**Answer 1**

Some functional advantages of programming include: Modularity: Developers can break down complex problems into smaller, reusable functions. This simplifies the development process and makes code more maintainable. Pure functions: Pure functions always produce the same output for the same arguments. Readability: Functional programming uses a declarative style that allows developers to focus on what needs to be done, rather than how it should be done. Scalability: Functional programming's inherent parallelism allows for better use of hardware resources, which can improve performance and scalability

**Answer 2**

When Specified: The commands under run: are executed when the step or task containing the run: block is triggered.The conditions for triggering can vary depending on the tool or framework being used.

**Examples include:**

● Default execution: The step runs if no conditions are specified, meaning it's part of the pipeline sequence.

● Conditional execution: The step runs only if conditions like if:, only:, or when: are satisfied.

2. When Called:

In some frameworks or automation systems, run: executes commands only when the task or step containing it is explicitly invoked

**. For example:**

● Manually triggered workflows: A specific job or workflow containing run: is triggered by a user action.

● Event-driven execution: A specific event (like a code push, merge, or cron job) calls the task containing the run: section.

**Answer 3**

The def keyword is used to create a function in Python: Steps to use def function:

1. Use the def keyword

2. Namethe function

3. Add optional parameters in parentheses

4. Write the code

5. Optionally return a value Here's an example of a function in Python: Example: def my\_function(): print("Hello from a function") 1. Function:

**Answer 4**

1. Function:

● Afunction is a block of reusable code defined using the def keyword or as a lambda expression.

● Itcan take inputs (parameters), execute code, and optionally return outputs

. ● Example:

● defadd(a, b): return a + b 2. Function Cell:

● Whenafunction is defined inside another function (a nested function), and it references variables from the enclosing scope, these variables are stored in cells. This happens because the nested function needs to "remember" the values of these variables even after the outer function has finished executing.

● Thesevariables (in cells) are stored in the \_\_closure\_\_ attribute of the nested function.

**Answer 5**

1. Global Scope:

● Theglobal scope is created when the Python interpreter starts running the program, and it persists until the program ends.

● Variables defined at the top level of the script or module belong to the global scope. 2. Local scopes:

● Eachfunction, method, or block of code that defines its own execution environment has its own local scope

**Answer 6**

When a function call in Python returns, the local scope of that function is destroyed, and all the variables defined within that scope are deallocated. This means:

1. The memory allocated to those variables is freed (if no other references to those objects exist elsewhere in the program).

2. The variables cease to exist—they are no longer accessible once the function call ends.

**Answer 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. The statements after the return statements are not executed. If the return statement is without any expression, then the special value None is returned. A return statement is overall used to invoke a function so that the passed statements can be executed.

**Answer 8**

In Python, if a function does not include a return statement, or the return statement is present without an explicit value (e.g., return by itself), the function automatically returns None. Example: def no\_return\_function(): print("This function has no return statement") result = no\_return\_function() print(result) # Output: None Here, result will be None because no\_return\_function does not have a return statement.

**Answer 9**

In Python, if you want a function variable to refer to a global variable, you can use the global keyword. This allows you to read from and modify the global variable directly from within the function. Key Points About global: ● Without global, any assignment to a variable inside a function creates a local variable by default, even if a global variable with the same name exists. ● Byusing the global keyword, you explicitly tell Python to use the global variable ●

**Answer 10**

In Python, the data type of None is NoneType. Example: value = None print(type(value)) # Output: None None is a singleton object in Python, meaning there is only one instance of it in a Python program. It is used to represent the absence of a value or a null value.

**Answer 11**

The sentence import areallyourpetsnamederic in Python is not valid unless there is a module named areallyourpetsnamederic installed or defined in your environment. Here's an explanation based on context:

1. Standard Behavior of import:

● InPython, import is used to include a module or library so you can use its functions, classes, or variables.

● Forexample, import math allows you to use the math module.

2. Nonexistent Module:

● Ifareallyourpetsnamederic is not an installed module, running this code will result in an error like: ● Example: ModuleNotFoundError: No module named 'areallyourpetsnamederic’

● Tosummarize: unless areallyourpetsnamederic is a real module in your environment, this code will throw an error. If it exists, it will work like any other import statement

**Answer 12**

If you imported the spam module in Python, you would call the bacon() function from the module like this: import spam spam.bacon() Here, spam is the module, and bacon() is the function within the module. You need to prefix the function name with the module name to call it unless you specifically import the function itself, like this: from spam import bacon bacon()

**Answer 13**

To prevent a program from crashing you can use exception handling with a try-except block. This allows your program to catch and handle errors gracefully instead of crashing. Here's how you can do it: Basic Example: try: # Code that might cause an error result = 10 / 0 except ZeroDivisionError: # Handle the specific error Print(“you can't devide by zero!”) Purpose of the try Clause:

**Answer 14**

● Thetry block is used to write code that might raise an exception.

● Itallows the program to attempt to execute the code within it.

● Ifanexception occurs, the program immediately stops executing the try block and looks for a corresponding except block to handle the error.

● Example: try: num=int(input("Enter a number: ")) # This might raise a ValueError if the input is not a number print(f"The number is {num}” Purpose of the except Clause:

● Theexcept block defines how the program should respond to a specific exception.

● It"catches" the exception and prevents the program from crashing.

● Youcanhandle specific exceptions (e.g., ValueError) or handle all exceptions generically using except:

. ● Example: try: num=int(input("Enter a number: ")) print(f"The number is {num}") except ValueError: Print(“you cannot divide by zero!”