LAB ASSIGNMENT - 4.2

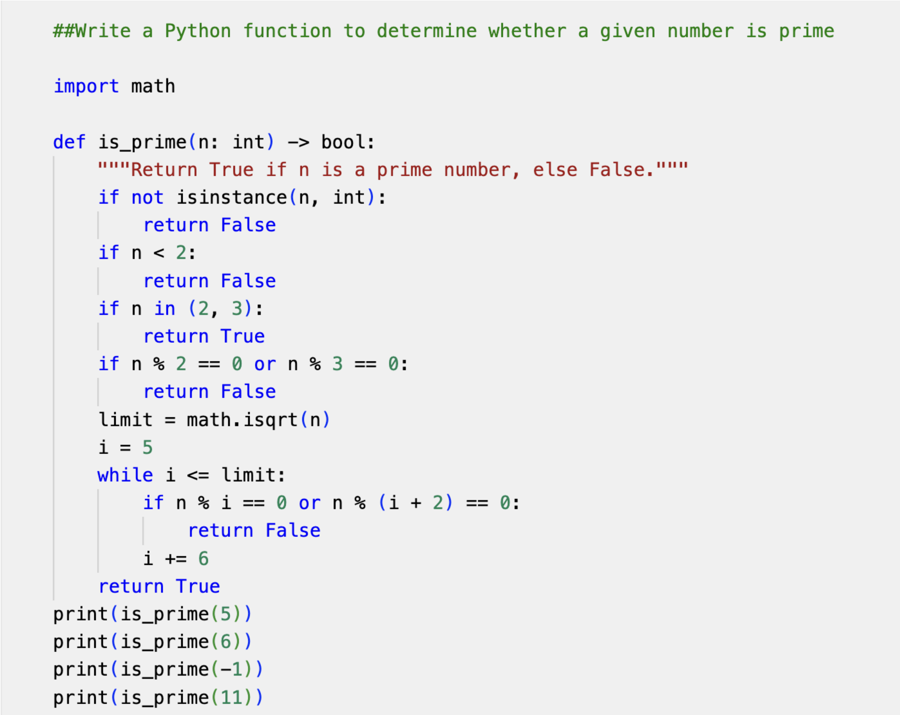
Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques

TASK DESCRIPTION 1:

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

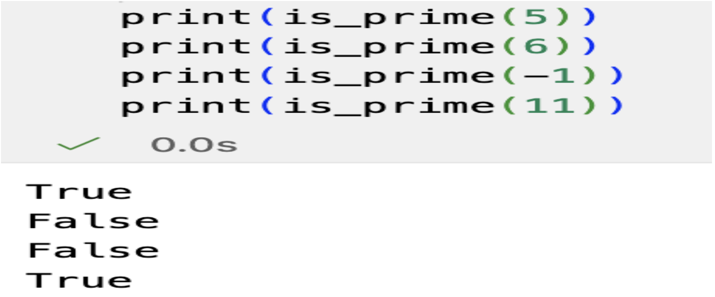
PROMPT: Write a Python function to determine whether a given number is prime.

CODE:



```python
##Write a Python function to determine whether a given number is prime

import math

def is_prime(n: int) -> bool:
    """Return True if n is a prime number, else False."""
    if not isinstance(n, int):
        return False
    if n < 2:
        return False
    if n in (2, 3):
        return True
    if n % 2 == 0 or n % 3 == 0:
        return False
    limit = math.isqrt(n)
    i = 5
    while i <= limit:
        if n % i == 0 or n % (i + 2) == 0:
            return False
        i += 6
    return True
print(is_prime(5))
print(is_prime(6))
print(is_prime(-1))
print(is_prime(11))
```

OUTPUT:



```
    print(is_prime(5))
    print(is_prime(6))
    print(is_prime(-1))
    print(is_prime(11))
 ✓   0.0s

True
False
False
True
```

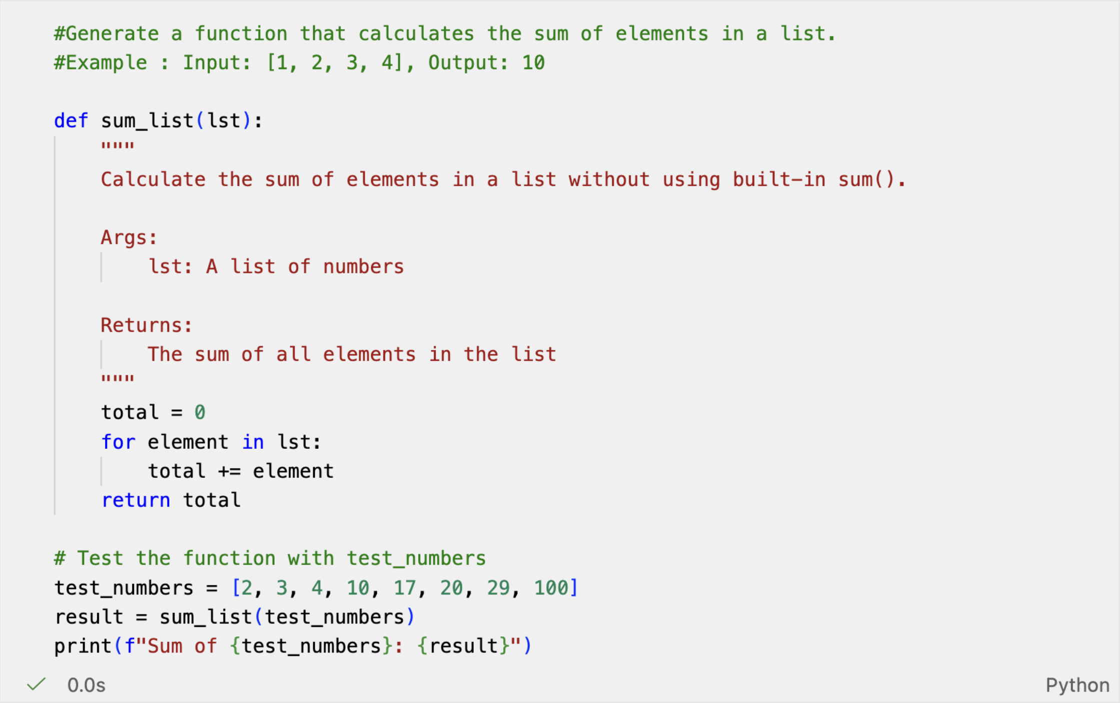
:

TASK DESCRIPTION 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.

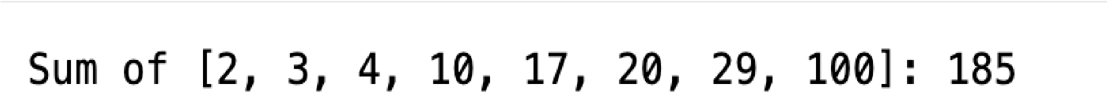
PROMPT: Generate a function that calculates the sum of elements in a list.
ONE SHOT Example: Input: [1, 2, 3, 4], Output: 10

CODE :



```python
#Generate a function that calculates the sum of elements in a list.
#Example : Input: [1, 2, 3, 4], Output: 10

def sum_list(lst):
    """
    Calculate the sum of elements in a list without using built-in sum().

    Args:
        lst: A list of numbers

    Returns:
        The sum of all elements in the list
    """
    total = 0
    for element in lst:
        total += element
    return total

# Test the function with test_numbers
test_numbers = [2, 3, 4, 10, 17, 20, 29, 100]
result = sum_list(test_numbers)
print(f"Sum of {test_numbers}: {result}")
```

OUTPUT:



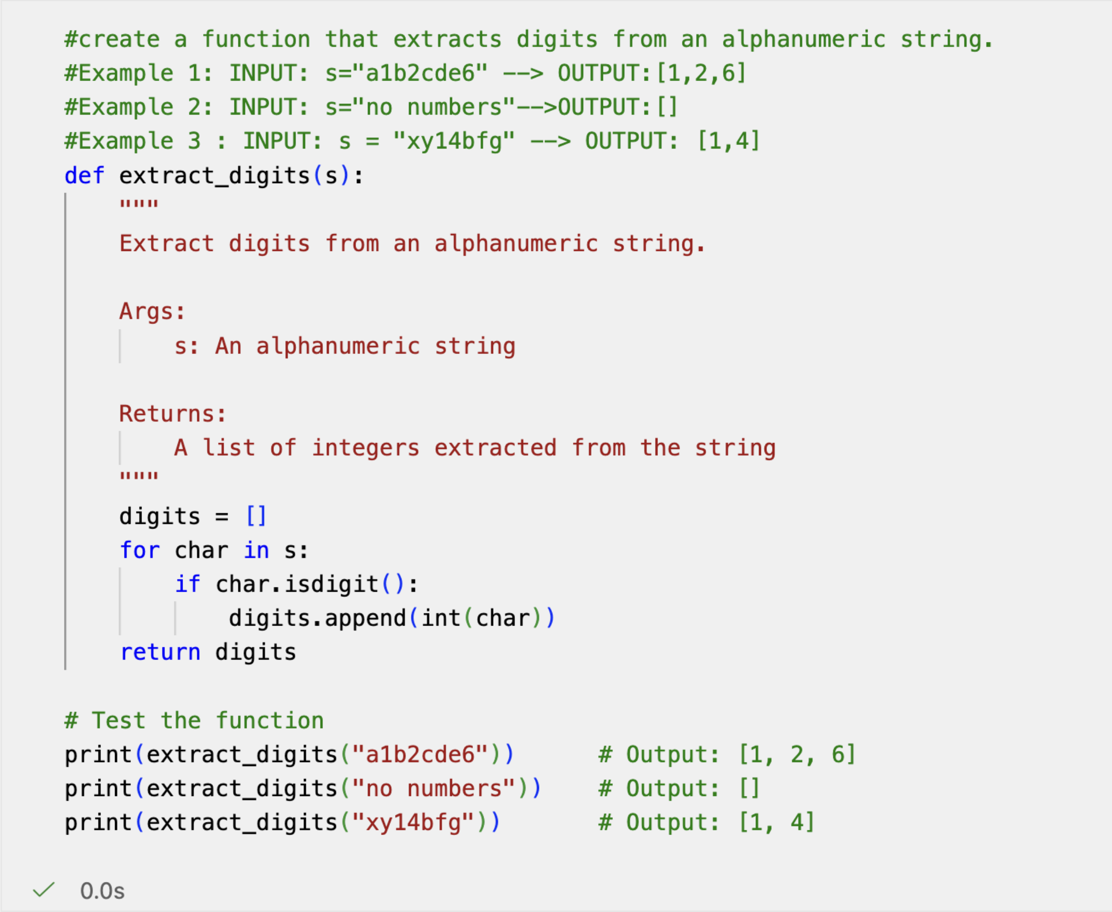
```
Sum of [2, 3, 4, 10, 17, 20, 29, 100]: 185
```

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

PROMPT:

create a function that extracts digits from an alphanumeric string.
Example 1: INPUT: s="a1b2cde6" --> OUTPUT:[1,2,6]
Example 2: INPUT: s="no numbers"-->OUTPUT:[]
Example 3 : INPUT: s = "xy14bfg" --> OUTPUT: [1,4]

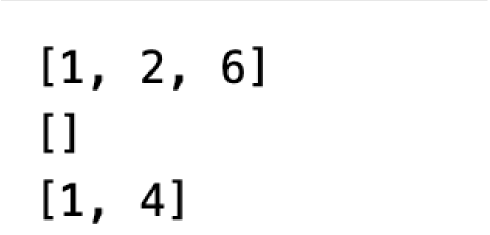
CODE:



```python
#create a function that extracts digits from an alphanumeric string.
#Example 1: INPUT: s="a1b2cde6" --> OUTPUT:[1,2,6]
#Example 2: INPUT: s="no numbers"-->OUTPUT:[]
#Example 3 : INPUT: s = "xy14bfg" --> OUTPUT: [1,4]
def extract_digits(s):
    """
    Extract digits from an alphanumeric string.

    Args:
        s: An alphanumeric string

    Returns:
        A list of integers extracted from the string
    """
    digits = []
    for char in s:
        if char.isdigit():
            digits.append(int(char))
    return digits

# Test the function
print(extract_digits("a1b2cde6"))      # Output: [1, 2, 6]
print(extract_digits("no numbers"))    # Output: []
print(extract_digits("xy14bfg"))       # Output: [1, 4]
```

✓ 0.0s

OUTPUT:

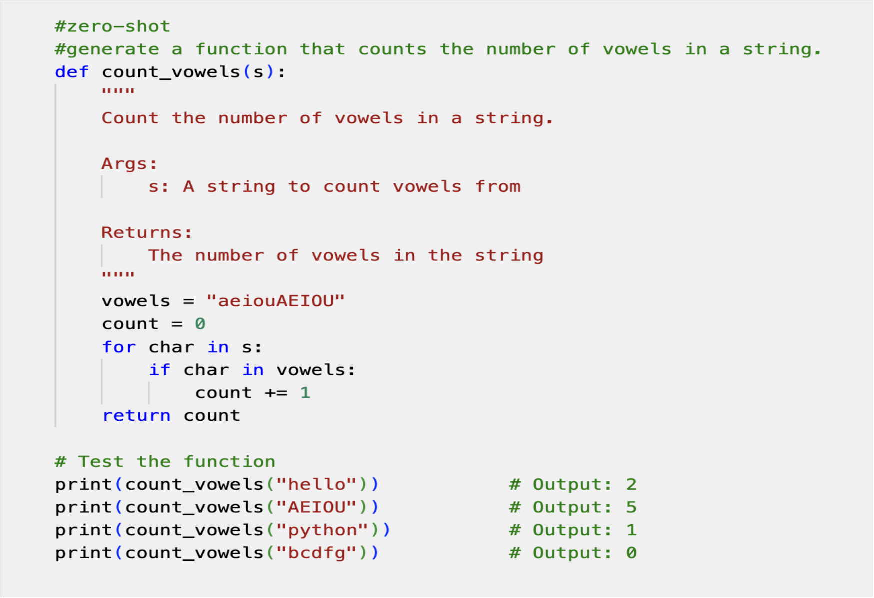


```
[1, 2, 6]
[]
[1, 4]
```

TASK 4 : Compare zero-shot vs few-shot prompting for generating a function that counts the number of vowels in a string.

ZERO-SHOT:

PROMPT: generate a function that counts the number of vowels in a string.



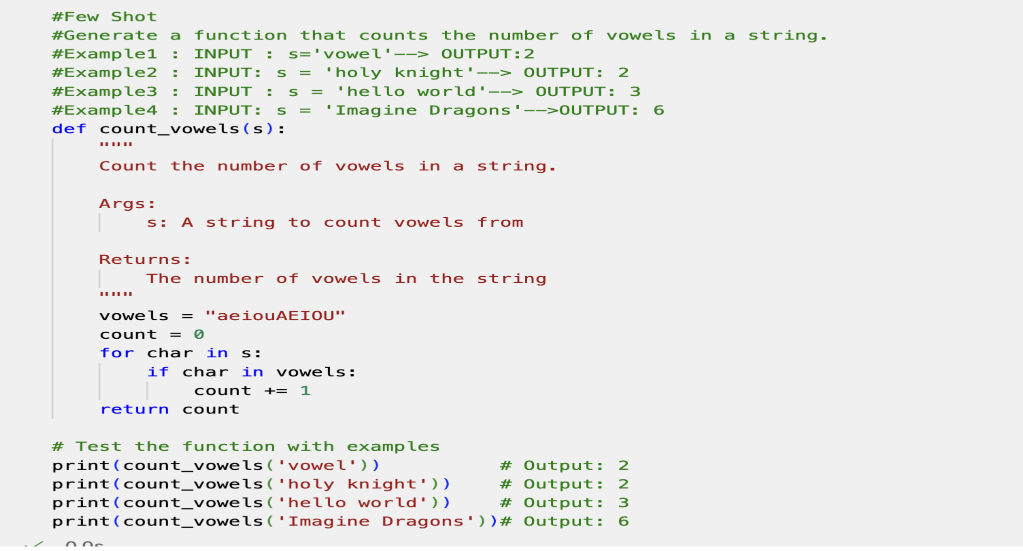
```
#zero-shot
#generate a function that counts the number of vowels in a string.
def count_vowels(s):
    """
    Count the number of vowels in a string.

    Args:
        s: A string to count vowels from

    Returns:
        The number of vowels in the string
    """
    vowels = "aeiouAEIOU"
    count = 0
    for char in s:
        if char in vowels:
            count += 1
    return count

# Test the function
print(count_vowels("hello"))           # Output: 2
print(count_vowels("AEIOU"))           # Output: 5
print(count_vowels("python"))          # Output: 1
print(count_vowels("bcdfg"))           # Output: 0
```

FEW-SHOT:

PROMPT:

Generate a function that counts the number of vowels in a string.

Example1 : INPUT : s='vowel'--> OUTPUT:2
Example2 : INPUT: s = 'holy knight'--> OUTPUT: 2
Example3 : INPUT : s = 'hello world'--> OUTPUT: 3
Example4 : INPUT: s = 'Imagine Dragons'-->OUTPUT: 6



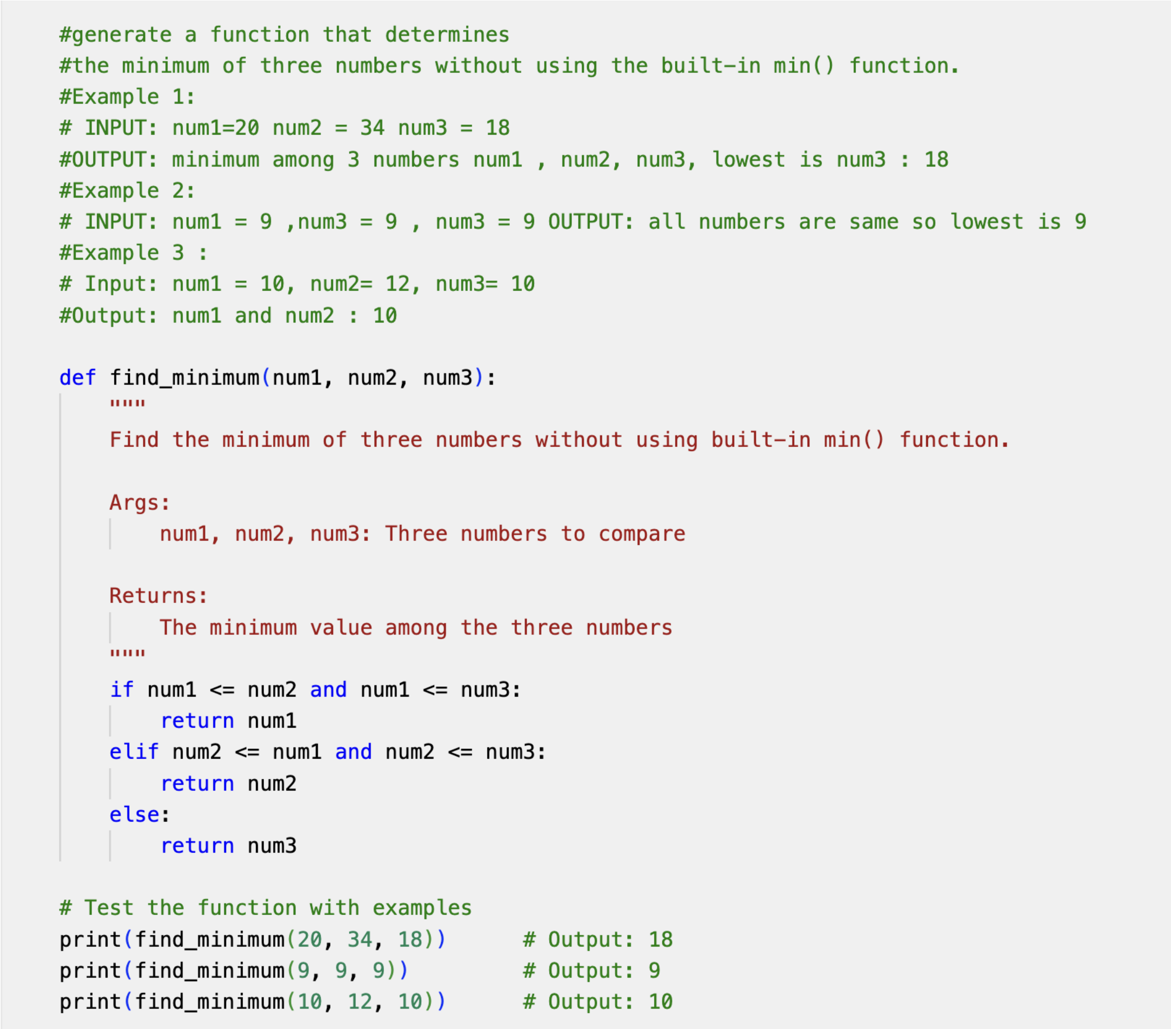
```
#Few Shot
#Generate a function that counts the number of vowels in a string.
#Example1 : INPUT : s='vowel'--> OUTPUT:2
#Example2 : INPUT: s = 'holy knight'--> OUTPUT: 2
#Example3 : INPUT : s = 'hello world'--> OUTPUT: 3
#Example4 : INPUT: s = 'Imagine Dragons'-->OUTPUT: 6
def count_vowels(s):
    """
    Count the number of vowels in a string.

    Args:
        s: A string to count vowels from

    Returns:
        The number of vowels in the string
    """
    vowels = "aeiouAEIOU"
    count = 0
    for char in s:
        if char in vowels:
            count += 1
    return count

# Test the function with examples
print(count_vowels('vowel'))          # Output: 2
print(count_vowels('holy knight'))    # Output: 2
print(count_vowels('hello world'))    # Output: 3
print(count_vowels('Imagine Dragons'))# Output: 6
```

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.

PROMPT: generate a function that determines the minimum of three numbers without using the built-in min() function.

Example 1:
INPUT: num1=20 num2 = 34 num3 = 18
OUTPUT: minimum among 3 numbers num1 , num2, num3, lowest is num3 : 18
Example 2:
INPUT: num1 = 9 ,num3 = 9 , num3 = 9
OUTPUT: all numbers are same so lowest is 9
Example 3 :
INPUT: num1 = 10, num2= 12, num3= 10
OUTPUT: num1 and num2 : 10

CODE AND OUTPUT:



```python
#generate a function that determines
#the minimum of three numbers without using the built-in min() function.
#Example 1:
# INPUT: num1=20 num2 = 34 num3 = 18
#OUTPUT: minimum among 3 numbers num1 , num2, num3, lowest is num3 : 18
#Example 2:
# INPUT: num1 = 9 ,num3 = 9 , num3 = 9 OUTPUT: all numbers are same so lowest is 9
#Example 3 :
# Input: num1 = 10, num2= 12, num3= 10
#Output: num1 and num2 : 10

def find_minimum(num1, num2, num3):
    """
    Find the minimum of three numbers without using built-in min() function.

    Args:
        num1, num2, num3: Three numbers to compare

    Returns:
        The minimum value among the three numbers
    """
    if num1 <= num2 and num1 <= num3:
        return num1
    elif num2 <= num1 and num2 <= num3:
        return num2
    else:
        return num3

# Test the function with examples
print(find_minimum(20, 34, 18))      # Output: 18
print(find_minimum(9, 9, 9))         # Output: 9
print(find_minimum(10, 12, 10))      # Output: 10
```