Computação I - Python
Aula 8: Estrutura de Repetição: for

Diferenças de uso entre while e for

Apresentado por: Rafael Machado Andrade

Produção DCC-UFRJ

Metodologia de referência https://doi.org/10.5753/wei.2016.9683

Estrutura de Repetição *for*

Faça uma função, chamada **sortear6**, que sorteie 6 números entre 1 e 10, e retorne uma lista contendo os números sorteados. Não podem ser sorteados números repetidos.

Estrutura de Repetição *for*

Faça uma função, chamada **sortear6**, que sorteie 6 números entre 1 e 10, e retorne uma lista contendo os números sorteados. Não podem ser sorteados números repetidos.

```
from random import randint
def sorteia6():
  '''Funcao que sorteia 6 numeros entre 1 e 10, sem repeticoes,
  insere-os em uma lista e retorna essa lista.
  None -> list'''

  lista = []
  for x in range(6):
    numero = randint(1,10)
    if numero not in lista:
      list.append(lista, numero)

  return lista
```

Um número sorteado só é inserido na lista se ele ainda não tiver sido inserido anteriormente. Os comandos dentro do for serão executados exatamente 6 vezes. Assim, se um número for sorteado mais de uma vez, a lista retornada terá menos de 6 elementos $->$ ERRADO!

Estrutura de Repetição *for*

IMPORTANTE: diferenças entre o uso de *while* e *for*:

- **while**: Decisão sobre repetir ou não baseia-se em teste booleano. Pode ser usado em situações onde não é possível determinar a priori quantas repetições devem ser realizadas, desde que essa quantidade possa ser definida por um teste booleano. Há risco de loop infinito.
- **for**: Contagem automática do número de repetições. Caso hajam condições booleanas que influenciem na quantidade de repetições, o for pode não ser adequado.

Estrutura de Repetição *for*

Faça uma função chamada **sortear6**, que sorteie 6 números entre 1 e 10, e retorne uma lista contendo os números sorteados. Não podem ser sorteados números repetidos.

Versão corrigida utilizando while:

```
from random import randint
def sorteia6_While():
  '''Funcao que sorteia 6 numeros entre 1 e 10, sem repeticoes,
  insere-os em uma lista e retorna essa lista.
  None -> list'''

  lista = []
  while len(lista) < 6:
    numero = randint(1,10)
    if numero not in lista:
      list.append(lista, numero)

  return lista
```

Como há um critério para se inserir um número sorteado na lista, que pode ser definido com um teste booleano, é recomendado utilizar a estrutura de repetição **while** ao invés do **for**.

Estrutura de Repetição *for*

Faça uma função que receba como argumento uma frase, e retorne essa frase com todas as suas consoantes em caixa alta (letras maiúsculas).

Estrutura de Repetição *for*

Faça uma função que receba como argumento uma frase, e retorne essa frase com todas as suas consoantes em caixa alta (letras maiúsculas).

Versão utilizando utilizando while:

```
def consoantesMaiusculasWhile(frase):
    '''Funcao que recebe uma frase e transforma
    todas as suas consoantes em maiusculas
    str -> str'''
    frase_tratada = ''
    i = 0
    while i < len(frase):
        caractere = frase[i]
        if caractere in 'bcdfghjklmnpqrstwxyz ':
            caractere = str.upper(caractere)
        frase_tratada = frase_tratada + caractere
        i = i + 1
    return frase_tratada
```

Ao se utilizar while, precisamos definir uma variável contadora, referenciar cada caractere da string utilizando essa variável como posição e incrementá-la ao final de cada iteração.

Estrutura de Repetição *for*

Faça uma função que receba como argumento uma frase, e retorne essa frase com todas as suas consoantes em caixa alta (letras maiúsculas).

Versão utilizando utilizando for:

```
def consoantesMaiusculasFor(frase):
    '''Funcao que recebe uma frase e transforma
    todas as suas consoantes em maiusculas
    str -> str'''
    frase_tratada = ''
    for caractere in frase:
        if caractere in 'bcdfghjklmnpqrstwxyz ':
            caractere = str.upper(caractere)
        frase_tratada = frase_tratada + caractere
    return frase_tratada
```

Ao se utilizar for, não utilizamos uma variável contadora
e nem referenciamos cada caractere da string através de sua posição,
já que fazemos diretamente a atribuição do valor de cada caractere à variável do for.

Estrutura de Repetição *for*

Um erro de tipo será gerado ao se executar um comando for especificando para ser percorrido um elemento não indexável ao invés de um elemento indexável:

```
>>> def consoantesMaiusculasFor(frase):
    '''Funcao que recebe uma frase e transforma
    todas as suas consoantes em maiusculas
    str -> str'''
    frase_tratada = ''
    for caractere in frase:
        if caractere in 'bcdfghjklmnpqrstwxyz ':
            caractere = str.upper(caractere)
        frase_tratada = frase_tratada + caractere
    return frase_tratada

>>> consoantesMaiusculasFor(3.14)
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    consoantesMaiusculasFor(3.14)
  File "<pyshell#1>", line 6, in consoantesMaiusculasFor
    for caractere in frase:
TypeError: 'float' object is not iterable
```

Autores

- João C. P. da Silva ▸ Lattes
- Carla Delgado ▸ Lattes
- Ana Luisa Duboc ▸ Lattes

Colaboradores

- Anamaria Martins Moreira ▸ Lattes
- Fabio Mascarenhas ▸ Lattes
- Leonardo de Oliveira Carvalho ▸ Lattes
- Charles Figueiredo de Barros ▸ Lattes
- Fabrício Firmino de Faria ▸ Lattes

Computação I - Python
Aula 8: Estrutura de Repetição: for

Diferenças de uso entre while e for

Apresentado por: Rafael Machado Andrade

Produção DCC-UFRJ

Metodologia de referência https://doi.org/10.5753/wei.2016.9683