TU856/858-2 Object Oriented Programming with Python
Semester 1, Lab 5, 23.10.2020
Dr. Bianca Schoen-Phelan

Please join the virtual classroom in brightspace first, before going to individual lab links. I will provide explanations and hints and tricks regarding the lab at the beginning of the lab sessions. These are recorded.

Please sign your group's lab sign in sheet!
Lab solutions will be discussed in the Monday lecture. Labs are not marked.
Don't forget about your revision item for the week!

Link to git course overview (click). **<<<<<<------ NEW RESOURCE (also in brightspace)**

Link to the lab groups (click).
Link to python documentation (click).
Link to TKinter documentation (click).

Objectives:

- File handling
- GUI
- Queues
- Exceptions

ONLINE guideline:

- We meet in the virtual classroom in brightspace first. Just like the last lab.
- You will be able to stay within your own lab group throughout this online lab. Each tutor offers a teams link. You will be required to sign a sign-in sheet for each lab group.

You have this week and next week's lab to work on this exercise, as it is very challenging and brings together all the elements that we have studied in previous weeks. Next week's lab is the last lab before the lab test, so please bring questions to the lab next week!

Lab Background Information:

1. **Card Game 21:**
    - 21 is a card game where a card is taken from a shuffled deck of cards (by clicking on the closed deck of cards on the right in Figures 2-4). The goal is to get as close to the value of 21 points, or get exactly 21 points, but not over. The player loses if the score reaches over 21.
    - Jack, Queen and King count as 10. This card deck does not have an Ace or a Joker. Instead the T-card counts as 1. The other cards count as their face values.
      The following image shows an overview of all of your cards, 52 in total:

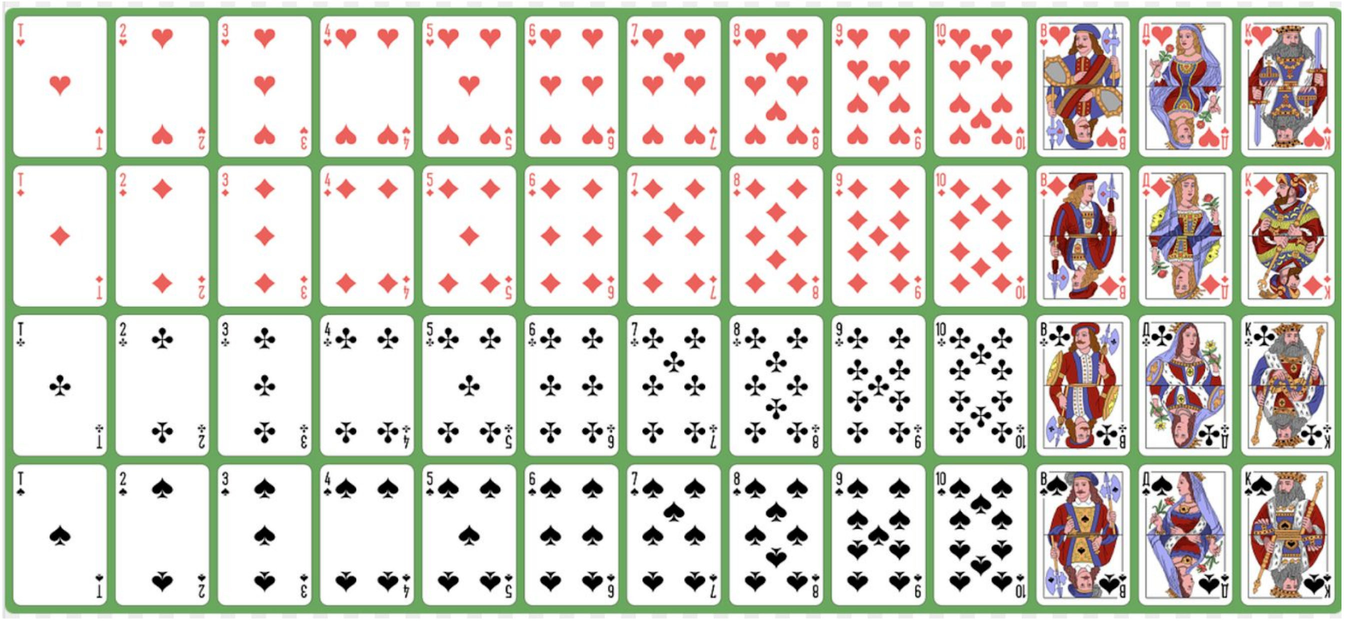

Figure 1: Overview of all available cards. See footnote for source reference.

- This week you are going to learn how to create a graphical user interface (GUI) in Python using the [TKinter library](). TKinter is one of the three main ways we can create a desktop program GUI. TKinter is part of Python and can be used by importing the library (this is done for you in the lab file). TKinter also has a reputation for an incredibly retro look. Skim the documentation for TKinter and then look at the provided Python file that sets up the GUI for you. Pay particular attention to how a button triggers an action via the `command=` argument, which points to a method that is defined to do some action (in the example of the exit button it closes the programme). Also, notice how some window GUI elements have the word `self.` In front of them. This means you can use the variable everywhere inside your class. Without it the variable only lives within the confines of its method. You will notice that there are several elements to your screen, arranged as follows, illustrated also in Figures 2-4:
    - A title bar that says “Card Game”
    - Two card images, the left one of the current card, the right one of the back side of a card, which illustrates the closed deck of cards from which new cards are drawn. The new card is supposed to appear on the left.
    - Three buttons on the right:
        - “I’m done!” for when the player does not want to draw another card. This is to trigger a final message in the score label.
        - “New Game” for when the player wants to start a new game of 21. This is to reset all values and start a fresh game.
        - “Exit” for when the player wants to exit the programme. This behaviour has already been implemented for you. Try it out!
    - A label at the bottom that shows the score and is supposed to display a message depending on the game logic, see Figure 3-4.

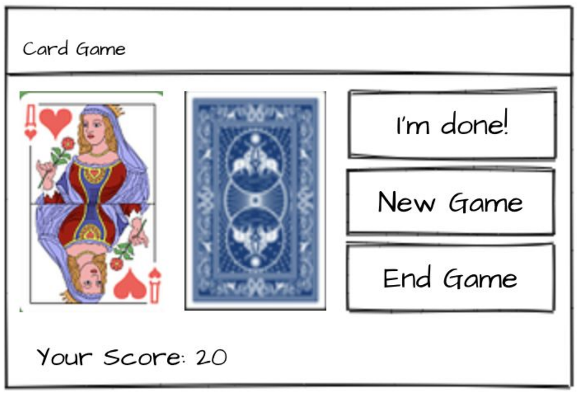

Figure 2: Design screen showing all the elements and the score count is set to 20.

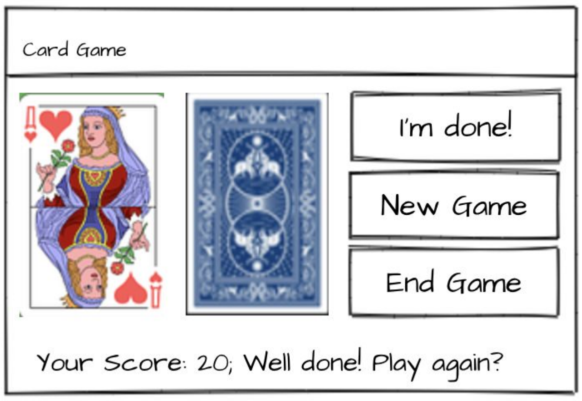

Figure 3: Design screen and the score label's output after clicking the "I'm done!" button

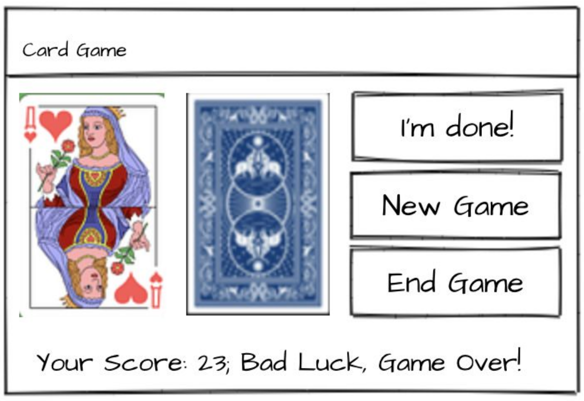

Figure 4: Design screen and the score label's output after a card has been drawn that pushes the score above 21.

YOUR TASK:

1.Fetch the lab 5 files from the BiancaSP/OOP2020-21 repository

- You have done this already last weeks, the same instructions apply.
- The following material is new this week:
    1. A folder called 'cards'[1]. See hints section on how to use the cards folder. This folder should be inside your Pycharm project folder when updating the repository.
    2. A python file called 'lab5_card_game.py'
- Create a new branch for lab5 on which you will be working on

2. Explore the code. Experiment with the new code. Set breakpoints. Do you understand the flow of the program? Make sure that you understand the existing code before adding new code. Watch a video of playing the game (click) Your solution should behave like this. Make a plan which functionality you will tackle first. Make a git commit for individual pieces of functionality that you have accomplished.

3. When the game starts the deck should be shuffled and one card is openly displayed. Count the score of this card to the player's score. This initial card is hard coded as the queen of hearts in the Python file you received for the lab. This should be changed to take a card from the deck after shuffling.

By clicking on the closed deck of cards, a new card is revealed on the left. The player's score should be updated to add this new card's score to the previous card's score.

For the card deck choose an appropriate queue data type, either LIFO or FIFO.

4. The score of the player should be displayed at the bottom in the label element. This needs to be updated frequently. This also needs to be reset if the *New Game* button is hit. The message should be changed if the score hits 21 or exceeds 21. If 21 has not yet been reached, a player can click on the *Done* button in order to indicate that he doesn't want to take any further cards. In this case, show the score, and deactivate the possibility to click on the closed deck again. The only options should be to Exit or to start a new Game.

5. The buttons "New Game" and "I'm done" and "Exit" should be used according to game logic.

- The "New Game" button should shuffle the closed deck and clear the score and display a fresh card.
- The "I'm done" button indicates that the player does not wish to take a new card. An appropriate message should be displayed, an example is given in Figure 3. In case the new card has pushed the score above 21, an appropriate game over message should be displayed, see Figure 4 for an

[1] The card images are taken from the royalty free image portal at pixabay: https://pixabay.com/vectors/atlasnye-deck-playing-cards-game-884206/

example. The player should not be able to select another card from the closed deck in these cases.

- The “Exit” button ends the game. This behaviour has been implemented for you as an example of how a button triggers an action when clicked.
- Clicking on the card deck reveals a new card. If the player has either reached 21 exactly, or has gotten above 21 clicking on the card deck should not reveal another card. The only actions open to the player should be to click New Game, or to exit the programme.

6. Be mindful of game logic:

- If 21 was reached the player should not be able to pick a new card from the closed card deck
- If the player lost because a score greater than 21 was achieved, the player should not be able to pick a new card from the closed deck but would have to first click “New Game” (or exit the programme).
- If the player has decided to click “I’m done”, the player should not be able to pick a card from the closed deck immediately after. The only possible actions after “I’m done” should be exiting the program or playing a new game.

7. Use exception handling as appropriate. For example, if there’s an issue loading the card images, the game should not go ahead.

8. Watch a video of playing the game (click) Your solution should behave like this.

9. How do I start?

1. When the window starts up, replace the hardcoded image with one from your shuffled deck. That means you need to load the cards into a list, shuffle the list, then load the list into either a FIFO or LIFO, get the first element in this stack/queue and display the respective image. The player score should be set to the value of this card. -> git add + commit
2. Now tackle the clicking on the closed deck. Every time you click, an item should be taken from your FIFO/LIFO structure. The respective card needs to be displayed on the open card deck. The value of the card needs to be added to the player score. -> git add + commit
3. Check if the value of the player score is 21, lower than 21 or above 21 and display a message. If your value is 21 it should not be possible to click for a new card on the closed deck. The only options are to get a new game or to exit. If you click done, which is typically while your player score is still below 21, display a message, and it should not be possible to click on the closed deck for another card. The only options are to get a new game or to exit the program. If the value hits above 21, display a message, and it should also not be possible to select a new card from the closed deck. Git add + commit
4. Functionality of “I’m done” button. The player score is locked, a message is displayed. It should not be possible to click for a new card. The only option is to ask for a new game or exit the program. Git add + git commit

5. Functionality of “New Game” button. Here we need to reset the player score to 0. The card deck needs to be shuffled again, as in empty the stack/queue, reshuffle your list and load it into the stack/queue. Get the first element from the stack/queue, display the respective card and update the player score. All buttons are active. A player can click on the closed deck of cards for a new card. Git add + git commit

HINTS:

- The path to the location of the cards will be different depending on what system you work at:
  - Windows syntax : `folder\\file`
    - Note the double \ here, remember about escaping characters!
  - Mac: `folder/file`
    - Forward slash is currently the default in the file provided.
  - Change this if needed!
- In order to update the label element that shows the scores or a button element that shows the card on the GUI use `updateIdleTasks()` on the element. An example for the label is as follows: `self.score_label.updateIdleTasks()` **after** setting a new text or new image. See the tutorial for examples.
- After starting up the freshly downloaded lab file and running the programme, you should get a screen that looks like the following:

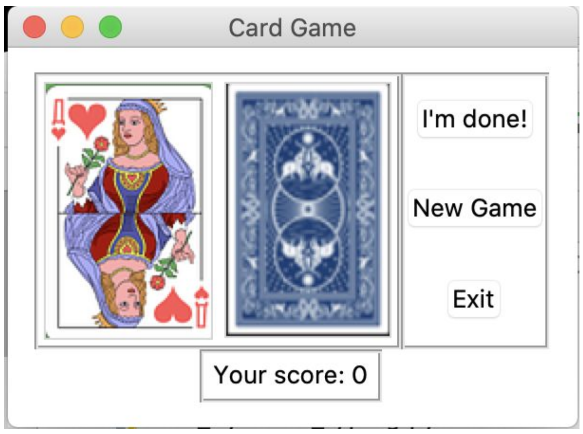

Figure 5: Screen Shot of the initial Game programme

Advanced Versions:

1. Keep an overall game score of how many games you have won, how many times you’ve hit the jackpot (=exactly 21), and how many games you’ve lost. Create a new `Label` for this.
2. Move the new cards a few pixels down on the screen to allow for the old card to still show.
3. Play with the look and feel of the elements of the window.
4. Add an option for the Ace card, which can either be 1 or 11, have the programme choose the most favourable one. You should use the T-card for this.
5. Implement a dealer player. For this you need to widen the area to display the cards for the dealer. You might like to add new labels too, in order to indicate which stack

of open cards belongs to the player and which one belongs to the dealer. Clicking on the closed deck will alternate between the player and the dealer in adding cards to the new deck. Both scores should be displayed.

6. Change the layout. Remember that using ttk brings the interface into the 21st century.

Well done!