Manual de Código do Jogão do Bacalhau

Alunos:
Gabriel Alexsander da Costa Pereira-2335590
Gabriel Duarte Spilka-2531003

1. Imports

```
import json
from random import *
from time import *
from collections import Counter
```

- Importa json, para ler o arquivo das perguntas o question.json .
- Importa tudo da livraria random e time, para números aleatórios e para adicionar delay.
- Importa Counter da livraria collections, para contar os numero de itens em um dicionário.

2. Variáveis globais

1-letters = ['a', 'b', 'c', 'd']

2-specialWords = ['pulo', '50', 'plat', 'plateia', 'uni', 'universitarios', 'parar', 'skip', 'pular', '50|50', 'metade', 'stop', 'para', 'eliminar', 'elimina']

3-stopWords = ['stop', 'parar', 'para']

4-platWords = ['plat', 'plateia']

5-cinWords = ['metade', '50', '50|50', 'eliminar', 'elimina']

6-uniWords = ['uni', 'universitarios']

7-skipWords = ['pular', 'pula', 'skip', 'pulo']

8-totalJumps = 0

9-total50 = 0

10-totalUni = 0

11-totalPlat = 0

12-totalPoints = 0

13-playerNick = ""

2. Variáveis globais

- Da linha 1 a 7, as variáveis são responsáveis para comparar a resposta dada para o Jogador!
- Da linha 8 a 13, serão as responsáveis por pegar o nome do Jogador e acompanhar quantos pontos e power ups usados!

3. Dicionários

```
Streaks = {
    "Posstreak": 0,
    "Negstreak": 0,
}

player = {
    "Quantidade_de_pontos": totalPoints,
    "Pulos_usados" : totalJumps,
    "50|50_usados": totalJumps,
    "Universitarios_usados": totalUni,
    "Plateias_usadas": totalUni,
    "Nome_do_jogador": playerNick,
}
```

- Dicionário Streaks é responsável por acompanhar quantos acertos e erros consecutivos o jogador tem!
- Dicionário player é responsável por guardar o nome do jogador, total de pontos e total de power ups usados.

4. Lendo as perguntas e respostas por um arquivo .json

```
1-file = open('questions.json', encoding="utf8", errors="ignore")
2-data = json.load(file)
```

- Primeira linha abre o arquivo das perguntas, codifica para utf-8 e ignora erros se encontrar algum na hora da leitura.
- Segunda linha 'analisa' o arquivo inteiro e salva.

4. Lendo as perguntas e respostas por um arquivo .json

- Assim, conseguindo ler o arquivo usando:

```
print(data["questions"])
#printa o arquivo questions.json inteiro

print(data["questions"][0])
#Prita primeira questao
```

5. Funções

I. ChooseCorrect()

- Descrição:

Dado o id da questão retorna a letra correta e uma lista de todas as respostas!

- Syntax

ChooseCorrect(int n)

- Parâmetros

n - um numero inteiro, que corresponde ao id da questão.

- Retorno

Retorna uma string e uma lista.

Exemplo:

```
print(ChooseCorrect(1))
#('b', ['Nao', 'Sim', 'TOGURO?', 'Nao sei'])
```

5. Funções

II. printdots()

- Descrição:

Espera 1 segundo e printa um '.' repete 3 vezes.

- Syntax:

printdots()

Exemplo:

```
printdots()
...
#Espera 1 segundo
.
#Espera 1 segundo
.
#Espera 1 segundo
.
#Espera 1 segundo
...
```

5. Funções

III. ShowQuestions()

- Descrição

Printa a pergunta e suas respostas, enumerando elas baseada no round atual.

- Syntax

ShowQuestions(lista(answers), int rounds, int ques)

- Parâmetros

answers - Lista com todas as respostas

rounds - round atual

ques - questão atual

5. Funções

III. ShowQuestions()

Exemplo:

```
questao = 1
correct, respostas = ChooseCorrect(questao)
ShowQuestions(respostas, 5, questao)
'''

_____________________________________________________

        Pergunta numero 5

_____________________________________________________

.
.
.

        P5--Conhece Elon musk?

_____________________________________________________

        Q1--a---Nao

_____________________________________________________

_____________________________________________________

        Q2--b---Nao sei

_____________________________________________________

_____________________________________________________

        Q3--c---TOGURO?

_____________________________________________________

_____________________________________________________

        Q4--d---Sim

_____________________________________________________

_____________________________________________________
'''
```

5. Funções

IV. fiftyfifty()

- Descrição

Elimina duas respostas erradas!

- Syntax

fiftyfifty(Lista(answers), string correct)

- Parâmetros

answers - Lista com todas as respostas

correct - String da resposta certa

5. Funções

IV. fiftyfifty()

Exemplo:

```
questao = 1
correct, respostas = ChooseCorrect(questao)
ShowQuestions(respostas, 5, questao)
'''
_____________________________________________________
        Pergunta numero 1
_____________________________________________________
.
.
.
        P1  Conhccc Elon musk?
_____________________________________________________
        Q1--a---Sim
_____________________________________________________
_____________________________________________________
        Q2--b---Nao
_____________________________________________________
_____________________________________________________
        Q3--c---TOGURO?
_____________________________________________________
_____________________________________________________
        Q4--d---Nao sei
_____________________________________________________
_____________________________________________________
'''
fiftyfifty(respostas, correct)
'''
Q1--a---Sim
Q2--c---TOGURO?
'''
```

5. Funções

V. DoPorcentage()

- Descrição

Calcula uma porcentagem bem nas 'coxas’(é bem nas coxa), pois não calcula realmente a porcentagem dos itens em um dicionário apenas conta o numero de itens e da uma porcentagem baseado nisso.

- Syntax

PlatCalculation(Lista(Arr1), Lista(arr2), string how)

- Parâmetros

Arr1 - Lista das respostas corretas

Arr2 - Lista das respostas erradas

how - como será feito o calculo => somente 'uni' e 'plat' fazem algo, 20 e 10 respectivamente

- Retorna

Retorna um dicionário com as porcentagem.

5. Funções

VI. PlatCalculation()

- Descrição

Apesar do nome, essa função calcula a porcentagem da plateia e dos universitários. O calculo da porcentagem é feito pela função DoPorcentage().

- Syntax

PlatCalculation(Lista(answers), string correct, int peps, string how, float num1, float num2)

5. Funções

VI. PlatCalculation()

- Parâmetros

answers - Lista de respostas.

correct - Letra correta.

num1 e num2(Opcional) - Ambos ajudam no calculo de escolher respostas de ambos o universitários e a plateia| sempre menores que num2 - age como uma segunda chance, valor default = 0 .

peps - Quantidades de pessoas, se traduz na quantidade de vezes que será loopado.

how - como mostrado o resultado final.

Exemplo:

```
correct, respostas = ChooseCorrect(questao)
PlatCalculation(respostas, correct, 0.3, 0.2, 10, 'plat')
#A plateia votou nas seguinte questoes {'a': '30%', 'b': '30%', 'c': '20%', 'd': '20%'}
```

5. Funções

VII. ShowQuestionVar()

- Descrição

Mostra as questões de maneiras diferente, baseados no 50|50, plateia e universitários.

- Syntax

ShowQuestionVar(Lista(answers), string how, int ques).

- Parâmetros

answers - Lista de respostas.

how - Qual versão mostrar| plat, 50 e uni.

ques - id da questão atual.

- Modo de funcionar

how = 'plat' -> Chama funco PlatCalculation()

how = 'uni' -> Chama funcao PlatCalculation()

how = '50' -> Chama funcao fiftyfifty()

5. Funções

VIII detectCase()

- Descrição

Baseado no input do jogador, detecta se é uma palavra especial.

- Syntax

detectCase(string word)

- Parâmetros

word - palavra a ser analisada.

- Retorno

Retorna 'skip' se o input esta em skipWords.
Retorna 'uni' se o input esta em uniWords.
Retorna 'stop' se o input esta em stopWords.
Retorna 'plat' se o input esta em platWords.
Retorna 'cin' se o input esta em cinWords.

5. Funções

VIII. detectCase()

Exemplo:

```
print(detectCase('plat'))
#plat
```

5. Funções

IX. CalcPoints()

- Descrição

Calcula a pontuação do jogador, a formula é dada por round(rounds**2 + 100/2).

- Syntax

CalcPoints(int rounds, string sign, int streak).

- Parâmetros

rounds - round atual.

sign - + ou - em string, para saber se o resultado é positivo ou negativo.

streak - quantidades de acertos ou erros consecutivos desde o ultimo acerto/erro.

- Retorno

se sign = '+' -> Retorna round(formula * streak / 1.7)

se sign = '-' -> round(-formula * streak / 1.3)

5. Funções

IX. CalcPoints()

Exemplo:

```
print(CalcPoints(5, '-', 4))
#-231
```

5. Funções

X. GetAnswer()

- Descrição

Pega a resposta do jogador para a pergunta atual, sendo perguntada.

- Syntax

GetAnswer(int num, int rounds)

- Parâmetros

num - id da questão

rounds - round atual

5. Funções

XI. ArrayGenerator()

- Descrição

Gera Lista entre 0 e o tamanho especificado.

- Syntax

ArrayGenerator(int n)

- Parâmetros

n - tamanho da lista a ser gerada

- Retorno

Retorna a lista criada.

Exemplo:

```
print(ArrayGenerator(5))
#[0, 1, 2, 3, 4]
```

5. Funções

XII. ShuffleArray()

- Descrição

Embaralha uma lista

- Syntax

ShuffleArray(list(arr))

- Parâmetros

arr - Lista a ser embaralhada.

Retorno

retorna a lista embaralhada.

Exemplo:

```
print(ShuffleArray([0, 1, 2, 3, 4]))
#[3, 2, 0, 1, 4]
```

XIII. SaveResults()

- Descrição

Cria ou atualiza um arquivo data.json salvando o dicionário player.

6. Funcao Main()

- Essa é a função que será chamada para iniciar o jogo.
- Depende de todas as outras funções anteriores.

```
def Main():
    num = 0
    totalQuestion = len(data["questions"])
    gameIsPlaying = False
    listOfQuestions = ArrayGenerator(totalQuestion)
    ShuffleArray(listOfQuestions)
    gameIsPlaying = True
    playerNick = input("Digite seu nome!\n")
    player["Nome_do_jogador"] = playerNick
    if(playerNick == ""):
        player["Nome_do_jogador"] = "Gamer"
    rounds = 1
    roundPlayed = 0
    while(gameIsPlaying):
        clause = GetAnswer(listOfQuestions[num], rounds)
        sleep(1)
        rounds += 1
        roundPlayed += 1
        num += 1
        print(f"Voce tem: {player['Quantidade_de_pontos']} pontos!")
        if(clause == 'STOP'):
            print("Voce escolheu parar")
            break
        elif(clause == 'SKIP'):
            print("Voce Pulou a questao")
        if(roundPlayed >= totalQuestion):
            print("Acabou!!")
            break
    SaveResults()


Main()
```