Mark as done

Description Submission view

Available from: Thursday, 4 September 2025, 9:20 AM
Due date: Thursday, 4 September 2025, 10:50 AM
Requested files: p1.py, p2.py, p3.py, p4.py, p5.py (Download)
Type of work: Individual work

Problem 1 - Liters to Gallons Converter

Description:
Write a program to convert a volume from liters to U.S. gallons.

Concept:
This problem assesses fundamental skills: reading user input, converting the input to a numeric type (**typecasting**), applying a given mathematical formula, and formatting the output to a specific number of decimal places.

Task:
Your program must produce output that exactly matches the format specified. Pay close attention to all text, spaces, and punctuation in your input prompts and your final output.
1. Prompt the user for a volume with the message: `Enter volume in Liters: `.
2. Convert the input to a floating-point number.
3. Calculate the volume in gallons using the conversion factor: `1 liter = 0.264172 U.S. gallons`.
4. Print the result in the format `Volume in U.S. gallons: [gallons_value]`, with the gallons value rounded to two decimal places.

Note: To *round* your final answer to two decimal places, you can use the built-in `round()` function. It takes two arguments: the number to round and the number of decimal places.
For example, `round(123.4567, 2)` will result in `123.46`.

Example:
*(Text in **bold** is what the user types.)*
```
Enter volume in Liters: 20
Volume in U.S. gallons: 5.28
```

(Another example)
```
Enter volume in Liters: 3.785
Volume in U.S. gallons: 1.00
```

Restrictions:
The input will be a positive number.
No external libraries are necessary.

Problem 2 - Sum of Special Numbers

Description:
Write a program that takes two positive integers, `a` and `b`, as input and prints the sum of all special numbers in the range from `a` to `b-1`. Special numbers are ***odd***, ***non-divisible by 5***, and whose ***sum of digits*** *is divisible by either **2 or 3***.
Range of input: `1 <= a < b <= 1000`

Concept:
This problem tests the ability to use loops with conditional checks to sum numbers based on multiple conditions.

Task:
Your program must produce output that exactly matches the format specified. Pay close attention to all text, spaces, and punctuation in your input prompts and your final output.
1. Prompt the user with: `Enter two numbers: `.
2. Loop through numbers starting from the first input (`a`) up to, but not including, the second input (`b`).
3. Calculate and print the sum of all special numbers in this range.

Example:

```
Enter two numbers: 3 20
Result: 72
```

Explanation:
Special numbers between 3 & 20 : *3, 9, 11, 13, 17, 19*
Sum of special numbers: 3 + 9 + 11 + 13 + 17 + 19 = 72

Restrictions:
The input will be two positive integers where `1 <= a < b <= 1000`
No external libraries are necessary.

Problem 3 - Reverse Number Pattern with Nested Loops

Description:
Write a program that takes a positive integer `n` as input and prints a number pattern where the first line contains the number `n` repeated `n` times, the second line contains the number `n-1` repeated `n-1` times, and so on, until the last line contains only the number `1`. The number of spaces after each number must be equal to the number itself.

Concept:
This problem tests the ability to use nested loops with decreasing counts to generate structured number patterns.

Task:
Your program must produce output that exactly matches the format specified. Pay close attention to all text, spaces, and punctuation in your input prompts and your final output.
1. Prompt the user with: `Enter a number:`
2. Use nested loops to print the required number pattern.
3. Print the pattern row by row. In each row, there must be spaces after every number, and the number of spaces must equal the number itself.

Example:

```
Enter a number:3
3   3   3
2  2
1
```

Restrictions:
The input will be a positive integer.
No external libraries are necessary.

Hint:
Use the command: *print('x', end=" ")* to add space(s) instead of end of line after character *'x'*

Problem 4: IITD Vacation Checker

Description
IIT Delhi vacations in 2025 are scheduled as:

- 15th May 2025 to 20th July 2025
- 3rd December 2025 to 31st December 2025

Write a program to check whether a given date falls in the vacation period. You can assume the input will be a valid date.

Concept
This problem tests **range checking using nested conditions**.

Task

1. Prompt the user with:
   `Enter day, month and year:`
2. Read three integers.
3. Apply rules:
   - If year < 2025 → print `Enter date from current year`

- If year >= 2026 → print `Calendar has not been decided yet`
- Else check if the date falls in vacation ranges. Print `Vacation` or `Not a vacation`.

Input format
Prompt before input:
`Enter day, month and year:`

Output format
Output must be one of:

- `Vacation`
- `Not a vacation`
- `Enter date from current year`
- `Calendar has not been decided yet`

Examples

Example 1

```
Enter day, month and year: 16 5 2025

Vacation
```

Example 2

```
Enter day, month and year: 10 8 2025

Not a vacation
```

Example 3

```
Enter day, month and year: 12 4 2024

Enter date from current year
```

Restrictions:

- The input will be a valid date.
- No external libraries are necessary.

Problem 5: Fibonacci Series

Description
Write a program to generate the Fibonacci series up to `n` terms.
The Fibonacci sequence is defined as:

- First term = 0
- Second term = 1
- Every next term = sum of the previous two terms

Task
Read an integer `n` and print the Fibonacci series up to `n` terms. You can assume input will always be greater than 0 .

Input format
Prompt before input:
`Enter n:`

Output format
Output must be the required series.

Examples

Example 1

```
Enter n: 5
```

```
0 1 1 2 3
```

Example 2

```
Enter n: 7
```

```
0 1 1 2 3 5 8
```

Restrictions

- Use only loops and conditionals (no libraries).
- Assume `n > 0`.

Requested files

p1.py

```
# write your code here below
```

p2.py

```
# write your code here below
```

p3.py

```
# write your code here below
```

p4.py

```
# write your code here below
```

p5.py

```
# write your code here below
```

VPL