Programming Assignment 1 – Basic Probability

2017-2018, Master of Logic, University of Amsterdam

Instructors: Jakub Dotlačil, Alexandre Cremers and Bas Cornelissen

Submission deadline: Wednesday, April 11, 8 am

Note: if the assignment is unclear to you or if you get stuck, do not hesitate to contact Jakub or Bas. You can also post a question on Canvas.

1 Exercise

This week we are going to build a simple game of Rock-Paper-Scissors. The rules are straightforward. Each of two players chooses one of rock, paper and scissors. Rock beats scissors, scissors beat paper and paper beats rock. The game is played by computer (no interaction with humans). It is played once and the results are printed.

You will implement the game in two functions:

- function `who_won`. This function takes two arguments, `value1` and `value2`. `value1` and `value2` can be one of three numbers, `0`, `1`, and `2`, where `0`=rock, `1`=paper, `2`=scissors. The function returns `1` if value1 beats value2, it returns `2` if value2 beats value1 and it returns `0` if neither wins, according to the rules of the game.
- function `what_was_played`. This function takes a value among `0`, `1` and `2` and prints the statement *The player chose* ⟨*item*⟩, where item=rock if the value is `0`, item=paper if the value is `1`, item=scissors if the value is `2`.

After implementing the functions, you should write the flow of one game. In the game, first, a random element among rock, paper and scissors (e.g., among `0`, `1` and `2`) is assigned to one player; similarly for the other player. After that, the program should print what each player chose (using the function `what_was_played`) and decide who won (using the function `who_won`).

For the code, you will need to use the module `random`. Check the documentation here.

A run of the code should look as follows (of course, the game is random, so other combinations can be printed):

Player 1:
The player chose scissors.

Player 2:
The player chose rock.
Player 2 won.

A skeleton of the code is already given for you in the file here. Copy-paste the code into CodeSkulptor and work on your code. You only need to fill in the commented parts.

2 Grading

This section tells you what to look out for when programming. It also tells you how you should grade the assignment. In general, whenever you grade someone else's program, try to break it! This means that you should test limit cases in which the program might fail despite working correctly on more common inputs. However, since we are at the start of this course, you should only provide input to the program that makes sense (for example, only values `0`, `1` or `2` should be inputed into `who_won`).

As a general rule, if you the program cannot perform one of the tasks below because it does not start up or crashes, award `0` points for that task. If the program works on some, but not other, values, and 2 points could be awarded, award 1 point.

2 points The function `who_won` works correctly according to the rules of the game (i.e., the function outputs `1` when it receives values `0` and `2`, it outputs `0` when it receives values `1` and `1`, it outputs `2` if it receives values `0` and `1` etc.) You can check this by manually testing all possibile combinations in the code and by inspecting the code.

2 points The function `what_was_played` works correctly – it prints *The player chose rock* if it receives value `0` etc.

1 point The computer randomly chooses an integer `0`, `1` or `2` for player 1. This can be checked by running the code many times and/or inspecting the code.

1 point The computer randomly chooses an integer `0`, `1` or `2` for player 2. This can be checked by running the code many times and/or inspecting the code.

2 points In one run of the game, the computer correctly prints what each player chose, using the function `what_was_played`.

2 points The program uses the function `who_won` to determine the winner. It then correctly prints who the winner was (player 1 or player 2, or neither in case of a tie).