1. Task: Creating and Modifying Variables

- **Objective:** Learn how to create and modify variables in Python.
- **Steps:**
    1. Create variables of different types: integer, float, string, and boolean.
    2. Modify the value of each variable.
    3. Print the variables to see the changes.
- **Question:** What happens if you try to change the value of a string variable to an integer? Does Python allow this type of modification?

---

2. Task: Understanding Data Types

- **Objective:** Understand how to work with Python's basic data types.
- **Steps:**
    1. Create variables of type `int`, `float`, `str`, and `bool`.
    2. Perform operations (e.g., addition, multiplication) using different data types.
    3. Print the results.
- **Question:** What is the result when you add a string to an integer? How does Python handle type compatibility in this case?

---

3. Task: Using Conditional Statements (if-else)

- **Objective:** Practice using `if-else` conditional statements.
- **Steps:**
    1. Write a program that checks if a number is positive, negative, or zero.
    2. Use the `if`, `elif`, and `else` statements.
- **Question:** How would the program behave if you enter a non-numeric input?

---

4. Task: Writing a Program with Multiple Conditions

- **Objective:** Work with multiple conditions using `elif`.
- **Steps:**
    1. Write a program that checks the grade of a student based on their score.
    2. Use `if`, `elif`, and `else` to assign the grade.
- **Question:** How can you modify the program to handle invalid scores (e.g., scores greater than 100)?

---

5. Task: Using While Loop

- **Objective:** Learn how to implement a `while` loop in Python.
- **Steps:**
    1. Create a `while` loop that prints numbers from 1 to 10.
    2. Modify the program to print only even numbers.
- **Question:** What will happen if the condition in the `while` loop is never updated?

---

6. Task: Using For Loop

- **Objective:** Understand the use of a `for` loop for iterating through a sequence.
- **Steps:**
    1. Use a `for` loop to print each item in a list.
    2. Modify the program to print the length of each string in the list.
- **Question:** How would you modify the loop to iterate through a dictionary?

---

7. Task: Break and Continue Statements

- **Objective:** Learn how to control loop execution using `break` and `continue`.
- **Steps:**
    1. Create a loop that iterates over numbers 1 through 10.
    2. Use `continue` to skip odd numbers.
    3. Use `break` to exit the loop when the number reaches 7.
- **Question:** How does the behavior of the loop change when `break` or `continue` is used?

---

8. Task: Type Casting (Implicit and Explicit)

- **Objective:** Understand type casting in Python.
- **Steps:**
    1. Create variables of type `int` and `float`.
    2. Add an integer and a float, and print the result (implicit casting).
    3. Explicitly cast a float to an integer and print the result.
- **Question:** What happens to the decimal part when you cast a float to an integer?

---

9. Task: Handling Exceptions (Try-Except)

- **Objective:** Learn how to handle errors using `try` and `except`.
- **Steps:**
    1. Write a program that asks the user for a number and divides 10 by that number.
    2. Use a `try-except` block to handle the `ZeroDivisionError`.
- **Question:** What will happen if the user enters a non-numeric value?

---

10. Task: Writing a Function with Arguments

- **Objective:** Learn how to define functions and use parameters.
- **Steps:**
    1. Write a function that takes two numbers as arguments and returns their sum.
    2. Call the function with different arguments.
- **Question:** How would you modify the function to return the product of the two numbers instead of the sum?

---

11. Task: Using Built-in Functions

- **Objective:** Practice using Python's built-in functions.
- **Steps:**
    1. Use the `len()` function to find the length of a string.
    2. Use `max()` and `min()` to find the largest and smallest values in a list.
    3. Use `sum()` to find the sum of all elements in the list.
- **Question:** How would you find the median of a list using built-in functions?

---

12. Task: Working with Lists (Add, Modify, Remove)

- **Objective:** Understand how to manipulate lists in Python.
- **Steps:**
    1. Create a list of five numbers.
    2. Add a new number to the list.
    3. Remove the first number from the list.
    4. Modify the third element in the list.
- **Question:** What happens if you try to access an index that doesn't exist in the list?

---

13. Task: Tuples and Immutability

- **Objective:** Learn how to work with tuples and their immutability.
- **Steps:**
    1. Create a tuple with different data types (string, integer, float).
    2. Try modifying an element in the tuple and see what happens.
- **Question:** Why can’t you change the values in a tuple?

---

14. Task: Working with Dictionaries (Key-Value Pairs)

- **Objective:** Learn how to work with dictionaries.
- **Steps:**
    1. Create a dictionary with name, age, and location as keys.
    2. Add a new key-value pair for "job".
    3. Update the "age" value.
    4. Remove the "location" key.
- **Question:** What happens if you try to access a key that doesn't exist in the dictionary?

---

15. Task: Sets and Set Operations

- **Objective:** Understand how to use sets in Python.
- **Steps:**
    1. Create two sets: one containing odd numbers and the other containing even numbers.
    2. Use the union and intersection operations to combine the sets.
    3. Check if an element exists in a set.
- **Question:** What happens when you try to add a duplicate element to a set?

---

16. Task: String Operations

- **Objective:** Practice working with strings in Python.
- **Steps:**
    1. Create a string and perform operations like `upper()`, `lower()`, and `replace()`.
    2. Concatenate two strings.
    3. Use slicing to extract a substring.
- **Question:** What happens if you try to slice a string using an index that is out of range?

---

17. Task: Writing a Program with Functions and Lists

- **Objective:** Combine functions and lists in a program.
- **Steps:**
    1. Write a function that accepts a list of numbers and returns the largest number in the list.
    2. Call the function with a sample list.
- **Question:** How would you modify the function to return the smallest number in the list?

---

18. Task: Classes and Objects

- **Objective:** Learn the basics of object-oriented programming with classes and objects.
- **Steps:**
    1. Create a `Car` class with attributes like `make`, `model`, and `year`.
    2. Define a method that displays the car's information.
    3. Create an object of the `Car` class and call the method.
- **Question:** How would you modify the class to include a method that updates the car's model?

---

19. Task: Inheritance

- **Objective:** Learn how to use inheritance in Python.
- **Steps:**
    1. Create a base class `Animal` with a method `speak()`.
    2. Create a derived class `Dog` that inherits from `Animal` and adds a `bark()` method.
    3. Create an object of the `Dog` class and call both methods.
- **Question:** Can you override the `speak()` method in the derived class? How would you do it?

---

20. Task: Constructor and Initialization

- **Objective:** Understand how to initialize object attributes using the constructor (`__init__`).
- **Steps:**
    1. Create a `Book` class with `title` and `author` as attributes.
    2. Define the `__init__` method to initialize the object.
    3. Create an object of the `Book` class and print its attributes.
- **Question:** How would you modify the class to include a `price` attribute?

---

21. Task: Method Overriding

- **Objective:** Learn how to override methods in a derived class.
- **Steps:**
    1. Create a base class `Shape` with a method `area()`.
    2. Create a derived class `Circle` and override the `area()` method.
    3. Create an object of `Circle` and call the `area()` method.
- **Question:** How would you call the method of the base class from the derived class?

---

22. Task: Static Methods in Classes

- **Objective:** Learn how to use static methods in Python classes.
- **Steps:**
    1. Create a class `Calculator` with a static method `add()`.
    2. Call the static method without creating an instance of the class.
- **Question:** What is the difference between static methods and instance methods?

---

23. Task: String Formatting

- **Objective:** Practice string formatting in Python.
- **Steps:**
    1. Use f-strings to print variables in a formatted string.
    2. Use `.format()` to achieve the same result.
- **Question:** How does f-string formatting improve readability in comparison to other methods?

---

24. Task: Reading and Writing Files

- **Objective:** Learn to read and write data to files.
- **Steps:**
    1. Write a program that reads data from a file.
    2. Write a program that writes data to a file.
- **Question:** What happens if you try to open a file that doesn't exist?

---

25. Task: List Comprehension

- **Objective:** Understand how to use list comprehensions.
- **Steps:**
    1. Write a list comprehension to create a list of squares from 1 to 10.
    2. Use a conditional expression in the list comprehension to filter out odd numbers.
- **Question:** How would you modify the list comprehension to create a list of even numbers?

---

26. Task: Lambda Functions

- **Objective:** Learn how to use lambda functions for simple operations.
- **Steps:**
    1. Write a lambda function that returns the square of a number.
    2. Use the lambda function in a `map()` operation to square all numbers in a list.
- **Question:** How does a lambda function differ from a regular function in terms of usage?

---

27. Task: Using `zip()` Function

- **Objective:** Understand how to use the `zip()` function.
- **Steps:**
    1. Create two lists and zip them together.
    2. Convert the result into a list of tuples and print it.
- **Question:** How can you use `zip()` to combine three lists?

---

28. Task: Dictionary Comprehension

- **Objective:** Learn how to use dictionary comprehensions.
- **Steps:**
    1. Create a dictionary comprehension that creates key-value pairs of numbers and their squares.
    2. Print the resulting dictionary.
- **Question:** How would you modify the comprehension to create a dictionary where the key is the number and the value is the cube?

---

29. Task: Recursion

- **Objective:** Understand the concept of recursion in Python.
- **Steps:**
    1. Write a recursive function to calculate the factorial of a number.
    2. Call the function with different inputs.
- **Question:** What happens if the base case of a recursive function is missing or incorrect?

---

30. Task: Handling Multiple Exceptions

- **Objective:** Handle multiple exceptions in a program.
- **Steps:**
    1. Write a program that handles both `ZeroDivisionError` and `ValueError`.
    2. Test the program with different inputs.
- **Question:** How does the order of exception blocks affect the program's behavior?