# Task 2 : Python

## 1. What is Dynamic typing in Python?

In Python, variables are not bound to a specific type at declaration.Instead, the type is determined at runtime based on the assigned value.This means that a single variable can hold data of different types throughout its lifetime hence making Python flexible and easy to use language for rapid development.

## 2. What is the difference between mutable and immutable data types?

Mutable objects can be modified in place their values can change without creating a new object.

Common mutable types (in Python):

- list
- dict
- set

Immutable objects cannot be changed once created. Any modification creates a new object.

Common immutable types (in Python):

- int
- float
- bool
- str
- tuple
- frozenset

## 3. What are local and global variables?

**Local Variables:**

- Local variables are declared within a specific block of code, such as within a function or a loop.
- They are only accessible within the block in which they are declared.
- Once the block of code in which they are declared exits, the memory allocated to these variables is released, and they are no longer accessible.
- Local variables can have the same name as variables in other blocks without conflict because their scope is limited to the block in which they are declared.

- They are typically used for temporary storage or only relevant data within a specific context.

**Global Variables:**

- Global variables are declared outside of any function or block of code, usually at the top of a program or in a separate file.
- They are accessible from any part of the program, including within functions, loops, or other blocks of code.
- Global variables retain their value throughout the lifetime of the program unless explicitly modified or reset.
- Due to their accessibility from anywhere in the program, global variables can introduce unintended side effects and make it harder to understand and debug code, especially in larger programs.
- They are typically used for values that need to be accessed and modified by multiple parts of the program.

## 4. What is the purpose of the return statement in a function?

The return statement is used inside a function to send a value back to the place where the function was called. Once return is executed, the function stops running, and any code written after it is ignored. If no value is returned, Python automatically returns None.

## 5. What is list comprehension?

List comprehension is a concise and powerful way to create new lists by applying an expression to each item in an existing iterable (like a list, tuple or range). It helps you write clean, readable and efficient code compared to traditional loops.

## 6. Difference between break, continue, and pass

### Break Statement in Python

The break statement in Python is used to exit or “break” out of a loop (either a for or while loop) prematurely, before the loop has iterated through all its items or reached its condition. When the break statement is executed, the program immediately exits the loop, and the control moves to the next line of code after the loop.

**Continue Statement in Python**

Python Continue statement is a loop control statement that forces to execute the next iteration of the loop while skipping the rest of the code inside the loop for the current iteration only, i.e. when the continue statement is executed in the loop, the code inside the loop following the continue statement will be skipped for the current iteration and the next iteration of the loop will begin.

**Pass Statement in Python**

Pass statement in Python is a null operation or a placeholder. It is used when a statement is syntactically required but we don't want to execute any code. It does nothing but allows us to maintain the structure of our program.

## 7. What is recursion? Give a simple example.

The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called a recursive function.

- A recursive algorithm takes one step toward solution and then recursively call itself to further move. The algorithm stops once we reach the solution.
- Since called function may further call itself, this process might continue forever. So it is essential to provide a base case to terminate this recursion process.

  **Steps to Implement Recursion**

  **Step1 - Define a base case:** Identify the simplest (or base) case for which the solution is known or trivial. This is the stopping condition for the recursion, as it prevents the function from infinitely calling itself.

  **Step2 - Define a recursive case:** Define the problem in terms of smaller subproblems. Break the problem down into smaller versions of itself, and call the function recursively to solve each subproblem.

  **Step3 - Ensure the recursion terminates:** Make sure that the recursive function eventually reaches the base case, and does not enter an infinite loop.

  **Step4 - Combine the solutions:** Combine the solutions of the subproblems to solve the original problem.

```
def factorial(n):
```

```
if n == 0:       # base case
    return 1
else:
    return n * factorial(n - 1)   # recursive call
```

## 8. What is a lambda function?

Lambda Functions are anonymous functions means that the function is without a name. As we already know def keyword is used to define a normal function in Python. Similarly, lambda keyword is used to define an anonymous function in Python.

## 9. What is the difference between is and ==?

== (Equality Operator)

- == is used to check value equality.
- It answers the question:
  "Do these two objects contain the same data?"
- Even if two objects are stored in different memory locations, == can return True if their contents are equal.
- The comparison depends on how the data type defines equality.
- Used when you care about what the object represents, not where it is stored.

is (Identity Operator)

- 'Is' is used to check object identity.
- It answers the question:
  "Are these two variables referring to the exact same object in memory?"
- It does not compare values or contents.
- It only checks whether both names point to one single memory location.
- Mainly used for checking singleton objects (objects that exist only once in memory).

## 10. What are docstrings and why are they important?

Docstrings (Documentation Strings) are special strings used to document Python code. They provide a description of what a module, class, function or method does.

- Declared using triple quotes (' ' ' or " " ").
- Written just below the definition of a function, class, or module.

- Unlike comments (#), docstrings can be accessed at runtime using __doc__ or help().

  **Improve code readability**

  Help others (and your future self) understand the purpose of the code.

  **Serve as built-in documentation**

  Tools like help() and documentation generators read docstrings automatically.

  **Make maintenance easier**

  Well-documented code is easier to debug, update, and extend.

  **Support collaboration**

  Team members can understand functions or classes without reading all the code.

  **Encourage good coding practices**

  Writing docstrings forces you to think clearly about what your code should do.