Lista_03

September 22, 2020

1 Lista de Exercícios 3

*Alguns exercícios foram retirados do livro: The Python Workbook - Ben Stephenson.

[1]:
```
import random as rd
```

1.1 QUESTÃO 1:

Crie uma função que retorne uma lista contendo todas as sublistas possíveis dessa lista. Por exemplo, as sublistas de $[1, 2, 3]$ são [], [1], [2], [3], $[1, 2]$, $[2, 3]$ e $[1, 2, 3]$. Observe que sua função sempre retornará uma lista contendo pelo menos a lista vazia porque a lista vazia é uma sublista de todas as listas.

[ ]:
```

```

1.2 QUESTÃO 2:

Duas palavras são **anagramas** se possuírem todas as mesmas letras (e na mesma quantidade), mas em uma ordem diferente. Por exemplo, "amor" e "roma" são anagramas, assim como "iracema" e "america". Note que não estamos levando em consideração letras maiúsculas nem acentuação.

Sua tarefa é criar uma função que recebe como parâmetros de entrada duas palavras e que determina se são ou não anagramas.

DICA 1: Talvez seja melhor você criar uma função auxiliar que conta a quatidade de ocorrência de cada letra. Pense em uma função em que a saída seja uma lista de listas, em que cada sublista é formada por dois elementos: o primeiro é a letra e o segundo a quantidade de vezes em que ela aparece na palavra.

DICA 2: Não é necessário, mas se você quiser fazer um tratamento da palavra antes de executar o código, você pode usar o procedimento `.lower()` para tornar todas as letras minúsculas. Também existe a função `unidecode()` da biblioteca `unidecode` que remove o acentos da palavra, mas essa biblioteca pode precisar de instalação. Por hora, não é necessário se preocupar com isso.

[2]:
```
#!pip3 install unidecode            #Caso a biblioteca não esteja instalada,␣
 ↪execute essa linha!
from unidecode import unidecode     #Importa somente a função unidecode da␣
 ↪biblioteca unidecode
```

[3]:
```
unidecode('áéíóúàç')
```

[3]:
```
'aeiouac'
```

[ ]:
```

```

1.3 QUESTÃO 3:

Crie uma função que simule um lançamento de dados. Lance o dado 100 vezes e armazene os resultados em um lista de listas que mostra a quantidade de vezes cada valor (de 1 a 6) foi obtido. Ou seja, cada sublista é formada por dois elementos: o valor e o número de vezes que ele apareceu.

Em seguida, crie outra função que encontra qual o valor que saiu mais vezes e qual foi essa quantidade (ou seja, a saída é uma tupla ou uma lista de dois elementos).

[ ]:
```

```

1.4 QUESTÃO 4:

Inicialmente, crie uma função que sorteia n (parâmetro de entrada) cartas de um baralho, podendo ocorrer repetição.

Em seguida cria outra função que sorteia essas n cartas sem que ocorra repetição. Uma **dica**, nesse caso, é criar um baralho como uma lista de listas, ou seja, cada sublista representa uma carta, em que o primeiro elemento é o número da carta e o segundo, o naipe dela.

Um baralho é formado por 52 cartas (números de 2 a 10, Valete (11), Rainha (12), Rei (13) e Ás (1)), dividido em 4 naipes (copas, ouros, paus e espadas).

DICA 1: As cartas nomeadas também correspondem a números. Trabalhe com as cartas variando de 1 a 13.

DICA 2: Na biblioteca `random` existe a função `choice()`, que recebe como parâmetro de entrada uma lista e sorteia aleatoriamente um dos elementos dela. Você pode usar essa função para sortear o naipe de uma carta.

[4]:
```
naipe=rd.choice(['copas','ouros','paus','espadas'])
naipe
```

[4]:
```
'copas'
```

[ ]:
```

```

1.5 QUESTÃO 5:

Na lista anterior (questão 6 da lista 2), foi pedido que você criasse uma função que encontrasse o número mínimo de vezes que você tem que jogar uma moeda antes de poder ter três lançamentos consecutivos que resultassem no mesmo resultado (ou todos os três são CARA (C) ou todos os três são COROA (K)).

Agora, crie uma função que execute uma simulação para saber quantos lançamentos são necessários fazer em média para obter tal feito? Para isso, você pode executar a função criada umas 100 vezes e calcular a média dos resultados.

[ ]: