UNIVERSIDADE FEDERAL DO PARANÁ

Fernando Augusto de Lima Filho - GRR20195837

Igor da Costa Medina - GRR20195843

Anne Caroline Tomaz da Silva Strapaçon - GRR20195826

Jogo do "Foda-se" em Python

Curitiba

2019

Fernando Augusto de Lima Filho - GRR20195837
Igor da Costa Medina - GRR20195843
Anne Caroline Tomaz da Silva Strapaçon - GRR20195826

Jogo do "Foda-se" em Python

Trabalho Acadêmico apresentado à disciplina Fundamentos de Programação de Computadores do Curso de Graduação em Matemática da Universidade Federal do Paraná, como exigência parcial para obtenção de nota.

Orientador: Jackson Antonio do Prado Lima

Curitiba
2019

"O que os olhos não veem,
cai na prova".
(Universitários)

Resumo

Este relatório apresenta a versão de um jogo conhecido vulgarmente entre os estudantes de graduação como “Foda-se”, que se baseia no Truco. Utilizando a biblioteca Pydealer que fornece um baralho completo de cartas, além de outras ferramentas e o truco feito em Python (1), adaptou-se as regras do "Foda-se"para construir uma nova versão do jogo.

Palavras-chave: Jogo de Carta. Python. Desespero.

Lista de ilustrações

Sumário

1 Introdução

1.1 Problematizaçao

Diferente do Truco, o jogo "Foda-se"não possui uma versão para computador ou qualquer dispositivo eletrônico, então buscou-se de forma simples construir uma versão adaptada do mesmo reunindo dados/informações com o propósito de responder ao seguinte problema: Como construir o jogo “Foda-se” em Python sem perder todas as características do jogo?

1.2 Objetivos

1.2.1 Objetivo Geral

Utilizar os conhecimentos adquiridos durante as aulas colocando eles em prática construindo um jogo de cartas.

1.2.2 Objetivos Específicos

- Definir as regras adaptadas do jogo "Foda-se";
- Definir com base no truco, as cartas, ordem e valores;
- Construir um algoritmo para que os outros players (computadores) joguem corretamente.

1.3 Justificativa

Com a necessidade de uma versão do jogo "Foda-se"para dispositivos eletrônicos, surge a proposta de criar de forma adaptada do mesmo, tal que as características básicas do jogo fossem mantidas.

2 Metodologia

2.1 Como o jogo "Foda-se"funciona

Informações Básicas: Um baralho convencional possui 52 cartas, com 4 naipes, cada naipe possui 13 cartas.

Nessa versão do jogo “Foda-se” tem-se :

- Jogadores: 5
- Número de cartas: 40 (retirando-se 8, 9 e 10)
- Distribuição: 4 cartas para cada participante
- Objetivo: O jogador que fizer 5 pontos, ganha o jogo.

Como foi descrito anteriormente o jogo "Foda-se"é baseado no jogo truco e, portanto, segue a mesma ordem de números e naipes, que por convenção é a seguinte:

Figura 1 – Ordem de valores (Maior para o menor)

Definições do jogo:

- Vira : É a carta que o computador vira quando distribui as cartas. É a carta que definirá as 4 manilhas
- Manilhas : São as cartas mais fortes do jogo, mais fortes ou igual ao 3.
- A manilha é variável, sendo determinada pelo vira.
- Rodada : Em cada rodada, os jogadores mostram uma carta. A carta mais forte ganha a rodada.
- No final da distribuição das cartas, vira-se uma carta para cima denominada “vira” e a carta que de acordo com a sequência é a próxima, em seus 4 diferentes naipes, são definidas como as Manilhas.
- Entre as manilhas, a ordem de “força” obedece o naipe, da seguinte maneira (do maior para o menor): Paus > Copas > Espadas > Ouros

Figura 2 – Ordem dos naipes (Maior para o menor)

- Você deve olhar as suas cartas e "palpitar"quantas das suas cartas são capazes de ganhar dos outros jogadores, mesmo sem saber a carta deles.
- O último jogador a dar o palpite do número de acertos que faz, ao somar todos os palpites que estão fazendo, não pode ser igual ao número de cartas na mão, que nesse caso são sempre 4 cartas. Essa regra deve ser respeitada para dificultar o jogo.
- Ganha ponto apenas o jogador, ou os jogadores que fizer exatamente a quantidade de palpites que disse que faria.
- Ganha o jogador que fizer 5 pontos primeiro!

2.2 Bibliotecas e Definindo ordens de jogo

Utilizamos a biblioteca Pydealer (2) que nos fornece um baralho completo, entre outras utilidades que serão vistas ao longo do trabalho.

```
1  import random
2  import pydealer.tools
3  from jogador import Jogador
4  from truco_ranks import *
5  from pydealer.const import DEFAULT_RANKS
6
```

Figura 3 – Bibliotecas utilizadas.

Como ha 5 jogadores, temos que definir a ordem jogada com base em uma ordem fixa e as 5 ordens possíveis. Para isso criamos uma lista para cada uma das possíveis ordens.

```
40  #Lista da ordem fixa de jogadores
41  ordem = ["Player 2", "Player 3", "Player 4", "Player 5", jogador]
42  #Listas das 5 ordens possiveis
43  ordem1 = ["Player 2", "Player 3", "Player 4", "Player 5", "Você"]
44  ordem2 = ["Player 3", "Player 4", "Player 5", "Voce", "Player 2"]
45  ordem3 = ["Player 4", "Player 5", "Você", "Player 2", "Player 3"]
46  ordem4 = ["Player 5", "Você", "Player 2", "Player 3", "Player 4"]
47  ordem5 = ["Voce", "Player 2", "Player 3", "Player 4", "Player 5"]
```

Figura 4 – Ordem de jogada.

2.3 Definindo os Jogadores

Definimos os 5 jogares de maneira trivial, sendo o usuário + 4 computadores.

```
28  #Definindo Jogadores
29  jogador = input("Digite seu nome: ")
30  P2 = "Player 2"
31  P3 = "Player 3"
32  P4 = "Player 4"
33  P5 = "Player 5"
34  print("\n")
35
36  placar = [0, 0, 0, 0, 0]
37  #Lista de jogadores
38  jogadores = ["Player 2", "Player 3", "Player 4", "Player 5", "Você"]
39
```

Figura 5 – Jogadores.

Criamos uma lista *placar* com o placar de cada jogador, cada posição corresponde ao ponto do jogador no qual a ordem depende de quem for o primeiro a jogar e a lista *jogadores* representa os jogadores presentes no jogo. Lembrando que só ganha o jogador que fizer 5 pontos.

2.4 Definindo as possíveis ordens de jogo

Como dito nas secção 2.2 ha 5 ordens possíveis então fizemos as condicionais *If, Elif, Else* que serão escolhidas a partir do comando *primeiro* que seleciona da lista *ordem* um jogador qualquer.

```
52   primeiro = random.choice(ordem)
53   if primeiro == "Player 2":...
232
233  elif primeiro == "Player 3":...
412
413  elif primeiro == "Player 4":...
591
592  elif primeiro == "Player 5":...
776
777  else:...
957
```

Figura 6 – As 5 possíveis ordens.

2.5 Se o primeiro jogador for o Player 2

Vamos supor que o primeiro jogador seja o Player 2 então temos a ordem correspondente a lista *ordem1* vista na sessão 2.2. Logo em seguida como queremos que as partidas se repitam ate que alguém faça 5 pontos, criamos um laço de repetição da seguinte maneira.

```
#Manten varias rodadas ate que alguem faca 5 pontos
while placar_[_0_] != 5 and placar_[_1_] != 5 and placar_[_2_] != 5 and placar_[_3_] != 5 and placar_[4] != 5_:
    #Pontos de cada jogador de cada rodada
    pontos = [_0, 0, 0, 0, 0_]
    print("O primeiro jogador a dizer quantos jogos faz é", primeiro)
    #Definindo o baralho tendo como ordem de valores e naipes as ordens padroes
    baralho = pydealer.Deck(ranks=DEFAULT_RANKS)
    print("\n")
    print("Embaralhando as cartas...\n")
    baralho.shuffle()
    print("Dando cartas...")
    print("\n")
    #Distribuindo 4 cartas para todos os jogadores
    P2 = baralho.deal(4)
    P3 = baralho.deal(4)
    P4 = baralho.deal(4)
    P5 = baralho.deal(4)
    voce = baralho.deal(4)
```

Figura 7 – Primeiro while.

Utilizando um baralho com 52 cartas e de valores padrões, podemos remover as 12 cartas de dentro da biblioteca e usamos a função *.deal()* para distribuir as 4 cartas para os jogadores. Em seguida o Vira e definido e com base nele criamos 10 ranks no qual as manilhas serão definidas.

```
#Define o Vira
vira = baralho.deal(1)
#Retorna o valor da carta na classe de Card
valor = vira_[_0_].value
```

Figura 8 – Vira

O comando *vira[0].value* (.value vem da classe Card() (3)) retorna o valor da carta sem o naipe, entao criamos as condicionas que dependem do valor da carta do vira. Cada *RANK* e definido da seguinte maneira.

```
90          #Ordem padrao de valores e naipes
91          ranking = DEFAULT_RANKS
92          #Define o vira com base no valor da carta
93          if valor == "4":
94                  baralho = pydealer.Deck(ranks=RANK1)
95                  ranking = RANK1
96          elif valor == "5":
97                  ranking = RANK2
98                  baralho = pydealer.Deck(ranks=RANK2)
99          elif valor == "6":
100             ranking = RANK3
101             baralho = pydealer.Deck(ranks=RANK3)
102         elif valor == "7":
103                 ranking = RANK4
104                 baralho = pydealer.Deck(ranks=RANK4)
105         elif valor == "Q":...
108         elif valor == "J":...
111         elif valor == "K":...
114         elif valor == "A":...
117         elif valor == "2":...
120         else:...
```

Figura 9 – Definindo manilhas.

```
1   RANK1 = {
2       "values": {
3           "5": 10,
4           "3": 9,
5           "2": 8,
6           "A": 7,
7           "K": 6,
8           "Q": 5,
9           "J": 4,
10          "7": 3,
11          "6": 2,
12          "4": 1,
13      },
14      "suits": {
15          "Paus": 4,
16          "Copas": 3,
17          "Espadas": 2,
18          "Ouros": 1
19      }
20  }
```

Figura 10 – Ranks das cartas.

Como na figura 10 a maior carta recebe o valor "10", ou seja, se a carta 5 recebe

o valor 10 isso significa que o vira e a carta 4 e assim funciona para todas as cartas do baralho, totalizando 10 ranks.

Agora iremos definir o palpite de cada jogador, para isso vamos criar uma lista *qnt* e *j* no qual representam a quantidade total de cartas de cada jogador e os palpites respectivamente.

```
        print("\n")
        #qnt = quantidade de cartas que cada jogador possui menos a sua
        qnt = [len(P2), len(P3), len(P4), len(P5)]
        #j = Os jogos que cada jogador diz que faria
        j = []
        #soma de todos os jogos
        soma = 0
        ordem = ["Player 2", "Player 3", "Player 4", "Player 5"]
        #Para cada jogador (computador) escolha na lista jogos a quantidade que cada um vai fazer e adiciona na lista j
        # e soma
        for k in range(len(ordem)):
            numero = random.choice(jogos)
            print(ordem[k], " faz {}".format(numero))
            numero = int(numero)
            j.append(numero)
            soma += numero

        seus = int(input("Quantos jogos você faz? "))
        #Enquanto a soma fechar na 4 digite novamente um valor no qual a soma nao feche em 4
        while soma + seus == max(qnt):
            print("Sua escolha tem que ser diferente de {}".format(max(qnt) - soma))
            seus = int(input("Quantos jogos você faz? "))

        j.append(seus)
        soma += seus
```

Figura 11 – Palpites.

Fazemos um for no qual para cada jogador ele escolhe um numero aleatório da lista *jogos* e adiciona na lista *j*, como o usuário nessa ordem e o ultimo a falar, segundo as regras do jogo a quantidade *soma* não pode ser igual a 4 ou em caso geral não pode ser igual ao numero de cartas da pessoa que tiver a maior quantidade de cartas, criamos um while que atenda a essa condição e em seguida adiciona a lista *j*.

Próximo passo e definir as 4 rodadas onde cada jogador ira jogar a sua carta. Criamos a carta mesa como sendo a carta escolhida por cada jogador e removemos a mesma da mão do mesmo.

```
156         #Define as rodadas onde cada jogador ira jogar a sua carta
157         rodada = 1
158         #Como sao 4 rodadas criamos um laço ate dar 4
159         while rodada < 5:
160             print("\n")
161             print("Vira: ")
162             print(vira)
163             print("\n")
164 
165             print("Suas cartas são: ")
166             for i in range(len(voce)):
167                 print("{0} - {1} ".format(str(i + 1), voce[i]))
168 
169             print("Joguem suas cartas...\n")
170             #Cada jogador sua uma carta que é removida da mao do mesmo
171             carta_mesa2 = random.choice(P2)
172             P2.remove(carta_mesa2)
173             carta_mesa3 = random.choice(P3)
174             P3.remove((carta_mesa3))
175             carta_mesa4 = random.choice(P4)
176             P4.remove(carta_mesa4)
177             carta_mesa5 = random.choice(P5)
178             P5.remove(carta_mesa5)
```

Figura 12 – Rodada.

Fazemos o mesmo com as cartas do usuário usando condicionais.

```
187             opção = int(input("Qual carta você escolhe? "))
188             if opção == 1:
189                 sua_carta = voce[0]
190                 voce.remove(sua_carta)
191                 print(jogador, "jogou ", sua_carta)
192             elif opção == 2:
193                 sua_carta = voce[1]
194                 voce.remove(sua_carta)
195                 print(jogador, "jogou ", sua_carta)
196             elif opção == 3:
197                 sua_carta = voce[2]
198                 voce.remove(sua_carta)
199                 print(jogador, "jogou ", sua_carta)
200             else:
201                 sua_carta = voce[3]
202                 voce.remove(sua_carta)
203                 print(jogador, "jogou ", sua_carta)
204 
```

Figura 13 – Escolhas do usuário

Para comparar as cartas que foram jogadas pelos jogadores usamos a função *carta.gt(carta, rank)* que compara duas cartas se uma e maior que a outra no ranking

definido pelo vira.

```
205         print("\n")
206         #Usanmos a funçao .gt() que compara duas cartas se uma e maior que a outra
207         #Se a carta do Player 2 for maior que todas entao Player dois ganha um ponto da rodada
208         if carta_mesa2.gt(carta_mesa3, ranking) and carta_mesa2.gt(carta_mesa4, ranking) and carta_mesa2.gt(carta_mesa5, ranking) and ca
209             print("Player 2 ganhou a rodada!")
210             pontos[0] += 1
211 
212         elif carta_mesa3.gt(carta_mesa2, ranking) and carta_mesa3.gt(carta_mesa4, ranking) and carta_mesa3.gt(carta_mesa5, ranking) and
213             print("Player 3 ganhou a rodada")
214             pontos[1] += 1
215 
216         elif carta_mesa4.gt(carta_mesa2, ranking) and carta_mesa4.gt(carta_mesa3, ranking) and carta_mesa4.gt(carta_mesa5, ranking) and
217             print("Player 4 ganhou a rodada!")
218             pontos[2] += 1
219 
220         elif carta_mesa5.gt(carta_mesa2, ranking) and carta_mesa5.gt(carta_mesa3, ranking) and carta_mesa5.gt(carta_mesa4, ranking) and
221             print("Player 5 ganhou a rodada!")
222             pontos[3] += 1
223 
224         elif sua_carta.gt(carta_mesa2, ranking) and sua_carta.gt(carta_mesa4, ranking) and sua_carta.gt(carta_mesa3,ranking) and sua_car
225             print("Você ganhou a rodada!")
226             pontos[4] += 1
227 
```

Figura 14 – Comparando as cartas.

Para manter os jogadores atualizados criamos uma lista com a ordem dos jogadores, quantidade de jogos que devem fazer e quantidade de jogos que estão fazendo na rodada.

```
228             print("\n")
229             print("[P2, P3, P4, P5, {}]".format(jogador))
230             print("Tabela de Jogos")
231             print(j, "\n")
232             print("Tabela de Jogos da Rodada")
233             print(pontos, "\n")
234             rodada += 1
235 
236 
237         for k in range(5):
238             if j[k] == pontos[k] and pontos[k]>0:
239                 placar[k] += pontos[k]
240             elif j[k] != pontos[k]:
241                 placar[k] = placar[k]
242             else:
243                 placar[k] += 1
244 
245         print("\n")
246         print("O placar é:", placar)
247 
```

Figura 15 – Tabelas e pontos.

Definimos os pontos da seguinte maneira:

- Se o palpite do jogador for igual a quantidade de jogos que ele fez, então seu placar vai ser igual a quantidade de jogos;

- Se o palpite do jogador for diferente então, ele não ganha nenhum ponto;
- Se o palpite do jogador for igual a zero e ele não fez nenhum jogo então ganha 1 ponto.

O vencedor e aquele que fizer cinco pontos, podendo haver mais de um sem nenhum problema. Para isso criamos um for no tamanho da lista *placar*, e duas condicionais uma para verificar a ordem e outra, o jogador que fez 5 pontos.

```
984     #Printando o vencedor dependendo da ordem definida anteriormente
985     for i in range(len(placar)):
986         if primeiro == "Player 2":
987             if placar[i] == 5:
988                 #Printa o jogador na respectiva ordem do jogo
989                 print("O vencedor é", ordem[i])
990         elif primeiro == "Player 3":
991             if placar[i] == 5:
992                 print("O vencedor é", ordem2[i])
993         elif primeiro == "Player 4":
994             if placar[i] == 5:
995                 print("O vencedor é", ordem3[i])
996         elif primeiro == "Player 5":
997             if placar[i] == 5:
998                 print("O vencedor é", ordem4[i])
999         else:
1000            if placar[i] == 5:
1001                print("O vencedor é", ordem5[i])
1002
```

Figura 16 – Vencedores.

2.6 Se o jogador não for o Player 2

O processo e o mesmo a única parte que muda e a escolha dos palpites, temos então que:

Sem problema algum podemos remover da lista *ordem* o usuário e o ultimo jogador, criamos um for igual aos anteriores e adicionamos uma condicional que verifica a posição do usuário correspondente a ordem definida no jogo, por exemplo, na figura 17 o usuário e o terceiro a dar o palpite, então na posição dois o programa deve perguntar quantos jogos o usuário faz. Em seguida faremos a escolha do ultimo jogador, usando *random.choice()* escolhemos um numero aleatório da lista *jogos* e criamos um laço para validar a opção escolhida pelo computador, se a opção não satisfazer a regra da soma, então ele escolhe outro numero ate sair do laco e adicionar esse numero na lista *j*.

```
#Diminuimos a quantidade de jogadores
ordem = ["Player 3", "Player 4", "Player 5"]
#Para o jogador Player 3 e Player 4 escolha numeros aleatorio como visto anteriormente
#Quando k = 2 pergunte ao usuario pois ele e o terceiro nesta ordem
#Para o Player 5 segue igual ao outros
for k in range(len(ordem)):
    numero = random.choice(jogos)
    if ordem[k] == ordem[2]:
        seus = int(input("Quantos jogos você faz? "))
        soma += seus
        j.append(seus)
    print(ordem[k], " faz {}".format(numero))
    numero = int(numero)
    j.append(numero)
    soma += numero
#Para o ultimo escolhe um numero aleatorio para a quantidade de jogos que ele ira fazer
escolhap2 = random.choice(jogos)
escolhap2 = int(escolhap2)
#Restringindo como anteriormente so que enquanto ele escolher um numero cuja a soma fecha em 4, escolha novamente
#Ate a soma ser != 4 ou de forma geral como explicado anteriomente
while soma + escolhap2 == max(qnt):
    escolhap2 = random.choice(jogos)
    escolhap2 = int(escolhap2)
print("Player 2 faz {}".format(escolhap2))
j.append(escolhap2)
soma += escolhap2
```

Figura 17 – Escolha dos computadores.

OBS: Se o primeiro jogador for qualquer outro diferente do Player 2, segue as mesmas alterações feitas nessa sessão.

2.7 Bibliotecas Utilizadas

PyDealer é uma biblioteca Python simples de usar para "simular"baralhos de cartas convencionais, entre outras utilidades. Nessa bibliotecas há varias funções que foram usadas nesse trabalho. Exemplo de funções:

- *pydealer.Deck* Constrói um baralho de 52 cartas.

```
#Definindo o baralho tendo como ordem de valores e naipes as ordens padroes
baralho = pydealer.Deck(ranks=DEFAULT_RANKS)
```

Figura 18 – Baralho.

- *.deal(i)* Distribui "i"cartas removendo do baralho tais cartas distribuidas.

```
P2 = baralho.deal(4)
P3 = baralho.deal(4)
P4 = baralho.deal(4)
P5 = baralho.deal(4)
voce = baralho.deal(4)
```

Figura 19 – Distribuindo 4 cartas para os jogadores.

```
63          baralho = pydealer.Deck(ranks=DEFAULT_RANKS)
64          print("\n")
65          print("Embaralhando as cartas...\n")
66          baralho.shuffle()
```

Figura 20 – Embaralhando cartas.

- *.shuffle()* Embaralha as cartas.
- *carta1.gt(carta2, rank)* Compara duas cartas em um dado ranking definido em truco-ranks.

```
208         if carta_mesa2.gt(carta_mesa3, ranking) and carta_mesa2.gt(carta_mesa4, ranking)
209             print("Player 2 ganhou a rodada!")
210             pontos[0] += 1
```

Figura 21 – Comparando cartas.

Random é uma biblioteca utilizada para gerar números pseudo-aleatórios para distribuições, nesse caso foi utilizado como meio para definir as escolhas dos jogadores (computadores). Exemplo de funções :

- *random.choice()* Escolhe de forma aleatória um elemento de uma lista.

```
878         for k in range(len(ordem)) :
879             numero = random.choice(jogos)
880             print(ordem [ k ], " faz {}".format(numero))
```

Figura 22 – Escolhendo um numero aleatório da lista *jogos* para o computador.

2.8 Bibliotecas Criadas

Truco ranks foi criado pelos desenvolvedores deste trabalho com o intuito de definir ordem (maior e menor) das cartas com base no vira (definir manilhas)

- *RANK* Utilizado para enumerar do maior para o menor o valor das cartas com base no vira.

```
91          ranking = DEFAULT_RANKS
92          #Define o vira com base no valor da carta
93          if valor == "4":
94                  baralho = pydealer.Deck(ranks=RANK1)
95                  ranking = RANK1
```

Figura 23 – Rank para o vira igual a 4.

```
RANK1 = {
    "values": {
        "5": 10,
        "3": 9,
        "2": 8,
        "A": 7,
        "K": 6,
        "Q": 5,
        "J": 4,
        "7": 3,
        "6": 2,
        "4": 1,
    },
    "suits": {
        "Paus": 4,
        "Copas": 3,
        "Espadas": 2,
        "Ouros": 1
    }
}
```

Figura 24 – Manilha 5 recebendo o valor 10.

2.9 Funções criadas

A única função criada nesse trabalho foi a função *Bemvindo* e *Instruções* ambas as funções são apenas "explicativas"não sendo parte principal da execução do programa.

```
def bem_vindo():
    print("Bem vindo ao jogo do FODASE!!!")
    print("Caso você não conheça o jogo, segue as instruções abaixo!!!")


def instruções():
    print("Cartas disponíveis no baralho: A, 2, 3, 4, 5, 6, 7, Q, J e K")
    print("Naipes: Paus, Copas, Ourous e Espadas")
    print("A ordem dos naipes e dos valores das cartas segue do Truco!!!")
    print("Cada jogador deve informar a quantidade de jogos que faz!!!")
    print("OBSERVAÇÃO: A quantidade total de jogos não pode ser igual "
          "ao número de cartas da pessoa que tiver mais cartas.")
    print("A seleção de cartas deve ser feita digitando o valor que aparecer nas alternativas.")
    print("Só ganha ponto o jogador que fizer a quantidade de jogos que disse que faria!!!")

```

Figura 25 – Funções criadas.

Cada linha de ambas as funções são auto-explicativas, apenas mostram o texto correspondente as instruções e recepção do usuário.

3 Resultados e Dificuldades

3.1 Resultados

Com a finalização do programa, obtivemos êxito em fazer o jogo "Foda-se"na linguagem Python, porém com ajustes nas regras mas mantendo a maior parte do jogo, fiel ao original, além disso, conseguimos manter a dificuldade do jogo razoável e igualmente divertido.

3.2 Dificuldades

- Algumas partes da biblioteca Pydealer e definida utilizando classes e métodos, e como não foi umas das matérias abordadas durante o semestre, gerou a dificuldade em entender e aplicar no código.
- E houve dificuldades em definir quais seriam as jogadas dos computadores, e definir comando aleatórios que satisfizessem as regras do jogo.

3.3 Sugestões e Desafios

Como não conseguimos manter a originalidade total do jogo, propomos um desafio aos leitores:

Proposta: Com base no nosso programa e utilizando técnicas mais "avançadas"sera dificil criar a versão original do jogo "Foda-se". Além disso sera difícil criar uma versão online do mesmo?

Referências

1 PAGANINI, G. P.; MORAES, E. T. K. *Truco - py*. 2016. Disponível em: <https://github.com/gabrielpapke/truco-py>.

2 CRAWFORD, A. *PyDealer: Playing Card Package*. 2015. Disponível em: <https://pydealer.readthedocs.io/en/latest/>.

3 SILVA, R. da. *Introdução a Classes e Métodos em Python (básico)*. 2014. Disponível em: <http://pythonclub.com.br/introducao-classes-metodos-python-basico.html>.