LAB # 07

FUNCTIONS

OBJECTIVE

Create python function using different argument types.

THEORY

Functions can be used to define reusable code and organize and simplify code.
Basically, we can divide functions into the following two types:

1. Built-in functions - Functions that are built into Python.
2. User-defined functions - Functions defined by the users themselves.

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

Defining a Function:

A function definition consists of the function's name, parameters, and body. Python allows to define functions to provide the required functionality. 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 code block within every function starts with a colon (:) and is indented.

Syntax:

def functioname (list of parameters):
    Statement

Example:

```
def greet_user(username):
    """Display a simple greeting"""
    print("Hello," , username , "!")
```

Calling a Function:

Calling a function executes the code in the function. If the function returns a value, a call to that function is usually treated as a value.

Example:

```
def greet_user(username):
    """Display a simple greeting"""
    print("Hello," , username , "!")
greet_user('Jesse')
```

Output:

```
>>> %Run task1.py
Hello, Jesse !
>>>
```

Return Value:

The python return statement is used in a function to return something to the caller program. Use the return statement inside a function only.

Example:

```
def xFunction(x, y):
    print("Add", x + y)
    return
xFunction(2, 3)
```

Output:

```
>>> %Run task2.py
Add: 5
>>>
```

When a function doesn't have a return statement and return statement has no value, the function returns ***None.***

Argument Types:

An argument is a piece of information that is passed from a function, When a function is called, it place the value to work with in parentheses.

Following are the ways to call a function by using the following types of formal arguments:

- Required arguments
- Keyword arguments
- Default arguments

Required Arguments:
Required arguments are the arguments passed to a function in correct positional order. Here, the number of arguments in the function call should match exactly with the function definition.

Example:

```
def square(x):
    y=x*x
    return y
x=10
result=square(x)
print("The result of" , x , "squared is", result)
```

Output:

```
>>> %Run task2.py
The result of 10 squared is 100
>>>
```

Keyword Arguments:
Keyword arguments are related to the function calls. When thw keyword argument is used in a function call, the caller identifies the arguments by the parameter name. This allows to skip the arguments or place them out of order because the Python interpreter is able to use the keywords provided to match the values with parameters.

Example:

```
def print_info(name,age):
    "This prints a passed info into this function"
    print("Name:", name)
    print("Age:" , age)
    return

print_info(name ='Mike',age=50)
```

Output:

```
>>> %Run task3.py
Name: Mike
Age: 50
>>>
```

Default Arguments:
A default argument is an argument that assumes a default value if a value is not provided in the function call for that argument.

Example:

```
def print_info(name,age=35):
    "This prints a passed info into this function"
    print("Name:", name)
    print("Age:" , age)
    return

print_info(name ='Mike',age=50)
print_info(name ='Mary')
```

Output:

```
>>> %Run task4.py
Name: Mike
Age: 50
Name: Mary
Age: 35
>>>
```

EXERCISE

A. Point out the errors, if any, and paste the output also in the following Python programs.

1. Code

```
define sub(x, y)
return x + y
```

Output

In this program define is used instead of (def) and there is no identation before return. These are causing an error.

2. Code

```
define describe_pet(pet_name, animal_type='dog'):
    print("\nI have a " , animal_type ,".")
    print("My " , animal_type + "'s name is " , pet_name + ".")
```

Output

In this program define is used instead of (def)

2. Code

```
def type_of_int(i):
    if i // 2 == 0:
        return 'even'
    else:
        return 'odd'
```

Output

```
In this program there is no error.
```

B. What will be the output of the following programs:

1. Code

```
def test(a):
    def add(b):
        a =+ 1
        return a+b
    return add
func = test(4)
print(func(4))
```

Output

```
>>> %Run cp.py

  5

>>>
```

2. Code

```
def return_none():
   return
print(return_none())
```

Output

```
>>> %Run cp.py

  None

>>>
```

3. Code

```
def test_range(n):
```

```
    if n in range(3,9):
        print( n,"is in the range")
    else :
        print("The number is outside the given range.")
test_range(7)
test_range(10)
test_range(4)
```

Output

```
>>> %Run cp.py

  7 is in the range
  The number is outside the given range.
  4 is in the range
```

C. Write Python programs for the following:

1. Write a function called favorite_book() that accepts one parameter, title. The function should print a message, such as One of my favorite books is Alice in Wonderland. Call the function, making sure to include a book title as an argument in the function call.

CODE:

```
#Abdul Moiz Chishti BSE-20F-022
def favourite_book(title):
   print(title,"is my favourite book")
favourite_book("The Great Gatsby")
favourite_book("Charlie and the chocolate Factory")
favourite_book("The Maze Runner")
```

OUTPUT:

```
>>> %Run 'task 1.py'

  The Great Gatsby is my favourite book
  Charlie and the chocolate Factory is my favourite book
  The Maze Runner is my favourite book

>>>
```

2. Write a function called max( ), that returns the maxium of three integer numbers.

CODE:

```
#Abdul Moiz Chishti BSE-20F-022
def max(a, b, c):
   if a>b and a>c:
      print(a)
   elif b>c and b>a:
      print(b)
   else:
```

```
        print(c)
a=int(input("first number="))
b=int(input("second number="))
c=int(input("third number="))
max(a, b, c)
print("is the maximum of three integers")
```

OUTPUT:

```
>>> %Run 'task 2.py'

   first number=78
   second number=90
   third number=164
   164
   is the maximum of three integers

>>>
```

3. Write a Python program to find GCD of two numbers

CODE:

```
#Abdul Moiz Chishti BSE-20F-022
def GCD(a,b):
    if a>b:
        c=a
    else:
        c=b
    for x in range(c,0,-1):
        if a%x==0 and b%x==0:
            return x

a=int(input("Enter any number:"))
b=int(input("Enter second number:"))
gcd=GCD(a,b)
print(gcd,"is the gcd among",a,"and",b)
```

OUTPUT:

```
>>> %Run 'task 3.py'

   Enter any number:56
   Enter second number:78
   2 is the gcd among 56 and 78
```

4. Write a function called describe_city() that accepts the name of a city and its country. The function should print a simple sentence, such as Reykjavik is in Iceland. Give the parameter for the country a default value. Call your function for three different cities, at least one of which is not in the default country.

CODE:

```
#Abdul Moiz Chishti BSE-20F-022
def describe_city(city,country):
    print(city,"is in",country)
describe_city("Karachi","Pakistan")
describe_city("Bonn","Germany")
```

```
describe_city("Moscow","Russia")
```

OUTPUT:

```
>>> %Run 'task 4.py'

  Karachi is in Pakistan
  Bonn is in Germany
  Moscow is in Russia
```