1. Python Intro

Notes:

- **Concepts to Learn**:
  - What is Python? History, Features
  - Installing Python
  - Python IDEs (IDLE, PyCharm, VS Code)
  - Running Python scripts

Problems:

- **Easy**:
  1. Install Python and run a "Hello, World!" script.
  2. Write a Python script that prints your name.
  3. Check the Python version installed on your system.
  4. Write a script that prints the current date and time.
  5. Write a script that prints "Welcome to Python programming!".
  6. Write a script that prints the first 10 natural numbers.
  7. Write a script that takes user input and prints it.
  8. Write a script that prints the current year.
  9. Write a script that prints the square of a number.
  10. Write a script that prints the sum of two numbers.
- **Medium**:
  1. Write a script that prints a multi-line string.
  2. Write a script that prints the even numbers between 1 and 50.
  3. Write a script that calculates the area of a rectangle using length and width.
  4. Write a script that takes two inputs and swaps their values.
  5. Write a script that prints the Fibonacci sequence up to 10 terms.
  6. Write a script that calculates the factorial of a number.
  7. Write a script that converts Celsius to Fahrenheit.
  8. Write a script that checks if a number is prime.
  9. Write a script that finds the maximum of three numbers.
  10. Write a script that reverses a string.
  11. Write a script that prints the ASCII value of a character.
  12. Write a script that converts kilometers to miles.
  13. Write a script that calculates the area of a triangle given its base and height.
  14. Write a script that prints the multiplication table of a given number.
  15. Write a script that finds the sum of digits of a number.
- **Difficult**:
  1. Write a script that prints all prime numbers between 1 and 100.
  2. Write a script that checks if a number is an Armstrong number.
  3. Write a script that generates a random number between 1 and 100 and asks the user to guess it.
  4. Write a script that calculates the GCD (Greatest Common Divisor) of two numbers.

5. Write a script that prints Pascal's triangle up to 5 levels.

2. Python Syntax

Notes:

- **Concepts to Learn**:
  - Basic syntax rules
  - Indentation importance
  - Writing and executing simple Python scripts
  - Commenting code

Problems:

- **Easy**:
  1. Write a script with multiple print statements and comments.
  2. Write a script that prints "Learning Python syntax!".
  3. Write a script that prints the current year and includes comments.
  4. Write a script that uses single-line comments.
  5. Write a script that uses multi-line comments.
  6. Write a script that prints numbers from 1 to 10 using a loop.
  7. Write a script that prints "Python Syntax Practice" three times.
  8. Write a script that uses both single-line and multi-line comments.
  9. Write a script that prints a simple arithmetic operation.
  10. Write a script that includes a commented-out section.
- **Medium**:
  1. Write a script that prints a pattern (e.g., triangle of stars).
  2. Write a script that prints even numbers between 1 and 50.
  3. Write a script that calculates and prints the square root of a number.
  4. Write a script that uses nested loops to print a pattern.
  5. Write a script that calculates the sum of the first 10 natural numbers.
  6. Write a script that prints a countdown from 10 to 1.
  7. Write a script that calculates the area and perimeter of a rectangle.
  8. Write a script that prints the multiplication table for numbers 1 to 5.
  9. Write a script that calculates the power of a number.
  10. Write a script that prints the first 10 Fibonacci numbers.
  11. Write a script that checks if a number is even or odd.
  12. Write a script that prints the reverse of a string.
  13. Write a script that counts the number of vowels in a string.
  14. Write a script that calculates the sum of a list of numbers.
  15. Write a script that prints a right-angled triangle pattern of numbers.
- **Difficult**:
  1. Write a script that finds and prints all prime numbers between 1 and 100.
  2. Write a script that generates a random number and asks the user to guess it.
  3. Write a script that calculates the factorial of a number using recursion.
  4. Write a script that prints Pascal's triangle up to 7 levels.

5. Write a script that calculates the GCD of two numbers using the Euclidean algorithm.

3. Python Variables

Notes:

- **Concepts to Learn**:
  - Variable declaration and assignment
  - Naming conventions
  - Dynamic typing
  - Variable scope

Problems:

- **Easy**:
  1. Assign values to variables and print them.
  2. Swap two variable values.
  3. Assign and print different data types to variables.
  4. Use variables to store and print your name and age.
  5. Calculate and print the area of a rectangle using length and width stored in variables.
  6. Assign a string to a variable and print it in uppercase.
  7. Assign the sum of two numbers to a variable and print it.
  8. Assign and print a list of your favorite fruits.
  9. Use variables to store and print the result of a mathematical expression.
  10. Create a variable with a floating-point number and print its integer part.
- **Medium**:
  1. Write a script that assigns and prints multiple variables in one line.
  2. Write a script that calculates the perimeter and area of a circle using variables.
  3. Write a script that stores and prints the first and last name separately.
  4. Write a script that swaps three variable values in a cyclic manner.
  5. Write a script that calculates the sum of squares of two numbers.
  6. Write a script that stores and prints a list of even numbers.
  7. Write a script that checks if a variable is an instance of a specific data type.
  8. Write a script that concatenates two strings stored in variables.
  9. Write a script that stores and prints the prime factors of a number.
  10. Write a script that assigns and prints a dictionary of student grades.
  11. Write a script that calculates the distance between two points in a 2D plane.
  12. Write a script that stores and prints the days of the week.
  13. Write a script that assigns and prints a tuple of colors.
  14. Write a script that finds and prints the maximum and minimum values from a list of numbers.
  15. Write a script that calculates the average of a list of numbers stored in a variable.
- **Difficult**:
  1. Write a script that generates and prints a Fibonacci sequence using variables.

2. Write a script that calculates the roots of a quadratic equation using variables.
3. Write a script that performs matrix addition using nested lists stored in variables.
4. Write a script that checks if a variable is a palindrome.
5. Write a script that simulates a basic banking system using variables for balance, deposit, and withdrawal.

4. Python Data Types

Notes:

- **Concepts to Learn**:
  - Numeric types (int, float, complex)
  - Strings
  - Lists, Tuples, Sets, Dictionaries
  - Type conversion and type checking

Problems:

- **Easy**:
  1. Assign different data types to variables and print them.
  2. Convert a float to an integer and print it.
  3. Create a string variable and print its length.
  4. Create a list of 5 elements and print the first and last elements.
  5. Create a tuple and print each element using a loop.
  6. Create a set and demonstrate adding and removing elements.
  7. Create a dictionary and print all key-value pairs.
  8. Convert an integer to a string and concatenate with another string.
  9. Check if a variable is of a specific type using `isinstance()`.
  10. Perform basic arithmetic operations on numeric variables.
- **Medium**:
  1. Write a script that takes a list of numbers and returns a new list with each number squared.
  2. Write a script that counts the frequency of each character in a string.
  3. Write a script that merges two dictionaries.
  4. Write a script that finds the intersection of two sets.
  5. Write a script that removes duplicates from a list.
  6. Write a script that converts a list of tuples into a dictionary.
  7. Write a script that slices a string to print every second character.
  8. Write a script that sorts a list of dictionaries by a specific key.
  9. Write a script that finds the maximum and minimum values in a dictionary.
  10. Write a script that concatenates two lists element-wise.
  11. Write a script that finds the common elements in three sets.
  12. Write a script that reverses a tuple.
  13. Write a script that converts a list of strings to a single string separated by commas.
  14. Write a script that finds the length of each word in a sentence.
  15. Write a script that checks if all elements in a list are unique.

- **Difficult**:
  1. Write a script that performs matrix multiplication using nested lists.
  2. Write a script that flattens a nested list.
  3. Write a script that finds the longest common substring in two strings.
  4. Write a script that implements a basic version of the set operations (union, intersection, difference) without using built-in functions.
  5. Write a script that generates a dictionary from a list of keys and a list of values.

5. Python Numbers

Notes:

- **Concepts to Learn**:
  - Numeric types (int, float, complex)
  - Arithmetic operations
  - Built-in functions for numbers (min, max, abs, etc.)

Problems:

- **Easy**:
  1. Perform basic arithmetic operations (addition, subtraction, multiplication, division).
  2. Find the remainder of a division using the modulus operator.
  3. Calculate the power of a number using the exponent operator.
  4. Round a floating-point number to two decimal places.
  5. Find the absolute value of a number.
  6. Find the maximum and minimum of a list of numbers.
  7. Convert a floating-point number to an integer.
  8. Generate a random number between 1 and 100.
  9. Find the square root of a number.
  10. Convert a decimal number to a binary string.
- **Medium**:
  1. Write a script that calculates the factorial of a number.
  2. Write a script that finds the greatest common divisor (GCD) of two numbers.
  3. Write a script that finds the least common multiple (LCM) of two numbers.
  4. Write a script that generates the first 10 prime numbers.
  5. Write a script that calculates the sum of the squares of the first 10 natural numbers.
  6. Write a script that checks if a number is a perfect square.
  7. Write a script that converts a binary string to a decimal number.
  8. Write a script that calculates the compound interest.
  9. Write a script that calculates the area of a circle given its radius.
  10. Write a script that finds the median of a list of numbers.
  11. Write a script that generates the Fibonacci sequence up to n terms.
  12. Write a script that finds the number of digits in a number.
  13. Write a script that checks if a number is an Armstrong number.

14. Write a script that finds the sum of digits of a number.
15. Write a script that generates a random floating-point number between 0 and 1.

- **Difficult**:
  1. Write a script that finds all prime numbers between 1 and 1000.
  2. Write a script that calculates the roots of a quadratic equation.
  3. Write a script that implements the Euclidean algorithm for finding GCD.
  4. Write a script that generates a magic square of order n.
  5. Write a script that calculates the sum of an arithmetic series.

6. Python Strings

Notes:

- **Concepts to Learn**:
  - Creating and manipulating strings
  - String methods and operations
  - String formatting

Problems:

- **Easy**:
  1. Create a string variable and print it.
  2. Concatenate two strings and print the result.
  3. Convert a string to uppercase and lowercase.
  4. Find the length of a string.
  5. Slice a string to get the first 5 characters.
  6. Replace a substring within a string.
  7. Check if a string starts with a specific substring.
  8. Split a string by spaces and print the result.
  9. Join a list of strings with a hyphen.
  10. Find the position of a substring in a string.
- **Medium**:
  1. Write a script that counts the number of vowels in a string.
  2. Write a script that reverses a string.
  3. Write a script that checks if a string is a palindrome.
  4. Write a script that removes all whitespace from a string.
  5. Write a script that finds the most frequent character in a string.
  6. Write a script that converts a string to title case.
  7. Write a script that counts the number of words in a string.
  8. Write a script that prints the ASCII value of each character in a string.
  9. Write a script that converts a list of strings to a single string.
  10. Write a script that finds all the substrings of a given length in a string.
  11. Write a script that checks if a string contains only digits.
  12. Write a script that prints a formatted string using f-strings.
  13. Write a script that removes a specific character from a string.
  14. Write a script that capitalizes the first letter of each word in a string.

15. Write a script that encodes a string using a basic Caesar cipher.

- **Difficult**:
  1. Write a script that finds the longest common substring between two strings.
  2. Write a script that implements a basic spell checker using a list of words.
  3. Write a script that compresses a string using run-length encoding.
  4. Write a script that checks if two strings are anagrams.
  5. Write a script that generates all permutations of a string.

7. Python Lists

Notes:

- **Concepts to Learn**:
  - Creating and manipulating lists
  - List methods and operations
  - List comprehensions

Problems:

- **Easy**:
  1. Create a list of 5 elements and print it.
  2. Access and print the first and last elements of a list.
  3. Add an element to the end of a list.
  4. Remove an element from a list.
  5. Find the length of a list.
  6. Check if an element is in a list.
  7. Sort a list in ascending order.
  8. Reverse the order of elements in a list.
  9. Create a list of numbers from 1 to 10 using `range()`.
  10. Print each element of a list using a loop.
- **Medium**:
  1. Write a script that finds the sum of all elements in a list.
  2. Write a script that finds the largest and smallest elements in a list.
  3. Write a script that removes duplicates from a list.
  4. Write a script that concatenates two lists.
  5. Write a script that finds the intersection of two lists.
  6. Write a script that flattens a nested list.
  7. Write a script that creates a list of squares of numbers from 1 to 10.
  8. Write a script that finds the second largest element in a list.
  9. Write a script that splits a list into two halves.
  10. Write a script that rotates a list by a given number of positions.
  11. Write a script that counts the occurrences of each element in a list.
  12. Write a script that finds the common elements in three lists.
  13. Write a script that removes all occurrences of a specific element from a list.
  14. Write a script that finds the cumulative sum of a list of numbers.
  15. Write a script that finds the maximum product of two elements in a list.

- **Difficult**:
    1. Write a script that sorts a list of tuples based on the second element.
    2. Write a script that finds all subsets of a list.
    3. Write a script that finds the longest increasing subsequence in a list.
    4. Write a script that merges two sorted lists into a single sorted list.
    5. Write a script that finds the k-th smallest element in a list using the Quickselect algorithm.

8. Python Tuples

Notes:

- **Concepts to Learn**:
    - Creating and accessing tuples
    - Tuple operations and methods
    - Immutability of tuples

Problems:

- **Easy**:
    1. Create a tuple with 5 elements and print it.
    2. Access and print the first and last elements of a tuple.
    3. Concatenate two tuples.
    4. Find the length of a tuple.
    5. Convert a list to a tuple.
    6. Create a tuple with a single element.
    7. Check if an element is in a tuple.
    8. Slice a tuple to get the first 3 elements.
    9. Find the index of an element in a tuple.
    10. Unpack a tuple into individual variables.
- **Medium**:
    1. Write a script that counts the occurrences of an element in a tuple.
    2. Write a script that converts a tuple to a string.
    3. Write a script that finds the maximum and minimum elements in a tuple.
    4. Write a script that converts a tuple of tuples to a list of lists.
    5. Write a script that finds the length of each word in a tuple of strings.
    6. Write a script that finds the product of all elements in a tuple.
    7. Write a script that finds the common elements between two tuples.
    8. Write a script that reverses a tuple.
    9. Write a script that finds the index of the first occurrence of a specific element in a tuple.
    10. Write a script that merges two tuples by alternating their elements.
    11. Write a script that converts a list of tuples into a dictionary.
    12. Write a script that finds the sum of elements in a tuple of numbers.
    13. Write a script that creates a tuple from a string.
    14. Write a script that removes all occurrences of a specific element from a tuple.

15. Write a script that converts a tuple of numbers into a tuple of their squares.

- **Difficult**:
  1. Write a script that sorts a list of tuples based on the second element of each tuple.
  2. Write a script that finds all unique elements in a tuple.
  3. Write a script that finds the Cartesian product of two tuples.
  4. Write a script that creates a tuple of tuples containing number pairs (i, i^2) for i from 1 to 10.
  5. Write a script that checks if a tuple is a palindrome.

9. Python Sets

Notes:

- **Concepts to Learn**:
  - Creating and manipulating sets
  - Set methods and operations
  - Set comprehensions.

Problems:

- **Easy**:
  1. Create a set with 5 elements and print it.
  2. Add an element to a set.
  3. Remove an element from a set.
  4. Check if an element is in a set.
  5. Find the length of a set.
  6. Clear all elements from a set.
  7. Create a set from a list.
  8. Find the union of two sets.
  9. Find the intersection of two sets.
  10. Find the difference between two sets.
- **Medium**:
  1. Write a script that finds the symmetric difference of two sets.
  2. Write a script that checks if one set is a subset of another.
  3. Write a script that finds the maximum and minimum elements in a set.
  4. Write a script that converts a set to a list.
  5. Write a script that finds the common elements between three sets.
  6. Write a script that removes duplicates from a list using a set.
  7. Write a script that finds the sum of elements in a set of numbers.
  8. Write a script that checks if two sets are disjoint.
  9. Write a script that finds all elements in a set that are not in another set.
  10. Write a script that finds the length of the longest set of consecutive numbers.
  11. Write a script that finds the union of multiple sets.
  12. Write a script that checks if a set is empty.
  13. Write a script that counts the number of unique words in a sentence.
  14. Write a script that creates a set from a string and prints the unique characters.

15. Write a script that finds the common divisors of two numbers using sets.

- **Difficult**:
  1. Write a script that generates all subsets of a set.
  2. Write a script that finds the Cartesian product of two sets.
  3. Write a script that finds all unique combinations of elements in a set.
  4. Write a script that implements a basic set calculator (union, intersection, difference).
  5. Write a script that finds the longest sequence of consecutive elements in a set.

10. Python Dictionaries

Notes:

- **Concepts to Learn**:
  - Creating and manipulating dictionaries
  - Dictionary methods and operations
  - Dictionary comprehensions

Problems:

- **Easy**:
  1. Create a dictionary with 5 key-value pairs and print it.
  2. Access and print a value from a dictionary using its key.
  3. Add a new key-value pair to a dictionary.
  4. Remove a key-value pair from a dictionary.
  5. Find the length of a dictionary.
  6. Check if a key is in a dictionary.
  7. Iterate over all key-value pairs in a dictionary.
  8. Create a dictionary from two lists: one of keys and one of values.
  9. Print all keys and values from a dictionary separately.
  10. Create a dictionary using dictionary comprehension.
- **Medium**:
  1. Write a script that merges two dictionaries.
  2. Write a script that finds the key with the maximum value in a dictionary.
  3. Write a script that counts the occurrences of each character in a string using a dictionary.
  4. Write a script that inverts a dictionary (keys become values and values become keys).
  5. Write a script that converts a list of tuples into a dictionary.
  6. Write a script that removes all keys with a specific value from a dictionary.
  7. Write a script that finds the sum of all values in a dictionary.
  8. Write a script that finds the common keys between two dictionaries.
  9. Write a script that creates a dictionary from a string, with characters as keys and their counts as values.
  10. Write a script that sorts a dictionary by its keys.
  11. Write a script that finds the average of values in a dictionary.

12. Write a script that finds the difference between two dictionaries.
13. Write a script that creates a dictionary of lists from a list of dictionaries.
14. Write a script that updates the values of a dictionary using another dictionary.
15. Write a script that checks if a dictionary is empty.

- **Difficult**:
  1. Write a script that flattens a nested dictionary.
  2. Write a script that finds the top 3 keys with the highest values in a dictionary.
  3. Write a script that merges two dictionaries, summing values for common keys.
  4. Write a script that creates a dictionary of word frequencies from a text file.
  5. Write a script that implements a basic contact book using a dictionary.

11. Python If...Else

Notes:

- **Concepts to Learn**:
  - Conditional statements (if, elif, else)
  - Nested conditions
  - Logical operators (and, or, not)

Problems:

- **Easy**:
  1. Write a script that checks if a number is positive or negative.
  2. Write a script that checks if a number is even or odd.
  3. Write a script that checks if a number is divisible by 5.
  4. Write a script that checks if a character is a vowel.
  5. Write a script that checks if a person is eligible to vote (age >= 18).
  6. Write a script that finds the largest of three numbers.
  7. Write a script that checks if a year is a leap year.
  8. Write a script that checks if a string is empty.
  9. Write a script that prints "Hello, World!" if a variable is equal to "Python".
  10. Write a script that checks if a number is in a given range.
- **Medium**:
  1. Write a script that checks if a string is a palindrome.
  2. Write a script that finds the grade of a student based on their score.
  3. Write a script that checks if a triangle is equilateral, isosceles, or scalene.
  4. Write a script that finds the smallest and largest digits in a number.
  5. Write a script that checks if a given year, month, and day form a valid date.
  6. Write a script that checks if a string starts and ends with the same character.
  7. Write a script that checks if two strings are anagrams.
  8. Write a script that categorizes a number as small, medium, or large.
  9. Write a script that checks if a number is a perfect square.
  10. Write a script that finds the maximum of three numbers using nested if-else.
  11. Write a script that checks if a character is an uppercase letter.
  12. Write a script that checks if a number is a prime number.

13. Write a script that finds the middle number among three different numbers.
14. Write a script that checks if a given time is AM or PM.
15. Write a script that checks if a string contains only alphabets.

- **Difficult**:
  1. Write a script that checks if a given password is strong (contains uppercase, lowercase, numbers, and special characters).
  2. Write a script that finds the intersection of two lists without using set operations.
  3. Write a script that categorizes a given angle as acute, right, obtuse, or straight.
  4. Write a script that checks if a matrix is an identity matrix.
  5. Write a script that finds the day of the week for a given date.

12. Python Loops

Notes:

- **Concepts to Learn**:
  - `for` and `while` loops
  - Loop control statements (break, continue, pass)
  - Nested loops

Problems:

- **Easy**:
  1. Write a script that prints numbers from 1 to 10 using a `for` loop.
  2. Write a script that prints the even numbers from 1 to 20.
  3. Write a script that prints the elements of a list using a `for` loop.
  4. Write a script that prints the characters of a string using a `for` loop.
  5. Write a script that prints numbers from 10 to 1 using a `while` loop.
  6. Write a script that calculates the sum of the first 10 natural numbers.
  7. Write a script that prints the multiplication table of a given number.
  8. Write a script that prints all the prime numbers between 1 and 50.
  9. Write a script that prints the Fibonacci sequence up to n terms.
  10. Write a script that reverses a string using a loop.
- **Medium**:
  1. Write a script that finds the factorial of a number using a loop.
  2. Write a script that finds the sum of all elements in a list.
  3. Write a script that finds the maximum and minimum elements in a list using a loop.
  4. Write a script that prints all the even-indexed characters of a string.
  5. Write a script that finds the sum of the digits of a number using a loop.
  6. Write a script that prints the first 10 multiples of a given number.
  7. Write a script that prints the elements of a list in reverse order using a loop.
  8. Write a script that finds the number of vowels in a string using a loop.
  9. Write a script that prints a pattern of stars in a right-angled triangle.
  10. Write a script that calculates the sum of squares of the first 10 natural numbers.
  11. Write a script that prints the ASCII values of all the characters in a string.

12. Write a script that finds the largest and smallest digits in a number using a loop.
13. Write a script that prints the sum of the even and odd numbers from 1 to 20.
14. Write a script that finds the number of occurrences of a specific character in a string.
15. Write a script that prints the factorial of a number using a `for` loop.

- **Difficult**:
  1. Write a script that checks if a number is a palindrome using a loop.
  2. Write a script that generates the Pascal's triangle up to n rows.
  3. Write a script that finds the GCD of two numbers using a loop.
  4. Write a script that prints all the unique elements in a list using a loop.
  5. Write a script that finds the longest word in a sentence using a loop.

13. Python Functions

Notes:

- **Concepts to Learn**:
  - Defining and calling functions
  - Function parameters and arguments
  - Return values
  - Lambda functions
  - Scope and lifetime of variables

Problems:

- **Easy**:
  1. Write a function that prints "Hello, World!".
  2. Write a function that takes two numbers as arguments and returns their sum.
  3. Write a function that takes a string as an argument and returns its length.
  4. Write a function that takes a list of numbers as an argument and returns the maximum value.
  5. Write a function that takes a number as an argument and returns its factorial.
  6. Write a function that takes a list of numbers as an argument and returns the average.
  7. Write a function that takes two strings as arguments and returns their concatenation.
  8. Write a function that takes a number as an argument and returns True if it's even, False otherwise.
  9. Write a function that takes a list of numbers as an argument and returns the sum of its elements.
  10. Write a function that takes a string as an argument and returns it in uppercase.
- **Medium**:
  1. Write a function that takes a list of strings as an argument and returns the longest string.
  2. Write a function that takes a string as an argument and returns the number of vowels in it.

3. Write a function that takes a number as an argument and returns True if it's a prime number, False otherwise.
4. Write a function that takes a list of numbers as an argument and returns a new list with only the even numbers.
5. Write a function that takes a number n as an argument and returns the Fibonacci sequence up to n terms.
6. Write a function that takes two lists as arguments and returns their intersection.
7. Write a function that takes a list of numbers as an argument and returns a new list with each element squared.
8. Write a function that takes a string as an argument and returns a dictionary with the count of each character.
9. Write a function that takes a list of numbers as an argument and returns the second largest number.
10. Write a function that takes a list of numbers as an argument and returns a sorted version of the list.

- **Difficult**:
  1. Write a function that takes two numbers as arguments and returns their GCD using recursion.
  2. Write a function that takes a list of numbers as an argument and returns the median.
  3. Write a function that takes a string as an argument and returns all possible permutations of the string.
  4. Write a function that takes a list of numbers as an argument and returns a list of the prime numbers in it.
  5. Write a function that takes a list of dictionaries and a key, and returns a new dictionary grouped by the values of the key.

14. Python Classes and Objects

Notes:

- **Concepts to Learn**:
  - Defining classes and creating objects
  - Class attributes and methods
  - Inheritance
  - Polymorphism

Problems:

- **Easy**:
  1. Create a class named `Person` with attributes `name` and `age`. Create an object of the class and print the attributes.
  2. Create a class named `Rectangle` with attributes `length` and `width`. Write a method to calculate the area of the rectangle.
  3. Create a class named `Circle` with an attribute `radius`. Write a method to calculate the circumference of the circle.

4. Create a class named `Student` with attributes `name` and `marks`. Write a method to display the student's details.
5. Create a class named `BankAccount` with attributes `account_number` and `balance`. Write methods to deposit and withdraw money.
6. Create a class named `Car` with attributes `make`, `model`, and `year`. Write a method to display the car's details.
7. Create a class named `Book` with attributes `title`, `author`, and `pages`. Write a method to display the book's details.
8. Create a class named `Employee` with attributes `name` and `salary`. Write a method to give the employee a raise.
9. Create a class named `Animal` with an attribute `species`. Write a method to make the animal sound.
10. Create a class named `House` with attributes `address` and `price`. Write a method to display the house's details.

- **Medium**:
  1. Create a class named `Person` with attributes `name` and `age`. Write a method to check if the person is an adult.
  2. Create a class named `Rectangle` with attributes `length` and `width`. Write a method to check if the rectangle is a square.
  3. Create a class named `Circle` with an attribute `radius`. Write a method to calculate the area of the circle.
  4. Create a class named `Student` with attributes `name` and `marks`. Write a method to calculate the average marks.
  5. Create a class named `BankAccount` with attributes `account_number` and `balance`. Write a method to transfer money to another account.
  6. Create a class named `Car` with attributes `make`, `model`, and `year`. Write a method to check if the car is new or old.
  7. Create a class named `Book` with attributes `title`, `author`, and `pages`. Write a method to check if the book is a novel or a short story.
  8. Create a class named `Employee` with attributes `name` and `salary`. Write a method to calculate the annual salary.
  9. Create a class named `Animal` with an attribute `species`. Write a method to check if the animal is a mammal.
  10. Create a class named `House` with attributes `address` and `price`. Write a method to compare the price of two houses.
  11. Create a class named `Person` with attributes `name` and `age`. Write a method to display the person's details and another method to change the person's name.
  12. Create a class named `Rectangle` with attributes `length` and `width`. Write a method to compare the area of two rectangles.
  13. Create a class named `Circle` with an attribute `radius`. Write a method to compare the circumference of two circles.
  14. Create a class named `Student` with attributes `name` and `marks`. Write a method to compare the marks of two students.
  15. Create a class named `BankAccount` with attributes `account_number` and `balance`. Write a method to check if the account has sufficient funds for a withdrawal.

- **Difficult**:
  1. Create a class named `Person` with attributes `name` and `age`. Create a subclass named `Employee` with an additional attribute `salary`. Write a method to display the employee's details.
  2. Create a class named `Rectangle` with attributes `length` and `width`. Create a subclass named `Square` with a method to check if the square is perfect.
  3. Create a class named `Circle` with an attribute `radius`. Create a subclass named `Cylinder` with an additional attribute `height` and a method to calculate the volume.
  4. Create a class named `Student` with attributes `name` and `marks`. Create a subclass named `GraduateStudent` with an additional attribute `degree` and a method to display the degree.
  5. Create a class named `BankAccount` with attributes `account_number` and `balance`. Create a subclass named `SavingsAccount` with an additional attribute `interest_rate` and a method to calculate the interest.

15. Python Modules

Notes:

- **Concepts to Learn**:
  - Importing and using modules
  - Creating custom modules
  - Standard library modules

Problems:

- **Easy**:
  1. Import the `math` module and print the value of pi.
  2. Import the `random` module and print a random number between 1 and 10.
  3. Import the `datetime` module and print the current date and time.
  4. Import the `os` module and print the current working directory.
  5. Import the `sys` module and print the Python version.
  6. Import the `time` module and print the current time.
  7. Import the `calendar` module and print the calendar for the current month.
  8. Import the `collections` module and create a `Counter` object.
  9. Import the `json` module and parse a JSON string.
  10. Import the `re` module and find all the digits in a string using a regular expression.
- **Medium**:
  1. Write a script that uses the `math` module to calculate the area of a circle.
  2. Write a script that uses the `random` module to shuffle a list of numbers.
  3. Write a script that uses the `datetime` module to find the difference between two dates.
  4. Write a script that uses the `os` module to list all files in a directory.
  5. Write a script that uses the `sys` module to print all command-line arguments.

6. Write a script that uses the `time` module to measure the execution time of a function.
7. Write a script that uses the `calendar` module to check if a year is a leap year.
8. Write a script that uses the `collections` module to count the frequency of elements in a list.
9. Write a script that uses the `json` module to convert a dictionary to a JSON string.
10. Write a script that uses the `re` module to find all the words in a string that start with a vowel.
11. Write a custom module with a function that calculates the factorial of a number.
12. Write a custom module with a function that checks if a number is prime.
13. Write a custom module with a function that reverses a string.
14. Write a custom module with a function that finds the maximum element in a list.
15. Write a custom module with a function that sorts a list of numbers.

- **Difficult**:
  1. Write a script that uses the `os` module to copy a file from one directory to another.
  2. Write a script that uses the `sys` module to dynamically import a module based on user input.
  3. Write a script that uses the `datetime` module to generate a list of dates between two given dates.
  4. Write a custom module with a function that calculates the GCD of two numbers using recursion.
  5. Write a custom module with a function that finds the longest common subsequence of two strings.

16. Python File Handling

Notes:

- **Concepts to Learn**:
  - Opening, reading, writing, and closing files
  - File modes (read, write, append)
  - Working with different file formats (text, CSV, JSON)

Problems:

- **Easy**:
  1. Write a script that opens a file in read mode and prints its contents.
  2. Write a script that opens a file in write mode and writes "Hello, World!" to it.
  3. Write a script that opens a file in append mode and appends a line of text to it.
  4. Write a script that reads a file line by line and prints each line.
  5. Write a script that counts the number of lines in a file.
  6. Write a script that reads the first n lines of a file.
  7. Write a script that reads the last n lines of a file.
  8. Write a script that opens a file and prints only the first word of each line.
  9. Write a script that reads a file and converts each line to uppercase.

10. Write a script that reads a file and replaces all occurrences of a specific word with another word.

- **Medium**:
  1. Write a script that counts the number of words in a file.
  2. Write a script that finds the longest word in a file.
  3. Write a script that reads a file and prints the unique words.
  4. Write a script that reads two files and writes their contents to a new file.
  5. Write a script that reads a file and prints the frequency of each word.
  6. Write a script that reads a CSV file and prints each row.
  7. Write a script that writes a list of dictionaries to a CSV file.
  8. Write a script that reads a JSON file and prints its contents.
  9. Write a script that writes a dictionary to a JSON file.
  10. Write a script that reads a file and removes all blank lines.
  11. Write a script that reads a file and counts the number of occurrences of each character.
  12. Write a script that reads a file and prints the lines that contain a specific word.
  13. Write a script that reads a file and prints the words that occur more than n times.
  14. Write a script that reads a file and prints the lines that start with a specific letter.
  15. Write a script that reads a file and prints the lines that end with a specific letter.
- **Difficult**:
  1. Write a script that reads a file and prints the lines in reverse order.
  2. Write a script that reads a file and removes all punctuation marks.
  3. Write a script that reads a file and writes only the unique lines to a new file.
  4. Write a script that reads a CSV file and calculates the average of a specific column.
  5. Write a script that reads a JSON file and updates a specific value.

17. Python Exception Handling

Notes:

- **Concepts to Learn**:
  - Try, except, finally blocks
  - Handling specific exceptions
  - Raising exceptions
  - Custom exceptions

Problems:

- **Easy**:
  1. Write a script that catches a `ZeroDivisionError` and prints an appropriate message.
  2. Write a script that catches a `ValueError` when converting a string to an integer.
  3. Write a script that catches a `FileNotFoundError` and prints an appropriate message.

4. Write a script that uses a `try` block to catch any exception and prints a generic error message.
5. Write a script that uses a `try` block to catch multiple specific exceptions and prints appropriate messages for each.
6. Write a script that uses a `finally` block to print a message whether an exception occurs or not.
7. Write a script that raises a `ValueError` with a custom error message.
8. Write a script that raises a `TypeError` with a custom error message.
9. Write a script that raises a `KeyError` with a custom error message.
10. Write a script that raises a `IndexError` with a custom error message.

- **Medium**:
  1. Write a script that reads a file and handles any exceptions that occur during file operations.
  2. Write a script that converts a string to an integer and handles any exceptions that occur during the conversion.
  3. Write a script that calculates the division of two numbers and handles any exceptions that occur during the calculation.
  4. Write a script that reads a JSON file and handles any exceptions that occur during file operations or JSON parsing.
  5. Write a script that reads a CSV file and handles any exceptions that occur during file operations or CSV parsing.
  6. Write a script that connects to a database and handles any exceptions that occur during the connection or queries.
  7. Write a script that fetches data from a URL and handles any exceptions that occur during the HTTP request.
  8. Write a script that performs multiple mathematical operations and handles any exceptions that occur during the calculations.
  9. Write a script that parses an XML file and handles any exceptions that occur during file operations or XML parsing.
  10. Write a script that reads a file and handles any exceptions that occur during file operations, printing specific messages for each exception type.
- **Difficult**:
  1. Write a script that defines a custom exception class for invalid age and raises it when an invalid age is encountered.
  2. Write a script that defines a custom exception class for invalid email addresses and raises it when an invalid email is encountered.
  3. Write a script that defines a custom exception class for invalid passwords and raises it when an invalid password is encountered.
  4. Write a script that defines a custom exception class for insufficient funds and raises it when a withdrawal is attempted with insufficient funds.
  5. Write a script that defines a custom exception class for out-of-stock products and raises it when a product is out of stock.

18. Python Testing

Notes:

- **Concepts to Learn**:
    - Writing test cases
    - Unit testing
    - Using testing frameworks (unittest, pytest)

Problems:

- **Easy**:
    1. Write a simple test case for a function that adds two numbers.
    2. Write a simple test case for a function that checks if a number is even.
    3. Write a simple test case for a function that returns the length of a string.
    4. Write a simple test case for a function that finds the maximum value in a list.
    5. Write a simple test case for a function that reverses a string.
    6. Write a simple test case for a function that returns the square of a number.
    7. Write a simple test case for a function that checks if a string is a palindrome.
    8. Write a simple test case for a function that returns the factorial of a number.
    9. Write a simple test case for a function that calculates the area of a circle.
    10. Write a simple test case for a function that checks if a number is prime.
- **Medium**:
    1. Write a test case for a function that calculates the sum of a list of numbers.
    2. Write a test case for a function that finds the longest word in a sentence.
    3. Write a test case for a function that checks if a list contains duplicate elements.
    4. Write a test case for a function that finds the GCD of two numbers.
    5. Write a test case for a function that merges two dictionaries.
    6. Write a test case for a function that calculates the Fibonacci sequence up to n terms.
    7. Write a test case for a function that sorts a list of numbers.
    8. Write a test case for a function that converts a string to uppercase.
    9. Write a test case for a function that removes all vowels from a string.
    10. Write a test case for a function that returns the intersection of two lists.
- **Difficult**:
    1. Write a test case for a function that reads a file and returns its contents as a list of lines.
    2. Write a test case for a function that parses a JSON string and returns a dictionary.
    3. Write a test case for a function that connects to a database and returns a list of records.
    4. Write a test case for a function that fetches data from a URL and returns the response content.
    5. Write a test case for a function that performs a complex mathematical calculation and returns the result.

Notes

- Each problem list follows a theme that correlates with the title of the section, covering a variety of concepts and increasing in complexity.
- For each section, the problems are divided into easy, medium, and difficult categories to cater to different skill levels.
- The notes section for each topic outlines the key concepts that should be understood or learned while working through the problems.

Suggestions for Implementation

To implement this plan, you can start by creating separate files or notebooks for each topic. For example:

- `01_python_basics.py`
- `02_python_data_types.py`
- `03_python_strings.py`

Each file can then be divided into sections for easy, medium, and difficult problems. Alternatively, you can create a directory for each topic and have separate files for easy, medium, and difficult problems within those directories.

This structure will make it easy to navigate through the problems and focus on specific areas for practice. Additionally, providing solutions and explanations for each problem can be immensely helpful for learners to understand their mistakes and learn the correct approaches.

Then Start Projects:

Beginner Projects:

1. **Personal Expense Tracker:**
   - Create a simple command-line application that allows users to input their expenses and track their spending habits over time. You can store data in a text file or a simple SQLite database.
2. **To-Do List Application:**
   - Build a to-do list app that allows users to add, update, and delete tasks. You can add features like task prioritization, due dates, and task categorization.
3. **Basic Web Scraper:**
   - Write a script that scrapes a website for specific data, such as news headlines or product prices. Use libraries like `requests` and `BeautifulSoup`.
4. **Simple Calculator:**
   - Develop a command-line calculator that performs basic arithmetic operations. Expand it by adding more complex operations and a simple GUI using Tkinter.

Intermediate Projects:

1. **Blog Website:**

   - Build a blog website using Flask or Django. Implement features like user authentication, post creation, comments, and tagging.
2. **Weather App:**
   - Create a web app that fetches and displays weather information based on the user's location. Use an API like OpenWeatherMap for real-time data.
3. **Chat Application:**
   - Develop a simple chat application using sockets. Implement features like user registration, private messaging, and group chats.
4. **Data Visualization Dashboard:**
   - Create a dashboard that visualizes data from a CSV or API using libraries like Matplotlib or Plotly. The dashboard could include charts, graphs, and filters.

Advanced Projects:

1. **E-commerce Website:**
   - Build a fully functional e-commerce platform using Django or Flask. Implement features like user authentication, product listings, shopping carts, and payment processing.
2. **Machine Learning Project:**
   - Create a machine learning model using libraries like Scikit-learn or TensorFlow. You could work on a classification problem, such as sentiment analysis, or a regression problem, like predicting house prices.
3. **API Development and Integration:**
   - Develop a RESTful API using FastAPI or Django REST Framework. The API could handle CRUD operations for a resource like products or users. Then, build a frontend to consume this API.
4. **Task Automation System:**
   - Write a script or application that automates a common task, such as file organization, data entry, or backup management. Use Python libraries like `os` and `shutil`.
5. **Social Media Bot:**
   - Create a bot that interacts with a social media platform like Twitter or Instagram. The bot could automate tasks such as posting updates, following users, or liking posts based on certain criteria.
6. **Real-Time Chat Application:**
   - Build a real-time chat application with a backend server (using Flask or Django) and a frontend using WebSockets. Implement features like online/offline status, typing indicators, and message history.

Capstone Project:

1. **Portfolio Website:**
   - Build a personal portfolio website that showcases your projects, resume, and blog posts. Include a contact form and links to your social media profiles. Deploy the site using services like Heroku or AWS.
2. **Comprehensive Data Pipeline:**

    - Develop a data pipeline that collects, processes, and visualizes data from multiple sources (APIs, databases, web scraping). Use tools like Apache Airflow for workflow management and Pandas for data processing.
3. **Smart Home Automation:**
    - Create an IoT-based project that controls and monitors home devices using Python. You can use Raspberry Pi and integrate it with services like MQTT or Google Assistant.