Problem Solving Exercises

The LiFi-project

Prof. Dr. Nicolas Meseth

For the following problems, apply the structured approach you learned in the course and solve the problem with a Python program.

1. **Define the problem**: Clearly state the problem that you want to solve. What do you want the program to do?
2. **Identify inputs and outputs**: What information does the program need to solve the problem? What information does the program need to produce as output?
3. **Plan the solution**: Break down the problem into smaller, manageable steps. What are the different tasks the program needs to perform to solve the problem? Write out a plan on paper or whiteboard before jumping into writing the code.
4. **Translate into code**: Write the code to implement the plan. Use a programming language (like Python) to translate the plan into code. You can start with comments and later fill in the code.
5. **Test and refine**: Test the program with different inputs to make sure it works correctly. Refine the program if necessary to handle any edge cases or unexpected inputs.

Easy Problems

Problem 1: Costs for Road Trip

You are planning a road trip and want to calculate the total cost of gas for the trip. Write a program that takes user input for the distance of the trip, the average miles per gallon of the car, and the price per gallon of gas, and calculates the total cost of gas for the trip.

Problem 2: Circle Area and Circumference

You want to create a program that calculates the area and circumference of a circle. Write a program that takes user input for the radius of a circle and calculates the area and circumference.

Problem 3: Farm Animals

Write a Python program that helps a farmer manage their livestock. The program should prompt the user to enter the number of cows, pigs, and chickens on the farm, and then calculate the total number of legs on the farm. Cows have 4 legs, pigs have 4 legs, and chickens have 2 legs.

Problem 4: Calories

Write a Python program that calculates the total number of calories in a food item based on its macronutrient composition. The program should prompt the user to enter the number of grams of fat, protein, and carbohydrates in the food item, and then calculate the total number of calories in the food item using the following formulas:

- 1 gram of fat contains 9 calories
- 1 gram of protein contains 4 calories
- 1 gram of carbohydrates contains 4 calories

Problem 5: Ticket Costs

You want to create a program that calculates the cost of a movie ticket based on the age of the person buying the ticket. Write a program that takes user input for the age of the person and calculates the cost of the ticket based on the following criteria:

- If the person is 12 years old or younger, the ticket costs €5.
- If the person is between 13 and 64 years old (inclusive), the ticket costs €10.
- If the person is 65 years old or older, the ticket costs €7.

Medium Problems

Problem 6: Guess the Number

You want to create a program that generates a random whole number and asks the user to guess the number. The program should provide feedback to the user indicating whether the guess is too high or too low, and allow the user to keep guessing until they correctly guess the number.

Problem 7: Simple Statistics

You want to create a program that takes user input for a list of numbers, and then calculates the sum, average, min and max of the numbers in the list. Write a program that prompts the user to enter a list of numbers, separates them using commas, and then calculates the sum and average of the numbers.

Problem 8: Convert Text to Lowercase

You want to create a program that takes an input from the user and converts it to lowercase letters only. Do not use any built-in or external string functions such as `lower()`. Instead, rely on your knowledge of the ASCII code system. You can use the `ord()` function to get the decimal number of a character in the ASCII code system, and the `chr()` function to do a reverse lookup.

Problem 9: Square Roots

Write a Python program that extracts the square root of a given number using only basic arithmetic operations (addition, subtraction, multiplication, division, and exponentiation). The program should prompt the user to enter a number, and then use the Babylonian square root algorithm to calculate and display the square root of the number.

Difficult Problems

Problem 10: Rock, Paper, Scissors

Write a Python program that simulates a simple game of Rock, Paper, Scissors. The program should prompt the user to enter their choice of Rock, Paper, or Scissors, then randomly generate a computer choice. The program should then determine the winner based on the following rules:

- Rock beats Scissors
- Scissors beat Paper
- Paper beats Rock

The program should keep track of the number of wins for the user and the computer, and print out the results after each round. The game should continue until the user chooses to quit.

Problem 11: Connect Four

Write a Python program that simulates a game of Connect Four on a board with 6 rows and 7 columns. The program should prompt the two players to enter their moves (a column number) and display the updated board with each move. The program should then alternate turns between the players until one player gets four of their pieces in a row, either horizontally, vertically, or diagonally, or until the board is full and the game is a tie.

Here's an example of what the start of the program might look like on the console:

```
Welcome to Connect Four!

   1  2  3  4  5  6  7
1  -  -  -  -  -  -  -
2  -  -  -  -  -  -  -
3  -  -  -  -  -  -  -
4  -  -  -  -  -  -  -
5  -  -  -  -  -  -  -
6  -  -  -  -  -  -  -
```

Problem 12: Tic Tac Toe

Write a Python program that simulates a game of Tic Tac Toe. The program should display an empty 3x3 board at the beginning of the game, and then prompt the players to enter their moves (a row and column number) and display the updated board with each move. The program should then alternate turns between the players until one player gets three of their pieces in a row, either horizontally, vertically, or diagonally, or until the board is full and the game is a tie.

Here's an example of what the start of the program might look like on the console:

```
Welcome to Tic Tac Toe!

   1  2  3
1  -  -  -
```

```
2  -  -  -
3  -  -  -
```

And here is an example of player 1 winning:

```
   1  2  3
1  X  -  O
2  -  X  -
3  O  -  X

Player 1 (X) wins!
```