ASSIGNMENT 3

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Question 1. Why are functions advantageous to have in your programs?**

**Ans 1:**

When we define any function example like below:

def add (a,b):

c = a + b

print (c )

I can use the add function multiple times to execute arithmetic addition operation with two variables having their own different defined value.

It is a time saving method to do addition or any other operation defined in that function by just calling the function any number of times. It means that piece of code need not be written again

Hence defining own functions in code are advantageous for programmers to save time and avoid repetition of same operations in code.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Question 2. When does the code in a function run: when it's specified or when it's called?**

**Ans 2:**

When a function is called with or without inputs or arguments inside brackets, then code executes the operations or other commands stated or specified in the function.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Question 3. What statement creates a function?**

**Ans 3:**

For creating a function, def keyword is used followed by any function name with bracket and colon

**Example 1 :**

def add (a,b) :

c = a + b

print(c)

add(4,5)

**Output 1 :**

9

**Example 2:**

def minus (a,b):

c = a -b

print ( c )

minus(4,5)

**Output 2 :**

-1

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Question 4. What is the difference between a function and a function call?**

**Ans 4:**

Function : Defining function in code means to define specific type of operation , execution of different types command , expressions, statements, number of functions like arithmetic, logical, bitwise, assignment, comparison operation between parameters

Function Call : It means we have to pass the parameters as required to execute commands, expressions, statements, number of functions etc mentioned in the defined function.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Question 5. How many global scopes are there in a Python program? How many local scopes?**

**Ans 5:**

Scope is defined as the availability of a variable inside a program, scope is basically the region of code in which a variable is available to use.

**Local Scope :**

Local Scope is created whenever a function is called. In local scope variables are created which execute only inside the function. These local variables exist within defined functions. Outside function , local variable are not accessible.

**Example 1:**

## Understanding the concept of local variable scope ##

def add():

a = 1

b = 10

c = a+b

return c

print("Print the value of a", a)

print("Print the value of b", b)

add()

**Output 1:**  -----------------------------------------------------------------------NameError Traceback (most recent call last)

[<ipython-input-18-d4214d11526d>](https://localhost:8080/#) in <module>

**4** c = a+b

**5** return c

----> 6 print("Print the value of a", a)

**7** print("Print the value of b", b)

**8** add()

NameError: name 'a' is not defined

**Global Scope :**

Global Scope is common for entire code. There is one Global scope in which a global variable is defined and a global variable is accessible both inside or outside a defined function in the entire program. In defined function, Global variable is declared by using keyboard ‘global’

**EXAMPLE 1:**

## Understanding the concept of global variable scope ##

a= 1

b= 10

def add():

c = a+b

return c

print("Print the value of a", a)

print("Print the value of b", b)

sum = add()

print("Print the value of c", sum)

**OUTPUT 1:**

Print the value of a 1

Print the value of b 10

Print the value of c 11

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Question 6. What happens to variables in a local scope when the function call returns?**

**Ans 6:**

In Local scope , a variable is defined as a local variable unless and until it is declared as a global variable.

When a function is called , the interpreter executes the functions, expressions, and local variables within the function. By calling return, after execution , all local variables , function scope out of defined function. Hence it is not accessible. All functions, variables defined in the function are destroyed. Finally, local variables will not have their own value declared in a defined function.

**Example 1:**

## Understanding the concept of local variable scope ##

def add():

a = 1

b = 10

c = a+b

return c

print("Print the value of a", a)

print("Print the value of b", b)

add()

**Output 1:**

---------------------------------------------------------------------------

NameError Traceback (most recent call last)

[<ipython-input-18-d4214d11526d>](https://localhost:8080/#) in <module>

**4** c = a+b

**5** return c

----> 6 print("Print the value of a", a)

**7** print("Print the value of b", b)

**8** add()

NameError: name 'a' is not defined

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Question 7. What is the concept of a return value? Is it possible to have a return value in an expression?**

**Ans 7:**

A return statement is used to end the execution of the function call and “returns” the result (value of the expression following the return keyword) to the caller. A return statement is overall used to invoke a function so that the passed statements can be executed. The statements after the return statements are not executed.

Return statement is a special statement that you can use inside a function or method to send the function’s result back to the caller

**Example 1:**

def cube(x):

r=x\*\*2

return r

cube(2)

**Output 1:**

4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Question 8. If a function does not have a return statement, what is the return value of a call to that function?**

**Ans 8:**

Return value is None.

Let's consider example:

**Example 1 having Return statement:**

def add(a, b):

result = a + b

return result

add(2, 2)

**Output 1 :**

4

**Example 2 having Return statement:**

def add(a, b):

result = a + b

return result

add(2, 2)

**Output 2 :**

Output is Blank

In Example 1 , output is displayed.

In example 2, output is blank it’s undefined or None

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Question 9. How do you make a function variable refer to the global variable?**

**Ans 9:**

To convert a function local variable into a global variable, we have to use the keyword ‘global’ to declare the variable as global variable.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Question 10. What is the data type of None?**

**Ans 10:**

Data type of None is NoneType.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Question 11. What does the sentence import areallyourpetsnamederic do?**

**Ans 11:**

Importing means import module name “ areallyourpetsnamederic “

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Question 12. If you had a bacon() feature in a spam module, what would you call it after importing spam?**

**Ans 12:**

To call the function bacon() from the module , using command spam.bacon().

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Question 13. What can you do to save a programme from crashing if it encounters an error?**

**Ans 13:**

As we know , first interpreter checks the whole programme whether there is error available or not in the programme . Errors may be any error like Syntax Error, Indentation Error, Zero Division Error, ModuleNot Found error( Import Error), I/O Error, Value Error, KeyboardError. If there is an error in code ( or Programme ) , then the interpreter shows an error which is found during execution and then stops execution of the programme.

There is a try except statement is used in the programme to bypass the Errors or save a programme from crashing due to error encounter.

Exception Handling :

An exception is an error that happens during the execution of a program. Exceptions are known to non-programmers as instances that do not conform to a general rule. The name "exception" in computer science has this meaning as well: It implies that the problem (the exception) doesn't occur frequently, i.e. the exception is the "exception to the rule". Exception handling is a construct in some programming languages to handle or deal with errors automatically. Many programming languages like C++, Objective-C, PHP, Java, Ruby, Python, and many others have built-in support for exception handling.

Error handling is generally resolved by saving the state of execution at the moment the error occurred and interrupting the normal flow of the program to execute a special function or piece of code, which is known as the exception handler. Depending on the kind of error ("division by zero", "file open error" and so on) which had occurred, the error handler can "fix" the problem and the programme can be continued afterwards with the previously saved data.

**Example 1:**

# Python program which is having simple runtime error

a = [1, 2, 3]

print ("Second element = %d" %(a[1]))

print ("Fourth element = %d" %(a[3]))

print ("An error occurred")

**Output 1 :**

Second element = 2

---------------------------------------------------------------------------

IndexError Traceback (most recent call last)

[<ipython-input-36-b2d63b28f48e>](https://localhost:8080/#) in <module>

**1** a = [1, 2, 3]

**2** print ("Second element = %d" %(a[1]))

----> 3 print ("Fourth element = %d" %(a[3]))

**4** print ("An error occurred")

IndexError: list index out of range

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Example 2 :**

Error Handling Solution for above Example 1 using try except statement as below:

# Python program to handle simple runtime error

a = [1, 2, 3]

try:

print ("Second element = %d" %(a[1]))

# Throws error since there are only 3 elements in array from index 0 to 2 not having index 3

print ("Fourth element = %d" %(a[3]))

except:

print ("An error occurred")

**Output 2 :**

Second element = 2

An error occurred

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Question 14. What is the purpose of the try clause? What is the purpose of the except clause?**

**Ans 14:**

Try Clause is used to test a block of code for errors

Except Clause is used to handle the error present in try clause.

**Example1:**

def divison( x, y):

final\_value = x / y

return final\_value

divison( 2, 0)

**Output 1 :**

---------------------------------------------------------------------------

ZeroDivisionError Traceback (most recent call last)

[<ipython-input-34-f349cc62c213>](https://localhost:8080/#) in <module>

**2** final\_value = x / y

**3** return final\_value

----> 4 divison( 2, 0)

[<ipython-input-34-f349cc62c213>](https://localhost:8080/#) in divison(x, y)

**1** def divison( x, y):

----> 2 final\_value = x / y

**3** return final\_value

**4** divison( 2, 0)

ZeroDivisionError: division by zero

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Example 2: using try Except statement:**

Error Handling Solution for above example 1 using try Except statement:

def divison1( x, y):

try:

final\_value = x / y

return final\_value

except:

print("error occured")

divison1(2,0)

**Output 2:**

error occured

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*