

ASSIGNMENT-4.2

HT.NO:2303A510I4

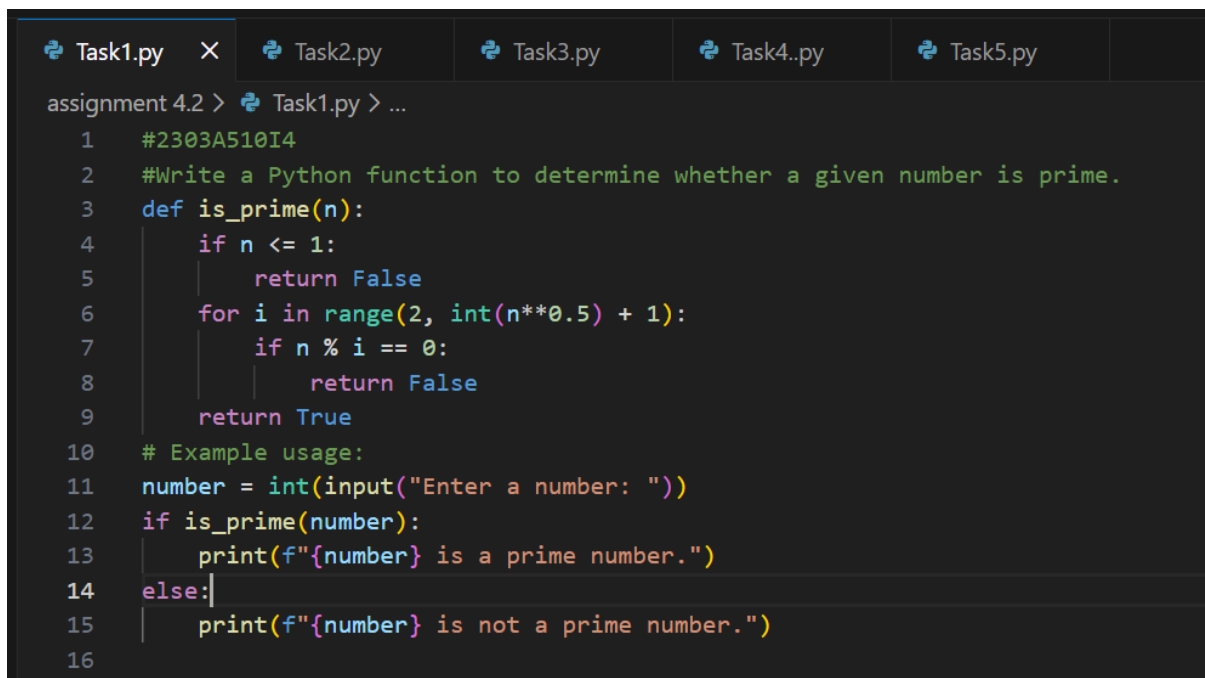
Batch.No:30

Task 1:

Prompt:

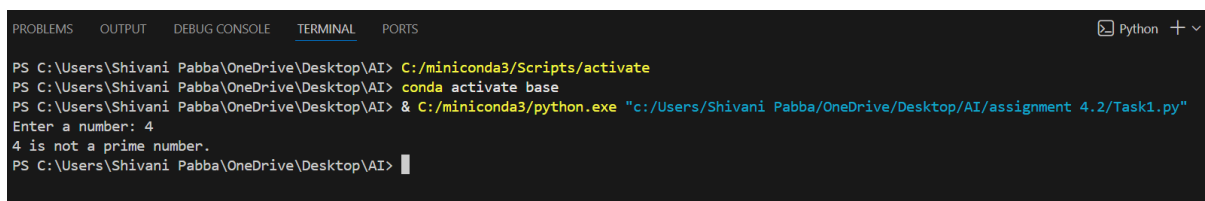
Write a Python function to determine whether a given number is prime.

Code:



```
Task1.py × Task2.py Task3.py Task4..py Task5.py
assignment 4.2 > Task1.py > ...
1  #2303A510I4
2  #Write a Python function to determine whether a given number is prime.
3  def is_prime(n):
4      if n <= 1:
5          return False
6      for i in range(2, int(n**0.5) + 1):
7          if n % i == 0:
8              return False
9      return True
10 # Example usage:
11 number = int(input("Enter a number: "))
12 if is_prime(number):
13     print(f"{number} is a prime number.")
14 else:
15     print(f"{number} is not a prime number.")
16
```

Output:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + v
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> C:/miniconda3/Scripts/activate
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> conda activate base
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> & C:/miniconda3/python.exe "c:/Users/Shivani Pabba/OneDrive/Desktop/AI/assignment 4.2/Task1.py"
Enter a number: 4
4 is not a prime number.
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI>
```

Observation:

Zero-shot prompting worked well because the instruction was clear. The model generated correct logic without needing examples. However, we had to explicitly call the function to see output.

Task 2:

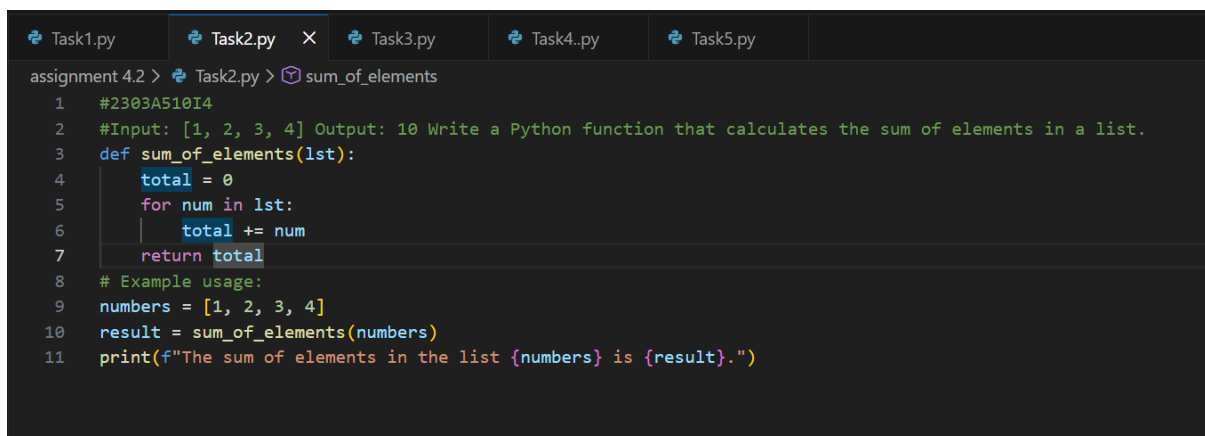
Prompt:

Input: [1, 2, 3, 4]

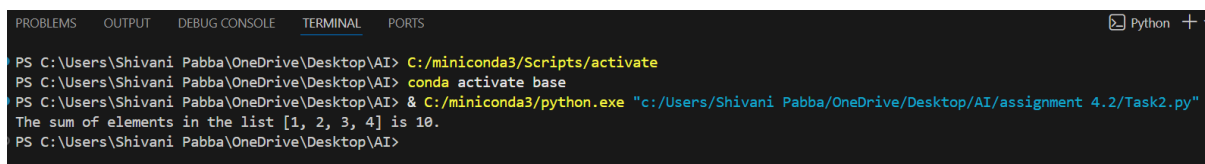
Output: 10

Write a Python function that calculates the sum of elements in a list.

Code:

A screenshot of a code editor with a dark theme. The editor has several tabs at the top: Task1.py, Task2.py (active), Task3.py, Task4.py, and Task5.py. The active tab shows a Python script. The script starts with a comment '#2303A510I4'. The first line of the prompt is '#Input: [1, 2, 3, 4] Output: 10 Write a Python function that calculates the sum of elements in a list.' The function 'def sum_of_elements(lst):' is defined with a 'total' variable initialized to 0. A 'for' loop iterates over 'num' in 'lst', adding each 'num' to 'total'. The function returns 'total'. Below the function, an example usage is shown: 'numbers = [1, 2, 3, 4]', 'result = sum_of_elements(numbers)', and 'print(f"The sum of elements in the list {numbers} is {result}.")'.

Output:

A screenshot of a terminal window with a dark theme. The terminal shows the following commands and output: 'PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> C:/miniconda3/Scripts/activate', 'PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> conda activate base', 'PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> & C:/miniconda3/python.exe "c:/Users/Shivani Pabba/OneDrive/Desktop/AI/assignment 4.2/Task2.py"', and the output 'The sum of elements in the list [1, 2, 3, 4] is 10.' The terminal also shows the prompt 'PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI>'.

Observation:

Providing one example helped the model clearly understand that all elements in the list must be added. The single example was enough to guide the correct logic.

Task 3:

Prompt:

Example 1:

Input: "abc123"

Output: "123"

Example 2:

Input: "a1b2c3"

Output: "123"

Example 3:

Input: "hello2026world"

Output: "2026"

Write a Python function that extracts only the digits from an alphanumeric string.

Code:

```
Task1.py Task2.py Task3.py X Task4..py Task5.py
assignment 4.2 > Task3.py > ...
1  #2303A510I4
2  #Example 1:
3  #Input: "abc123"
4  #Output: "123"
5  #Example 2:
6  # Input: "a1b2c3"
7  #Output: "123"
8  #Example 3:
9  #Input: "hello2026world"
10 #Output: "2026"
11 #Write a Python function that extracts only the digits from an alphanumeric string.
12 def extract_digits(s):
13     digits = ""
14     for char in s:
15         if char.isdigit():
16             digits += char
17     return digits
18 # Example usage:
19 input_string = input("Enter an alphanumeric string: ")
20 result = extract_digits(input_string)
21 print(f"The digits extracted from the string '{input_string}' are: {result}.")
22
```

Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python + -
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> C:/miniconda3/Scripts/activate
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> conda activate base
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> & C:/miniconda3/python.exe "c:/Users/Shivani Pabba/OneDrive/Desktop/AI/assignment 4.2/Task3.py"
Enter an alphanumeric string: abc123
The digits extracted from the string 'abc123' are: 123.
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI>
```

Observation:

Multiple examples made the pattern very clear. The model understood that only numeric characters should be kept and everything else ignored. Few-shot improved clarity and accuracy.

Task 4:

Prompt:

Zero-shot:

Write a Python function that counts the number of vowels in a string.

Few-shot with examples:

Example 1:

Input: "hello"

Output: 2

Example 2:

Input: "AEIOU"

Output: 5

Example 3:

Input: "Python"

Output: 1

Write a Python function that counts the number of vowels in a string.

Code:

```
Task1.py Task2.py Task3.py Task4..py X Task5.py
assignment 4.2 > Task4..py > ...
1 #2303A510I4
2 #Zero-shot: Write a Python function that counts the number of vowels in a string.
3 def count_vowels(s):
4     vowels = "aeiouAEIOU"
5     count = 0
6     for char in s:
7         if char in vowels:
8             count += 1
9     return count
10 # Example usage:
11 input_string = input("Enter a string: ")
12 result = count_vowels(input_string)
13 print(f"The number of vowels in the string '{input_string}' is: {result}.")
14
15 #Few-shot:
16 # Example 1:Input: "hello" Output: 2
17 #Example 2:Input: "AEIOU" Output: 5
18 #Example 3:Input: "Python" Output: 1
19 #Write a Python function that counts the number of vowels in a string.
20 def count_vowels(s):
21     vowels = "aeiouAEIOU"
22     count = 0
23     for char in s:
24         if char in vowels:
25             count += 1
26     return count
27 # Example usage:
28 input_string = input("Enter a string: ")
29 result = count_vowels(input_string)
30 print(f"The number of vowels in the string '{input_string}' is: {result}.")
```

Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
• PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> C:/miniconda3/Scripts/activate
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> conda activate base
• PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> & C:/miniconda3/python.exe "c:/Users/Shivani Pabba/OneDrive/Desktop/AI/assignment 4.2/Task4..py"
Enter a string: aeiou
The number of vowels in the string 'aeiou' is: 5.
Enter a string: assist
The number of vowels in the string 'assist' is: 2.
```

Observation:

Zero-shot generated correct logic, but few-shot was more reliable. The examples helped ensure uppercase and lowercase vowels were handled properly. Few-shot reduced ambiguity.

Task 5:

Prompt:

Example 1:

Input: (3, 7, 5)

Output: 3

Example 2:

Input: (10, 2, 8)

Output: 2

Example 3:

Input: (4, 4, 9)

Output: 4

Write a Python function that determines the minimum of three numbers **without using the built-in min() function**.

Code:

```
Task1.py Task2.py Task3.py Task4.py Task5.py X
assignment 4.2 > Task5.py > ...
1 #2303A51014
2 #Example 1: Input: (3, 7, 5) Output: 3
3 #Example 2: Input: (10, 2, 8) Output: 2
4 #Example 3: Input: (4, 4, 9) Output: 4
5 #Write a Python function that determines the minimum of three numbers without using the built-in min() function.
6 def find_minimum(a, b, c):
7     minimum = a # Assume a is the minimum
8     if b < minimum:
9         minimum = b # Update minimum if b is smaller
10    if c < minimum:
11        minimum = c # Update minimum if c is smaller
12    return minimum
13 # Example usage:
14 num1 = float(input("Enter the first number: "))
15 num2 = float(input("Enter the second number: "))
16 num3 = float(input("Enter the third number: "))
17 result = find_minimum(num1, num2, num3)
18 print(f"The minimum of the numbers {num1}, {num2}, and {num3} is: {result}.")
```

Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> C:/miniconda3/Scripts/activate
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> conda activate base
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI> & C:/miniconda3/python.exe "c:/Users/Shivani Pabba/OneDrive/Desktop/AI/assignment 4.2/Task5.py"
Enter the first number: 32
Enter the second number: 23
Enter the third number: 75
The minimum of the numbers 32.0, 23.0, and 75.0 is: 23.0.
PS C:\Users\Shivani Pabba\OneDrive\Desktop\AI>
```

Observation:

The three examples helped the model understand comparison logic, including equal values.

The function correctly compares numbers without using the built-in `min()` function.