

SCHOOL OF COMPUTER SCIENCE AND ARTIFICIAL INTELLIGENCE		DEPARTMENT OF COMPUTER SCIENCE ENGINEERING	
Program Name: M.Tech. and MCA		Assignment Type: Lab	
Course Coordinator Name		Venkataramana Veeramsetty	
Course Code		Course Title	AI Assisted Problem Solving Using Python
Year/Sem	I/I	Regulation	R24
Date and Day of Assignment	Week3 - Monday	Time(s)	
Duration	2 Hours	Applicable to Batches	M.Tech. and MCA
AssignmentNumber: 4.3(Present assignment number)/24(Total number of assignments)			

Q.No.	Question	Expected Time to complete
1	<p>Lab 4: Advanced Prompt Engineering – Zero-shot, One-shot, and Few-shot Techniques</p> <p>Lab Objectives:</p> <ul style="list-style-type: none"> • To explore and apply different levels of prompt examples in AI-assisted code generation. • To understand how zero-shot, one-shot, and few-shot prompting affect AI output quality. • To evaluate the impact of context richness and example quantity on AI performance. • To build awareness of prompt strategy effectiveness for different problem types. <p>Lab Outcomes (LOs):</p> <p>After completing this lab, students will be able to:</p> <ul style="list-style-type: none"> • Use zero-shot prompting to instruct AI with minimal context. • Use one-shot prompting with a single example to guide AI code generation. • Apply few-shot prompting using multiple examples to improve AI 	Week3 - Monday

	<p>responses.</p> <ul style="list-style-type: none"> • Compare AI outputs across the three prompting strategies. <p>Task Description#1</p> <ul style="list-style-type: none"> • Zero-shot: Prompt AI to write a function that checks whether a given year is a leap year. <p>Expected Output#1</p> <ul style="list-style-type: none"> • AI-generated function with no examples provided <p>Task Description#2</p> <ul style="list-style-type: none"> • One-shot: Give one input-output example to guide AI in writing a function that converts centimeters to inches. <p>Expected Output#2</p> <ul style="list-style-type: none"> • Function with correct conversion logic <p>Task Description#3</p> <ul style="list-style-type: none"> • Few-shot: Provide 2–3 examples to generate a function that formats full names as “Last, First”. <p>Expected Output#3</p> <ul style="list-style-type: none"> • Well-structured function respecting the examples <p>Task Description#4</p> <ul style="list-style-type: none"> • Compare zero-shot and few-shot prompts for writing a function that counts the number of vowels in a string. <p>Expected Output#4</p> <ul style="list-style-type: none"> • Functional output and comparative reflection <p>Task Description#5</p> <ul style="list-style-type: none"> • Use few-shot prompting to generate a function that reads a .txt file and returns the number of lines. <p>Expected Output#5</p> <ul style="list-style-type: none"> • Working file-processing function with AI-guided logic <p>Note: Report should be submitted a word document for all tasks in a single document with prompts, comments & code explanation, and output and if required, screenshots</p> <p>Evaluation Criteria:</p>	
--	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--

	Criteria	Max Marks		
	Zero Shot (Task #1)	2.5		
	One Shot (Task#2)	2.5		
	Few Shot (Task#3 & Task #5)	2.5		
	Comparison (Task#4)	2.5		
	Total	10 Marks		

Task Description#1

- Zero-shot: Prompt AI to write a function that checks whether a given year is a leap year.

PROMPT: Write a program that takes a year from the user and prints whether it is a leap year or not.

```
C: > Users > rimsha > OneDrive > Desktop > Mohammed Farnas Ali Mudabbir > LAB 4 > TASK 1 > ✎ TASK 1.py > ...
1 def is_leap(year: int) -> bool:
2     """ Return True if year is a leap year, otherwise False.
3     Leap year rule:
4         - divisible by 4 and not divisible by 100, OR
5         - divisible by 400 """
6     return (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)
7
8 def get_year_from_user(prompt: str = "Enter a year: ") -> int:
9     """Prompt the user until a valid integer year is entered and return it."""
10    while True:
11        s = input(prompt).strip()
12        try:
13            year = int(s)
14            return year
15        except ValueError:
16            print("Invalid input. Please enter a valid integer year.")
17
18
19 def main():
20     year = get_year_from_user()
21     if is_leap(year):
22         print(f"{year} is a leap year.")
23     else:
24         print(f"{year} is not a leap year.")
25
26
27 if __name__ == "__main__":
28     main()
```

Expected Output#1

- AI-generated function with no examples provided

Practical output:

```
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1> Enter a year: 2012
2012 is a leap year.
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1>
```

Task Description#2

- One-shot: Give one input-output example to guide AI in writing a function that converts centimeters to inches.

PROMPT: Write a function that converts a value in centimeters to inches.

```
C: > Users > rimssha > OneDrive > Desktop > Mohammed Farnas Ali Mudabbir > LAB 4 > TASK 1 > TASK 2.py > ...
1 import sys
2
3 def cm_to_inches(cm):
4     """Convert centimeters to inches. Returns value rounded to 3 decimal
5     try:
6         value = float(cm)
7     except (TypeError, ValueError):
8         raise ValueError("Input must be a number")
9     inches = value / 2.54
10    return round(inches, 3)
11
12
13 if __name__ == "__main__":
14
15     if len(sys.argv) > 1:
16         inp = sys.argv[1]
17     else:
18         inp = input("Enter centimeters: ").strip()
19
20     try:
21         print(cm_to_inches(inp))
22     except ValueError as e:
23         print("Error:", e)
```

CHAT

Write a function to convert centimeters to inches.

import sys

> Used 1 reference

> Drafting function message...

Updated function that converts centimeters to inches (accepts numbers or numeric strings, returns float rounded to 3 decimals):

```
// ...existing code...
def cm_to_inches(cm: float) -> float
    """Convert centimeters to inches
    Returns value rounded to 3 decimal
    try:
        value = float(cm)
    except (TypeError, ValueError):
        raise ValueError("Input must be a number")
    return round(value / 2.54, 3)
```

TASK 2.py

Add context (#), extensions (@), commands (/)

Ask Auto

Expected Output#2

- Function with correct conversion logic

Practical output:

```

Enter centimeters: 20
7.874
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1> 

```

Task Description#3

Few-shot: Provide 2–3 examples to generate a function that formats full names as “Last, First”.

PROMPT: Write a function that asks for a full name and prints it as ‘Last, First’.

```

C: > Users > rimsha > OneDrive > Desktop > Mohammed Farnas Ali Mudabbir > LAB 4 > TASK 1 > TASK 3.py > ...
1 def get_formatted_name(prompt="Enter full name: "):
2     """
3         Prompts the user for a full name and returns it formatted as "Last,
4         If only one name is entered, it is returned unchanged.
5         """
6     full = input(prompt).strip()
7     if not full:
8         return ""
9     parts = full.split()
10    if len(parts) == 1:
11        return parts[0]
12    last = parts[-1]
13    first = " ".join(parts[:-1])
14    return f"{last}, {first}"
15
16 if __name__ == "__main__":
17     result = get_formatted_name()
18     if result:
19         print(result)

// ...existing code...
def get_formatted_name(prompt="Enter full name: "):
    """
    Prompts the user for a full name
    If only one name is entered, it is returned unchanged.
    """
    full = input(prompt).strip()
    if not full:
        return ""
    parts = full.split()
    if len(parts) == 1:
        return parts[0]

```

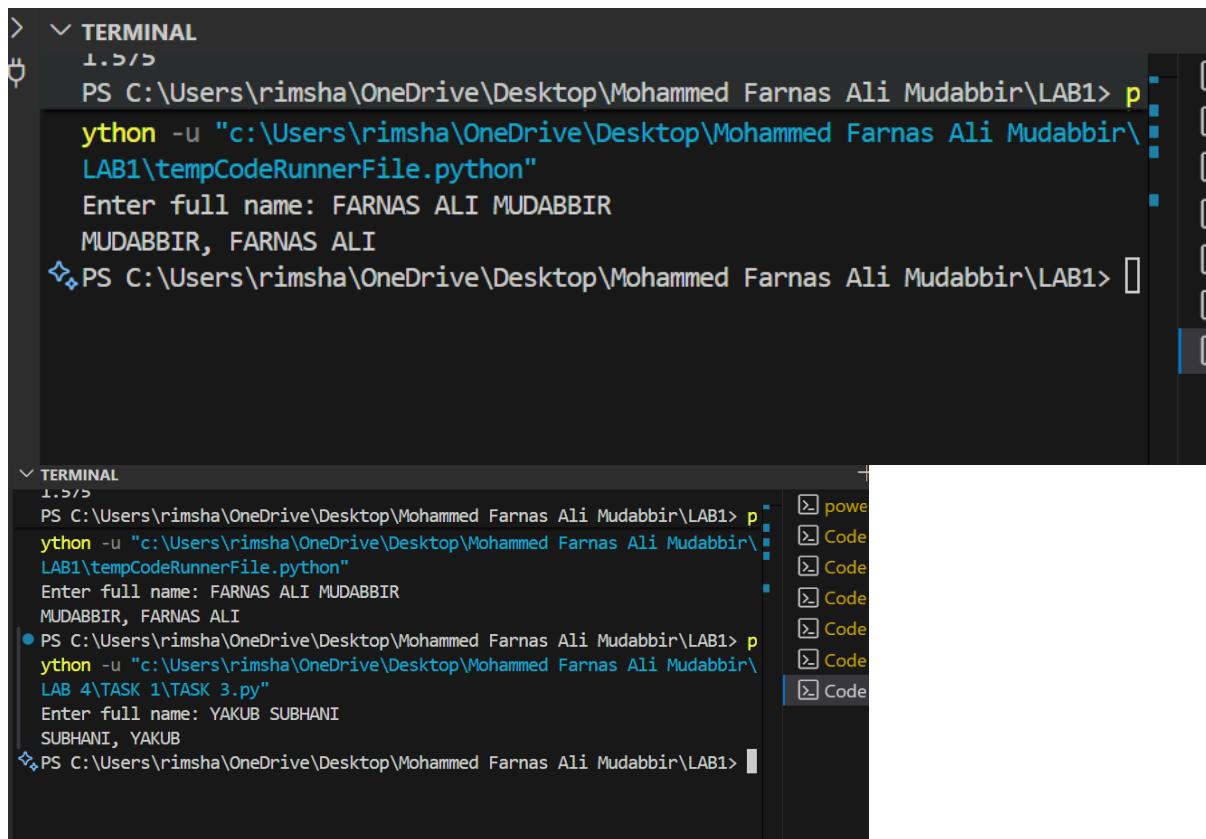
Expected Output#3

- Well-structured function respecting the examples

Practical output:

```
> ▾ TERMINAL
  ↴ 1.5/5
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1> p
  ython -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\
  LAB1\tempCodeRunnerFile.python"
Enter full name: FARNAS ALI MUDABBIR
MUDABBIR, FARNAS ALI
↳ PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1> █

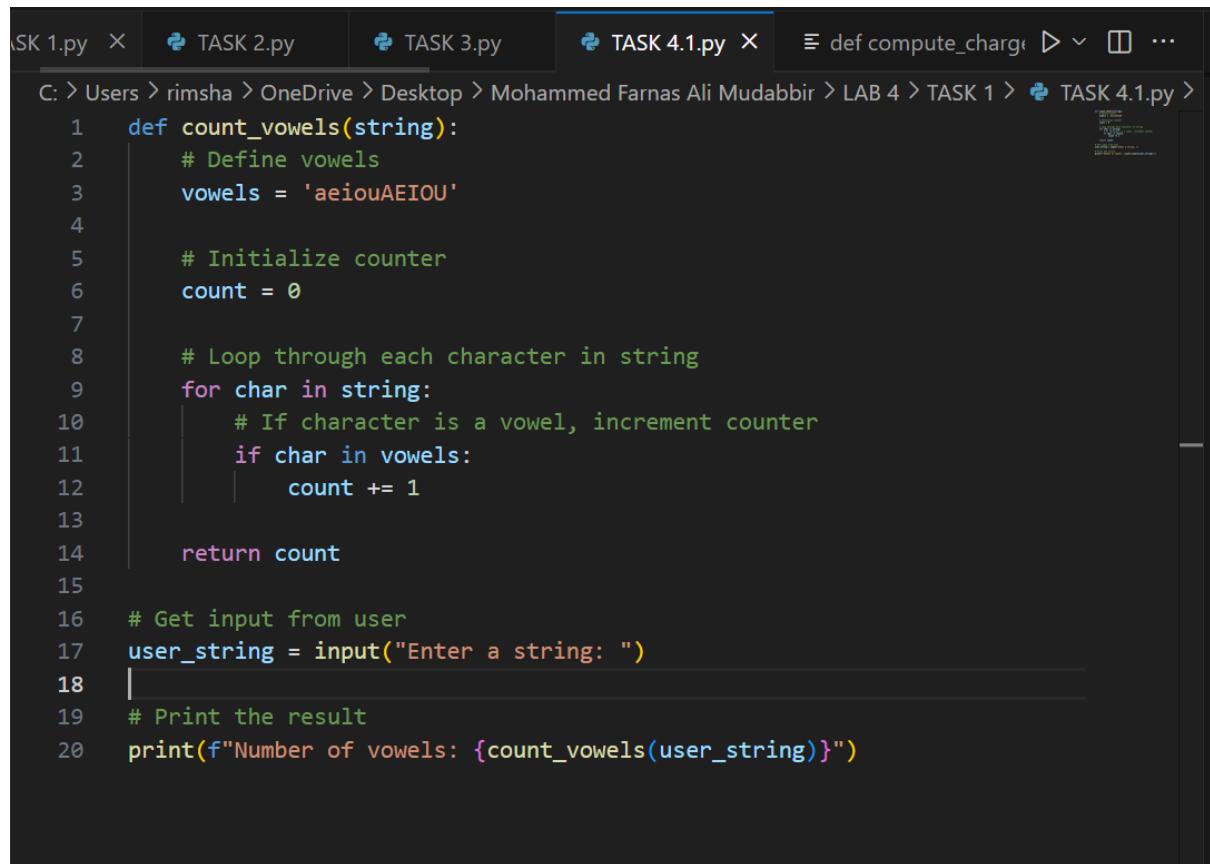
▀ TERMINAL
  ↴ 1.5/5
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1> p
  ython -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\
  LAB1\tempCodeRunnerFile.python"
Enter full name: FARNAS ALI MUDABBIR
MUDABBIR, FARNAS ALI
● PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1> p
  ython -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\
  LAB 4\TASK 1\TASK 3.py"
Enter full name: YAKUB SUBHANI
SUBHANI, YAKUB
↳ PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1> █
```



Task Description#4

Compare zero-shot and few-shot prompts for writing a function that counts the number of vowels in a string.

ZERO-SHOT PROMPT: Write a function that takes a string as input and returns the number of vowels in it.



The screenshot shows a code editor with multiple tabs open. The active tab is 'TASK 4.1.py'. The code in the editor is as follows:

```
C: > Users > rimsha > OneDrive > Desktop > Mohammed Farnas Ali Mudabbir > LAB 4 > TASK 1 > TASK 4.1.py >
1 def count_vowels(string):
2     # Define vowels
3     vowels = 'aeiouAEIOU'
4
5     # Initialize counter
6     count = 0
7
8     # Loop through each character in string
9     for char in string:
10         # If character is a vowel, increment counter
11         if char in vowels:
12             count += 1
13
14     return count
15
16 # Get input from user
17 user_string = input("Enter a string: ")
18
19 # Print the result
20 print(f"Number of vowels: {count_vowels(user_string)}")
```

FEW-SHOT PROMPT: Write a user input function that reads a string and prints how many vowels it has.

```
C: > Users > rimsha > OneDrive > Desktop > Mohammed Farnas Ali Mudabbir > LAB 4 > TASK 1 > TASK 4.2.py
 1  def count_vowels(text: str) -> int:
 2      """
 3          Return the number of vowels (a, e, i, o, u) in the given string.
 4          Counts both uppercase and lowercase vowels.
 5      """
 6
 7      if text is None:
 8          return 0
 9      vowels = set("aeiouAEIOU")
10
11
12     # Example usage / quick tests
13     if __name__ == "__main__":
14         samples = [
15             "Hello, World!",
16             "AEIOU aeiou",
17             "",
18             "Python Programming",
19             None
20         ]
21         for s in samples:
22             print(repr(s), "->", count_vowels(s))
```

Difference Between Zero-Shot and Few-Shot Prompts

- Zero-Shot Prompt:
The model is given only the instruction (no examples).
Example: Write a function to count vowels.
- Few-Shot Prompt:
The model is given one or more examples along with the instruction.
Example: Showing sample input-output before asking to write the function.

Expected Output#4

- Functional output and comparative reflection

Practical output:

ZERO-SHOT:

```
● ython -u "c:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1\tempCodeRunnerFile.py"
Enter a string: FARNAS ALI MUDABBIR
Number of vowels: 7
❖ PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1> []
```

FEW-SHOT:

```
'Hello, World!' -> 3
'AEIOU aeiou' -> 10
'' -> 0
'Python Programming' -> 4
None -> 0
❖ PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1> []
```

Task Description#5

- Use few-shot prompting to generate a function that reads a .txt file and returns the number of lines.

```
C: > Users > rimsha > OneDrive > Desktop > Mohammed Farnas Ali Mudabbir > LAB 4 > TASK 1 > task 5.py > count_lines_in_file
 1  # ...existing code...
 2  def _count_lines_in_file(path: str) -> int:
 3      """Return the number of lines in the given file path."""
 4      with open(path, "r", encoding="utf-8") as f:
 5          return sum(1 for _ in f)
 6
 7  def count_lines_in_file(path: str) -> int:
 8      """
 9          Return the number of lines in the given .txt file path.
10          Appends '.txt' if the extension is omitted and raises exceptions on error.
11      """
12
13      if not path:
14          raise ValueError("path must be a non-empty string")
15      path = path.strip().strip(['"', "'"])
16      if not path.lower().endswith(".txt"):
17          path += ".txt"
18      return _count_lines_in_file(path)
19
20  def count_lines_from_user_input(prompt: str = "Enter path to a .txt file: ") -> int:
21      """
22          Prompt the user for a .txt filename (or path), open it and return the number of lines.
23          The function will append '.txt' if the user omits the extension and will retry on errors
24      """
25
26  C: > Users > rimsha > OneDrive > Desktop > Mohammed Farnas Ali Mudabbir > LAB 4 > TASK 1 > task 5.py > count_lines_in_file
27  def count_lines_from_user_input(prompt: str = "Enter path to a .txt file: ") -> int:
28      """
29          Prompt the user for a .txt filename (or path), open it and return the number of lines.
30          The function will append '.txt' if the user omits the extension and will retry on errors
31      """
32
33      while True:
34          try:
35              user_input = input(prompt).strip().strip(['"', "''])
36          except (KeyboardInterrupt, EOFError):
37              raise # let caller handle interruption
38
39          if not user_input:
40              print("No filename entered. Please try again.")
41              continue
42
43          if not user_input.lower().endswith(".txt"):
44              user_input += ".txt"
45
46          try:
47              return _count_lines_in_file(user_input)
48          except FileNotFoundError:
49              print(f"File not found: {user_input}")
50          except IsADirectoryError:
51              print(f"Path is a directory, not a file: {user_input}")

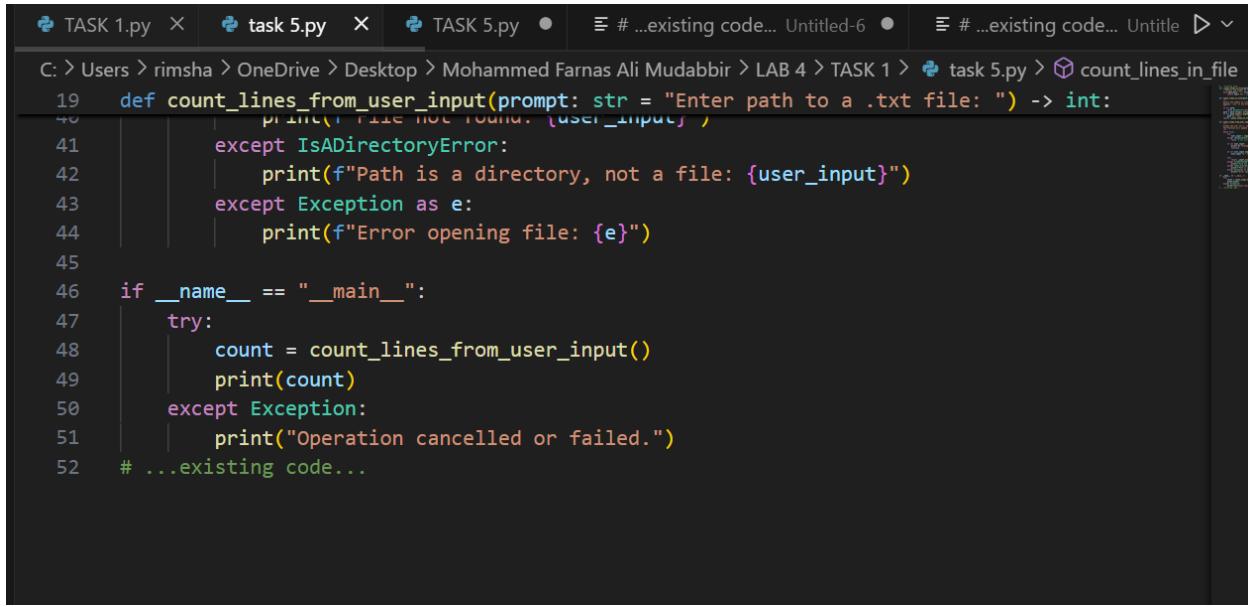

I'll modify the
input and display
code:

class Student:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def display_info(self):
        print(f"Name: {self.name}, Age: {self.age}")

def main():
    try:
        student = Student("John Doe", 20)
        student.display_info()
    except Exception as e:
        print(f"An error occurred: {e}")

if __name__ == "__main__":
    main()
```



A screenshot of a code editor interface. At the top, there are several tabs: 'TASK 1.py' (closed), 'task 5.py' (active), 'TASK 5.py' (closed), '# ...existing code...', 'Untitled-6' (closed), '# ...existing code...', 'Untitled' (closed). The main area shows Python code for counting lines in a file. The code includes error handling for non-existent files, directories, and other exceptions. It also includes a main block that tries to run the function and prints the result or an error message.

```
C: > Users > rimsha > OneDrive > Desktop > Mohammed Farnas Ali Mudabbir > LAB 4 > TASK 1 > task 5.py > count_lines_in_file
19     def count_lines_from_user_input(prompt: str = "Enter path to a .txt file: ") -> int:
20         print(f"File not found: {user_input}")
21     except IsADirectoryError:
22         print(f"Path is a directory, not a file: {user_input}")
23     except Exception as e:
24         print(f"Error opening file: {e}")
25
26 if __name__ == "__main__":
27     try:
28         count = count_lines_from_user_input()
29         print(count)
30     except Exception:
31         print("Operation cancelled or failed.")
32 # ...existing code...
```

Expected output:

```
Enter path to a .txt file: "C:\Users\rimsha\OneDrive\Desktop\SAMPLE_txt.txt"
\Desktop\Mohammed Farnas Ali Mudabbir\LAB1\tempCodeRunnerFile.python"
Enter path to a .txt file: "C:\Users\rimsha\OneDrive\Desktop\SAMPLE_txt.txt"
Enter path to a .txt file: "C:\Users\rimsha\OneDrive\Desktop\SAMPLE_txt.txt"
7
PS C:\Users\rimsha\OneDrive\Desktop\Mohammed Farnas Ali Mudabbir\LAB1> []
```