

School of Computer Science and Artificial Intelligence

Lab Assignment # 3.1

Name of Student : GINNE SAI TEJA
Enrollment No. : 2303A51526
Batch No. : 22

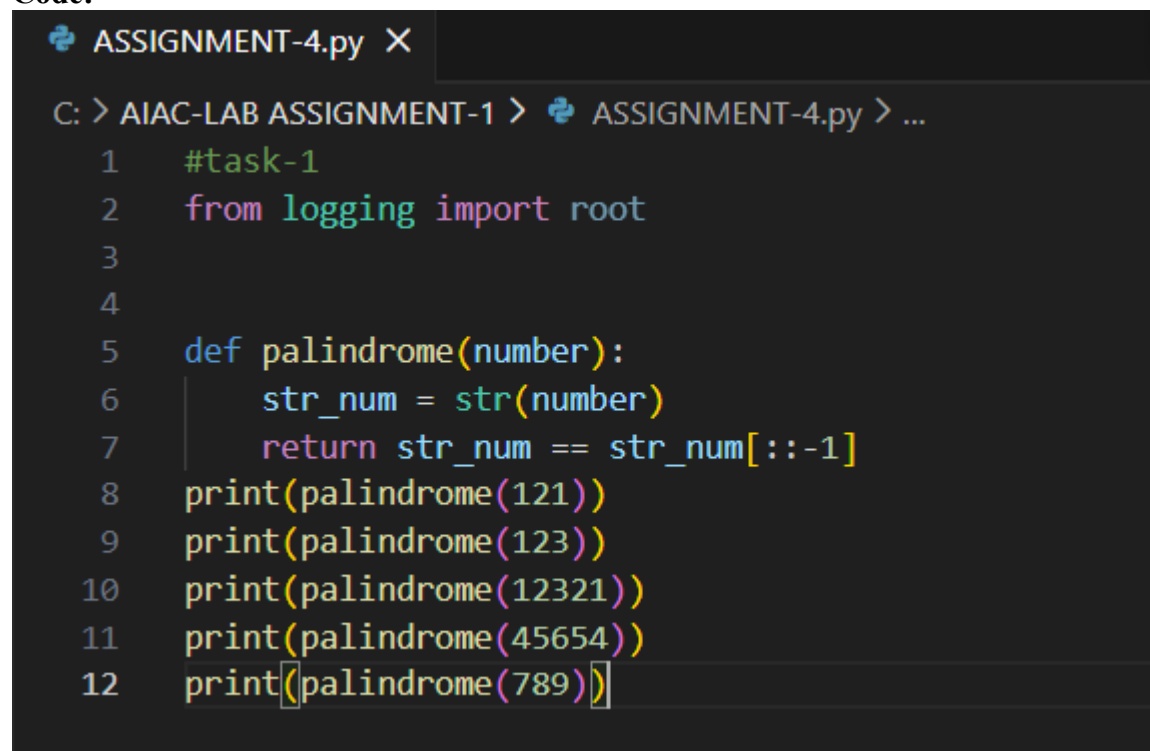
Question 1: Zero-Shot Prompting (Palindrome Number Program)

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

Task:

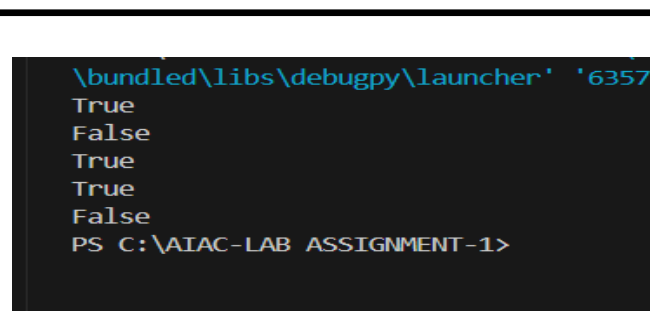
- Record the AI-generated code.
- Test the code with multiple inputs.
- Identify any logical errors or missing edge-case handling.

Code:-



```
ASSIGNMENT-4.py X
C: > AIAC-LAB ASSIGNMENT-1 > ASSIGNMENT-4.py > ...
1  #task-1
2  from logging import root
3
4
5  def palindrome(number):
6      str_num = str(number)
7      return str_num == str_num[::-1]
8  print(palindrome(121))
9  print(palindrome(123))
10 print(palindrome(12321))
11 print(palindrome(45654))
12 print(palindrome(789))
```

Output:-



```
\bundled\libs\debugpy\launcher' '63572
True
False
True
True
False
PS C:\AIAC-LAB ASSIGNMENT-1>
```

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.

Prompt:-

Write a Python function factorial(n) that returns the factorial of a non-negative integer n.

Code:-

```
#task-2
#Write a Python function factorial(n) that returns the factorial of a non-negative integer n.
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
print(factorial(5))
print(factorial(0))
print(factorial(7))
```

Output:-

```
120
1
5040
```

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.

(Optional Extension)

Prompt:-

Write a Python function is_armstrong(n) that returns True if n is an Armstrong number and False otherwise.

Code:-

```
#task-3
#Write a Python function is_armstrong(n) that returns True if n is an Armstrong number and False otherwise.
def is_armstrong(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
print(is_armstrong(153))
print(is_armstrong(9474))
print(is_armstrong(123))
print(is_armstrong(370))
print(is_armstrong(9475))
print(is_armstrong(0))
print(is_armstrong(1))
```

Output:-

```
True
True
False
True
False
True
True
```

Question 4: Context-Managed Prompting (Optimized Number Classification)

Design a context-managed prompt with clear instructions and constraints to generate an optimized Python program that classifies a number as prime, composite, or neither.

Task:

- Ensure proper input validation.
- Optimize the logic for efficiency.
- Compare the output with earlier prompting strategies.

Prompt:-

Write a Python program using a context manager that classifies a number as prime, composite, or neither.

Code:-

```
42 #task-4
43 #Write a Python program using a context manager that classifies a number as prime, composite, or neither.
44 class NumberClassifier:
45     def __init__(self, number):
46         self.number = number
47
48     def __enter__(self):
49         return self.classify_number()
50
51     def __exit__(self, exc_type, exc_value, traceback):
52         pass
53
54     def classify_number(self):
55         if self.number < 2:
56             return "neither"
57         for i in range(2, int(self.number ** 0.5) + 1):
58             if self.number % i == 0:
59                 return "composite"
60         return "prime"
61 with NumberClassifier(7) as classification:
62     print(f"7 is {classification}.")
63 with NumberClassifier(10) as classification:
64     print(f"10 is {classification}.")
65 with NumberClassifier(1) as classification:
66     print(f"1 is {classification}.")
67 with NumberClassifier(0) as classification:
68     print(f"0 is {classification}.")
69 with NumberClassifier(13) as classification:
70     print(f"13 is {classification}.")
```

Output:-

```
7 is prime.
10 is composite.
1 is neither.
0 is neither.
13 is prime.
PS C:\Users\SaiTeja>
```

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:-

```
72 #task-5
73 def perfect_number(n):
74     if n < 2:
75         return False
76     divisors_sum = sum(i for i in range(1, n) if n % i == 0)
77     return divisors_sum == n
78 print(perfect_number(6))
79 print(perfect_number(28))
80 print(perfect_number(12))
81 print(perfect_number(496))
82 print(perfect_number(15))
```

Output:-

```
True
True
False
True
False
```

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.

Prompt:-

Write a Python function `check_even_odd(n)` that prints "Even" if `n` is even and "Odd" if `n` is odd.

Code:-

```
83
84 #task-6
85 #Write a Python function check_even_odd(n) that prints "Even" if n is even and "Odd" if n is odd.
86 def check_even_odd(n):
87     if n % 2 == 0:
88         print("Even")
89     else:
90         print("Odd")
91 check_even_odd(4)
92 check_even_odd(7)
93 check_even_odd(0)
94 check_even_odd(-3)
```

Output:-

```
Even
Odd
Even
Odd
```