CSB110MC: Python Programming – Coursework
Nim

Lauren Powell

Due: 11:00am, Thursday 6th April 2023

In the game of Nim, an arbitrary number of piles of tokens are formed, each with an arbitrary number of tokens in them, and two players alternate in removing one or more tokens from any one pile. The player who takes the last token is declared to be the winner.

In this assignment, you are asked to write a program for playing Nim.

There are a total of 25 marks available for this coursework. This coursework is worth 25% of the module.

Read the entire document before starting this exercise.

How to get marks

This assignment is broken down into 4 parts. You should ensure that you have a working solution to each part **before** moving on to the next. You must submit a single, working program.

These four parts make up 20 out of the 25 marks available. The remaining 5 marks are awarded for styling and code quality. You should be thinking about these marks the whole time you are writing your program.

Part A: Displaying Two Token Piles [4 Marks]

For the first part of this exercise, your program should prompt the user for two numbers, representing the number of tokens in two piles.

Your program should then print a representation of these two token piles to the screen.

Part B: Two Pile Nim – Player vs Player [8 Marks]

For the second part of this exercise, your program should implement a full game of two-pile Nim played between two human players.

Your program should prompt the user for two numbers, representing the number of tokens in two piles; and then

- ask Player 1 which pile they want to take tokens from, and how many;
- ask Player 2 which pile they want to take tokens from, and how many;
- ask Player 1 which pile they want to take tokens from, and how many;
- ask Player 2 which pile they want to take tokens from, and how many;

  ⋮

until the two piles are both empty, at which point your program should declare who is the winner (i.e., which player took the last token).

Your program should print a representation of the two piles to the screen before asking a player what they want to do; and it should never allow a player to try to remove more tokens from a pile than are in that pile.

Part C: Two Pile Nim – Player vs Computer [4 Marks]

For the third part of this exercise, your program should give the user the option of playing against the computer rather than a second human player.

If the user chooses to play against the computer, instead of asking Player 2 for a move, your program should randomly choose one of the token piles and a random number of tokens to remove from that pile. Your program should display a message indicating what move it has made.

Once both piles are empty, your program should display a message announcing the winner (player or computer).

Part D: Any Number of Piles Nim [4 Marks]

For the fourth part of this exercise, your program should allow the game of Nim to be played with an arbitrary number of token piles.

Your program should begin by asking the user how many piles they would like to play with, and how many tokens they want in each pile. You should create a list that stores the size of each token pile.

The game should then be played as before, ending when all token piles are empty.

Part E: Styling and Code Quality [5 Marks]

Your program should follow the styling conventions we have used in class. You should use appropriate comments, name variables appropriately, and indent your code properly. You should also try to structure your code well to avoid repeated code. Your output should be presented in a clean and clear format.

You do not need to define and implement your own functions, though it may improve your code quality. If you decide to implement your own functions, you should place all statements in the program into functions and write a `main` function. The only statement not placed into a function should be the statement calling the `main` function.

Submission

Before you submit, you should make sure your program works without crashing. Test it with several inputs until you are confident that it works as you want in all cases.

To submit your work, you need to login in to Canvas and go to:

Assignments ⟶ Coursework

You should submit a **single** zip file containing the following:

- Your Python source file.
- An optional README file (plain text file, PDF, or word document) which describes any special information you want the marker to know.

To create a zip file in Windows, do the following. Make sure all the files you want in the zip file are in the same folder. Select (highlight) the files that you want in the zip file. Right-click one of the selected files and from the context menu select `Send To` → `Compressed (zipped) folder`. This will produce a zip file with your selected files. You can check and see the contents by double-clicking the zip file.

Submit your work via Canvas before **11:00am, Thursday 6th April 2023**.