

# AI ASSISTANT CODING

## ASSIGNMENT-3.2

### 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.

### PROMPT:

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.

**This function performs basic arithmetic operations**

**a and b are numbers**

**operator can be +, -, \*, /**

**Usage Examples:**

**Example:**

**Input: a = 7, b = 3, operator = '+'**

**Output: 10**

**Input: a = 9, b = 3, operator = '/'**

**Output: 3.0**

## CODE:

```
calculator.py
calculator.py > calculator
1
2 def calculator(a, b, operator):
3     if operator == '+':
4         return a + b
5     if operator == '-':
6         return a - b
7     if operator == '*':
8         return a * b
9     if operator == '/':
10        if b == 0:
11            raise ZeroDivisionError('division by zero')
12        return a / b
13    raise ValueError(f"Invalid operator: {operator}")
14
15 if __name__ == "__main__":
16     try:
17         a = float(input("Enter a: "))
18         b = float(input("Enter b: "))
19         operator = input("Enter operator (+, -, *, /): ").strip()
20         result = calculator(a, b, operator)
21     except ValueError as e:
22         print("Error:", e)
23     except ZeroDivisionError as e:
24         print("Error:", e)
25     else:
26         print("Result:", result)
27
```

## OUTPUT:

```
PS C:\Users\mahee\OneDrive\Documents\Ai_coding> & C:/Users/mahee/OneDrive/Documents/Ai_coding/calculator.py
Enter a: 2
Enter b: 3
Enter operator (+, -, *, /): +
Result: 5.0
PS C:\Users\mahee\OneDrive\Documents\Ai_coding> & C:/Users/mahee/OneDrive/Documents/Ai_coding/calculator.py
Enter a: 2
Enter b: 3
Enter operator (+, -, *, /): -
Result: -1.0
PS C:\Users\mahee\OneDrive\Documents\Ai_coding> & C:/Users/mahee/OneDrive/Documents/Ai_coding/calculator.py
Enter a: 2
Enter b: 3
Enter operator (+, -, *, /): *
Result: 6.0
PS C:\Users\mahee\OneDrive\Documents\Ai_coding> & C:/Users/mahee/OneDrive/Documents/Ai_coding/calculator.py
Enter a: 2
Enter b: 3
Enter operator (+, -, *, /): /
Result: 0.6666666666666666
```

## 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.

## PROMPT:

Start with a vague prompt for sorting student marks, then refine it to clearly specify sorting order and constraints. and sorting function evolves from ambiguous logic to an accurate and efficient implementation

## CODE:

```
"""Simple integer array sorter.

Provides `sort_array(arr)` which returns a new list with the integers
sorted in ascending order.
"""

from typing import List

def sort_array(arr: List[int]) -> List[int]:
    """Return a new list containing the elements of `arr` sorted ascending.

    Example:
    | sort_array([3, 1, 2]) -> [1, 2, 3]
    """
    return sorted(arr)

if __name__ == '__main__':
    print(sort_array([3, 1, 4, 1, 5, 9, 2, 6]))
```

## OUTPUT:

```
PS C:\Users\mahee\OneDrive\Documents\Ai_coding> & C:/U
n/Python313/python.exe c:/Users/mahee/OneDrive/Documen
[1, 1, 2, 3, 4, 5, 6, 9]
PS C:\Users\mahee\OneDrive\Documents\Ai_coding> █
```

### 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.

## PROMPT:

```
You are given an integer n.
Write a function is_prime(n) that returns True if the number is prime
and False otherwise.

Consider prime-checking function with better edge-case handling

Examples (Few-Shot):

Input: n = 2
Output: True

Input: n = 7
Output: True

Input: n = 10
Output: False

Input: n = 1
Output: False

Input: n = 0
Output: False
```

## CODE:

```
def is_prime(n):
    # Reject booleans (bool is subclass of int in Python)
    if isinstance(n, bool):
        return False

    # Accept integer-valued floats, else require int
    if not isinstance(n, int):
        if isinstance(n, float) and n.is_integer():
            n = int(n)
        else:
            raise TypeError("n must be an integer (or integer-valued float)")

    if n < 2:
        return False
    if n in (2, 3):
        return True
    if n % 2 == 0 or n % 3 == 0:
        return False

    i = 5
    # check divisors of form 6k-1 and 6k+1
    while i * i <= n:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True

if __name__ == '__main__':
    tests = [2, 7, 10, 1, 0, -3, 17, 25, 97]
    for t in tests:
        print(t, is_prime(t))
```

## OUTPUT:

```
PS C:\Users\mahee\OneDrive\Documents\Ai_coding> & C:\Python39\python.exe c:/Users/mahee/OneDrive/Documents/Ai_coding/is_prime.py
2 True
7 True
10 False
1 False
0 False
-3 False
17 True
25 False
97 True
```

## 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.

## PROMPT:

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. and also UI code with accurate calculations and clear output display.

## CODE:

```
studentUi.PY
studentUi.PY > ...
1 import tkinter as tk
2 from tkinter import messagebox
3 def calculate():
4     try:
5         marks = [float(e.get()) for e in entries]
6     except ValueError:
7         messagebox.showerror("Input error", "Please enter numeric marks for all subjects.")
8         return
9
10    # Validate ranges
11    for m in marks:
12        if m < 0 or m > 100:
13            messagebox.showerror("Input error", "Marks must be between 0 and 100.")
14            return
15
16    total = sum(marks)
17    max_total = len(marks) * 100
18    percentage = (total / max_total) * 100
19
20    if percentage >= 90:
21        grade = 'A+'
22    elif percentage >= 80:
23        grade = 'A'
24    elif percentage >= 70:
25        grade = 'B'
26    elif percentage >= 60:
27        grade = 'C'
28    elif percentage >= 50:
29        grade = 'D'
30    else:
31        grade = 'F'
32
33    result_var.set(f"Total: {total:.0f} / {max_total}    Percentage: {percentage:.2f}%    Grade: {grade}")
34
35
36 def clear():
37     name_var.set("")
```

Ln 2, Col 31 Spaces: 4 UTF-8 CRLF { } Python

```

        result_var.set(f"Total: {total:.0f} / {max_total}    Percentage: {percentage:.2f}%    Grade: {grade}")

def clear():
    name_var.set("")
    for e in entries:
        e.delete(0, tk.END)
    result_var.set("")

root = tk.Tk()
root.title("Student Grading System")

tk.Label(root, text="Student Name:").grid(row=0, column=0, sticky='w', padx=6, pady=6)
name_var = tk.StringVar()
tk.Entry(root, textvariable=name_var, width=30).grid(row=0, column=1, columnspan=3, padx=6, pady=6)

# Create entries for 5 subjects (simple integer list array use-case)
entries = []
for i in range(5):
    tk.Label(root, text=f"Subject {i+1}:").grid(row=i+1, column=0, sticky='w', padx=6, pady=4)
    e = tk.Entry(root, width=10)
    e.grid(row=i+1, column=1, padx=6, pady=4)
    entries.append(e)

btn_calc = tk.Button(root, text="Calculate", command=calculate, width=12)
btn_calc.grid(row=1, column=2, rowspan=2, padx=6)

btn_clear = tk.Button(root, text="Clear", command=clear, width=12)
btn_clear.grid(row=3, column=2, rowspan=2, padx=6)

result_var = tk.StringVar()
tk.Label(root, textvariable=result_var, fg='blue').grid(row=7, column=0, columnspan=4, padx=6, pady=12)

root.mainloop()

```

## OUTPUT:

The screenshot displays a Python IDE with the 'calculate' function defined. The function takes marks for five subjects, calculates the total, maximum total, and percentage, and then determines the grade based on the percentage. A window titled 'Student Grading System' is shown in the foreground, featuring input fields for the student's name and five subjects, along with 'Calculate' and 'Clear' buttons.

```

studentUi.PY > calculate
def calculate():
    marks = [float(e.get()) for e in entries]
    except ValueError:
        messagebox.showerror("Input error", "Please enter numeric marks for a")
        return

    # Validate ranges
    for m in marks:
        if m < 0 or m > 100:
            messagebox.showerror("Input error", "Marks must be between 0 and 100")
            return

    total = sum(marks)
    max_total = len(marks) * 100
    percentage = (total / max_total) * 100

    if percentage >= 90:
        grade = 'A+'
    elif percentage >= 80:
        grade = 'A'
    elif percentage >= 70:
        grade = 'B'
    elif percentage >= 60:
        grade = 'C'
    elif percentage >= 50:
        grade = 'D'
    else:
        grade = 'F'

    result_var.set(f"Total: {total:.0f} / {max_total}    Percentage: {percentage:.2f}%    Grade: {grade}")

def clear():
    name_var.set("")
    for e in entries:
        e.delete(0, tk.END)

```

Student Grading System

Student Name:

Subject 1:

Subject 2:

Subject 3:

Subject 4:

Subject 5:

Calculate

Clear

```
ent01.PY > calculate
def calculate():
    marks = [float(e.get()) for e in entries]
    except ValueError:
        messagebox.showerror("Input error", "Please enter numeric marks for a")
        return

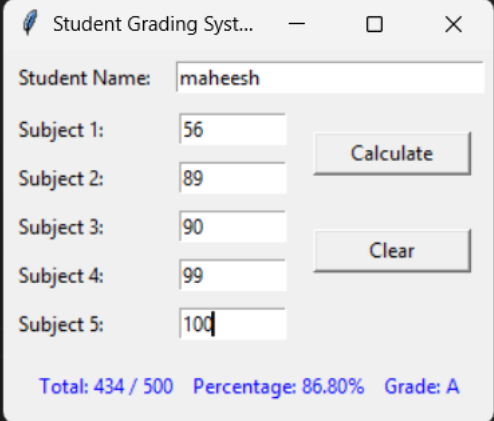
    # Validate ranges
    for m in marks:
        if m < 0 or m > 100:
            messagebox.showerror("Input error", "Marks must be between 0 and 100")
            return

    total = sum(marks)
    max_total = len(marks) * 100
    percentage = (total / max_total) * 100

    if percentage >= 90:
        grade = 'A+'
    elif percentage >= 80:
        grade = 'A'
    elif percentage >= 70:
        grade = 'B'
    elif percentage >= 60:
        grade = 'C'
    elif percentage >= 50:
        grade = 'D'
    else:
        grade = 'F'

    result_var.set(f"Total: {total:.0f} / {max_total}    Percentage: {percentage:.1f}%    Grade: {grade}")

def clear():
    name_var.set("")
    for e in entries:
        e.delete(0, tk.END)
```



### 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.

### PROMPT:

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



## CODE:

```
MILES_PER_KM = 1.0 / KM_PER_MILE
def _to_numeric(value, name="value"):
    if isinstance(value, bool):
        raise TypeError(f"{name} must be a numeric type, not bool")
    try:
        return float(value)
    except Exception:
        raise TypeError(f"{name} must be numeric (int/float or numeric-string)")

def km_to_miles(km, precision=None):
    val = _to_numeric(km, "km")
    miles = val * MILES_PER_KM
    return round(miles, precision) if precision is not None else miles

def miles_to_km(miles, precision=None):
    val = _to_numeric(miles, "miles")
    km = val * KM_PER_MILE
    return round(km, precision) if precision is not None else km

if __name__ == '__main__':
    # Simple demo and a few sanity checks.
    examples = [0, 1, 5, 42]
    print("Kilometres -> Miles")
    for k in examples:
        print(f"{k} km = {km_to_miles(k, precision=6)} miles")

    print('\nMiles -> Kilometres')
    for m in examples:
        print(f"{m} miles = {miles_to_km(m, precision=6)} km")
```

## OUTPUT:

```
PS C:\Users\mahee\OneDrive\Documents\Ai_coding> & C:/Users/mahee/OneDrive\Documents\Ai_coding\Python313/python.exe c:/Users/mahee/OneDrive\Documents\Ai_coding\km_to_miles.py
Kilometres -> Miles
0 km = 0.0 miles
1 km = 0.621371 miles
5 km = 3.106856 miles
42 km = 26.09759 miles

Miles -> Kilometres
0 miles = 0.0 km
1 miles = 1.609344 km
5 miles = 8.04672 km
42 miles = 67.592448 km
PS C:\Users\mahee\OneDrive\Documents\Ai_coding> 
```