**AI ASSISTED CODING LAB 3.1**

**Hall ticket number : 2403A52373**

**Name :KODURU ESWAR REDDY**

**Prompt 01:** generate a Python code to find the compound interest using functions

**Code:**

**A computer screen shot of a computer program

AI-generated content may be incorrect.**

**Code explanation:**

* def compound\_interest(principal, rate, time): This line defines a function named compound\_interest that takes three arguments: principal, rate, and time.
* """ Calculates compound interest...""": This is a docstring, explaining what the function does, its arguments, and what it returns.
* amount = principal \* (1 + rate) \*\* time: This line calculates the total amount after applying compound interest using the formula: Amount = Principal \* (1 + Rate)^Time.
* interest = amount - principal: This line calculates the compound interest earned by subtracting the initial principal from the total amount.
* return interest: This line returns the calculated compound interest.
* principal = 1000, rate = 0.05, time = 10: These lines set the values for the principal, annual interest rate (5%), and time in years.
* interest\_earned = compound\_interest(principal, rate, time): This line calls the compound\_interest function with the specified values and stores the returned interest in the interest\_earned variable.
* print(f"Compound interest earned: ${interest\_earned:.2f}"): This line prints the calculated compound interest, formatted to two decimal places.

**Output:**

**A grey screen with white text

AI-generated content may be incorrect.**

**Prompt 02:** Generate a Python code to calculate the average, median, and mode of a list of numbers

**Code:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Code explanation:**

* import statistics: This line imports the statistics module, which provides functions for calculating various statistical properties of data.
* def analyze\_list(data):: This defines a function called analyze\_list that takes one argument, data, which is expected to be a list of numbers.
* average = statistics.mean(data): This line calculates the mean (average) of the numbers in the data list using the statistics.mean() function and stores it in the average variable.
* median = statistics.median(data): This line calculates the median (the middle value when the data is sorted) of the numbers in the data list using the statistics.median() function and stores it in the median variable.
* try...except statistics.StatisticsError:: This block handles potential errors. The statistics.mode() function raises a StatisticsError if there is no unique mode (e.g., if all numbers appear the same number of times).
* mode = statistics.mode(data): Inside the try block, this line attempts to calculate the mode (the most frequent number) of the numbers in the data list using the statistics.mode() function and stores it in the mode variable.
* mode = "No unique mode found": If a StatisticsError occurs (meaning there's no unique mode), the code inside the except block is executed, setting the mode variable to the string "No unique mode found".
* return { ... }: The function returns a dictionary containing the calculated "average", "median", and "mode".
* my\_list = [...]: This line creates a sample list of numbers called my\_list.
* analysis\_results = analyze\_list(my\_list): This line calls the analyze\_list function with my\_list as input and stores the returned dictionary in the analysis\_results variable.
* print(...): These lines print the results stored in the analysis\_results dictionary in a user-friendly format.

**Output:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**Prompt 03:** Generate a Python script to convert a number to a binary number by using functions

**Code:**

**A computer screen shot of a program

AI-generated content may be incorrect.**

* **Code explanation:**
* def decimal\_to\_binary(number):: This defines a function named decimal\_to\_binary that takes one argument, number, which is the decimal integer you want to convert.
* """Docstring""": The text within the triple quotes is a docstring, which explains what the function does, its arguments (Args), and what it returns (Returns).
* if number == 0:: This is a base case. If the input number is 0, its binary representation is simply "0", so the function immediately returns "0".
* binary = "": Initializes an empty string called binary. This string will be built up to store the binary digits.
* while number > 0:: This is the main loop that continues as long as the number is greater than 0. The process of converting to binary involves repeatedly dividing the number by 2.
* remainder = number % 2: Calculates the remainder when number is divided by 2. This remainder will be either 0 or 1, which are the binary digits.
* binary = str(remainder) + binary: Converts the remainder to a string and prepends it to the binary string. We prepend because the binary digits are generated in reverse order of their significance (least significant first).
* number = number // 2: Performs integer division of number by 2, updating the number for the next iteration of the loop.
* return binary: Once the while loop finishes (when number becomes 0), the function returns the final binary string.

The code then demonstrates how to use this function with two example numbers (25 and 10) and prints their binary representations.

**Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**Prompt 04:**

Create a user interface for a hotel to generate a bill based on customer requirements using Python

**Code:**

import tkinter as tk

from tkinter import messagebox

# Menu items and prices

MENU = {

    "Pizza": 200,

    "Burger": 120,

    "Pasta": 150,

    "Sandwich": 100,

    "French Fries": 80,

    "Grilled Chicken": 250,

    "Salad": 90,

    "Soup": 70,

    "Ice Cream": 60,

    "Coffee": 50,

    "Fresh Juices": 80,

    "Cakes & Pastries": 110,

    "Steak": 300,

    "Seafood": 350

}

class HotelBillApp:

    def \_\_init\_\_(self, root):

        self.root = root

        self.root.title("Hotel Bill Generator")

        self.entries = {}

        tk.Label(root, text="Select Items and Enter Quantity", font=("Arial", 14)).grid(row=0, column=0, columnspan=3, pady=10)

        for idx, (item, price) in enumerate(MENU.items(), start=1):

            tk.Label(root, text=f"{item} ({price} ₹)", anchor="w", width=20).grid(row=idx, column=0, padx=5, pady=2)

            entry = tk.Entry(root, width=5)

            entry.grid(row=idx, column=1, padx=5)

            self.entries[item] = entry

        tk.Button(root, text="Generate Bill", command=self.generate\_bill, bg="green", fg="white").grid(row=len(MENU)+1, column=0, columnspan=2, pady=10)

        self.bill\_text = tk.Text(root, height=12, width=40)

        self.bill\_text.grid(row=1, column=2, rowspan=len(MENU), padx=10)

    def generate\_bill(self):

        total = 0

        bill\_details = "Item\tQty\tPrice\n"

        bill\_details += "-"\*25 + "\n"

        for item, entry in self.entries.items():

            qty = entry.get()

            if qty.isdigit() and int(qty) > 0:

                price = MENU[item] \* int(qty)

                total += price

                bill\_details += f"{item}\t{qty}\t{price} ₹\n"

        bill\_details += "-"\*25 + f"\nTotal Bill: {total} ₹"

        self.bill\_text.delete(1.0, tk.END)

        self.bill\_text.insert(tk.END, bill\_details)

        if total == 0:

            messagebox.showinfo("No Items", "Please enter quantity for at least one item.")

if \_\_name\_\_ == "\_\_main\_\_":

    root = tk.Tk()

    app = HotelBillApp(root)

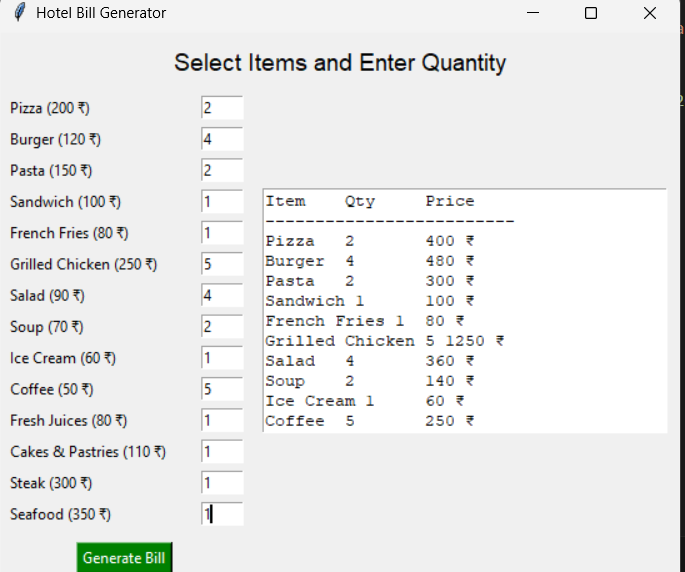
    root.mainloop()

**Code explanation:**

This Python code creates a simple hotel bill generator GUI using [tkinter](vscode-file://vscode-app/c:/Users/eswar/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o "). Here’s how it works:

* Imports:
  + [tkinter](vscode-file://vscode-app/c:/Users/eswar/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) for GUI components.
  + [messagebox](vscode-file://vscode-app/c:/Users/eswar/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) for popup messages.
* MENU Dictionary:
  + Contains menu items as keys and their prices as values.
* HotelBillApp Class:
  + [\_\_init\_\_](vscode-file://vscode-app/c:/Users/eswar/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) method:
    - Sets up the main window title.
    - Creates a label for instructions.
    - For each menu item, adds a label (with price) and an entry box for quantity.
    - Adds a "Generate Bill" button.
    - Adds a text area to display the bill.
  + [generate\_bill](vscode-file://vscode-app/c:/Users/eswar/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html) method:
    - Reads quantities entered by the user.
    - Calculates the price for each item (if quantity > 0).
    - Adds item details and total to the bill.
    - Displays the bill in the text area.
    - Shows a popup if no items are selected

**Output:**

****

**Prompt 05:**

Generate a Python script to convert the temperature

**Code:**

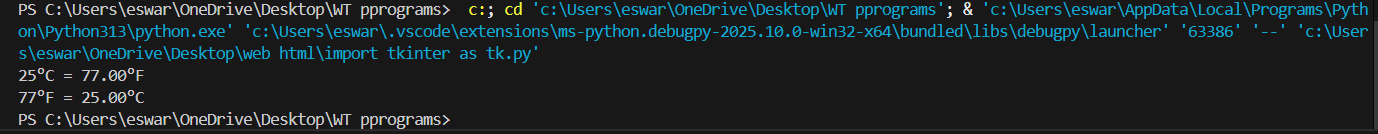
**A screen shot of a computer program

AI-generated content may be incorrect.**

**Code explanation:**

* Function [convert\_temp](vscode-file://vscode-app/c:/Users/eswar/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o "):
  + Takes a temperature value ([temp](vscode-file://vscode-app/c:/Users/eswar/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html)) and a boolean ([to\_fahrenheit](vscode-file://vscode-app/c:/Users/eswar/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ")).
  + If [to\_fahrenheit](vscode-file://vscode-app/c:/Users/eswar/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ") is True, it converts Celsius to Fahrenheit using the formula:  
    F = C \* 9/5 + 32
  + If [to\_fahrenheit](vscode-file://vscode-app/c:/Users/eswar/AppData/Local/Programs/Microsoft%20VS%20Code/resources/app/out/vs/code/electron-browser/workbench/workbench.html" \o ") is False, it converts Fahrenheit to Celsius using the formula:  
    C = (F - 32) \* 5/9
  + Returns the converted temperature.
* Example usage:
  + Converts 25°C to Fahrenheit and prints the result.
  + Converts 77°F to Celsius and prints the result.

**Output:**

****