3 Days Training on Python3

Day 1 : Module 4

Muhammad Saufy Rohmad

Module 4 (90 minutes)

Objectives

1. Function in Python
2. Scope and Lifetime of Variables
3. Implementing a Calculator Using Functions

1. Function in Python

- As discussed in the last chapter; when you build an application of any size you will want to break it down into more manageable units; these units can then be worked on separately, tested and maintained separately.
- One way in which these units can be defined is as Python functions

1. Function in Python(2)

- In Python functions are groups of related statements that can be called together, that typically perform a specific task, and which may or may not take a set of parameters or return a value.
- Functions can be defined in one place and called or invoked in another.
- This helps to make code more modular and easier to understand.
- It also means that the same function can be called multiple times or in multiple locations.
- This help to ensure that although a piece of functionality is used in multiple places; it is only defined once and only needs to be maintained and tested in one location

1. Function in Python(2)

- There are two types of functions in Python; built-in functions and user-defined functions.
- Example function

```
def print_msg():
    print('Hello World!')
    print_my_msg('Hello World')
    ---------------------------------------------------
    print_my_msg('Good day')
    print_my_msg('Welcome')
    print_my_msg('Ola')
```

1. Function in Python(3)

- Returning value from function

```
def square(n):
    return n * n
```

1. Function in Python(4)

- Returning multiple values from function

```
def swap(a, b):
    return b, a
```

- Assign the return values

```
a = 2
b = 3
x, y = swap(a, b)
print(x, ',', y)
```

1. Function in Python(5)

- Returning multiple values from function

```
def swap(a, b):
    return b, a
```

- Assign the return values in tuple

```
z = swap(a, b)
print(z) #will print (3,2)
```

1. Function in Python(6)

- Docstring

```
def get_integer_input(message):
    """
    This function will display the message to the user and request that they
    input an integer. If the user enters something that is not a number then
    the input will be rejected and an error message will be displayed. The
    user will then be asked to try again."""
    value_as_string = input(message)
    while not value_as_string.isnumeric():
        print('The input must be an integer')
        value_as_string = input(message)
        return int(value_as_string)
```

1. Function in Python(7)

- Docstring

  print(get_integer_input.__doc__)

- Which Generates

  This function will display the message to the user and request that they input an integer. If the user enters something that is not a number then the input will be rejected and an error message will be displayed. The user will then be asked to try again.

1. Function in Python(8)

- Function Parameters

```
def greeter(name, message):
    print('Welcome', name, '-', message)
```

- Call by:

```
greeter('Eloise', 'Hope you like Rugby’)
```

1. Function in Python(9)

- Default Parameter Values

  def greeter(name, message = 'Live Long and Prosper'):

  print('Welcome', name, '-', message)

- Call by:

  greeter('Eloise')

  greeter('Eloise', 'Hope you like Python')

1. Function in Python(10)

- Named Arguments

```
def greeter(name,title = 'Dr',prompt = 'Welcome',
message = 'Live Long and Prosper'):
    print(prompt, title, name, '-', message)
```

- Call by:

```
greeter(message = 'We like Python', name =
'Lloyd')
```

1. Function in Python(11)

- Arbitrary Arguments

```
def greeter(*args):
    for name in args:
        print('Welcome', name)
```

- Call by:

```
greeter('John', 'Denise', 'Phoebe', 'Adam',
'Gryff', 'Jasmine')
```

1. Function in Python(12)

- Positional and Keyword Arguments

```
def my_function(*args, **kwargs):
    for arg in args:
        print('arg:', arg)
    for key in kwargs.keys():
        print('key:', key, 'has value: ', kwargs[key])
```

- Call by:

```
my_function('John', 'Denise', daughter='Phoebe',
son='Adam')
```

1. Function in Python(13)

- Anonymous Function

  func0 = lambda: print(‘No args’)

  func1 = lambda x:x*x

  func2 = lambda x,y:x*y

  func3 = lambda x,y,x:x+y+z
- Used by:

  func0()

  print(func1(4))

  print(func2(3, 4))

  print(func3(2, 3, 4))
- Output Produce

  no args

  16

  12

  9

2. Scope and Lifetime of Variables

- Local variables

```
def my_function():
    a_variable = 100
    print(a_variable)
```

- Call with:

```
my_function():
```

2. Scope and Lifetime of Variables(2)

- Global Keyword

```
max = 100
def print_max():
    global max
    max = max + 1
    print(max)
print_max()
print(max)
```

2. Scope and Lifetime of Variables(3)

- Non local variables

```
def outer():
    title = 'original title'
    def inner():
        nonlocal title
        title = 'another title'
        print('inner:', title)
    inner()
    print('outer:', title)
outer()
```

3. Implementing a Calculator Using Functions

- The calculator operations

```
def add(x, y):
    """"" Adds two numbers """
    return x + y
def subtract(x, y):
    """ Subtracts two numbers """
    return x - y
def multiply(x, y):
    """ Multiples two numbers """
    return x * y
def divide(x, y):
    """Divides two numbers"""
    return x / y
```

3. Implementing a Calculator Using Functions(2)

- We will need a while loop to determine whether the user has finished or not and a variable to hold the result and print it out

```
finished = False
while not finished:
    result = 0
    # Get the operation from the user
    # Get the numbers from the user
    # Select the operation
    print('Result:', result)
    print('=================')
    # Determine if the user has finished
print('Bye')
```

3. Implementing a Calculator Using Functions(3)

- Identifying User finish or not

```
finished = False
while not finished:
    result = 0
    # Get the operation from the user
    # Get the numbers from the user
    # Select the operation
    print('Result:', result)
    print('==================')
    finished = check_if_user_has_finished(()
print('Bye')
```

3. Implementing a Calculator Using Functions(4)

- Select the operation

```
def get_operation_choice():
    input_ok = False
    while not input_ok:
        print('Menu Options are:')
        print('\t1. Add')
        print('\t2. Subtract')
        print('\t3. Multiply')
        print('\t4. Divide')
        print('-----------------')
        user_selection = input('Please make a selection: ')
        if user_selection in ('1', '2', '3', '4'):
            input_ok = True
        else:
            print('Invalid Input (must be 1 - 4)')
    print('-----------------')
return user_selection
```

3. Implementing a Calculator Using Functions(5)

- Obtain Input Numbers

```
def get_numbers_from_user():
    num1 = get_integer_input('Input the first number: ')
    num2 = get_integer_input('Input the second number: ')
    return num1, num2


def get_integer_input(message):
    value_as_string = input(message)
    while not value_as_string.isnumeric():
        print('The input must be an integer')
        value_as_string = input(message)
    return int(value_as_string)
```

3. Implementing a Calculator Using Functions(6)

- Determining the operation to execute

```
finished = False
while not finished:
    result = 0
    menu_choice = get_operation_choice()
    n1, n2 = get_numbers_from_user()
    if menu_choice == '1':
        result = add(n1, n2)
    elif menu_choice == '2':
        result = subtract(n1, n2)
    elif menu_choice == '3':
        result - multiply(n1, n2)
    elif menu_choice == '4':
        result = divide(n1, n2)
    print('Result:', result)
    print('=================')
    finished = check_if_user_has_finished(()
print('Bye')
```