Homeworks link	https://drive.google.com/drive/folders/1lxU6psZPj1Kcl2e0u9Kjr-LHdGfNUjou?usp=sharing
Week	Topics
1	Introduction to Programming and Python: - Language Introduction - Writing your first program: print("Hello, World!"). - User-defined Functions (Defining a function using def, passing information to functions using arguments, getting a result back using the return statement, calling a function) - Indentation and Comments - Python Style Guide (Variable Names) - Online help, help(), and dir() Project: Set up your development environment (installing Python and a code editor like PyCharm or VS Code). https://developers.google.com/edu/python/set-up PyCharm: https://www.jetbrains.com/pycharm/download/ VS Code: https://code.visualstudio.com/download
2	Data Types and Basic Operations: - Data types: Strings (text), Numbers (Integers (whole numbers), and Floats (decimal numbers)). - Variables (storing data like text and numbers). - Arithmetic operators (+, -, *, /, % for remainder), type conversion (e.g., turning a user's text input into a number). - Lists and Operations - Using input() to get user data and print() to display it. Project: Basic Tip Calculator. Write a program that asks for the total bill amount and the tip percentage. It should then calculate and display the tip amount and the total bill including the tip. Advanced: You can also ask for a name (using input()) and print the total amount with that name in it. Solve the problems in list1.py in the provided python_exercises folder that **do not use sorting**
3	Strings and Operations: - Basic String Operations - String Slices - String Formatting Project: Complete the string1.py exercise in the provided python_exercises folder. Try string_2 too, if you're curious.
4	Logic and Conditional Statements: - Introduction to Boolean values (True and False). - Using if, elif, and else statements to run different blocks of code based on a condition. - Comparison operators (==, !=, <, >) and logical operators (and, or, not). Project: Upgrade: Advanced Tip Calculator. Modify last week's Tip Calculator. Add a feature that asks how many people are splitting the bill and their names. The program should then calculate and display how much each person needs to pay. Add a condition: if the tip percentage is over 20%, print a "Thank you for your generosity!" message.
5	Repetitive Tasks with Loops: - For loop and using range() - While loop - Break and continue Project: Number Guessing Game. The program chooses a secret random number between 1 and 100. The user has a limited number of attempts to guess it. After each guess, the program tells the user if their guess was "too high," "too low," or "correct." Project: Simple To-Do List. Create a program that allows a user to manage a to-do list. The user should be able to: Add a new task. View all tasks. Delete a task by its name or number. The program should loop until the user decides to quit.
6	Dictionaries: - Creating dictionaries, accessing values using keys, adding new key-value pairs, and iterating over keys and values. Project: Simple Word Counter. Write a program that asks the user for a sentence. The program should then count the occurrences of each word in the sentence and display the result as a dictionary.
7	Sorting, Tuples, List Comprehensions: - List Comprehensions vs for loop - Tuples - in-built and custom sorting Project: - Solve the rest of the problems in list1.py that use sorting and tuples. - Advanced Word Counter. Combining all the basic Python material -- strings, lists, dicts, tuples, files -- try the summary wordcount.py exercise in the provided python_exercises folder
8	Files and Error Handling: - Reading data from a text file and writing data to a text file. - try...except blocks to gracefully handle potential errors Project: Modify your To-Do List program from earlier. When the program starts, it should read any existing tasks from a file named tasks.txt. When the user adds or deletes a task, the program should update the file. This way, the tasks are not lost when the program closes.
9	Modules and the Python Standard Library: - Using the import statement to bring in useful modules. - Exploring popular modules like random (for random numbers), math (for mathematical functions), urllib, and datetime (for working with dates and times). Project: Dice Rolling Simulator. Use the random module to create a function that simulates rolling two six-sided dice and returns their sum. Then, create a program that asks the user how many times to roll the dice and uses a dictionary to track and display the frequency of each sum.
10	Classes and Objects: - Defining a class as a blueprint. - Using the __init__ method to create an object (instance) with specific attributes (data) and methods (functions). Project: Simple Bank Account Class. Create a BankAccount class. Each account should have an owner name and a balance. The class should have methods to deposit() money, withdraw() money (don't allow withdrawal if funds are insufficient), and display_balance().
11	Algorithms and Complexity: - The idea of "time complexity" (how the runtime of an algorithm grows as the input size grows). - A high-level overview of Big O notation. - Introduction to a fundamental algorithm: Linear Search. Project: Linear Search Implementation. Write a function linear_search(data_list, target) that takes a list and a target value. It should loop through the list and return the index of the target if found, or -1 if not found. Test it with various lists and targets.
12	Leetcode: - Walkthrough of a classic beginner LeetCode problem: "Two Sum". - A structured approach to solving coding problems: Understand, Plan, Code, Test. - Discussing 2-3 common patterns. - The "Frequency Counter" pattern using dictionaries (as seen in the word counter and dice simulator homeworks). If there's time: The "Two Pointers" technique for iterating/searching in lists. Project: Solve 2 more easy Leetcode problems