**Assignment - 3**

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

Functions are advantageous for the following reasons:

|  |  |  |
| --- | --- | --- |
|  | Advantages | Advantages Explained |
| 1 | Modularity | Breakdown a complex problem into simple, manageable & reusable piece of code. So code is easier to read, understand & to maintain. Can focus on individual functions instead of worrying about the entire program |
| 2 | Reusability | Once function is defined, it can be called multiple times from various part of the code: reduces redundancy , makes programming efficient and concise |
| 3 | Abstraction | We can encapsulate a set of operations behind a well defined interface. How function is implemented internally need not be known, only need to know how to use it. Code is simplified and manageable |
| 4 | Debugging and testing | Isolate blocks of code, making it easier to identify and fix bugs. Thus improving the overall quality of the code. |
| 5 | Readability and Maintainability | When use proper or descriptive names for functions, makes it easier for others (& even us on a later date) to understand what each function does leading to better maintenance of the codebase |
| 6 | Scalability | The function breaks the program into smaller, self-contained compartments. As the program grows, we can manage and expand it more effectively. |
| 7 | Code organization | Organize the code logically, similar functions can be grouped together, can easily navigate through the codebase |

Thus functions promote code organization, maintainability and reusability of the code, making it easier to develop, understand and maintain.

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

The code will run only when it is called. The function is first defined (it is specified), then when it is called, it gets executed, giving the result.

Declaring the function

def func():

  print("Will print, only when function is called") # no output

Calling the function

func() # now function is called and results in below output

# Will print, only when function is called

3. What statement creates a function?

The statement that creates a function is function definition or declaration. First we use the keyword ‘def’ (in python), followed by function name, and then may include parameters and the block of code.

def function\_name (parameters):

.......... # Block of code

......... # code

return some\_value

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

|  |  |  |  |
| --- | --- | --- | --- |
| Function | We first define (create) a function | def add\_numbers ( a , b )  return a + b | No result, only declaring a function |
| Function call | We execute the code | add\_numbers (2, 5) | Result (output) is 7 |

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

There is typically only one global scope in a Python program and there are multiple local scopes.

Global scope: it is defined outside the function, can be accessed from anywhere in the program.

Local scopes: the local variables are defined within the function and can be accesses only within the function

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

1. Function call returns: When the function call completes its execution, it returns to the calling code outside the function
2. Local scope destruction: At this point, the local scope associated with the function is destroyed. All the variables that were defined within the local scope are removed from the memory.
3. Variable inaccessibility: Now, if we try to access the local variable from outside the function, it will result in an error since, local variable no longer exists in memory.

Example:

def func():

  local\_variable = 14

  return(local\_variable)

func() # calling the function, output is 14

Now if we call the local variable, it will result in error

print(local\_variable)  # Error: name 'local\_variable' is not defined

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

The return value concept is related to functions in programming. When we call a function, it can produce an output (result) which is referred as ‘return value’ of the function. This allows functions to communicate the result to the calling code for further processing or utilization of the data.

First define a function, and then specify a return statement. When we call the function, we get the return value which can be captured in a variable or use it directly in an expression

Example

def subtract\_number (x,y):

return x – y

subtract\_number (10, 3) # output is 7

We can have a return value in an expression.

For example, in the above function, we can use it in an expression.

print (subtract\_number (10, 3 ) \* 4) # output is 28 (i.e., 10-3 = 7 \*4 = 28)

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

If a function does not have a return statement, the return value of a call to that function will be ‘None’.

Example

def func():

  var =10

result = func() #

print(result) # Output is None

Here, we are calling the function and trying to print the result. The output is ‘None’ indicating that there is no explicit return value.

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

A function variable is local, when it is created within a function. It is global, when it is created outside the function. But to make a local variable global, we have to use the keyword ‘global’.

Example: Making local variable as global by adding ‘global’ keyword

def new():

  global local\_var

  local\_var = 5

new()

print(local\_var) # output 5

10. What is the data type of None?

The data type of None is NoneType. None is a data type of its own. There is only one instance of None in memory. All variables assigned to None, point to the same object.

type(None) # output NoneType

11. What does the sentence import areallyourpetsnamederic do?

We get a Module-not-found-error. It says, no module named areallyourpetsnamederic. This error is raised when the Python interpreter cannot find the module in its search paths.

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

After importing spam module, to call the given function, we have to prefix it with spam. This way we can call the function and execute it. When we import any module , all its classes, functions, variables become accessible using module name as a prefix. This will avoid naming collisions when using functions, classes, variables of different module. It also helps to organize code.

Import spam # importing spam module

spam.bacon( ) # calling bacon feature after importing spam

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

We can use Exception handling techniques to save a program from crashing if it encounters an error. This allows us to handle errors gracefully. It provides a fallback or alternative course of action, when something unexpected happens. If error occurs, when we use exception handling, it continues to execute, thus preventing crashes and providing better user experience.

In Python, we have ‘try’, ‘except’, ‘else’ and ‘finally’ blocks for handling exceptions.

Example

try:

  result = 10 / 2

except ZeroDivisionError: # when zero division error occurs

  print("Error encountered, cannot divide by zero")

except Exception as e: # any other error occurs

  print(f" An unexpected error occured: {e}")

else:

  print(f" The result is {result}. Successful execution of code")

finally:

  print("End of the program")

# Output

Error encountered, cannot divide by zero

End of the program

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

The purpose of the ‘try’ clause is for exception handling. The ‘try’ clause defines a block of code where we expect errors or exceptions to occur. When an exception is raised within the ‘try’ block, program immediately stops executing, jumps to the corresponding ‘except’ block (if there is any) that matches the type of exception. Exception handling will handle errors gracefully and prevents program from crashing.

The purpose of the ‘Except’ clause is to define a block of code to handle specific exceptions raised within a ‘try’ block. When an exception occurs, Python (program) checks if there is a matching ‘except’ block for that exception type. If it finds a matching ‘except’ block, the code inside that ‘except’ block is executed to handle the exception.

Example

try:

  # code that may cause an error

except someException:

  # code to handle the above exception

---------- X ------------