**Interview Questions:**

**Python :**

31. Name some common built-in exceptions in Python.

Solution-Python provides a variety of built-in exceptions to cover a wide range of error conditions that may occur during program execution

1. **SyntaxError**: Raised when the Python parser encounters a syntax error in the code.
2. **IndentationError**: Subclass of **SyntaxError**, raised when there is incorrect indentation in the code.
3. **NameError**: Raised when a local or global name is not found.
4. **TypeError**: Raised when an operation or function is applied to an object of an inappropriate type.
5. **ValueError**: Raised when a function receives an argument of the correct type but with an inappropriate value.
6. **IndexError**: Raised when a sequence subscript is out of range.
7. **KeyError**: Raised when a dictionary key is not found in the set of existing keys.
8. **FileNotFoundError**: Raised when a file or directory is requested but cannot be found.
9. **IOError**: Base class for I/O-related errors.
10. **PermissionError**: Raised when trying to perform an operation without the adequate permissions.
11. **ZeroDivisionError**: Raised when division or modulo by zero occurs.
12. **AttributeError**: Raised when an attribute reference or assignment fails.
13. **ImportError**: Raised when an import statement fails to find the module, or when a **from ... import** fails.
14. **KeyboardInterrupt**: Raised when the user interrupts the execution of the program, typically by pressing Ctrl+C.
15. **EOFError**: Raised when the built-in **input()** function hits an end-of-file condition (EOF) without reading any data.

32. Describe what regular expressions (regex) are used for.

Solution-   
Regular expressions, often abbreviated as regex or regexp, are powerful tools for pattern matching and text manipulation. They provide a concise and flexible means of searching, extracting, and manipulating strings based on specified patterns.

33. How do you use regular expressions in Python?

Solution- In Python, you can use regular expressions by importing the built-in **re** module. The **re** module provides functions and classes for working with regular expressions.

* **re.match(pattern, string)**: Attempts to match the pattern at the beginning of the string.
* **re.search(pattern, string)**: Searches for the first occurrence of the pattern in the string.
* **re.findall(pattern, string)**: Returns a list of all occurrences of the pattern in the string.
* **re.finditer(pattern, string)**: Returns an iterator yielding match objects for all occurrences of the pattern in the string.
* **re.sub(pattern, replacement, string)**: Substitutes occurrences of the pattern in the string with the replacement string.
* **re.split(pattern, string)**: Splits the string at occurrences of the pattern and returns a list of substrings.

34. Explain the functions of `re.match()`, `re.search()`, and `re.findall()`.

Solution-

* **re.match(pattern, string)**: Attempts to match the pattern at the beginning of the string.
* **re.search(pattern, string)**: Searches for the first occurrence of the pattern in the string.

**re.findall(pattern, string)**: Returns a list of all occurrences of the pattern in the string

35. What are user-defined functions in Python?

Solution- User-defined functions in Python are functions that you create yourself to perform specific tasks within your Python programs. These functions are defined by the user (you) rather than being built into Python or provided by external libraries.

36. How are lambda functions different from regular functions?

Solution- Lambda functions, also known as anonymous functions, are a concise way of creating small, one-line functions in Python. They are defined using the **lambda** keyword instead of the **def** keyword used for regular functions.

37. Name some built-in functions in Python and explain their usage.

Solution- Here are some commonly used built-in functions in Python along with their usage:

**print()**

**input()**

**len()**

**range()**

**sum()**

**min() and max()**

**abs()**

**sorted()**

**enumerate()**

38. Describe the purpose of libraries and packages in Python.

Solution-   
Libraries and packages in Python are essential components of the Python ecosystem that help extend its functionality and provide additional tools and resources for developers

39. How do you install and import external libraries in Python?

Solution-   
To install external libraries in Python, you typically use a package manager like pip (Python's package installer). Once installed, you can import these libraries into your Python scripts or interactive sessions to use their functionalities. Here's a step-by-step guide:

### Installing External Libraries with pip:

1. **Open a Terminal or Command Prompt**: On your computer, open a terminal or command prompt window.
2. **Install the Library**: Use the following command to install the library using pip. Replace **library\_name** with the name of the library you want to install.

Copy code

pip install library\_name

For example, to install the popular library **numpy**, you would run:

Copy code

pip install numpy

This command will download and install the specified library and its dependencies from the Python Package Index (PyPI).

### Importing External Libraries in Python:

Once the library is installed, you can import it into your Python scripts or interactive sessions using the **import** statement. Here's how:

1. **Open a Python Script or Start an Interactive Session**: Open a Python script in your text editor or start an interactive Python session (e.g., using IDLE, Jupyter Notebook, or a Python REPL).
2. **Import the Library**: Use the **import** statement followed by the name of the library to import it into your script or session. You can optionally use an alias to refer to the library by a different name.

python

Copy code

import library\_name

or with an alias:

python

Copy code

import library\_name as alias

For example, to import the **numpy** library:

python

Copy code

import numpy

or with an alias:

python

Copy code

import numpy as np

1. **Use the Library**: Once imported, you can use the functions, classes, and other components provided by the library in your code.

python

Copy code

# Example using numpy import numpy as np # Create an array arr = np.array([1, 2, 3, 4, 5]) # Perform operations on the array print(np.mean(arr)) # Calculate the mean

That's it! You've successfully installed an external library using pip and imported it into your Python script or interactive session. You can now use the library's functionalities in your code.

Top of Form

40. Explain the characteristics of lists in Python.

Solution- Lists in Python are ordered collections of items, which can be of any data type, including integers, floats, strings, or even other lists. Lists are mutable, meaning you can modify their contents after creation

41. What are tuples and how are they different from lists?

Solution- Tuples in Python are similar to lists in that they are ordered collections of items

* Tuples, on the other hand, are immutable, meaning their contents cannot be changed after creation. Once a tuple is created, its elements cannot be modified, added, or removed.

42. Describe the properties of sets and dictionaries in Python.

### Solution- Sets:

1. **Unordered Collection**: Sets are unordered collections of unique elements. This means that sets do not maintain the order of elements, and duplicate elements are automatically removed.
2. **Mutable**: Sets are mutable, meaning you can add or remove elements from a set after it has been created.
3. **Unique Elements**: Sets contain only unique elements. If you try to add a duplicate element to a set, it will be ignored.
4. **Operations**: Sets support various set operations such as union (**|**), intersection (**&**), difference (**-**), and symmetric difference (**^**). These operations allow you to perform common set operations like combining sets, finding common elements, or finding unique elements.
5. **Membership Testing**: Sets provide efficient membership testing using the **in** operator. You can quickly check whether an element exists in a set.
6. **No Indexing**: Sets do not support indexing or slicing. You cannot access individual elements of a set by index.

### Dictionaries:

1. **Key-Value Pairs**: Dictionaries are collections of key-value pairs, where each key is associated with a value. Keys are unique within a dictionary, and each key can only map to one value.
2. **Unordered**: Like sets, dictionaries are unordered collections. This means that the order of key-value pairs in a dictionary is not guaranteed.
3. **Mutable**: Dictionaries are mutable, meaning you can add, remove, or modify key-value pairs after the dictionary has been created.
4. **Keys**: Keys in a dictionary must be hashable, meaning they must be immutable and have a stable hash value. Common examples of hashable types include strings, integers, and tuples (if they only contain hashable elements).
5. **Values**: Values in a dictionary can be of any data type, including lists, tuples, sets, other dictionaries, or even custom objects.
6. **Operations**: Dictionaries support various operations for accessing, adding, modifying, and deleting key-value pairs. These operations allow you to manipulate the contents of a dictionary efficiently.
7. **Iteration**: Dictionaries support iteration over keys, values, or key-value pairs using methods like **keys()**, **values()**, and **items()**.
8. **Membership Testing**: Dictionaries support membership testing for keys using the **in** operator. You can quickly check whether a key exists in a dictionary.

43. How do you perform operations like insertion and deletion in lists and dictionaries?

Solution- LIST

To insert an element at a specific position in a list, you can use the **insert()** method.

To add an element to the end of a list, you can use the **append()** method

To remove an element from a list by value, you can use the **remove()** method

DICTIONARY

**Insertion**:

* To add a key-value pair to a dictionary, simply assign a value to a new key

To remove a key-value pair from a dictionary, you can use the **del** statement or the **pop()** method.

44. Discuss the differences between mutable and immutable data types.

### Solution- Mutable Data Types:

1. **Definition**: Mutable data types are data types whose contents can be modified after creation. This means that you can change, add, or remove elements or properties of a mutable object without creating a new object.
2. **Examples**: Examples of mutable data types in Python include lists, dictionaries, and sets. Objects of these types can be modified after creation.
3. **Modification**: Since mutable objects can be modified, any changes made to them directly affect the original object. There's no need to create a new object to store the modified data.
4. **Identity**: Mutable objects have a fixed identity (memory address) but a variable value. The identity of the object remains the same throughout its lifetime, even if its value changes.

### Immutable Data Types:

1. **Definition**: Immutable data types are data types whose contents cannot be modified after creation. Once an immutable object is created, its contents remain fixed and cannot be changed.
2. **Examples**: Examples of immutable data types in Python include integers, floats, strings, tuples, and frozensets. Objects of these types cannot be modified after creation.
3. **Modification**: Since immutable objects cannot be modified, any attempt to change their value results in the creation of a new object with the modified value. The original object remains unchanged.
4. **Identity**: Immutable objects have a fixed identity (memory address) and a fixed value. Once an immutable object is created, its value cannot be altered, so its identity remains the same throughout its lifetime.

45. How are if-else statements nested in Python?

Solution- In Python, you can nest if-else statements within each other to create more complex conditional logic. Nesting allows you to check multiple conditions sequentially and execute different blocks of code based on the outcome of these conditions