**Python Assignment-3**

**By Abhishek Sachan**

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

* Modularity: Functions allow you to break down your program into smaller, modular pieces, each of which has a specific task or purpose. This makes your code more organized, easier to understand, and easier to maintain.
* Reusability: Once you write a function, you can use it multiple times throughout your program, without having to rewrite the same code each time. This saves time and reduces the risk of errors.
* Abstraction: Functions can be used to abstract away complex or repetitive tasks, allowing you to write simpler, more readable code. This can also make your code more flexible, as you can change the implementation of a function without affecting the rest of your program.
* Encapsulation: Functions allow you to encapsulate data and behavior, which can help to prevent accidental modifications to your program's state. This can make your code more reliable and easier to debug.
* Testing: Functions are easier to test in isolation than large chunks of code. This can help you to catch bugs and errors earlier in the development process, making it easier to fix them before they become bigger problems.

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

When you define a function, you are essentially creating a blueprint or a set of instructions that tell the computer what to do when the function is called. The code inside the function is not executed until the function is called in your program.

For example, consider the following Python function:

Def mutil(a,b): //function definition

Return a\*b

Print multi(5,3) //function calling

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

In python def is used for the function creation

def function\_name():

Pass

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

A function is a block of code that performs a specific task or set of tasks. It is a reusable piece of code that can be called from other parts of a program.

On the other hand, a function call is the act of invoking or executing a function. When you call a function, you are telling the program to run the code inside the function and return the result (if any)

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

In a Python program, there is only one global scope, which is shared by all functions and code outside of functions. This means that any variable defined outside of a function is in the global scope and can be accessed from any part of the program.

On the other hand, local scopes are created each time a function is called. When a function is called, a new local scope is created for that function, which is destroyed when the function returns. This means that any variables defined inside the function are only accessible within that function and are destroyed when the function returns.

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

When a function call returns, the local variables that were defined within the function are destroyed and their values are no longer accessible.

In Python, local variables are created within the local scope of a function and are only accessible within that function. Once the function call returns, the local scope is destroyed and the local variables are removed from memory.

This means that any attempts to access those variables outside of the function after the function call has returned will result in an error.

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

In Python, a return value is the value that a function returns to the caller when it completes its execution. When a function is called, it can perform some operations and computations, and then return a value to the caller. The return value can be used in the calling code to perform further operations or computations.

Def multi(a,b):

Return a\*b

Yes, It is possible to return a expression and variable as well.

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

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

In Python, when a function executes all of its statements and reaches the end of the code block without encountering a **return** statement, it implicitly returns **None**. This means that if you call a function that does not have a **return** statement, the function will still execute, but the result of the function call will be **None**.

For example, consider the following function that does not have a **return** statement:

def print\_hello():  
 print("Hello, world!")

If you call this function:

print\_hello()  
The function will execute and print "Hello, world!" to the console, but it will not return a value. If you try to assign the result of the function call to a variable, the variable will contain **None**:

result = print\_hello()  
print(result) # prints "None"

In general, it's good practice to include a **return** statement in your functions to explicitly return a value, even if it's just **None**. This makes the behavior of your function more clear and can prevent unexpected bugs in your code.

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

In Python, you can use the **global** keyword to declare a variable inside a function as a global variable. This allows you to modify the value of a variable defined in the global scope from inside the function.

X=20

def modify():

global x

X=30

print(modify())

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

In Python, **None** is a built-in constant that represents the absence of a value or the null value. It is often used to indicate that a variable or function argument does not have a value, or to indicate the result of a function that doesn't return anything.

The data type of **None** is **NoneType**. It is a singleton object, which means that there is only one instance of **None** in memory, regardless of how many times it is used in a program.

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

The sentence "import areallyourpetsnamederic" is not a valid Python import statement and would raise a **ModuleNotFoundError**. In Python, the **import** keyword is used to import modules, which are files containing Python code that can be used in other Python programs.

To import a module named **areallyourpetsnamederic**, you would need to have a file with that name and a **.py** extension

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

Import spam

Spam.bacon()

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

1. Use try-except blocks: Wrap the code that might cause an error in a try block and catch the error in an except block. This will allow your program to continue running, even if an error occurs.
2. Validate input: Check user input and make sure it meets the expected format and type before using it in your program. This can help prevent errors caused by unexpected input.

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

The purpose of the **try** clause in Python is to enclose a block of code that may raise an exception. This block of code is executed normally until an exception is raised.

The purpose of the **except** clause is to handle the exception that was raised in the **try** clause. It specifies what should be done when a specific exception occurs. If an exception occurs within the **try** block, Python looks for an **except** block that matches the type of the exception. If a matching **except** block is found, the code in that block is executed to handle the exception.

Here's an example:

try:  
 x = int(input("Enter a number: "))  
 y = 10 / x  
 print("The result is:", y)  
except ValueError:  
 print("Invalid input. Please enter a number.")  
except ZeroDivisionError:  
 print("Cannot divide by zero.")

In this example, the **try** block is used to capture user input and perform a calculation. If the user enters an invalid input or tries to divide by zero, the corresponding **except** block is executed to handle the error.