Glossário de funções de Python 3

Tipos sequenciais em geral

`len(seq)`

Parâmetros:

- `seq`: um dado de tipo sequencial (lista, tupla, string, dicionário, etc.).

Efeito: nenhum
Retorno: o comprimento da sequência `seq`.

`sum(seq)`

Parâmetros:

- `seq`: um dado de tipo sequencial contendo apenas valores numéricos.

Efeito: nenhum
Retorno: a soma de todos os valores da sequência.

Listas

`del(nome[indice])`

Parâmetros:

- `nome`: uma lista.
- `indice`: um índice da lista `nome`.

Efeito: Remove o elemento que ocupa o índice `indice` da lista `nome`.
Retorno: nenhum

`list.append(nome, elemento)`

Parâmetros:

- `nome`: uma lista.
- `elemento`: um valor qualquer.

Efeito: adiciona `elemento` ao final da lista `nome`
Retorno: nenhum

`list.count(nome, elemento)`

Parâmetros:

- `nome`: uma lista.
- `elemento`: um valor qualquer.

Efeito: nenhum
Retorno: o número de vezes que `elemento` aparece na lista `nome`.

`list.extend(nome, seq)`

Parâmetros:

- `nome`: uma lista.
- `seq`: uma sequência qualquer.

Efeito: adiciona cada elemento da sequência `seq` ao final da lista `nome`, um por vez.
Retorno: nenhum
Exemplo:

```
>>> x = ['hugo']
>>> list.extend(x, 'nobrega')
>>> x
['hugo', 'n', 'o', 'b', 'r', 'e', 'g', 'a']
```

`list.index(nome, elemento, inicio=0, fim=len(nome))`

Parâmetros:

- `nome`: uma lista.
- `elemento`: um elemento que apareça na lista `nome[inicio:fim]`.
- `inicio, fim`: índices da lista `nome`.

Efeito: nenhum
Retorno: o primeiro índice entre `inicio` e `fim`-1 (inclusive) no qual apareça `elemento` na lista `nome`
Exemplo:

```
>>> x = ['a', 'b', 'c', 'a', 'd']
>>> list.index(x, 'a', 2, 4)
3
```

`list.insert(nome, elemento, indice)`

Parâmetros:

- `nome`: uma lista.
- `elemento`: um valor qualquer.
- `indice`: um índice da lista `nome`

Efeito: adiciona `elemento` à lista `nome` na posição `indice`, "empurrando" os outros elementos da lista para a direita conforme necessário.
Retorno: nenhum
Exemplo:

```
>>> x = ['a', 'b', 'c']
>>> list.insert(x, 1, 'd')
>>> x
['a', 'd', 'b', 'c']
```

`list.pop(nome, indice=-1)`

Parâmetros:

- `nome`: uma lista.
- `indice`: um índice da lista `nome`.

Efeito: remove o elemento que ocupa o índice `indice` da lista `nome`.
Retorno: o elemento removido.
Exemplo:

```
>>> x = ['a', 'b', 'c']
>>> y = list.pop(x, 0)
>>> x
['b', 'c']
>>> y
'a'
```

`list.remove(nome, elemento)`

Parâmetros:

- `nome`: uma lista.
- `elemento`: um elemento que apareça na lista `nome`.

Efeito: remove a primeira ocorrência de `elemento` da lista `nome`
Retorno: nenhum
Exemplo:

```
>>> x = ['a', 'b', 'c', 'a']
>>> list.remove(x, 'a')
>>> x
['b', 'c', 'a']
```

`list.reverse(nome)`

Parâmetros:

- `nome`: uma lista.

Efeito: inverte a ordem dos elementos da lista `nome`
Retorno: nenhum

`list.sort(nome)`

Parâmetros:

- `nome`: uma lista.

Efeito: ordena os elementos da lista `nome` em ordem crescente.
Retorno: nenhum

`range(n)`

Parâmetros:

- `n`: um número inteiro

Efeito: nenhum
Retorno: uma sequência contendo todos os números inteiros de 0 a $n-1$ (incluindo 0 e $n-1$).

`range(n, m, k=1)`

Parâmetros:

- `n, m, k`: números inteiros

Efeito: nenhum
Retorno: uma sequência contendo todos os números inteiros $n, n+k, n+2k, n+3k, \ldots$, parando no maior destes números que não iguale nem ultrapasse m.
Exemplo:

```
>>> range(10, 28, 4)
[10, 14, 18, 22, 26]
```

Strings

`str.count(nome, substring, inicio=0, fim=len(nome))`

Parâmetros:

- `nome`: uma string

Efeito: nenhum
Retorno: o número de vezes que `substring` ocorre como parte da string `nome[inicio:fim]`.
Exemplo:

```
>>> x = 'Casa da asa na brasa'
>>> str.count(x, 'asa', 2, 150)
2
```

`str.find(nome, substring, inicio=0, fim=len(nome))`

Parâmetros:

- `nome, substring`: duas strings.
- `inicio, fim`: dois índices da string `nome`.

Efeito: nenhum
Retorno: o primeiro índice entre `inicio` e `fim`-1 (inclusive) no qual comece uma ocorrência da string `substring` dentro da string `nome`, ou o número -1 caso esse índice não exista.
Exemplo:

```
>>> x = 'Casa da asa na brasa'
>>> str.find(x, 'asa', 10, len(x))
17
```

`str.index(nome, substring, inicio=0, fim=len(nome))`

Parâmetros:

- `nome, substring`: duas strings, sendo que `substring` ocorre na string `nome[inicio:fim]`.
- `inicio, fim`: dois índices da string `nome` (opcionais; caso não sejam fornecidos a busca é na lista inteira)

Efeito: nenhum
Retorno: o primeiro índice entre `inicio` e `fim`−1 (inclusive) no qual comece uma ocorrência da string `substring` dentro da string `nome`.
Exemplo:

```
>>> x = 'Casa da asa na brasa'
>>> str.index(x, 'asa')
1
```

`str.join(separador, nome)`

Parâmetros:

- `separador`: uma string
- `nome`: uma sequência de strings

Efeito: nenhum
Retorno: uma string composta de todos os elementos da lista `nome`, com o `separador` inserido entre eles.
Exemplo:

```
>>> x = ['uma', 'lista', 'de', 'strings']
>>> sep = '#%$'
>>> str.join(sep, x)
'uma#%$lista#%$de#%$strings'
```

`str.lower(nome)`

Parâmetros:

- `nome`: uma string

Efeito: nenhum
Retorno: a string `nome` com todos os caracteres em letras minúsculas.

`str.partition(nome, separador)`

Parâmetros:

- `nome, separador`: duas strings.

Efeito: nenhum
Retorno:

- se `separador` ocorre como parte de `nome`, então retorna-se uma tupla cujo primeiro elemento é a fatia da string `nome` antes da primeira ocorrência de `separador`, cujo segundo elemento é o próprio separador, e cujo terceiro elemento é a fatia da string `nome` após a primeira ocorrência de `separador`.
- se `separador` não ocorre como parte de `nome`, então retorna-se uma tupla cujo primeiro elemento é a string `nome`, e cujos segundo e terceiro elementos são a string vazia.

Exemplo:

```
>>> x = 'litro de sorvete de creme'
>>> str.partition(x, 'de')
('litro ', 'de', ' sorvete de creme')
>>> str.partition(x, 'picolé')
('litro de sorvete de creme', '', '')
```

`str.replace(nome, velho, novo, n=len(nome))`

Parâmetros:

- `nome, velho, novo`: três strings
- `n`: um número inteiro não negativo.

Efeito: nenhum
Retorno: o resultado de substituir as n primeiras ocorrências da string `velho` pela string `novo` na string `nome`.
Exemplo:

```
>>> x = 'bolo'
>>> str.replace(x, 'o', 'e', 1)
'belo'
```

`str.split(nome, separador=qualquer sequência de espaços em branco)`

Parâmetros:

- `nome, separador`: duas strings

Efeito: nenhum
Retorno: uma lista contendo todas as strings que ocorram em `nome` ...

- antes da primeira ocorrência da string `separador`, ou
- entre duas ocorrências da string `separador`, ou
- após a última ocorrência da string `separador`.

Exemplo:

```
>>> x = 'litro de     sorvete de creme'
>>> str.split(x, 'de')
['litro ', '     sorvete ', ' creme']
>>> str.split(x)
['litro', 'de', 'sorvete', 'de', 'creme']
```

`str.strip(nome, remove=qualquer sequência de espaços em branco)`

Parâmetros:

- `nome, remove`: duas strings

Efeito: nenhum
Retorno: o resultado da remoção de todas as ocorrências da string `remove` do começo e do final da string `nome`.
Exemplo:

```
>>> x = 'ababhugoabab'
>>> str.strip(x, 'ab')
'hugo'
```

`str.upper(nome)`

Parâmetros:

- `nome`: uma string

Efeito: nenhum
Retorno: a string `nome` com todos os caracteres em letras maiúsculas.

Tuplas

`tuple.count(nome, elemento)`

Parâmetros:

- `nome`: uma tupla.
- `elemento`: um valor qualquer.

Efeito: nenhum
Retorno: o número de vezes que `elemento` aparece na tupla `nome`.

`tuple.index(nome, elemento, inicio=0, fim=len(nome))`

Parâmetros:

- `nome`: uma tupla.
- `elemento`: um elemento que apareça na tupla `nome[inicio:fim]`.
- `inicio, fim`: índices da tupla `nome`.

Efeito: nenhum
Retorno: o primeiro índice entre `inicio` e `fim`-1 (inclusive) no qual apareça `elemento` na tupla `nome`.
Exemplo:

```
>>> x = ('a', 'b', 'c', 'b', 'a')
>>> tuple.index(x,'b',2,4)
3
```

Módulo `math`

`math.cos(n)`

Parâmetros:

- `n`: um número.

Efeito: nenhum
Retorno: o cosseno do ângulo n (em radianos).

`math.sin(n)`

Parâmetros:

- `n`: um número.

Efeito: nenhum
Retorno: o seno do ângulo n (em radianos).

`math.sqrt(n)`

Parâmetros:

- n: um número.

Efeito: nenhum
Retorno: um float que aproxima a raiz quadrada do número n.

`math.pi`

Uma aproximação da constante matemática π.
Exemplo:

```
>>> import math
>>> math.pi
3.141592653589793
```

Módulo `random`

`random.randint(nome1,nome2)`

Parâmetros:

- `nome1`, `nome2`: dois números inteiros, sendo `nome1` menor ou igual a `nome2`.

Efeito: nenhum
Retorno: um número inteiro sorteado aleatoriamente no intervalo entre `nome1` (inclusive) e `nome2` (inclusive).

Conversão entre tipos

`int(nome)`

Parâmetro:

- `nome`: um valor que possa ser convertido para um número inteiro.

Efeito: nenhum
Retorno: o resultado da conversão.

`float(nome)`

Parâmetro:

- `nome`: um valor que possa ser convertido para um número de ponto flutuante.

Efeito: nenhum
Retorno: o resultado da conversão.

`complex(nome)`

Parâmetro:

- `nome`: um valor que possa ser convertido para um número complexo.

Efeito: nenhum
Retorno: o resultado da conversão.

`str(nome)`

Parâmetro:

- `nome`: um valor que possa ser convertido para uma string.

Efeito: nenhum
Retorno: o resultado da conversão.

`tuple(nome)`

Parâmetro:

- `nome`: uma sequência.

Efeito: nenhum
Retorno: a representação da sequência `nome` como uma tupla.

`list(nome)`

Parâmetro:

- `nome`: uma sequência.

Efeito: nenhum
Retorno: a representação da sequência `nome` como uma lista.

Dicionários

`dict.keys(nome)`

Parâmetros:

- `nome`: um dicionário.

Efeito: nenhum
Retorno: uma sequência contendo todas as chaves do dicionário `nome`.

`dict.values(nome)`

Parâmetros:

- `nome`: um dicionário.

Efeito: nenhum
Retorno: uma sequência contendo todos os valores do dicionário `nome`.

`dict.items(nome)`

Parâmetros:

- `nome`: um dicionário.

Efeito: nenhum
Retorno: uma sequência de pares `(chave,valor)`, contendo todas as chaves do dicionário `nome` e seus respectivos valores.

`dict.get(nome,chave,retorno=None)`

Parâmetros:

- `nome`: um dicionário.
- `chave`, `retorno`: dois valores quaisquer

Efeito: nenhum
Retorno: o conteúdo do dicionário `nome` correspondente à chave `chave`, se existir, ou o valor `retorno` em caso contrário.

`dict.clear(nome)`

Parâmetros:

- `nome`: um dicionário.

Efeito: apaga todas as chaves e valores do dicionário `nome`.
Retorno: nenhum

`dict.copy(nome)`

Parâmetros:

- `nome`: um dicionário.

Efeito: nenhum
Retorno: uma cópia do dicionário `nome`. Atenção! Para os valores contidos no dicionário `nome` que sejam mutáveis, a cópia do dicionário retornada conterá apenas `aliases` (pseudônimos), e não cópias de fato.

Interação com o usuário

`input(nome)`

Parâmetros:

- `nome`: uma string.

Efeito: imprime na tela a string `nome` e aguarda a entrada de dados do usuário.
Retorno: os dados digitados pelo usuário (em formato string).

`print(nome1, nome2, nome3, ..., sep = ' ', end = '\n')`

Parâmetros:

- `nome1, nome2, nome3,` ...: uma quantidade qualquer de parâmetros de quaisquer tipos.
- `sep`, `end`: strings

Efeito: imprime na tela os valores dos parâmetros `nome1`, `nome2`, etc., separados por `sep`, e imprime `end` ao final.
Retorno: nenhum