1. Defining a function in python

def function_name (parameters):

    function body

In [56]:
```
def simple():
    print ("My first function")
```

In [57]:
```
simple()
```

```
My first function
```

In [58]:
```
def plus_ten(a):
    return a + 10
```

In [59]:
```
plus_ten(15)
```

Out[59]:
```
25
```

In [60]:
```
plus_ten(-105)
```

Out[60]:
```
-95
```

How to Create a Function with a Parameter - Exercise #1

Define a function called multiplication_by_2(x) that returns a value equal to its argument multiplied by 2.

In [61]:
```
def multipliction_by_2(x):
    return x * 2
```

In [62]:
```
 multipliction_by_2(3.5)
```

Out[62]:
```
7.0
```

Exercise 2

How to Create a Function with a Parameter - Exercise #2

Define a function called division_by_2(x) that returns a float value equal to its argument divided by 2

In [63]:
```
def division_by_2(x):
    return float(x) / 2
```

In [64]:
```
division_by_2(65)
```

Out[64]:
```
32.5
```

In [65]:
```
def division_by_2(x):
    return (x) / 2.0
```

In [66]:
```
division_by_2(78)
```

Out[66]:
```
39.0
```

2. Defining a function part 2

In [67]:
```
def plus_ten(a):
    result = a + 15
    print("Outcome")
    return result
```

In [68]:
```
plus_ten(45)
```

Out[68]:
```
Outcome
60
```

2. Exercise 1

Defining a Function in Python - Exercise #1

Define a function called exponentiation_exp_2(x) that states the value of the argument accompanied by the phrase "Raised to the power of 2:" and returns a value equal to its argument raised to the power of 2. This time, use a new variable, called result, in the body of the Function. Call the function with some argument to verify it works properly.

In [69]:
```
def exponentiation_exp_2(x):
    result = x ** 2
    print(x, "Raised to the power of 2:")
    return result
```

In [70]:
```
exponentiation_exp_2(45.7)
```

Out[70]:
```
45.7 Raised to the power of 2:
2088.4900000000002
```

3. How to use a function in another function

In [71]:
```
#function - wage - calculates your daily wage
#assuming u are paid $25/hr
#with a parameter - w_hours
#returns w_hours multiplied by 25

def wage(w_hours):
    return w_hours * 25
# 2. function - with_bonus()
#where you have a bonus of $50

def with_bonus(w_hours):
    return wage(w_hours) + 50
```

In [72]:
```
wage(8), with_bonus(7)
```

Out[72]:
```
(200, 225)
```

In [73]:
```
7 * 25
```

Out[73]:
```
175
```

In [74]:
```
175+50
```

Out[74]:
```
225
```

Hence

- the first function: wage()
- the second function: with_bonus()

Exercise

- How to Use a Function within a Function - Exercise #
  - Define a function called ***plus_five()*** that adds 5 to its argument.
  - Then, define another function named ***m_by_3()*** that multiplies the argument (the result obtained from plus_five()) by 3.
  - Verify your code was correct by calling the second function with an argument of 5.
  - **Was your output equal to 30?**

In [75]:
```
def plus_five(x):
    return x + 5

def m_by_3(x):
    return plus_five(x) * 3

m_by_3(5)
```

Out[75]:
```
30
```

4. Combining Conditional Statements and Functions

- **if johnny has saved at least $100 by the end of the week he gets a bonus of** +$50 **.**
- if not **$0**

In [76]:
```
def add_10(m):
    if m >= 100:
        m = m + 10
        return m, "Good job!"
    else:
        return "Save more!", 100 - m
```

In [77]:
```
add_10(70)
```

Out[77]:
```
('Save more!', 30)
```

In [78]:
```
add_10(120)
```

Out[78]:
```
(130, 'Good job!')
```

Exercise

- Conditional Statements and Functions - Exercise #1
  - Define a function, called **compare_the_two()**, with **two arguments.**
  - If the **first one is greater than the second one**,

- let it **print "Greater"**.
- If the **second one is greater, it should print "Less".**
- Let it **print "Equal" if the two values are the same number.**

In [79]:
```
def compare_the_two(x,y):
    if x > y:
        return "Greater"
    elif y > x:
        return "Less"
    else:
        return "Equal"
```

In [80]:
```
compare_the_two(8,8)
```

Out[80]:
```
'Equal'
```

In [81]:
```
compare_the_two(8,5)
```

Out[81]:
```
'Greater'
```

In [82]:
```
compare_the_two(-2.5,6)
```

Out[82]:
```
'Less'
```

5. Creating functions containing a few arguements

syntax

def function_name (parameter #1, parameter#2, ...):

```
    function body
```

def subtract_bc(a, b, c): result = a - b * c print ('Parameter a equals', a) print ('Parameter b equals', b) print ('Parameter c equals', c) return result

In [85]:
```
def subtract_bc():
    subtract_bc(15, 10, 3)
```

6. Built-in functions

- are functions built-in the computer

Example 1. type() function - describes the type of function

In [ ]:
```
type(10)
```

2. data types: int(); float(); str() - transforms their arguements in an integer, float and string.

In [ ]:
```
int(5.0)
```

In [ ]:
```
float(4)
```

In [ ]:
```
str('nine')
```

In [ ]:
```
str(5)
```

3. max() - points to the maximum number in a sequence.

In [ ]:
```
max(15,5,-9.5,10,30,2)
```

4. min() - returns the lowest value from a sequence of numbers.

In [ ]:
```
min(15,5,-9.5,10,30,2)
```

5. abs() - allows you to obtain the absolute value of its arguement.

In [ ]:
```
z = -45
abs(z)
```

In [ ]:
```
z = -20.15
abs(z)
```

6. sum() - calculates the sum of all elements in a list designated as an arguement.

In [ ]:
```
list_1 = [15,5,-9.5,10,30,2]
sum(list_1)
```

7. round(x,y) - returns the float of its arguement (x), rounded to a specified number of digits (y) after the decimal point.

In [ ]:
```
round(3.2516, 1)
```

8. pow(x,y) returns x to the power y

In [ ]:
```
pow(5,2)
```

9. len() - returns the number of elements in an object.

In [ ]:
```
len('Kenneth Kibe')
```

In [ ]:
```
len('KennethKibe')
```

EXERCISE

Built-in Functions in Python - Exercise #1

Obtain the maximum number among the values 25, 65, 890, and 15.

In [ ]:
```
max(25, 65, 890, 15)
```

Obtain the minimum number among the values 25, 65, 890, and 15.

In [ ]:
```
min(25, 65, 890, 15)
```

Find the absolute value of -100.

In [ ]:
```
abs(-100)
```

Round the value of 55.5.

In [ ]:
```
round(55.5)
```

Round 35.56789 to the third digit.

In [ ]:
```
round(35.56789, 3)
```

In [ ]:
```
Numbers = [1, 5, 64, 24.5]
sum(Numbers)
```

Use a built-in function to raise 10 to the power of 3.

In [ ]:
```
pow(10, 3)
```

In one line of code, find how many characters there are in the word "Elephant"?

In [ ]:
```
len("Elephant")
```

Create a function, called distance_from_zero(), that returns the absolute value of a provided single argument and prints a statement "Not Possible" if the argument provided is not a number. To solve the task, use the type() function in the body of distance_from_zero().

Call the function with the values of -10 and "cat" to verify it works correctly.

In [ ]:
```
def distance_from_zero(x):
    if type(x) == int or type(x) == float:
        return abs(x)
    else:
        print("Not Possible")

distance_from_zero(-10)
```

In [ ]:
```
distance_from_zero("cat")
```

In [ ]:
```

```