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

Below are some common types of applications:

Desktop Applications

Web Applications

Mobile Applications

Cloud Applications

1. **What is programing?**

Programming is the process of designing and creating instructions that a computer can follow to perform specific tasks.

1. **What is Python?**

Python is a high-level, versatile, and easy-to-learn programming language. It is widely used across industries and disciplines due to its simplicity, readability, and powerful libraries.

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

Memory management in Python is handled automatically by the **Python Memory Manager**. Python provides a dynamic and efficient memory management mechanism to optimize the use of resources.

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

The continue statement in Python is used to **skip the rest of the code inside the current iteration of a loop** and proceed directly to the next iteration.

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

**negative indexes** are used to access elements in a sequence from the **end** rather than from the beginning.

 s[-1] → last element

 s[-2] → second to last element

 s[-3] → third to last element, and so on.

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

A **list** in Python is an ordered collection of items. Lists are one of the most versatile data structures in Python, and they allow you to store multiple items in a single variable. Lists are mutable, meaning you can change the content of a list after it's been created.

There are multiple ways to reverse a list in Python. Here are the most common ones:

#### 1. ****Using the**** reverse() ****method****:

The reverse() method reverses the elements of the list **in place**

#### 2. ****Using Slicing****:

You can use Python's slicing technique to reverse a list. This method returns a new reversed list without modifying the original one.

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

### ****Using**** pop() ****method****:

The pop() method removes and returns the last element from the list. If no index is provided, it removes the last element by default.

### ****Using**** del ****statement****:

You can use the del statement to remove an element by specifying its index. To remove the last element, you can use the index -1.

### ****Using Slicing****:

You can also use slicing to remove the last element by creating a new list that excludes the last element.

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

list1 = [2, 33, 222, 14, 25]

print(list1[-1])

list1[-1] will access the **last element** of the list, which is 25.

1. **Differentiate between append () and extend () methods?**

**1. append() Method**

Adds a single element to the end of the list. The append() method takes **one argument**, which can be any object, and adds it as a single item to the list.

**2. extend() Method**

Adds **each element** of an iterable (like a list, tuple, or string) to the end of the list.

The extend() method takes an iterable as an argument (like another list, tuple, or string) and adds its elements to the end of the list.

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

### ****Using the**** == ****Operator****

The == operator compares two lists element by element. It returns True if the lists have the same length and contain; otherwise, it returns False.

### ****Using the**** != ****Operator****

It checks if the lists are not equal, and returns True if they are different in length, order, or content.

### ****Using the**** sort() ****Method****

If you want to compare two lists but do not care about the order of the elements, you can **sort** both lists first and then compare them.

### ****Using**** set() ****for Comparing Lists without Duplicates****

If you want to compare two lists and do not care about the number of duplicates, you can convert both lists to sets and compare the sets.

### ****Using the**** all() ****Function for Element-wise Comparison****

You can also use the all() function to check if all elements in two lists are the same in order and value.

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

A **tuple** is a collection data type in Python that is **ordered**, **immutable**, and **can store multiple items** in a single variable. Tuples are similar to lists but have the key difference that **once created, their values cannot be modified**.

1. **How will you create a dictionary using tuples in python?**

You can use the dict() constructor to convert a list of tuples into a dictionary. If you have individual tuples, you can add them directly as key-value pairs in a dictionary.You can use a dictionary comprehension to create a dictionary from a list of tuples.

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

Traversing through a dictionary in Python can be done in multiple ways depending on what you want to iterate over (keys, values, or key-value pairs).

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

To check the presence of a key in a dictionary in Python, you can use one of the following methods:

**Using the in Operator**

**Using the get() Method**

**Using dict.keys()**

**Using Exception Handling**

1. **How Many Basic Types of Functions Are Available in Python?**

**Built-in functions**:

* These functions are predefined in Python and are available for use without any need for importing additional libraries.
* Examples of built-in functions include:
  + print()
  + len()
  + type()
  + int(), str(), list(), etc.

**User-defined functions**:

* These are functions that you define yourself to perform specific tasks in your program.
* You define them using the def keyword.

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

To pick a random item from a list or tuple in Python, you can use the random module, which provides a function called choice() to select a random element.

1. **How can you pick a random item from a range?**

To pick a random item from a range in Python, you can use the random.choice() method in combination with the range() function.

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

In Python, you can generate random numbers using the random module.

1. **Random IntegerRandom**
2. **Floating-point Number**
3. **Random Floating-point Number in a Specified Range**
4. **Random Number from a List**
5. **How will you set the starting value in generating random numbers?**

To set the starting value for generating random numbers, you use a **seed**. A seed initializes the random number generator .

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

You can use the random.shuffle() method from the random module

random.shuffle() modifies the list directly and does not return a new list.

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

**file functions** are methods used to work with files, such as creating, opening, reading, writing, and closing files.

|  |  |
| --- | --- |

|  |  |
| --- | --- |
| 'r' | Read mode. Opens the file for reading (default). |

|  |  |
| --- | --- |
| 'w' | Write mode. Creates or overwrites a file. |

|  |  |
| --- | --- |
| 'a' | Append mode. Adds content to the end of a file. |

|  |  |
| --- | --- |
| 'x' | Exclusive creation. Fails if the file already exists. |

|  |  |
| --- | --- |
| 'b' | Binary mode. |

|  |  |
| --- | --- |
| 't' | Text mode. Default mode for text files. |

|  |  |
| --- | --- |
| 'r+' | Read and write mode. |

|  |  |
| --- | --- |
| 'w+' | Write and read mode (overwrites existing content). |

|  |  |
| --- | --- |
|  | . |
|  |  |
|  |  |

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

* **Exception handling** in Python allows you to manage and respond to unexpected situations (errors) that occur during the execution of a program. Instead of letting the program crash, you can handle these exceptions gracefully, often by using the try, except, else, and finally blocks.

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

**Number of except Statements in a try-except Block**

A try block in Python can have **multiple except statements** to handle different types of exceptions.

**Some Built-in Exception Classes in Python**

* Here are some commonly used built-in exception classes:

| **Exception Class** | **Description** |
| --- | --- |
| ArithmeticError | Base class for errors in numeric calculations. |
| ZeroDivisionError | Raised when dividing by zero. |
| ValueError | Raised when a function receives an argument of the correct type but invalid value. |
| TypeError | Raised when an operation or function is applied to an object of inappropriate type. |
| IndexError | Raised when accessing a list/tuple index that is out of range. |
| KeyError | Raised when accessing a dictionary key that does not exist. |
| FileNotFoundError | Raised when trying to access a file that doesn’t exist. |

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

The **else** part of a try-except-else block will be executed **only if no exception occurs in the try block**.

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

Yes, **one except block** can handle **multiple exceptions** in Python. This is achieved by specifying the exceptions as a **tuple** in the except statement.

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

The **finally block** in Python is **always executed** after the try block. It is used for cleanup actions, such as releasing resources, closing files, or other necessary finalization tasks.

1. **What happens when “1‟== 1 is executed?**

When the expression "1" == 1 is executed in Python, it evaluates to **False** because the two values being compared are of **different types**:

1. "1" is a **string**.
2. 1 is an **integer**.

The comparison "1" == 1 returns False because Python does not implicitly convert the types when comparing values with the == operator.