

**Name: T.Akanksh**

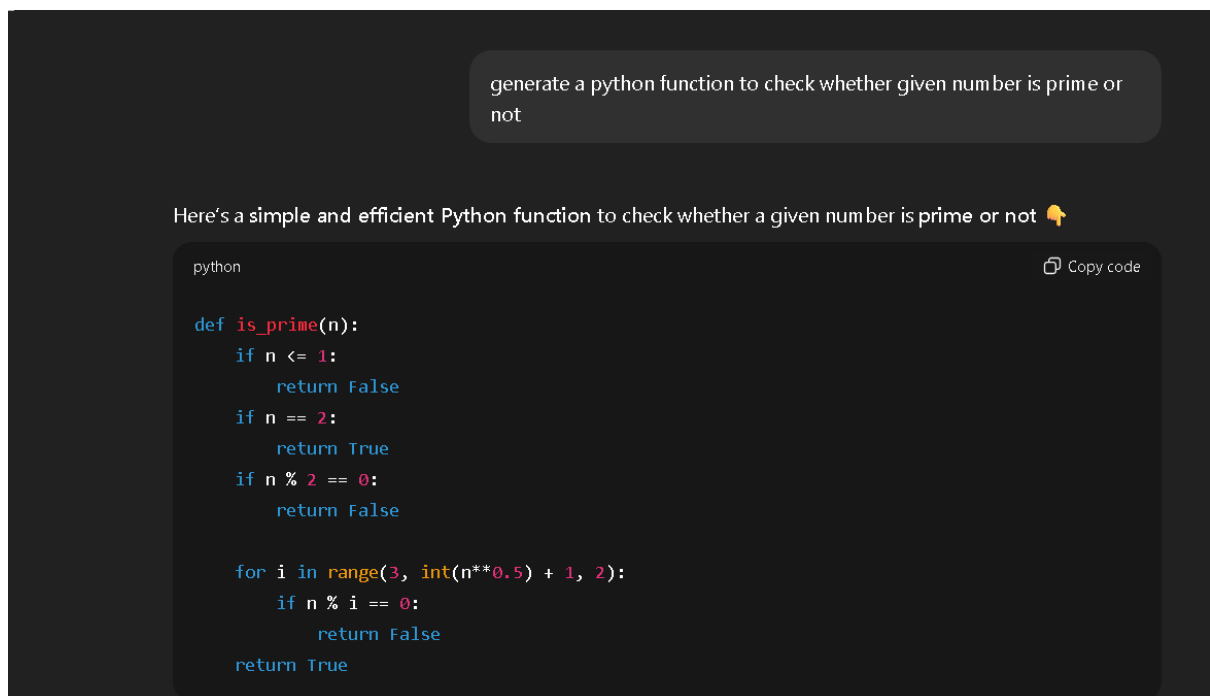
**Hno:2303A51078**

## **Lab 4:** Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques

.....

### **Task 1:**

Zero-shot: Prompt AI with only the instruction. Write a Python function to determine whether a given number is prime



### **Task 2:**

One-shot: Provide one example: Input: [1, 2, 3, 4], Output: 10 to help AI generate a function that calculates the sum of elements in a list

generate a python program to calculate sum of elements in alist take input as [1,2,3,4] and output 10

Here is a simple Python program to calculate the sum of elements in a list, taking input exactly in this format 🍌 [1,2,3,4] and producing output 10.

```
python
# Take input in list format
lst = input("Enter list elements: ")

# Convert string input to list of integers
numbers = []
num = ""

for ch in lst:
    if ch.isdigit() or ch == '-':
        num += ch
    else:
        if num != "":
            numbers.append(int(num))
            num = ""
if num != "":
```

Copy code



```
if num != "":
    numbers.append(int(num))

# Calculate sum without using sum()
total = 0
for i in numbers:
    total += i

print("Sum =", total)
```

Copy code

#### ▶ Sample Input

csharp

Copy code

[1,2,3,4]

#### ▶ Output

ini

Copy code

Sum = 10

## Task 3:

Few-shot: Give 2–3 examples to create a function that extracts digits from an alphanumeric string.

generate a python function to return digits from the alphanumeric strings examples like (input1:yash1020@ and output:1020,input2:yash@ output: ,input3:Priyu5122 output:5122)

Here is a clean Python function that extracts and returns digits from an alphanumeric string, exactly matching your examples 🙌

python

Copy code

```
def extract_digits(s):
    result = ""
    for ch in s:
        if ch.isdigit():
            result += ch
    return result
```

#### ▶ Example Usage

python

Copy code

```
print(extract_digits("yash1020@")) # Output: 1020
print(extract_digits("yash@")) # Output: 
print(extract_digits("Priyu5122")) # Output: 5122
```

## Task 4:

Compare zero-shot vs few-shot prompting for generating a function that counts the number of vowels in a string.

Write a Python function to count the number of vowels in a string.

Here is a simple and correct Python function to count the number of vowels in a string 🙌

python

Copy code

```
def count_vowels(s):
    vowels = "aeiouAEIOU"
    count = 0
    for ch in s:
        if ch in vowels:
            count += 1
    return count
```

#### ▶ Example Usage

python

Copy code

```
print(count_vowels("hello")) # Output: 2
print(count_vowels("AEIOU")) # Output: 5
print(count_vowels("python")) # Output: 1
```

Example 1:  
Input: "hello" → Output: 2


Example 2:  
Input: "AEIOU" → Output: 5

Example 3:  
Input: "xyz" → Output: 0

Now write a Python function to count the number of vowels in a string.

Here is a Python function that matches all the given examples:

python

 Copy code

```
def count_vowels(s):  
    count = 0  
    for ch in s.lower():  
        if ch in ['a', 'e', 'i', 'o', 'u']:  
            count += 1  
    return count
```



### 3 Output Comparison Table

Aspect	Zero-Shot	Few-Shot
Examples provided	✗ No	✓ Yes
Case handling	Implicit	Explicit (from examples)
Output reliability	Good	Better
Control over logic	Low	High
Suitability for exams/interviews	Good	Very Good

## Explanation:

In zero-shot prompting, the model generates the function using its general understanding of programming and vowels. Since no examples are given, the model decides on its own how to handle cases like uppercase letters.

In few-shot prompting, the examples clearly show what the expected input and output look like. From these examples, the model learns important rules such as:

- Uppercase vowels should be counted
- Strings without vowels should return 0

Because of this guidance, the model produces a more accurate and user-aligned function. Examples act like hints, reducing ambiguity and improving correctness.

## Task 5:

Use few-shot prompting with 3 sample inputs to generate a function that determines the minimum of three numbers without using the built-in `min()` function.

```
example1:  
input:1,5,7  output:1  
example2:  
input:40,60,50  output:40  
example3:  
input:9,6,5  output:5  
now generate a python function to find minimum of 3 three numbers  
without using thr inbuilt function min()
```

Here is a Python function to find the minimum of three numbers without using the inbuilt `min()` function, matching all your examples 🙌

python

 Copy code

```
def find_min(a, b, c):  
    if a <= b and a <= c:  
        return a  
    elif b <= a and b <= c:  
        return b  
    else:  
        return c
```

