Programming in Python

By Faaiz Alshajalee

1 Repeating Actions(coverd) [...]

2 Conditions and Control Flow (coverd) [...]

3 List, Tuple, and Dictionary Comprehension (coverd) [...]

4 Functions: Reusable block of codes(coverd)

Functions are python objects that facilitates re-usability of a program/code. They allow you to write smarter, neater codes that will increase your code efficiency.

DRY: Do not rpeat yourself.... A function run when you call it....

```
The general format of a function is:


    def function_name(inputs):
        """
        docstrings
        """
        instructions to execute
        return output
```

4.1 No input

In [1]:
```
# Example 1
def hello_function():
    print ("Helloe everyone")
```

In [2]:
```
hello_function()
```
```
Helloe everyone
```

4.2 One input

In [3]:
```
# Example 1:
def equ_function(x):
    y=x+1
    return y
```

In [4]:
```
equ_function(1)
```
Out[4]:
```
2
```

In [5]:
```
# Example 2:
a, b, c = 'Python', "Matlab", "MathCad"

def sw_function(x):
    print (x, "is the best codding language")
```

In [7]:
```
sw_function(a)
```
```
Python is the best codding language
```

4.3 Many inputs

In [8]:
```
# Example 1:
def add_function(x1,x2):
    y=x1+x2
    return y
```

In [9]:
```
add_function(50,40)
```
Out[9]:
```
90
```

In [10]:
```
# Example 2:
def add2_function(x1,x2):
    y=int(x1)+int(x2)
    return y
```

In [11]:
```
add2_function("50",40)
```
Out[11]:
```
90
```

In [12]:
```
# Example 3:
def add3_function(x1,x2):
    if type(x1) != int or type(x2) != int:
        print("Error: only integer allowed")
    else:
        y=x1+x2
        print (y)
```

In [13]:
```
add3_function("50",40)
```

```
Error: only integer allowed
```

In [14]:
```
add3_function(50,40)
```

```
90
```

In [15]:
```
# Example 4:
def full_name(first, middle, last):
    print (f"{first.strip().capitalize()} {middle.upper():.1s} {last.capitali
```

In [16]:
```
full_name("  faaiz  ", 'hadi', 'alshajalee')
```

```
Faaiz H Alshajalee
```

4.4 Unkown inputs: Packing, Unpacking

4.4.1 Tuple inputs

In [17]:
```
# Example 1:
def say_Hi1(a, b):
    people=[a, b]
    for name in people:
        print(f"Hi {name}")
```

In [18]:
```
say_Hi1('Jhon', "Sam")
```

```
Hi Jhon
Hi Sam
```

In [19]:
```
say_Hi1('Jhon', "Sam", "kaf") # 3 names cause error
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-19-fdeea5e127b7> in <module>
----> 1 say_Hi1('Jhon', "Sam", "kaf") # 3 names cause error

TypeError: say_Hi1() takes 2 positional arguments but 3 were given
```

In [20]:
```
# Example 2:
def say_Hi2(*people):
    for name in people:
        print(f"Hi {name}")
```

In [21]:
```
say_Hi2('Jhon', "Sam", "kaf")
```

```
Hi Jhon
Hi Sam
Hi kaf
```

In [22]:
```
# Example 3:
def student(uni, *people):
    for name in people:
        print(f"{name} studies at {uni}")
```

In [23]:
```
student('Curtin University','Jhon', "Sam", "kaf", "Jaf", "Naf")
```

```
Jhon studies at Curtin University
Sam studies at Curtin University
kaf studies at Curtin University
Jaf studies at Curtin University
Naf studies at Curtin University
```

In [24]:
```
# Example 4:
def show_skills(name, *skills):

    print (f"{name} \n - skills without progress:")
    for skill in skills:
        print (f"  -- {skill}")
```

In [25]:
```
show_skills("Sam", "Python", "Matlab", "VB")
```

```
Sam
 - skills without progress:
  -- Python
  -- Matlab
  -- VB
```

In [26]:
```
show_skills("Sam", "Python", "Matlab", "VB", "Mathcad")
```

```
Sam
 - skills without progress:
  -- Python
  -- Matlab
  -- VB
  -- Mathcad
```

4.4.2 Dictionery inputs

In [27]:
```
# Example 1:
def people_info(**info):
    for name, age in info.items():
        print (f"{name} is {age} year old")
```

In [28]:
```
# Method 1:
people_info(Sam= 28, Kaf= 25)
```

```
Sam is 28 year old
Kaf is 25 year old
```

In [29]:
```
# Method 2:
my_info={"Sam": 28, "Kaf": 25}
people_info(**my_info)
```

```
Sam is 28 year old
Kaf is 25 year old
```

4.4.3 Tuple & Dictionery inputs

In [30]:
```
# Example 1:
def show_skills2(name, *skills, ** skills_with_progress):

    print (f"{name} \n  1- Skills without progress:")
    for skill in skills:
        print (f"  --- {skill}")

    print (f" \n  2- Skills with progress:")
    for skill_key, skill_value in skills_with_progress.items():
        print (f"   --- {skill_key} : {skill_value}")
```

In [31]:
```
show_skills2("Sam", "Python", "Matlab", "VB", MathCad="60")
```

```
Sam
  1- Skills without progress:
  --- Python
  --- Matlab
  --- VB

  2- Skills with progress:
   --- MathCad : 60
```

In [32]:
```
show_skills2("Sam", MathCad="60")
```

```
Sam
  1- Skills without progress:

  2- Skills with progress:
   --- MathCad : 60
```

In [33]:
```
show_skills2("Sam", "Python", "Matlab")
```

```
Sam
  1- Skills without progress:
  --- Python
  --- Matlab

  2- Skills with progress:
```

In [34]:
```
myTuple=("Python", "Matlab", "VB")
show_skills2("Sam",*myTuple, MathCad="60")
```

```
Sam
  1- Skills without progress:
  --- Python
  --- Matlab
  --- VB

  2- Skills with progress:
   --- MathCad : 60
```

In [35]:
```
myTuple=("Python", "Matlab", "VB")
my_Dict={"MathCad": 28, "SQL": 25}
```

In [36]:
```
show_skills2("Sam",*myTuple, **my_Dict)
```

```
Sam
  1- Skills without progress:
  --- Python
  --- Matlab
  --- VB

  2- Skills with progress:
   --- MathCad : 28
   --- SQL : 25
```

4.5 Default inputs:

In [37]:
```
# Example 1:
def info(name, age, country):
    print (f"{name} is {age} year old from {country}")
```

In [38]:
```
info("Sam", 28, "Australia")
```

```
Sam is 28 year old from Australia
```

In [39]:
```
# What if the country is unkown, try below
info("Sam", 28)  # This will result in error
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-39-c947c38eb292> in <module>
      1 # What if the country is unkown, try below
----> 2 info("Sam", 28)  # This will result in error

TypeError: info() missing 1 required positional argument: 'country'
```

In [40]:
```
# Example 2:
def info(name, age, country="Unknown"):  # Default input should be the last
    print (f"{name} is {age} year old from {country}")
```

In [41]:
```
info("Sam", 28)
```

```
Sam is 28 year old from Unknown
```

In [42]:
```
# Example 3:
def info(name, age="Unknown", country="Australia"):  # Default input2 should
    print (f"{name} is {age} year old from {country}")
```

In [43]:
```
info("Sam")
```

```
Sam is Unknown year old from Australia
```

4.6 Many outputs

In [44]:
```
# Example 1:
def many_outputs(a, b):
    x=a**2
    y=b**2
    return x, y
```

In [45]:
```
many_outputs(2, 5)
```

Out[45]:
```
(4, 25)
```

In [46]:
```
x,y=many_outputs(2, 5)
print ("x=", x)
print ("y=", y)
```

```
x= 4
y= 25
```

4.7 Variable scope

- Global variables
- Function variables

In [47]:
```
n= 1 # Global variable

def var_scope1():
    print (f" Print the global variable from the function scope: {n}")
```

In [48]:
```
print (f" Print the global variable from the global scope: {n}")
```

```
 Print the global variable from the global scope: 1
```

In [49]:
```
var_scope1()
```

```
 Print the global variable from the function scope: 1
```

In [51]:
```
m= 1 # Global variable. Try to disable it, this will cause error

def var_scope2():
    m=999  # Function scope
    print (f" Print the variable from the function scope {m} : What happens i
```

In [53]:
```
print (f" Print the variable from the global scope {m}")
```

```
 Print the variable from the global scope 1
```

In [54]:
```
var_scope2()
```

```
 Print the variable from the function scope 999 : What happens in Vegas sta
ys in Vegas
```

In [55]:
```
# Convert fucntion scope to Global scope

k= 1  #Global_1

def var_scope3():
    global k #Global_2
    k=999  # Function scope
    print (f" Print the variable from the function scope: {k}. What happens i
```

In [56]:
```
print (f" Print the variable from the global_1: {k}")
```

```
 Print the variable from the global_1: 1
```

In [57]:
```
var_scope3()
```

```
 Print the variable from the function scope: 999. What happens in Vegas sta
ys in Vegas
```

In [58]:
```
print (f" Print the variable from the global_2: {k}. What happens in Vegas wi
```

```
 Print the variable from the global_2: 999. What happens in Vegas will not
stay in Vegas if you tell the glob
```

4.8 Lambda function

```
1 A function without name
2 Can call it inline without defining it
3 Can return data from another fucntion
4 Used for simple fucntion
5 One single expression
```

4.8.1 One input

In [59]:
```
# The normal function
def seq_num_1(num): return num*num
```

In [60]:
```
seq_num_1(3)
```

Out[60]:
```
9
```

In [61]:
```
# Lambda fucntion
#             input: output
seq_num_2=lambda num: num*num
```

In [62]:
```
seq_num_2(3)
```

Out[62]:
```
9
```

4.8.2 Multi inputs

In [64]:
```
Hi=lambda name, age: f"{name} age is {age} years"
Hi("Sam", 20)
```

Out[64]:
```
'Sam age is 20 years'
```

4.8.3 Example: Lambda & Filter

In [67]:
```
# Find the even numbers in the below list
Data_list=[1,2,3,4,5,6,7,8,9,10]
even_num= list(filter(lambda x: x%2==0, Data_list))
even_num
```

Out[67]:
```
[2, 4, 6, 8, 10]
```

4.8.4 Example: Lambda & Sorted

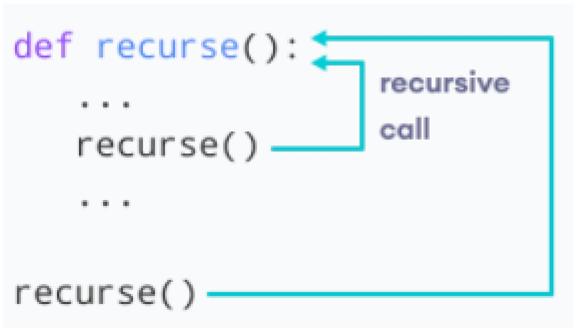
In [68]:
```
# Example 2: sorting data in dictionery based on a column
people=[{"name":"Jan", "age":39},
        {"name":"Fan", "age":25},
        {"name":"Dan", "age":34},
        {"name":"Kan", "age":27}]

sorted(people, key=lambda x: x["age"])
```

Out[68]:
```
[{'name': 'Fan', 'age': 25},
 {'name': 'Kan', 'age': 27},
 {'name': 'Dan', 'age': 34},
 {'name': 'Jan', 'age': 39}]
```

4.9 Function recursion

Reusing a function inside the fucction itself


```
def recurse():
    ...
    recurse()
    ...

recurse()
```

In [69]:
```
def factorial(x):
    """This is a recursive function
    to find the factorial of an integer"""

    if x == 1:
        return 1
    else:
        return (x * factorial(x-1))
```

In [70]:
```
num = 4 # 4!=4*3*2*1
print("The factorial of", num, "is", factorial(num))

# step 1: = 4*f(3),     where f(3)=3*f(2):
# step 2: = 4*3*f(2),   where f(2)=2*f(1):
# step 3: = 4*3*2*f(1),   where f(1)=1
# step 3: = 4*3*2*1
```

```
The factorial of 4 is 24
```

In [71]:
```
# H.W.:
# Given: x="wwoorrdd"
# Requred: Remoev repeated letters from x
```

```
Out[72]: 'word'
```

5 Extra libraries (Preparing stage) [...]