



O jogo da forca



Objetivo

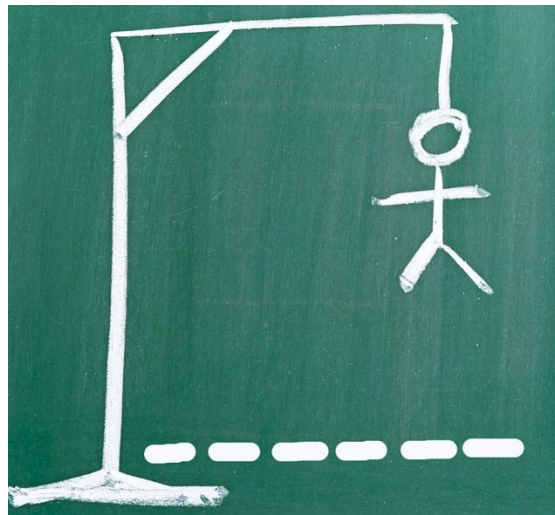
O projeto tem como base o exercitar boa parte do conteúdo visto até então (e ampliar os horizontes).

- Tipos primitivos (int, boolean)
- Coleções (arrays)
- Strings
- Funções
- Extrair uma situação real para o código (Lógica)
- **Aprender**

Lógica e implementação

Um simples jogo de conhecimentos sobre palavras. E vamos usar essa simplicidade para treinar a nossa lógica!

Antes de codificar, vamos pensar e extrair todas as regras do jogo





Regras do jogo (1/3)

- Uma palavra secreta deve ser sorteada aleatoriamente e seus caracteres trocados por "_" (ou qualquer outro caractere que não seja igual a letra original da posição)
- O usuário deve selecionar uma letra, caso a letra exista, a letra selecionada deve ser colocada na posição da palavra secreta (substituindo o caracter "_")
- Caso a letra não exista, o usuário perde uma vida



Regras do jogo (2/3)

- O usuário deve ser informado de quantas vidas ainda tem e quais as letras já foram tentadas (tanto das letras que existem ou que não existem)
- Caso o usuário tente digitar uma letra que já foi utilizada, o jogo deve pedir que outra letra seja selecionada



Regras do jogo (3/3)

- O jogo acaba quando o usuário perde todas as vidas, ou se todas as letras da palavra secreta sejam descobertos
- Quando o jogo acabar, o resultado deve ser exibido (se ganhou ou perdeu). A palavra secreta deve ser exibida em caso de derrota



Pontos importantes

- As letras devem estar sempre em *lowercase* (tanto as tentativas, quanto a palavra secreta)
- Atenção a palavras com caracteres repetidos, por exemplo, digamos que a palavra sorteada seja "carro". Se o usuário digitar a letra "r" deverá ser exibido "__ r r_" ao usuário.
- Qualquer tipo de acento deverá ser removido da palavra secreta, para facilitar a vida do usuário. Dado que a palavra secreta "____" (você) e o usuário digitar "e" deverá ser exibido "___ê"

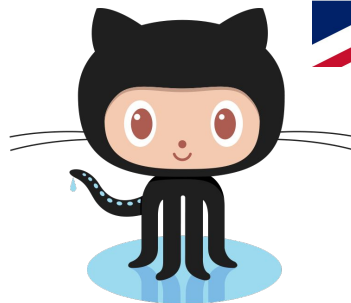
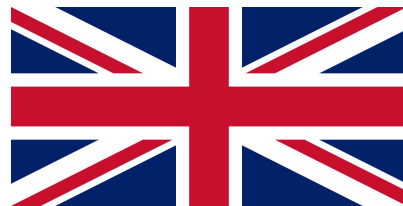
Especificações técnicas

O projeto deverá seguir as boas práticas do Python (PEP-8)

Toda a codificação deverá ser feita em **inglês**. *(o jogo pode ser em português)*

Deve ser criado um **repositório privado** no github para o projeto

**se não souber, pesquise*

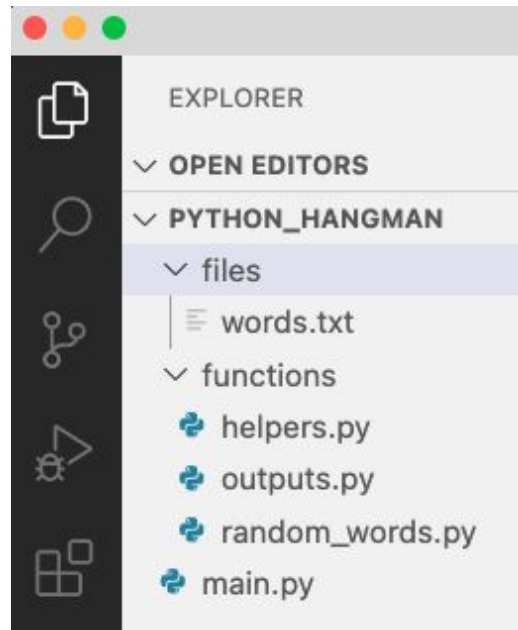


Requisitos do projeto

O projeto deve ter (pelo menos) a seguinte configuração

- main.py
- files/words.txt
- functions/helpers.py
- functions/outputs.py
- functions/random_words.py

(pode adicionar mais arquivos caso queira)





main.py

Deverá tratar o fluxo original do projeto.
Consumindo as funções dos outros arquivos

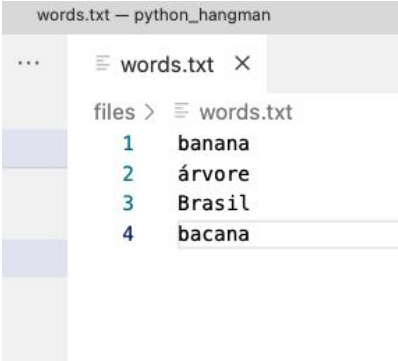
Deve ser feito um **import** dos arquivos/funções do diretório de functions

**se não souber, pesquise*

files/words.txt

O arquivo contém a lista de nomes a serem sorteados. Separados por *enters* (\n)

Exemplo do conteúdo do *words.txt*:



```
words.txt — python_hangman
...
files > words.txt
1 banana
2 árvore
3 Brasil
4 bacana
```



functions/outputs.py

O arquivo de outputs **deverá obrigatoriamente** agregar e **utilizar** as seguintes funções:

- `game_start()` deverá printar na tela que o jogo começou
- `you_lost()` deverá informar que o jogo acabou e o usuário perdeu
- `you_won()` deverá informar que o jogo acabou e o usuário ganhou
- `reveal_secret_word(word)` ser passado por parâmetro a palavra secreta, e exibir ao usuário (só depois que o jogo acabar)
- `doesnt_exists(letter)` deverá informar ao usuário que a letra em questão não existe e selecionar outra

```
def game_start():  
def you_lost():  
def you_won():  
def reveal_secret_word(word):  
def doesnt_exists(letter):
```



functions/outputs.py

O arquivo de outputs **deverá obrigatoriamente** agregar e **utilizar** as seguintes funções:

- *users_life()* deverá imprimir mostrar ao usuário quantas vidas ainda tem
- *letters_tried(letters)* deverá informar ao usuário quais letras ele já usou. A função vai receber como parâmetro **um array** letras já utilizadas e exibir na tela de forma amigável a lista de letras

```
def users_life(life):  
def letters_tried(letters):
```



functions/outputs.py

- **Nenhuma** das funções dentro do arquivo output terão retornos todas vão ser to tipo *void*
- As funções deverão exibir na tela mensagens amigáveis e claras

Exemplo de mensagens amigáveis:

"Parabéns, você ganhou!" (exemplo de implementação para função `you_won()`)

"Desculpe, mas a letra (a) não existe na palavra secreta" (exemplo de implementação para função `doesnt_exists(letter)`)

**se não souber, pesquise*



functions/helpers.py

Deverá ser implementado **OBRIGATORIAMENTE** as seguintes

funções:

- `change_word_to_underline(word)` deve receber uma palavra e trocar por underlines intercalados por espaços. Essa **função retorna uma STRING**

Exemplo:

```
print(change_word_to_underline("banana"))  
//o valor do print será -> _ _ _ _ _
```

```
def change_word_to_underline(word)-> str:  
def remove_special_characters(word)-> str:  
def check_if_exists_on_word(word, letter) -> bool:  
def show_correct_letters(word, tries) -> str:  
def parse_input(input)-> str:
```



functions/helpers.py

Deverá ser implementado **OBRIGATORIAMENTE** as seguintes

funções:

- `remove_special_characters(word)` deve receber uma palavra e remover qualquer tipo de acentuação/hifens/espacos. Essa **função retorna uma STRING**

Exemplo:

```
print(remove_special_characters("exceção"))  
//o valor do print será -> excecao
```

```
def change_word_to_underline(word)-> str:  
def remove_special_characters(word)-> str:  
def check_if_exists_on_word(word, letter) -> bool:  
def show_correct_letters(word, tries) -> str:  
def parse_input(input)-> str:
```



functions/helpers.py

Deverá ser implementado **OBRIGATORIAMENTE** as seguintes funções:

- `check_if_exists_on_word(word, letter)` Deverá receber uma palavra e uma letra, e retornar se a *letter* existe ou não em *word*. Essa **função retorna um BOOLEANO**

Exemplo:

```
print(check_if_exists_on_word("banana", "a"))  
//o valor do print será -> True
```

```
def change_word_to_underline(word)-> str:  
def remove_special_characters(word)-> str:  
def check_if_exists_on_word(word, letter) -> bool:  
def show_correct_letters(word, tries) -> str:  
def parse_input(input)-> str:
```




functions/helpers.py

Deverá ser implementado **OBRIGATORIAMENTE** as seguintes

funções:

- `show_correct_letters(word, tries)` Deverá receber uma palavra (*word*) e um array de letras (*tries*), e retorna uma string com as letras que existem em *word*. Essa **função retorna uma STRING**

Exemplo:

```
print(show_correct_letters("banana", ["b", "n"]))  
//o valor do print será -> "b_n_n_"
```

```
def change_word_to_underline(word) -> str:  
def remove_special_characters(word) -> str:  
def check_if_exists_on_word(word, letter) -> bool:  
def show_correct_letters(word, tries) -> str:  
def parse_input(input) -> str:
```



functions/helpers.py

Deverá ser implementado **OBRIGATORIAMENTE** as seguintes

funções:

- `parse_input(input)` será a função responsável por receber a letra digitada pelo usuário, essa função deve retornar um caractere. Essa **função retorna uma STRING**

Exemplo:

```
parse_input()  
//Favor digite uma letra: a
```

```
def change_word_to_underline(word)-> str:  
def remove_special_characters(word)-> str:  
def check_if_exists_on_word(word, letter) -> bool:  
def show_correct_letters(word, tries) -> str:  
def parse_input(input)-> str:
```



functions/helpers.py

A implementação das funções em helpers é **obrigatória**, porém o seu uso no projeto é **opcional**

Todas as funções em helpers fazem uso de ***type hinting***. Caso o seu python seja inferior ao 3.5 o type hinting não vai funcionar, mas cada uma das funções deverá retornar o que foi descrito nos slides anteriores

**se não souber, pesquise*



functions/random_words.py

O arquivo de random_words.py **deverá obrigatoriamente** agregar e **utilizar** as seguinte função:

```
def get_random_name() -> str:
```

Esse arquivo terá apenas uma função responsável por retornar uma string aleatória.

A função deverá ler o arquivo `files/words.txt` e aleatoriamente retornar uma das palavras lá contidas.

**se não souber, pesquise*



Entrega

- O projeto deve ser entregue dia 20/07
 - 16/06 (Implementação do arquivo `random_words.py`)
 - 20/06 (Implementação do `outputs.py`)
 - 06/07 (Implementação do `helpers.py`)
 - 20/07 **Projeto completo**
- Deve ser feito uma apresentação para explicar o código (10 min)

Seja criativo!

A única obrigação é ter um jogo funcional seguindo as regras e implementando as funções descritas.

Seja livre para adicionar **funcionalidades**, *libraries* e/ou outros assuntos do Python!

