Intro Ex2 – Flow Control

Submission deadline

Thursday, 31 October, 20:55

Objectives

This exercise purpose is to get familiar with some more concepts of python programming, which cover the concepts we covered at the 2nd Tirgul
This exercise contains several tasks; these should not take long separately but might take a while together. Do not wait until the last day to start it. We recommend starting as soon as possible.
Your code must obey the Coding Style Guidelines in the website.

Note: all relevant files could be found in this tar file.

Task 1 – Square printing:

Background: In this task you are to print to the standard output a square with a rhombus (מעויין) inside it.

Your Task: To implement the function **square_printing** (skeleton attached in the file ex2_square.py) which receives an input, n, and prints to a square of size (2n+1) * (2n+1) with a rhombus (מעויין) with an edge of length n inside of it.

Examples for output:

```
>>> square_printing(5)
###########
#     *     #
#    * *    #
#   *   *   #
# *       * #
#*         *#
# *       * #
#   *   *   #
#    * *    #
#     *     #
###########
```

```
>>> square_printing(4)
#########
#   *   #
#  * *  #
# *   * #
#*     *#
# *   * #
#  * *  #
#   *   #
#########
```

```
>>> square_printing(3)
#######
#  *  #
# * * #
#*   *#
# * * #
#  *  #
#######
```

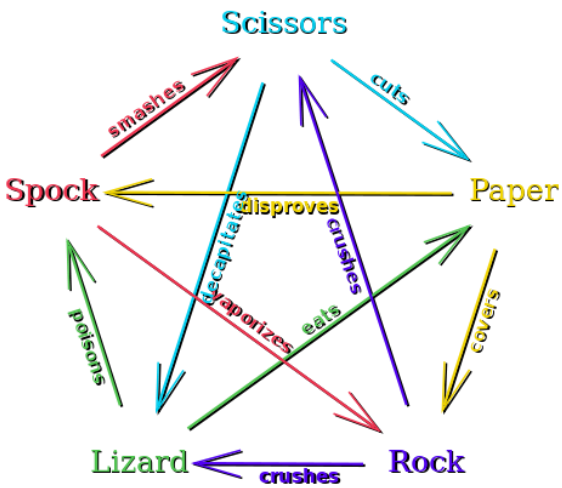
Assumptions for the input: All numbers entered for the function are positive numbers (e.g., 1,2,3, 500).

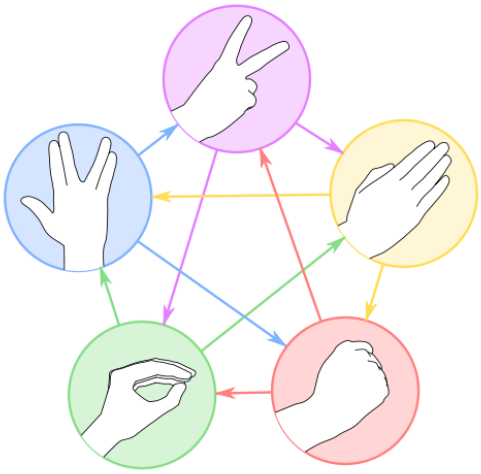
Expected question lines:
No question lines needed.

Task 2 – Rock-Paper-Scissors-Lizard-Spock (RPSLS):

Read Task 3(and all the ex) before implementing this, it might help you to design your code

Background: Rock-Paper-Scissors-Lizard-Spock is an extension to the Rock-Paper-Scissors game, which induces fewer ties. It could be described using this winning graph:





As you can see, there are a total of 5 options (X → Y means X wins over Y):

1. Rock → Lizard & Scissors
2. Paper → Rock & Spock
3. Scissors → Paper & Lizard
4. Lizard → Spock & Paper
5. Spock → Scissors & Rock

Two moves of same type are considered as tie. A demonstration could be found here: Demo.

Your Task: To implement the game in a function called **rpsls_game** (skeleton attached in the file ex2_rpsls.py) which gets no input. When calling this function the game should begin (**Note:** the game is meant for a single player playing against the computer), the flow of the game is like so, when actual expected output is soon to be followed.

1. The game continues in rounds until one side has 2 more wins than the other, in each round:
    a. Ask the player for his selection – 1 (Rock), 2 (Paper), 3 (Scissors), 4 (Lizard) and 5 (Spock).
        i. Users might enter the wrong numbers, be sure to check what they entered, if an unknown number was entered the appropriate message should be printed.
    b. Announce the player selection.
    c. Get the computers "selection", using *random.randint(a,b)*.
    d. Print the computer choice (you might want to use our attached helper_class, explained at the end of Task 3).

e. Determine the winner for this round, and continue for the next round.

. The game could end up in a draw, be sure to notice that.
After a winner for the game is found, print who he was.
Print the game statistics.
Return 1 if the player won or -1 if the computer won.

Expected question lines:
The exact texts you need are provided in the text file we provide you.
Remark 1: Some of the strings end with '\n'. These are used to add extra empty lines.

Assumptions for the input: you can assume all input will be legal numbers, though they may not be legal options.

Task 3 – Rock-Paper-Scissors-Lizard-Spock "Best of" mode:

Your Task: You have implemented in task 2 the *rpls_game* method for determining if the computer or the player has won. You are now to use that function to create the following RPSLS "best of" game in the function **rpsls_play** (skeleton attached in the file ex2_rpsls.py):

1. Print the welcome message.
2. Until the user has chose to quit the game:
    a) Ask the user how many games he would like to play, denote that as N (you may assume N is at least 1), this number should not be changed **unless** the scores have been reset by the user.
    b) Play the N RPSLS plays.
        i. Announce the start of the current play when started.
        ii. One player has won more than N/2 games and the other has won less.
        iii. If N is even, it is possible that that at the end of the N games there is a tie. In such a case, play until one player leads by two.
    c) Declare the current status of the game.
    d) Declare the winner. A winner is one who won the most of the plays;
    e) Print the number of games the user played so far, and his score.
    f) Ask the user if he wants to quit, reset all scores or continue.
        i. If the user has asked to reset his score, print that it had been done.

Expected question lines:
The exact texts you need are provided in the text file we provide you.
Remark 1: Some of the strings end with '\n'. These are used to add extra exmpy lines.

Assumptions for the input: you can assume all input will be legal inputs, meaning no wrong options will be entered.

Program notes

1. **We remind again – your program must behave exactly as ours (See an example below)**
2. **The name of these files must be ex2_rpls.py and ex2_square.py**.

The helper class:
Only for task 2 you need to use our attached helper file, it is already imported into the ex2_rpls.py file (like you did with turtle), this file is used to tell which choice the computer made. This is to help you convert the computers choice (using *random.randint*) into the string representing it, it is used like so:
compRes = random.randint(1,5)
getSelection(compRes)
Which will result with the string "Lizard", assuming compRes == 4.

We have also helped you and provided you with a text file containing all output required by you.

The ex2_rpsls.txt file:
As there is plenty of question lines on tasks 2+3 we have provided you with this text file which contains all lines you should print. For example on line 5 you have:
"Player has selected: @@@." #@@@ is one of Rock/Paper/…
You should replace the @@@ (in this case) with either Rock/Paper/Scissors/Lizard/Spock and you should only copy the text **inside** the double quotes (and without the @@@) **don't forget to copy the '\n' as well**.

The helper file should be **in the same directory** as the ex2_rpsls.py file.
Also if you are using Windows you should place that directory in a path without spaces, e.g.:

1. Good – C:\Intro2cs\ex2
2. Bad – C:\Users\my user name\My Documents\Intro 2 cs\ex2

Submission and Further Guidelines:

Creating a tar file

- By now, you should have created/edited the following files:
  1. ex2_square.py
  2. ex2_rpsls.py
  3. README (as explained in the course guidelines)
- **The helper class we provide should NOT be inside the TAR file.**
- Create a TAR file named ex2.tar containing only the above files by invoking the shell command:
  tar cvf ex2.tar ex2_rpls.py ex2_square.py README (See tirgul 1 slides for details about tars). The TAR file should contain only these files!

- It is recommended to check your TAR file by copying it to a different directory, opening it by executing the command: **tar xvf ex2.tar** and verifying that **ex2_rpls.py** and **ex2_square.py** and **README** were indeed successfully created.

Submitting the tar file

- If you haven't done so already, you must register for the course grading system (**only once!**)
- You should submit the file ex2.tar via the "Upload File" link on the course home page, under the ex2 link.
- Note that submission requires you to be registered as a student and logged in.
- Shortly after you upload the file you should receive an email to your university mailbox. That email contains the test results of your submission and the pdf file that was passed to the grader just like in ex0 and ex1.

Running the tests yourself

- There are two ways you can run these testers and see the results without submitting your exercise and getting the pdf.
- The first – place your tar file in an empty directory, go to that directory using the shell and type:
  ~intro2cs/bin/testers/ex2 ex2.tar
- The second – download a file (ex2testing.tar.bz2) from here. Extract the contents (using **tar –xjf ex2testing.tar.bz2**) of the file and follow the instructions on the file named TESTING.

School Solution:

- To test how the school solution behaves on legal input run in your linux shell:
  - Task 1: ~intro2cs/bin/ex2/ex2_square
  - Task 2+3: ~intro2cs/bin/ex2/ex2_rpsls