Done

Description Submission view

Available from: Monday, 1 September 2025, 9:15 AM
Due date: Monday, 1 September 2025, 10:45 AM
Requested files: p1.py, p2.py, p3.py, p4.py, p5.py (Download)
Type of work: Individual work

Problem 1 - Circle Area Calculator

Description:
Write a program that calculates the area of a circle based on its radius provided by the user.

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 the circle's radius with the message: `Enter the radius of the circle:` .
2. Convert the input to a floating-point number.
3. Calculate the area using the formula: `Area = π * r^2`. Use `3.14159` as the value for π.
4. Print the final area in the format `Area: [calculated_area]`, 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 the radius of the circle: 5
Area: 78.54
```

(Another example)
```
Enter the radius of the circle: 12.5
Area: 490.87
```

Restrictions:
The input will be a positive number.
You must use `3.14159` for the value of π.
No external libraries are necessary.

Problem 2 - Sum of Squares of Numbers

Description:
Write a program that takes two positive integers, `a` and `b`, as input and prints the sum of the squares of all numbers from `a` to `b-1`. Range of input: 1 <= a < b <= 1000

Concept:
This problem tests the ability to use **loops with arithmetic operations** over a given range of numbers.

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 the squares of these numbers.

Example:

```
Enter two numbers:3 7
Result: 99
```

```
Explanation:
3^2 + 4^2 + 5^2 + 6^2 = 9 + 16 + 25 + 36 = 99
```

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

Problem 3 - Number Pattern with Nested Loops

Description:
Write a program that takes a positive integer `n` as input and prints a number pattern where each line `i` (from 1 to `n`) contains the number `i`, repeated `i` times.

Concept:
This problem tests the ability to use **nested loops** to generate structured 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 is a space after every number.

Example:

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

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

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

Problem 4: Descending Order of Three Numbers

Description

Write a program that reads three integers and prints them in descending order (from largest to smallest).

Note: In this context, "descending" means **non-increasing** order.

Concept
This problem tests your understanding of **comparisons and conditional logic** without built-in helpers.

Task

1. Prompt the user with:
   `Enter three numbers:`
2. Read three integers in one line.
3. Use if–else conditions to rearrange them in descending order.
4. Print the numbers on one line, separated by spaces, after:
   `Descending order:`

Input format
Prompt before input:
`Enter three numbers:`

Output format
Output must begin with:
`Descending order:`

Examples

Example 1
```
Enter three numbers: 5 2 9

Descending order: 9 5 2
```

Example 2

```
Enter three numbers: 3 1 2

Descending order: 3 2 1
```

Restrictions
Do not use any built-in Python functions.

Problem 5: Factorial Series

Description
Generate the series of factorials up to `n` terms.
The factorial of a number `i` is given by:
`i! = 1 × 2 × 3 × … × i`

Task
Read an integer `n` and print the series of factorials till `n` terms. The input will always be an integer greater than 0.

Input format
Prompt before input:
`Enter n:`

Output format
Output must be the required series:

Examples

Example 1
```
Enter n: 4

1 2 6 24
```

Example 2

```
Enter n: 6

1 2 6 24 120 720
```

Restrictions

- The input will always be greater than zero(`n>0`).
- Do not use any in-built functions.
- Use only basic loops and multiplication.

Requested files

p1.py

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

p2.py

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

p3.py

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

p4.py

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

p5.py

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

VPL