# AI ASSISTED CODING

Name: E. Shivani
Ht. No: 2303A51098
Batch:2

## Assignment-3.1 Task-1:

Experiment – Prompt Engineering Techniques

Task Description

Design and refine prompts using different prompting strategies to generate Python programs for basic computational problems.

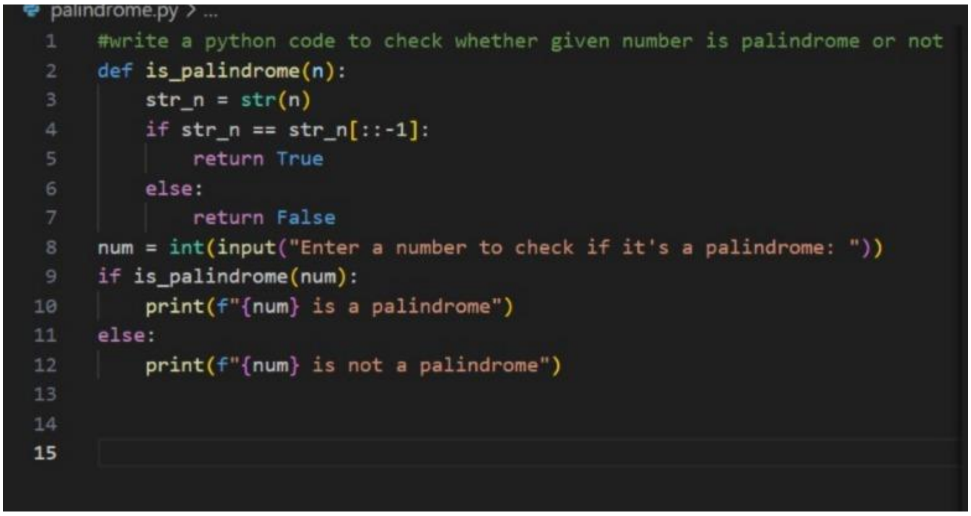
**Question 1:** Zero-Shot Prompting (Palindrome Number Program)

Write a zero-shot prompt (without providing any examples) to generate a Python function that checks whether a given number is a palindrome.

**Task:**

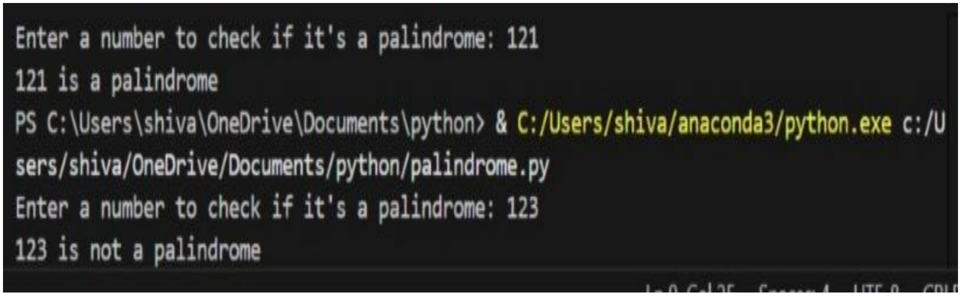
- Record the AI-generated code.
- Test the code with multiple inputs.
- Identify any logical errors or missing edge-case handling.

**Code:**



```python
#write a python code to check whether given number is palindrome or not
def is_palindrome(n):
    str_n = str(n)
    if str_n == str_n[::-1]:
        return True
    else:
        return False
num = int(input("Enter a number to check if it's a palindrome: "))
if is_palindrome(num):
    print(f"{num} is a palindrome")
else:
    print(f"{num} is not a palindrome")
```

## Output:



```
Enter a number to check if it's a palindrome: 121
121 is a palindrome
PS C:\Users\shiva\OneDrive\Documents\python> & C:/Users/shiva/anaconda3/python.exe c:/U
sers/shiva/OneDrive/Documents/python/palindrome.py
Enter a number to check if it's a palindrome: 123
123 is not a palindrome
```

**Question 2:** One-Shot Prompting (Factorial Calculation)
Write a one-shot prompt by providing one input-output example and ask the AI to generate a Python function to compute the factorial of a given number.

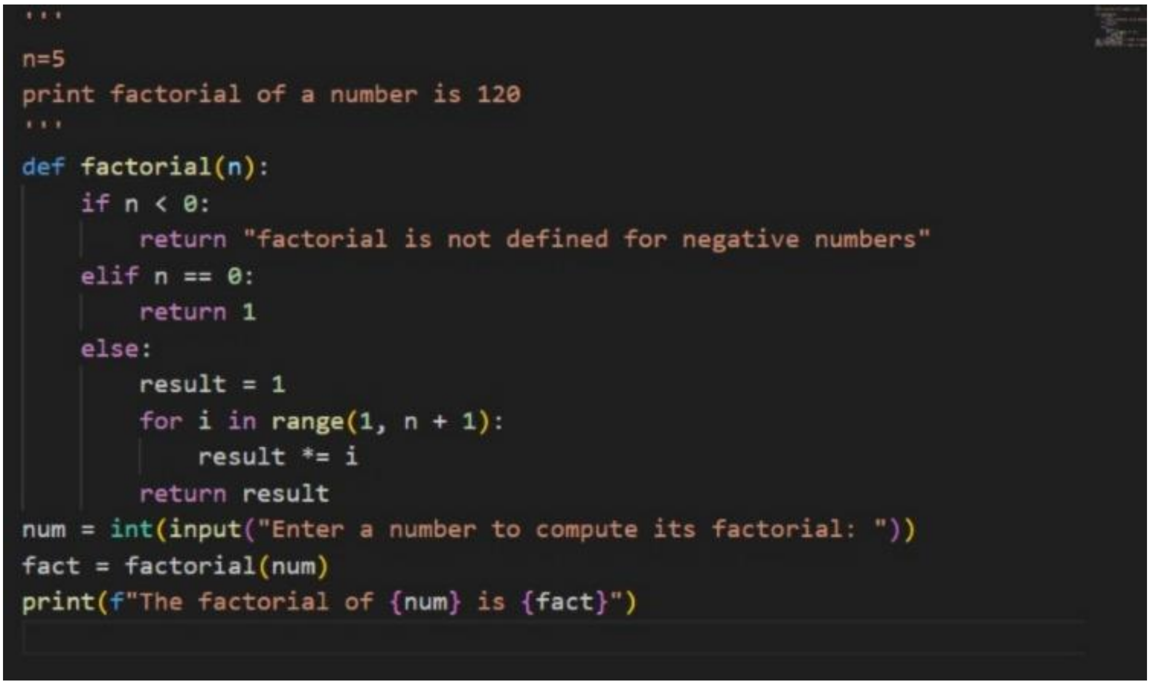
### Example:

Input: 5 → Output: 120

### Task:

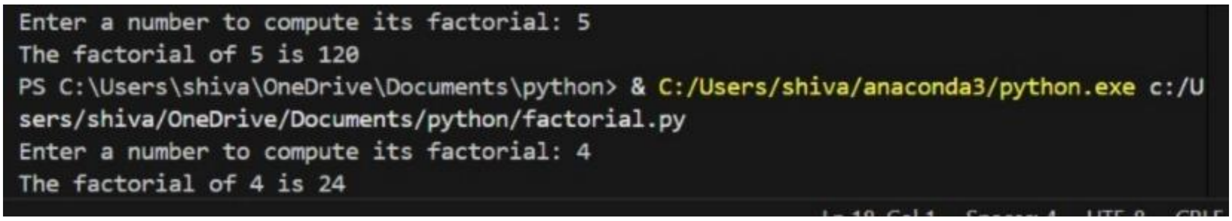
- Compare the generated code with a zero-shot solution.
- Examine improvements in clarity and correctness.

### Code:



```python
'''
n=5
print factorial of a number is 120
'''
def factorial(n):
    if n < 0:
        return "factorial is not defined for negative numbers"
    elif n == 0:
        return 1
    else:
        result = 1
        for i in range(1, n + 1):
            result *= i
        return result
num = int(input("Enter a number to compute its factorial: "))
fact = factorial(num)
print(f"The factorial of {num} is {fact}")
```

### Output:



```
Enter a number to compute its factorial: 5
The factorial of 5 is 120
PS C:\Users\shiva\OneDrive\Documents\python> & C:/Users/shiva/anaconda3/python.exe c:/U
sers/shiva/OneDrive/Documents/python/factorial.py
Enter a number to compute its factorial: 4
The factorial of 4 is 24
```

**Question 3:** Few-Shot Prompting (Armstrong Number Check)

Write a few-shot prompt by providing multiple input-output examples to guide the AI in generating a Python function to check whether a given number is an Armstrong number.

## Examples:

• Input: 153 → Output: Armstrong Number

• Input: 370 → Output: Armstrong Number

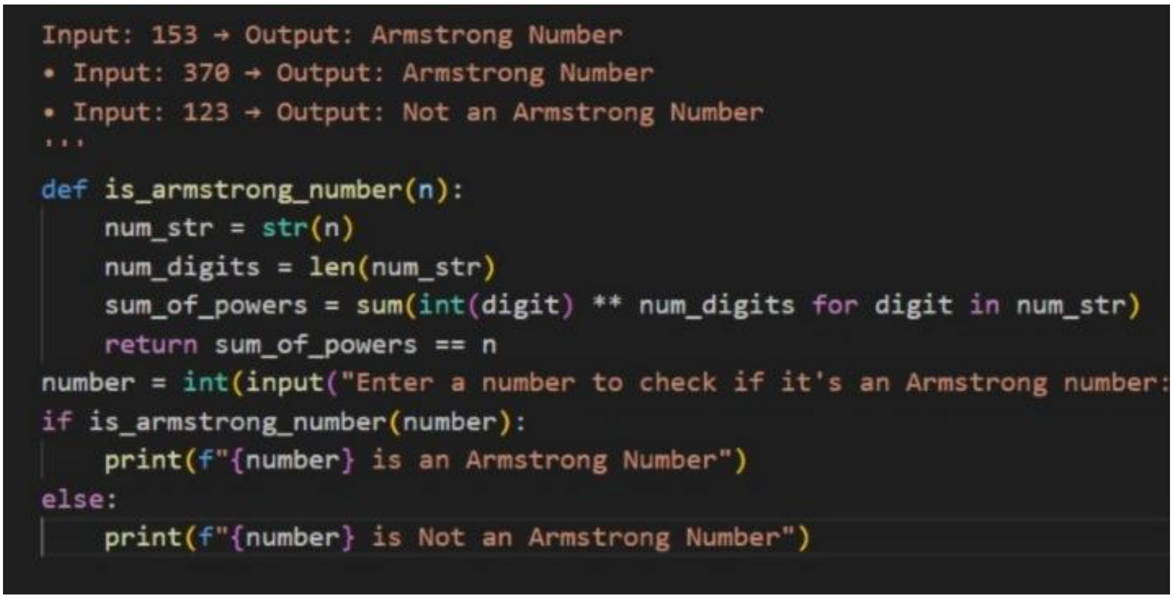
• Input: 123 → Output: Not an Armstrong Number

## Task:

• Analyze how multiple examples influence code structure and accuracy.

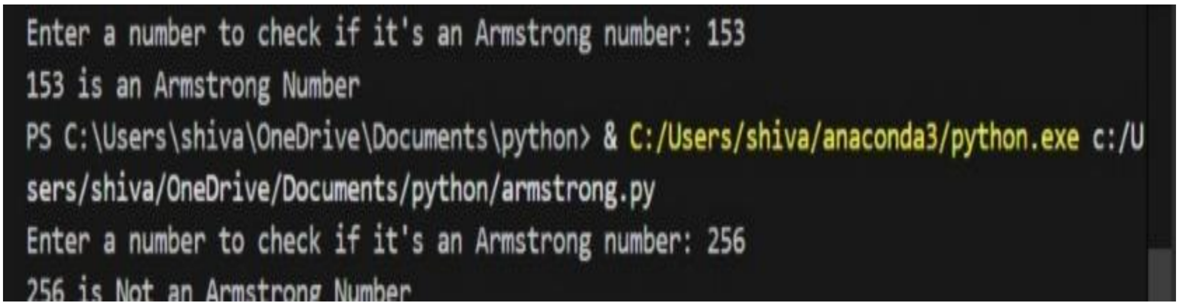
• Test the function with boundary values and invalid inputs.

## Code:



```
Input: 153 → Output: Armstrong Number
• Input: 370 → Output: Armstrong Number
• Input: 123 → Output: Not an Armstrong Number
'''
def is_armstrong_number(n):
    num_str = str(n)
    num_digits = len(num_str)
    sum_of_powers = sum(int(digit) ** num_digits for digit in num_str)
    return sum_of_powers == n
number = int(input("Enter a number to check if it's an Armstrong number:
if is_armstrong_number(number):
    print(f"{number} is an Armstrong Number")
else:
    print(f"{number} is Not an Armstrong Number")
```

## Output:



```
Enter a number to check if it's an Armstrong number: 153
153 is an Armstrong Number
PS C:\Users\shiva\OneDrive\Documents\python> & C:/Users/shiva/anaconda3/python.exe c:/U
sers/shiva/OneDrive/Documents/python/armstrong.py
Enter a number to check if it's an Armstrong number: 256
256 is Not an Armstrong Number
```

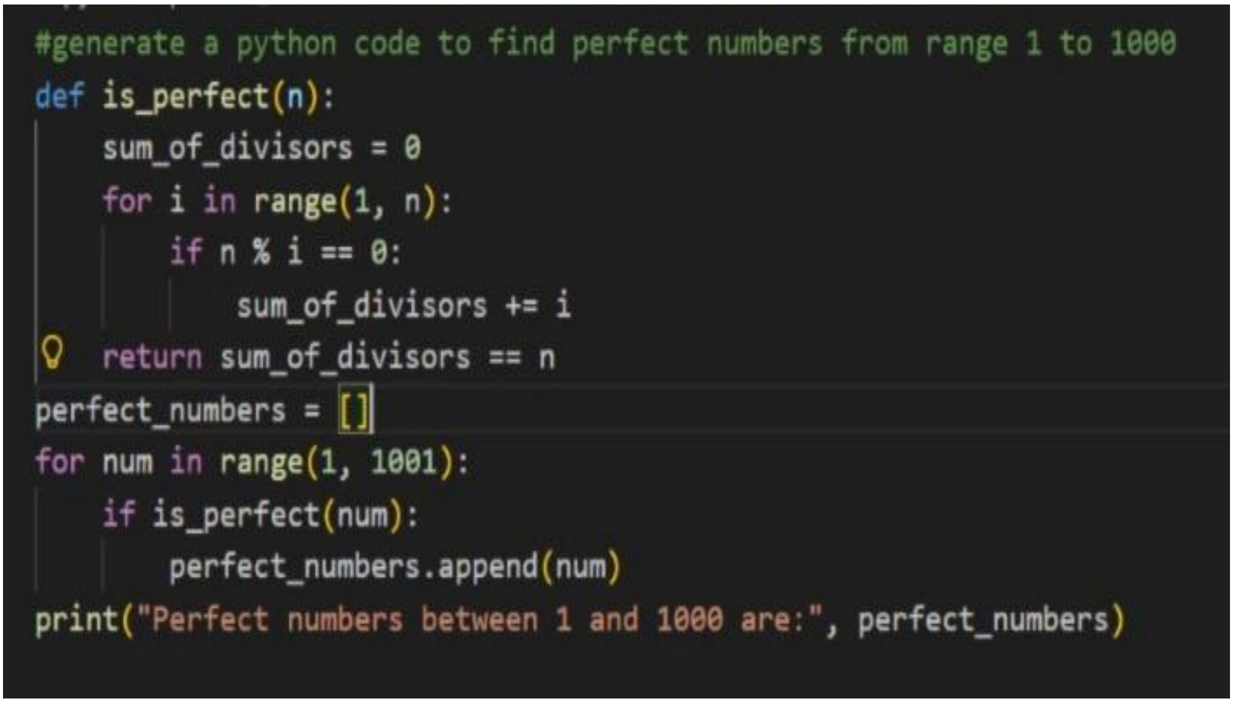
**Question 5:** Zero-Shot Prompting (Perfect Number Check)

Write a zero-shot prompt (without providing any examples) to generate a Python function that checks whether a given number is a perfect number.

## Task:

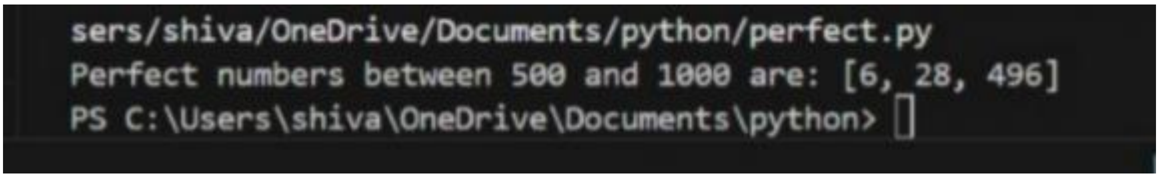
- Record the AI-generated code.
- Test the program with multiple inputs.
- Identify any missing conditions or inefficiencies in the logic.

## Code:



```
#generate a python code to find perfect numbers from range 1 to 1000
def is_perfect(n):
    sum_of_divisors = 0
    for i in range(1, n):
        if n % i == 0:
            sum_of_divisors += i
    return sum_of_divisors == n
perfect_numbers = []
for num in range(1, 1001):
    if is_perfect(num):
        perfect_numbers.append(num)
print("Perfect numbers between 1 and 1000 are:", perfect_numbers)
```

## Output:



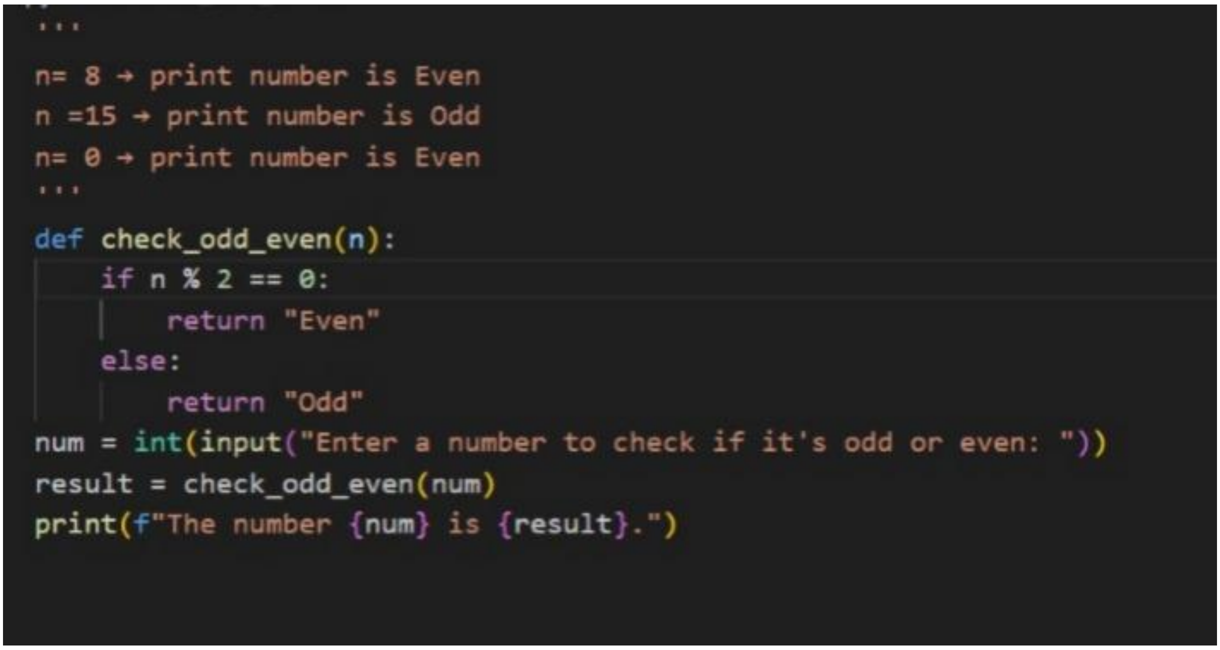
```
sers/shiva/OneDrive/Documents/python/perfect.py
Perfect numbers between 500 and 1000 are: [6, 28, 496]
PS C:\Users\shiva\OneDrive\Documents\python> 
```

**Question 6:** Few-Shot Prompting (Even or Odd Classification with Validation)
Write a few-shot prompt by providing multiple input-output examples to guide the AI in generating a Python program that determines whether a given number is even or odd, including proper input validation.

**Examples:**
• Input: 8 → Output: Even
• Input: 15 → Output: Odd
• Input: 0 → Output: Even

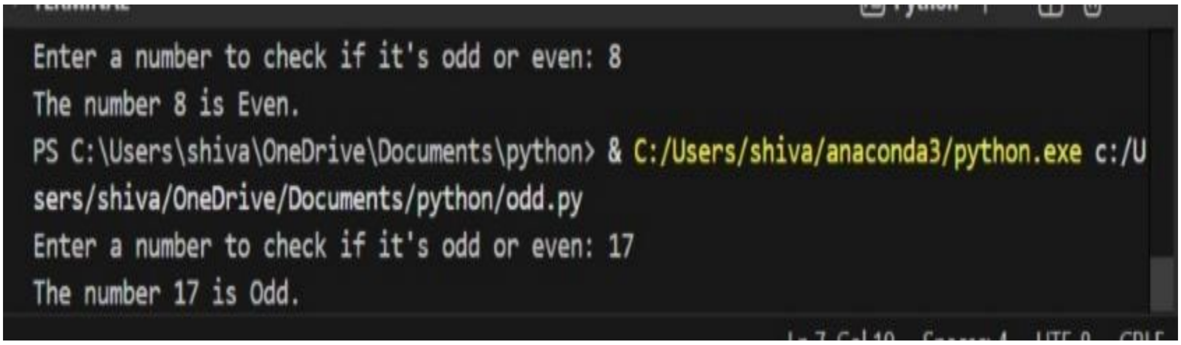
**Task:**
• Analyze how examples improve input handling and output clarity.
• Test the program with negative numbers and non-integer inputs.

## Code:



```
'''
n= 8 → print number is Even
n =15 → print number is Odd
n= 0 → print number is Even
'''
def check_odd_even(n):
    if n % 2 == 0:
        return "Even"
    else:
        return "Odd"
num = int(input("Enter a number to check if it's odd or even: "))
result = check_odd_even(num)
print(f"The number {num} is {result}.")
```

## Output:



```
Enter a number to check if it's odd or even: 8
The number 8 is Even.
PS C:\Users\shiva\OneDrive\Documents\python> & C:/Users/shiva/anaconda3/python.exe c:/U
sers/shiva/OneDrive/Documents/python/odd.py
Enter a number to check if it's odd or even: 17
The number 17 is Odd.
```