### **Session 1,2 and 3:**

### **1. Installing Python**

* **Python Installation**:
  + To begin using Python, you first need to install it on your system. Python is available for all major operating systems (Windows, macOS, and Linux).
  + **Windows**: Download the Python installer from [Python's official website](https://www.python.org/downloads/). After downloading, run the installer, and make sure to check the box that says "Add Python to PATH" before clicking Install Now.
  + **macOS/Linux**: Python usually comes pre-installed on macOS and most Linux distributions. You can check if Python is installed by typing python3 --version in the terminal. If not installed, use a package manager like brew for macOS or apt for Ubuntu to install it.
* **Setting up a Python environment**:
  + After installation, you can start using Python by opening the command line (Command Prompt, Terminal, or shell) and typing python (or python3 on macOS/Linux).
  + For development, you can use a Python virtual environment (using venv) to create isolated spaces for your Python projects.

### **2. Introduction to Python**

* **What is Python?**: Python is a high-level, interpreted programming language, designed to be easy to read and write. Its syntax is clean, and it encourages developers to write code that is simple and readable.
* **Why Python?**:
  + **Simple Syntax**: Python’s syntax is clear, intuitive, and easy to learn for beginners.
  + **Versatility**: Python is used in many domains such as web development (using frameworks like Django and Flask), data science (with libraries like Pandas, Numpy, and Matplotlib), machine learning (via TensorFlow, Keras, Scikit-learn), and more.
  + **Large Community and Libraries**: Python has a massive collection of libraries that simplify tasks like data analysis, web scraping, machine learning, etc.
* **Python Interpreter**: Python code is executed by an interpreter. This means you don't need to compile your Python code before running it, which makes it a highly dynamic language.
* **Basic Python Features**:
  + Interpreted language
  + Object-oriented
  + High-level language
  + Dynamically typed
  + Extensible with C/C++ and other languages

### **3. Introduction to Different Python IDEs**

An **Integrated Development Environment (IDE)** is a software application that provides tools to make the programming process easier, such as writing, debugging, and executing code. Here are some popular Python IDEs:

* **PyCharm**: A powerful and widely used Python IDE, developed by JetBrains. It offers features like code completion, error highlighting, debugging, and version control integration.
* **Visual Studio Code (VS Code)**: A lightweight, open-source editor with excellent Python support through extensions. It is customizable and integrates with a lot of version control tools and debuggers.
* **Jupyter Notebooks**: Often used for data analysis and machine learning. Jupyter allows you to write and execute Python code in a notebook format, which is especially useful for exploratory data analysis.
* **Spyder**: A scientific IDE designed for data analysis, commonly used in data science and scientific computing.
* **IDLE**: Python's built-in IDE, simple and easy to use for beginners, but not as feature-rich as the others.

### **4. Basic Syntax**

Python has a simple and clean syntax, which is why it is often recommended for beginners. Here are some essential points:

* **Comments**:
  + Comments are lines of code that the Python interpreter ignores. They help explain the code. You can write comments in Python using # for single-line comments and triple quotes (''' or """) for multi-line comments.
* **Indentation**:
  + Python uses indentation to define the blocks of code (instead of curly braces {}). It is essential to maintain proper indentation; otherwise, the code will result in an error.
* **Statements and Expressions**:
  + **Statement**: A line of code that performs an action (e.g., assignments, function calls).
  + **Expression**: A combination of variables, operators, and values that Python interprets and evaluates to produce a result.

**Example**:  
python  
CopyEdit  
# This is a comment

x = 5 # Assignment statement

print(x) # Function call statement

### **5. Data Types, Variables, Operators, Input/Output**

* **Data Types**:
  + Python has various built-in data types, including:
    - **Numeric Types**: int (integer), float (floating-point number), complex (complex numbers).
    - **Sequence Types**: list, tuple, range.
    - **Text Type**: str (string).
    - **Mapping Type**: dict (dictionary).
    - **Set Types**: set, frozenset.
    - **Boolean Type**: bool (True/False).
    - **None Type**: None (used to represent null or no value).
* **Variables**:

Variables in Python are used to store data values. A variable is created by assigning a value to a name, like:  
python  
CopyEdit  
x = 10 # Integer variable

name = "John" # String variable

* **Operators**:
  + **Arithmetic Operators**: +, -, \*, /, // (floor division), % (modulus), \*\* (exponentiation).
  + **Comparison Operators**: ==, !=, >, <, >=, <=.
  + **Logical Operators**: and, or, not.
  + **Assignment Operators**: =, +=, -=, \*=, /=.
  + **Bitwise Operators**: &, |, ^, <<, >>, ~.
* **Input/Output**:

**Input**: Use the input() function to get input from the user. It returns a string by default, which can be converted to other types using functions like int(), float(), etc.  
python  
CopyEdit  
name = input("Enter your name: ")

age = int(input("Enter your age: ")) # Convert input to an integer

**Output**: Use print() to display output to the console.  
python  
CopyEdit  
print("Hello, " + name + "! You are " + str(age) + " years old.") # Concatenation

### **6. Declaring Variables, Data Types in Programs**

* **Declaring Variables**:
  + In Python, you don't need to explicitly declare a variable before assigning it a value. Variables are created when you assign a value to them.
  + The syntax for declaring a variable is simple: variable\_name = value

Example:  
python  
CopyEdit  
x = 10 # Integer variable

name = "Alice" # String variable

is\_active = True # Boolean variable

* **Data Types**:
  + Python has several built-in data types:
    - **Integers (int)**: Whole numbers, e.g., x = 5
    - **Floating-point numbers (float)**: Numbers with decimals, e.g., y = 3.14
    - **Strings (str)**: Sequence of characters, e.g., name = "John"
    - **Booleans (bool)**: Logical values True or False, e.g., is\_valid = True
    - **Lists (list)**: Ordered collection of items, e.g., fruits = ["apple", "banana", "cherry"]
    - **Tuples (tuple)**: Immutable ordered collection, e.g., point = (1, 2)
    - **Dictionaries (dict)**: Key-value pairs, e.g., person = {"name": "John", "age": 25}
    - **Sets (set)**: Unordered collection of unique items, e.g., unique\_numbers = {1, 2, 3}
* **Type Checking**:

You can check the type of a variable using the type() function:  
python  
CopyEdit  
print(type(x)) # Output: <class 'int'>

### **7. Your First Python Program**

* The "Hello, World!" program is often the first program written in any language. It demonstrates how to output a simple message to the screen.

**Example**:

python

CopyEdit

print("Hello, World!") # Output: Hello, World!

* In this example:
  + print() is a built-in function in Python that outputs text or variables to the console.
  + The text "Hello, World!" is passed as an argument to the print() function, which is displayed on the screen.

### **8. Flow of Control (Modules, Branching)**

* **Modules**:
  + A module is a file containing Python code (functions, variables, classes, etc.) that can be imported and used in other programs. Python has built-in modules, and you can create your own.

To use a module, you use the import statement:  
python  
CopyEdit  
import math

print(math.sqrt(16)) # Output: 4.0

* + Here, we import the math module and use its sqrt() function to find the square root of 16.
* **Branching (Control Flow)**:
  + Control flow refers to the order in which individual statements, instructions, or function calls are executed or evaluated.
  + Conditional branching is done using if, elif, and else statements, which allow you to execute certain blocks of code based on conditions.

### **9. If, If-else, Nested If-else**

* **If Statement**:
  + The if statement is used to execute a block of code if a specified condition is true.

Example:  
python  
CopyEdit  
age = 18

if age >= 18:

print("You are an adult.") # This will execute because age is 18

* **If-else Statement**:
  + The else block is executed if the condition in the if statement is false.

Example:  
python  
CopyEdit  
age = 16

if age >= 18:

print("You are an adult.")

else:

print("You are a minor.") # This will execute because age is 16

* **Nested If-else**:
  + You can use if-else inside another if-else block. This is called a nested if-else statement.

Example:  
python  
CopyEdit  
age = 20

if age >= 18:

if age >= 21:

print("You are eligible to drink alcohol.")

else:

print("You are an adult but not eligible to drink alcohol.") # This will execute

else:

print("You are a minor.")

### **10. Looping, For, While**

* **Loops** are used to execute a block of code repeatedly based on a condition. Python provides two primary types of loops: for and while.
* **For Loop**:
  + The for loop is used to iterate over a sequence (like a list, tuple, string, etc.).

Example:  
python  
CopyEdit  
fruits = ["apple", "banana", "cherry"]

for fruit in fruits:

print(fruit) # Prints each fruit in the list

* + The loop iterates through the fruits list and prints each element.
* **While Loop**:
  + The while loop runs as long as the specified condition is True. If the condition becomes False, the loop stops.

Example:  
python  
CopyEdit  
count = 0

while count < 5:

print(count) # Prints 0 to 4

count += 1 # Increases count by 1 after each iteration

* + In this example, the loop continues until count is no longer less than 5.
* **Break and Continue**:

**Break**: Stops the loop immediately and exits.  
python  
CopyEdit  
for i in range(10):

if i == 5:

break # Exits the loop when i equals 5

print(i)

**Continue**: Skips the current iteration and moves to the next iteration.  
python  
CopyEdit  
for i in range(10):

if i == 5:

continue # Skips the current iteration when i equals 5

print(i)

### **11. Nested Loops**

* **Definition**: A nested loop refers to having one loop inside another loop. The inner loop will run completely for every single iteration of the outer loop. It is commonly used when working with multi-dimensional data structures like lists of lists (2D arrays).

**Syntax**: You can nest for loops or while loops inside each other. Here's how it works:  
python  
CopyEdit  
for i in range(3): # Outer loop

for j in range(2): # Inner loop

print(f"i = {i}, j = {j}")

**Output**:  
css  
CopyEdit  
i = 0, j = 0

i = 0, j = 1

i = 1, j = 0

i = 1, j = 1

i = 2, j = 0

i = 2, j = 1

* **How Nested Loops Work**:
  + The outer loop runs first and for each iteration, the inner loop runs completely.
  + Example: If the outer loop runs 3 times, and the inner loop runs 2 times for each outer loop iteration, the inner loop will run a total of 6 times (3 \* 2).
* **Use Case**: Nested loops are useful when dealing with matrix-like data, such as a grid of rows and columns. For example, when performing operations on 2D arrays or grids.

### **12. Control Structure**

* **Definition**: A control structure is a block of programming that decides the flow of control in a program. Python has several control structures, which can be broadly categorized into:
  + **Conditional Statements (Branching)**:
    - These include if, elif, else to make decisions.

Example:  
python  
CopyEdit  
age = 20

if age >= 18:

print("Adult")

else:

print("Minor")

* + **Loops (Repetition)**:
    - These allow you to repeat a block of code multiple times (e.g., for and while loops).
  + **Function Calls**:
    - Functions allow you to group related statements, and call them as needed to execute blocks of code in specific parts of your program.
  + **Exception Handling**:
    - Python provides mechanisms to handle exceptions using try, except, finally. This allows programs to handle unexpected errors gracefully without crashing.

Example:  
python  
CopyEdit  
try:

x = 10 / 0

except ZeroDivisionError:

print("Cannot divide by zero!")

* **Types of Control Structures**:
  + **Sequential**: The default behavior in Python where statements execute one after another.
  + **Selection**: Based on conditions using if, elif, and else statements.
  + **Repetition**: Repeating code using for and while loops.
  + **Exception Handling**: Using try, except, and finally to handle errors.
* **Control Flow**: The control flow determines the order in which the statements in the program are executed. Based on conditions (using if statements), the flow can either proceed sequentially, skip certain parts of code, or loop through parts of the code.

### **13. Uses of Break & Continue**

* **Break**:
  + The break statement is used to exit a loop prematurely, regardless of whether the loop condition is still True.
  + It can be used in both for and while loops to terminate the loop when a specific condition is met.

Example:  
python  
CopyEdit  
for i in range(10):

if i == 5:

break # Exit the loop when i is 5

print(i)

**Output**:  
CopyEdit  
0

1

2

3

4

* + In this example, the loop will stop when i reaches 5, even though the loop was supposed to run until 9.
* **Continue**:
  + The continue statement is used to skip the current iteration of a loop and move on to the next iteration. It only skips the current iteration and does not terminate the loop.

Example:  
python  
CopyEdit  
for i in range(5):

if i == 2:

continue # Skip the iteration when i is 2

print(i)

**Output**:  
CopyEdit  
0

1

3

4

* + In this case, when i is 2, the continue statement is executed, and the print(i) statement is skipped for that iteration. The loop continues with the next value of i.
* **Common Use Cases**:
  + **Break**: When you need to exit a loop early once a certain condition is met. For example, when searching for an item in a list and stopping once it’s found.
  + **Continue**: When you want to skip certain iterations in a loop. For example, when skipping even numbers in a loop over a range.

### **Easy Questions:**

1. What is the main purpose of Python?
   * A) To build websites
   * B) To solve mathematical equations
   * C) To automate tasks and process data
   * D) To create mobile apps  
     **Answer**: C) To automate tasks and process data
2. How do you install Python on Windows?
   * A) Download the .exe installer from the official Python website and run it
   * B) Use the terminal to install it via a package manager
   * C) Use the pip command to install it
   * D) It comes pre-installed with Windows  
     **Answer**: A) Download the .exe installer from the official Python website and run it

What will the following Python code output?  
python  
CopyEdit  
print("Hello, World!")

* + A) Hello World
  + B) hello world
  + C) Hello, World!
  + D) Hello World!  
    **Answer**: C) Hello, World!

1. What is the purpose of the print() function in Python?
   * A) To display text on the console
   * B) To take user input
   * C) To declare a variable
   * D) To return a value  
     **Answer**: A) To display text on the console

What will the following Python code output?  
python  
CopyEdit  
x = 5

print(x)

* + A) 5
  + B) x
  + C) 0
  + D) Error  
    **Answer**: A) 5

1. What is the correct way to declare a string variable in Python?
   * A) x = 'Hello'
   * B) x = Hello
   * C) string = "Hello"
   * D) Both A and C  
     **Answer**: D) Both A and C
2. Which of the following is a valid data type in Python?
   * A) Integer
   * B) Float
   * C) String
   * D) All of the above  
     **Answer**: D) All of the above

What will the following code output?  
python  
CopyEdit  
x = 10

y = 5

print(x + y)

* + A) 15
  + B) 5
  + C) 105
  + D) Error  
    **Answer**: A) 15

What will the following code output?  
python  
CopyEdit  
name = "Alice"

print("Hello, " + name)

* + A) Hello, Alice
  + B) Hello Alice
  + C) Hello, Alice!
  + D) Error  
    **Answer**: A) Hello, Alice

What is the output of the following code?  
python  
CopyEdit  
age = int(input("Enter your age: "))

print("You are", age, "years old")

* + A) The program will crash
  + B) The program will ask for input and then print the age
  + C) The program will always print "You are years old"
  + D) None of the above  
    **Answer**: B) The program will ask for input and then print the age

### **Medium Questions:**

What is the result of the following comparison?  
python  
CopyEdit  
x = 10

y = 5

print(x > y)

* + A) True
  + B) False
  + C) Error
  + D) 10  
    **Answer**: A) True

What is the output of the following code?  
python  
CopyEdit  
x = 10

y = 20

if x > y:

print("x is greater")

else:

print("y is greater")

* + A) x is greater
  + B) y is greater
  + C) Error
  + D) No output  
    **Answer**: B) y is greater

1. Which of the following statements will print the numbers 1 through 5?
   * A) for i in range(5): print(i)
   * B) for i in range(1, 5): print(i)
   * C) for i in range(1, 6): print(i)
   * D) for i in range(6): print(i)  
     **Answer**: C) for i in range(1, 6): print(i)

What is the output of the following code?  
python  
CopyEdit  
x = 5

if x > 3:

print("Yes")

else:

print("No")

* + A) Yes
  + B) No
  + C) Error
  + D) 3  
    **Answer**: A) Yes

1. What does the else keyword do in a Python if-else statement?
   * A) It runs a block of code if the condition is false
   * B) It runs a block of code only if the condition is true
   * C) It allows you to exit the function
   * D) None of the above  
     **Answer**: A) It runs a block of code if the condition is false

What is the result of the following code?  
python  
CopyEdit  
x = 5

while x > 0:

print(x)

x -= 1

* + A) 5, 4, 3, 2, 1
  + B) 1, 2, 3, 4, 5
  + C) Error
  + D) Infinite loop  
    **Answer**: A) 5, 4, 3, 2, 1

1. Which of the following is correct syntax for a for loop in Python?
   * A) for x in range(5):
   * B) for x = 1 to 5:
   * C) for x in 1, 5:
   * D) for 1 to 5 in x:  
     **Answer**: A) for x in range(5):

What is the output of the following code?  
python  
CopyEdit  
x = [1, 2, 3]

y = [4, 5, 6]

for i in zip(x, y):

print(i)

* + A) (1, 4), (2, 5), (3, 6)
  + B) (1, 2), (3, 4), (5, 6)
  + C) 1, 2, 3, 4, 5, 6
  + D) Error  
    **Answer**: A) (1, 4), (2, 5), (3, 6)

1. What does the continue statement do in a loop?
   * A) It skips the current iteration and moves to the next iteration
   * B) It breaks out of the loop
   * C) It stops the program
   * D) None of the above  
     **Answer**: A) It skips the current iteration and moves to the next iteration
2. What does the break statement do in a loop?
   * A) It skips the current iteration
   * B) It stops the loop entirely
   * C) It moves to the next iteration
   * D) It continues to the next block of code  
     **Answer**: B) It stops the loop entirely

### **Hard Questions:**

What is the output of the following code?  
python  
CopyEdit  
fruits = ["apple", "banana", "cherry"]

for fruit in fruits:

if fruit == "banana":

break

print(fruit)

* + A) apple, banana
  + B) apple
  + C) apple, banana, cherry
  + D) Error  
    **Answer**: B) apple

1. How do you handle exceptions in Python?
   * A) Using try and except
   * B) Using if and else
   * C) Using try only
   * D) Python does not support exception handling  
     **Answer**: A) Using try and except
2. What is the difference between a **list** and a **tuple** in Python?
   * A) A tuple is mutable, while a list is immutable
   * B) A list is mutable, while a tuple is immutable
   * C) Both are mutable
   * D) Both are immutable  
     **Answer**: B) A list is mutable, while a tuple is immutable
3. Which data type does Python use to store key-value pairs?
   * A) List
   * B) Dictionary
   * C) Set
   * D) Tuple  
     **Answer**: B) Dictionary

What will the following code output?  
python  
CopyEdit  
x = "Python"

y = "Programming"

print(x + y)

* + A) PythonProgramming
  + B) Python Programming
  + C) Python
  + D) Error  
    **Answer**: A) PythonProgramming

What is the output of this Python program?  
python  
CopyEdit  
n = int(input("Enter a number: "))

factorial = 1

for i in range(1, n+1):

factorial \*= i

print(factorial)

* + A) Factorial of the number
  + B) Number
  + C) Error
  + D) None  
    **Answer**: A) Factorial of the number

1. What is the use of the zip() function in Python?
   * A) It merges two lists
   * B) It splits a string into individual characters
   * C) It combines multiple lists element-wise into tuples
   * D) It joins two strings  
     **Answer**: C) It combines multiple lists element-wise into tuples
2. How can you find the largest number in a list using Python?
   * A) Using the max() function
   * B) Using the sorted() function
   * C) Using a loop
   * D) All of the above  
     **Answer**: D) All of the above

What will be the output of this code?  
python  
CopyEdit  
n = 5

if n > 0:

print("Positive")

else:

print("Negative")

* + A) Positive
  + B) Negative
  + C) Error
  + D) None  
    **Answer**: A) Positive

1. Which of the following is a correct way to declare a dictionary in Python?
   * A) d = {1: "one", 2: "two"}
   * B) d = ["one", "two"]
   * C) d = (1, "one", 2, "two")
   * D) d = {1, 2, 3}  
     **Answer**: A) d = {1: "one", 2: "two"}
2. How do you access the value of a key in a dictionary?
   * A) d[key]
   * B) d.get(key)
   * C) Both A and B
   * D) None of the above  
     **Answer**: C) Both A and B

What will the following code output?  
python  
CopyEdit  
x = "hello"

print(x[1:4])

* + A) hel
  + B) ell
  + C) hello
  + D) Error  
    **Answer**: B) ell

1. How do you handle division by zero error in Python?
   * A) Use try and except
   * B) Use if statements
   * C) Python automatically handles it
   * D) Both A and B  
     **Answer**: D) Both A and B
2. Which statement would you use to terminate a loop early in Python?
   * A) break
   * B) continue
   * C) exit()
   * D) stop()  
     **Answer**: A) break

What does this code print?  
python  
CopyEdit  
for i in range(3):

print(i)

* + A) 1, 2, 3
  + B) 0, 1, 2
  + C) 1, 2, 3, 4
  + D) 0, 1, 2, 3  
    **Answer**: B) 0, 1, 2

1. How would you reverse a list in Python?
   * A) list.reverse()
   * B) reversed(list)
   * C) list[::-1]
   * D) All of the above  
     **Answer**: D) All of the above

What will this code output?  
python  
CopyEdit  
n = 8

while n > 0:

print(n)

n -= 2

* + A) 8, 6, 4, 2
  + B) 8, 6, 4
  + C) 2, 4, 6, 8
  + D) Error  
    **Answer**: A) 8, 6, 4, 2

1. Which of these is an immutable data type in Python?
   * A) List
   * B) Dictionary
   * C) Tuple
   * D) Set  
     **Answer**: C) Tuple
2. What is a palindrome?
   * A) A word that is the same backward as forward
   * B) A number divisible by 2
   * C) A string with even length
   * D) A function in Python  
     **Answer**: A) A word that is the same backward as forward

What is the output of the following code?  
python  
CopyEdit  
s = "hello"

print(s[::-1])

* + A) olleh
  + B) hello
  + C) Error
  + D) None  
    **Answer**: A) olleh

### **Session 4: Strings**

### **1. Accessing Strings**

In Python, strings are sequences of characters, and each character in the string can be accessed using an index. String indexing starts at 0 for the first character.

**Positive Indexing**: Starts from 0 for the first character.  
python  
CopyEdit  
string = "Hello"

print(string[0]) # Output: H

**Negative Indexing**: Starts from -1 for the last character.  
python  
CopyEdit  
string = "Hello"

print(string[-1]) # Output: o

**Range of Indices**: You can access a range of characters using slicing.  
python  
CopyEdit  
string = "Hello"

print(string[1:4]) # Output: ell

### **2. Basic Operations on Strings**

Python allows a variety of basic operations on strings:

**Concatenation**: You can concatenate strings using the + operator.  
python  
CopyEdit  
string1 = "Hello"

string2 = "World"

result = string1 + " " + string2

print(result) # Output: Hello World

**Repetition**: You can repeat a string using the \* operator.  
python  
CopyEdit  
string = "Hello"

result = string \* 3

print(result) # Output: HelloHelloHello

**Length**: Use len() to get the length of a string.  
python  
CopyEdit  
string = "Hello"

print(len(string)) # Output: 5

**Membership**: You can check if a substring exists in a string using in.  
python  
CopyEdit  
string = "Hello"

print("e" in string) # Output: True

### **3. Assigning Multiple Values at Once**

In Python, you can assign multiple variables at once in a single statement. This is a simple way to unpack values into variables.

**Example of Assigning Multiple Values**:  
python  
CopyEdit  
a, b, c = 1, 2, 3

print(a, b, c) # Output: 1 2 3

**Swapping Values**: You can also swap values of two variables.  
python  
CopyEdit  
a, b = 5, 10

a, b = b, a

print(a, b) # Output: 10 5

**Unpacking Strings**: You can unpack a string into individual characters.  
python  
CopyEdit  
a, b, c = "ABC"

print(a, b, c) # Output: A B C

### **4. Formatting Strings**

String formatting allows you to create strings with dynamic content. There are different ways to format strings in Python:

**f-strings** (Python 3.6+): The most modern and readable way.  
python  
CopyEdit  
name = "Alice"

age = 25

print(f"My name is {name} and I am {age} years old.")

# Output: My name is Alice and I am 25 years old.

**str.format()** method: Older but still widely used.  
python  
CopyEdit  
name = "Alice"

age = 25

print("My name is {} and I am {} years old.".format(name, age))

# Output: My name is Alice and I am 25 years old.

**Percent (%) formatting**: An older way to format strings (still supported).  
python  
CopyEdit  
name = "Alice"

age = 25

print("My name is %s and I am %d years old." % (name, age))

# Output: My name is Alice and I am 25 years old.

### **5. String Slicing**

String slicing is a way to access parts of a string using a range of indices. The syntax is string[start:end:step]:

* **start**: The index where the slice starts (inclusive).
* **end**: The index where the slice ends (exclusive).
* **step**: The interval at which characters are selected.

Examples:

python

CopyEdit

string = "Hello, World!"

print(string[0:5]) # Output: Hello

print(string[7:]) # Output: World!

print(string[:5]) # Output: Hello

print(string[::2]) # Output: Hoo ol!

### **6. Historical Note on String Methods**

In earlier versions of Python, string methods were not as advanced as they are today. Over time, many helpful methods have been added to Python's string class (str), making string manipulation more efficient and intuitive. Python's string methods include:

**lower()**: Converts a string to lowercase.  
python  
CopyEdit  
string = "HELLO"

print(string.lower()) # Output: hello

**upper()**: Converts a string to uppercase.  
python  
CopyEdit  
string = "hello"

print(string.upper()) # Output: HELLO

**strip()**: Removes leading and trailing spaces.  
python  
CopyEdit  
string = " Hello "

print(string.strip()) # Output: Hello

**replace(old, new)**: Replaces occurrences of a substring with another.  
python  
CopyEdit  
string = "Hello, World!"

print(string.replace("World", "Python")) # Output: Hello, Python!

**split(delimiter)**: Splits a string into a list based on a delimiter.  
python  
CopyEdit  
string = "Hello, World!"

print(string.split(", ")) # Output: ['Hello', 'World!']

**join(iterable)**: Joins a list of strings into a single string.  
python  
CopyEdit  
words = ["Hello", "World"]

print(" ".join(words)) # Output: Hello World

These methods provide flexibility in manipulating and processing strings effectively in Python.

**How do you access the first character of a string in Python?**

* A) string[1]
* B) string[0]
* C) string[-1]
* D) string[-2]
* **Answer**: B) string[0]

**What is the output of the following code?**python  
CopyEdit  
string = "Python"

print(string[1:4])

* A) Pyt
* B) yth
* C) y
* D) P
* **Answer**: B) yth

**What does the + operator do when applied to strings?**

* A) Concatenates two strings
* B) Repeats the string
* C) Finds the length of the string
* D) Joins two lists
* **Answer**: A) Concatenates two strings

**What is the correct way to multiply a string by 3 in Python?**

* A) string \* 3
* B) string + 3
* C) string \* (3)
* D) Both A and C
* **Answer**: D) Both A and C

**Which method would you use to convert a string to uppercase?**

* A) string.upper()
* B) string.capitalize()
* C) string.lower()
* D) string.title()
* **Answer**: A) string.upper()

**How do you check if a substring exists in a string?**

* A) substring in string
* B) string.contains(substring)
* C) string.find(substring)
* D) substring.search(string)
* **Answer**: A) substring in string

**Which of the following is a valid string slice in Python?**

* A) string[1:3:2]
* B) string[::1]
* C) string[:4]
* D) All of the above
* **Answer**: D) All of the above

**What is the output of the following code?**python  
CopyEdit  
s = "Hello"

print(s[-1])

* A) o
* B) H
* C) Error
* D) None
* **Answer**: A) o

**Which method would you use to remove leading and trailing whitespaces from a string?**

* A) strip()
* B) trim()
* C) clean()
* D) remove()
* **Answer**: A) strip()

**What will the following code output?**python  
CopyEdit  
s = "Python Programming"

print(s[7:])

* A) Python
* B) Programming
* C) Pro
* D) Error
* **Answer**: B) Programming

**How do you format a string in Python using str.format()?**

* A) "Hello, {}".format(name)
* B) "Hello, {0}".format(name)
* C) "Hello, {name}".format(name=name)
* D) All of the above
* **Answer**: D) All of the above

**What will the following code output?**python  
CopyEdit  
word = "Apple"

print(f"The word is {word}")

* A) The word is word
* B) The word is Apple
* C) The word is "Apple"
* D) The word is {word}
* **Answer**: B) The word is Apple

**Which method would you use to replace a substring in a string with another substring?**

* A) replace()
* B) replacing()
* C) sub()
* D) change()
* **Answer**: A) replace()

**What is the result of the following operation?**python  
CopyEdit  
"Hello" \* 2

* A) HelloHello
* B) Hello 2
* C) Error
* D) None
* **Answer**: A) HelloHello

**Which of the following string methods returns a list of substrings?**

* A) split()
* B) slice()
* C) join()
* D) partition()
* **Answer**: A) split()

**What is the output of this code?**python  
CopyEdit  
s = "Python"

print(s[::-1])

* A) Python
* B) nohtyP
* C) Error
* D) None
* **Answer**: B) nohtyP

**What is the output of the following code?**python  
CopyEdit  
string = "Python"

print(string[2:5])

* A) Pyt
* B) thon
* C) tho
* D) tho
* **Answer**: C) tho

**Which of these is an immutable data type in Python?**

* A) List
* B) Dictionary
* C) Tuple
* D) Set
* **Answer**: C) Tuple

**What is the output of the following code?**python  
CopyEdit  
name = "Alice"

print(name + " and Bob")

* A) Alice and Bob
* B) Alice and Bob.
* C) Alice + and Bob
* D) None
* **Answer**: A) Alice and Bob

**How would you reverse a string in Python?**

* A) string.reverse()
* B) reversed(string)
* C) string[::-1]
* D) string.reverse()
* **Answer**: C) string[::-1]

**What is the output of the following code?**python  
CopyEdit  
string = "Python"

print(string[:2])

* A) Py
* B) th
* C) P
* D) Error
* **Answer**: A) Py

**What method is used to convert a string to lowercase?**

* A) lower()
* B) lowercase()
* C) toLower()
* D) down()
* **Answer**: A) lower()

**Which of the following is used to format a string in Python 3.6+?**

* A) % operator
* B) str.format()
* C) f-strings
* D) None of the above
* **Answer**: C) f-strings

**Which of the following will result in an error?**

* A) "Hello".replace("e", "a")
* B) "Hello".split("e")
* C) "Hello".join("e")
* D) "Hello".strip()
* **Answer**: C) "Hello".join("e")

**What is the output of the following code?**python  
CopyEdit  
s = "Welcome to Python!"

print(s.split())

* A) ['Welcome', 'to', 'Python!']
* B) ['Welcome to Python!']
* C) Error
* D) None
* **Answer**: A) ['Welcome', 'to', 'Python!']

**Which of the following will give the same result as "Hello"[::-1]?**

* A) "Hello".reverse()
* B) "Hello".reversed()
* C) reversed("Hello")
* D) "Hello"[::-1]
* **Answer**: D) "Hello"[::-1]

**Which method is used to find the first occurrence of a substring in a string?**

* A) index()
* B) find()
* C) search()
* D) locate()
* **Answer**: B) find()

**Which of these methods is used to remove a specific substring from a string?**

* A) remove()
* B) replace()
* C) delete()
* D) clear()
* **Answer**: B) replace()

**What will the following code output?**python  
CopyEdit  
s = "Hello World"

print(s[::3])

* A) HoWrd
* B) Hel
* C) HloWr
* D) Error
* **Answer**: C) HloWr

**Which statement is correct regarding string slicing?**

* A) String slicing cannot be used with negative indices
* B) String slicing allows negative indices
* C) String slicing requires both start and end parameters
* D) String slicing cannot be done without the step parameter
* **Answer**: B) String slicing allows negative indices

### **Session 5: Dictionary 1. Introducing Dictionaries**

Dictionaries are one of Python's built-in data types used to store key-value pairs. Each key in a dictionary is unique, and the value can be of any data type. Dictionaries are unordered collections, meaning the items are not stored in a specific order. They are mutable, which means you can change them after they are created.

**Example**:  
python  
CopyEdit  
my\_dict = {"name": "John", "age": 25, "city": "New York"}

print(my\_dict) # Output: {'name': 'John', 'age': 25, 'city': 'New York'}

* **Key-Value Structure**: A dictionary is a collection of pairs where each key is associated with a value, written as key: value.

### **2. Defining Dictionaries**

You can define a dictionary in Python using curly braces {} and separating key-value pairs with a colon (:). Multiple key-value pairs are separated by commas.

**Basic Syntax**:  
python  
CopyEdit  
my\_dict = {"key1": "value1", "key2": "value2", "key3": "value3"}

**Empty Dictionary**: To create an empty dictionary:  
python  
CopyEdit  
empty\_dict = {}

**Dictionary with Mixed Data Types**: Dictionaries can hold different types of data, including numbers, lists, and even other dictionaries.  
python  
CopyEdit  
person = {

"name": "Alice",

"age": 30,

"children": ["Bob", "Charlie"],

"address": {"city": "Paris", "postal\_code": 75000}

}

print(person)

# Output: {'name': 'Alice', 'age': 30, 'children': ['Bob', 'Charlie'], 'address': {'city': 'Paris', 'postal\_code': 75000}}

### **3. Modifying Dictionaries**

Dictionaries in Python are mutable, so you can modify them by adding, updating, or removing key-value pairs.

* **Adding or Updating Items**: You can add a new key-value pair to a dictionary or update the value of an existing key by assigning a value to a key.

Adding a new key-value pair:  
python  
CopyEdit  
person = {"name": "Alice", "age": 30}

person["city"] = "Paris" # Adding a new key-value pair

print(person) # Output: {'name': 'Alice', 'age': 30, 'city': 'Paris'}

Updating an existing key:  
python  
CopyEdit  
person["age"] = 31 # Updating the value of an existing key

print(person) # Output: {'name': 'Alice', 'age': 31, 'city': 'Paris'}

**Accessing Values**: You can access the values in a dictionary using their keys:  
python  
CopyEdit  
print(person["name"]) # Output: Alice

**Using get() Method**: The get() method is used to access a value by its key. It is safer because it doesn't raise an error if the key doesn't exist (returns None by default).  
python  
CopyEdit  
print(person.get("city")) # Output: Paris

print(person.get("gender")) # Output: None (since "gender" is not in the dictionary)

### **4. Deleting Items from Dictionaries**

You can delete key-value pairs from a dictionary using different methods.

**Using del Statement**: The del statement removes the key-value pair from the dictionary.  
python  
CopyEdit  
del person["age"] # Removes the key "age" and its associated value

print(person) # Output: {'name': 'Alice', 'city': 'Paris'}

**Using pop() Method**: The pop() method removes a key-value pair and returns the value associated with the key.  
python  
CopyEdit  
removed\_value = person.pop("city") # Removes the "city" key and returns its value

print(removed\_value) # Output: Paris

print(person) # Output: {'name': 'Alice'}

**Using popitem() Method**: The popitem() method removes and returns the last inserted key-value pair (in versions of Python before 3.7, it removed an arbitrary pair).  
python  
CopyEdit  
last\_item = person.popitem() # Removes and returns the last inserted key-value pair

print(last\_item) # Output: ('name', 'Alice')

print(person) # Output: {}

**Using clear() Method**: The clear() method removes all key-value pairs from the dictionary, leaving it empty.  
python  
CopyEdit  
person.clear() # Clears the dictionary

print(person) # Output: {}

**What is the syntax for creating an empty dictionary in Python?**

* A) {}
* B) []
* C) ()
* D) None
* **Answer**: A) {}

**Which of the following is the correct way to access the value of a key in a dictionary?**

* A) dict.key
* B) dict["key"]
* C) dict.key()
* D) dict(key)
* **Answer**: B) dict["key"]

**How do you add a new key-value pair to a dictionary?**

* A) dict.add(key, value)
* B) dict[key] = value
* C) dict.insert(key, value)
* D) dict.append(key, value)
* **Answer**: B) dict[key] = value

**What will the following code output?**python  
CopyEdit  
my\_dict = {"name": "Alice", "age": 30}

print(my\_dict["age"])

* A) Alice
* B) 30
* C) Error
* D) None
* **Answer**: B) 30

**How can you remove a key-value pair from a dictionary in Python?**

* A) remove(key)
* B) pop(key)
* C) delete(key)
* D) clear()
* **Answer**: B) pop(key)

**Which of the following methods removes all items from a dictionary?**

* A) remove()
* B) clear()
* C) pop()
* D) delete()
* **Answer**: B) clear()

**Which method is used to access the value for a given key in a dictionary safely (without causing an error if the key doesn't exist)?**

* A) get()
* B) find()
* C) search()
* D) lookup()
* **Answer**: A) get()

**What is the output of the following code?**python  
CopyEdit  
my\_dict = {"name": "Alice", "age": 30}

print(my\_dict.get("name"))

* A) Alice
* B) 30
* C) Error
* D) None
* **Answer**: A) Alice

**What is the output of the following code?**python  
CopyEdit  
my\_dict = {"name": "Alice", "age": 30}

my\_dict["city"] = "New York"

print(my\_dict)

* A) {"name": "Alice", "age": 30, "city": "New York"}
* B) {"name": "Alice", "age": "New York"}
* C) {"name": "Alice", "city": "New York"}
* D) {"age": 30, "city": "New York"}
* **Answer**: A) {"name": "Alice", "age": 30, "city": "New York"}

**How do you check if a key exists in a dictionary?**

* A) key in dict
* B) dict.has\_key(key)
* C) dict.contains(key)
* D) key.exists(dict)
* **Answer**: A) key in dict

**What will the following code output?**python  
CopyEdit  
my\_dict = {"name": "Alice", "age": 30}

print(my\_dict.pop("age"))

* A) Alice
* B) 30
* C) Error
* D) None
* **Answer**: B) 30

**What does the popitem() method do?**

* A) Removes a key-value pair and returns it as a tuple
* B) Removes a key and returns its value
* C) Removes a key and returns a tuple of keys
* D) Removes all items in a dictionary
* **Answer**: A) Removes a key-value pair and returns it as a tuple

**Which method is used to update the value associated with a given key in a dictionary?**

* A) update()
* B) change()
* C) modify()
* D) set()
* **Answer**: A) update()

**Which of the following will remove the key "age" and its associated value from the dictionary?**

* A) del dict["age"]
* B) dict.delete("age")
* C) dict.remove("age")
* D) dict.pop("age")
* **Answer**: A) del dict["age"]

**What is the output of the following code?**python  
CopyEdit  
my\_dict = {"name": "Alice", "age": 30}

del my\_dict["age"]

print(my\_dict)

* A) {"name": "Alice", "age": 30}
* B) {"name": "Alice"}
* C) {"age": 30}
* D) Error
* **Answer**: B) {"name": "Alice"}

**What will the following code output?**python  
CopyEdit  
my\_dict = {"name": "Alice", "age": 30}

print(my\_dict.get("city"))

* A) Error
* B) None
* C) {}
* D) city
* **Answer**: B) None

**Which of the following methods can be used to add multiple key-value pairs at once in a dictionary?**

* A) update()
* B) add()
* C) insert()
* D) append()
* **Answer**: A) update()

**How do you get all the keys from a dictionary in Python?**

* A) dict.keys()
* B) dict.get\_keys()
* C) dict.all\_keys()
* D) dict.fetch\_keys()
* **Answer**: A) dict.keys()

**What is the output of the following code?**python  
CopyEdit  
my\_dict = {"name": "Alice", "age": 30}

print(list(my\_dict.keys()))

* A) ['name', 'age']
* B) ['Alice', 30]
* C) ('name', 'age')
* D) ('Alice', 30)
* **Answer**: A) ['name', 'age']

**Which of the following methods is used to get all the values in a dictionary?**

* A) dict.values()
* B) dict.get\_values()
* C) dict.all\_values()
* D) dict.fetch\_values()
* **Answer**: A) dict.values()

**What will be the output of the following code?**python  
CopyEdit  
my\_dict = {"name": "Alice", "age": 30}

print(my\_dict.popitem())

* A) ('name', 'Alice')
* B) ('age', 30)
* C) None
* D) Error
* **Answer**: B) ('age', 30)

**What is the output of the following code?**python  
CopyEdit  
my\_dict = {"name": "Alice", "age": 30, "city": "New York"}

my\_dict.update({"age": 31, "city": "Los Angeles"})

print(my\_dict)

* A) {"name": "Alice", "age": 31, "city": "Los Angeles"}
* B) {"name": "Alice", "age": 30, "city": "Los Angeles"}
* C) {"name": "Alice", "age": 31, "city": "New York"}
* D) Error
* **Answer**: A) {"name": "Alice", "age": 31, "city": "Los Angeles"}

**Which of the following will result in an error?**

* A) del dict["key"]
* B) dict.pop("key")
* C) dict.get("key")
* D) dict.remove("key")
* **Answer**: D) dict.remove("key")

**What does the keys() method return?**

* A) A list of all values in the dictionary
* B) A list of all keys in the dictionary
* C) A list of key-value pairs
* D) None
* **Answer**: B) A list of all keys in the dictionary

**What will be the output of the following code?**python  
CopyEdit  
my\_dict = {"name": "Alice", "age": 30}

my\_dict["age"] = 31

print(my\_dict)

* A) {"name": "Alice", "age": 31}
* B) {"name": "Alice", "age": 30}
* C) {"age": 31}
* D) Error
* **Answer**: A) {"name": "Alice", "age": 31}

**Which method removes a key-value pair and returns the value associated with the key?**

* A) remove()
* B) pop()
* C) clear()
* D) del()
* **Answer**: B) pop()

**Which of these is the correct way to create a dictionary with multiple keys and values?**

* A) {"name": "Alice", "age": 30}
* B) {name: Alice, age: 30}
* C) {"name", "Alice", "age", 30}
* D) {name="Alice", age=30}
* **Answer**: A) {"name": "Alice", "age": 30}

**What does the pop() method return?**

* A) A list of all values
* B) The removed value
* C) The removed key
* D) None
* **Answer**: B) The removed value

**What is the output of the following code?**python  
CopyEdit  
my\_dict = {"name": "Alice", "age": 30}

del my\_dict["name"]

print(my\_dict)

* A) {"age": 30}
* B) {"name": "Alice", "age": 30}
* C) {}
* D) Error
* **Answer**: A) {"age": 30}

**How do you update a dictionary with another dictionary in Python?**

* A) dict.update()
* B) dict.add()
* C) dict.insert()
* D) dict.append()
* **Answer**: A) dict.update()

### **Session 6 & 7: Workin with List**

### **1. Introducing Lists**

In Python, a **list** is a collection of ordered, mutable, and heterogeneous elements, which can store a sequence of items. Lists are one of the most commonly used data structures because they can store a wide range of data types, including numbers, strings, and other objects. A list is defined by enclosing elements within square brackets [].

**Example**:  
python  
CopyEdit  
my\_list = [1, 2, 3, 4, 5]

print(my\_list) # Output: [1, 2, 3, 4, 5]

* **Key Features of Lists**:
  + **Ordered**: Lists maintain the order of insertion.
  + **Mutable**: You can modify lists after they are created.
  + **Indexed**: Lists allow random access using an index, starting from 0.
  + **Heterogeneous**: Lists can store different data types like integers, strings, or even other lists.

### **2. Defining Lists**

To define a list, you simply use square brackets [] and separate the elements with commas.

**Example**:  
python  
CopyEdit  
fruits = ["apple", "banana", "cherry"]

numbers = [1, 2, 3, 4, 5]

mixed\_list = [1, "apple", 3.14, True]

print(fruits) # Output: ['apple', 'banana', 'cherry']

**Empty List**: To create an empty list, just use empty square brackets:  
python  
CopyEdit  
empty\_list = []

### **3. Declare, Assign, and Retrieve Values from Lists**

* **Declaring a List**: Lists are created by specifying elements inside square brackets.
* **Assigning Values to a List**: Values are assigned to a list at the time of creation or through assignment.
* **Accessing (Retrieving) Values**: You can access individual elements by using their index. Lists are zero-indexed, meaning the first element is at index 0, the second at index 1, and so on.

**Example**:  
python  
CopyEdit  
my\_list = [10, 20, 30, 40, 50]

# Accessing elements

print(my\_list[0]) # Output: 10 (first element)

print(my\_list[2]) # Output: 30 (third element)

**Negative Indexing**: You can also access elements from the end using negative indices.  
python  
CopyEdit  
print(my\_list[-1]) # Output: 50 (last element)

print(my\_list[-2]) # Output: 40 (second-last element)

### **4. Accessing Lists**

There are several ways to access and manipulate elements in a list:

* **Indexing**: Accessing a specific element using an index.
* **Slicing**: Extracting a sublist using a range of indices.

**Example of Slicing**:  
python  
CopyEdit  
my\_list = [10, 20, 30, 40, 50]

print(my\_list[1:4]) # Output: [20, 30, 40]

print(my\_list[:3]) # Output: [10, 20, 30] (first 3 elements)

print(my\_list[2:]) # Output: [30, 40, 50] (from index 2 to end)

**Modifying Lists**: You can change values in a list by directly assigning new values to an index.  
python  
CopyEdit  
my\_list[2] = 35 # Change element at index 2

print(my\_list) # Output: [10, 20, 35, 40, 50]

### **5. Operations in Lists**

Lists in Python come with a variety of built-in methods and operators to perform operations like adding, removing, and searching for elements. Some of the common operations include:

**Concatenation**: Using the + operator to combine two lists.  
python  
CopyEdit  
list1 = [1, 2, 3]

list2 = [4, 5, 6]

list3 = list1 + list2

print(list3) # Output: [1, 2, 3, 4, 5, 6]

**Repetition**: Using the \* operator to repeat the list multiple times.  
python  
CopyEdit  
list4 = [1, 2, 3]

list5 = list4 \* 2

print(list5) # Output: [1, 2, 3, 1, 2, 3]

**Membership**: Using the in keyword to check if an element exists in the list.  
python  
CopyEdit  
fruits = ["apple", "banana", "cherry"]

print("apple" in fruits) # Output: True

print("grape" in fruits) # Output: False

**List Length**: You can get the length of a list using the len() function.  
python  
CopyEdit  
fruits = ["apple", "banana", "cherry"]

print(len(fruits)) # Output: 3

### **6. Adding Elements to Lists**

You can add elements to a list in different ways:

**Using append()**: Adds a single element to the end of the list.  
python  
CopyEdit  
fruits = ["apple", "banana"]

fruits.append("cherry")

print(fruits) # Output: ['apple', 'banana', 'cherry']

**Using insert()**: Adds an element at a specified index.  
python  
CopyEdit  
fruits = ["apple", "banana"]

fruits.insert(1, "cherry") # Insert at index 1

print(fruits) # Output: ['apple', 'cherry', 'banana']

**Using extend()**: Adds multiple elements (another list) to the end of the list.  
python  
CopyEdit  
fruits = ["apple", "banana"]

fruits.extend(["cherry", "date"])

print(fruits) # Output: ['apple', 'banana', 'cherry', 'date']

### **7. Searching Lists**

To search for an element in a list, you can use several methods:

**Using in keyword**: Checks if an item exists in the list.  
python  
CopyEdit  
fruits = ["apple", "banana", "cherry"]

print("banana" in fruits) # Output: True

print("orange" in fruits) # Output: False

**Using index()**: Returns the index of the first occurrence of a specified element.  
python  
CopyEdit  
fruits = ["apple", "banana", "cherry"]

print(fruits.index("banana")) # Output: 1

**Using count()**: Returns the count of occurrences of an element in the list.  
python  
CopyEdit  
fruits = ["apple", "banana", "banana", "cherry"]

print(fruits.count("banana")) # Output: 2

### **8. Deleting List Elements**

Python provides several ways to remove elements from a list. You can delete elements by index, by value, or by using list methods. Here are some common methods:

**Using del**: The del statement can delete an element by its index or the entire list.  
python  
CopyEdit  
my\_list = [10, 20, 30, 40, 50]

# Deleting an element by index

del my\_list[2]

print(my\_list) # Output: [10, 20, 40, 50]

# Deleting the entire list

del my\_list

# print(my\_list) # This will raise a NameError because the list is deleted

**Using remove()**: The remove() method deletes the first occurrence of a specific value from the list.  
python  
CopyEdit  
fruits = ["apple", "banana", "cherry", "banana"]

fruits.remove("banana")

print(fruits) # Output: ['apple', 'cherry', 'banana']

**Using pop()**: The pop() method removes an element from a specific index (default is the last element) and returns it.  
python  
CopyEdit  
my\_list = [10, 20, 30, 40, 50]

# Pop last element

popped\_value = my\_list.pop()

print(popped\_value) # Output: 50

print(my\_list) # Output: [10, 20, 30, 40]

# Pop element by index

popped\_value = my\_list.pop(1)

print(popped\_value) # Output: 20

print(my\_list) # Output: [10, 30, 40]

**Using clear()**: The clear() method removes all elements from the list, making it an empty list.  
python  
CopyEdit  
my\_list = [10, 20, 30]

my\_list.clear()

print(my\_list) # Output: []

### **9. Using List Operators**

Python provides several operators that you can use to work with lists. These operators make it easy to manipulate lists:

**Concatenation (+)**: You can combine two or more lists using the + operator.  
python  
CopyEdit  
list1 = [1, 2, 3]

list2 = [4, 5, 6]

result = list1 + list2

print(result) # Output: [1, 2, 3, 4, 5, 6]

**Repetition (\*)**: You can repeat a list multiple times using the \* operator.  
python  
CopyEdit  
list1 = [1, 2, 3]

result = list1 \* 3

print(result) # Output: [1, 2, 3, 1, 2, 3, 1, 2, 3]

**Membership (in)**: You can check if an element exists in a list using the in keyword.  
python  
CopyEdit  
fruits = ["apple", "banana", "cherry"]

print("banana" in fruits) # Output: True

print("orange" in fruits) # Output: False

**Indexing ([])**: You can use the square brackets [] to access specific elements or slices of the list.  
python  
CopyEdit  
my\_list = [10, 20, 30, 40]

print(my\_list[1]) # Output: 20

print(my\_list[1:3]) # Output: [20, 30]

**Length (len())**: You can use the len() function to get the number of elements in a list.  
python  
CopyEdit  
my\_list = [10, 20, 30, 40]

print(len(my\_list)) # Output: 4

### **10. Mapping Lists**

Mapping allows you to apply a function to each element in a list. Python provides the map() function for this purpose. The map() function applies a given function to each item of an iterable (like a list) and returns a map object (which can be converted into a list).

**Syntax**:  
python  
CopyEdit  
map(function, iterable)

**Example**:  
python  
CopyEdit  
# Function to square a number

def square(num):

return num \* num

numbers = [1, 2, 3, 4, 5]

squared\_numbers = map(square, numbers)

print(list(squared\_numbers)) # Output: [1, 4, 9, 16, 25]

**Using Lambda with map()**: Instead of defining a function separately, you can use a lambda function directly inside the map() function.  
python  
CopyEdit  
numbers = [1, 2, 3, 4, 5]

squared\_numbers = map(lambda x: x \* x, numbers)

print(list(squared\_numbers)) # Output: [1, 4, 9, 16, 25]

**Example with Strings**:  
python  
CopyEdit  
fruits = ["apple", "banana", "cherry"]

upper\_fruits = map(str.upper, fruits)

print(list(upper\_fruits)) # Output: ['APPLE', 'BANANA', 'CHERRY']

### **11. Joining Lists and Splitting Strings**

In Python, you can join a list of strings into a single string using the join() method. Similarly, you can split a string into a list using the split() method.

* **Joining Lists**: The join() method takes an iterable (like a list) and joins its elements into a single string, with a specified separator between elements.

**Syntax**:  
python  
CopyEdit  
separator.join(iterable)

**Example**:  
python  
CopyEdit  
words = ["Hello", "World", "Python"]

result = " ".join(words)

print(result) # Output: "Hello World Python"

You can use any character as a separator (not just a space):  
python  
CopyEdit  
words = ["apple", "banana", "cherry"]

result = "-".join(words)

print(result) # Output: "apple-banana-cherry"

* **Splitting Strings**: The split() method splits a string into a list of words based on a specified delimiter. The default delimiter is any whitespace (space, tab, newline).

**Syntax**:  
python  
CopyEdit  
string.split(separator)

**Example**:  
python  
CopyEdit  
text = "apple,banana,cherry"

fruits = text.split(",")

print(fruits) # Output: ['apple', 'banana', 'cherry']

**Using Default Separator**:  
python  
CopyEdit  
text = "Hello World Python"

words = text.split()

print(words) # Output: ['Hello', 'World', 'Python']

**Limiting Splits**: You can specify a maximum number of splits.  
python  
CopyEdit  
text = "apple,banana,cherry,orange"

result = text.split(",", 2) # Split only into 3 parts

print(result) # Output: ['apple', 'banana', 'cherry,orange']

### 

### **MCQ**

1. What will be the output of the following code?

python

CopyEdit

my\_list = [10, 20, 30, 40]

print(my\_list[1])

A) 10  
B) 20  
C) 30  
D) 40

**Answer: B) 20**

2. What is the result of the following code?

python

CopyEdit

my\_list = [1, 2, 3, 4, 5]

my\_list[2] = 100

print(my\_list)

A) [1, 2, 100, 4, 5]  
B) [100, 2, 3, 4, 5]  
C) [1, 2, 3, 4, 100]  
D) [1, 2, 3, 100, 5]

**Answer: A) [1, 2, 100, 4, 5]**

3. Which method is used to add a new item at the end of the list?

A) append()  
B) extend()  
C) insert()  
D) push()

**Answer: A) append()**

4. What will be the output of the following code?

python

CopyEdit

fruits = ["apple", "banana", "cherry"]

fruits.remove("banana")

print(fruits)

A) ["apple", "banana", "cherry"]  
B) ["apple", "cherry"]  
C) ["banana", "cherry"]  
D) ["apple", "banana"]

**Answer: B) ["apple", "cherry"]**

5. What will the following code output?

python

CopyEdit

my\_list = [10, 20, 30]

print(my\_list \* 2)

A) [10, 20, 30, 10, 20, 30]  
B) [20, 30, 10, 20, 30, 10]  
C) [10, 20, 30]  
D) Error

**Answer: A) [10, 20, 30, 10, 20, 30]**

6. Which of the following will remove the last element from a list and return it?

A) pop()  
B) remove()  
C) del  
D) clear()

**Answer: A) pop()**

7. What does the following code do?

python

CopyEdit

my\_list = [1, 2, 3]

del my\_list[1]

print(my\_list)

A) Deletes the entire list  
B) Removes the second element and returns it  
C) Removes the element at index 1  
D) Deletes all the odd numbers

**Answer: C) Removes the element at index 1**

8. What is the result of the following operation?

python

CopyEdit

list1 = [1, 2]

list2 = [3, 4]

print(list1 + list2)

A) [1, 2, 3, 4]  
B) [1, 2, 1, 2, 3, 4]  
C) [1, 3, 2, 4]  
D) Error

**Answer: A) [1, 2, 3, 4]**

9. What will be the output of this code?

python

CopyEdit

my\_list = ["apple", "banana", "cherry"]

print("banana" in my\_list)

A) True  
B) False  
C) None  
D) Error

**Answer: A) True**

10. How can you get the number of elements in a list?

A) len()  
B) count()  
C) size()  
D) length()

**Answer: A) len()**

11. Which function will convert a map object to a list?

A) list()  
B) map()  
C) set()  
D) convert()

**Answer: A) list()**

12. What will be the output of this code?

python

CopyEdit

fruits = ["apple", "banana", "cherry"]

fruits.append("orange")

print(fruits)

A) ['apple', 'banana', 'cherry', 'orange']  
B) ['apple', 'banana', 'cherry']  
C) ['apple', 'banana', 'cherry', 'apple']  
D) Error

**Answer: A) ['apple', 'banana', 'cherry', 'orange']**

13. What is the default value of the sep parameter in join()?

A) ","  
B) " " (a single space)  
C) "-"  
D) ""

**Answer: B) " " (a single space)**

14. What does the following code do?

python

CopyEdit

text = "apple,banana,cherry"

result = text.split(",")

print(result)

A) Splits the string into a list of individual characters  
B) Splits the string into a list at each comma  
C) Joins the list into a string with commas  
D) Converts the string into a set

**Answer: B) Splits the string into a list at each comma**

15. What will the following code return?

python

CopyEdit

my\_list = [10, 20, 30, 40]

print(my\_list.index(30))

A) 0  
B) 1  
C) 2  
D) 3

**Answer: C) 2**

16. Which of the following methods is used to join elements of a list into a string?

A) split()  
B) append()  
C) join()  
D) map()

**Answer: C) join()**

17. Which method is used to add an element at a specific index in a list?

A) append()  
B) insert()  
C) extend()  
D) push()

**Answer: B) insert()**

18. How can you check if an element is in a list?

A) check()  
B) in  
C) exists()  
D) find()

**Answer: B) in**

19. What is the output of the following code?

python

CopyEdit

my\_list = [1, 2, 3]

print(len(my\_list))

A) 1  
B) 3  
C) 2  
D) 6

**Answer: B) 3**

20. What will the following code do?

python

CopyEdit

my\_list = [1, 2, 3]

my\_list.append([4, 5])

print(my\_list)

A) [1, 2, 3, [4, 5]]  
B) [1, 2, 3, 4, 5]  
C) [1, 2, 3, "4, 5"]  
D) Error

**Answer: A) [1, 2, 3, [4, 5]]**

21. What is the result of the following code?

python

CopyEdit

my\_list = [10, 20, 30]

del my\_list

print(my\_list)

A) [10, 20, 30]  
B) Error, because the list is deleted  
C) []  
D) None

**Answer: B) Error, because the list is deleted**

22. What will be the output of the following code?

python

CopyEdit

fruits = ["apple", "banana", "cherry"]

fruits.insert(1, "grape")

print(fruits)

A) ['apple', 'grape', 'banana', 'cherry']  
B) ['grape', 'apple', 'banana', 'cherry']  
C) ['apple', 'banana', 'grape', 'cherry']  
D) Error

**Answer: A) ['apple', 'grape', 'banana', 'cherry']**

23. What does the pop() method return when used without an index?

A) The first element of the list  
B) The last element of the list  
C) A new list  
D) The entire list

**Answer: B) The last element of the list**

24. Which of the following will remove all items from a list?

A) clear()  
B) del  
C) pop()  
D) remove()

**Answer: A) clear()**

25. What will be the output of the following code?

python

CopyEdit

my\_list = ["apple", "banana", "cherry"]

my\_list[1] = "blueberry"

print(my\_list)

A) ["apple", "blueberry", "cherry"]  
B) ["apple", "banana", "blueberry"]  
C) ["blueberry", "banana", "cherry"]  
D) Error

**Answer: A) ["apple", "blueberry", "cherry"]**

26. What does the count() method do in lists?

A) Counts the total number of elements  
B) Counts occurrences of a specific element  
C) Counts the number of odd elements  
D) Counts the indices in the list

**Answer: B) Counts occurrences of a specific element**

27. What is the result of the following code?

python

CopyEdit

my\_list = [1, 2, 3]

my\_list.append([4, 5])

print(my\_list)

A) [1, 2, 3, [4, 5]]  
B) [1, 2, 3, 4, 5]  
C) [1, 2, 3, "4, 5"]  
D) Error

**Answer: A) [1, 2, 3, [4, 5]]**

28. What will be the output of the following code?

python

CopyEdit

my\_list = [1, 2, 3]

my\_list.pop()

print(my\_list)

A) [1, 2]  
B) [2, 3]  
C) [1, 3]  
D) [1, 2, 3]

**Answer: A) [1, 2]**

29. Which of the following operators can be used to check membership in a list?

A) in  
B) ==  
C) is  
D) and

**Answer: A) in**

30. What is the output of the following code?

python

CopyEdit

fruits = ["apple", "banana", "cherry"]

print(fruits[1:3])

A) ['apple', 'banana']  
B) ['banana', 'cherry']  
C) ['apple', 'cherry']  
D) ['banana']

**Answer: B) ['banana', 'cherry']**

### **Session 8 & 9: Working with Functions**

### **1. Defining a Function**

A **function** is a block of reusable code that performs a specific task. Functions help break down a complex problem into smaller, more manageable tasks. In Python, functions are defined using the def keyword.

#### **Syntax:**

python

CopyEdit

def function\_name(parameters):

# function body

# statements to be executed

return value # Optional

* **def**: Keyword to define a function.
* **function\_name**: The name of the function (should follow Python's naming rules).
* **parameters**: Optional; values that are passed into the function.
* **return**: Optionally returns a value from the function.

#### **Example:**

python

CopyEdit

def greet(name):

return f"Hello, {name}!"

### **2. Calling a Function**

To execute the function, you **call** it by using its name followed by parentheses, passing any required arguments inside the parentheses.

#### **Syntax:**

python

CopyEdit

function\_name(arguments)

#### **Example:**

python

CopyEdit

greet("Alice") # Output: Hello, Alice!

In the example above, "Alice" is passed as the argument to the greet() function.

### **3. Types of Functions**

There are mainly two types of functions in Python:

#### **1. Built-in Functions:**

These are functions that Python provides by default. For example, print(), len(), type(), etc.

* **print()**: Displays output on the screen.
* **len()**: Returns the length of an object.

#### **2. User-Defined Functions:**

These are the functions that you define yourself to perform specific tasks (like the greet() function shown above).

### **4. Function Arguments**

Function arguments allow passing data into functions. There are several types of arguments that can be passed:

#### **1. Positional Arguments:**

These arguments are passed to the function in the order in which they are defined.

#### **Example:**

python

CopyEdit

def add(a, b):

return a + b

print(add(5, 3)) # Output: 8

#### **2. Keyword Arguments:**

These arguments are passed by explicitly specifying the parameter name.

#### **Example:**

python

CopyEdit

def greet(name, age):

return f"Hello {name}, you are {age} years old."

print(greet(name="Alice", age=25)) # Output: Hello Alice, you are 25 years old.

#### **3. Default Arguments:**

You can assign default values to parameters in the function definition. These default values are used if no argument is provided for them.

#### **Example:**

python

CopyEdit

def greet(name="Stranger"):

return f"Hello, {name}!"

print(greet()) # Output: Hello, Stranger!

print(greet("Alice")) # Output: Hello, Alice!

#### **4. Variable-Length Arguments:**

Sometimes, you don’t know how many arguments you need to pass to the function. You can use \*args (for non-keyword arguments) and \*\*kwargs (for keyword arguments) to handle this.

* \*args is used for **non-keyword variable-length arguments** (i.e., passing a list of arguments).
* \*\*kwargs is used for **keyword variable-length arguments** (i.e., passing a dictionary of arguments).

#### **Example:**

python

CopyEdit

# Using \*args

def print\_names(\*args):

for name in args:

print(name)

print\_names("Alice", "Bob", "Charlie")

# Using \*\*kwargs

def print\_info(\*\*kwargs):

for key, value in kwargs.items():

print(f"{key}: {value}")

print\_info(name="Alice", age=25, city="New York")

### **5. Anonymous Functions (Lambda Functions)**

An **anonymous function** is a function that is defined without a name. In Python, you can create anonymous functions using the lambda keyword.

Lambda functions are typically used for short, simple functions that are passed as arguments to higher-order functions like map(), filter(), and reduce().

#### **Syntax:**

python

CopyEdit

lambda arguments: expression

#### **Example:**

python

CopyEdit

# Normal function

def add(x, y):

return x + y

# Anonymous (lambda) function

add\_lambda = lambda x, y: x + y

print(add\_lambda(2, 3)) # Output: 5

Lambda functions are commonly used in places where a simple function is needed temporarily, and defining a full function would be unnecessarily verbose.

#### **Example with map():**

python

CopyEdit

numbers = [1, 2, 3, 4, 5]

squared\_numbers = list(map(lambda x: x\*\*2, numbers))

print(squared\_numbers) # Output: [1, 4, 9, 16, 25]

### **Key Points Recap:**

* Functions are defined using def and called by their name followed by parentheses.
* Functions can take positional, keyword, and default arguments.
* Python allows variable-length arguments with \*args and \*\*kwargs.
* Anonymous functions are defined using the lambda keyword, and they are used for simple operations.

### **1. Global and Local Variables**

In Python, variables can be categorized into two types based on their scope:

#### **Local Variables**

* A **local variable** is a variable defined within a function or block, which can only be used inside that function or block.
* Local variables are created when the function is called and destroyed when the function terminates.

#### **Example:**

python

CopyEdit

def my\_function():

local\_var = 10 # Local variable

print(local\_var)

my\_function()

# print(local\_var) # This would result in an error because local\_var is not accessible outside the function.

#### **Global Variables**

* A **global variable** is a variable defined outside of any function, making it accessible throughout the entire program.
* Global variables can be accessed from any function within the program. However, to modify a global variable inside a function, the global keyword must be used.

#### **Example:**

python

CopyEdit

global\_var = 20 # Global variable

def my\_function():

print(global\_var) # Accessing the global variable

my\_function() # Output: 20

#### **Modifying Global Variable Inside a Function:**

python

CopyEdit

global\_var = 20

def modify\_global():

global global\_var # Declare the variable as global

global\_var = 30

modify\_global()

print(global\_var) # Output: 30

### **2. Using Optional and Named Arguments**

In Python, functions can have **optional arguments** (those that are not required) and **named arguments** (where you can specify arguments by name when calling the function).

#### **Optional Arguments:**

You can assign default values to arguments in the function definition. If no value is provided when calling the function, the default value will be used.

#### **Example:**

python

CopyEdit

def greet(name="Stranger", age=18):

return f"Hello, {name}. You are {age} years old."

print(greet()) # Output: Hello, Stranger. You are 18 years old.

print(greet("Alice", 25)) # Output: Hello, Alice. You are 25 years old.

#### **Named Arguments:**

You can also pass arguments using their names, which allows you to pass them in any order.

#### **Example:**

python

CopyEdit

def greet(name, age):

return f"Hello, {name}. You are {age} years old."

print(greet(age=25, name="Alice")) # Output: Hello, Alice. You are 25 years old.

### **3. Using type(), str(), dir(), and Other Built-In Functions**

Python provides several built-in functions that are useful for inspecting and manipulating objects. Some of the commonly used functions are:

#### **type():**

* The type() function returns the type of an object. It can be useful for checking the type of a variable or value.

#### **Example:**

python

CopyEdit

x = 10

print(type(x)) # Output: <class 'int'>

y = "Hello"

print(type(y)) # Output: <class 'str'>

#### **str():**

* The str() function converts an object into a string.

#### **Example:**

python

CopyEdit

x = 10

y = str(x)

print(y) # Output: '10'

print(type(y)) # Output: <class 'str'>

#### **dir():**

* The dir() function returns a list of the attributes and methods available for an object.

#### **Example:**

python

CopyEdit

x = [1, 2, 3]

print(dir(x)) # Output: ['\_\_add\_\_', '\_\_class\_\_', '\_\_contains\_\_', ..., 'sort']

#### **Other Built-In Functions:**

* **len()**: Returns the length of an object (e.g., list, string).
* **max()**: Returns the largest item in an iterable.
* **min()**: Returns the smallest item in an iterable.
* **sum()**: Returns the sum of items in an iterable.
* **abs()**: Returns the absolute value of a number.

#### **Example:**

python

CopyEdit

numbers = [1, 2, 3, 4]

print(len(numbers)) # Output: 4

print(max(numbers)) # Output: 4

print(min(numbers)) # Output: 1

print(sum(numbers)) # Output: 10

### **4. Regular Expressions Using Python**

A **regular expression (regex)** is a sequence of characters that defines a search pattern. It is often used for pattern matching within strings, such as validating input or extracting specific parts of text. Python provides the re module to work with regular expressions.

#### **Using the re Module**

The re module in Python allows you to perform various operations like searching, matching, and replacing strings using regular expressions.

#### **Commonly Used Functions in the re Module:**

* **re.match()**: Checks for a match only at the beginning of the string.
* **re.search()**: Scans the string for a match and returns the first match found.
* **re.findall()**: Returns all non-overlapping matches in a list.
* **re.sub()**: Replaces the matched substring with a given string.

#### **Example - Using re.match():**

python

CopyEdit

import re

pattern = "^a"

text = "apple"

if re.match(pattern, text):

print("Match found!")

else:

print("Match not found!")

Output: Match found! (because the string starts with 'a')

#### **Example - Using re.search():**

python

CopyEdit

import re

pattern = "a"

text = "apple"

if re.search(pattern, text):

print("Match found!")

else:

print("Match not found!")

Output: Match found! (because 'a' is present in the string)

#### **Example - Using re.findall():**

python

CopyEdit

import re

pattern = "\d+" # Matches one or more digits

text = "There are 12 apples and 15 oranges"

matches = re.findall(pattern, text)

print(matches) # Output: ['12', '15']

#### **Example - Using re.sub():**

python

CopyEdit

import re

pattern = "apple"

replacement = "orange"

text = "I like apple pie."

new\_text = re.sub(pattern, replacement, text)

print(new\_text) # Output: I like orange pie.

### **Key Points Recap:**

* **Global and Local Variables**: Local variables are inside functions, and global variables are outside. You can modify a global variable using the global keyword.
* **Optional and Named Arguments**: Functions can have default values for parameters, and you can pass arguments using their names.
* **Built-In Functions**: Functions like type(), str(), dir(), etc., are built into Python for type checking, string conversion, and inspecting objects.
* **Regular Expressions**: The re module allows pattern matching and manipulation of strings using regular expressions, with functions like match(), search(), findall(), and sub().

### **MCQs**

1. What is the output of the following code?

python

CopyEdit

def greet(name):

return "Hello, " + name

print(greet("Alice"))

A) Hello, Alice  
B) Alice  
C) "Hello, Alice"  
D) Error

**Answer: A) Hello, Alice**

2. What is the result of calling type() on the following code?

python

CopyEdit

x = 10

print(type(x))

A) <class 'str'>  
B) <class 'int'>  
C) <class 'float'>  
D) <class 'list'>

**Answer: B) <class 'int'>**

3. Which of the following is the correct syntax for defining a function in Python?

A) function my\_func():  
B) def my\_func():  
C) create function my\_func:  
D) func my\_func():

**Answer: B) def my\_func():**

4. What will be the output of the following code?

python

CopyEdit

def add(x, y=5):

return x + y

print(add(3))

A) 8  
B) 3  
C) 5  
D) Error

**Answer: A) 8**

5. Which keyword is used to define an anonymous function in Python?

A) def  
B) lambda  
C) function  
D) func

**Answer: B) lambda**

6. What will be the output of this code?

python

CopyEdit

def multiply(x, y):

return x \* y

print(multiply(2, 3))

A) 6  
B) 23  
C) 5  
D) Error

**Answer: A) 6**

7. What does the str() function do in Python?

A) Converts an integer to a string  
B) Converts a string to an integer  
C) Converts an integer to a float  
D) Checks if a string is numeric

**Answer: A) Converts an integer to a string**

8. How do you call a function with keyword arguments?

A) my\_function("Alice", 25)  
B) my\_function(name="Alice", age=25)  
C) my\_function(Alice, 25)  
D) my\_function(name="Alice")

**Answer: B) my\_function(name="Alice", age=25)**

9. What is the result of the following code?

python

CopyEdit

x = "Python"

y = " is fun"

print(x + y)

A) Python is fun  
B) Python + is fun  
C) Python + y  
D) Error

**Answer: A) Python is fun**

10. What will be the output of this code?

python

CopyEdit

def greeting(name, age=30):

print(f"Hello {name}, you are {age} years old.")

greeting("Alice")

A) Hello Alice, you are 30 years old.  
B) Hello Alice  
C) Error  
D) Hello Alice, you are 25 years old.

**Answer: A) Hello Alice, you are 30 years old.**

11. What is the correct way to declare a global variable inside a function?

A) global x  
B) global x = 10  
C) x = global 10  
D) declare global x = 10

**Answer: A) global x**

12. What will be the output of the following code?

python

CopyEdit

def square(x):

return x \*\* 2

print(square(4))

A) 16  
B) 8  
C) 4  
D) Error

**Answer: A) 16**

13. What is the output of the following code?

python

CopyEdit

def multiply(x, y=2):

return x \* y

print(multiply(5))

A) 5  
B) 10  
C) Error  
D) 2

**Answer: B) 10**

14. What does the dir() function return?

A) List of available methods and attributes of an object  
B) The type of an object  
C) A string representation of an object  
D) A memory address of an object

**Answer: A) List of available methods and attributes of an object**

15. How do you define a function that takes a variable number of arguments?

A) def my\_func(\*args):  
B) def my\_func(args):  
C) def my\_func(args\*):  
D) def my\_func(...args):

**Answer: A) def my\_func(\*args):**

16. Which of the following is used to replace parts of a string in Python?

A) str.sub()  
B) re.replace()  
C) re.sub()  
D) str.replace()

**Answer: C) re.sub()**

17. What does re.search() do in Python?

A) Checks if a pattern matches at the start of the string  
B) Returns all matches in a list  
C) Returns the first match found in the string  
D) Returns the last match found

**Answer: C) Returns the first match found in the string**

18. What will be the output of the following code?

python

CopyEdit

def func(x=1, y=2):

return x + y

print(func())

A) 1  
B) 2  
C) 3  
D) Error

**Answer: C) 3**

19. What will the following code return?

python

CopyEdit

import re

pattern = "dog"

text = "The dog barks."

result = re.findall(pattern, text)

print(result)

A) ["dog"]  
B) ["dog", "barks"]  
C) ["dog barks"]  
D) []

**Answer: A) ["dog"]**

20. Which function is used to convert an integer to a string?

A) str()  
B) int()  
C) string()  
D) convert()

**Answer: A) str()**

21. What is the output of the following code?

python

CopyEdit

def greet(name):

return "Hello, " + name

name = "Bob"

print(greet(name))

A) Hello, Bob  
B) Hello name  
C) Error  
D) Bob

**Answer: A) Hello, Bob**

22. What is the output of the following code?

python

CopyEdit

x = 10

def func():

global x

x = 20

func()

print(x)

A) 10  
B) 20  
C) Error  
D) None

**Answer: B) 20**

23. What is the purpose of the re.match() function?

A) Searches for a pattern anywhere in the string  
B) Checks for a match only at the start of the string  
C) Finds all occurrences of a pattern in a string  
D) Replaces a pattern in the string

**Answer: B) Checks for a match only at the start of the string**

24. What will the following code return?

python

CopyEdit

import re

pattern = "cat"

text = "The cat is cute."

result = re.search(pattern, text)

print(result.group())

A) cat  
B) The cat is cute.  
C) None  
D) Error

**Answer: A) cat**

25. What will the following code output?

python

CopyEdit

numbers = [1, 2, 3, 4, 5]

print(sum(numbers))

A) 5  
B) 15  
C) 10  
D) Error

**Answer: B) 15**

26. What will be the output of the following code?

python

CopyEdit

def func(x):

return x + 1

result = func(5)

print(result)

A) 6  
B) 5  
C) Error  
D) None

**Answer: A) 6**

27. What will be the output of the following code?

python

CopyEdit

def multiply(x, y):

return x \* y

print(multiply(5, 3))

A) 15  
B) 35  
C) 5  
D) Error

**Answer: A) 15**

28. What is the correct way to handle optional arguments in a function?

A) def func(name="John"):  
B) def func(name):  
C) def func(optional=True):  
D) def func(name?, age?):

**Answer: A) def func(name="John"):**

29. What will be the result of this code?

python

CopyEdit

def greet(name):

return "Hello " + name

names = ["Alice", "Bob", "Charlie"]

for name in names:

print(greet(name))

A) Hello Alice Hello Bob Hello Charlie  
B) Hello Alice Hello Charlie Hello Bob  
C) Error  
D) None

**Answer: A) Hello Alice Hello Bob Hello Charlie**

30. Which of the following methods will return all the methods and attributes of an object?

A) type()  
B) dir()  
C) str()  
D) len()

**Answer: B) dir()**

### **Session 10:**

### **Working with Tuple & sets**

### **Deep Description of Sets in Python**

#### **What is a Set in Python?**

A **set** is an **unordered, mutable, and unindexed** collection of unique elements in Python. It is defined using **curly braces {}** or the built-in set() constructor. Unlike lists or tuples, sets do not allow duplicate elements and are optimized for membership testing (checking if an element exists in a set).

## **1. Creating a Set**

You can create a set using curly braces {} or the set() function.

### **Example 1: Creating a Set**

python

CopyEdit

# Creating a set with curly braces

fruits = {"apple", "banana", "cherry"}

print(fruits)

# Creating an empty set (must use set(), {} creates an empty dictionary)

empty\_set = set()

print(empty\_set) # Output: set()

⚠ **Note:** You cannot create an empty set using {} because it will be treated as a dictionary.

## **2. Properties of Sets**

### **a) Unordered**

Sets do not maintain the order of elements.

python

CopyEdit

my\_set = {3, 1, 4, 2, 5}

print(my\_set) # Output: {1, 2, 3, 4, 5} (Order may vary)

### **b) No Duplicate Elements**

A set automatically removes duplicate values.

python

CopyEdit

num\_set = {1, 2, 2, 3, 4, 4, 5}

print(num\_set) # Output: {1, 2, 3, 4, 5}

## **3. Adding Elements to a Set**

You can use add() to add a single element and update() to add multiple elements.

### **Example: Adding Elements**

python

CopyEdit

numbers = {1, 2, 3}

numbers.add(4) # Adding a single element

print(numbers) # Output: {1, 2, 3, 4}

numbers.update([5, 6, 7]) # Adding multiple elements

print(numbers) # Output: {1, 2, 3, 4, 5, 6, 7}

## **4. Removing Elements from a Set**

You can remove elements using remove(), discard(), or pop().

### **Example: Removing Elements**

python

CopyEdit

numbers = {1, 2, 3, 4, 5}

numbers.remove(3) # Removes 3 (throws an error if not found)

print(numbers) # Output: {1, 2, 4, 5}

numbers.discard(10) # No error if 10 is not in the set

print(numbers)

removed\_item = numbers.pop() # Removes a random item

print(removed\_item)

print(numbers)

⚠ **Note:** remove() raises an error if the element is not found, while discard() does not.

## **5. Set Operations**

Sets support powerful mathematical operations like union, intersection, and difference.

### **a) Union (| or union())**

Combines two sets (removing duplicates).

python

CopyEdit

A = {1, 2, 3}

B = {3, 4, 5}

print(A | B) # Output: {1, 2, 3, 4, 5}

print(A.union(B)) # Output: {1, 2, 3, 4, 5}

### **b) Intersection (& or intersection())**

Finds common elements.

python

CopyEdit

print(A & B) # Output: {3}

print(A.intersection(B)) # Output: {3}

### **c) Difference (- or difference())**

Elements in A but not in B.

python

CopyEdit

print(A - B) # Output: {1, 2}

print(A.difference(B)) # Output: {1, 2}

### **d) Symmetric Difference (^ or symmetric\_difference())**

Elements that are in A or B but not both.

python

CopyEdit

print(A ^ B) # Output: {1, 2, 4, 5}

print(A.symmetric\_difference(B)) # Output: {1, 2, 4, 5}

## **6. Set Membership**

You can check if an element exists in a set using in.

python

CopyEdit

numbers = {1, 2, 3, 4, 5}

print(3 in numbers) # Output: True

print(10 in numbers) # Output: False

## **7. Iterating Through a Set**

Since sets are iterable, you can loop through them.

python

CopyEdit

for item in {"apple", "banana", "cherry"}:

print(item)

## **8. Frozensets**

A **frozenset** is an immutable version of a set, meaning you cannot modify it after creation.

python

CopyEdit

fs = frozenset([1, 2, 3, 4])

print(fs) # Output: frozenset({1, 2, 3, 4})

# Trying to add an element (will raise an error)

fs.add(5) # AttributeError: 'frozenset' object has no attribute 'add'

## **9. When to Use Sets?**

* Removing duplicate elements from a list
* Checking for membership (faster than lists)
* Performing mathematical set operations

## **10. Example: Removing Duplicates from a List**

python

CopyEdit

names = ["Alice", "Bob", "Alice", "Charlie", "Bob"]

unique\_names = set(names)

print(unique\_names) # Output: {'Charlie', 'Alice', 'Bob'}

### **1. Introducing Tuples**

A **tuple** in Python is an ordered collection of elements, similar to lists. However, unlike lists, tuples are **immutable**, meaning once created, their values cannot be modified. Tuples are useful when you want to store multiple items in a single variable and ensure the data remains unchanged.

#### **Key Features of Tuples:**

* They are defined using parentheses () instead of square brackets [] (which are used for lists).
* Tuples can store any type of data: integers, strings, objects, etc.
* The elements of a tuple are indexed, starting from 0.
* **Tuples are immutable**, which means once defined, their elements cannot be altered (no adding, removing, or changing values).
* You can store a tuple inside another tuple or even within lists or dictionaries.

#### **Example of a Tuple:**

python

CopyEdit

my\_tuple = (1, 2, 3, "hello", 4.5)

print(my\_tuple)

Output:

arduino

CopyEdit

(1, 2, 3, 'hello', 4.5)

### **2. Accessing Tuples**

Accessing elements in a tuple is similar to accessing elements in a list. You can use indexing or slicing to retrieve the elements.

#### **Indexing:**

You can access a tuple element by referring to its index, starting from 0.

python

CopyEdit

my\_tuple = (10, 20, 30, 40, 50)

print(my\_tuple[0]) # Access the first element (10)

Output:

CopyEdit

10

#### **Negative Indexing:**

Negative indexing allows you to access elements from the end of the tuple.

python

CopyEdit

print(my\_tuple[-1]) # Access the last element (50)

Output:

CopyEdit

50

#### **Slicing:**

You can also access a range of elements using slicing.

python

CopyEdit

print(my\_tuple[1:4]) # Get elements from index 1 to 3 (20, 30, 40)

Output:

scss

CopyEdit

(20, 30, 40)

### **3. Operations with Tuples**

Tuples support several operations, although, due to their immutability, they differ from lists in terms of modifications. Here are some common operations:

#### **a. Concatenation:**

You can combine two or more tuples using the + operator.

python

CopyEdit

tuple1 = (1, 2, 3)

tuple2 = (4, 5)

result = tuple1 + tuple2

print(result)

Output:

scss

CopyEdit

(1, 2, 3, 4, 5)

#### **b. Repetition:**

You can repeat the elements of a tuple using the \* operator.

python

CopyEdit

tuple3 = (1, 2)

result = tuple3 \* 3

print(result)

Output:

scss

CopyEdit

(1, 2, 1, 2, 1, 2)

#### **c. Membership Test:**

You can check if an element exists in a tuple using the in keyword.

python

CopyEdit

my\_tuple = (10, 20, 30)

print(20 in my\_tuple) # True

print(40 in my\_tuple) # False

Output:

graphql

CopyEdit

True

False

#### **d. Length of a Tuple:**

You can get the number of elements in a tuple using the len() function.

python

CopyEdit

my\_tuple = (1, 2, 3, 4)

print(len(my\_tuple))

Output:

CopyEdit

4

#### **e. Iteration:**

Tuples can be iterated over using loops.

python

CopyEdit

my\_tuple = (1, 2, 3)

for item in my\_tuple:

print(item)

Output:

CopyEdit

1

2

3

#### **f. Indexing with index():**

You can find the index of an element in a tuple using the index() method.

python

CopyEdit

my\_tuple = (10, 20, 30, 40)

print(my\_tuple.index(30))

Output:

CopyEdit

2

#### **g. Count with count():**

You can count how many times an element appears in a tuple using the count() method.

python

CopyEdit

my\_tuple = (10, 20, 30, 10, 10)

print(my\_tuple.count(10))

Output:

CopyEdit

3

#### **h. Nested Tuples:**

Tuples can also contain other tuples as elements.

python

CopyEdit

nested\_tuple = ((1, 2), (3, 4), (5, 6))

print(nested\_tuple[1])

Output:

scss

CopyEdit

(3, 4)

### **Conclusion:**

* Tuples are immutable, ordered collections, useful when you need a fixed set of values.
* You can access elements using indexing and slicing.
* Operations like concatenation, repetition, membership test, and counting are common with tuples.
* They are efficient for storing read-only data.

### **MCQs**

1. Which of the following is the correct syntax for creating a tuple in Python?

A) my\_tuple = [1, 2, 3]  
B) my\_tuple = (1, 2, 3)  
C) my\_tuple = {1, 2, 3}  
D) my\_tuple = 1, 2, 3

**Answer: B) my\_tuple = (1, 2, 3)**

2. What will be the output of the following code?

python

CopyEdit

my\_tuple = (1, 2, 3, 4)

print(my\_tuple[2])

A) 1  
B) 2  
C) 3  
D) Error

**Answer: C) 3**

3. Which operation can be performed on a tuple in Python?

A) Adding elements to the tuple  
B) Removing elements from the tuple  
C) Concatenating two tuples  
D) Modifying elements of a tuple

**Answer: C) Concatenating two tuples**

4. What is the result of the following code?

python

CopyEdit

my\_tuple = (1, 2, 3)

print(my\_tuple \* 2)

A) (1, 2, 3, 1, 2, 3)  
B) (1, 1, 2, 2, 3, 3)  
C) (1, 2, 3)  
D) Error

**Answer: A) (1, 2, 3, 1, 2, 3)**

5. How can you check if an element exists in a tuple?

A) element in tuple  
B) tuple.contains(element)  
C) tuple.index(element)  
D) tuple.check(element)

**Answer: A) element in tuple**

6. What will be the output of the following code?

python

CopyEdit

my\_tuple = (1, 2, 3, 4)

print(len(my\_tuple))

A) 3  
B) 4  
C) 5  
D) Error

**Answer: B) 4**

7. What is the result of the following code?

python

CopyEdit

my\_tuple = (1, 2, 3, 4, 5)

print(my\_tuple[1:4])

A) (1, 2, 3)  
B) (2, 3, 4)  
C) (3, 4, 5)  
D) (1, 2, 3, 4)

**Answer: B) (2, 3, 4)**

8. Which method is used to count the occurrences of an element in a tuple?

A) tuple.count()  
B) tuple.sum()  
C) tuple.index()  
D) tuple.size()

**Answer: A) tuple.count()**

9. What will be the result of the following code?

python

CopyEdit

my\_tuple = (1, 2, 3, 4)

print(my\_tuple.index(3))

A) 2  
B) 3  
C) 1  
D) Error

**Answer: A) 2**

10. Which of the following is a valid way to create a tuple with one element?

A) my\_tuple = (1)  
B) my\_tuple = (1, )  
C) my\_tuple = [1]  
D) my\_tuple = {1}

**Answer: B) my\_tuple = (1, )**

11. Which of the following operations is not allowed with a tuple?

A) Concatenation  
B) Iteration  
C) Modifying an element  
D) Slicing

**Answer: C) Modifying an element**

12. What will be the output of the following code?

python

CopyEdit

my\_tuple = (10, 20, 30, 40)

print(20 in my\_tuple)

A) True  
B) False  
C) Error  
D) None

**Answer: A) True**

13. What is the output of the following code?

python

CopyEdit

nested\_tuple = ((1, 2), (3, 4))

print(nested\_tuple[0])

A) (1, 2)  
B) (3, 4)  
C) 1  
D) Error

**Answer: A) (1, 2)**

14. What will be the output of the following code?

python

CopyEdit

my\_tuple = (1, 2, 3)

my\_tuple[0] = 10

A) (10, 2, 3)  
B) Error  
C) (1, 2, 3, 10)  
D) 10

**Answer: B) Error**

15. How can you create a tuple without using parentheses?

A) tuple = 1, 2, 3  
B) tuple = [1, 2, 3]  
C) tuple = {1, 2, 3}  
D) tuple = (1, 2, 3,)

**Answer: A) tuple = 1, 2, 3**

16. What is the result of the following code?

python

CopyEdit

my\_tuple = (10, 20, 30)

result = my\_tuple + (40, 50)

print(result)

A) (10, 20, 30, 40, 50)  
B) (10, 20, 30, (40, 50))  
C) (10, 20, 30)  
D) (40, 50)

**Answer: A) (10, 20, 30, 40, 50)**

17. What is the output of the following code?

python

CopyEdit

my\_tuple = (1, 2, 3)

print(my\_tuple[-1])

A) 1  
B) 2  
C) 3  
D) Error

**Answer: C) 3**

18. What is the correct way to access the second element in a tuple my\_tuple = (1, 2, 3)?

A) my\_tuple[2]  
B) my\_tuple[1]  
C) my\_tuple[-1]  
D) my\_tuple[0]

**Answer: B) my\_tuple[1]**

19. Which of the following methods is used to get the number of elements in a tuple?

A) tuple.size()  
B) tuple.count()  
C) len(tuple)  
D) tuple.length()

**Answer: C) len(tuple)**

20. What will the following code output?

python

CopyEdit

my\_tuple = (5, 6, 7)

print(my\_tuple \* 0)

A) (5, 6, 7)  
B) Error  
C) ()  
D) (0)

**Answer: C) ()**

### **Session 11, 12, 13: OOPs**

### **1. Object-Oriented Python**

Object-Oriented Programming (OOP) is a programming paradigm that organizes code into **objects**, which are instances of **classes**. It focuses on the following key concepts:

* **Encapsulation**: Wrapping data (variables) and functions (methods) into a single unit.
* **Abstraction**: Hiding complex implementation details and showing only essential features.
* **Inheritance**: A mechanism to allow new classes to inherit properties and methods from existing classes.
* **Polymorphism**: A concept that allows one method to have different meanings based on the object calling it.

OOP provides better modularity and code reusability.

### **2. OOPs Concepts**

Object-Oriented Programming is based on four primary principles:

#### **a) Encapsulation**

Encapsulation is the bundling of data (attributes) and methods (functions) into a single unit or class. It restricts direct access to some of the object’s components, which can help prevent unintended interference and misuse of data.

Example:

python

CopyEdit

class Person:

def \_\_init\_\_(self, name, age):

self.name = name # public attribute

self.\_\_age = age # private attribute (encapsulation)

def get\_age(self):

return self.\_\_age # getter method to access private attribute

# Creating an object of the class

person = Person("Alice", 30)

print(person.get\_age()) # Accessing private data via getter method

#### **b) Abstraction**

Abstraction hides the complex implementation details and shows only the necessary parts of the object. In Python, abstraction can be achieved using abstract classes and methods.

python

CopyEdit

from abc import ABC, abstractmethod

class Animal(ABC):

@abstractmethod

def sound(self):

pass

class Dog(Animal):

def sound(self):

return "Bark"

# dog = Animal() # This will raise an error since Animal is abstract

dog = Dog()

print(dog.sound()) # Bark

#### **c) Inheritance**

Inheritance allows a new class (child class) to inherit the properties and methods from an existing class (parent class). This promotes code reusability.

python

CopyEdit

class Animal:

def speak(self):

print("Animal speaks")

class Dog(Animal):

def speak(self):

print("Dog barks")

# Creating an object of the child class

dog = Dog()

dog.speak() # Dog barks

#### **d) Polymorphism**

Polymorphism allows methods to behave differently based on the object calling them. It enables objects of different classes to be treated as objects of a common superclass.

python

CopyEdit

class Animal:

def sound(self):

print("Animal sound")

class Dog(Animal):

def sound(self):

print("Bark")

class Cat(Animal):

def sound(self):

print("Meow")

# Polymorphism: both Dog and Cat objects are treated as Animal objects

def make\_sound(animal: Animal):

animal.sound()

make\_sound(Dog()) # Bark

make\_sound(Cat()) # Meow

### **3. What's an Object?**

An **object** is an instance of a **class**. It is a self-contained unit that consists of both **data** (attributes) and **functions** (methods) that operate on the data. Objects are the building blocks of OOP.

Example:

python

CopyEdit

class Car:

def \_\_init\_\_(self, make, model):

self.make = make

self.model = model

def display\_info(self):

print(f"Car Make: {self.make}, Model: {self.model}")

# Creating an object

my\_car = Car("Toyota", "Corolla")

my\_car.display\_info() # Car Make: Toyota, Model: Corolla

### **4. Native Data Types**

Python provides several **native data types** for storing data:

* **Numbers**: int, float, complex
* **String**: str
* **Boolean**: bool
* **List**: list
* **Tuple**: tuple
* **Dictionary**: dict
* **Set**: set

### **5. Declaring Variables**

Variables in Python do not need explicit declaration of type. Python automatically assigns the type based on the value assigned.

Example:

python

CopyEdit

x = 10 # integer

y = 3.14 # float

name = "Alice" # string

### **6. Referencing Variables**

Variables in Python hold **references** to objects, not the objects themselves. When you assign a variable to another, you are copying the reference, not the actual object.

Example:

python

CopyEdit

a = [1, 2, 3]

b = a # b refers to the same list as a

b.append(4)

print(a) # [1, 2, 3, 4]

### **7. Object References**

An object reference is a pointer to the object in memory. When you assign one object to another variable, both variables point to the same object.

python

CopyEdit

a = [1, 2, 3]

b = a

b[0] = 9

print(a) # [9, 2, 3]

### **8. Class and Object**

A **class** defines a blueprint for objects. It specifies what attributes and methods the objects will have. An **object** is an instance of a class.

Example:

python

CopyEdit

class Person:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

person1 = Person("John", 25) # Creating an object of class Person

print(person1.name) # John

### **9. Decorators**

A **decorator** is a function that takes another function and extends its behavior without explicitly modifying it. It is commonly used for logging, access control, etc.

python

CopyEdit

def decorator(func):

def wrapper():

print("Before function call")

func()

print("After function call")

return wrapper

@decorator

def say\_hello():

print("Hello!")

say\_hello()

### **10. Attributes**

Attributes are variables bound to an object. They are used to store data relevant to the object.

python

CopyEdit

class Person:

def \_\_init\_\_(self, name, age):

self.name = name # attribute

self.age = age # attribute

person = Person("Alice", 30)

print(person.name) # Alice

### **11. Inheritance**

Inheritance allows a class (child) to inherit methods and attributes from another class (parent), allowing code reuse and creating hierarchies.

python

CopyEdit

class Animal:

def speak(self):

print("Animal speaks")

class Dog(Animal):

def speak(self):

print("Dog barks")

dog = Dog()

dog.speak() # Dog barks

### **12. Overloading and Overriding**

* **Overloading**: Python does not support method overloading in the traditional sense. Instead, it supports multiple methods with default parameters.
* **Overriding**: A subclass can override a method from its superclass to change or extend its behavior.

python

CopyEdit

class Animal:

def sound(self):

print("Animal sound")

class Dog(Animal):

def sound(self): # Overriding the sound method

print("Bark")

dog = Dog()

dog.sound() # Bark

### **13. Data Hiding**

Data hiding refers to restricting access to an object's internal data. In Python, this can be achieved using private or protected attributes.

python

CopyEdit

class Person:

def \_\_init\_\_(self, name):

self.\_\_name = name # private attribute

def get\_name(self):

return self.\_\_name

person = Person("Alice")

print(person.get\_name()) # Alice

### **MCQs - Advanced Python (OOP)**

1. What is an example of an object in Python?

a) int  
b) str  
c) Person("John", 25)  
d) def function(): pass

**Answer**: c) Person("John", 25)

2. What does the \_\_init\_\_ method in a class do?

a) Initializes the values of variables  
b) Initializes the class name  
c) Initializes the object instance  
d) Initializes the module

**Answer**: c) Initializes the object instance

3. What will be the output of the following code?

python

CopyEdit

class Dog:

def \_\_init\_\_(self, name):

self.name = name

dog = Dog("Buddy")

print(dog.name)

a) None  
b) Buddy  
c) dog  
d) name

**Answer**: b) Buddy

4. Which of the following is an example of Encapsulation in Python?

a) Defining a private method  
b) Inheriting a class  
c) Using inheritance  
d) Using polymorphism

**Answer**: a) Defining a private method

5. What will be the output of the following code?

python

CopyEdit

class Animal:

def \_\_init\_\_(self):

self.sound = "Animal sound"

class Dog(Animal):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.sound = "Bark"

dog = Dog()

print(dog.sound)

a) Animal sound  
b) Bark  
c) None  
d) Error

**Answer**: b) Bark

6. Which of the following concepts allows an object to inherit properties and methods from another class?

a) Encapsulation  
b) Inheritance  
c) Polymorphism  
d) Abstraction

**Answer**: b) Inheritance

7. What is the output of the following code?

python

CopyEdit

class Animal:

def speak(self):

print("Animal speaks")

class Dog(Animal):

def speak(self):

print("Dog barks")

dog = Dog()

dog.speak()

a) Dog barks  
b) Animal speaks  
c) Dog speaks  
d) Animal barks

**Answer**: a) Dog barks

8. Which of the following is not a built-in Python data type?

a) List  
b) Set  
c) Object  
d) Queue

**Answer**: d) Queue

9. Which of the following statements about Python sets is true?

a) Sets maintain the order of elements  
b) Sets can store duplicate values  
c) Sets store elements of different data types  
d) Sets do not allow mutable objects

**Answer**: c) Sets store elements of different data types

10. What is the purpose of the super() function?

a) It calls a method in the current class  
b) It calls a method in the parent class  
c) It returns the parent class object  
d) It calls the \_\_init\_\_ method

**Answer**: b) It calls a method in the parent class

11. What will be the output of the following code?

python

CopyEdit

class Person:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

person1 = Person("Alice", 30)

person2 = Person("Bob", 25)

print(person1.name)

a) Bob  
b) 30  
c) Alice  
d) None

**Answer**: c) Alice

12. Which of the following is an example of Data Hiding?

a) Using public methods  
b) Declaring a class as private  
c) Using \_\_ before an attribute name  
d) Inheriting a class

**Answer**: c) Using \_\_ before an attribute name

13. What does the @staticmethod decorator do in Python?

a) It defines a method that can be called on an instance of the class  
b) It defines a method that does not need an instance to be called  
c) It defines an instance method  
d) It creates a class-level variable

**Answer**: b) It defines a method that does not need an instance to be called

14. Which of the following statements about class variables is true?

a) They are specific to instances of the class  
b) They are shared across all instances of the class  
c) They can only be used inside the class  
d) They can only store integers

**Answer**: b) They are shared across all instances of the class

15. What will be the output of the following code?

python

CopyEdit

class Car:

def \_\_init\_\_(self, make, model):

self.make = make

self.model = model

car1 = Car("Toyota", "Corolla")

car2 = Car("Honda", "Civic")

print(car1.make)

a) Honda  
b) Toyota  
c) None  
d) Civic

**Answer**: b) Toyota

16. What is Polymorphism in Python?

a) Combining multiple classes into one  
b) Method overloading  
c) A subclass using methods from its parent class  
d) One method behaves differently based on the object calling it

**Answer**: d) One method behaves differently based on the object calling it

17. Which of the following is not a characteristic of OOP in Python?

a) Classes  
b) Objects  
c) Functions  
d) Loops

**Answer**: d) Loops

18. What does the dir() function do in Python?

a) Returns the list of directories in the current working directory  
b) Returns a list of attributes and methods of an object  
c) Changes the current directory  
d) Returns the current directory path

**Answer**: b) Returns a list of attributes and methods of an object

19. What will be the output of the following code?

python

CopyEdit

class Shape:

def area(self):

return 0

class Circle(Shape):

def area(self):

return 3.14 \* 5 \* 5

circle = Circle()

print(circle.area())

a) 25  
b) 3.14 \* 25  
c) 78.5  
d) None

**Answer**: c) 78.5

20. What will be the output of the following code?

python

CopyEdit

class Person:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

person1 = Person("Alice", 30)

person2 = person1

person2.age = 35

print(person1.age)

a) 30  
b) 35  
c) None  
d) Error

**Answer**: b) 35

21. Which of the following is not a feature of Python classes?

a) Methods  
b) Inheritance  
c) Static variables  
d) Functions

**Answer**: d) Functions

22. What is the purpose of self in a Python class?

a) It is used to define class variables  
b) It is used to refer to the class itself  
c) It refers to the current object instance  
d) It is used to define methods in the class

**Answer**: c) It refers to the current object instance

23. What will be the output of the following code?

python

CopyEdit

class Animal:

def sound(self):

print("Animal makes a sound")

class Dog(Animal):

def sound(self):

print("Dog barks")

dog = Dog()

dog.sound()

a) Dog barks  
b) Animal makes a sound  
c) Animal barks  
d) None

**Answer**: a) Dog barks

24. What does the \_\_str\_\_ method in a Python class do?

a) It defines how the class is represented as a string  
b) It defines a method for initializing the object  
c) It defines a method for equality comparison  
d) It defines how to print the class object

**Answer**: a) It defines how the class is represented as a string

25. What is the output of the following code?

python

CopyEdit

class Shape:

def area(self):

print("This is a shape")

class Circle(Shape):

def area(self):

print("This is a circle")

circle = Circle()

circle.area()

a) This is a shape  
b) This is a circle  
c) This is a shape This is a circle  
d) Error

**Answer**: b) This is a circle

26. What does the isinstance() function do in Python?

a) Checks if an object is an instance of a class  
b) Compares two classes for equality  
c) Converts an object to a string  
d) Checks if an object is callable

**Answer**: a) Checks if an object is an instance of a class

27. What will be the output of the following code?

python

CopyEdit

class Test:

def \_\_init\_\_(self, a):

self.a = a

def \_\_add\_\_(self, other):

return self.a + other.a

obj1 = Test(5)

obj2 = Test(10)

result = obj1 + obj2

print(result)

a) 15  
b) 50  
c) 5  
d) Error

**Answer**: a) 15

28. What will be the output of the following code?

python

CopyEdit

def greet(name):

print("Hello, " + name)

greet("Alice")

a) Hello,  
b) Alice  
c) Hello, Alice  
d) None

**Answer**: c) Hello, Alice

29. Which of the following is not a method for defining a Python class?

a) class keyword  
b) def keyword  
c) Inheritance  
d) object keyword

**Answer**: b) def keyword

30. What is the role of a constructor in Python classes?

a) Initializes the class  
b) Initializes the attributes of an object  
c) Creates the object  
d) Defines methods for the class

**Answer**: b) Initializes the attributes of an object

31. What will be the output of the following code?

python

CopyEdit

class Animal:

def \_\_init\_\_(self):

self.name = "Animal"

def display\_name(self):

print(self.name)

class Dog(Animal):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.name = "Dog"

dog = Dog()

dog.display\_name()

a) Animal  
b) Dog  
c) None  
d) Error

**Answer**: b) Dog

32. What does the @classmethod decorator do in Python?

a) Defines a class-level method  
b) Defines an instance method  
c) Defines a static method  
d) Defines a method that does not require self

**Answer**: a) Defines a class-level method

33. What will be the output of the following code?

python

CopyEdit

class Animal:

def \_\_init\_\_(self, species):

self.species = species

class Dog(Animal):

def \_\_init\_\_(self, species, breed):

super().\_\_init\_\_(species)

self.breed = breed

dog = Dog("Canine", "Labrador")

print(dog.species, dog.breed)

a) Canine Labrador  
b) None None  
c) Canine  
d) Labrador

**Answer**: a) Canine Labrador

34. Which of the following is true about Python class inheritance?

a) A child class can inherit multiple parent classes  
b) A class can inherit multiple parent classes without using super()  
c) A class can inherit from a parent class only once  
d) Inheritance in Python is not supported

**Answer**: a) A child class can inherit multiple parent classes

35. What will be the output of the following code?

python

CopyEdit

class Base:

def \_\_init\_\_(self):

self.value = 10

class Derived(Base):

def \_\_init\_\_(self):

super().\_\_init\_\_()

self.value = 20

obj = Derived()

print(obj.value)

a) 10  
b) 20  
c) None  
d) Error

**Answer**: b) 20

36. Which of the following is a valid way to define a static method in Python?

a) @staticmethod decorator  
b) staticmethod() function  
c) @classmethod decorator  
d) def static() method

**Answer**: a) @staticmethod decorator

37. What will be the output of the following code?

python

CopyEdit

class MyClass:

def method(self):

print("Method of MyClass")

class ChildClass(MyClass):

def method(self):

print("Method of ChildClass")

obj = ChildClass()

obj.method()

a) Method of MyClass  
b) Method of ChildClass  
c) Method of MyClass Method of ChildClass  
d) Error

**Answer**: b) Method of ChildClass

38. What is the significance of the \_\_del\_\_ method in Python classes?

a) It is used to initialize the object  
b) It is called when an object is destroyed  
c) It is used to add functionality to the constructor  
d) It is used to hide internal class data

**Answer**: b) It is called when an object is destroyed

39. Which of the following is true about Python's \_\_new\_\_ method?

a) It is used to initialize class attributes  
b) It is called to create a new instance of a class  
c) It is called before the \_\_init\_\_ method  
d) It is used for object destruction

**Answer**: b) It is called to create a new instance of a class

40. What is the purpose of the pass keyword in Python?

a) To stop the function from executing  
b) To define an empty function or class  
c) To declare a method  
d) To define an infinite loop

**Answer**: b) To define an empty function or class

### **Session 14 & 15:**

### **1. Exception Handling**

Exception handling in Python is a mechanism used to handle runtime errors, which can occur during the execution of a program. It helps prevent the program from crashing by providing a way to handle errors gracefully. In Python, exception handling is done using the try, except, else, and finally blocks.

* **try block**: This block contains the code that might throw an exception. If an exception occurs, the rest of the code inside the try block is skipped.
* **except block**: This block is executed if an exception occurs in the try block. It allows you to handle the exception and take appropriate action (e.g., print an error message, log the error, etc.).
* **else block**: This block, if present, is executed if no exception occurs in the try block.
* **finally block**: This block, if present, is always executed, regardless of whether an exception occurs or not. It is typically used for clean-up actions, like closing files or network connections.

### **Syntax:**

python

CopyEdit

try:

# Code that might raise an exception

except ExceptionType:

# Code to handle the exception

else:

# Code to run if no exception occurs

finally:

# Code that runs no matter what

### **Example:**

python

CopyEdit

try:

num1 = int(input("Enter a number: "))

num2 = int(input("Enter another number: "))

result = num1 / num2

except ZeroDivisionError:

print("Error! Division by zero.")

except ValueError:

print("Invalid input! Please enter an integer.")

else:

print("The result is:", result)

finally:

print("Execution completed.")

**Explanation**:

* If a division by zero occurs, the except ZeroDivisionError block will execute.
* If invalid input (non-integer) is entered, the except ValueError block will execute.
* If no exception occurs, the else block will execute.
* The finally block will always execute, regardless of any exceptions.

### **2. except Clause**

The except clause is used to catch exceptions that are raised within the try block. You can specify the type of exception you want to catch or use a generic Exception to catch all exceptions.

**Example 1: Catching specific exceptions**

python

CopyEdit

try:

num = int(input("Enter a number: "))

except ValueError:

print("Error: Invalid input! Please enter an integer.")

**Example 2: Catching any exception**

python

CopyEdit

try:

num = int(input("Enter a number: "))

except Exception as e:

print(f"An error occurred: {e}")

In the second example, the Exception class is the base class for all exceptions in Python, so it catches all types of exceptions.

### **3. try-finally Clause**

The finally block is used for clean-up actions. It will always execute whether an exception occurs or not. It is typically used to close files, release resources, or close network connections.

**Syntax**:

python

CopyEdit

try:

# Code that might raise an exception

finally:

# Code that runs no matter what

**Example**:

python

CopyEdit

file = None

try:

file = open("example.txt", "r")

content = file.read()

print(content)

except FileNotFoundError:

print("File not found.")

finally:

if file:

file.close()

print("File has been closed.")

In this example, the finally block ensures that the file is closed, even if an error occurs while reading the file.

### **4. User-Defined Exceptions**

Python allows you to define your own exceptions by creating a custom exception class. This is helpful when you want to raise errors that are specific to your program logic. A user-defined exception must inherit from the built-in Exception class or one of its subclasses.

**Syntax**:

python

CopyEdit

class CustomError(Exception):

pass

Once the exception class is defined, you can raise the exception using the raise keyword.

**Example**:

python

CopyEdit

class NegativeNumberError(Exception):

"""Custom exception for negative numbers."""

pass

def calculate\_square\_root(x):

if x < 0:

raise NegativeNumberError("Cannot calculate the square root of a negative number")

return x \*\* 0.5

try:

result = calculate\_square\_root(-4)

except NegativeNumberError as e:

print(f"Error: {e}")

In this example, the NegativeNumberError is a custom exception that is raised when a negative number is passed to the calculate\_square\_root function.

### **Benefits of User-Defined Exceptions:**

* Custom exceptions allow you to define specific error conditions and handle them differently.
* They provide better readability and maintainability, especially in large projects.

### **Summary:**

1. **Exception Handling** in Python involves using the try, except, else, and finally blocks to manage errors.
2. The **except clause** catches exceptions raised in the try block, allowing the program to continue running instead of terminating abruptly.
3. The **try-finally clause** ensures that cleanup actions (like closing files or releasing resources) are performed regardless of whether an exception occurs.
4. **User-defined exceptions** allow you to create custom error types, which can make your code more readable and handle specific error conditions more effectively.

### **MCQs**

1. What will the following code output?

python

CopyEdit

try:

num = int(input("Enter a number: "))

except ValueError:

print("Invalid input!")

else:

print("Input is valid.")

a) Invalid input!  
b) Input is valid.  
c) Error  
d) Nothing

**Answer**: b) Input is valid.

2. What happens when an exception occurs inside the try block, and there is no matching except block?

a) The program crashes  
b) The else block is executed  
c) The finally block is skipped  
d) A NameError is raised

**Answer**: a) The program crashes

3. Which of the following is true about the finally block?

a) It is executed only if no exception occurs.  
b) It is executed only if an exception occurs.  
c) It is executed regardless of whether an exception occurs.  
d) It is executed only when there is an error in the except block.

**Answer**: c) It is executed regardless of whether an exception occurs.

4. What is the output of the following code?

python

CopyEdit

try:

x = 5 / 0

except ZeroDivisionError:

print("Error")

finally:

print("Finally")

a) Error  
b) Error\nFinally  
c) Finally  
d) ZeroDivisionError

**Answer**: b) Error\nFinally

5. What is the purpose of the else block in exception handling?

a) It runs when an exception is raised.  
b) It runs when no exception is raised.  
c) It is executed when the program crashes.  
d) It is used to catch the exception.

**Answer**: b) It runs when no exception is raised.

6. What will the following code print?

python

CopyEdit

try:

a = int(input("Enter an integer: "))

except ValueError:

print("Invalid input!")

else:

print(f"The number is {a}")

finally:

print("Execution complete.")

a) Invalid input!  
b) The number is {a}  
c) The number is 5  
d) Execution complete.

**Answer**: b) The number is {a} (if valid integer is entered)

7. Which statement is used to raise an exception in Python?

a) throw  
b) raise  
c) except  
d) finally

**Answer**: b) raise

8. What will happen if an exception is raised in the finally block?

a) The program will terminate immediately.  
b) The exception will be caught by the nearest except block.  
c) The exception will be ignored.  
d) The program will ignore any further exceptions in the try block.

**Answer**: b) The exception will be caught by the nearest except block.

9. What is the output of the following code?

python

CopyEdit

try:

x = 5 / 0

except ZeroDivisionError:

print("Zero Division Error")

except Exception:

print("General Error")

a) Zero Division Error  
b) General Error  
c) Error  
d) None

**Answer**: a) Zero Division Error

10. Which of the following is the correct way to define a custom exception in Python?

a) class MyException: pass  
b) class MyException(Exception): pass  
c) def MyException(): pass  
d) raise MyException()

**Answer**: b) class MyException(Exception): pass

11. What will the following code print?

python

CopyEdit

class NegativeNumberError(Exception):

pass

def square\_root(x):

if x < 0:

raise NegativeNumberError("Cannot calculate square root of a negative number.")

return x \*\* 0.5

try:

print(square\_root(-9))

except NegativeNumberError as e:

print(e)

a) Cannot calculate square root of a negative number.  
b) -3.0  
c) None  
d) Error

**Answer**: a) Cannot calculate square root of a negative number.

12. What is the output of the following code?

python

CopyEdit

try:

print(10 / 2)

except ZeroDivisionError:

print("Cannot divide by zero")

else:

print("No errors occurred")

a) Cannot divide by zero  
b) No errors occurred  
c) 5  
d) Error

**Answer**: c) 5

13. What happens if no except block is provided for a specific exception?

a) The program crashes with an uncaught exception.  
b) The program ignores the exception and continues.  
c) The program catches the exception using the finally block.  
d) The program raises another exception.

**Answer**: a) The program crashes with an uncaught exception.

14. How would you handle multiple types of exceptions in a single except block?

a) Use multiple except blocks.  
b) Use an or condition in the except block.  
c) Use the as keyword to handle multiple exceptions.  
d) Use a tuple to list exceptions in the except block.

**Answer**: d) Use a tuple to list exceptions in the except block.

15. What is the output of the following code?

python

CopyEdit

try:

x = int("abc")

except ValueError:

print("ValueError caught")

finally:

print("Execution complete")

a) ValueError caught  
b) Execution complete  
c) ValueError caught\nExecution complete  
d) Error

**Answer**: c) ValueError caught\nExecution complete

16. Which of the following is true about try blocks in Python?

a) A try block can exist without an except block.  
b) A try block is required to have both except and else blocks.  
c) A try block should be followed by a finally block.  
d) A try block must have at least one except block.

**Answer**: a) A try block can exist without an except block.

17. What will happen if an exception is raised in the else block?

a) The exception will be caught by the nearest except block.  
b) The program will terminate immediately.  
c) The finally block will still execute.  
d) None of the above.

**Answer**: c) The finally block will still execute.

18. What does the raise keyword do?

a) It raises the priority of an exception.  
b) It is used to call an exception explicitly.  
c) It ends the current execution of the program.  
d) It terminates the program.

**Answer**: b) It is used to call an exception explicitly.

19. What is the purpose of the \_\_init\_\_ method in a user-defined exception class?

a) To initialize the custom exception message.  
b) To raise the exception.  
c) To define the exception type.  
d) To handle the exception.

**Answer**: a) To initialize the custom exception message.

20. What will be the output of the following code?

python

CopyEdit

class CustomError(Exception):

pass

def demo():

raise CustomError("This is a custom error")

try:

demo()

except CustomError as e:

print(f"Caught error: {e}")

a) Caught error: This is a custom error  
b) This is a custom error  
c) CustomError: This is a custom error  
d) Error

**Answer**: a) Caught error: This is a custom erro

## **Session 16: Libraries and Functional Programming, Debugging, and Logging in Python**

### **1. Libraries and Functional Programming in Python**

Python is well-known for its extensive library support, making it a powerful language for various applications. Additionally, functional programming is a paradigm that Python supports alongside object-oriented and procedural programming.

#### **Libraries in Python**

A **library** in Python is a collection of modules and functions that provide pre-written code to perform specific tasks. Instead of writing everything from scratch, you can use libraries to speed up development.

##### **Popular Python Libraries**

* **NumPy** – Used for numerical computing and handling arrays.
* **Pandas** – Used for data manipulation and analysis.
* **Matplotlib & Seaborn** – Used for data visualization.
* **Scikit-learn** – Used for machine learning.
* **TensorFlow & PyTorch** – Used for deep learning.
* **Requests** – Used for making HTTP requests.
* **BeautifulSoup** – Used for web scraping.

##### **How to Install and Use Libraries**

Libraries can be installed using pip:

bash

CopyEdit

pip install numpy

To use a library, you need to import it in your Python script:

python

CopyEdit

import numpy as np

arr = np.array([1, 2, 3, 4, 5])

print(arr)

#### **Functional Programming in Python**

Functional programming is a programming paradigm that treats computation as the evaluation of mathematical functions and avoids changing state or mutable data.

##### **Key Functional Programming Concepts in Python**

1. **Higher-order functions** – Functions that take other functions as arguments or return functions.
2. **Lambda functions** – Anonymous functions that are useful for short operations.
3. **Map, Filter, and Reduce** – Functional operations used to process collections.
4. **Recursion** – A function calling itself to solve problems.
5. **Immutability** – Encourages using immutable data structures.

##### **Examples of Functional Programming**

* **Lambda Functions:**

python

CopyEdit

square = lambda x: x \* x

print(square(5)) # Output: 25

* **Map Function:**

python

CopyEdit

nums = [1, 2, 3, 4, 5]

squared = list(map(lambda x: x \* x, nums))

print(squared) # Output: [1, 4, 9, 16, 25]

* **Filter Function:**

python

CopyEdit

nums = [1, 2, 3, 4, 5]

evens = list(filter(lambda x: x % 2 == 0, nums))

print(evens) # Output: [2, 4]

* **Reduce Function:**

python

CopyEdit

from functools import reduce

nums = [1, 2, 3, 4, 5]

sum\_nums = reduce(lambda x, y: x + y, nums)

print(sum\_nums) # Output: 15

### **2. Debugging Basics**

Debugging is the process of finding and fixing errors in a program. Python provides several tools and techniques to help debug code.

#### **Common Types of Errors in Python**

1. **Syntax Errors** – Occur when the code is not written correctly.
2. **Runtime Errors (Exceptions)** – Occur when the program encounters invalid operations.
3. **Logical Errors** – The program runs but produces incorrect output.

#### **Debugging Techniques**

* **Print Statements:** Using print() to check values at different points in the program.
* **Using the assert Statement:** Helps in testing assumptions.

python

CopyEdit

x = 10

assert x > 0, "x should be positive"

* **Using Python Debugger (pdb):** Allows stepping through code execution.

python

CopyEdit

import pdb

def divide(a, b):

pdb.set\_trace() # Debugging starts here

return a / b

print(divide(10, 2))

Commands in pdb:

* n – Next line
* s – Step into a function
* p variable – Print variable
* q – Quit debugging
* **Using IDE Debugging Tools:** PyCharm, VS Code, and Jupyter Notebook have built-in debugging tools.

### **3. Logging Using Python**

Logging is a way to track events that happen during a program’s execution. It helps in debugging and monitoring applications.

#### **Why Use Logging Instead of Print?**

* print() is for debugging, while logging is for recording events.
* Logging allows setting different levels of importance.
* Logging can be saved to a file for future analysis.

#### **Logging Levels**

Python provides different logging levels to categorize messages:

* DEBUG – Detailed information, typically used for diagnosing issues.
* INFO – Confirmation that things are working as expected.
* WARNING – An indication of something unexpected, but the program still runs.
* ERROR – A more serious issue that prevents the program from continuing.
* CRITICAL – A serious error that forces the program to stop.

#### **Basic Logging Example**

python

CopyEdit

import logging

# Configure logging

logging.basicConfig(level=logging.DEBUG, format='%(levelname)s: %(message)s')

logging.debug("This is a debug message")

logging.info("This is an info message")

logging.warning("This is a warning")

logging.error("This is an error")

logging.critical("This is critical")

**Output:**

vbnet

CopyEdit

DEBUG: This is a debug message

INFO: This is an info message

WARNING: This is a warning

ERROR: This is an error

CRITICAL: This is critical

#### **Logging to a File**

python

CopyEdit

import logging

logging.basicConfig(filename="app.log", level=logging.DEBUG,

format='%(asctime)s - %(levelname)s - %(message)s')

logging.info("This message will be written to a file")

#### **Using Different Logging Levels in a Function**

python

CopyEdit

import logging

def divide(a, b):

try:

result = a / b

logging.info(f"Division successful: {result}")

return result

except ZeroDivisionError:

logging.error("Attempted to divide by zero!")

divide(10, 2)

divide(10, 0)

## **Summary**

### **Libraries and Functional Programming**

* Libraries in Python provide pre-written functions for various tasks.
* Functional programming in Python involves higher-order functions, lambda expressions, and operations like map, filter, and reduce.

### **Debugging Basics**

* Debugging is identifying and fixing errors in a program.
* Python provides tools like print(), assert, and the pdb debugger.

### **Logging in Python**

* Logging helps record important events during program execution.
* Python has built-in logging levels: DEBUG, INFO, WARNING, ERROR, and CRITICAL.
* Logs can be written to a file for monitoring.

## MCQs on Libraries & Functional Programming

1. What is the main advantage of using Python libraries?

A) Increases program execution time  
B) Provides reusable code and speeds up development  
C) Makes Python slower  
D) Forces the use of object-oriented programming

✅ **Answer:** B) Provides reusable code and speeds up development

2. How do you install external libraries in Python?

A) install package <library\_name>  
B) pip install <library\_name>  
C) python -m install <library\_name>  
D) import <library\_name>

✅ **Answer:** B) pip install <library\_name>

3. Which library is used for machine learning in Python?

A) Pandas  
B) NumPy  
C) Scikit-learn  
D) Matplotlib

✅ **Answer:** C) Scikit-learn

4. What will be the output of the following code?

python

CopyEdit

square = lambda x: x \* x

print(square(3))

A) 6  
B) 9  
C) 3  
D) Error

✅ **Answer:** B) 9

5. Which function is used to apply a function to each element of a list?

A) map()  
B) filter()  
C) reduce()  
D) apply()

✅ **Answer:** A) map()

6. What does the filter() function return?

A) A new list with transformed elements  
B) A list of elements that satisfy a condition  
C) The sum of all elements  
D) The first element of the list

✅ **Answer:** B) A list of elements that satisfy a condition

7. What is the output of the following code?

python

CopyEdit

from functools import reduce

nums = [1, 2, 3, 4]

result = reduce(lambda x, y: x + y, nums)

print(result)

A) 10  
B) [1, 2, 3, 4]  
C) 24  
D) (1, 2, 3, 4)

✅ **Answer:** A) 10

8. Which functional programming technique does NOT exist in Python?

A) map()  
B) filter()  
C) reduce()  
D) execute()

✅ **Answer:** D) execute()

9. What does the following code do?

python

CopyEdit

numbers = [2, 4, 6, 8]

doubles = list(map(lambda x: x \* 2, numbers))

print(doubles)

A) Multiplies all elements in the list by 2  
B) Filters out even numbers  
C) Concatenates two lists  
D) Returns an error

✅ **Answer:** A) Multiplies all elements in the list by 2

10. What will be the output of this Python code?

python

CopyEdit

nums = [1, 2, 3, 4, 5]

evens = list(filter(lambda x: x % 2 == 0, nums))

print(evens)

A) [1, 3, 5]  
B) [2, 4]  
C) [1, 2, 3, 4, 5]  
D) Error

✅ **Answer:** B) [2, 4]

MCQs on Debugging Basics

11. Which module in Python is used for debugging?

A) debug  
B) pdb  
C) traceback  
D) log

✅ **Answer:** B) pdb

12. What does pdb.set\_trace() do?

A) Stops the execution of the program  
B) Prints the variable values  
C) Starts an interactive debugging session  
D) Skips to the next statement

✅ **Answer:** C) Starts an interactive debugging session

13. What will be the output of the following code?

python

CopyEdit

x = 5

assert x > 10, "x should be greater than 10"

A) No output  
B) AssertionError: x should be greater than 10  
C) x should be greater than 10  
D) Error in syntax

✅ **Answer:** B) AssertionError: x should be greater than 10

14. Which statement helps in stopping code execution for debugging?

A) assert  
B) break  
C) pause  
D) exit()

✅ **Answer:** A) assert

15. What does the traceback module do?

A) Logs error messages  
B) Provides debugging commands  
C) Shows the recent call stack of an error  
D) Stops the program execution

✅ **Answer:** C) Shows the recent call stack of an error

MCQs on Logging in Python

16. Which function is used to log an error message?

A) log.error()  
B) logging.error()  
C) log.debug()  
D) log.warning()

✅ **Answer:** B) logging.error()

17. What will the following logging configuration do?

python

CopyEdit

import logging

logging.basicConfig(level=logging.WARNING)

A) Only log messages with level WARNING and above  
B) Log all messages  
C) Log only ERROR messages  
D) Raise an exception for warnings

✅ **Answer:** A) Only log messages with level WARNING and above

18. What is the correct way to log messages to a file?

A) logging.to\_file('logfile.log')  
B) logging.write('logfile.log')  
C) logging.basicConfig(filename='logfile.log', level=logging.DEBUG)  
D) logfile = open('logfile.log', 'w')

✅ **Answer:** C) logging.basicConfig(filename='logfile.log', level=logging.DEBUG)

19. What is the default logging level in Python?

A) DEBUG  
B) INFO  
C) WARNING  
D) ERROR

✅ **Answer:** C) WARNING

20. What is the output of the following code?

python

CopyEdit

import logging

logging.basicConfig(level=logging.INFO)

logging.debug("Debug message")

logging.info("Info message")

logging.warning("Warning message")

A)

mathematica

CopyEdit

Debug message

Info message

Warning message

B)

CopyEdit

Info message

Warning message

C)

CopyEdit

Warning message

D) No output

✅ **Answer:** B)

CopyEdit

Info message

Warning message

### **Session 17, 18 & 19: NumPy, SciPy, Pandas, and Data Wrangling with Pandas**

These sessions focus on the **NumPy, SciPy, and Pandas** libraries, which are fundamental for data analysis, numerical computing, and data manipulation in Python. Let's explore each topic in detail.

## **1️⃣ Working with NumPy**

NumPy (Numerical Python) is a powerful library for numerical computing in Python. It provides support for **multi-dimensional arrays**, **mathematical functions**, and **linear algebra operations**.

### **🔹 Installing and Importing NumPy**

To use NumPy, you need to install it first:

bash

CopyEdit

pip install numpy

Then import it into your Python program:

python

CopyEdit

import numpy as np

### **🔹 NumPy Arrays (ndarray)**

NumPy arrays (ndarray) are faster and more efficient than Python lists.

#### **Creating NumPy Arrays**

python

CopyEdit

arr1 = np.array([1, 2, 3, 4]) # 1D array

arr2 = np.array([[1, 2, 3], [4, 5, 6]]) # 2D array

arr3 = np.zeros((2, 3)) # Array of zeros

arr4 = np.ones((3, 3)) # Array of ones

arr5 = np.arange(0, 10, 2) # Array from 0 to 10 with step 2

arr6 = np.linspace(0, 1, 5) # 5 equally spaced values from 0 to 1

### **🔹 NumPy Array Operations**

python

CopyEdit

a = np.array([1, 2, 3])

b = np.array([4, 5, 6])

print(a + b) # [5 7 9]

print(a \* 2) # [2 4 6]

print(np.dot(a, b)) # Dot product: (1\*4 + 2\*5 + 3\*6) = 32

### **🔹 Indexing and Slicing**

python

CopyEdit

arr = np.array([10, 20, 30, 40, 50])

print(arr[2]) # 30

print(arr[1:4]) # [20, 30, 40]

print(arr[::-1]) # Reverse array: [50, 40, 30, 20, 10]

## **2️⃣ Working with SciPy**

SciPy (Scientific Python) is built on top of NumPy and provides additional functionality for scientific computing, including optimization, signal processing, and statistics.

### **🔹 Installing and Importing SciPy**

bash

CopyEdit

pip install scipy

python

CopyEdit

import scipy as sp

from scipy import stats, optimize

### **🔹 Key SciPy Modules**

| **Module** | **Purpose** |
| --- | --- |
| scipy.stats | Statistical functions |
| scipy.optimize | Optimization functions |
| scipy.linalg | Linear algebra operations |
| scipy.spatial | Spatial data structures |

### **🔹 Example: Finding Mean and Standard Deviation**

python

CopyEdit

from scipy import stats

data = [10, 20, 30, 40, 50]

print("Mean:", stats.tmean(data)) # Mean

print("Standard Deviation:", stats.tstd(data)) # Standard Deviation

### **🔹 Example: Solving an Optimization Problem**

python

CopyEdit

from scipy.optimize import minimize

def function\_to\_minimize(x):

return x\*\*2 + 5\*x + 6

result = minimize(function\_to\_minimize, x0=0) # x0 is the starting point

print("Optimal Value:", result.x)

## **3️⃣ Working with Pandas**

Pandas is a library used for **data manipulation and analysis**. It provides two main data structures:

* **Series** (1D labeled array)
* **DataFrame** (2D labeled table)

### **🔹 Installing and Importing Pandas**

bash

CopyEdit

pip install pandas

python

CopyEdit

import pandas as pd

### **🔹 Creating a Pandas Series**

python

CopyEdit

data = [10, 20, 30, 40]

series = pd.Series(data, index=["A", "B", "C", "D"])

print(series)

**Output:**

css

CopyEdit

A 10

B 20

C 30

D 40

dtype: int64

### **🔹 Creating a Pandas DataFrame**

python

CopyEdit

data = {"Name": ["Alice", "Bob", "Charlie"],

"Age": [25, 30, 35],

"Salary": [50000, 60000, 70000]}

df = pd.DataFrame(data)

print(df)

**Output:**

markdown

CopyEdit

Name Age Salary

0 Alice 25 50000

1 Bob 30 60000

2 Charlie 35 70000

### **🔹 Accessing Data in DataFrames**

python

CopyEdit

print(df["Name"]) # Access a column

print(df.iloc[1]) # Access a row by index

print(df.loc[0, "Age"]) # Access a specific value

### **🔹 Adding and Removing Columns**

python

CopyEdit

df["City"] = ["New York", "London", "Paris"] # Add a new column

df.drop(columns=["Salary"], inplace=True) # Remove column

print(df)

## **4️⃣ Data Wrangling with Pandas**

Data Wrangling (or Data Cleaning) is the process of preparing messy data for analysis.

### **🔹 Handling Missing Values**

python

CopyEdit

df = pd.DataFrame({"A": [1, 2, None], "B": [5, None, 9]})

df.fillna(0, inplace=True) # Replace NaN with 0

print(df)

### **🔹 Filtering Data**

python

CopyEdit

df = pd.DataFrame({"Name": ["Alice", "Bob", "Charlie"],

"Age": [25, 30, 35]})

filtered\_df = df[df["Age"] > 28] # Select rows where Age > 28

print(filtered\_df)

### **🔹 Grouping and Aggregation**

python

CopyEdit

df = pd.DataFrame({"Department": ["HR", "IT", "HR", "IT"],

"Salary": [50000, 60000, 52000, 62000]})

grouped = df.groupby("Department").mean()

print(grouped)

### **🔹 Merging DataFrames**

python

CopyEdit

df1 = pd.DataFrame({"ID": [1, 2, 3], "Name": ["Alice", "Bob", "Charlie"]})

df2 = pd.DataFrame({"ID": [1, 2, 3], "Salary": [50000, 60000, 70000]})

merged\_df = pd.merge(df1, df2, on="ID")

print(merged\_df)

## **🔹 Summary**

| **Topic** | **Description** |
| --- | --- |
| **NumPy** | Efficient numerical computing with arrays |
| **SciPy** | Scientific computing, optimization, and statistics |
| **Pandas** | Data analysis and manipulation |
| **Data Wrangling** | Cleaning and preparing data for analysis |

These topics are **foundational** for working with data science, machine learning, and analytics in Python. Let me know if you need **MCQs** to test your knowledge! 🚀

### **🔹 Easy Level**

**1. What does NumPy stand for?**a) Numerical Python ✅  
b) Number Python  
c) New Python  
d) Named Python

**2. How do you import NumPy in Python?**a) import numpy as np ✅  
b) import np  
c) import NumPy  
d) import numpy\_package

**3. What is the correct function to create a NumPy array?**a) np.array([1,2,3]) ✅  
b) np.create([1,2,3])  
c) np.list([1,2,3])  
d) np.array[1,2,3]

**4. What will np.zeros((2,3)) return?**a) A 2x3 matrix filled with zeros ✅  
b) A 3x2 matrix filled with ones  
c) A single zero value  
d) An empty array

**5. What will np.arange(1,10,2) return?**a) [1, 3, 5, 7, 9] ✅  
b) [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]  
c) [1, 10, 2]  
d) [10, 1, 2]

**6. What is Pandas primarily used for?**a) Image Processing  
b) Data Manipulation ✅  
c) Game Development  
d) Operating System Design

**7. What function creates a Pandas DataFrame?**a) pd.DataFrame() ✅  
b) pd.createData()  
c) pd.makeDF()  
d) pd.listDF()

**8. How do you check the first five rows of a Pandas DataFrame?**a) df.head() ✅  
b) df.top(5)  
c) df.start(5)  
d) df.rows(5)

**9. How do you remove missing values from a DataFrame?**a) df.dropna() ✅  
b) df.removeNaN()  
c) df.fillna()  
d) df.cleanNaN()

**10. What will df['column\_name'].mean() return?**a) The mean of the column values ✅  
b) The sum of column values  
c) The max value in the column  
d) The median value

### **🔹 Medium Level**

**11. What function returns the shape of a NumPy array?**a) array.shape ✅  
b) array.size  
c) array.length()  
d) array.form()

**12. What will np.linspace(0,10,5) generate?**a) [0, 2.5, 5, 7.5, 10] ✅  
b) [0, 5, 10]  
c) [0, 10, 5, 15]  
d) [0, 1, 2, 3, 4]

**13. What will df.iloc[1] return?**a) The first row ✅  
b) The last row  
c) The first column  
d) The last column

**14. What does df.loc[2, 'Age'] do?**a) Access the value in row index 2, column 'Age' ✅  
b) Delete row index 2  
c) Print the column 'Age'  
d) Return the entire DataFrame

**15. What is SciPy used for?**a) Scientific Computing ✅  
b) Web Development  
c) Game Design  
d) Cryptography

**16. What function is used to optimize a function in SciPy?**a) minimize() ✅  
b) optimize()  
c) reduce()  
d) maximization()

**17. What does df.groupby('Department').mean() do?**a) Groups data by 'Department' and finds the mean of each group ✅  
b) Deletes 'Department' column  
c) Sorts the DataFrame by 'Department'  
d) Filters out missing values

**18. What function merges two Pandas DataFrames?**a) pd.merge(df1, df2, on='ID') ✅  
b) pd.combine(df1, df2, on='ID')  
c) pd.concat(df1, df2, on='ID')  
d) df1.add(df2, on='ID')

**19. What function is used to check for missing values in a DataFrame?**a) df.isnull() ✅  
b) df.findNaN()  
c) df.checkNaN()  
d) df.nan()

**20. What function replaces NaN values with a specific value?**a) df.fillna(0) ✅  
b) df.removeNaN()  
c) df.replaceNaN(0)  
d) df.clearNaN()

### **🔹 Hard Level**

**21. How do you create a 3x3 identity matrix in NumPy?**a) np.eye(3) ✅  
b) np.identity\_matrix(3)  
c) np.create\_identity(3,3)  
d) np.unit\_matrix(3)

**22. What does np.dot(a, b) compute?**a) Dot product of two arrays ✅  
b) Sum of two arrays  
c) Mean of two arrays  
d) The maximum of two arrays

**23. What will df.sort\_values(by='Salary', ascending=False) do?**a) Sort DataFrame by 'Salary' in descending order ✅  
b) Remove the 'Salary' column  
c) Filter rows where 'Salary' > 0  
d) Count unique 'Salary' values

**24. What is the default join type for pd.merge()?**a) Inner Join ✅  
b) Outer Join  
c) Left Join  
d) Right Join

**25. What does df.pivot\_table(values='Salary', index='Department', aggfunc=np.mean) do?**a) Creates a pivot table with mean salary per department ✅  
b) Deletes 'Salary'  
c) Counts salary values  
d) Finds min salary

### **🔹 Bonus Code-Based Questions**

**26. What will be the output?**

python

CopyEdit

arr = np.array([10, 20, 30, 40])

print(arr[arr > 20])

a) [30, 40] ✅  
b) [20, 30, 40]  
c) [10, 20]  
d) Error

**27. What does np.random.rand(3,3) generate?**a) A 3x3 matrix of random values between 0 and 1 ✅  
b) A 3x3 matrix of random integers  
c) A 3x3 matrix of zeros  
d) A 3x3 matrix of ones

**28. What will be the output of the following Pandas code?**

python

CopyEdit

df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})

print(df.T)

a) **Transpose of the DataFrame** ✅  
b) **Swaps rows and columns**c) **Adds a new column**d) **Throws an error**

**29. What will the following code output?**

python

CopyEdit

df = pd.DataFrame({'col1': [1, 2, None, 4]})

print(df.dropna())

a) **Removes the row with None value** ✅  
b) **Replaces None with 0**c) **Throws an error**d) **Returns the DataFrame unchanged**

**30. What function is used to concatenate two DataFrames vertically?**a) pd.concat([df1, df2], axis=0) ✅  
b) pd.merge(df1, df2, axis=0)  
c) pd.append([df1, df2])  
d) df1.join(df2)

**31. What function is used to convert a Pandas DataFrame into a NumPy array?**a) df.to\_numpy() ✅  
b) df.array()  
c) df.as\_matrix()  
d) df.to\_array()

**32. What will np.max(np.array([[1,2,3], [4,5,6], [7,8,9]]), axis=0) return?**a) [7, 8, 9] ✅  
b) [3, 6, 9]  
c) [1, 4, 7]  
d) [1, 2, 3]

**33. What will np.argmax(np.array([3, 7, 1, 8, 5])) return?**a) 3 ✅  
b) 1  
c) 2  
d) 4

**34. How do you create a NumPy array of random integers between 1 and 100 with a shape of (3,3)?**a) np.random.randint(1, 100, (3,3)) ✅  
b) np.randint(1, 100, (3,3))  
c) np.rand(1, 100, (3,3))  
d) np.array.random((3,3))

**35. Which function finds unique values in a NumPy array?**a) np.unique(arr) ✅  
b) np.find\_unique(arr)  
c) np.distinct(arr)  
d) np.unique\_values(arr)

**36. What does df.describe() return in Pandas?**a) **Summary statistics (count, mean, std, min, max, etc.)** ✅  
b) **DataFrame column names**c) **Unique values in each column**d) **A list of missing values**

**37. What will df['Salary'].nlargest(3) return?**a) **Top 3 highest salaries** ✅  
b) **Bottom 3 salaries**c) **Average salary**d) **Median salary**

**38. What will np.sort(np.array([4, 1, 3, 9, 2])) return?**a) [1, 2, 3, 4, 9] ✅  
b) [4, 1, 3, 9, 2]  
c) [9, 4, 3, 2, 1]  
d) [2, 3, 1, 4, 9]

**39. What function is used to apply a function to all elements in a Pandas DataFrame?**a) df.apply() ✅  
b) df.function()  
c) df.transform()  
d) df.use()

**40. What will be the output of the following code?**

python

CopyEdit

df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})

df['C'] = df['A'] + df['B']

print(df)

a) A new column 'C' with element-wise sum of 'A' and 'B' ✅  
b) An error  
c) Only column 'C'  
d) A DataFrame with only column 'A' and 'B'

### **Session 20 & 21: Data Visualization in Python**

In these sessions, we will explore some of the most powerful **Python libraries for data visualization**:

* **Matplotlib & Seaborn**
* **ggplot & Plotly**

## **🔹 Working with Matplotlib & Seaborn**

### **1️⃣ Matplotlib: The Foundation of Data Visualization in Python**

Matplotlib is a **low-level** but powerful plotting library that provides complete control over **graphs, charts, and plots**.

### **🔹 Key Features of Matplotlib**

✅ **Supports multiple plot types** (Line plot, Bar plot, Histogram, Scatter plot, etc.)  
✅ **Highly customizable** (Colors, labels, grid, titles, etc.)  
✅ **Can integrate with Pandas & NumPy**

### **🔹 Installing & Importing Matplotlib**

python

CopyEdit

# Install Matplotlib (if not already installed)

pip install matplotlib

# Import Matplotlib

import matplotlib.pyplot as plt

### **📌 Basic Line Plot in Matplotlib**

python

CopyEdit

import matplotlib.pyplot as plt

x = [1, 2, 3, 4, 5]

y = [10, 20, 30, 40, 50]

plt.plot(x, y, marker='o', linestyle='-', color='b')

plt.xlabel('X-axis')

plt.ylabel('Y-axis')

plt.title('Simple Line Plot')

plt.show()

🔹 **Explanation:**

* plt.plot(x, y): Plots a line connecting points (x, y).
* marker='o': Marks each data point.
* linestyle='-': Uses a solid line.
* color='b': Uses **blue** color.
* plt.show(): Displays the plot.

### **📌 Creating a Bar Chart in Matplotlib**

python

CopyEdit

import matplotlib.pyplot as plt

categories = ['A', 'B', 'C', 'D']

values = [10, 20, 15, 25]

plt.bar(categories, values, color=['red', 'blue', 'green', 'purple'])

plt.xlabel('Categories')

plt.ylabel('Values')

plt.title('Bar Chart Example')

plt.show()

### **📌 Creating a Histogram in Matplotlib**

Histograms are used to visualize the **distribution of data**.

python

CopyEdit

import numpy as np

import matplotlib.pyplot as plt

data = np.random.randn(1000) # Generate 1000 random numbers

plt.hist(data, bins=20, color='skyblue', edgecolor='black')

plt.xlabel('Value')

plt.ylabel('Frequency')

plt.title('Histogram Example')

plt.show()

🔹 **Key Argument**: bins=20 defines the **number of bars** in the histogram.

### **2️⃣ Seaborn: Simplified Data Visualization**

Seaborn is **built on top of Matplotlib** and provides a **simpler, high-level interface** for creating beautiful statistical plots.

### **🔹 Installing & Importing Seaborn**

python

CopyEdit

pip install seaborn

import seaborn as sns

import matplotlib.pyplot as plt

### **📌 Creating a Scatter Plot in Seaborn**

python

CopyEdit

import seaborn as sns

import pandas as pd

import matplotlib.pyplot as plt

# Sample dataset

data = pd.DataFrame({'X': [1, 2, 3, 4, 5], 'Y': [10, 15, 25, 30, 50]})

sns.scatterplot(x='X', y='Y', data=data, color='red')

plt.title('Seaborn Scatter Plot')

plt.show()

### **📌 Creating a Box Plot in Seaborn**

Box plots are useful for visualizing the **distribution and outliers** in data.

python

CopyEdit

import seaborn as sns

import pandas as pd

import matplotlib.pyplot as plt

# Sample data

data = pd.DataFrame({'Category': ['A', 'B', 'A', 'B', 'A', 'B'],

'Values': [10, 20, 15, 25, 30, 35]})

sns.boxplot(x='Category', y='Values', data=data)

plt.title('Seaborn Box Plot')

plt.show()

## **🔹 Working with ggplot & Plotly**

### **3️⃣ ggplot: Inspired by R’s ggplot2**

**ggplot** is a Python visualization library inspired by **ggplot2 in R**. It is part of the plotnine library.

### **🔹 Installing & Importing ggplot**

python

CopyEdit

pip install plotnine

from plotnine import ggplot, aes, geom\_point

import pandas as pd

### **📌 Creating a Scatter Plot with ggplot**

python

CopyEdit

from plotnine import ggplot, aes, geom\_point

import pandas as pd

df = pd.DataFrame({'x': [1, 2, 3, 4, 5], 'y': [10, 20, 30, 40, 50]})

plot = ggplot(df, aes(x='x', y='y')) + geom\_point(color='red')

print(plot)

🔹 **ggplot Uses a Layered Approach:**

* aes(x, y): Maps **x & y** values.
* geom\_point(): Creates a scatter plot.

### **4️⃣ Plotly: Interactive Plots in Python**

Plotly is a **web-based visualization** library that allows for **interactive plots**.

### **🔹 Installing & Importing Plotly**

python

CopyEdit

pip install plotly

import plotly.express as px

### **📌 Creating an Interactive Line Plot in Plotly**

python

CopyEdit

import plotly.express as px

df = px.data.gapminder().query("country == 'India'")

fig = px.line(df, x="year", y="gdpPercap", title="GDP Over Time")

fig.show()

🔹 **Key Features:**✅ **Zooming & Hover Effects**✅ **Interactive Legends**✅ **Web-based & Responsive**

## **🔹 Summary Table: Comparing Visualization Libraries**

| **Feature** | **Matplotlib** | **Seaborn** | **ggplot** | **Plotly** |
| --- | --- | --- | --- | --- |
| **Ease of Use** | Medium | Easy | Medium | Easy |
| **Customization** | High | Medium | Low | High |
| **Interactivity** | No | No | No | Yes |
| **Best For** | Basic Charts | Statistical Plots | Layered Approach | Web-Based Interactive Charts |

## **💡 Conclusion**

1️⃣ **Matplotlib** – Great for basic **line plots, histograms, and bar charts**.  
2️⃣ **Seaborn** – Simplifies statistical visualization (**box plots, scatter plots, etc.**).  
3️⃣ **ggplot** – Inspired by **ggplot2 in R**, uses a **layered** approach.  
4️⃣ **Plotly** – Ideal for **interactive, web-based visualizations**.

### **📌 MCQs on Matplotlib**

#### **(1) What is Matplotlib used for in Python?**

🔘 A) Web development  
🔘 B) Data visualization  
🔘 C) Machine learning  
🔘 D) File handling  
✅ **Answer:** B) Data visualization

#### **(2) What is the correct way to import Matplotlib?**

🔘 A) import matplotlib.plot as plt  
🔘 B) import matplotlib.pyplot as plt  
🔘 C) import matplotlib as pyplt  
🔘 D) from matplotlib import graph  
✅ **Answer:** B) import matplotlib.pyplot as plt

#### **(3) What does the following code do?**

python

CopyEdit

import matplotlib.pyplot as plt

plt.plot([1, 2, 3], [4, 5, 6])

plt.show()

🔘 A) Displays a scatter plot  
🔘 B) Displays a line graph  
🔘 C) Displays a bar chart  
🔘 D) Produces an error  
✅ **Answer:** B) Displays a line graph

#### **(4) What parameter is used to change the line style in plt.plot()?**

🔘 A) linetype  
🔘 B) style  
🔘 C) linestyle  
🔘 D) line  
✅ **Answer:** C) linestyle

#### **(5) Which function is used to create a bar chart in Matplotlib?**

🔘 A) plt.bar()  
🔘 B) plt.hist()  
🔘 C) plt.plot()  
🔘 D) plt.pie()  
✅ **Answer:** A) plt.bar()

#### **(6) What does the following code do?**

python

CopyEdit

plt.hist([1, 2, 3, 1, 2, 3, 1, 1], bins=3)

plt.show()

🔘 A) Creates a bar chart  
🔘 B) Creates a histogram  
🔘 C) Creates a scatter plot  
🔘 D) Produces an error  
✅ **Answer:** B) Creates a histogram

#### **(7) How do you add a title to a Matplotlib plot?**

🔘 A) plt.heading("My Title")  
🔘 B) plt.label("My Title")  
🔘 C) plt.title("My Title")  
🔘 D) plt.name("My Title")  
✅ **Answer:** C) plt.title("My Title")

### **📌 MCQs on Seaborn**

#### **(8) What is Seaborn mainly used for?**

🔘 A) Machine Learning  
🔘 B) Deep Learning  
🔘 C) Statistical Data Visualization  
🔘 D) Web Development  
✅ **Answer:** C) Statistical Data Visualization

#### **(9) How do you import Seaborn?**

🔘 A) import seaborn as sb  
🔘 B) import seaborn as sns  
🔘 C) import seaborn as plt  
🔘 D) import seaborn as snsplt  
✅ **Answer:** B) import seaborn as sns

#### **(10) What type of plot does sns.boxplot() generate?**

🔘 A) Line Plot  
🔘 B) Scatter Plot  
🔘 C) Box Plot  
🔘 D) Bar Chart  
✅ **Answer:** C) Box Plot

#### **(11) What does the following code do?**

python

CopyEdit

import seaborn as sns

import matplotlib.pyplot as plt

sns.set(style="darkgrid")

sns.scatterplot(x=[1, 2, 3], y=[4, 5, 6])

plt.show()

🔘 A) Creates a line graph  
🔘 B) Creates a scatter plot  
🔘 C) Creates a bar chart  
🔘 D) Produces an error  
✅ **Answer:** B) Creates a scatter plot

#### **(12) What function in Seaborn is used for heatmaps?**

🔘 A) sns.heatmap()  
🔘 B) sns.map()  
🔘 C) sns.barplot()  
🔘 D) sns.scatterplot()  
✅ **Answer:** A) sns.heatmap()

### **📌 MCQs on ggplot**

#### **(13) What library provides the ggplot functionality in Python?**

🔘 A) matplotlib  
🔘 B) seaborn  
🔘 C) plotnine  
🔘 D) plotly  
✅ **Answer:** C) plotnine

#### **(14) What does the following ggplot code do?**

python

CopyEdit

from plotnine import ggplot, aes, geom\_line

import pandas as pd

df = pd.DataFrame({'x': [1, 2, 3], 'y': [2, 4, 6]})

ggplot(df, aes(x='x', y='y')) + geom\_line()

🔘 A) Creates a line plot  
🔘 B) Creates a scatter plot  
🔘 C) Creates a histogram  
🔘 D) Produces an error  
✅ **Answer:** A) Creates a line plot

#### **(15) What function in ggplot is used to define the mapping of aesthetics?**

🔘 A) geom\_aesthetics()  
🔘 B) aes()  
🔘 C) geom\_map()  
🔘 D) map\_aes()  
✅ **Answer:** B) aes()

### **📌 MCQs on Plotly**

#### **(16) What is Plotly used for?**

🔘 A) Static plots  
🔘 B) Interactive web-based plots  
🔘 C) Image processing  
🔘 D) Data cleaning  
✅ **Answer:** B) Interactive web-based plots

#### **(17) What is the correct way to import Plotly Express?**

🔘 A) import plotly.express as px  
🔘 B) import plotly.plot as plt  
🔘 C) import plotly.graph\_objects as go  
🔘 D) import plotly.visual as pv  
✅ **Answer:** A) import plotly.express as px

#### **(18) What does fig.show() do in Plotly?**

🔘 A) Saves the plot  
🔘 B) Displays the plot  
🔘 C) Closes the plot  
🔘 D) Exports the plot  
✅ **Answer:** B) Displays the plot

#### **(19) What is the purpose of px.scatter() in Plotly?**

🔘 A) Creates a bar chart  
🔘 B) Creates a line graph  
🔘 C) Creates a scatter plot  
🔘 D) Creates a histogram  
✅ **Answer:** C) Creates a scatter plot

#### **(20) Which library provides 3D plotting in Plotly?**

🔘 A) plotly.graph\_objects  
🔘 B) plotly.express  
🔘 C) plotly.3dplot  
🔘 D) plotly.3dgraphs  
✅ **Answer:** A) plotly.graph\_objects

## **Session22 & 23: Web-Based Frameworks: Flask and Django**

Python provides two powerful web frameworks for building web applications: **Flask** and **Django**. Both frameworks help developers create dynamic websites, APIs, and web applications. However, they have different approaches and use cases.

## **1️⃣ Flask: The Lightweight Framework**

Flask is a **micro-framework** for Python that is simple, flexible, and lightweight. It is used to build small to medium-sized web applications.

### **🔹 Features of Flask**

* **Minimalist and lightweight**
* **Built-in development server & debugger**
* **Jinja2 templating engine**
* **Support for RESTful APIs**
* **Uses Werkzeug for WSGI**
* **Modular (extensions available for ORM, authentication, etc.)**

### **🔹 Installing Flask**

To install Flask, run:

bash

CopyEdit

pip install flask

### **🔹 Creating a Simple Flask App**

python

CopyEdit

from flask import Flask

app = Flask(\_\_name\_\_) # Create Flask application

@app.route('/') # Define a route

def home():

return "Hello, Flask!"

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True) # Run the application

✅ This code:

* Creates a Flask app
* Defines a route (/) that returns "Hello, Flask!"
* Runs the application with debugging enabled

### **🔹 Routing in Flask**

Routing allows users to navigate different URLs in a web app.

python

CopyEdit

@app.route('/about')

def about():

return "Welcome to the About Page!"

Now, accessing /about in the browser will show "Welcome to the About Page!".

### **🔹 Flask Templates**

Flask uses Jinja2 as its templating engine.

**Example: Using an HTML template**

* **Create a file:** templates/index.html

html

CopyEdit

<!DOCTYPE html>

<html>

<head><title>Flask Template</title></head>

<body>

<h1>Hello, {{ name }}!</h1>

</body>

</html>

* **Render the template in Flask**

python

CopyEdit

from flask import render\_template

@app.route('/user/<name>')

def user(name):

return render\_template('index.html', name=name)

✅ This will display **Hello, UserName!** based on the URL input.

### **🔹 Handling Forms in Flask**

python

CopyEdit

from flask import request

@app.route('/login', methods=['GET', 'POST'])

def login():

if request.method == 'POST':

username = request.form['username']

return f'Welcome, {username}!'

return '''

<form method="post">

<input type="text" name="username" placeholder="Enter your name">

<input type="submit">

</form>

'''

✅ This allows users to submit a form via POST request.

## **2️⃣ Django: The Full-Stack Framework**

Django is a **high-level, full-stack web framework** that provides built-in features for rapid development. It follows the **Model-View-Template (MVT)** architecture.

### **🔹 Features of Django**

* **Full-stack framework** (Authentication, ORM, Admin panel, etc.)
* **Highly secure** (Prevents SQL Injection, CSRF, XSS)
* **Scalable and robust**
* **Built-in admin panel**
* **Built-in database support (ORM)**
* **Supports both static and dynamic content**

### **🔹 Installing Django**

To install Django, run:

bash

CopyEdit

pip install django

### **🔹 Creating a Django Project**

bash

CopyEdit

django-admin startproject myproject

cd myproject

python manage.py runserver

This sets up a basic Django project.

✅ Access http://127.0.0.1:8000/ in a browser to see the default Django welcome page.

### **🔹 Django App Structure**

A Django project consists of multiple **apps**. To create an app:

bash

CopyEdit

python manage.py startapp myapp

### **🔹 Creating a View in Django**

* **Edit views.py in myapp**

python

CopyEdit

from django.http import HttpResponse

def home(request):

return HttpResponse("Hello, Django!")

* **Configure the URL** Edit urls.py in myapp:

python

CopyEdit

from django.urls import path

from . import views

urlpatterns = [

path('', views.home),

]

✅ Now, visiting the home page (/) will display "Hello, Django!".

### **🔹 Django Models (Database)**

Django uses an **ORM (Object-Relational Mapping)** system.

* **Define a model in models.py**

python

CopyEdit

from django.db import models

class User(models.Model):

name = models.CharField(max\_length=100)

email = models.EmailField()

* **Migrate the model to the database**

bash

CopyEdit

python manage.py makemigrations

python manage.py migrate

✅ This creates a table for User in the database.

### **🔹 Django Admin Panel**

Django provides an admin panel for managing data.

* **Enable the admin panel in admin.py**

python

CopyEdit

from django.contrib import admin

from .models import User

admin.site.register(User)

* **Create a superuser**

bash

CopyEdit

python manage.py createsuperuser

✅ Now, log in to http://127.0.0.1:8000/admin/ to manage data.

## **🔥 Flask vs. Django: A Comparison**

| **Feature** | **Flask** | **Django** |
| --- | --- | --- |
| **Type** | Micro-framework | Full-stack framework |
| **Flexibility** | Highly flexible | Structured and opinionated |
| **Built-in Features** | Minimal (Extensions needed) | Many built-in features (Admin, Auth, ORM) |
| **Performance** | Faster for small projects | Optimized for large apps |
| **Best For** | Small to medium apps, APIs | Large-scale applications |

✅ **Use Flask** when you need flexibility and minimal setup.  
✅ **Use Django** when you need a full-featured, scalable framework.

## **🎯 Conclusion**

* **Flask** is a simple and flexible framework, best for small applications.
* **Django** is a powerful full-stack framework with built-in tools, ideal for large applications.
* Both frameworks have their own advantages, and the choice depends on the project requirements.

### **1-10: Basic Concepts (Easy)**

**1. What is Flask in Python?**A) A full-stack framework  
B) A micro-framework  
C) A database library  
D) A front-end framework

✅ **Answer:** B) A micro-framework

**2. How do you install Flask?**A) pip install flask  
B) install flask  
C) flask install  
D) python -m flask install

✅ **Answer:** A) pip install flask

**3. What is the correct way to create a basic Flask app?**

python

CopyEdit

A) from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def home():

return "Hello, Flask!"

app.run()

python

CopyEdit

B) from flask import WebApp

app = WebApp(\_\_name\_\_)

@app.page('/')

def home():

return "Hello, Flask!"

app.start()

python

CopyEdit

C) import flask

app = flask()

@app.route('/')

def home():

return "Hello, Flask!"

app.run()

✅ **Answer:** A) The first code block

**4. What is Django?**A) A Python web framework  
B) A JavaScript framework  
C) A database management system  
D) A command-line tool

✅ **Answer:** A) A Python web framework

**5. Which architecture does Django follow?**A) Model-View-Template (MVT)  
B) Model-View-Controller (MVC)  
C) Microservices  
D) Procedural

✅ **Answer:** A) Model-View-Template (MVT)

**6. How do you start a new Django project?**A) django createproject myproject  
B) django start myproject  
C) django-admin startproject myproject  
D) django create myproject

✅ **Answer:** C) django-admin startproject myproject

**7. What does the runserver command do in Django?**A) Starts the development server  
B) Deploys the project to the cloud  
C) Runs database migrations  
D) Installs project dependencies

✅ **Answer:** A) Starts the development server

**8. What is the default port for Django's development server?**A) 8080  
B) 5000  
C) 8000  
D) 3000

✅ **Answer:** C) 8000

**9. In Flask, how do you define a dynamic route that accepts a string parameter?**A) @app.route('/user<int:name>')  
B) @app.route('/user<string:name>')  
C) @app.route('/user{name}')  
D) @app.route('/user[]')

✅ **Answer:** B) @app.route('/user<string:name>')

**10. What is Jinja2 in Flask?**A) A database framework  
B) A CSS framework  
C) A templating engine  
D) A logging system

✅ **Answer:** C) A templating engine

### **11-20: Intermediate Questions**

**11. How do you define a database model in Django?**

python

CopyEdit

A) class User(object):

name = str

email = str

python

CopyEdit

B) class User(models.Model):

name = models.CharField(max\_length=100)

email = models.EmailField()

python

CopyEdit

C) class User():

name = CharField(100)

email = Email()

✅ **Answer:** B) The second code block

**12. What command is used to apply Django database migrations?**A) python manage.py makemigrations && python manage.py migrate  
B) python manage.py runserver  
C) python migrate  
D) django-db migrate

✅ **Answer:** A) python manage.py makemigrations && python manage.py migrate

**13. How do you register a Django model in the admin panel?**A) admin.site.register(MyModel)  
B) admin.create(MyModel)  
C) admin.site.add(MyModel)  
D) admin.MyModel.register()

✅ **Answer:** A) admin.site.register(MyModel)

**14. Which of the following is a valid way to handle errors in Flask?**A) @app.errorhandler(404)  
B) @app.catch\_error(404)  
C) @app.on\_error(404)  
D) @app.handle\_error(404)

✅ **Answer:** A) @app.errorhandler(404)

**15. How do you retrieve all objects from a Django model?**A) User.objects.all()  
B) User.get\_all()  
C) User.retrieve()  
D) User.select\_all()

✅ **Answer:** A) User.objects.all()

**16. How do you pass a context variable to a Django template?**

python

CopyEdit

A) return render(request, 'index.html', context={'name': 'Alice'})

python

CopyEdit

B) return render\_template('index.html', name='Alice')

python

CopyEdit

C) return view('index.html', name='Alice')

✅ **Answer:** A) The first code block

**17. Which of the following is used for URL mapping in Django?**A) urls.py  
B) routes.py  
C) views.py  
D) django\_urls.py

✅ **Answer:** A) urls.py

**18. How do you define a decorator in Python?**A) @decorator  
B) # decorator  
C) def decorator:  
D) \*decorator\*

✅ **Answer:** A) @decorator

**19. What does request.form do in Flask?**A) Retrieves data from a form submission  
B) Displays a form  
C) Deletes a form  
D) Validates form data

✅ **Answer:** A) Retrieves data from a form submission

**20. How do you define a custom 404 error page in Flask?**

python

CopyEdit

A) @app.errorhandler(404)

def not\_found(error):

return "Page Not Found", 404

python

CopyEdit

B) @app.route('/404')

def not\_found():

return "Page Not Found", 404

python

CopyEdit

C) @app.handle(404)

def not\_found():

return "Page Not Found", 404

✅ **Answer:** A) The first code block

### **21-30: Advanced Questions**

**21. What is the difference between @staticmethod and @classmethod in Django models?**A) @classmethod takes a class parameter, @staticmethod doesn’t  
B) Both are identical  
C) @staticmethod modifies the class, @classmethod doesn’t  
D) @classmethod is only for Flask

✅ **Answer:** A) @classmethod takes a class parameter, @staticmethod doesn’t

### **21-30: Advanced Questions (continued)**

**21. Which method would you use in Flask to define a route that accepts both GET and POST requests?**A) @app.route('/path', methods=['GET'])  
B) @app.route('/path', methods=['POST'])  
C) @app.route('/path', methods=['GET', 'POST'])  
D) @app.route('/path', methods=['PUT'])

✅ **Answer:** C) @app.route('/path', methods=['GET', 'POST'])

**22. What is the purpose of flask\_sqlalchemy in Flask?**A) It provides database migration support.  
B) It allows Flask to interact with SQL databases using ORM.  
C) It handles HTTP requests in Flask.  
D) It provides authentication features for Flask apps.

✅ **Answer:** B) It allows Flask to interact with SQL databases using ORM.

**23. What does the @app.route('/') decorator do in Flask?**A) Maps the URL '/' to a function, triggering that function when the URL is accessed.  
B) Routes the request to a database table.  
C) Creates a new view for the specified URL.  
D) Redirects users to the home page.

✅ **Answer:** A) Maps the URL '/' to a function, triggering that function when the URL is accessed.

**24. In Django, which of the following methods is used to get a single object from a model based on a query?**A) Model.get\_object()  
B) Model.objects.get()  
C) Model.objects.first()  
D) Model.objects.all()

✅ **Answer:** B) Model.objects.get()

**25. How would you handle a 404 Not Found error in Django?**A) Override the get() method in the view.  
B) Use HttpResponseNotFound() to return the error.  
C) Create a custom 404 view function in views.py.  
D) Both B and C are correct.

✅ **Answer:** D) Both B and C are correct.

**26. In Flask, how would you access form data from a POST request?**

python

CopyEdit

A) form\_data = request.form['fieldname']

B) form\_data = request.get('fieldname')

C) form\_data = request.query['fieldname']

D) form\_data = request.body['fieldname']

✅ **Answer:** A) form\_data = request.form['fieldname']

**27. Which of the following is NOT a valid way to handle Django form validation?**A) Override the clean() method in the form class.  
B) Use form.is\_valid() to check if the form is valid.  
C) Define custom validators using clean\_fieldname().  
D) Manually validate form fields in views.py.

✅ **Answer:** D) Manually validate form fields in views.py.

**28. What is the purpose of urlpatterns in Django?**A) It defines the URL patterns for routing requests to views.  
B) It stores session data for the application.  
C) It maps models to database tables.  
D) It configures the app’s settings.

✅ **Answer:** A) It defines the URL patterns for routing requests to views.

**29. How would you retrieve a specific key-value pair from a Flask request.args dictionary?**A) value = request.args['key']  
B) value = request.args.get('key')  
C) value = request.args.fetch('key')  
D) value = request.args.get\_value('key')

✅ **Answer:** B) value = request.args.get('key')

**30. In Flask, how can you pass data to a template?**A) By using the render\_template() function and passing keyword arguments.  
B) By setting the data in a global variable.  
C) By modifying the request object directly.  
D) By appending the data to the URL query string.

✅ **Answer:** A) By using the render\_template() function and passing keyword arguments.

### **31-40: Advanced Questions (Continued)**

**31. In Django, which of the following is true about class-based views?**A) They allow you to handle HTTP methods (GET, POST) separately in methods.  
B) They must be instantiated every time they are used.  
C) They can only handle GET requests.  
D) They don’t support URL routing.

✅ **Answer:** A) They allow you to handle HTTP methods (GET, POST) separately in methods.

**32. Which of the following Django commands creates the initial database schema from models?**A) python manage.py syncdb  
B) python manage.py migrate  
C) python manage.py create\_db  
D) python manage.py setupdb

✅ **Answer:** B) python manage.py migrate

**33. How would you implement a simple authentication system in Flask?**A) Use Flask-SQLAlchemy and create a user model manually.  
B) Use Flask-Login for session-based user authentication.  
C) Use Flask-Admin to automatically generate user registration forms.  
D) Use Flask-Security for complex user management features.

✅ **Answer:** B) Use Flask-Login for session-based user authentication.

**34. In Django, what is the purpose of the Form class?**A) It allows you to generate HTML form elements programmatically.  
B) It provides an interface to manipulate URLs.  
C) It controls the flow of application logic.  
D) It enables database interaction via ORM.

✅ **Answer:** A) It allows you to generate HTML form elements programmatically.

**35. How do you add static files like CSS or JavaScript in Django?**A) Place them in the static/ directory and link them in templates.  
B) Place them in the media/ directory.  
C) Use django-static for automatic static file generation.  
D) Add them directly to the views.

✅ **Answer:** A) Place them in the static/ directory and link them in templates.

**36. What does Flask-SQLAlchemy provide for working with databases in Flask?**A) It is a tool for routing HTTP requests.  
B) It allows you to interact with databases using Python classes.  
C) It provides a pre-built template system.  
D) It manages authentication and user permissions.

✅ **Answer:** B) It allows you to interact with databases using Python classes.

**37. What does the django.shortcuts.render() function do?**A) It renders a JSON response.  
B) It loads a template and passes a context dictionary to it.  
C) It redirects the user to a new URL.  
D) It handles static file loading.

✅ **Answer:** B) It loads a template and passes a context dictionary to it.

**38. Which Flask extension would you use for handling database models?**A) Flask-SQLAlchemy  
B) Flask-Login  
C) Flask-WTF  
D) Flask-Mail

✅ **Answer:** A) Flask-SQLAlchemy

**39. What is the correct way to create a custom 500 server error page in Flask?**

python

CopyEdit

A) @app.errorhandler(500)

def internal\_error(error):

return "Internal Server Error", 500

python

CopyEdit

B) @app.route('/error')

def error\_page():

return "500 Internal Error"

python

CopyEdit

C) @app.server\_error(500)

def server\_error\_page():

return "Error 500"

✅ **Answer:** A) The first code block

**40. What is the primary benefit of Django’s admin interface?**A) It automatically manages URL routing.  
B) It provides an easy-to-use interface for managing database records.  
C) It helps developers manage user authentication.  
D) It generates dynamic HTML pages automatically.

✅ **Answer:** B) It provides an easy-to-use interface for managing database records.

### **Session 25: Request and URL-Lib & Working with Scrapy**

#### **1. Request and URL-Lib**

requests and urllib are two popular libraries used to interact with web resources, like APIs or web pages, in Python.

##### **1.1. requests Library**

The requests library simplifies making HTTP requests (like GET, POST, etc.) to interact with web servers. It's one of the most commonly used libraries for web scraping and interacting with REST APIs.

**Basic Usage:**

python

CopyEdit

import requests

response = requests.get('https://api.github.com')

print(response.status\_code) # Status code of the response (e.g., 200 for success)

print(response.text) # Response body (HTML, JSON, etc.)

**Methods:**

* requests.get(): Sends a GET request to the specified URL.
* requests.post(): Sends a POST request (useful for submitting data).
* requests.put(): Sends a PUT request (for updating data).
* requests.delete(): Sends a DELETE request (for removing data).

**Handling Responses:**

* .status\_code: HTTP response code.
* .text: The content of the response.
* .json(): If the response is in JSON format, this method will parse the JSON data into a Python dictionary.

**Handling Parameters:**

python

CopyEdit

params = {'q': 'Python'}

response = requests.get('https://www.google.com/search', params=params)

print(response.url) # Will print the URL with query parameters

##### **1.2. urllib Library**

urllib is a built-in Python library for working with URLs. It's a more manual approach compared to requests, but it's included in the Python standard library.

**Basic Usage:**

python

CopyEdit

import urllib.request

url = 'http://example.com'

response = urllib.request.urlopen(url)

print(response.read()) # Read the content of the response

**Modules in urllib:**

* urllib.request: For opening and reading URLs.
* urllib.parse: For parsing URLs and managing query parameters.
* urllib.error: For handling URL-related errors.
* urllib.robotparser: For reading robots.txt files to check if scraping is allowed.

**Example of parsing URLs:**

python

CopyEdit

from urllib.parse import urlparse

url = 'https://www.example.com/path?query=python#fragment'

parsed\_url = urlparse(url)

print(parsed\_url.scheme) # https

print(parsed\_url.netloc) # www.example.com

print(parsed\_url.path) # /path

#### **2. Working with Scrapy**

**Scrapy** is a powerful web scraping framework in Python. It provides a set of tools to scrape websites and extract structured data from web pages. Unlike simple libraries like requests or urllib, Scrapy is designed for more complex and scalable web scraping tasks.

##### **2.1. Key Features of Scrapy:**

* **Spider:** A spider is a class that you define and that Scrapy uses to scrape a website and extract the required data.
* **Selectors:** Scrapy uses CSS and XPath selectors to extract data from the HTML content of web pages.
* **Pipeline:** After the spider extracts the data, it passes it through a pipeline to process or store the data (e.g., save to a database or CSV file).
* **Crawling:** Scrapy can follow links and crawl multiple pages on a website automatically.

##### **2.2. Setting Up Scrapy**

**Install Scrapy:**bash  
CopyEdit  
pip install scrapy

**Create a Scrapy Project:**bash  
CopyEdit  
scrapy startproject myproject

**Creating a Spider:** A spider is a class that contains the logic for scraping. Here's an example spider that scrapes quotes from the "quotes.toscrape.com" website.  
**Create a Spider (quotes\_spider.py):**python  
CopyEdit  
import scrapy

class QuotesSpider(scrapy.Spider):

name = 'quotes'

start\_urls = ['http://quotes.toscrape.com']

def parse(self, response):

for quote in response.css('div.quote'):

yield {

'text': quote.css('span.text::text').get(),

'author': quote.css('span small::text').get(),

'tags': quote.css('div.tags a.tag::text').getall(),

}

next\_page = response.css('li.next a::attr(href)').get()

if next\_page is not None:

yield response.follow(next\_page, self.parse)

**Running the Spider:** After defining the spider, you can run it from the command line:  
bash  
CopyEdit  
scrapy crawl quotes

##### **2.3. Scrapy Selectors**

Scrapy provides XPath and CSS selectors to extract data from the HTML response.

**XPath Example:**python  
CopyEdit  
response.xpath('//h1/text()').get() # Extracts text inside the <h1> tag

**CSS Selector Example:**python  
CopyEdit  
response.css('h1::text').get() # Extracts text inside the <h1> tag

##### **2.4. Scrapy Pipelines**

Pipelines in Scrapy are used to process scraped data. After data is extracted, it can be cleaned, validated, or stored in a database or file format.

**Example Pipeline (pipelines.py):**python  
CopyEdit  
class MyPipeline:

def process\_item(self, item, spider):

item['author'] = item['author'].upper() # Convert author to uppercase

return item

##### **2.5. Storing Data**

Scrapy can export scraped data in different formats like JSON, CSV, or XML.

**Run Spider and Save Output to JSON:**bash  
CopyEdit  
scrapy crawl quotes -o quotes.json

##### **2.6. Scrapy Settings**

In Scrapy, the settings define how the spider works and can be adjusted in settings.py to customize behaviors like download delay, user-agent string, and concurrent requests.

For example, you can set a custom user-agent:

python

CopyEdit

USER\_AGENT = 'MyCrawler (+http://www.mydomain.com)'

### **Summary of Key Points:**

* **Requests and URL-Lib:** Used for making HTTP requests and handling URLs. requests is simpler and more powerful for HTTP requests, while urllib is built into Python and provides lower-level functionality.
* **Scrapy:** A web scraping framework designed for scalable and efficient scraping. It provides a robust set of tools for managing spiders, handling selectors, processing data through pipelines, and exporting scraped content to files.

Scrapy is a great choice for large-scale web scraping tasks that need to follow links and extract structured data automatically. On the other hand, requests and urllib are better for smaller, one-off web scraping tasks or API requests.

### **MCQs:**

1. **What is the purpose of the requests.get() method in Python?**
   * A) To send a GET request to a specified URL
   * B) To send a POST request to a specified URL
   * C) To parse URLs
   * D) To handle exceptions

**Answer:** A

1. **Which method in the requests library is used to send data to a server?**
   * A) requests.get()
   * B) requests.post()
   * C) requests.put()
   * D) requests.delete()

**Answer:** B

1. **Which of the following is used to parse URLs in Python?**
   * A) urllib.request
   * B) urllib.parse
   * C) requests
   * D) urlparse

**Answer:** B

1. **What does response.json() do in the requests library?**
   * A) Parses the response as HTML
   * B) Converts the response to a Python dictionary
   * C) Converts the response to a list
   * D) Returns the response status code

**Answer:** B

1. **Which of the following methods is used to handle URL-related errors in urllib?**
   * A) urllib.request
   * B) urllib.error
   * C) urllib.parse
   * D) urllib.robotparser

**Answer:** B

1. **What is the default user agent used by requests?**
   * A) Mozilla/5.0
   * B) Python/3.8
   * C) requests/2.25.0
   * D) Scrapy/2.3

**Answer:** C

1. **Which of the following is NOT a method available in the requests library?**
   * A) requests.head()
   * B) requests.put()
   * C) requests.delete()
   * D) requests.exit()

**Answer:** D

1. **Which of the following functions is used to open a URL in urllib?**
   * A) urllib.request.open()
   * B) urllib.request.urlopen()
   * C) urllib.request.get()
   * D) urllib.request.request()

**Answer:** B

1. **What is a primary benefit of using Scrapy over requests for web scraping?**
   * A) Scrapy is simpler to use
   * B) Scrapy provides built-in selectors for parsing
   * C) Scrapy doesn't need Python
   * D) Scrapy doesn't require HTTP requests

**Answer:** B

1. **What is a Spider in Scrapy?**
   * A) A database connector
   * B) A class that defines how to scrape a website
   * C) A web server
   * D) A module for handling user agents

**Answer:** B

1. **In Scrapy, which method is used to extract data from a webpage?**
   * A) .parse()
   * B) .extract()
   * C) .xpath()
   * D) .response()

**Answer:** A

1. **Which Scrapy method is used to handle pagination or follow links to other pages?**
   * A) response.follow()
   * B) response.next()
   * C) response.crawl()
   * D) response.goto()

**Answer:** A

1. **Which Scrapy selector is used to extract data using CSS selectors?**
   * A) response.select()
   * B) response.css()
   * C) response.xpath()
   * D) response.get()

**Answer:** B

1. **In Scrapy, what is the role of the item pipeline?**
   * A) To crawl through different websites
   * B) To validate and process scraped data
   * C) To handle requests and responses
   * D) To execute scraping asynchronously
2. **Answer:** B
3. **Which of the following is used to save data in Scrapy to a file?**
   * A) scrapy.save()
   * B) scrapy.export()
   * C) scrapy crawl -o output.json
   * D) scrapy.write()
4. **Answer:** C
5. **What is the purpose of the USER\_AGENT setting in Scrapy?**
   * A) To define how the spider handles exceptions
   * B) To specify the user-agent string in requests to the server
   * C) To define the output file format
   * D) To limit the number of requests
6. **Answer:** B
7. **Which command in Scrapy starts a spider?**
   * A) scrapy startproject
   * B) scrapy crawl spidername
   * C) scrapy runspider
   * D) scrapy startspider
8. **Answer:** B
9. **Which of the following is a key benefit of using Scrapy's scrapy crawl command?**
   * A) It runs the spider and saves the data automatically
   * B) It downloads pages in parallel and processes them asynchronously
   * C) It prevents spider crashes by handling errors
   * D) It automatically parses data into HTML
10. **Answer:** B
11. **Which method in Scrapy is used to follow links in a page?**
    * A) scrapy.follow()
    * B) scrapy.link()
    * C) response.follow()
    * D) response.link()
12. **Answer:** C
13. **Which module in urllib is used to parse URL components?**
    * A) urllib.request
    * B) urllib.parse
    * C) urllib.error
    * D) urllib.robotparser
14. **Answer:** B

**In the following Scrapy code, which statement is correct?**python  
CopyEdit  
class QuotesSpider(scrapy.Spider):

name = 'quotes'

start\_urls = ['http://quotes.toscrape.com']

def parse(self, response):

for quote in response.css('div.quote'):

yield {'text': quote.css('span.text::text').get()}

* + A) It sends a GET request to http://quotes.toscrape.com
  + B) It sends a POST request to http://quotes.toscrape.com
  + C) It scrapes quotes only from the first page
  + D) It writes the data to a database

1. **Answer:** A
2. **In the Scrapy pipeline, what is the main purpose of process\_item()?**
   * A) To start the spider
   * B) To handle errors during scraping
   * C) To process and clean the extracted data
   * D) To extract data from a webpage
3. **Answer:** C
4. **What does response.css('span.text::text').get() return in Scrapy?**
   * A) The entire HTML content of the span.text element
   * B) The first matching element's text inside span.text
   * C) All elements with class text
   * D) A list of all tags named span.text
5. **Answer:** B
6. **Which command in Scrapy is used to create a new project?**
   * A) scrapy startproject myproject
   * B) scrapy newproject myproject
   * C) scrapy createproject myproject
   * D) scrapy project myproject
7. **Answer:** A
8. **Which of the following is used to handle HTTP errors in Scrapy?**
   * A) handle\_error()
   * B) errback()
   * C) parse\_error()
   * D) response\_error()
9. **Answer:** B
10. **Which of the following methods in Scrapy is used to extract a specific part of a page's HTML content using XPath?**
    * A) response.xpath()
    * B) response.select()
    * C) response.find()
    * D) response.xpath\_select()
11. **Answer:** A
12. **What is the purpose of using requests.post() in Python?**
    * A) To make a POST request and send data to a server
    * B) To fetch data from a server
    * C) To parse the response data
    * D) To handle URL errors
13. **Answer:** A
14. **In which file does Scrapy store its settings by default?**
    * A) settings.py
    * B) config.py
    * C) scrapy.py
    * D) scrapy\_settings.py
15. **Answer:** A
16. **What does response.text return in requests?**
    * A) HTML content as a string
    * B) JSON content as a string
    * C) Response status code
    * D) Response headers
17. **Answer:** A
18. **Which of the following functions is used to extract the domain from a URL using urllib?**
    * A) urllib.parse.urlparse()
    * B) urllib.request.urlopen()
    * C) urllib.parse.urlencode()
    * D) urllib.parse.urljoin()
19. **Answer:** A