01

Organizando o código em funções

Transcrição

A função `jogar()` possui um código muito complexo, com muitas funcionalidades e responsabilidades.

Entre as funcionalidades que o código possui, está a apresentação do jogo, leitura do arquivo e inicialização da palavra secreta, entre outras. Vamos então separar as responsabilidades do código em **funções**, melhorando a sua legibilidade e organização.

Função que imprime a mensagem de apresentação do jogo

Vamos começar com a mensagem de apresentação do nosso jogo, vamos exportar o código para a função `imprime_mensagem_abertura()`. Não podemos nos esquecer de chamar essa função no início da função `jogar()`:

```
import random

def imprime_mensagem_abertura():
    print("**********************************")
    print("***Bem vindo ao jogo da Forca!***")
    print("**********************************")

def jogar():

    imprime_mensagem_abertura()

    # restante do código omitido
```

Não importa o local da função, ela pode ser declarada antes ou depois da função `jogar()`.

Função de leitura do arquivo e inicialização da palavra secreta

Agora, vamos separar o código que realiza a leitura do arquivo e inicializa a palavra secreta na função `carrega_palavra_secreta()`:

```
import random

def jogar():

    imprime_mensagem_abertura()

    carrega_palavra_secreta()

    letras_acertadas = ["_" for letra in palavra_secreta]

    # restante do código omitido

def carrega_palavra_secreta():
    arquivo = open("palavras.txt", "r")
```

```
    palavras = []

    for linha in arquivo:
        linha = linha.strip()
        palavras.append(linha)

    arquivo.close()

    numero = random.randrange(0, len(palavras))
    palavra_secreta = palavras[numero].upper()
```

Só que a função `jogar()` irá reclamar que a `palavra_secreta` não existe. O que queremos é que, ao executar a função `carrega_palavra_secreta()`, que ela retorne a palavra secreta para nós, assim poderemos guardá-la em uma variável:

```
import random

def jogar():

    imprime_mensagem_abertura()

    palavra_secreta = carrega_palavra_secreta()

    letras_acertadas = ["_" for letra in palavra_secreta]

    # restante do código omitido
```

Retornando um valor em uma função

Só que como faremos a função `carrega_palavra_secreta()` retornar um valor, no caso a `palavra_secreta`?

A `palavra_secreta` já existe, mas só dentro da função `carrega_palavra_secreta()`. Para que ela seja retornada, utilizamos a palavra-chave `return`:

```
def carrega_palavra_secreta():
    arquivo = open("palavras.txt", "r")
    palavras = []

    for linha in arquivo:
        linha = linha.strip()
        palavras.append(linha)

    arquivo.close()

    numero = random.randrange(0, len(palavras))
    palavra_secreta = palavras[numero].upper()

    return palavra_secreta
```

Passando valores por parâmetro para a função

Agora vamos criar uma função que inicializa a lista de letras acertadas com o caractere `_` . Criaremos a função `inicializa_letras_acertadas()` :

```
import random

def jogar():

    imprime_mensagem_abertura()

    palavra_secreta = carrega_palavra_secreta()

    letras_acertadas = inicializa_letras_acertadas()

    # restante do código omitido

def inicializa_letras_acertadas():
    return ["_" for letra in palavra_secreta]
```

Mas a função `inicializa_letras_acertadas()` precisa ter acesso à `palavra_secreta` , pois ela não existe dentro da função, já que uma função define um **escopo**, e as variáveis declaradas dentro de uma função só estão disponíveis **dentro dela**.

Então, ao chamar a função `inicializa_letras_acertadas()` , vamos passar `palavra_secreta` para ela por parâmetro:

```
import random

def jogar():

    imprime_mensagem_abertura()

    palavra_secreta = carrega_palavra_secreta()

    letras_acertadas = inicializa_letras_acertadas(palavra_secreta)

    # restante do código omitido

def inicializa_letras_acertadas(palavra):
    return ["_" for letra in palavra]
```

Vocês podem estar se perguntando por que encapsulamos uma simples linha de código em uma função. Fizemos isso somente para deixar claro o que estamos fazendo, melhorando a legibilidade do código, mas precisamos tomar cuidado com a criação de funções, pois criar funções desnecessariamente pode aumentar a complexidade do código.

Por fim, podemos executar o nosso código, para verificar que o mesmo continua funcionando normalmente. Lembrando que o código que verifica se o programa é o principal, deve ficar no final do arquivo:

```
import random

def jogar():

    imprime_mensagem_abertura()
    palavra_secreta = carrega_palavra_secreta()
```

```
    letras_acertadas = inicializa_letras_acertadas(palavra_secreta)

    # restante do código omitido

def imprime_mensagem_abertura():
    print("*********************************")
    print("***Bem vindo ao jogo da Forca!***")
    print("*********************************")

def carrega_palavra_secreta():
    arquivo = open("palavras.txt", "r")
    palavras = []

    for linha in arquivo:
        linha = linha.strip()
        palavras.append(linha)

    arquivo.close()

    numero = random.randrange(0, len(palavras))

    palavra_secreta = palavras[numero].upper()

    return palavra_secreta

def inicializa_letras_acertadas(palavra):
    return ["_" for letra in palavra]

if (__name__ == "__main__"):
    jogar()
```

Tudo continua funcionando normalmente, mas agora com o nosso código um pouco mais legível e organizado. No próximo vídeo extrairemos mais código para mais funções.