

AI ASSISTED CODING

Assignment-3.2

N. Madhuvani

2303A52117

Batch-40

Task Description-1

- Progressive Prompting for Calculator Design: Ask the AI to design a simple calculator

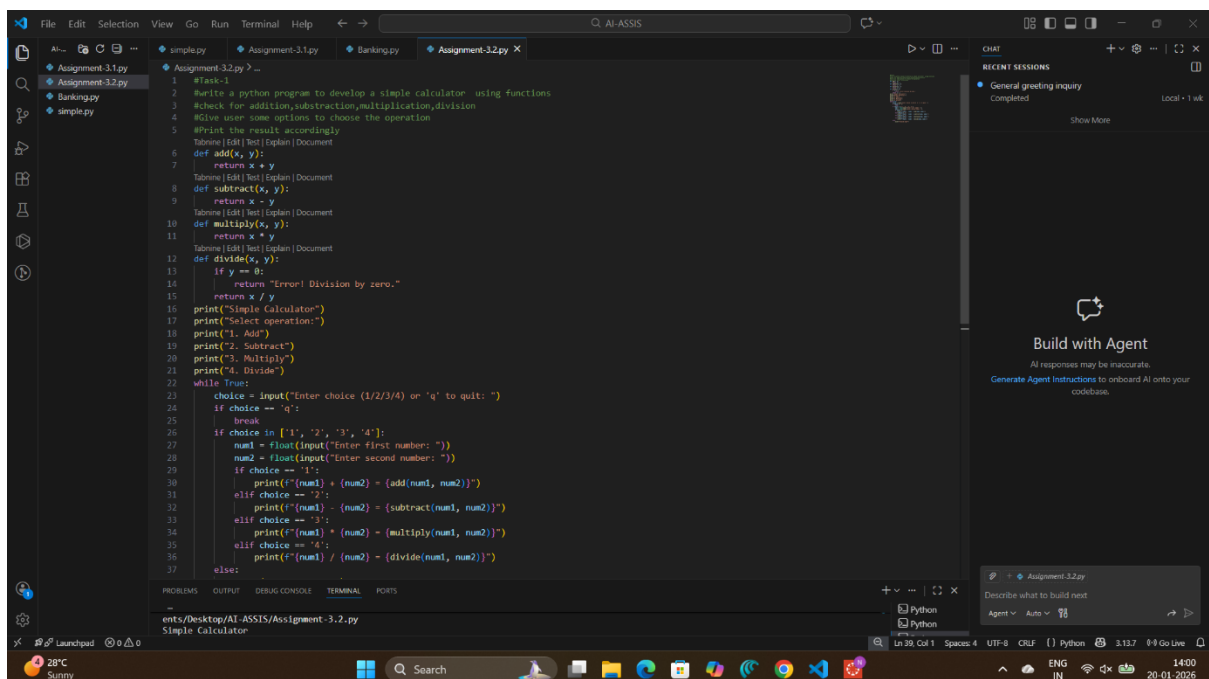
program by initially providing only the function name. Gradually enhance the prompt by

adding comments and usage examples.

Expected Output-1

- Comparison showing improvement in AI-generated calculator logic and structure.

Code and Output:



The screenshot displays a Visual Studio Code editor with a Python file named 'Assignment-3.2.py'. The code implements a simple calculator with functions for addition, subtraction, multiplication, and division, along with a main loop for user interaction. A chat window on the right side of the editor shows a conversation with an AI agent, where the user has prompted the AI to 'Build with Agent' and the AI has responded with a message about generating agent instructions.

```
1 #Task-1
2 Write a python program to develop a simple calculator using functions
3 Check for addition, subtraction, multiplication, division
4 Give user some options to choose the operation
5 #Print the result accordingly
6 Tabnine [Edi] Test | Explain | Document
7 def add(x, y):
8     return x + y
9 Tabnine [Edi] Test | Explain | Document
10 def subtract(x, y):
11     return x - y
12 Tabnine [Edi] Test | Explain | Document
13 def multiply(x, y):
14     return x * y
15 Tabnine [Edi] Test | Explain | Document
16 def divide(x, y):
17     if y == 0:
18         return "Error! Division by zero."
19     return x / y
20 print("Simple Calculator")
21 print("Select operation:")
22 print("1. Add")
23 print("2. Subtract")
24 print("3. Multiply")
25 print("4. Divide")
26 while True:
27     choice = input("Enter choice (1/2/3/4) or 'q' to quit: ")
28     if choice == 'q':
29         break
30     if choice in ['1', '2', '3', '4']:
31         num1 = float(input("Enter first number: "))
32         num2 = float(input("Enter second number: "))
33         if choice == '1':
34             print(f"{num1} + {num2} = {add(num1, num2)}")
35         elif choice == '2':
36             print(f"{num1} - {num2} = {subtract(num1, num2)}")
37         elif choice == '3':
38             print(f"{num1} * {num2} = {multiply(num1, num2)}")
39         elif choice == '4':
40             print(f"{num1} / {num2} = {divide(num1, num2)}")
41     else:
42         print("Invalid choice")
```

Chat window content:

RECENT SESSIONS

- General greeting inquiry Completed Local • 1 wk Show More

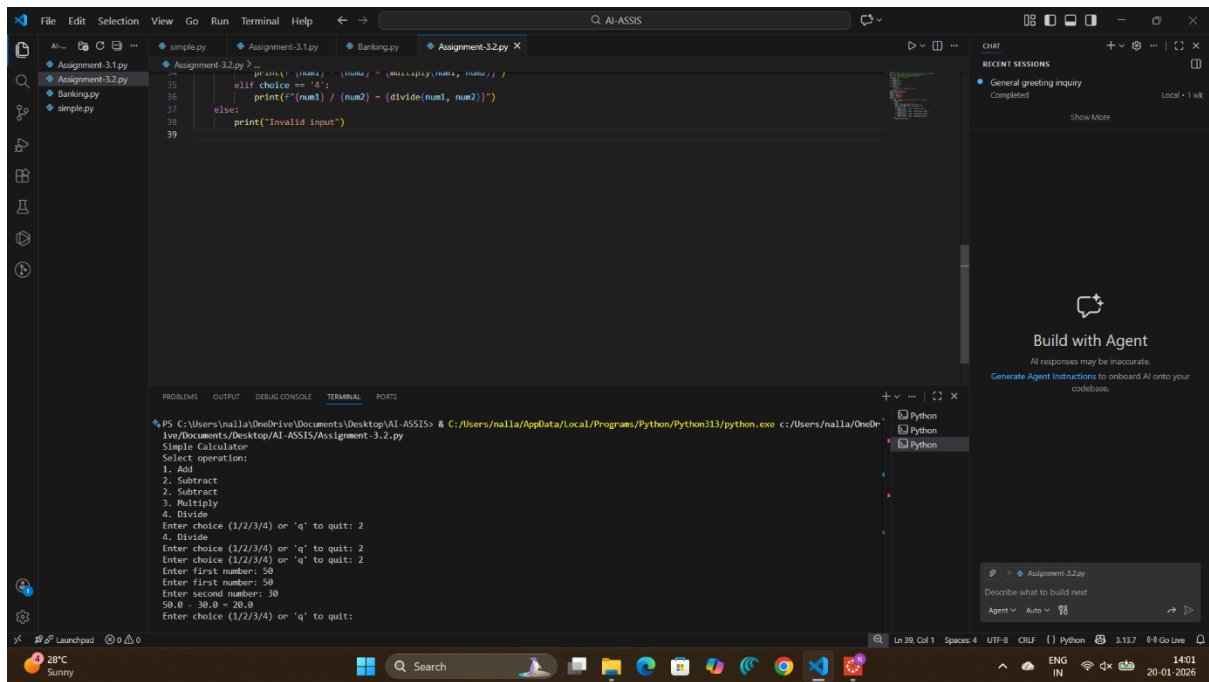
Build with Agent

AI responses may be inaccurate.

Generate Agent instructions to onboard AI onto your codebase.

Describe what to build next

Agent Auto ENG 3.13.7 1400 20-01-2026



Analysis:

Analysis: Progressive Prompting for Calculator Design

Progressive prompting means giving instructions step by step to the AI.

First, only the function name is given, so the output is basic.

Next, comments are added to explain what the calculator should do.

Then, usage examples are included to show how the program works.

Each step improves clarity and accuracy of the AI response.

This method helps in getting a complete and correct calculator program.

Task Description-2

- Refining Prompts for Sorting Logic: Start with a vague prompt for sorting student marks,

then refine it to clearly specify sorting order and constraints.

Expected Output-2

- AI-generated sorting function evolves from ambiguous logic to an accurate and efficient

implementation.

Code and Output:

```
40 #Task-2
41 #Write a python program to sort the students marks print after soring
42 #The code should have dynamic input from the user
43 #Constraints:Ensure all marks are between 0 and 100
44 #Print the output after sorting
45 #Tabnine | Edit | Test | Explain | Document
46 def sort_marks(marks_list):
47     return sorted(marks_list)
48
49 user_input = input("Enter student marks separated by spaces: ")
50 marks_list = list(map(int, user_input.split()))
51 sorted_marks = sort_marks(marks_list)
52 print("Sorted student marks:", sorted_marks)
53
54 for mark in marks_list:
55     if mark < 0 or mark > 100:
56         print(f"Invalid mark found: {mark}. Marks should be between 0 and 100.")
57         break
58     else:
59         print("All marks are valid.")
```

Terminal Output:

```
PS C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS> & c:/Users/nalla/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nalla/OneDrive/Documents/Desktop/AI-ASSIS/Assignment-3.2.py
Enter student marks separated by spaces: 98 45 76 89 12 90 100 56
Sorted student marks: [12, 45, 56, 76, 89, 90, 98, 100]
All marks are valid.
PS C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS> & c:/Users/nalla/AppData/Local/Programs/Python/Python313/python.exe c:/Users/nalla/OneDrive/Documents/Desktop/AI-ASSIS/Assignment-3.2.py
Enter student marks separated by spaces: 99 100 56 76 89 54 34 77 49
Sorted student marks: [34, 49, 54, 56, 76, 77, 89, 99, 100]
All marks are valid.
PS C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS>
```

Analysis:

Refining Prompts for Sorting Logic

Initially, a vague prompt is given to sort student marks.

Due to less clarity, the AI may give an incomplete or unclear solution.

The prompt is then refined to specify sorting order (ascending or descending).

Additional constraints (such as valid marks or number of students) are added.

Clear instructions help the AI produce accurate and expected output.

Refining prompts reduces confusion and errors in the solution.

Task Description-3

- Few-Shot Prompting for Prime Number Validation: Provide multiple input-output

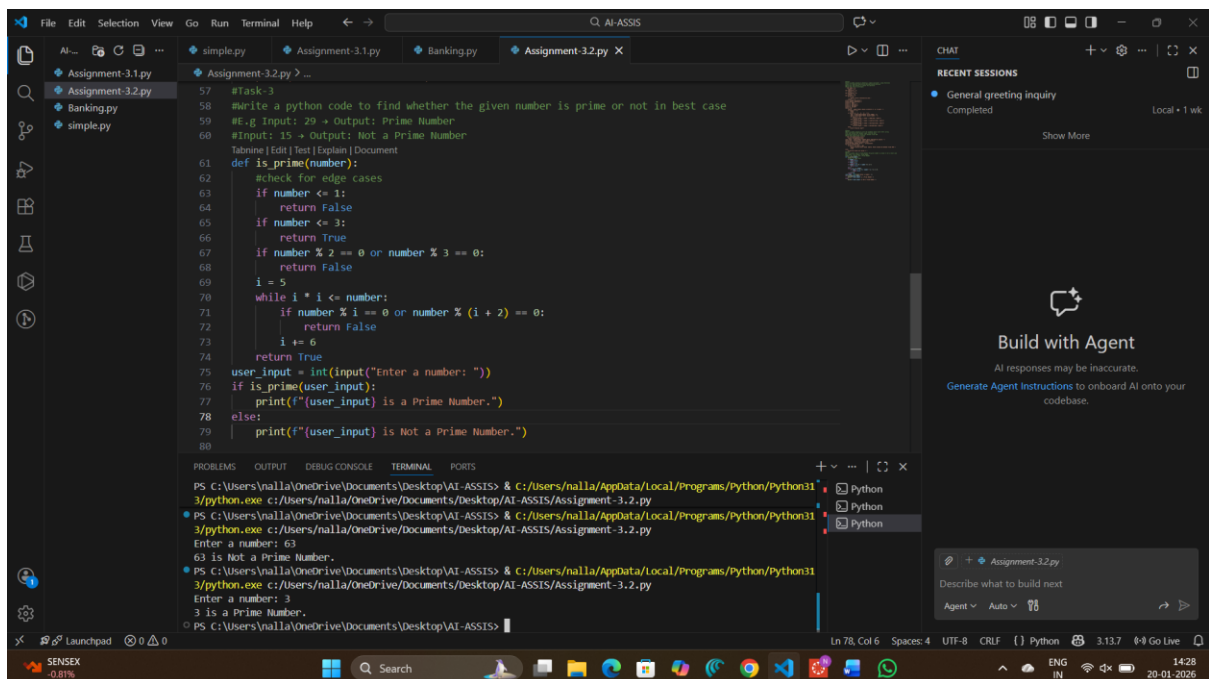
examples for a function that checks whether a number is prime. Observe how few-shot

prompting improves correctness.

Expected Output-3

- Improved prime-checking function with better edge-case handling.

Code and Output:



The screenshot shows a VS Code editor with a Python file named 'Assignment-3.2.py'. The code is a function to check if a number is prime. It includes comments and a docstring. The code is as follows:

```
#Task-3
#Write a python code to find whether the given number is prime or not in best case
#E.g Input: 29 → Output: Prime Number
#Input: 15 → Output: Not a Prime Number
#Title: [Edit] [Test] [Explain] [Document]
def is_prime(number):
    #check for edge cases
    if number <= 1:
        return False
    if number <= 3:
        return True
    if number % 2 == 0 or number % 3 == 0:
        return False
    i = 5
    while i * i <= number:
        if number % i == 0 or number % (i + 2) == 0:
            return False
        i += 6
    return True
user_input = int(input("Enter a number: "))
if is_prime(user_input):
    print(f"{user_input} is a Prime Number.")
else:
    print(f"{user_input} is Not a Prime Number.")
```

The terminal output shows the execution of the script:

```
PS C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS> & c:\Users\nalla\AppData\Local\Programs\Python\Python311\python.exe c:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS\Assignment-3.2.py
3/python.exe c:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS> & c:\Users\nalla\AppData\Local\Programs\Python\Python311\python.exe c:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS\Assignment-3.2.py
Enter a number: 63
63 is Not a Prime Number.
PS C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS> & c:\Users\nalla\AppData\Local\Programs\Python\Python311\python.exe c:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS\Assignment-3.2.py
Enter a number: 3
3 is a Prime Number.
PS C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS>
```

Analysis:

Few-Shot Prompting for Prime Number Validation

Few-shot prompting means giving multiple input-output examples to the AI.

The examples show how prime and non-prime numbers behave.

This helps the AI understand the exact logic of prime number checking.

The AI produces more correct and reliable results.

Errors are reduced compared to a prompt with no examples.

Few-shot prompting improves accuracy and consistency of the function.

Task Description-4

- Prompt-Guided UI Design for Student Grading System: Create a user interface for a

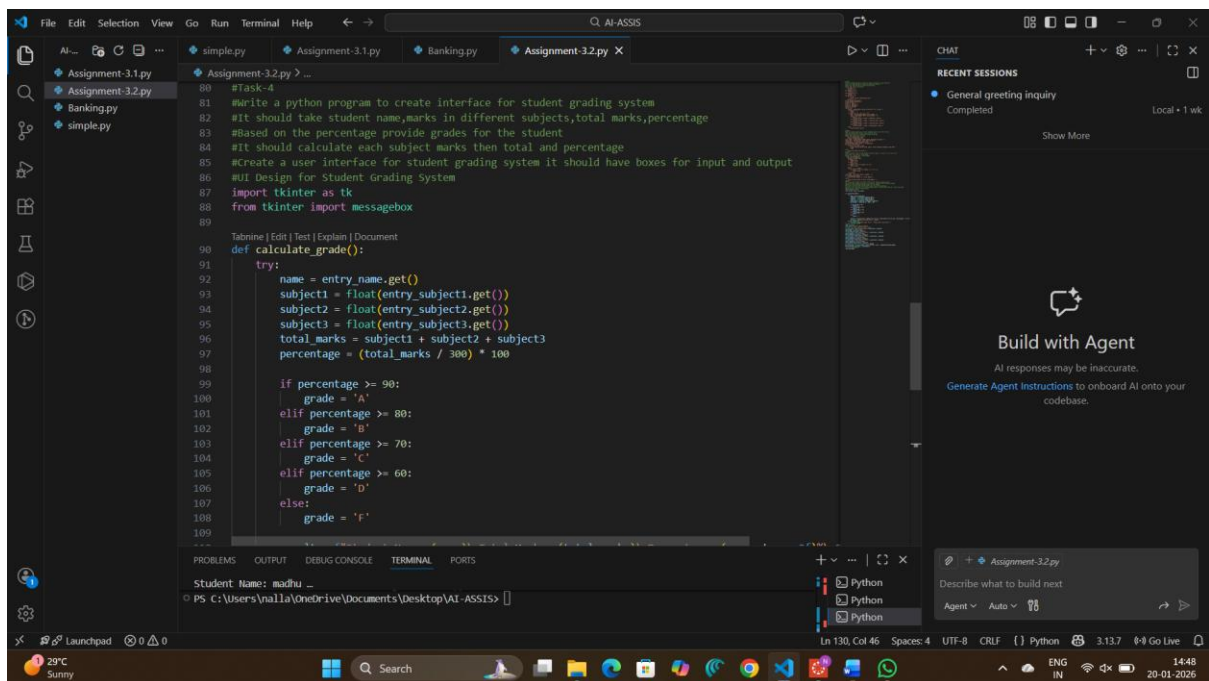
student grading system that calculates total marks, percentage, and grade based on user

input.

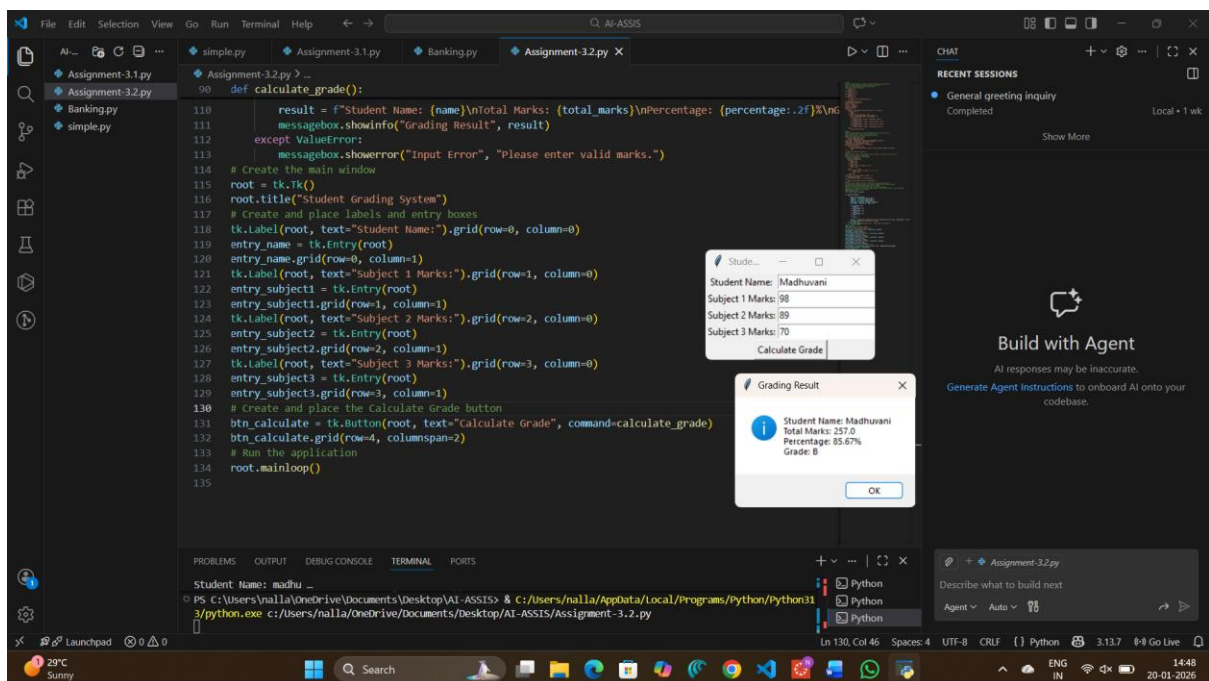
Expected Output-4

- Well-structured UI code with accurate calculations and clear output display.

Code and Output:



```
80 #Task-4
81 write a python program to create interface for student grading system
82 #it should take student name,marks in different subjects,total marks,percentage
83 #based on the percentage provide grades for the student
84 #it should calculate each subject marks then total and percentage
85 #create a user interface for student grading system it should have boxes for input and output
86 #UI Design for Student Grading System
87 import tkinter as tk
88 from tkinter import messagebox
89
90 Tabnine | Edit | Test | Explain | Document
91 def calculate_grade():
92     try:
93         name = entry_name.get()
94         subject1 = float(entry_subject1.get())
95         subject2 = float(entry_subject2.get())
96         subject3 = float(entry_subject3.get())
97         total_marks = subject1 + subject2 + subject3
98         percentage = (total_marks / 300) * 100
99
100         if percentage >= 90:
101             grade = 'A'
102         elif percentage >= 80:
103             grade = 'B'
104         elif percentage >= 70:
105             grade = 'C'
106         elif percentage >= 60:
107             grade = 'D'
108         else:
109             grade = 'F'
110     except ValueError:
111         messagebox.showerror("Input Error", "Please enter valid marks.")
112
113 # Create the main window
114 root = tk.Tk()
115 root.title("Student Grading System")
116 # Create and place labels and entry boxes
117 tk.Label(root, text="Student Name:").grid(row=0, column=0)
118 entry_name = tk.Entry(root)
119 entry_name.grid(row=0, column=1)
120 tk.Label(root, text="Subject 1 Marks:").grid(row=1, column=0)
121 entry_subject1 = tk.Entry(root)
122 entry_subject1.grid(row=1, column=1)
123 tk.Label(root, text="Subject 2 Marks:").grid(row=2, column=0)
124 entry_subject2 = tk.Entry(root)
125 entry_subject2.grid(row=2, column=1)
126 tk.Label(root, text="Subject 3 Marks:").grid(row=3, column=0)
127 entry_subject3 = tk.Entry(root)
128 entry_subject3.grid(row=3, column=1)
129 # Create and place the Calculate Grade button
130 btn_calculate = tk.Button(root, text="Calculate Grade", command=calculate_grade)
131 btn_calculate.grid(row=4, columnspan=2)
132 # Run the application
133 root.mainloop()
134
135
```



```
90 def calculate_grade():
110     result = f"Student Name: {name}\nTotal Marks: {total_marks}\nPercentage: {percentage:.2f}%\nGrade: {grade}"
111     messagebox.showinfo("Grading Result", result)
112 except ValueError:
113     messagebox.showerror("Input Error", "Please enter valid marks.")
114 # Create the main window
115 root = tk.Tk()
116 root.title("Student Grading System")
117 # Create and place labels and entry boxes
118 tk.Label(root, text="Student Name:").grid(row=0, column=0)
119 entry_name = tk.Entry(root)
120 entry_name.grid(row=0, column=1)
121 tk.Label(root, text="Subject 1 Marks:").grid(row=1, column=0)
122 entry_subject1 = tk.Entry(root)
123 entry_subject1.grid(row=1, column=1)
124 tk.Label(root, text="Subject 2 Marks:").grid(row=2, column=0)
125 entry_subject2 = tk.Entry(root)
126 entry_subject2.grid(row=2, column=1)
127 tk.Label(root, text="Subject 3 Marks:").grid(row=3, column=0)
128 entry_subject3 = tk.Entry(root)
129 entry_subject3.grid(row=3, column=1)
130 # Create and place the Calculate Grade button
131 btn_calculate = tk.Button(root, text="Calculate Grade", command=calculate_grade)
132 btn_calculate.grid(row=4, columnspan=2)
133 # Run the application
134 root.mainloop()
135
```

Analysis:

Prompt-Guided UI Design for Student Grading System

Prompt-guided design gives clear instructions for UI creation.

The prompt specifies inputs like student marks.

The system calculates total marks and percentage automatically.

It assigns grades based on rules given in the prompt.

Clear prompts help design a user-friendly interface.

Prompt guidance reduces errors and confusion in UI behavior.

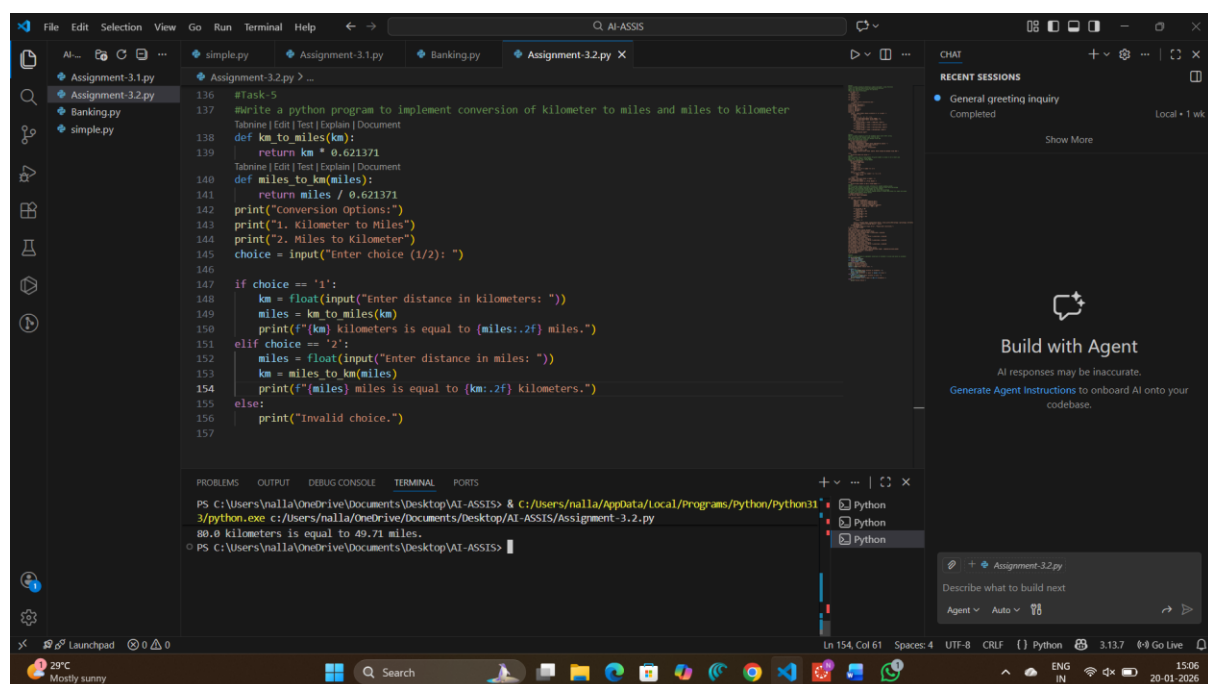
Task Description-5

- Analyzing Prompt Specificity in Unit Conversion Functions: Improving a Unit Conversion Function (Kilometers to Miles and Miles to Kilometers) Using Clear Instructions.

Expected Output-5

- Analysis of code quality and accuracy differences across multiple prompt variations.

Code and Output:



The screenshot displays a Visual Studio Code editor window with a Python file named 'Assignment-3.2.py'. The code implements a unit conversion function that takes a choice (1 for km to miles, 2 for miles to km) and a distance value, then returns the converted distance. The terminal output shows the program running successfully, converting 80.0 kilometers to 49.71 miles.

```
136 #Task-5
137 write a python program to implement conversion of kilometer to miles and miles to kilometer
138 def km_to_miles(km):
139     return km * 0.621371
140 def miles_to_km(miles):
141     return miles / 0.621371
142 print("Conversion Options:")
143 print("1. Kilometer to Miles")
144 print("2. Miles to Kilometer")
145 choice = input("Enter choice (1/2): ")
146
147 if choice == '1':
148     km = float(input("Enter distance in kilometers: "))
149     miles = km_to_miles(km)
150     print(f"{km} kilometers is equal to {miles:.2f} miles.")
151 elif choice == '2':
152     miles = float(input("Enter distance in miles: "))
153     km = miles_to_km(miles)
154     print(f"{miles} miles is equal to {km:.2f} kilometers.")
155 else:
156     print("Invalid choice.")
157
```

Terminal Output:

```
PS C:\Users\nalla\OneDrive\Documents\Desktop\AI-ASSIS> & C:/Users/nalla/AppData/Local/Programs/Python/Python311/python.exe c:/Users/nalla/OneDrive/Documents/Desktop/AI-ASSIS/Assignment-3.2.py
80.0 kilometers is equal to 49.71 miles.
```

Analysis:

Prompt Specificity in Unit Conversion Functions

Prompt specificity means giving clear and detailed instructions.

Initially, a vague prompt may cause incorrect or incomplete conversion.

Clear instructions specify conversion type (km to miles, miles to km).

The prompt defines input and output format clearly.

Specific prompts improve accuracy and correctness of the function.

This reduces errors and misunderstanding in unit conversion.