

LAB ASSIGNMENT - 3

G. Yeshashwini

2303A52023

Batch-38

Question

Lab 3: Prompt Engineering – Improving Prompts and Context Management

Lab Objectives:

- To understand how prompt structure and wording influence AI-generated code.
- To explore how context (like comments and function names) helps AI generate relevant output.
- To evaluate the quality and accuracy of code based on prompt clarity.
- To develop effective prompting strategies for AI-assisted programming.

Lab Outcomes (LOs):

After completing this lab, students will be able to:

- Generate Python code using Google Gemini in Google Colab.
- Analyze the effectiveness of code explanations and suggestions by Gemini.
- Set up and use Cursor AI for AI-powered coding assistance.
- Evaluate and refactor code using Cursor AI features.
- Compare AI tool behavior and code quality across different platforms.

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:

design a simple calculator that can perform addition, subtraction, multiplication, and division operations. The calculator should take two numbers and an operator as input from the user and display the result.

CODE:

```

Function Calculator.py X
C: > Users > yesha > OneDrive > Desktop > Function Calculator.py > ...
1  #Design a simple calculator that can perform addition, subtraction, multiplication, and division operations.
2  def calculator():
3      num1 = float(input("Enter first number: "))
4      operator = input("Enter operator (+, -, *, /): ")
5      num2 = float(input("Enter second number: "))
6      if operator == '+':
7          result = num1 + num2
8      elif operator == '-':
9          result = num1 - num2
10     elif operator == '*':
11         result = num1 * num2
12     elif operator == '/':
13         if num2 != 0:
14             result = num1 / num2
15         else:
16             print("Error: Division by zero is not allowed.")
17             return
18     else:
19         print("Invalid operator.")
20         return
21     print(f"Result: {result}")
22 calculator()

```

OUTCOME:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\yesha\OneDrive\Desktop> & 'C:\Python314\python.exe' 'c:\Users\yesha\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\la
uncher' '60033' '--' 'c:\Users\yesha\OneDrive\Desktop\Function Calculator.py'
Enter first number: 10
Enter operator (+, -, *, /): +
Enter second number: 5
Result: 15.0

```

PROMPT:

design a simple calculator that can perform addition, subtraction, multiplication, and division operations. The calculator should take two numbers and an operator as input from the user and display the result and add comment lines and example usages

CODE:

```

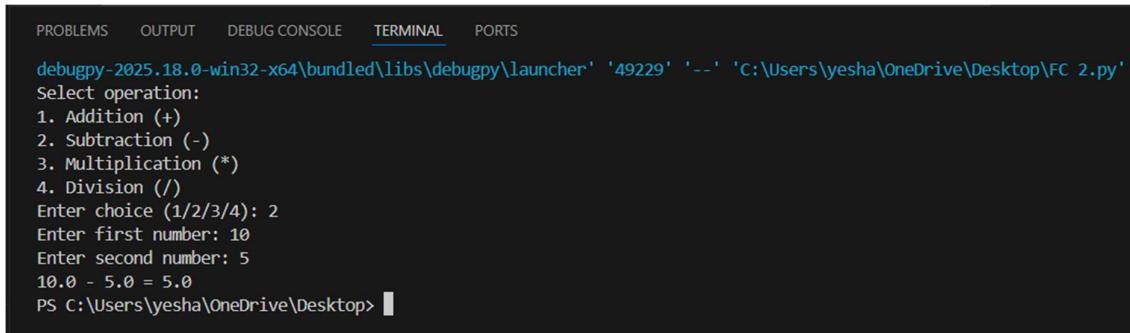
#design a simple calculator that can perform addition, subtraction, multiplication, and division operations. The calculator should take two numbers and an operator as input
def add(x, y):
    """Return the sum of x and y."""
    return x + y
def subtract(x, y):
    """Return the difference of x and y."""
    return x - y
def multiply(x, y):
    """Return the product of x and y."""
    return x * y
def divide(x, y):
    """Return the quotient of x and y. Raises an error if y is zero."""
    if y == 0:
        raise ValueError("Cannot divide by zero.")
    return x / y
def calculator():
    """A simple calculator that performs basic arithmetic operations."""
    print("Select operation:")
    print("1. Addition (+)")
    print("2. Subtraction (-)")
    print("3. Multiplication (*")
    print("4. Division (/)")

    # Take input from the user
    choice = input("Enter choice (1/2/3/4): ")
    num1 = float(input("Enter first number: "))
    num2 = float(input("Enter second number: "))

    if choice == '1':
        print(f"{num1} + {num2} = {add(num1, num2)}")
    elif choice == '2':
        print(f"{num1} - {num2} = {subtract(num1, num2)}")
    elif choice == '3':
        print(f"{num1} * {num2} = {multiply(num1, num2)}")
    elif choice == '4':
        try:
            print(f"{num1} / {num2} = {divide(num1, num2)}")
        except ValueError as e:
            print(e)
    else:
        print("Invalid input")
# Example usage:
if __name__ == "__main__":
    calculator()

```

OUTPUT:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '49229' '--' 'C:\Users\yesha\OneDrive\Desktop\FC 2.py'
Select operation:
1. Addition (+)
2. Subtraction (-)
3. Multiplication (*)
4. Division (/)
Enter choice (1/2/3/4): 2
Enter first number: 10
Enter second number: 5
10.0 - 5.0 = 5.0
PS C:\Users\yesha\OneDrive\Desktop>
```

Explanation:

First, a basic prompt was given to create a simple calculator. The code worked but did not have comments or examples, so it was not easy to understand.

Then, the prompt was improved by asking for comments and example usage. Because of this, the generated code became clearer, well-documented, and easier to use.

This shows that improving the prompt helps in getting better and more understandable code.

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:

write a program to sort the student marks

CODE:

```
1 # write a program to sort the student marks
2 def sort_student_marks(marks):
3     return sorted(marks)
4 # Example usage
5 student_marks = [88, 92, 79, 85, 95]
6 sorted_marks = sort_student_marks(student_marks)
7 print("Sorted student marks:", sorted_marks)
8 # Output: Sorted student marks: [79, 85, 88, 92, 95]
9
```

OUTPUT:



```
PS C:\Users\yesha\OneDrive\Desktop> cd 'c:\Users\yesha\OneDrive\Desktop'; & 'c:\Python314\python.exe' 'c:\Users\yesha\vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '65117' '--' 'C:\Users\yesha\OneDrive\Desktop\Student Marks.py'
Sorted student marks: [79, 85, 88, 92, 95]
PS C:\Users\yesha\OneDrive\Desktop>
```

PROMPT:

write a program to sort the student marks in ascending order and by using constraints by user input

CODE:

```
1 # write a program to sort the student marks in ascending order and by using constraint
2 def sort_student_marks(marks):
3     # Sort the marks in ascending order
4     sorted_marks = sorted(marks)
5     return sorted_marks
6 if __name__ == "__main__":
7     # Get number of students from user
8     num_students = int(input("Enter the number of students: "))
9
10    # Initialize an empty list to store marks
11    student_marks = []
12
13    # Get marks for each student
14    for i in range(num_students):
15        mark = float(input(f"Enter marks for student {i + 1}: "))
16        student_marks.append(mark)
17
18    # Sort the student marks
19    sorted_marks = sort_student_marks(student_marks)
20
21    # Display the sorted marks
22    print("Sorted student marks in ascending order:")
23    for mark in sorted_marks:
24        print(mark)
```

OUTPUT:

```
Enter the number of students: 5
Enter marks for student 1: 87
Enter marks for student 2: 55
Enter marks for student 3: 45
Enter marks for student 4: 64
Enter marks for student 5: 90
Sorted student marks in ascending order:
45.0
55.0
64.0
87.0
90.0
```

EXPLANATION:

First, a general prompt was given to sort student marks. Since it did not mention the sorting order or any conditions, the output was basic and unclear.

Then, the prompt was refined by specifying ascending order and user input constraints. This helped the AI generate a more accurate and well-defined sorting program.

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:

Write a program to check whether a number is prime or not

CODE:

```
1  #Write a program to check whether a number is prime or not
2  def is_prime(num):
3      if num <= 1:
4          return False
5      for i in range(2, int(num**0.5) + 1):
6          if num % i == 0:
7              return False
8      return True
9  number = int(input("Enter a number: "))
10 if is_prime(number):
11     print(f"{number} is a prime number.")
12 else:
13     print(f"{number} is not a prime number.")
```

OUTPUT:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\yesha\OneDrive\Desktop> & 'c:\Python314\python.exe' 'c:\Users\yesha\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '61497' '--' 'C:\Users\yesha\OneDrive\Desktop\Prime not prime.py'
Enter a number: 19
19 is prime number.
PS C:\Users\yesha\OneDrive\Desktop>
```

PROMPT:

Write a program to check whether a number is prime.

Input: 2

Output: Prime

CODE:

```
1 #Write a program to check whether a number is prime.
2 # Input: 2
3 # Output: Prime
4 def is_prime(n):
5     if n <= 1:
6         return "Not Prime"
7     for i in range(2, int(n**0.5) + 1):
8         if n % i == 0:
9             return "Not Prime"
10    return "Prime"
11 # Example usage
12 number = 2
13 result = is_prime(number)
14 print(result) # Output: Prime
```

OUTPUT:

```
PS C:\Users\yesha\OneDrive\Desktop> c;; cd 'c:\Users\yesha\OneDrive\Desktop'; & 'C:\Python314\python.exe' 'c:\Users\yesha\.vscode\extensions\ms-python.debugpy-2
025.18.0-win32-x64\bundled\libs\debugpy\launcher' '51974' '--' 'C:\Users\yesha\OneDrive\Desktop\prime_not_prime.py'
Prime
PS C:\Users\yesha\OneDrive\Desktop> []
```

EXPLANATION:

The program checks whether a given number is a prime number.

It takes a number as input and tests if it can be divided by any number other than 1 and itself.

For the input 2, it has no other divisors, so the program prints Prime.

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:

Create a user interface for a student grading system that calculates total marks, percentage, and grade based on user input.

CODE:

```

1 #Create a user interface for a student grading system that calculates total marks, percentage, and grade based on userinput
2 def calculate_grade(marks_obtained, total_marks):
3     percentage = (marks_obtained / total_marks) * 100
4
5     if percentage >= 90:
6         grade = 'A'
7     elif percentage >= 80:
8         grade = 'B'
9     elif percentage >= 70:
10        grade = 'C'
11    elif percentage >= 60:
12        grade = 'D'
13    else:
14        grade = 'F'
15
16    return percentage, grade
17 def main():
18     print("Student Grading System")
19     name = input("Enter student name: ")
20     total_marks = float(input("Enter total marks: "))
21     marks_obtained = float(input("Enter marks obtained: "))
22
23     percentage, grade = calculate_grade(marks_obtained, total_marks)
24
25     print(f"\nStudent Name: {name}")
26     print(f"Total Marks: {total_marks}")
27     print(f"Marks Obtained: {marks_obtained}")
28     print(f"Percentage: {percentage:.2f}%")
29     print(f"Grade: {grade}")
30 if __name__ == "__main__":
31     main()

```

OUTPUT:

● Student Grading System
 Enter student name: sony
 Enter total marks: 20
 Enter marks obtained: 5

Student Name: sony
 Total Marks: 20.0
 Marks Obtained: 5.0
 Percentage: 25.00%
 Grade: F

EXPLANATION:

The prompt generated a student grading system that takes user input, calculates percentage, assigns a grade, and displays the result clearly.

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:

Write a program to convert kilometers to miles and miles to kilometers.

CODE:

```

1 #Write a program to convert kilometers to miles and miles to kilometers.
2 def km_to_miles(km):
3     return km * 0.621371
4
5 def miles_to_km(miles):
6     return miles * 1.60934
7
8 print("Kilometers to Miles and Miles to Kilometers Converter")
9
10 while True:
11     print("\n1. Kilometers to Miles")
12     print("2. Miles to Kilometers")
13     print("3. Exit")
14
15     choice = input("Enter your choice (1/2/3): ")
16
17     if choice == '1':
18         km = float(input("Enter distance in kilometers: "))
19         miles = km_to_miles(km)
20         print(f"{km} kilometers is equal to {miles:.2f} miles.")
21
22     elif choice == '2':
23         miles = float(input("Enter distance in miles: "))
24         km = miles_to_km(miles)
25         print(f"{miles} miles is equal to {km:.2f} kilometers.")
26
27     elif choice == '3':
28         print("Exiting the program.")
29         break
30
31     else:
32         print("Invalid choice. Please enter a valid option (1/2/3).")

```

OUTPUT:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\yesha\OneDrive\Desktop>
PS C:\Users\yesha\OneDrive\Desktop> c:; cd 'c:\Users\yesha\OneDrive\Desktop'; & 'C:\Python314\python.exe' 'c:\Users\yesha\.vscode\extensions\ms-python.python.debug\2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '50873' '--' 'C:\Users\yesha\OneDrive\Desktop\kilometers.py'
Kilometers to Miles and Miles to Kilometers Converter

1. Kilometers to Miles
2. Miles to Kilometers
3. Exit
Enter your choice (1/2/3): 1
Enter distance in kilometers: 10
10.0 kilometers is equal to 6.21 miles.

1. Kilometers to Miles
2. Miles to Kilometers
3. Exit
Enter your choice (1/2/3): 3
Exiting the program.
PS C:\Users\yesha\OneDrive\Desktop>

```

EXPLANATION:

The program converts kilometers to miles and miles to kilometers using user input and displays accurate results through a menu-driven interface.