Python viva

1. **What is Python, and what are its advantages?**

python is a high-level, interpreted programming language known for its simplicity, readability, and versatility. It was created by Guido van Rossum and first released in 1991.

**Advantages of Python:**

1. Readability and Simplicity:
2. Wide Range of Applications:
   * Web Development
   * Data Science and Machine Learning:
   * Automation:
   * Game Development:
3. Large Community and Support
4. Integration Capabilities:
5. Portability
6. Extensive Libraries
7. Rapid Development
8. Open-Source

Disadvantages:

* Slower Execution Speed
* Memory Consumption

**2.What are the basic data types in Python?**

**Python has several basic data types that form the building blocks for data manipulation. These types are used to store various kinds of data, such as numbers, text, or collections of items.**

1. **int (Integer):**
   * **Used to represent whole numbers, either positive or negative.**
   * **Examples: 5, -42, 1000**
2. **float (Floating-point number):**
   * **Represents real numbers with decimal points.**
   * **Examples: 3.14, -0.001, 2.0**
3. **str (String):**
   * **A sequence of characters, used to represent text.**
   * **Strings are enclosed in single quotes ('...') or double quotes ("...").**
   * **Examples: "Hello, World!", 'Python'**
4. **bool (Boolean):**
   * **Represents one of two values: True or False.**
   * **Commonly used for conditional statements or comparisons.**
   * **Examples: True, False**
5. **list:**
   * **A collection of ordered, mutable items (elements), which can be of mixed data types.**
   * **Lists are defined by square brackets [...].**
   * **Examples: [1, 2, 3], ['apple', 10, 3.14]**
6. **tuple:**
   * **Similar to a list, but immutable (cannot be changed after creation).**
   * **Tuples are defined by parentheses (...).**
   * **Examples: (1, 2, 3), ('apple', 10, 3.14)**
7. **set:**
   * **An unordered collection of unique items. Sets do not allow duplicate elements.**
   * **Defined by curly braces {...} or using the set() function.**
   * **Examples: {1, 2, 3}, {'apple', 'banana', 'orange'}**
8. **dict (Dictionary):**
   * **A collection of key-value pairs, where each key is associated with a value.**
   * **Dictionaries are defined by curly braces {key: value, ...}.**
   * **Examples: {'name': 'Alice', 'age': 25}, {1: 'one', 2: 'two'}**

**--------------------------------------------------------------------------------------------------------------------------------**

**3.** **What is the difference between a list and a tuple in Python?**

| **Feature** | **List** | **Tuple** |
| --- | --- | --- |
| **Mutability** | **Mutable (can change)** | **Immutable (cannot change)** |
| **Syntax** | **[]** | **()** |
| **Performance** | **Slower** | **Faster** |
| **Use Case** | **Dynamic data (can change)** | **Static data (fixed, constant)** |
| **Methods** | **Many methods (e.g., append)** | **Fewer methods (e.g., count)** |
| **Memory** | **Uses more memory** | **More memory-efficient** |
| **Hashable** | **No** | **Yes (if elements are hashable)** |

**4.** **How do you declare a variable in Python?**

**In Python, declaring a variable is simple and does not require specifying its type. You just assign a value to a variable name using the assignment operator (=).**

**variable\_name = value**

**variable\_name: The name you want to give to the variable.**

**value: The data you want to store in the variable.**

**5.** **What is the difference between Python 2 and Python 3?**

**Python 2 stores strings by ASCII; Python 3 uses Unicode. Python 2 has a more complex syntax than Python 3.**

**Many Python 2 libraries aren't forward compatible; many libraries exclusively use Python 3.**

**Python discontinued Python 2 support in January 2020; Python 3 remains the most popular choice.**

**6.** **What is a dictionary in Python, and how is it different from a list?**

**Dictionary:**

**Collections of multiple values stored In a single variable through key value pairs is called dictionary.**

**It is ordered, mutable, and does not allow duplicate values.**

**7.How do you handle exceptions in Python?**

**In Python, exceptions are handled using the try-except block. This allows you to catch and handle errors gracefully, preventing your program from crashing when an exception occurs.**

**try:**

**# Code that may raise an exception**

**except ExceptionType:**

**# Code to handle the exception**

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

**except block: Defines how to handle the exception if it occurs.**

**Optionally, you can use else, finally, and catch multiple exceptions**

**8.What is the difference between a function and a method in Python?**

**In Python, the main difference between a function and a method is that a function is independent and can be called from anywhere, while a method is associated with a class or object and must be called on that object or within that class:**

**9. What is a lambda function in Python?**

**A lambda function in Python is a small, anonymous function defined using the lambda keyword. It can have any number of arguments but only one expression. Lambda functions are often used for short, simple operations and are typically used in situations where a function is needed for a short period.**

**# A lambda function to add two numbers**

**add = lambda x, y: x + y**

**# Using the lambda function**

**print(add(3, 4)) # Output: 7**

**10. How do you read and write files in Python?**

**In Python, you can read from and write to files using the built-in open() function.**

**# Open a file in read/write mode ('r+')**

**with open('filename.txt', 'r+') as file:**

**content = file.read()**

**file.write("\nNew content added.")**

**16. What is a loop in Python, and what is its purpose?**

* **Loop: A control structure that repeatedly executes a block of code based on a condition or over a sequence of items.**
* **Purpose: To efficiently perform repetitive tasks, iterate over data, and minimize manual code writing.**

**Types of loops in Python:**

1. **for loop: Used for iterating over a sequence (e.g., list, range, dictionary).**
2. **while loop: Continues executing as long as a specific condition is met.**

**17. What are the types of loops in Python?  
The types of loops in Python are:**

1. **for loop:**
   * **Iterates over a sequence (e.g., list, tuple, string, or range).**
   * **Executes code for each element in the sequence.**
2. **while loop:**
   * **Repeats execution as long as a specific condition is true.**
   * **Continues until the condition becomes false.**

**Key controls:**

* **break: Terminates the loop.**
* **continue: Skips the current iteration and moves to the next.**

**18. What is the difference between a for loop and a while loop in Python?**

1. **Iteration Mechanism:**
   * **for loop: Iterates over a sequence (e.g., list, range, string) or any iterable object. The number of iterations is known in advance.**
   * **while loop: Repeats as long as a specified condition is true. The number of iterations is not necessarily known upfront and depends on the condition.**
2. **Use Case:**
   * **for loop: Best used when iterating through elements in a collection.**
   * **while loop: Ideal when the loop should run until a condition is met.**
3. **Structure:**
   * **for loop: Automatically increments through the sequence.**
   * **while loop: Requires manual update of the condition to avoid infinite loops.**

**19. How do you iterate over a list in Python?**

**Using a for loop:**

**Using for loop with range() and len():**

**Using a while loop:**

**Using list comprehension:**

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

**[print(item) for item in my\_list]**

**20. How do you use the range function in Python?**

**The range() function in Python generates a sequence of numbers, typically used for looping a specific number of times. Here's how to use it:**

**Syntax:**

**python**

**Copy code**

**range(start, stop, step)**

* **start (optional): The starting value (inclusive). Default is 0.**
* **stop (required): The ending value (exclusive).**
* **step (optional): The difference between each number. Default is 1.**

**21. What is the purpose of the break statement in Python?**

**The break statement in Python is used to immediately exit a loop, regardless of the loop's condition. This can be helpful when you want to stop the loop once a certain condition is met, rather than waiting for the loop to finish all iterations.**

* **Stop the loop when a certain condition is met, preventing unnecessary iterations.**
* for i in range(10):
* if i == 5:
* break # Exit the loop when i equals 5
* print(i)
* **Output**:
* 0
* 1
* 2
* 3
* 4
* Here, the loop stops when i becomes 5, skipping the rest of the iterations.

22. What is the purpose of the continue statement in Python?

The **continue statement** in Python is used to **skip the current iteration** of a loop and move to the next iteration, without terminating the loop entirely.

**Purpose:**

* **Skip certain iterations** based on a condition, while allowing the loop to continue with the next iteration.

**Example:**

for i in range(5):

if i == 3:

continue # Skip the iteration when i equals 3

print(i)

**Output**:

Copy code

0

1

2

4

Here, the iteration where i == 3 is skipped, and the loop continues with the next iteration.

**23. How do you iterate over a dictionary in Python?**

**Iterating over keys:**

**Iterating over values:**

**Iterating over key-value pairs:**

**Iterating over keys using .keys():**

**24. How do you use the enumerate function in Python?**

**The enumerate() function in Python adds a counter to an iterable (like a list or tuple) and returns it as an enumerate object. This is useful for obtaining both the index and the value during iteration.**

**Syntax:**

**python**

**Copy code**

**enumerate(iterable, start=0)**

* **iterable: The collection you want to enumerate (e.g., list, tuple).**
* **start (optional): The starting index (default is 0).**

**Example:**

**python**

**Copy code**

**my\_list = ['apple', 'banana', 'cherry']**

**for index, value in enumerate(my\_list):**

**print(index, value)**

**Output:**

**Copy code**

**0 apple**

**1 banana**

**2 cherry**

**Using a Custom Start Index:**

**python**

**Copy code**

**my\_list = ['apple', 'banana', 'cherry']**

**for index, value in enumerate(my\_list, start=1): # Starting index at 1**

**print(index, value)**

**Output:**

**Copy code**

**1 apple**

**2 banana**

**3 cherry**

**Using enumerate() allows you to easily access both the index and the value of items in an iterable.**

**25. How do you use nested loops in Python?**

**Nested loops in Python involve placing one loop inside another loop. This allows you to iterate over multiple dimensions or collections.**

**for i in range(3): # Outer loop**

**for j in range(2): # Inner loop**

**print(f"i: {i}, j: {j}")**

**Output:**

**i: 0, j: 0**

**i: 0, j: 1**

**i: 1, j: 0**

**i: 1, j: 1**

**i: 2, j: 0**

**i: 2, j: 1**

**26. How do you use the zip() function in Python?**

**The zip() function combines multiple iterables (like lists or tuples) into a single iterable of tuples. Each tuple contains elements from the corresponding position of each iterable.**

**list1 = [1, 2, 3]**

**list2 = ['a', 'b', 'c']**

**zipped = zip(list1, list2)**

**print(list(zipped)) # Convert to list for display**

**Output:**

**[(1, 'a'), (2, 'b'), (3, 'c')]**

**27. Using the sorted() Function in Python**

**The sorted() function in Python returns a new list containing all elements from an iterable (like a list, tuple, or string) in sorted order. It does not modify the original iterable but returns a new, sorted version of it.**

**sorted(iterable, key=None, reverse=False)**

**numbers = [4, 2, 7, 1]**

**sorted\_numbers = sorted(numbers, reverse=True)**

**print(sorted\_numbers) # Output: [7, 4, 2, 1]**

**28. Using the reversed() Function in Python**

**The reversed() function returns an iterator that accesses elements of an iterable (like a list, tuple, or string) in reverse order. Unlike sorted(), reversed() does not produce a sorted order; it simply reverses the current order of the elements.**

**numbers = [1, 2, 3, 4]**

**reversed\_numbers = list(reversed(numbers))**

**print(reversed\_numbers) # Output: [4, 3, 2, 1]**

**29. What is the purpose of the else statement in a loop in Python?**

**The else statement in a loop in Python executes a block of code after the loop finishes normally (without hitting a break).**

**Purpose: To indicate that the loop has completed without interruption**

**for num in range(3):**

**if num == 5:**

**break**

**else:**

**print("5 not found.") # Executes if the loop completes normally**

**What is a decorators?**

**In Python, a decorator is a special type of function that allows you to modify or extend the behavior of another function or method without changing its code.**

**Decorators are defined using the @decorator\_name syntax, which is placed above the function you want to decorate. When you apply a decorator to a function, you're essentially "wrapping" that function in another function, which can add extra functionality.**

**1. map() Function**

* **Purpose: map() applies a given function to each item of an iterable (like a list) and returns a new iterable (a map object) with the results.**
* **Use Case: Transform or modify each element in the original iterable, often to create a new collection with altered or computed values.**
* **Output: Returns a map object containing the results of applying the function to each item.**

**2. filter() Function**

* **Purpose: filter() applies a given function to each item in an iterable and returns only those items for which the function returns True.**
* **Use Case: Filter out items based on a condition, keeping only elements that meet specific criteria.**
* **Output: Returns a filter object containing only the items that satisfy the condition.**

An **abstract class** in Python is a class that cannot be instantiated on its own and is intended to serve as a blueprint for other classes. It typically contains one or more **abstract methods**—methods that are declared in the abstract class but lack implementation. Subclasses of an abstract class are expected to implement these abstract methods.

* In Python, abstract classes are created using the abc (Abstract Base Class) module. The purpose of an abstract class is to provide a common interface and structure for its subclasses, enforcing that specific methods are implemented in subclasses.
* **1. sorted() Function**
* The sorted() function returns a new sorted list from the elements of any iterable. It does not modify the original iterable but instead returns a new sorted list.

**The reversed() function returns an iterator that accesses the given sequence in reverse order. It does not create a new reversed list but rather an iterator that generates elements in reverse.**

**Class Method**

* **Definition: A class method is defined using the @classmethod decorator.**
* **Parameter: Takes cls as the first parameter, which represents the class itself, not an instance.**
* **Access: Can access and modify class-level attributes but cannot directly access or modify instance-specific data.**
* **Use Case: Useful for factory methods (methods that return instances of the class) or when you need to operate on or modify class-level data across all instances.**

**Static Method**

* **Definition: A static method is defined using the @staticmethod decorator.**
* **Parameter: Does not take self or cls as a first parameter; it is just a regular function within a class.**
* **Access: Cannot modify or access the class or instance attributes. It acts like a standalone function that belongs to the class's namespace.**
* **Use Case: Useful for utility functions that perform a task related to the class but do not require access to class or instance attributes.**