PYTHON

Variables and Data Types:

1. Write a program to declare an integer variable and print its value.
2. Create a program that takes user input for their age and prints it.
3. Write a program that calculates the area of a rectangle (length * width).
4. Create a program that swaps the values of two variables.
5. Write a program that concatenates two strings and prints the result.
6. Develop a program that converts a Fahrenheit temperature to Celsius.
7. Create a program that takes a number as input, squares it, and prints the result.
8. Write a program that checks if a number is divisible by both 5 and 7.
9. Develop a program that calculates the average of three numbers.
10. Create a program that uses different data types (int, float, string) in one statement.
11. Write a program to check if a given year is a leap year.
12. Create a program that generates a random number and prints it.
13. Write a program that calculates the simple interest using user input for principal, rate, and time.
14. Develop a program that converts a string to uppercase.
15. Create a program that checks if a given number is positive, negative, or zero.
16. Write a program that concatenates three strings and prints the result.
17. Develop a program that converts a given number to a binary representation.
18. Create a program that uses the `len()` function to find the length of a list.
19. Write a program that uses the `input()` function to get the user's name and prints a greeting.
20. Develop a program that calculates the sum of digits in a given number.

Print() method Python:

1. **Basic Print:**
   - Write a program that prints "Hello, World!" to the console.

2. **Multiple Prints:**
   - Create a program that uses multiple `print` statements to output a message with line breaks.

3. **Formatted Output:**
   - Write a program that uses formatted strings to print variables with descriptive messages.

4. **String Concatenation:**
   - Develop a program that prints the concatenation of two strings.

5. **Escape Characters:**
   -Create a program that uses escape characters to print a multi-line string.

6. **Print without Line Break:**
   - Write a program that prints multiple items on the same line without line breaks.

7. **Variable Printing:**
   - Develop a program that takes user input and prints a personalized greeting.

8. **Number Printing:**
   - Create a program that prints the result of a mathematical expression.

9. **Precision Printing:**
  - Write a program that uses formatted strings to print a floating-point number with two decimal places.

10. **Print with Separator:**
  - Develop a program that prints multiple items separated by a specific character.

11. **Printing Variables and Constants:**
  - Create a program that prints the value of a variable and a constant.

12. **Print with End Parameter:**
  - Write a program that prints multiple items with a custom `end` parameter.

13. **Printing Raw Strings:**
  - Develop a program that uses a raw string to print a path without interpreting escape characters.

14. **Print with Triple Quotes:**
  - Create a program that uses triple-quoted strings to print a multi-line message.

15. **Printing Special Characters:**
  - Write a program that prints special characters (e.g., copyright symbol, newline) using Unicode escape sequences.

16. **Print with Separator and End:**
  - Develop a program that uses both the `sep` and `end` parameters in the `print` function.

17. **Formatted Output with Arithmetic:**
  - Create a program that uses formatted strings to print the result of an arithmetic expression.

18. **Print with File Parameter:**
  - Write a program that prints to a file using the `file` parameter in the `print` function.

19. **Conditional Printing:**
  - Develop a program that uses the `print` function inside a conditional statement.

20. **Print with f-strings:**
  - Write a program that uses f-strings to print variables and expressions.

Mathematical Operations in Python:

1. **Addition:**
  - Write a program that adds two numbers and prints the result.

2. **Subtraction:**
  - Create a program that subtracts one number from another and prints the result.

3. **Multiplication:**
  - Write a program that multiplies two numbers and prints the result.

4. **Division:**
  - Develop a program that divides one number by another and prints the result.

5. **Power:**

  - Create a program that calculates the square of a number.

6. **Square Root:**
  - Write a program that calculates the square root of a given number.

7. **Modulus:**
  - Develop a program that finds the remainder when one number is divided by another.

8. **Absolute Value:**
  - Create a program that calculates the absolute value of a number.

9. **Rounding:**
  - Write a program that rounds a floating-point number to the nearest integer.

10. **Ceiling and Floor:**
  - Develop a program that uses the `math` module to find the ceiling and floor of a given number.

11. **Trigonometric Functions:**
  - Write a program that uses the `math` module to calculate the sine and cosine of an angle.

12. **Exponential Function:**
  - Create a program that calculates the exponential function of a given number.

13. **Logarithm:**
  - Develop a program that uses the `math` module to find the natural logarithm of a number.

14. **Random Number:**
  - Write a program that generates a random number between 1 and 10.

15. **Area of a Circle:**
  - Create a program that calculates the area of a circle using the formula $A = \pi r^2$.

16. **Volume of a Sphere:**
  - Develop a program that calculates the volume of a sphere using the formula $V = (4/3)\pi r^3$.

17. **Percentage Calculation:**
  - Write a program that calculates the percentage of a given number.

18. **Sum of Natural Numbers:**
  - Create a program that finds the sum of the first 10 natural numbers.

19. **Hypotenuse Calculation:**
  - Develop a program that uses the Pythagorean theorem to find the hypotenuse of a right-angled triangle.

20. **Average of Three Numbers:**
  - Write a program that calculates the average of three numbers.

Conditional and Control Flow (if statements, loops):

21. Write a program that checks if a number is even or odd.
22. Create a program that prints the even numbers from 1 to 20 using a loop.

23. Write a program that prints the multiplication table of a given number up to 10.
24. Develop a program that checks if a year is a leap year.
25. Create a program that uses a loop to print the first 10 multiples of 4.
26. Write a program that finds the maximum of three numbers.
27. Develop a program that prints a countdown from 10 to 1 using a loop.
28. Create a program that uses a loop to print the factorial of a given number.
29. Write a program that prints the Fibonacci sequence up to the 15th term.
30. Develop a program that checks if a given number is a prime number.
31. Create a program that uses a loop to calculate the sum of the first 100 natural numbers.
32. Write a program that checks if a given character is a vowel or consonant.
33. Develop a program that uses a loop to print the square of numbers from 1 to 10.
34. Create a program that calculates the average of a list of numbers using a loop.
35. Write a program that uses a loop to find the factorial of a given number.
36. Develop a program that checks if a number is a perfect square.
37. Create a program that prints a pattern of stars in the shape of a right-angled triangle.
38. Write a program that uses nested loops to print a pattern of numbers.
39. Develop a program that checks if a number is a palindrome.
40. Create a program that uses a loop to print the sum of digits in a given number.

Lists and Strings:

41. Write a program that creates a list of your favorite foods and prints it.
42. Develop a program that adds an element to the end of a list.
43. Create a program that prints the length of a string.
44. Write a program that uses string slicing to reverse a word.
45. Develop a program that checks if a given word is a palindrome.
46. Create a program that finds the largest element in a list.
47. Write a program that removes duplicates from a list.
48. Develop a program that concatenates two lists.
49. Create a program that counts the number of occurrences of a specific element in a list.
50. Write a program that converts a string to lowercase.
51. Develop a program that uses list comprehension to create a list of squares of even numbers.
52. Create a program that uses a loop to find the average of a list of numbers.
53. Write a program that checks if a word contains the letter 'e'.
54. Develop a program that capitalizes the first letter of each word in a sentence.
55. Create a program that sorts a list of numbers in ascending order.
56. Write a program that finds the index of the first occurrence of a specific element in a list.
57. Develop a program that extracts vowels from a given string.
58. Create a program that checks if two lists have any common elements.
59. Write a program that removes the last element from a list.
60. Develop a program that uses string formatting to print a sentence.

String slicing, appending, accessing elements using indexing, deleting, and other string operations in Python:

1. **String Slicing:**
   - Write a program that uses string slicing to extract a substring from a given string.

2. **String Concatenation:**
   - Create a program that concatenates two strings and prints the result.

3. **Accessing Characters by Index:**
   - Write a program that accesses and prints individual characters of a string using indexing.

4. **Changing Characters by Index:**
   - Develop a program that modifies a specific character of a string using indexing.

5. **String Length:**
   - Create a program that calculates and prints the length of a string.

6. **Appending to a String:**
   - Write a program that appends a new string to an existing string.

7. **String Repetition:**
   - Develop a program that repeats a string a certain number of times and prints the result.

8. **Deleting Characters by Slicing:**
   - Create a program that deletes characters from a string using slicing.

9. **String Interpolation:**
   - Write a program that uses string interpolation to insert a variable into a string.

10. **Uppercase and Lowercase Conversion:**
   - Develop a program that converts a given string to uppercase and lowercase.

11. **Removing Whitespaces:**
   - Create a program that removes leading and trailing whitespaces from a string.

12. **String Formatting:**
   - Write a program that uses string formatting to display variables in a sentence.

13. **Checking Substring:**
   - Develop a program that checks if a given substring is present in a larger string.

14. **Replacing Substring:**
   - Create a program that replaces a specific substring with another string.

15. **Counting Occurrences:**
   - Write a program that counts the occurrences of a specific character in a string.

16. **Swapping Case:**
   - Develop a program that swaps the case of each character in a string.

17. **Reversing a String:**
   - Create a program that reverses a given string.

18. **Removing Characters:**
   - Write a program that removes a specific character from a string.

19. **String to List Conversion:**
   - Develop a program that converts a string to a list of characters.

20. **Deleting Entire String:**
   - Create a program that deletes an entire string variable.

Logical Operators:

1. Write a Python function that checks if a given number is both even and divisible by 3.
2. Create a program that uses logical operators to determine if a student passed both the math and science exams.
3. Implement a function that evaluates the expression (a > 10) and (b < 5) and returns the result.
4. Write a Python script that checks if a person is eligible to vote based on age (18 or older) and citizenship.
5. Develop a program that uses logical operators to verify if a user inputted password meets security criteria (length > 8 and includes a special character).
6. Create a function that determines whether a given year is a leap year using logical operators.
7. Write a Python program that checks if a number is positive and odd using logical operators.
8. Implement a function that evaluates the expression (x > 0) or (y < 0) and prints the result.
9. Develop a script that uses logical operators to check if a point is outside a given rectangular area.
10. Create a function that checks if a given string contains both uppercase and lowercase letters using logical operators.
11. Write a Python program that determines if a person is eligible for a senior discount based on age (65 or older) or membership.
12. Implement a function that checks if a triangle is equilateral (all sides equal) and has an angle greater than 90 degrees.
13. Develop a program that uses logical operators to check if a user inputted date is in a valid range (e.g., between 1900-01-01 and 2100-12-31).
14. Create a function that checks if a given list contains both positive and negative numbers using logical operators.
15. Write a Python script that evaluates a complex logical expression involving multiple operators and prints the result.
16. Implement a program that uses logical operators to determine if a person is eligible for a discounted bus fare based on age and disability.
17. Develop a function that checks if a given string is a valid email address using logical operators.
18. Write a Python program that determines if a number is a prime number and greater than 10.
19. Create a function that checks if a given character is a vowel or a digit using logical operators.
20. Develop a script that uses logical operators to validate user input for a username and password combination.

Randomization: random module

1. Write a Python function that generates a random integer between 1 and 100.
2. Create a program that simulates rolling a six-sided die and prints the result.
3. Implement a function that shuffles a list of numbers using randomization.
4. Develop a Python script that generates a random password of a given length.
5. Write a program that simulates flipping a coin and prints "Heads" or "Tails" accordingly.
6. Create a function that randomly selects a name from a list of names.
7. Develop a program that generates a random date within a given range (e.g., between 2000-01-01 and 2023-01-01).
8. Write a Python function that generates a random floating-point number between 0 and 1.
9. Create a program that randomly selects a card from a standard deck of playing cards.
10. Implement a function that generates a random alphanumeric string of a specified length.
11. Develop a Python script that randomly selects a color from a predefined list.
12. Write a program that uses randomization to simulate a simple game of rock-paper-scissors.

13. Create a function that randomly rearranges the letters in a given word.
14. Develop a program that generates a random sentence by combining random words from different lists.
15. Write a Python function that simulates a random walk, where each step is determined by a random direction (e.g., north, south, east, west).
16. Implement a program that uses randomization to simulate the outcome of a two-player dice game.
17. Create a function that generates a random matrix of integers with a given number of rows and columns.
18. Write a Python script that randomly selects a country from a list and prints its capital.
19. Develop a program that uses randomization to generate a sequence of musical notes for a melody.
20. Implement a function that randomly assigns tasks to team members from a list of tasks.

Error Handling:

1. ValueError Handling:
    a. Write a program that takes user input for an integer and handles a ValueError if the input is not a valid number.

2. FileNotFoundError Handling:
    a. Create a program that attempts to open a file, and if the file is not found, handles the FileNotFoundError gracefully.

3. ZeroDivisionError Handling:
a. Implement a program that calculates the division of two numbers but catches and handles the ZeroDivisionError.

4. KeyError Handling:
a. Write a Python script that uses a try-except block to handle a KeyError when accessing a nonexistent key in a dictionary.

5. Input Retry with FileNotFoundError:
a. Develop a program that prompts the user for a filename, tries to open the file, and handles the FileNotFoundError by asking for a different filename.

6. String to Integer Conversion with ValueError Handling:
a. Create a function that converts a given string to an integer and handles the ValueError if the conversion fails.

8. File Reading with FileNotFoundError and IOError Handling:
a. Implement a function that opens a file and reads its contents, handling both FileNotFoundError and IOError.

9. Non-Numeric Input Handling:
a. Create a Python script that asks the user to input two numbers and handles the exception if the input is not a number.

10. List Access with IndexError Handling:

a. Write a function that takes a list and an index as parameters and handles the IndexError gracefully.

11. Division Function with Exception Handling:
a. Create a function that attempts to divide two numbers, catching and handling any exception that may occur.

12. AssertionError for Positive Number:
a. Write a Python script that uses the assert statement to check if a given number is positive and raises an AssertionError if not.

13.String Conversion with AttributeError Handling:
Develop a function that takes a string and converts it to lowercase, handling any potential attribute error if the input is not a string.

14.Date Input with ValueError Handling:
Write a Python program that reads user input for a date in the format YYYY-MM-DD and handles ValueError for an invalid date.

15.Average Calculation with ZeroDivisionError Handling:
Develop a program that calculates the average of a list of numbers, handling a possible ZeroDivisionError.

16.TypeError in Data Type Concatenation:
Write a Python script that uses a try-except block to handle a TypeError when trying to concatenate incompatible data types.

17.Positive Integer Validation with ValueError and TypeError Handling:
Implement a function that validates user input for a positive integer, handling both ValueError and TypeError.

Functions:

1. Write a Python function that takes two parameters and returns their sum.
2. Create a program that defines a function to calculate the area of a rectangle given its length and width.
3. Implement a function that takes a list of numbers and returns the average.
4. Write a Python script that defines a function to check if a number is prime.
5. Develop a program that uses a function to find the maximum value in a list of numbers.
6. Create a function that takes a string as input and returns the reversed version of the string.
7. Write a Python script that defines a function to calculate the factorial of a given number.
8. Implement a function that checks if a given string is a palindrome.
9. Create a program that uses a function to determine if a year is a leap year.
10. Write a Python function that takes a list of words and returns the longest word.
11. Develop a program that defines a function to calculate the distance between two points in a 2D plane.
12. Implement a function that takes a number as input and returns a list of its factors.
13. Write a Python script that defines a function to convert Celsius to Fahrenheit.
14. Create a program that uses a function to count the occurrences of a specific element in a list.
15. Develop a function that simulates a simple calculator, taking two numbers and an operator as input and returning the result.

16. Write a Python function that generates a random password with a specified length.
17. Implement a function that generates a Fibonacci sequence of a given length.
18. Create a program that uses a function to validate a user's email address.
19. Write a function that takes a list of strings and returns a new list with the strings in alphabetical order.
20. Develop a Python script that uses a function to check if a given number is a perfect square.

For Loops:

1. Write a Python program that uses a for loop to print the numbers from 1 to 10.
2. Create a program that uses a for loop to calculate the sum of the first 100 natural numbers.
3. Implement a function that uses a for loop to find the factorial of a given number.
4. Write a Python script that uses a for loop to iterate over a list and print each element.
5. Develop a program that uses a for loop to print the square of each number in a given list.
6. Create a function that uses a for loop to count the number of vowels in a given string.
7. Write a Python program that uses a for loop to iterate over the characters of a string and print their ASCII values.
8. Implement a function that uses a for loop to check if a given list contains any negative numbers.
9. Develop a program that uses a for loop to find the maximum value in a list of numbers.
10. Create a Python script that uses a for loop to print the multiplication table for a given number.
11. Write a function that uses a for loop to calculate the sum of digits in a given number.
12. Implement a program that uses a for loop to reverse a given list.
13. Develop a Python script that uses a for loop to generate a list of squares for the numbers 1 to 10.
14. Create a program that uses a for loop to find the average of a list of numbers.
15. Write a Python function that uses a for loop to check if a given string is a palindrome.
16. Implement a function that uses a for loop to extract even numbers from a list.
17. Create a program that uses a for loop to find the prime numbers in a given range.
18. Write a Python script that uses a for loop to concatenate strings in a list.
19. Develop a function that uses a for loop to find the common elements between two lists.
20. Create a Python program that uses a for loop to print the first 10 terms of the Fibonacci sequence.

Code Blocks and Indentation:

1. Write a Python program with conditional statements and indentation to determine if a number is even or odd.
2. Create a program that uses indentation to define a function that prints the square of a given number.
3. Implement a function that uses indentation to create a block of code that checks if a number is positive or negative.
4. Write a Python script that uses indentation to define a loop that prints the numbers from 1 to 5.
5. Develop a program that uses indentation to create a block of code that calculates the average of a list of numbers.
6. Create a function that uses indentation to define a block of code that checks if a given string is empty.
7. Write a Python program that uses indentation to define a block of code within a class.
8. Implement a function that uses indentation to create a block of code that finds the maximum value in a list.
9. Develop a program that uses indentation to define a block of code that checks if a given year is a leap year.

10. Create a Python script that uses indentation to define a block of code that simulates a simple if-else statement.
11. Write a function that uses indentation to create a block of code that iterates over a list and prints each element.
12. Implement a program that uses indentation to define a block of code that calculates the factorial of a given number.
13. Develop a Python script that uses indentation to define a block of code that checks if a number is divisible by both 2 and 3.
14. Create a program that uses indentation to define a block of code that converts Celsius to Fahrenheit.
15. Write a Python function that uses indentation to define a block of code that counts the number of vowels in a given string.
16. Implement a function that uses indentation to create a block of code that filters out even numbers from a list.
17. Create a program that uses indentation to define a block of code that extracts uppercase letters from a string.
18. Write a Python script that uses indentation to define a block of code that finds the smallest element in a list.
19. Develop a function that uses indentation to create a block of code that checks if a given number is a perfect square.
20. Create a Python program that uses indentation to define a block of code that generates a list of prime numbers.

While Loops:

1. Write a Python program that uses a while loop to print the numbers from 1 to 10.
2. Create a program that uses a while loop to calculate the sum of the first 100 natural numbers.
3. Implement a function that uses a while loop to find the factorial of a given number.
4. Write a Python script that uses a while loop to iterate over a list and print each element.
5. Develop a program that uses a while loop to print the square of each number in a given list.
6. Create a function that uses a while loop to count the number of vowels in a given string.
7. Write a Python program that uses a while loop to iterate over the characters of a string and print their ASCII values.
8. Implement a function that uses a while loop to check if a given list contains any negative numbers.
9. Develop a program that uses a while loop to find the maximum value in a list of numbers.
10. Create a Python script that uses a while loop to print the multiplication table for a given number.
11. Write a function that uses a while loop to calculate the sum of digits in a given number.
12. Implement a program that uses a while loop to reverse a given list.
13. Develop a Python script that uses a while loop to generate a list of squares for the numbers 1 to 10.
14. Create a program that uses a while loop to find the average of a list of numbers.
15. Write a Python function that uses a while loop to check if a given string is a palindrome.
16. Implement a function that uses a while loop to extract even numbers from a list.
17. Create a program that uses a while loop to find the prime numbers in a given range.
18. Write a Python script that uses a while loop to concatenate strings in a list.
19. Develop a function that uses a while loop to find the common elements between two lists.
20. Create a Python program that uses

Pattern Printing:

Print these patterns:

Pattern #1 - Number Pattern Semi-Pyramid

```
1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

1 2 3 4 5 6

1 2 3 4 5 6 7
```

Pattern #2 - Printing the Same Digit Pattern Using Inverted Pyramid

```
7 7 7 7 7 7 7

7 7 7 7 7 7

7 7 7 7 7

7 7 7 7

7 7 7

7 7

7
```

Pattern #3 - Printing Numbers Using Inverted Pyramids

```
1 1 1 1 1 1 1

2 2 2 2 2 2

3 3 3 3 3

4 4 4 4

5 5 5

6 6

7
```

Pattern #4 - Printing Numbers in a Simple Triangle Pattern

```
1

2 2

3 3 3

4 4 4 4

5 5 5 5 5

6 6 6 6 6 6

7 7 7 7 7 7 7
```

Pattern #5 - Printing Descending Numbers Using the Inverted Pyramid

```
7 7 7 7 7 7 7

6 6 6 6 6 6

5 5 5 5 5

4 4 4 4

3 3 3

2 2

1
```

Pattern #6 - Printing Natural Numbers < 17 Using a Pyramid

```
1

2 3 4

5 6 7 8 9

10 11 12 13 14 15 16
```

Pattern #7 - Printing Numbers in a Reverse Semi-pyramid

```
1

2 1

3 2 1

4 3 2 1

5 4 3 2 1

6 5 4 3 2 1

7 6 5 4 3 2 1
```

Pattern #8 - Printing Digits From 22 in a Reverse Pattern

```
1

2

4 3

7 6 5

11 10 9 8

16 15 14 13 12

22 21 20 19 18 17
```

Pattern #9 - Printing a Number Pattern Using an Inverted Semi-pyramid

```
0 1 2 3 4 5 6 7 8

0 1 2 3 4 5 6 7

0 1 2 3 4 5 6

0 1 2 3 4 5

0 1 2 3 4

0 1 2 3

0 1 2

0 1
```

Pattern #10 - Printing a Number Pyramid in a Connected Pyramid

```
1234567

 234567

  34567

   4567

    567

     67

      7
```

Pattern #11 - Printing a Horizontal Table Using a Pyramid

```
0

0 1

0 2 4

0 3 6 9

0 4 8 12 16

0 5 10 15 20 25

0 6 12 18 24 30 36

0 7 14 21 28 35 42 49

0 8 16 24 32 40 48 56 64

0 9 18 27 36 45 54 63 72 81
```

Pattern #12 - Printing a Right-Angled Triangle of Number Pyramid by Mirroring

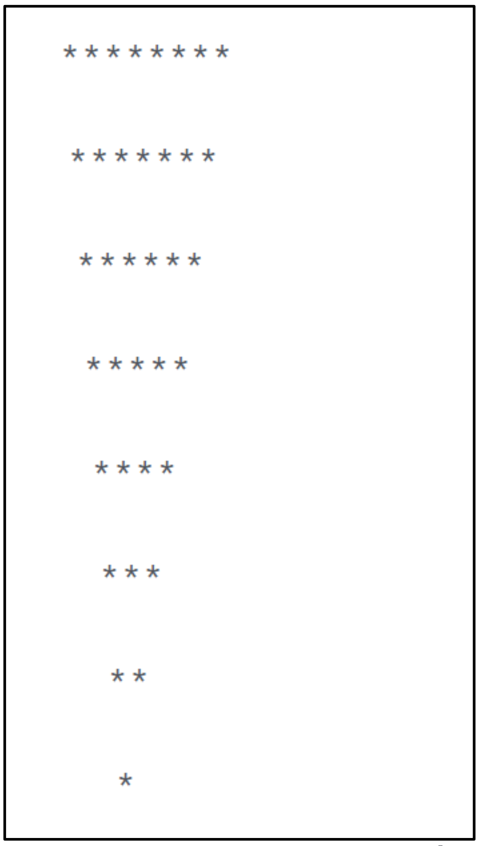
```
      1
     1 2
    1 2 3
   1 2 3 4
  1 2 3 4 5
 1 2 3 4 5 6
1 2 3 4 5 6 7
```

Pattern #13 - Printing a Pattern of Unique Digits Using a Pyramid

```
1
1 2 1
1 2 3 2 1
1 2 3 4 3 2 1
1 2 3 4 5 4 3 2 1
1 2 3 4 5 6 5 4 3 2 1
1 2 3 4 5 6 7 6 5 4 3 2 1
```

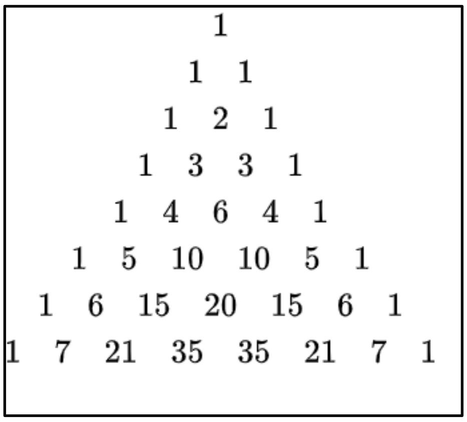
Pattern #15 - Printing a Star Equilateral Triangle

Pattern #16 - Printing a Star Pattern Triangle That Goes Downwards



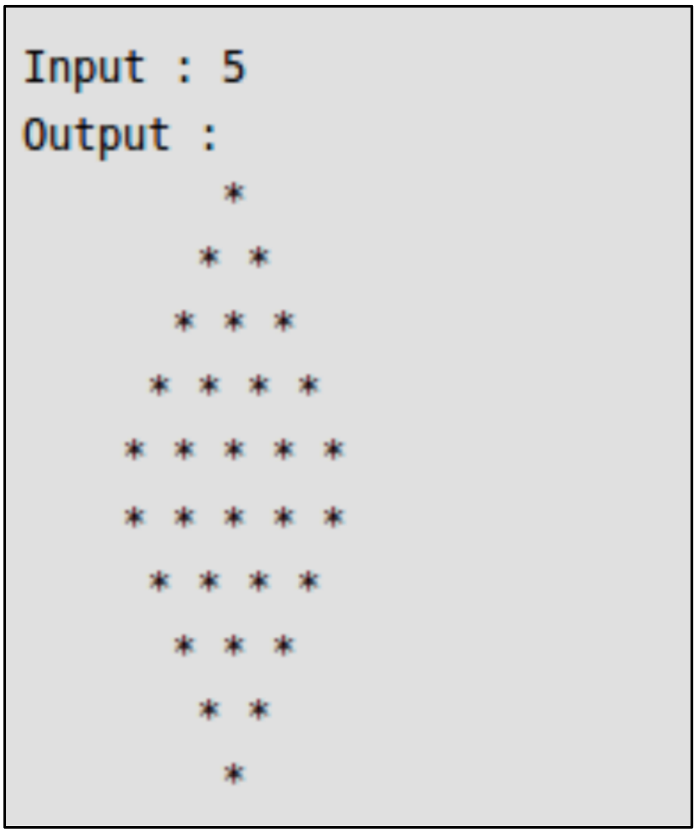
```
* * * * * * * *
 * * * * * * *
  * * * * * *
   * * * * *
    * * * *
     * * *
      * *
       *
```

Pattern #17 - Printing a Pascals Triangle



```
                1
              1   1
            1   2   1
          1   3   3   1
        1   4   6   4   1
     1   5   10   10   5   1
  1   6   15   20   15   6   1
1   7   21   35   35   21   7   1
```

Pattern #18 – Print this pattern



```
Input : 5
Output :
        *
       * *
      * * *
     * * * *
    * * * * *
    * * * * *
     * * * *
      * * *
       * *
        *
```

Positional and Keyword Arguments

1.Simple Addition Function:
Write a Python function named add_numbers that takes two positional arguments and returns their sum.

2.Exponential Function:
Create a function named power that accepts two keyword arguments, "base" and "exponent." The function should return the result of raising the base to the given exponent.

3.Rectangle Area with Defaults:
Define a function named rectangle_area that calculates the area of a rectangle. Include default values for length and width, and return the area.

4.Average Calculator:
Write a function named calculate_average that accepts a variable number of positional arguments and returns their average.

5.Conditional Transformation:
Develop a function named transform_value that takes a mandatory positional argument and an optional keyword argument "square." If "square" is True, square the value; otherwise, return the value as is.

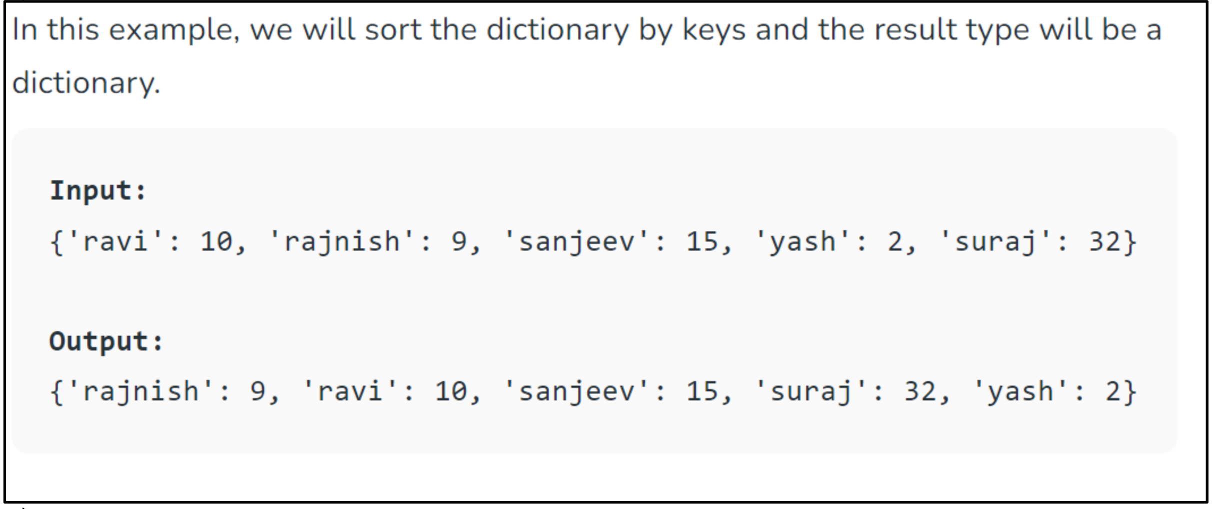
6.Triangle Validator:
Implement a function named is_valid_triangle that takes three positional arguments representing sides of a triangle. Check whether the sides can form a valid triangle, and raise a custom exception if not.

7.Nested Function Demo:
Create a function named outer_function that takes a positional argument. Inside, define a nested function named inner_function that takes a keyword argument. Demonstrate calling both functions.

8.Recursive Factorial:
Write a recursive function named factorial to calculate the factorial of a given number. Use a keyword argument to track the current result during recursion.

Python Dictionaries and Lists

1)

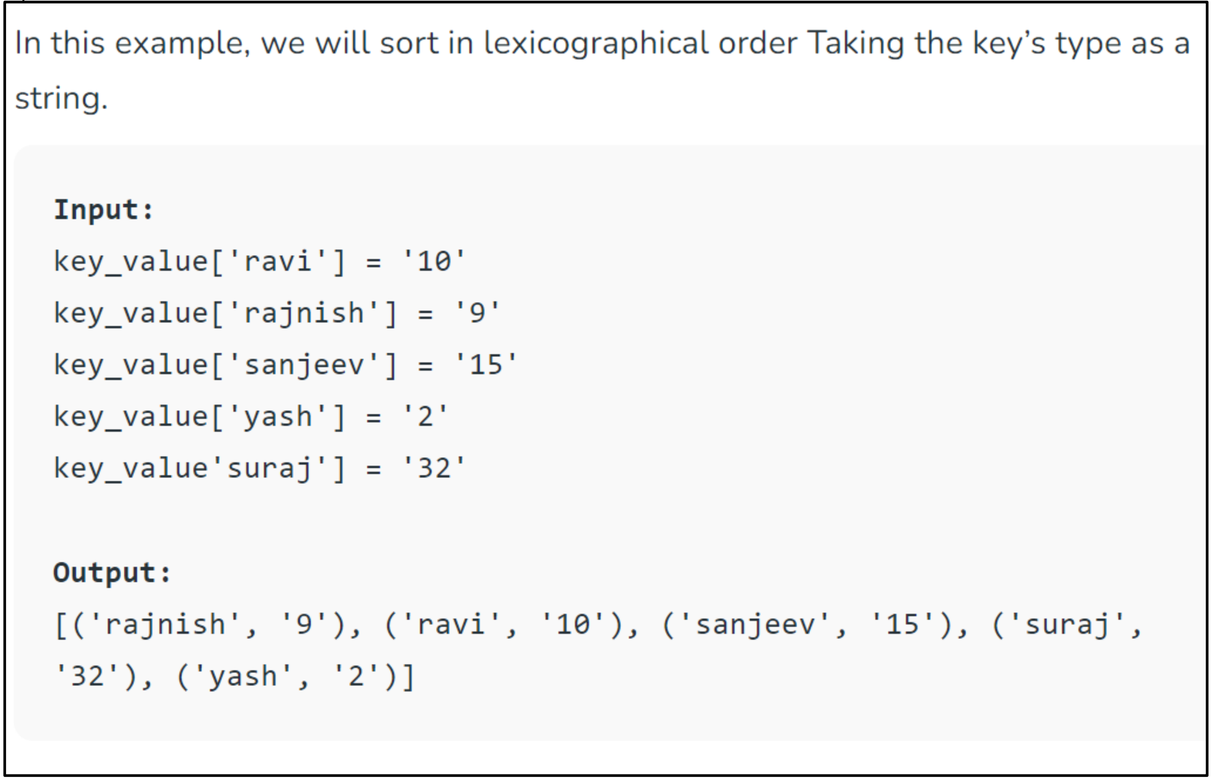


In this example, we will sort the dictionary by keys and the result type will be a dictionary.

```
Input:
{'ravi': 10, 'rajnish': 9, 'sanjeev': 15, 'yash': 2, 'suraj': 32}

Output:
{'rajnish': 9, 'ravi': 10, 'sanjeev': 15, 'suraj': 32, 'yash': 2}
```

2)



In this example, we will sort in lexicographical order Taking the key's type as a string.

```
Input:
key_value['ravi'] = '10'
key_value['rajnish'] = '9'
key_value['sanjeev'] = '15'
key_value['yash'] = '2'
key_value'suraj'] = '32'

Output:
[('rajnish', '9'), ('ravi', '10'), ('sanjeev', '15'), ('suraj',
'32'), ('yash', '2')]
```

3.Dictionary Basics:

Create an empty dictionary named my_dict and add key-value pairs for "name," "age," and "city."
4.List Manipulation:

Initialize a list named my_list with integers. Write a function to calculate the sum of all elements in the list.

5.Dictionary Operations:

Given a dictionary student_info with keys "name," "age," and "grade," write a function to print each key and its corresponding value.

6.List Comprehension:

Generate a list of squares for numbers 1 to 10 using list comprehension.

7.Dictionary Update:

Create two dictionaries, dict1 and dict2. Write a function to update dict1 with the key-value pairs from dict2.

8.List Slicing:

Given a list of 10 elements, write a program to print the first 5 elements and the last 5 elements using list slicing.

9.Nested Dictionaries:

Create a nested dictionary named school with keys "class1" and "class2." Each class should have a list of student names. Write a function to print the names of students in a specific class.

10.List Sorting:

Initialize a list of strings. Write a function to sort the list in alphabetical order.

11. 1Dictionary Iteration:

Given a dictionary of shopping items and their prices, write a program to calculate the total cost of items in the dictionary.

12.List Modification:

Create a list of numbers. Write a function to replace all even numbers in the list with their squares.

13.Dictionary Filtering:

Given a dictionary of students and their exam scores, write a function to filter out students who scored below a certain threshold.

14. List Concatenation:

Create two lists, list1 and list2, and write a function to concatenate them into a new list named combined_list.

15.Dictionary Key Check:

Write a function that checks if a given key exists in a dictionary. If the key exists, print its corresponding value; otherwise, print a default message.

16.List Removal:

Given a list of numbers, write a program to remove all occurrences of a specific number from the list.

17.Dictionary Merging:

Create two dictionaries, dict_a and dict_b, with some common keys. Write a function to merge the dictionaries, resolving conflicts by summing the values for common keys.

18.List Reversal:

Take a list of strings and write a function to reverse the order of elements in the list.

19.Dictionary Sorting:

Given a dictionary with keys as months and values as the number of days in each month, write a function to sort the dictionary based on the number of days.

20.List Unique Elements:

Create a list with duplicate elements. Write a function to generate a new list containing only the unique elements.

21.Nested Dictionary Access:

Given a nested dictionary representing a library, write a function to access the information about a specific book.

22.List Element Count:

Write a program that counts the occurrences of each element in a given list and stores the results in a dictionary.

Nested Collections

1.Nested List Manipulation:

Create a nested list representing a matrix. Write a function to find the sum of all elements in the matrix.

2.Nested Dictionary Access:

Given a nested dictionary representing a company's departments and employees, write a function to access information about a specific employee.

3.List of Dictionaries:

Create a list of dictionaries where each dictionary represents information about a person (name, age, city). Write a program to print the names of all individuals in the list.

4.Nested List Flattening:

Write a function to flatten a nested list, converting it into a single-level list.

5.Dictionary of Lists:

Create a dictionary where the keys are subjects, and the values are lists of students enrolled in each subject. Write a program to print the students in a specific subject.

6.List of Lists Sorting:

Generate a list of lists where each inner list contains a student's name and their scores in two subjects. Write a function to sort the list based on the average score of each student.

7.Dictionary of Dictionaries:

Create a dictionary of dictionaries to store information about books in a library. Each inner dictionary can represent a different book. Write a program to access details about a specific book.

8.List and Dictionary Combination:

Design a data structure to represent a school where classes are dictionaries (with students' names as keys and their scores as values), and the entire school is a list of classes. Write a function to calculate the average score for a specific student across all classes.

9.Nested Collections Filtering:

Given a list of dictionaries, each representing a person with information like name, age, and a list of hobbies, write a function to filter out people who have a specific hobby.

10.Dynamic Nested Collection Creation:

Write a program that dynamically creates a nested collection (either a list of lists or a dictionary of dictionaries) based on user input for dimensions and values.

Scope and Local/Global Variables

1.Local Variable Scope:

Write a function that defines a local variable inside it and tries to access it outside the function. Explain the result.

2.Global Variable Access:

Create a global variable outside of any function and try to modify its value within a function. Explain the result.

3.Local Variable Modification:

Define a function that takes an argument and modifies it inside the function. Check whether the original variable outside the function is affected.

4.Global Variable within a Function:

Write a function that uses a global variable and modifies it. Demonstrate how to properly declare a global variable within a function.

5.Scope of Variables:

Create a nested function, and within the inner function, try to access a variable defined in the outer function. Explain the concept of variable scope.
6.Function Parameters and Scope:

Write a function that takes parameters with the same names as global variables. Discuss how the function's behavior is affected by the scope of variables.

7.Global Variable Shadowing:

Declare a global variable with a specific value. Write a function that declares a local variable with the same name. Discuss how Python handles variable shadowing.

8.Modifying Global List:

Define a global list, and write a function that appends an element to the list. Discuss how global mutable objects (like lists) are affected by functions.

9.Global and Local Variables with the Same Name:

Write a program that declares a global variable and a function with a local variable having the same name. Demonstrate how Python differentiates between the two.

10.Global Variable Across Modules:

Create two Python files, one with a global variable and another that imports and modifies that variable. Discuss how global variables can be shared across modules.

Return vs. Print

1.Create a function that calculates the sum of two numbers and prints the result. Modify the function to return the result instead of printing it. Discuss the difference between using print and return in this context.

2.Function Return Values:

Write a function that takes two arguments and returns their product. Demonstrate calling the function and printing the result.

3.Print Inside a Function:

Develop a function that calculates the square of a number and prints the squared value inside the function. Discuss whether this is good practice and in what scenarios it might be useful.

4.Using Return for Multiple Values:

Create a function that takes two numbers as input and returns both their sum and product. Demonstrate calling the function and printing both values.

5.Print Statements in a Loop:

Write a program that uses a loop to print the squares of numbers from 1 to 5. Discuss how the use of print in a loop differs from using a function with return.

6.Function with No Return:

Define a function that performs a task (e.g., printing a message) but doesn't return any value. Discuss scenarios where a function without a return statement might be appropriate.

7.Returning None:

Create a function that explicitly returns None. Discuss when it might be necessary to use return None in a function.

8.Print vs. Return in a Loop:

Write a program that uses a loop to calculate the sum of numbers from 1 to 10. Compare the use of print statements inside the loop with using a function that returns the final sum.
9.Returning Multiple Data Types:

Develop a function that takes a number as input and returns both its square and its square root. Discuss how Python handles functions that return multiple data types.

10.Function Documentation:

Write a function with a docstring that explains whether the function uses print or return and provides information on what the function does.

Lamda Functions

1.Basic Lambda Function:

Write a lambda function that calculates the square of a given number.

2.Lambda with Filter:

Use the filter function and a lambda function to filter out even numbers from a list.

3.Lambda with Map:

Use the map function and a lambda function to square each element in a list.

4.Lambda with Sorting:

Sort a list of tuples based on the second element of each tuple using a lambda function.

5.Lambda with Conditional Expression:

Write a lambda function that returns "Even" if a given number is even and "Odd" otherwise.
6.Multiple Arguments in Lambda:

Create a lambda function that takes two arguments and returns their product.

7.Lambda in List Comprehension:

Use a lambda function in a list comprehension to create a new list containing the squares of numbers from 1 to 5.

8.Lambda in Key Function:

Sort a list of strings based on the length of each string using a lambda function as the key.

9.Lambda in Reduce:

Use the functools.reduce function and a lambda function to find the product of elements in a list.

10.Lambda in Dictionary Sorting:

Given a dictionary with string keys and integer values, sort the dictionary based on the values using a lambda function.