Degree in Software Engineering – Computing Basics

Unit 2.4 Exercises: Control Structures

This document includes a collection of exercises from Unit 2.4: Control Structures. It is recommended that you try doing the exercises without looking at the solutions first, and then you check your answers.

Exercise 1

The On-Board Diagnostics (OBD) system of a vehicle must control, among other things, that the tire pressure is adequate at all times. Develop a program in Python that, given the tire pressure of the four wheels, indicate which of them are below or above the recommended tire pressure of 2.5 ± 0.2 bar.

Assume that all values introduced by the user are valid.

Example:

```
Pressure of the front-left tire (bar): 2.4
Pressure of the front-right tire (bar): 2.2
Pressure of the back-left tire (bar): 2.8
Pressure of the back-right tire (bar): 2.7
The FRONT-RIGHT tire has LOW PRESSURE
The BACK-LEFT tire has HIGH PRESSURE
```

Exercise 2

The triangle inequality theorem states that "*the sum of the lengths of any two sides must be greater than or equal to the length of the remaining side*." Write a program in Python that takes the lengths of the three sides of a triangle in any order, and indicates whether that triangle is geometrically possible.

Assume that all values introduced by the user are valid.

Example:

```
Length of the first side: 1.5
Length of the second side: 9.2
Length of the third side: 2.4
The triangle does NOT exist
```

Exercise 3

In simple words, the Blackjack game consists in drawing cards from a deck and adding their value without exceeding 21 points. Whereas this game is usually played against a dealer, we will assume that we are playing a simplified version of the game in which two players draw cards from a deck until they exceed 21 points, or they surrender. Develop a program in Python that, given the scores obtained by each of the players, indicates the winner of the round. In this way, we will assume the following rules:

- If both players exceed 21 points, the program must indicate that both lose.
- If one of the players exceeds 21 points, but the other one does not, the player obtaining 21 points or less wins. The program must indicate who won in this case.
- If none of them exceeds 21 points, the player obtaining the highest score wins, unless there is a tie, in which case the program must indicate this situation.

Assume that all values introduced by the user are valid.

Example:

```
Player 1 score: 19
Player 2 score: 17
Player 1 wins with 19 points!
```

Exercise 4

Modify the program of the previous exercise so that, instead of specifying the total score obtained by each player, the user must specify the points of each of the cards drawn by the players in their corresponding turns. Player 1 will play their entire round first, and then player 2. The player's turn will end either when the player surrenders (a value lower than or equals to 0 must be introduced to indicate this situation), or when they exceed 21 points. The program must control that the user does not introduce values greater than 11, given that there are no cards with such a value.

Example:

```
Player 1 (0 points): 2
Player 1 (2 points): 6
Player 1 (8 points): 11
Player 1 (19 points): 0
Player 1 ends their turn with 19 points
Player 2 (0 points): 8
Player 2 (8 points): 6
Player 2 (14 points): 2
Player 2 (16 points): 9
Player 2 ends their turn with 25 points
Player 1 wins with 19 points!
```

Exercise 5

Write a program in Python that calculates the average of all the values introduced by the user as input. The end of the sequence is indicated by introducing a 0. Show the result using 2 decimal digits.

Example:

```
Next number: 3.8
Next number: -15.6
Next number: 4.7
Next number: 0
The arithmetic mean is -2.37
```

Exercise 6

The real number *e*, with value 2,718281828459045235…, can be approximated as follows:

$$e \approx 1 + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \cdots + \frac{1}{n!} = \sum_{i=0}^{n} \frac{1}{i!}$$

where *n* is a non-negative integer number. The greater the *n*, the greater the precision of the approximation. Develop a program in Python that calculates an approximation of number *e* by using as many terms of the above expression as required so that the difference between term *i* and *i – 1* is lower than a threshold in the interval (0, 1] specified as input.

Note: you can store the value of the factorial in an auxiliary variable, so that you increase it in each iteration of the loop.

Example:

```
Specify the threshold to be used in the approximation of e: 0.00001
The approximation of e after 10 iterations is 2.7182815255731922
```

Exercise 7

Modify the program of the previous exercise so that, instead of specifying the threshold used to calculate an approximation of *e*, the user specifies the number of iterations (terms) to be used to calculate it.

Example:

```
Specify the number of iterations used to calculate the approximation of e: 18
The approximation of e is 2.7182818284590455
```

Exercise 8

Writing a program in Python to determine the Fibonacci series is too mainstream. For this reason, in this exercise you are going to determine the following sequence:

$$1, 2, 4, 8, 16, 22, 26, 38, 62, 74, 102 \ldots$$

Can you see the pattern? Starting with any positive integer, the next element in the sequence is calculated as the sum of the current element and the product of all the non-zero digits in it. In this way, for example, assuming that the last element of the sequence is 38, the next element is calculated as 38 + 3 · 8 = 62. In the case of 102, in turn, the next element is calculated as 102 + 1 · 2 = 104, given that we skip the 0, as it is a zero digit.

Write a program in Python that prints all the elements of the series that are lower than a threshold specified by the user. Moreover, the program must also take the positive integer used as starting value for the sequence as input.

Warning: in this exercise you will need to nest loops, so it is not an easy exercise.

Example:

```
Specify the starting value of the sequence: 3
Specify a threshold: 100
3 6 12 14 18 26 38 62 74
```

Exercise 9

If you successfully developed the program of the previous exercise, modify it so that, instead of specifying a threshold, the user specifies the number of elements in the sequence to show.

Example:

```
Specify the starting value of the sequence: 2
Specify the number of elements to show: 12
2 4 8 16 22 26 38 62 74 102 104 108
```

Exercise 10

Write a program in Python that shows on screen all the multiples of a given positive integer, *x*, in the closed interval *[a, b]*, being *a* and *b* any two integers. The loop developed in your program must do the lowest number of iterations possible.

Example:

```
Specify the number whose multiples will be shown: 3
Specify the beginning of the interval: 17
Specify the end of the interval: 36
18 21 24 27 30 33 36
```

In this example, the loop should only iterate 7 times.

Exercise 11

What is the value of x after the execution of the following program in Python?

```python
x = 0
n = 19
while n > 0:
    n = n // 2
    x = x + 1
```

Exercise 12

What would be printed on screen after the execution of the following Python snippet?

```python
for i in range(5):
    for j in range(i + 1):
        print(j, end=" ")
```