# 

# COMP 5070 Statistical Programming for Data Science

# Blackjack Simulator

## Test 1 DUE by 11:00pm (CST), Sunday 21th August

* This exercise is a part of the continuous assessment that is worth 25% of your overall grade.
* Your code should be submitted as a single .py file using LearnOnline. Do not hardcode any paths on your computer in the code, as I should able to load and run your code without any changes.
* The exercise is out of 100 marks. To obtain the maximum available marks you should aim to:

1. Code the requested program (70%).
2. Use a clear coding style (10%). Code clarity is an important part of your submission. Thus, you should choose meaningful variable names and adopt the use of comments - you don't need to comment every single line, as this will affect readability - however you should aim to comment at least each section of code.
3. Have the code run successfully (10%).
4. Document code limitations including, but not limited to, the requested functionalities (10%).

This assessable exercise can be openly discussed within the group online and you are welcome to share tips and tricks (not entire programs, however).

Having said that, the ground rules are:

* If you use another person’s code in your file, please note the source and how much of the code is not yours.
* If you submit a program cobbled together by other peoples’ code with no, or little, original input from yourself, you will automatically receive zero mark and will be reported to Academic Integrity office. The idea is to develop your own programming style with (or without) the help of others. Any code used should support your approach to how you write the program, not replace your own efforts.

If you’re unsure at any point, you’re welcomed to check with me.

Late submission will be penalized by 10-point deduction for each day or part of it after the due date.

**Disclaimer:** This exercise does not promote or endorse gambling in any way. It serves to practice fundamental programming skills in Python and overall problem solving.

**If you have any questions – feel free to ask them on the forum or by email.**

# Blackjack

You should write a simulator of the game of Blackjack. Week 2 exercise “21” was a good preparation for this task, however Blackjack is a different game.

You can read full rules of Blackjack on the internet. Just google for “Blackjack rules” and you will find many other sources, e.g., <https://wizardofodds.com/games/blackjack/basics/> (really good and clear rules).

You will code not all but only major rules of a Blackjack, so you create a somewhat simplified version. For example, you will ignore options to Double or Split or Surrender, as they are just extra complications on the top of major rules. Look for functions description below.

Your code will have several main components – all of them prepared as functions:

* Function to prepare the deck of card. There are 52 cards in a deck. The game can be played with more than one deck of cards at the same time. As the suit of the card is not important for the game, keep all cards as numbers 2 to 11 with the correct number of 10s.
* Function to draw the card. Beware, when you draw the card, it should be taken out of the deck. You cannot draw the same card twice. List’s method *pop()* does the trick. After several games in a row the deck becomes small, so you have to create a new deck.
* Function for a player. Player starts with two cards and sees all cards on the table, that is, its own cards and dealer’s and other players’ cards. Hence, **function input will be three lists**: player’s cards, dealer’s card (dealer get only one card) and all other players’ cards if you have other players. If there is just one player, then the third list is empty. However, potentially you can use the same player function to represent multiple players.  
  Based on the analysis, player makes a decision to take another card or to hold. Your **function output will be 0** (stand or hold) **or 1** (hit - take a card). Try to make this function “smart”.
* Function for a dealer. Dealer plays last and follows a very simple strategy: if total sum is less than 17, dealer takes the next card, otherwise dealer stops. Hence, **function input is a list of dealer’s cards and output is 0 or 1**.
* There can be some other supporting functions, e.g., a function to count total sum of cards or a function to manage the game, that is, to maintain the order who gets cards first, who stays in the game and who gets busted. Alternatively, game management can be organized as a for-loop.

Blackjack is a cash game. The important part of the game is to make a decision about a bet size. We ignore all that stuff. We run the game for 1000 (or more) times and count how many times player won and lost (that is, dealer won).

In your simulation, you will practice creating algorithm and coding. There is no communication with the human as human does not play. No need to use input() function. Computer plays against itself and then reports statistics for the player and dealer.