**FSDS MAY BATCH 2022(Python Basics -3)**

**Submitted by: Shubham Tiwari**

Q1. Why are functions advantageous to have in your programs?

Ans: Functions have several advantages in programming:

1. **Reusability**: Functions can be used multiple times in a program, reducing code duplication and making the code easier to maintain.
2. **Abstraction:** Functions allow you to isolate and encapsulate a specific piece of functionality, making your code more organized and easier to understand.
3. **Modularity**: Functions make it easier to break down a complex problem into smaller, manageable parts that can be worked on independently.
4. **Testing and Debugging**: Functions make it easier to test and debug your code because you can test each function separately, rather than testing the entire program at once.
5. **Better organization**: Functions allow you to organize your code into separate, reusable blocks of code. This makes it easier to maintain and modify your code in the future.

Q2. When does the code in a function run: when it’s specified or when it’s called?

Ans: The code in a function runs when the function is called. The function definition specifies the code that will be executed, but the code is only executed when the function is called in our program.

Q3. What statement creates a function?

Ans: The **def** statement creates a function in Python. Example:

**def my\_function(arg1, arg2):**

**# function code here**

**return result**

Q4. What is the difference between a function and a function call?

Ans: **A function** is a reusable piece of code defined using the **def** keyword in Python. It specifies the code that will be executed when the function is called.

**A function call is** an expression that invokes the function and executes its code. It involves providing the required arguments to the function, which are then processed and used in the execution of the function. The result of the function call can be used or stored for later use.

Q5. How many global scopes are there in a Python program? How many local scopes?

Ans: In a Python program, **there is only one global scope**, but there can be multiple local scopes.

**The global scope** contains all variables that are defined at the top level of the program, outside of any functions or classes. These variables can be accessed from any part of the program.

**Local scopes,** on the other hand, are created whenever a function or a class is defined. Each function or class definition creates its own local scope, which contains the variables defined within that function or class. Local variables can only be accessed within the function or class in which they are defined.

It is possible for a local scope to contain nested local scopes, each with its own set of variables. The innermost scope takes precedence over outer scopes in case of variable name conflicts.

Q6. What happens to variables in a local scope when the function call returns?

Ans: When a function call returns, the local scope that was created for that function call is destroyed, along with all of the variables defined within that scope. These variables are no longer accessible and their values are lost. This is known as the property of function scope being local and ephemeral.

It's important to note that the destruction of the local scope and its variables does not affect variables in the global or other local scopes. Variables defined in a higher scope, such as the global scope, persist even after a function call returns.

Q7. What is the concept of a return value? Is it possible to have a return value in an expression?

Ans: A return value is the value that is returned by a function when it terminates execution. A return value can be any valid expression in the language, including a function call.

**Yes, it is possible** to have a return value in an expression. For example, the return value of a function can be used as part of another expression.

Q8. If a function does not have a return statement, what is the return value of a call to that function?

Ans: If a function does not have a return statement, it returns **None**.

Q9. How do you make a function variable refer to the global variable?

Ans: To make a function variable refer to a global variable, use the "global" keyword before the variable name in the function. Example:

**x = 10**

**def access\_global\_variable():**

**global x**

**print(x)**

**access\_global\_variable()**

**Outputs 10**

Q10. What is the data type of None?

Ans: **None** is a special constant in Python and it's data type is **NoneType**.

Q11. What does the sentence import areallyourpetsnamederic do?

Ans: The sentence "import areallyourpetsnamederic" is not a valid statement in Python and would result in a syntax error. "import" is a keyword used to include a module or library in a Python script, but "areallyourpetsnamederic" is not a recognized module name.

Q12. If you had a bacon() feature in a spam module, what would you call it after importing spam?

Ans: After importing the "spam" module, we would call the bacon() feature as "spam.bacon()".

Q13. What can you do to save a programme from crashing if it encounters an error?

Ans: To prevent a program from crashing when it encounters an error, you can use the following techniques:

1. **Exception handling:** Use try-except blocks to handle exceptions and prevent the program from crashing.
2. **Input validation**: Validate the inputs to ensure that they are of the correct type and within the expected range.
3. **Debugging:** Use debugging tools such as print statements or a debugger to identify and fix errors.
4. **Logging:** Log errors and messages to a file to help you track down the source of problems.
5. **Testing:** Test the program thoroughly to catch any errors before deployment.

Q14. What is the purpose of the try clause? What is the purpose of the except clause?

Ans: The "try" clause is used to specify a block of code that may raise an exception. If an exception occurs within the "try" block, the program execution jumps to the corresponding "except" clause, which contains the code to handle the exception.

The purpose of the "except" clause is to provide a way to handle exceptions that might be raised within the "try" block. The code within the "except" clause is executed when an exception occurs, and it can take appropriate actions, such as logging the error, displaying a message to the user, or returning a default value. The "except" clause is used to handle specific exceptions or catch any exceptions by using the keyword "except Exception as e" where "e" is a variable containing the exception instance.