Functions – User defined

Introduction

- A function is a block of organized, reusable code that is used to perform a single, related action. Functions provides better modularity for your application and a high degree of code reusing.
- As you already know, Python gives you many built-in functions like print() etc. but you can also create your own functions.
- These functions are called *user-defined functions.*

Defining a Function

Here are simple rules to define a function in Python:

- Function blocks begin with the keyword **def** followed by the function name and parentheses (( )).
- Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
- The first statement of a function can be an optional statement - the documentation string of the function or *docstring*.
- The code block within every function starts with a colon (:) and is indented.
- The statement return [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

Defining a Function

- Python keyword cant be used as function name
- Programmer or user can define function by using def *function name*
- Then follow by *parameter* in the parenthesis ()
- Function can be defined with and without arguments or parameter

- **Syntax:**

```
def functionname( parameters ):
    function_statements
return [expression]
```

Defining a Function

```
def function_name):
    statement (must be indent)
call the function
```

```
def hello (): # we define the function as hello
    print ("Hello, World!") # statement
hello() # we call the function
```

Calling a Function

- Following is the example to call printme() function:

```
def printme( str ): #This is a print function
  print (str)

printme("I'm first call to user defined function!")
printme("Again second call to the same function")
```

- This would produce following result:

```
I'm first call to user defined function!
Again second call to the same function
```

Function without argument/parameter

def function_name (): - function without argument / parameter

```
def UTM_slogan (): # we define the function
    print("I am UTM") # state the statement
    print("I Love UTM") # state the statement
    print("UTM My Home") # state the statement
UTM_slogan () # call the function
```

Function with argument/parameter

```
def function_name (parameter1@argument1,
   parameter2@argument2):

def kira_purata (x, y): #define function
    print("purata ",x," dan ",y, " adalah ",(x+y)/2)
   #pass the parameter
kira_purata (100, 12) #call the function
```

The output will be the average of 100 and 12

Function with passing parameter

Return – use to pass value from parameter to the function

```
def nsquare(x, y): # define the function
    return (x*x + 2*x*y + y*y) # calculate and return the value of (x+y)^2
print("The square of the sum of 2 and 3 is : ", nsquare(2, 3)) #call the function
```

Parameter in Order

```
def add_numbers(x, y, z):
    a1 = x + y #state the statement
    a2 = x + z
    a3 = y + z
    print(a1, a2, a3)

add_numbers (30, 100, 4) # pass the value into
    the parameter
```

the process of defining the function and parameter / argu ment

Order

x = 30
y = 100
z = 4

Call function with Parameter

Instruction to call the functi on

print (a1, a2, a3)

add_numbers (30, 100, 4)

pass the value into the function (x, y and z)

Parameter using Keyword Argument

By using keyword argument, the parameter can be called using its name instead of order.

```
def login_info (userID, katalaluan):
    print (“Username: “ + userID)
    print (“Password: “ + katalaluan)

login_info (”Alpha”, “deltaecho”)
login_info (katalaluan = “zuluquebec”, userID =
    “Beta”)
```

Parameter using Keyword Argument

By using keyword argument, the parameter can be called using its name instead of order.

```
login_info (”Alpha”, “deltaecho”) # call by order
login_info (katalaluan = “zuluquebec”, userID =
    “Beta”) # call by name
```

Using Loop in function

```
def contoh_loop ():
    for x in range (0,10):
        print (x)
        if x == 9:
            return
    print ("This is the last line")

contoh_loop()
```

define the function

We want it to stop at 9

Call function

Using if-else in function

```
def names(): # Define function names()
    name = str(input('Enter your name: ‘)) # Set up name variable with input
    # Check whether name has a vowel
    if set('aeiou').intersection(name.lower()):
        print('Your name contains a vowel.')
    else:
        print('Your name does not contain a vowel.')
    # Iterate over name
    for letter in name:
        print(letter)

names() # Call the function
```

The End