# O jogo das cartas

It is a card game for two players (one human and the computer), which takes place in 10 moves with a 40-card deck of cards for the computer and a 40-card deck of paper cards for the human player.

Similar to the so-called "battle game", the rules for each move are quite simple:

* Each player takes a card from his deck and the two cards are confronted;
* The player with the highest card wins 2 points (1 point per card);
* If the cards have the same figure, there is a tie, and each player draws one more card. Whoever wins gets 4 points (2 points for each card).
* And in the event of consecutive draws, 2 more points will always be added for every two cards that face each other.

# First version of the game

To create the game, we need to define some global variables and literals.

import random

BARALHO = "23456QJK7A"

# score for the two players

scoreComputer = 0

scoreHuman = 0

# maximum number of moves

moves = 10

# conta as cartas em caso de empate

tie = 0

The literal BARALHO contains the sequence of figures of the cards in the digital deck, ordered in ascending value, therefore, from the lowest value (2) to the highest (A).

|  |  |
| --- | --- |
| **Head with gears** | Observe the position of card 7. It does not appear immediately after card 6!  This is a special card that is only beaten by the ace. |

Then, the score variables of the human player and the computer are set, initialized to zero. Finally the maximum number of moves (10). There is also an additional variable (tie) that allows counting the score to be awarded in the event of a tie or consecutive ties.

We will define a function jogada to interact with the human player. The player takes a card from his deck (on paper) and the function played jogada is called with the card's figure as an argument.

The algorithm for each move can be as follows:

1. The computer takes out a card;
2. The cards of both players are presented;
3. **Sub-problem** logic of the game;
4. If the 10 moves are over then:  
   **Subproblem** presenting the winner

To simplify, we leave as subproblems: 1) logic of the game, which consists of the application of the rules of the game and determination of the scores, which are global variables; 2) present the winner.

Analise carefully, the implementation of the function jogada:

def jogada(cardHuman):

print("\*\*\* Move of the game \*\*\*")

# 1. The computer takes out a card

cardComputer = random.choice(BARALHO)

# 2. The cards of both players are presented

print("Computer:", cardComputer)

print("Human:", cardHuman)

# 3. Sub-problem logic of the game

logica\_jogo(cardComputer, cardHuman)

# 4. If game is over

if moves == 0:

print("O jogo terminou...")

# Subproblem presenting the winner

vencedor()

else:

print("You have ", moves, "moves.")

|  |  |
| --- | --- |
| **Eye** | The figure of the computer's card is obtained through the function random.choice()with the string BARALHO.  But this function does not guarantee that the same figure will not come out repeatedly... |

Before starting the game logic, it is convenient to define a simple function that determines the value of the figure of the card.

# returns the value of the figure

def valor(figure):

return BARALHO.index(figure)

Since the literal BARALHO have the card figures in sequence, we use the method [index()](https://www.w3schools.com/python/ref_list_index.asp) to determine the sequence number.

The game logic algorithm will be a multiple selection structure with the following conditions:

* If the computer card > human card then:
  + Give points to the human player
  + decrement the moves
* If not, If the computer card < human card then
  + Give points to the computer
  + decrement the moves
* If not... Tie.

Notice that the scoring of points to the winner includes not only the 2 cards in confrontation, but also the previous cards, in case of previous ties. In the event of a tie, the same move remains until a tiebreak occurs.

The game logic code is as follows.

def logica\_jogo(comp, hum):

global scoreComputer, scoreHuman, tie, moves

if valor(comp) > valor(hum):

print(2 + tie \* 2, "points for the computer.")

scoreComputer += 2 + tie \* 2

moves -= 1

tie = 0

elif valor(comp) < valor(hum):

scoreHuman += 2 + tie \* 2

print(2 + tie \* 2, "points to the human player.")

moves -= 1

tie = 0

else:

tie += 1

print("Tie... Play again...")

# score

print("Computer", scoreComputer, "vs", scoreHuman, "Human")

|  |  |
| --- | --- |
| **Balloon animal** | Try this first version of the game, through the command line.  *For each move, call the function* jogada() *with the figure of the card obtained by you:*  >>> jogada ("7") |

# An improved version!

|  |  |
| --- | --- |
| **Head with gears** | You certainly felt that the need to call the function jogada() at each move somewhat broke the feeling of "immersion" in the game. Would it be possible for the computer itself to iterate between moves, automatically?  The answer is yes... We can use a cicle - [while](https://www.w3schools.com/python/python_while_loops.asp). We will cover this control structure in the next chapter. For now, keep in mind that this structure always repeats your block of code while the conditional expression is evaluated as True. |

Notice the fact that the use of global variables complicates the reading of the algorithm. In this new version, we will only use local variables, as a principle of good programming. Only the literal BARALHO will remain.

Function jogo() that we're going to create, encompasses all the game variables and also the game logic.

Before steping into function jogo() , we will remember function valor()and introduce function vencedor().

The winning function follows the following algorithm:

1. Header indicating the end of the game
2. Multiple selection structure to determine the game's final result:
   1. Victory of the human player;
   2. Computer victory;
   3. A tie.

import random

BARALHO = "23456QJK7A"

# returns the serial number of the card in the deck

def valor(figure):

return BARALHO.index(figure)

# shows the winner

def vencedor(pontosComputador, pontosHumano):

print("")

print("Game Over...")

print("---- Final Result ----")

print("Computer", pontosComputador, "vs", pontosHumano, "Human")

if pontosHumano > pontosComputador:

print ("You won!")

elif pontosHumano < pontosComputador:

print("You lost.")

else:

print("Tie...")

def jogo(jogadas):

# score of both players

pontosComputador = 0

pontosHumano = 0

pontos = 2

while jogadas>0:

print("\*\*\* Move \*\*\*")

print("Computer", pontosComputador, "vs", pontosHumano, "Human")

# input - jogada do computador e do jogador humano

cartaHumano = input("Human: ")

cartacomputador = random.choice(BARALHO)

print("Computer:", cartacomputador)

# game logic

if valor(cartacomputador) == valor(cartaHumano):

# tie

print("Tie... Play again...")

pontos += 2

elif valor(cartacomputador) > valor(cartaHumano):

print(pontos, "points to the computer.")

pontosComputador += pontos

jogadas -= 1

pontos = 2

else:

print(pontos, "points to the human player.")

pontosHumano += pontos

jogadas -= 1

pontos = 2

# end of the move

print("You have", jogadas, "moves.")

# end of the game

vencedor(pontosComputador, pontosHumano)

|  |  |
| --- | --- |
| **Head with gears** | Look at function jogo and determine the main changes compared to the previous version of the game. |

Check out the main changes that occurred:

* The global variables scoreComputer and scoreHuman became local variables pontosComputador and pontosHumano of the function jogo;
* The global variable moves has become a parameter of function jogo. This allows you to configure the game according to the number of moves you prefer to play;
* The global variable tie was replaced by the local variable pontos;
* The interaction with the player started to be done with the function input().
* The game end condition has passed to the while cycle, which repeats the moves indefinitely until the variable jogadas reaches the value 0.

# Final challenge

Change the game so that it does not end after a certain number of moves, but when one of the players reaches a certain score value (10 points, for example, that would be defined as a function parameter).

|  |  |
| --- | --- |
| **Share with person** | Share **your** game! |