**Python Assignment**

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

Web Applications – Django, Flask, FastAPI

Desktop GUI Applications – Tkinter, PyQt, Kivy

Data Science & Machine Learning – NumPy, Pandas, TensorFlow

Game Development – Pygame, Panda3D, Godot

Automation & Scripting – Selenium, BeautifulSoup, OS module

Command Line Applications – Argparse, Click, Typer

Blockchain & Cryptocurrency – Web3.py, Ethereum, Hyperledger

IoT Applications – Raspberry Pi (GPIO), MQTT, Arduino

Cybersecurity & Ethical Hacking – Scapy, Pwntools, Socket

Cloud & DevOps – Boto3 (AWS SDK), Docker SDK, Ansible

**2.What is programming?**

Programming is the process of writing instructions that a computer can execute to perform specific tasks. It involves using a programming language to create software, automate processes, and solve problems. Programming enables the development of applications, websites, games, and systems by giving computers step-by-step commands. It involves concepts like logic, algorithms, data structures, and debugging to ensure efficient and error-free execution.

**3.What is Python?**

Python is a high-level, interpreted programming language known for its simplicity and readability. It is widely used in various fields, including web development, data science, artificial intelligence, automation, and cybersecurity. Python supports multiple programming paradigms, such as procedural, object-oriented, and functional programming. It has a vast standard library, making it powerful and versatile for different applications. Python’s dynamic typing and ease of syntax make it beginner-friendly while remaining efficient for complex software development.

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

Memory Allocation – Python divides memory into stack (for function calls & local variables) and heap (for objects & dynamic memory).

Private Heap Space – Python manages memory using a private heap that stores all objects.

Reference Counting – Objects are deleted when their reference count becomes zero.

Garbage Collection – Python’s garbage collector automatically removes unused objects.

Memory Optimization – Python reuses small objects, uses interning for strings, and provides **\_\_slots\_\_** to reduce memory usage.

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

The continue statement in Python is used inside loops to skip the current iteration and move to the next iteration without executing the remaining code in the loop body.

**Purpose of continue Statement:**

Skips specific iterations based on a condition.

Avoids unnecessary execution of certain code inside loops.

Improves efficiency by controlling loop flow dynamically.

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

In Python, negative indexing allows accessing elements from the end of a sequence (like lists, strings, or tuples). Instead of counting from the beginning (0, 1, 2...), negative indexes count from the end (-1, -2, -3...).

Use of Negative Indexing

Easier access to last elements without needing len() function.

Convenient for slicing from the end.

Avoids IndexErrors when working with unknown-length sequences.

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

list is a datatype where you can store multiple values/items under single name, It is more like dynamic arrays which means you can add more items when needed.

**Characteristics of list**

Ordered, Mutable/changeable, Hetrogeneous, can have duplicates, dynamic, can be nested, items can be accessed

reversing string->

1) Using reverse() method – Modifies the original list in place.

2) Using slicing ([::-1]) – Creates a new reversed list without modifying the original.

3) Using reversed() function – Returns an iterator that can be converted into a list.

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

**Ways to Remove the Last Object from a List in Python**

**1) Using pop() (Recommended)**

Removes and returns the last element from the list.

Modifies the original list.

**2) Using Slicing ([:-1])**

Creates a new list without the last element.

Does not modify the original list.

**3) Using del Statement**

Deletes the last element in place.

Modifies the original list.

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

Difference Between append() and extend() in Python Lists

**append() Method:**

Adds a single element to the end of the list.

If you pass a list, it adds the whole list as one element, not merging its contents.

**extend() Method:**

Adds each element of an iterable (like a list or tuple) individually to the list.

Merges the contents instead of adding them as a single element.

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

###Ways to Compare Two Lists in Python

**1) Using == (Equality Operator) :** Checks if both lists have the same elements in the same order.

**2) Using sorted() :** Compares lists regardless of order by sorting them first.

**3) Using set() :** Compares lists as unordered collections, ignoring duplicates.

**4) Using Iteration (for loop or zip()) :** Compares element by element for a custom comparison.

**5) Using collections.Counter() :** Checks if both lists have the same elements with the same frequency, ignoring order.

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

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

Difference between list and tuple

**List:**

mutable

Defined using []

Uses more memory

methods:append, remove, sort, etc

Used when data needs to change

**Tuple:**

Immutable

Defined using ()

Uses less memory

methods:count, index

Used for fixed data like coordinates, dates, etc.

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

**Traversing a Dictionary in Python**

Traversing a dictionary means iterating through its keys, values, or both. You can do this using loops and dictionary methods.

**Methods to Traverse a Dictionary:**

Iterating Through Keys → Using a for loop directly on the dictionary or .keys().

Iterating Through Values → Using the .values() method.

Iterating Through Key-Value Pairs → Using the .items() method.

Using enumerate() → To get an index along with key-value pairs.

Using a while Loop → Converting keys to a list and iterating using an index.

**52.How Do You Check the Presence of a Key in A Dictionary?**

**Ways to Check the Presence of a Key in a Dictionary**

Using the 'in' Operator → The most efficient way to check if a key exists in a dictionary.

Using the '.get()' Method → Returns None if the key is not found, avoiding errors.

Using 'Exception Handling' (try-except) → Catches a KeyError when trying to access a missing key.

**66.How can you pick a random item from a list or tuple?**

Using the random.choice() function from the random module

**68.How can you get a random number in python?**

Use random.random() to get a random float between 0 and 1, or other random module functions for different ranges.

**69.How will you set the starting value in generating random numbers?**

Use random.seed(value) to initialize the random number generator with a specific seed value.

**70.How will you randomize the items of a list in place?**

To randomize the items of a list in place, you can use the random.shuffle() function from the random module. This function rearranges the elements of the list in a random order directly (in place), without returning a new list.

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

In Python, files are handled using built-in functions. These functions allow you to open, read, write, and manipulate files.

**Common File Functions in Python:**

**->open():**

Used to open a file and return a file object.

Syntax: file = open("filename", "mode")

Modes:

**'r' – Read (default mode, opens the file for reading).**

**'w' – Write (creates a new file or truncates an existing file).**

**'a' – Append (opens a file for appending new content).**

**'b' – Binary mode (used for non-text files like images).**

**'x' – Exclusive creation (fails if the file already exists).**

**'t' – Text mode (default, handles text files).**

**->read():**

Reads the entire content of the file.

Syntax: content = file.read()

**->write():**

Writes a string to the file.

Syntax: file.write("text")

**->close():**

Closes the file when done.

Syntax: file.close()

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

**Exception Handling in Python**

Exception handling is a mechanism in Python used to handle runtime errors and prevent a program from crashing unexpectedly. It allows the program to catch errors and take appropriate actions instead of stopping execution.

**Key Keywords in Exception Handling:**

1.try → The block where we write code that may cause an exception.

2.except → Handles the error that occurs inside the try block.

3.else → Executes if no exception occurs.

**Types of Errors in Python**

1.Syntax Errors → Errors due to incorrect Python syntax.

2.Indentation Errors → Occur when the indentation (spacing) is incorrect.

3.Type Errors → Happen when an operation is performed on incompatible data types.

4.Name Errors → Occur when a variable or function is not defined.

5.Index Errors → Raised when trying to access an invalid index in a list or tuple.

6.Key Errors → Occur when accessing a non-existing key in a dictionary.

7.Attribute Errors → Raised when calling an invalid method or attribute on an object.

8.Value Errors → Occur when a function receives an argument of the correct type but an invalid value.

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

A try-except block can have multiple except statements, each handling a different type of exception. There is no strict limit on the number of except blocks, but each should handle a specific exception type.

**Some built-in classes:**

SyntaxError → Raised when there is a syntax mistake in the code.

IndentationError → Raised for incorrect indentation.

TypeError → Occurs when an operation is performed on an incompatible data type.

ValueError → Raised when a function receives an argument of the right type but invalid value.

IndexError → Occurs when accessing an invalid index in a list or tuple.

KeyError → Raised when trying to access a non-existing key in a dictionary.

ZeroDivisionError → Raised when dividing a number by zero.

FileNotFoundError → Raised when a file does not exist.

AttributeError → Raised when calling an invalid attribute or method on an object.

ImportError → Raised when an imported module is not found.

RuntimeError → Raised when a generic runtime error occurs.

NameError → Raised when using an undefined variable.

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

The else block in a try-except-else statement is executed only if no exception occurs in the try block.

**Flow of Execution:**

1.The try block runs.

2.If no exceptions occur, the else block executes.

3.If an exception occurs, the except block handles it, and the else block is skipped

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

Yes! A single except block can handle multiple exceptions by specifying them in a tuple

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

The finally block is always executed, regardless of whether an exception occurs or not.

**Key Points:**

1.The finally block runs after the try-except blocks.

2.It executes even if an exception is raised and not handled.

3.Used for cleanup operations like closing files, releasing resources, or disconnecting from databases.

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

**Evaluates to False because:**

**Different Data Types**

"1" is a string (str type).

1 is an integer (int type).

Python does not automatically convert one type to another in comparison.

**Strict Equality Check (==)**

The == operator in Python checks for both value and data type.

Since a string and an integer are not the same type, Python returns False.