions

def f_name(name):
  print("Entered Name : {}".format(name))
  return name*2

ret_val = f_name("Virat")
print(ret_val)
print(type(ret_val))
```

```
Entered Name : Virat
ViratVirat
<class 'str'>
```

```
def f_name(name):
  print("Entered Name : {}".format(name))
  return name*2

ret_val = f_name(10)
print(ret_val)
print(type(ret_val))
```

```
Entered Name : 10
20
<class 'int'>
```

```
def f_name(name):
  print("Entered Name : {}".format(name))
  return name*2

ret_val = f_name([1,2,3])
print(ret_val)
print(type(ret_val))
```

```
Entered Name : [1, 2, 3]
[1, 2, 3, 1, 2, 3]
<class 'list'>
```

```
def sub_calculator(num):
  print("Entered Num : {}".format(num))
  a = 100
  print("abc")
  print(num*5)
  print(num+"5")
  return a-num

ret_val = sub_calculator("50")
print(ret_val)
print(type(ret_val))
```

```
Entered Num : 50
abc
5050505050
505
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-19-eb714e582bbf> in <cell line: 9>()
      7   return a-num
      8 
----> 9 ret_val = sub_calculator("50")
     10 print(ret_val)
     11 print(type(ret_val))

<ipython-input-19-eb714e582bbf> in sub_calculator(num)
      5   print(num*5)
      6   print(num+"5")
----> 7   return a-num
      8 
      9 ret_val = sub_calculator("50")

TypeError: unsupported operand type(s) for -: 'int' and 'str'
```

Next steps: Explain error

```
def f_name(name):
  print("Entered Name : {}".format(name))
  return name*2

ret_val = f_name()
print(ret_val)
print(type(ret_val))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-20-1a3d5dbfa375> in <cell line: 5>()
      3   return name*2
      4 
----> 5 ret_val = f_name()
      6 print(ret_val)
      7 print(type(ret_val))

TypeError: f_name() missing 1 required positional argument: 'name'
```

Next steps: Explain error

```
def info_data(name, age, occupation):
  print("Name: {}".format(name))
  print("Age: {}".format(age))
  print("Occupation: {}".format(occupation))

ret_val = info_data("Venky", 26, "ML Engineer")
print(ret_val)
print(type(ret_val))
```

```
Name: Venky
Age: 26
Occupation: ML Engineer
None
<class 'NoneType'>
```

```
def info_data(name, age, occupation):
  print("Name: {}".format(name))
  print("Age: {}".format(age))
  print("Occupation: {}".format(occupation))

ret_val = info_data(26, "ML Engineer", [12,23])
print(ret_val)
print(type(ret_val))
```

```
Name: 26
Age: ML Engineer
Occupation: [12, 23]
None
<class 'NoneType'>
```

```
# to pass any number of arguments, number of arguments is unknown

def string_concatenation(*s):
  print(s)
  print(type(s))

string_concatenation("a", "b", "c")
```

```
('a', 'b', 'c')
<class 'tuple'>
```

```
def string_concatenation(*s):
  print(s)
  print(type(s))

string_concatenation("a")
```

```
('a',)
<class 'tuple'>
```

```
def string_concatenation(*s):
  print(s)
  print(type(s))

string_concatenation()
```

```
()
<class 'tuple'>
```

```
def string_concatenation(*s):
  print(s)
  print(type(s))
  for i in s:
    print(len(i))
string_concatenation("")
```

```
('',)
<class 'tuple'>
0
```

```
def string_concatenation(*s): # *args , *name
  print(s)
  print(type(s))

string_concatenation("a", "b", "c", 12,12.5,False, [1,2,3], (2,3,4), {2,3,4}, {1:2,2:3})
```

```
('a', 'b', 'c', 12, 12.5, False, [1, 2, 3], (2, 3, 4), {2, 3, 4}, {1: 2, 2: 3})
<class 'tuple'>
```

```
# keyword arguments

def info_data(name, age, occupation):
  print("Name: {}".format(name))
  print("Age: {}".format(age))
  print("Occupation: {}".format(occupation))

ret_val = info_data(occupation = "ML Engineer", name = "Venky", age = 26)
```

```
Name: Venky
Age: 26
Occupation: ML Engineer
```

```
def info_data(name, age, occupation):
  print("Name: {}".format(name))
  print("Age: {}".format(age))
  print("Occupation: {}".format(occupation))

ret_val = info_data(occupation = "ML Engineer", name_2 = "Venky", age = 26)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-36-4472ff6a0350> in <cell line: 6>()
      4   print("Occupation: {}".format(occupation))
      5
----> 6 ret_val = info_data(occupation = "ML Engineer", name_2 = "Venky", age = 26)

TypeError: info_data() got an unexpected keyword argument 'name_2'
```

Next steps: Explain error

```
# let say we don't know how many keyword arguments were there, i.e keyword arguments is unknown

def info_data(**k): # **kwargs, **knames
  print(k)
  print(type(k))

info_data(name = "Venky", occupation = "ML Engineer", age = 26)
```

```
{'name': 'Venky', 'occupation': 'ML Engineer', 'age': 26}
<class 'dict'>
```

```
def info_data(**k):
  print(k)
  print(type(k))

info_data(name = "Venky", occupation = "ML Engineer", age = 26, teaching_subjects = ['Calculus', 'Python', 'Aptit
```

```
{'name': 'Venky', 'occupation': 'ML Engineer', 'age': 26, 'teaching_subjects': ['Calculus', 'Python', 'Aptitu
<class 'dict'>
```

```
info_data("Venky", "ML Engineer", 26,['Calculus', 'Python', 'Aptitude'])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-39-1cb702eaef81> in <cell line: 1>()
----> 1 info_data("Venky", "ML Engineer", 26,['Calculus', 'Python', 'Aptitude'])

TypeError: info_data() takes 0 positional arguments but 4 were given
```

Next steps: Explain error

```
# default parameter value

def my_info(country="India"):
  print("I'm from {}".format(country))
  return 10

print(my_info())
print(my_info("Russia"))
```

```
I'm from India
10
I'm from Russia
10
```

```
print(my_info())
```

```
I'm from India
10
```

```
def my_info(country="India", name = "viky"):
  print("I'm {} and I'm from {}".format(name, country))

my_info("venky", "us")
```

```
I'm us and I'm from venky
```

```
my_info(name = "US", country = "Venky")
```

```
I'm US and I'm from Venky
```

```
# pass statements

def simple_function(*s):
  pass

simple_function()
```

## Positional - Only Arguments

```
def my_func(a):
  print(a)

my_func(3)
```

```
3
```

```
my_func(a = 3)
```

```
3
```

```
# add ",/" after the positional arguments to indicate the function takes only positional arguments

def my_func(a,/):
  print(a)

my_func(3)
```

```
3
```

```
my_func(a = 3)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-59-d61aa39cd22d> in <cell line: 1>()
----> 1 my_func(a = 3)

TypeError: my_func() got some positional-only arguments passed as keyword arguments:
'a'
```

Next steps: Explain error

```
# keyword only arguments

def my_keyfunc(*,x):
  print(x)


my_keyfunc(4)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-61-d91bdd8bfe9b> in <cell line: 1>()
----> 1 my_keyfunc(4)

TypeError: my_keyfunc() takes 0 positional arguments but 1 was given
```

Next steps: Explain error

```
my_keyfunc(x = 4)
```

```
4
```

```
def my_keyfunc(x):
  print(x)

my_keyfunc(3)
```

```
3
```

```
# combining positional only arguments and keyword only arguments

def posKeyFunc(a,b,/,*,c,d):
  print("Values of a {} b {} c {} d {}".format(a,b,c,d))


posKeyFunc(10,5,12,14)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-65-7215fd1d77ea> in <cell line: 1>()
----> 1 posKeyFunc(10,5,12,14)

TypeError: posKeyFunc() takes 2 positional arguments but 4 were given
```

Next steps: Explain error

```
posKeyFunc(10,5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-66-9d74d80c02b8> in <cell line: 1>()
----> 1 posKeyFunc(10,5)

TypeError: posKeyFunc() missing 2 required keyword-only arguments: 'c' and 'd'
```

Next steps: Explain error

```
posKeyFunc(10,5, c = 12, d=10)
```

```
Values of a 10 b 5 c 12 d 10
```

```
posKeyFunc(10,5, d = 12, c=10)
```

```
Values of a 10 b 5 c 10 d 12
```

```
posKeyFunc(d = 12, c=10, 5, 10)
```

```
  File "<ipython-input-69-e4e46c234dec>", line 1
    posKeyFunc(d = 12, c=10, 5, 10)
                                  ^
SyntaxError: positional argument follows keyword argument
```

Next steps: Fix error

```
def myFunc(*s, **k):
  print(s)
  print(type(s))
  print(k)
  print(type(k))
```

```
myFunc(1.5,1,True, [1,2,3],(1,2,3),{1,2,3}, name = "Venky", age = 26, occupation = "Teaching")
```

```
(1.5, 1, True, [1, 2, 3], (1, 2, 3), {1, 2, 3})
<class 'tuple'>
{'name': 'Venky', 'age': 26, 'occupation': 'Teaching'}
<class 'dict'>
```

```
def posKeyFunc(a,b,c,d):
  print("Values of a {} b {} c {} d {}".format(a,b,c,d))

posKeyFunc(10,5, 100, d=12)
```

```
Values of a 10 b 5 c 100 d 12
```

```
# docstring

def pythonOperators(a,b):
  """
  Description: This function prints basic python operations like addtion, sub, mul and remainder operators
  Inputs: a, b
  Returns: None
  """
  print("Addition: {}".format(a+b))
  print("Sub: {}".format(a-b))
  print("Mul: {}".format(a*b))
  print("Remainder: {}".format(a%b))
  """
  print statements were done
  """
  print("Div: {}".format(a/b))
```

```
pythonOperators(15,5)
```

```
Addition: 20
Sub: 10
Mul: 75
Remainder: 0
Div: 3.0
```

## Passing Variables to the function

```
def myFunc(a): # a = b
  print("Inside Function")
  print(a)
  print(type(a))
  print(id(a))
  print("Returning From Function")
  return a

b = 100
print(b)
print(id(b))
c = myFunc(b) # c = a
print(c)
print(id(c))
```

```
100
132475442253136
Inside Function
100
<class 'int'>
132475442253136
Returning From Function
100
132475442253136
```

```
def myFunc(a): # a = b
  print("Inside Function")
  print("a: {}, id: {}".format(a,id(a)))
  a += 120
  print("a: {}, id: {}".format(a,id(a)))
  print("Returning From Function")
  return a

b = 100
print("b: {}, id: {}".format(b,id(b)))
c = myFunc(b) # c = a
print("c: {}, id: {}".format(c,id(c)))
print("b: {}, id: {}".format(b,id(b)))
```

```
b: 100, id: 132475442253136
Inside Function
a: 100, id: 132475442253136
a: 220, id: 132475442256976
Returning From Function
c: 220, id: 132475442256976
b: 100, id: 132475442253136
```

```
def myFunc(a): # a = b
  print("Inside Function")
  print("a: {}, id: {}".format(a,id(a)))
  a[2] = 2000000
  print("a: {}, id: {}".format(a,id(a)))
  print("Returning From Function")
  return a

b = [10,20,30,40]
print("b: {}, id: {}".format(b,id(b)))
print("b: {}, id: {}".format(b,id(b)))
c = myFunc(b) # c = a
print("c: {}, id: {}".format(c,id(c)))
print("b: {}, id: {}".format(b,id(b)))
```

```
b: [10, 20, 30, 40], id: 132474158227328
Inside Function
a: [10, 20, 30, 40], id: 132474158227328
a: [10, 20, 2000000, 40], id: 132474158227328
Returning From Function
c: [10, 20, 2000000, 40], id: 132474158227328
b: [10, 20, 2000000, 40], id: 132474158227328
```

```
a = 100
print(a)
print(id(a))

a += 50
print(a)
print(id(a))
```

```
100
132475442253136
150
132475442254736
```

```
def myfun(a):
  print("Before variable update in function: {}".format(a))
  a = 100
  print("Inside Function: {}".format(a))

a = 200
print("Before calling function: {}".format(a))
myfun(a)
print("After calling function, Outside the function: {}".format(a))
```

```
Before calling function: 200
Before variable update in function: 200
Inside Function: 100
After calling function, Outside the function: 200
```

```
def myfun(_c):
  print("Before variable update in function: {}".format(_c))
  _c = 100
  print("Inside Function: {}".format(_c))

b = 200
print("Before calling function: {}".format(b))
myfun(b)
print("After calling function, Outside the function: {}".format(_c))
```

```
Before calling function: 200
Before variable update in function: 200
Inside Function: 100
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-96-4f16ca90a45c> in <cell line: 9>()
      7 print("Before calling function: {}".format(b))
      8 myfun(b)
----> 9 print("After calling function, Outside the function: {}".format(_c))

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

Next steps: Explain error

```
def myfunc(a):
  print(id(a))
  a = 2457
  print("ID Inside function: {}".format(id(a)))

a = 2457
myfunc(a)
print("ID Outside function: {}".format(id(a)))
```

```
132474159318384
```

```
a = 2457
b = 2457

print(id(a))
print(id(b))
```

```
132474159332880
132474159328592
```

```
def func():
  a = 248900000
  b = 248900000
  print(id(a))
  print(id(b))

func()
```

```
132474159322864
132474159322864
```

```
def func(a):
  a = list(a)
  a[0] = 100  # scope of this variable a is limited to this particular function, outside the function if you call
  return tuple(a)

b = (10,20,30)
c = func(b)
print(b)
print(c)
```

```
(10, 20, 30)
(100, 20, 30)
```

```
def myfunc():
  a = [1,2,3]
  b = a
  print(id(a))
  print(id(b))
  b[0] = 100
  print(b)
  print(a)

myfunc()
```

```
132474160065408
132474160065408
[100, 2, 3]
[100, 2, 3]
```

```
a = [1,2,3]
```