# Module7 | DA-Introduction to python

1. **What are the types of Applications?**

ANS: Types of Applications

1. **Web Applications**
   * **Examples:** Gmail, Google Docs, Facebook, LinkedIn, and YouTube.
2. **Mobile Applications (Mobile Apps)**
   * **Examples:** WhatsApp, Instagram, TikTok, Uber, and Spotify.
3. **Desktop Applications**
   * **Examples:** Microsoft Word, Adobe Photoshop, VLC Media Player, and Visual Studio Code.
4. **Hybrid Applications**
   * **Examples:** Twitter, Uber, Instagram (parts of these apps are built using web technologies)..
5. **Enterprise Applications**
   * **Examples:** SAP, Oracle ERP, Microsoft Dynamics, Salesforce.
6. **Cloud Applications**
   * **Examples:** Google Drive, Dropbox, Microsoft OneDrive, Slack.
7. **Embedded Applications**
   * **Examples:** Firmware in a smart TV, software in an ATM machine, automotive control systems.
8. **Gaming Applications**
   * **Examples:** Fortnite, Minecraft, Call of Duty, Candy Crush.

**2) What is programing?**

**Programming** is the process of writing instructions (called code) that a computer can understand and execute to perform specific tasks. These instructions are written in programming languages like Python, Java, or C++.

**3) What is Python?**

**Python** is a high-level, interpreted programming language known for its simplicity and readability. It was created by Guido van Rossum and first released in 1991. Python is widely used for web development, data analysis, artificial intelligence, scientific computing, automation, and more.

**7)How memory is managed in Python?**

**Automatic Memory Management**: Python automatically handles memory allocation and deallocation using reference counting and garbage collection.

**Reference Counting**: Each object has a reference count that tracks how many references point to it. When the count reaches zero, the memory is freed.

**Garbage Collection**: Python uses a cyclic garbage collector to clean up objects involved in reference cycles, and it uses a generation-based approach to manage memory efficiently:

* **Generation 0**: New objects.
* **Generation 1**: Objects that survived one collection.
* **Generation 2**: Objects that survived multiple collections.

**Memory Pools**: Python manages small objects using memory pools and arenas to reduce fragmentation and improve efficiency.

**Memory Tracking**: Tools like sys.getsizeof(), memory\_profiler, and tracemalloc can help track and analyze memory usage

**8) What is the purpose continuing statement in python?**

The **continue** statement in Python skips the rest of the code in the current loop iteration and proceeds to the next iteration. It helps control the loop flow by bypassing certain conditions and continuing with the next loop cycle.

**17) What are negative indexes and why are they used?**

**Definition**: In Python, negative indexing starts from the end of a sequence, with the last element having an index of -1, the second last element having an index of -2, and so on.

**Usage**: Negative indexes allow you to easily access elements from the end of a sequence without needing to calculate the position based on the length of the sequence.

**25) What is List? How will you reverse a list?**

A list in Python is an ordered, mutable collection of items, which can be of different types. Lists are defined using square brackets [].

How to Reverse a List:

1. **Using reverse() Method**:
   * **In-place**: Modifies the original list.

my\_list = [1, 2, 3]

my\_list.reverse()

1. **Using Slicing**:
   * **Creates a new reversed list**: Does not modify the original list.

my\_list = [1, 2, 3]

reversed\_list = my\_list[::-1]

1. **Using reversed() Function**:
   * **Creates an iterator**: Can be converted to a list.

my\_list = [1, 2, 3]

reversed\_list = list(reversed(my\_list))

**26)How will you remove last object from a list?**

**Using pop() Method**:

* **Removes and returns** the last item of the list.

my\_list = [1, 2, 3, 4]

my\_list.pop()

print(my\_list) # Output: [1, 2, 3**]**

**27)Suppose list1 is [2, 33, 222, 14, and 25], what is list1 [- 1]?**

For a list list1 = [2, 33, 222, 14, 25]:

* list1[-1] refers to the last element of the list.

So, list1[-1] would give you 25.

**28) Differentiate between append() and extend () methods?**

* **append()**: Adds its argument as a single element to the end of the list.
* **extend()**: Iterates over its argument, adding each element to the list.

**30)How will you compare two lists?**

* **Equality (==)**: Checks if the lists contain the same elements in the same order.
* **Identity (is)**: Checks if the lists are the same object in memory.
* **Set Operations**: Useful for checking subsets, intersections, or differences between lists.

**43) What is tuple? Difference between list and tuple.**

A tuple is an ordered, immutable collection of elements in Python, defined using parentheses ().

**Differences Between List and Tuple**

**Mutability**:

**List**: Mutable (can be changed).

**Tuple**: Immutable (cannot be changed).

**Syntax**:

**List**: Defined with square brackets [].

**Tuple**: Defined with parentheses ().

**Performance**:

**List**: Slightly slower and uses more memory.

**Tuple**: Generally faster and uses less memory.

**Use Case**:

**List**: For collections where changes are expected.

**Tuple**: For fixed collections that should not be modified.

**Methods**:

**List**: Many methods (e.g., append(), remove()).

**Tuple**: Few methods (e.g., count(), index())

**51) How Do You Traverse Through a Dictionary Object in Python?**

**My\_dict**: the dictionary you want to traverse.

**for key in my\_dict**: loops through each key.

**my\_dict.values()**: returns a view object of all the values.

**my\_dict.items()**: returns a view object of all the key-value pairs.

**65) How Many Basic Types of Functions Are Available in Python?**

**Built-in Functions**: Provided by Python and available by default.

**User-defined Functions**: Created by the user using the def keyword.

**Lambda Functions**: Small anonymous functions created using the lambda keyword.

**71) What is File function in python? What are keywords to create and write file.**

In Python, the file function was used in Python 2 to create and handle files. However, in Python 3, this function has been replaced by the open() function, which is more versatile and widely used for file operations

* Key Modes
  + 'r' : Read (default)
  + 'w' : Write (creates or overwrites)
  + 'a' : Append (adds to the end)
  + 'b' : Binary mode (e.g., 'rb' for binary read

**83) Explain Exception handling? What is an Error in Python?**

**Exception handling** in Python is a mechanism that allows you to manage runtime errors (exceptions) gracefully. Instead of letting the program crash when an error occurs, you can catch and handle exceptions, ensuring that your program can recover or provide useful error messages

**Errors** in Python are problems that occur during the execution of a program.

**84) How many except statements can a try-except block have? Name Some built-in exception classes:**

A try block can be followed by multiple except blocks to handle different types of exceptions. Python allows you to have as many except blocks as needed. Each except block can handle a specific exception type, providing tailored error handling for different scenarios.

**85) When will the else part of try-except-else be executed?**

The **else** part of a **try-except-else** block in Python is executed when no exceptions are raised in the try block. Here’s how it works:

1. **try Block**: Contains the code that may potentially raise an exception.
2. **except Block**: Contains code to handle any exceptions raised by the try block.
3. **else Block**: Contains code that is executed if the try block does not raise any exceptions.

### Execution Flow

* **If an exception occurs** in the try block, the corresponding except block is executed, and the else block is skipped.
* **If no exception occurs** in the try block, the else block is executed.

**86) Can one block of except statements handle multiple exception?**

Yes, a single except block can handle multiple exceptions in Python. You can specify multiple exceptions in a tuple, allowing the except block to handle any of the exceptions listed. This is useful when you want the same handling logic for different types of exceptions.

**87) When is the finally block executed?**

The **finally** block in a try-**except-finally** construct is executed **regardless of whether an exception is raised or not**. It is used for cleanup actions that must occur whether or not an error occurs in the try block. This ensures that resources are properly released or other necessary final steps are completed

**88) What happens when „1‟== 1 is executed?**

When the expression **'1' == 1** is executed in Python, it evaluates to **False**. This is because '1' is a string and 1 is an integer, and Python does not consider them to be equal.

**89) How Do You Handle Exceptions with Try/Except/Finally in Python?**

**try Block**: Contains code that might raise an exception.

**except Block**: Catches and handles exceptions that occur in the try block.

**finally Block**: Executes code that must run no matter what, whether an exception occurred or not.