Python-Foundation Repository:

This repository showcases a collection of projects that demonstrate my journey from learning Python through various tutorials to building semi-original applications on my own. Through foundational tutorials, I learned essential programming concepts such as user input, conditionals, loops, and the basics of object-oriented programming (OOP). As I progressed, I began creating more complex applications like Blackjack, incorporating classes and encapsulating game logic. My skills in user interaction and program flow were reinforced by projects such as a Computer Systems Quiz and a Timed Math Challenge. My proudest accomplishment is the independently developed Kindergarten to Grade 12 Math Challenge, where I applied a wide range of learned concepts to create a multi-level math quiz covering topics from basic arithmetic to advanced subjects like trigonometry. I also maintained consistency by building smaller projects, which helped solidify my understanding. This repository not only reflects the technical knowledge I gained but also showcases my growth in creating structured, logical, and engaging projects.

Project 1: Rock Paper Scissors:

This Python script is a simple "Rock, Paper, Scissors" game where the player inputs their choice, and the computer randomly selects one as well. The game then compares the choices using the `check_win()` function to determine if the player wins, loses, or ties based on the standard rules (rock beats scissors, paper beats rock, and scissors beat paper). Finally, it prints the result of the match. Moreover, In this project, I followed a Free Code Camp tutorial to create a simple Rock-Paper-Scissors game using Python. The game begins by importing the built-in random module, which provides a function to randomly select a choice for the computer. I defined two main functions: get_choices and check_win. The get_choices function prompts the player for an input using the input() function and stores the player's and computer's choices in a list and a dictionary, respectively. This demonstrated my understanding of taking user inputs, using lists, and organizing related data with dictionaries. The check_win function compares the player's and computer's choices using conditional statements (if, elif, and else), allowing me to practice working with multiple conditions to determine the game's outcome. I used f-strings to display the choices in a readable format and employed the return statement to provide the result of the game based on these comparisons. Finally, I executed the main logic by calling the functions and tying everything together. Through this project, I solidified my understanding of Python basics such as working with lists, dictionaries, functions, conditionals, user interaction, and randomness, all while creating an interactive game that responds dynamically based on player input.

Project 2: Choose Your Adventure Mini Game;

In this project, I followed a tutorial from Tech With Tim to create a simple text-based adventure game using Python. The game starts with a welcome message and prompts the player for their name and age using the input() function. I then converted the age input to an integer to allow for age-based checks later in the code. After greeting the player, the program initializes a variable health with a value of 10 to keep track of the player's health throughout the game. If the player's age is 18 or older, they are allowed to play, and a prompt appears to confirm if they want to continue. If the player chooses to play, they are reminded of their initial health and prompted to make their first choice between going left or right. If the player chooses to go left, they reach a lake and must decide whether to swim across or go around. Depending on their choice, the player might lose health. Next, the player faces another decision to

approach a house or a river. Choosing to enter the house leads to a health penalty if the owner is unfriendly. Throughout the game, health points are subtracted based on the player's decisions, and the game checks if the player's health drops to zero or below, indicating a loss. If the player survives these choices, they win the game. This project helped me solidify several programming concepts, such as using conditional statements (if, elif, and else) to control the game's flow based on user choices, handling user input effectively, and tracking and updating variables like health. I also learned how to structure a simple interactive game with multiple scenarios and outcomes, reinforcing the importance of logic in programming.

Project 3: Black jack;

In this project, I followed a Free Code Camp tutorial to build a Blackjack game in Python, which helped me strengthen my understanding of object-oriented programming (OOP) principles. The game is structured around four main classes: Card, Deck, Hand, and Game. The Card class represents individual cards, each defined by a suit and a rank, and includes a __str__ method to return a string representation of the card, like "Ace of Spades." The Deck class creates a full deck of cards using nested loops to combine suits (spades, clubs, hearts, diamonds) with various ranks. It includes methods to shuffle the deck using the random.shuffle() function and deal cards by popping them from the deck. The Hand class manages the cards held by either the player or the dealer. It features methods to add cards to the hand, calculate the hand's value while accounting for Aces, and check for a Blackjack. A notable method here is calculate_value(), which iterates over the cards in the hand and adjusts the value if the player has an Ace and the total value exceeds 21. Additionally, the class includes a display method to reveal the hand, hiding one of the dealer's cards unless all should be shown. The Game class orchestrates the overall flow, starting with a prompt for the number of rounds to be played. Within each round, it initializes a deck, shuffles it, and deals two cards each to the player and the dealer. The game loop includes user interactions where the player can choose to "Hit" (add a card) or "Stand" (end their turn). Depending on these choices, the game recalculates hand values and checks for winning conditions using the check_winner method. The method evaluates whether the player or dealer has busted, if either has a Blackjack, or if there's a tie. This project helped me apply OOP concepts like classes, methods, and encapsulation to a more complex structure. It also reinforced essential programming concepts like loops, conditionals, handling user input, and creating reusable code through functions and classes. By combining all these elements, I gained a deeper understanding of OOP principles while building an interactive and logical game application.

Project 4: Quiz Game;

In this project, I followed a Tech With Tim tutorial to create a simple Computer Systems Quiz using Python. The game begins with a welcome message and prompts the player to decide whether they want to participate. If the player chooses not to play, the program exits gracefully using the quit() function. If the player agrees, the quiz starts with an initial score set to zero. The game consists of a series of questions about computer systems, such as "What does CPU stand for?" and "What does RAM stand for?" For each question, the player's input is compared to the correct answer using a case-insensitive check (str().lower()). When the player answers correctly, a congratulatory message is displayed, and the score increases by one. If the player answers incorrectly, a message encourages them to check their spelling, creating a friendly and interactive experience. After the player answers all the questions, the

program displays the final score, showing how many questions were answered correctly out of the total six. This project helped me practice handling user input, using conditional statements to check for correct answers, and incrementing a variable to keep track of the score. It also reinforced the importance of clear program flow and user interaction to create an engaging experience.

Project 5: Timed Math Challenge;

In this project, I followed a Tech With Tim tutorial to create a Python program that generates random math problems for a timed challenge. The program starts by defining a list of mathematical operators (+, -, *, /) and a range for the operands (MIN_OPER and MAX_OPER). The generate_problem function selects random values for the left and right operands within the specified range and picks an operator from the list. It then combines these elements into an expression string and calculates the answer using Python's built-in eval() function, simplifying the code by avoiding multiple conditional statements. The main program begins by generating and displaying a sample problem, then waits for the user to press enter to start the timed challenge. It records the starting time using time.time(). The program then enters a loop that generates and displays ten random math problems, prompting the user for their answer. If the user's answer matches the calculated result,the loop moves to the next problem. If the answer is incorrect, it increments a counter for wrong answers. After all the problems are completed, the program calculates the total time taken by subtracting the starting time from the end time and rounding it to two decimal places. Finally, it displays the user's total time and congratulates them. This project reinforced my understanding of generating random values, using lists, handling user input, and working with Python's time module for creating timed challenges. It also taught me how to use eval() effectively while being cautious of its limitations and risks in more complex projects.

Project 6: Kindergarten to Grade 12 Math Challenge;

This code is a math challenge game designed to test users' math skills through four sections: Kindergarten, Elementary, Middle School, and High School. Each section generates random math problems based on the level of difficulty. In the Kindergarten section, simple addition problems are presented, while the Elementary section includes addition, subtraction, and multiplication. The Middle School section introduces more complex operations like square roots, exponentiation, and division. The High School section involves trigonometric functions, logarithms, and their reciprocals, all requiring input in radians, with answers rounded to two decimal places. For each problem, the user must input the correct answer to proceed, with incorrect answers prompting additional attempts. The game records the total time taken to complete all sections and displays it at the end. This challenge offers users a way to refresh and test their math skills progressively from basic arithmetic to advanced functions. This is my first jab at making a project for ME! Other projects in the foundation repository have been me learning from tutorials. This is the first project where I had to use my knowledge in python and what I learned from previous tutorials and make something semi-original. It was a lot of trial and error, but I'm happy I came through! Notice how this project has similar functions
to project 5, its just more applied (Note the other projects on this list will follow the same semi-original theme, applying my concepts from what i have learned from tutorials to make something semi-original).

Project 7: Random Coin Flipper;

This simple coin flip project took less than five minutes to complete and was a quick exercise to help me stay consistent and focused on improving my coding skills. I used Python's built-in module random to simulate a coin flip by randomly selecting between "heads" and "tails" when the user presses Enter. The project allowed me to practice basic syntax, functions, and user input while creating something small but functional. Uploading this project serves as a reminder to keep pushing toward more complex challenges and bigger goals as I continue learning and growing as a programmer.

Project 8: Random Coin Flipper Plotting;

I was not quite satisfied with the base "easiness" that I uploaded earlier. It was sad to see me go from a project like K-12 where I had to follow the usual procedure of programming: Research, Code, Debug, Edits, Repeat; to go to a simple one function program. I then decided to utilize my skills and learned the library of matplotlib which is straightforward to understand and execute as It's the computer doing the more tedious and strainful work of graphing the results. I applied this library to my coin flipper project which allows a user to select how many flips they want to generate and then see the probability displayed on a simple bar graph.

Project 9: In Python Terminal Calculator;

My friend challenged me to race him in making a calculator. While he focused on speed, I prioritized efficiency and simplicity in my design. Instead of using a series of if-else statements to handle different operations, I opted for a more streamlined approach. By leveraging Python's built-in eval() function, my calculator dynamically evaluates the user's input as an arithmetic expression. This method allows the program to interpret and execute basic arithmetic operations (+, -, *, /) with minimal code. Additionally, by validating the operator with a simple conditional check, I ensured that the calculator would only evaluate valid operations, preventing unnecessary complexity.

Project 10: Simple Contact Management System;

The Simple Contact Management System project is a culmination of all the fundamental Python concepts I have learned through previous tutorials. In this project, I independently developed a program that allows users to manage a list of contacts by adding, viewing, updating, deleting, searching, and saving contacts to a file. I applied my understanding of lists and dictionaries to organize contact information efficiently. By defining separate functions for each feature, I utilized my knowledge of modular programming to maintain a clean and organized code structure. I employed loops to navigate through the contact list, control statements to handle user choices, and input/output functions to interact with the user. To ensure data persistence, I implemented file handling techniques to save and load contact details to and from a text file. I also incorporated basic error handling to manage invalid inputs and file operations smoothly. This project not only helped me solidify my understanding of core Python syntax and functions but also allowed me to gain confidence in independently creating a fully functional application.