Mark as done

Description Submission view

Available from: Wednesday, 3 September 2025, 9:15 AM
Due date: Wednesday, 3 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 - Kilometer to Miles Converter

Description:
Write a program to convert a distance from kilometers to miles.

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 distance with the message: `Enter distance in Kilometers: `.
2. Convert the input to a floating-point number.
3. Calculate the distance in miles using the conversion factor: `1 km = 0.621371 miles`.
4. Print the result in the format `Distance in Miles: [miles_value]`, with the miles 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 distance in Kilometers: 100
Distance in Miles: 62.14
```

(Another example)
```
Enter distance in Kilometers: 3.5
Distance in Miles: 2.17
```

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

Problem 2 - Sum of Numbers with Digit-Sum Divisible by 2 or 3

Description:

Write a program that takes two positive integers, `a` and `b`, as input and prints the sum of all numbers in the range from `a` to `b-1` whose **sum of digits** is divisible by **2 or 3**.
1. Each number is at most 3 digits.
2. Range of input: `1 <= a < b <= 1000`.

Concept:
1. Iterating through a numeric range using loops.
2. Extracting digits of a number and calculating digit sum.
3. Using modulus (`%`) operator for divisibility checks.
4. Applying conditions to decide inclusion in a cumulative sum.

?

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. For each number, compute the sum of its digits. If the digit sum is divisible by 2 **or** 3, add the number to the total sum.
4. Print the result in the following format: **Result: <sum>**

Example:
```
Enter two numbers:10 20
Result: 105
```

Explanation:
Valid numbers = 11, 12, 13, 15, 17, 18, 19
Sum = 105

Restrictions:
The input will be two positive integers where `1 <= a < b <= 1000`.
Each number considered will have **at most 3 digits**.
No external libraries are necessary.

Problem 3 - Decreasing Number Pattern with Nested Loops (Formatted)

Description:
Write a program that takes a positive integer `n` as input and prints a number pattern. The first line must contain numbers from `1` to `n`. Each subsequent line must contain numbers from `1` up to one less than the previous line's maximum, continuing until the last line contains only `1`. Each number in the pattern must be followed by a specific number of spaces based on its digit count.

Concept:
This problem requires the application of **nested loops** to generate structured number patterns with decreasing ranges. Additionally, it demands the implementation of **conditional formatting** to control the spacing after each number based on whether it has one or two digits.

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. Implement nested loops to construct and print the required number pattern.
3. The outer loop must iterate to control the rows, and the inner loop must print numbers from `1` up to the current maximum value for that row.
4. For each number printed:
- If the number has **1 digit**, it must be followed by **one space**.
- If the number has **2 digits**, it must be followed by **two spaces**.
5. After the inner loop completes printing all numbers for a given row, a newline character must be printed to advance to the next line. (You can use `print()` in the outer loop after the inner loop)

Example:
```
Enter a number:12
1 2 3 4 5 6 7 8 9 10  11  12
1 2 3 4 5 6 7 8 9 10  11
1 2 3 4 5 6 7 8 9 10
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8
1 2 3 4 5 6 7
1 2 3 4 5 6
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

Restrictions:
The input will be a positive integer, and `n` will be at most 99 (numbers will have at most 2 digits).
No external libraries are necessary.

Problem 4: Airline Luggage Payment

Description
Write a program to calculate extra luggage payment for airline passengers based on weight and ticket class.

Concept
This problem checks **if–else decision making** with multiple conditions.

Task

1. Prompt the user with:
   `Enter luggage weight and ticket class:`
2. Read the weight (integer) and ticket class (`Economy` or `Business`).
3. Apply rules:
   - Economy: free up to 15 kg.
     - Extra ≤ 5 → ₹100 per kg.
     - Extra > 5 → ₹200 per kg.
   - Business: free up to 25 kg. Extra → ₹150 per kg.
4. Print the payment amount.

Input format
Prompt before input:
`Enter luggage weight and ticket class:`

Output format
Output must begin with:
`Payment:`

Examples

Example 1

```
Enter luggage weight and ticket class: 18 Economy

Payment: 300
```

Example 2

```
Enter luggage weight and ticket class: 30 Business

Payment: 750
```

Restrictions

- Do not use any built-in Python functions.
- You can asuume input weight will always be non-negative integer(n>=0).

Problem 5: Product of Previous Two (Modified Series)

Description
Generate a series where:

- First term = 1
- Second term = 2
- Every next term = product of the previous two terms

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

Input format
Prompt before input:
`Enter n:`

Output format
Output must be the required series.

Examples

Example 1

```
Enter n: 6

1 2 2 4 8 32
```

Example 2

```
Enter n: 5

1 2 2 4 8
```

Restrictions

- Use loops and multiplication only.
- The input, `n>0`.

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

VPL