# Blackjack Project

## Algorithms and python programming

Students: Abdelkoddous Hounkara, Vasileios Skarleas

**Introduction:**

On the UE course of INF101 under the general idea of learning algorithms and programming in Python, we are forced to develop a blackjack game by using Python and following the different programming techniques as well as skills that we developed during this semester. Below are written the indications that we needed to follow, the procedure that we followed in order to resolve any problems, as well as a summary of the problems / difficulties that we met. Our goal through this project is to make it even more clear how to connect many different functions in a Python programming environment while we are staying focused on the indications given by the project’s paper.

**Indications:**

We need to create a blackjack game by using the Python programming language. The project is divided into 4 different sections, where, by combining them we have a completed result. More specifically:

1. Initialisations
2. Game Management
3. Some intelligence
4. UI (graphical interface)

We note that parts A and B are directly connected the one to other, while C provides more functionality onto the part B, and finally the part D is a complete separate task from all the others in terms of functionality.

As blackjack is a cards game, the creation of deck or decks of cards is essential, as well as the possibility to count the different points gained from every player according to the following criteria:

* Cards numbered from 2 to 10 are valued as many points as their number (so between 2 and 10 points)
* The ace is valued either 1 or 11 points, at the player’s choice
* Figures (jack, queen, king) are valued 10 points each

The data structure provides two different options so that we can play several games in a row. This includes the management of many different players’ scores. This is why it’s preferred to use lists or dictionaries for saving and retrieving data.

* Option 1: scores, number of victories, and money gains of each player are stored in lists, so they can be accessed from the player number (index in the corresponding list).
* Option 2: scores, number of victories, and money gains of players are stored in dictionaries whose keys are the players’ names, so they can be accessed with the player’s name instead of the player’s number.

**Procedure:**

DISCLAIMER: We followed the steps as mentioned onto the project document and indications but maybe there are sections that we have changed the data structure to make the final game more efficient.

As team we decided to work to the corresponding workflow below:

* Organization, transparency and communication are our standards for a good team work
* According to the mantra mentioned above, we used the softwares mentioned below:
  1. GitHub: Includes our private repository with all the code that has been submitted by the team members. It tracks the live versions that every team member is watching every time on his personal desktop workstation. Also, GitHub provides a great way to visualize any changes and to restore previous versions of the code.
  2. VSCode: Instead of the Python idle, we decided to load python and work with this programming language, in our TD, TP, Caseine and current project, on VSCode since it is a universal development app, with some great extensions to integrate onto the workflow and to visualize better the different commands. In addition to that, it offers a direct connection to our GitHub repository.
  3. PythonTutor: Online program service indicated by the UE that helps to the deeper understanding of the corresponding code that we are examining every single time.
  4. W3Schools: Used for classes documentation that we integrated for part c on our separate version (explanation at paragraph Difficulties/Project – section 11).
* From the first view of the project, we knew that the part D would be the most difficult one and there was a possibility that we couldn’t be able to make it so far. So, our goal was set to complete as better as possible the rest of the tasks.
* For every step on the different four sections mentioned above, we decided to follow the “exams technique”. This means, that we will follow the exact steps by including the required elements mentioned on the specific task (function, returned value, required arguments, lists, etc.) as part of project’s understanding. The rest of the code will get completed according to the personal point of view through the understanding process of the task. This will allow us to have the biggest understanding that is possible, for this demanding project.
* We decided to set as default value for Ace the number 11 and instead of posting the question inside the valueCard module, we are asking the player at the moment of his turn to decide before we show up or not the next card.
* Via object-oriented programming techniques, we defined the different functions in a way that enables the automatic update and synchronization of lists and dictionaries, without the need to initialize any variables. Everything is automatically handled by the players list that is created by the system.

**Difficulties/Problems:**

In this section we are presenting the different challenges that we met and the workflows around them in order to be solved:

1. The biggest difficulty that we had was the concept the idea of the game. Since we had never played that game, nor having experience with these types of games, it was extremely difficult to understand what we are expecting from the program to do, something that corresponds at how will combine all the functions that we initialized on part A and result at least at one game play on part B. On the other hand, the UNO game for instance that we have in our copies, was a game that we know how it works and what changes to do in the program in order to make it work correctly.
2. One of the first challenges was the introduction and use of dictionaries. The moment that we started the development of this project, we didn’t have done it in our course. Thanks to Python’s documentation, the different internal functions() of dictionaries in pythons and key words like “keys” and “values” were used as expected on the different sections (like initPlayers()).
3. Another challenge was the initialization of cards’ deck. According to different resources, to most direct way is to use classes in Python which will then also help to the UI (fourth part of the project) by calling the executing the specified task. But we wanted a more direct way that is closed to our recent knowledges in python, and that’s why in our workflow we used lists to initialize the deck.
4. It was challenging to write the drawPick() function as long as we don’t know how a real blackjack game is played. This is also connected with the gameTurn() and completeGame() functions. YouTube was a very helpful tools in this understanding and searching procedure but again, without the real-life experience, we think we did not succeed.
5. Onto the firstTurn() function, even if we have constructed the drawCard() function to send the requested amount of cards in x variable from the p list of cards, when we were compacting these two cards that are requested on firstTurn() function onto a new list and then we were performing a “for” loop to take the different values, when we’re removing every specific card that was already triggered from deck, we were receiving an error that was changing the length of the two cards list that we were creating in very repeation of this section of the code. This is why we divided the process in one task that repeats in total 2 times for every player.
6. The valueCard() function sometimes was causing the firstTurn() function to crush due to NoType defined error in the returned values. In order to resolve the issue, we changed the examination formula of specific cards value in a generic one which permits not to worry for any typos on the defined lists that we create for the decks.
7. For playerTurn() function is was needed to parse the modified deck after the first turn that we count the initial points and this is why we are returning two elements onto the firstTurn() function, the dictionary with the players and the scores, as well as the modified deck of cards.
8. It could be possible to use gui to create a real graphical interface for the sectiond D of this project. Since we didn’t cover it in our lessons, we didn’t have the time to learn in depth the gui functions. Instead, we created a console-based UI which prints out the cards and visualizes them.
9. Since the initial deck of cards was a list of letters and numbers in one string corresponding to one card, in every step that there is modification of the deck, we are synchronizing it with a list that includes keyboards symbols (ASCII), which allows to use this synchronized list in order to print out the cards onto the console.
10. In order to count the victories we know that we need to use the winner() function but we are not forced to do it. We the different data structure that we created; in the end of every round, we receive the dictionary with the remaining players. With the use of winner() we determine the winner between the remaining players instead and we parse these data in a local list that is added via the main program. When the players want in general to terminate the game (main program), then we filter this local list and we take the final winner instead.
11. For the section C of the project, we knew that had to initialize the croupier as an existing player on the game but due to limited time to redefine the whole process, we developed a version of single-player game between the croupier (AI decisions) and the player. We strongly believe that this approach helped us to do a short reference on the classes that we are using, as a way to minimize the existing code. Also, we learned how to connect different project files together and to combine their mechanism inside one file. We totally understand that this is not exactly what was requested considering that other equips may have achieved it, but from our side that we had never worked on python and algorithmic techniques in the past, is an extremely important achievement.

**Project’s code:**

We are attaching our project’s code divided into the different sections of the project mentioned on this document’s introduction. For the section D, we have integrated a partially autonomous UI printing system inside the corresponding functions where drawCard() is used.