Lista de Exercícios no. 6 :: Listas

Instruções Gerais

- Os exercícios são de resolução individual.
- Sempre tente resolver os exercícios por conta própria, mesmo aqueles que já tenham sido feitos pelo professor em aula.
- Crie um arquivo (ex: lista.py) e faça cada exercício em uma ou mais funções.
- Utilize a extensão .py e o editor VS Code (ou outro de sua preferência).
- **Não é permitido o uso de recursos da linguagem ou bibliotecas que não foram abordados na disciplina até o momento da publicação desta lista.**
    - A relação de operadores e funções de listas que podem ser utilizadas está em anexo. Note que os exercícios podem adicionar restrições sobre o uso de recursos dessa relação.

1. Escreva uma função que recebe uma lista **vet** a imprime em ordem reversa. Não é permitido o uso de list.reverse().

   ```
   def print_reverse(vet: list)
   ```

2. Escreva uma função que recebe uma lista **vet** e imprime apenas os valores pares.

   ```
   def print_even(vet: list)
   ```

3. Escreva uma função que recebe uma lista **vet** contendo números inteiros. A função deve modificar a lista, invertendo o sinal dos números negativos, passando-os para positivo.

   ```
   def set_positive(vet: list)

   Ex: Entrada:[1, -5, 67, -45, -1, -1, 0, 48] → Saída:[1, 5, 67, 45, 1, 1, 0, 48]
   ```

4. Escreva uma função que recebe uma lista **vet** e devolve a média aritmética simples dos elementos.

   ```
   def list_sum(vet: list) -> float

   Ex: Entrada: [1, 23, 4, 8, 41, 7, 3] → Saída: 12
   ```

5. Escreva uma função que recebe uma lista **vet**, bem como, um elemento **elem** a ser procurado. A função deve retornar a posição (índice) do elemento ou **None** caso ele não esteja no vetor.

   ```
   def find(vet: list, elem: int)
   ```

6. Escreva uma função que recebe uma lista **vet** ordenado crescentemente, bem como, um elemento **elem** a ser procurado. A função deve retornar a posição (índice) do elemento ou **None** caso ele não esteja na lista. **OBS: Tente usar o fato da lista estar ordenada para criar uma solução melhor que a anterior**

   ```
   def find(vet: list, elem: int)
   ```

7. Escreva uma função que recebe uma lista **vet** e um número **value**. A função deve retornar uma outra lista, contendo os múltiplos de **value** que estão em **vet**.

   def get_multiples(vet: list, value: int) -> list

8. Escreva uma função que recebe uma lista **vet** e retorna uma outra lista, com os primos em vet.

   def get_primes(vet: list) -> list

9. Escreva uma função que verifica se os elementos de uma lista estão em ordem crescente. A função deve retornar True, caso os elementos estejam dispostos em ordem crescente, ou False, em caso contrário. Obs: Não é permitido o uso de list.sort().

   def is_sorted(vet: list) -> bool

   Exemplo de uso da função:

   ```
   v = [1,4,7,9,15,22,48,512]
   print(is_sorted(v)) # True
   ```

10. Escreva uma função que recebe uma lista **vet** e devolve o segundo maior elemento. Dica: list.sort()

    def find_max2(vet: list) -> int

11. Escreva uma função que recebe uma lista **vet** e devolve o maior e o menor elementos. Obs: Não é permitido o uso de mint() e max()

    def find_min_max(vet: list) -> tuple[int, int]

12. Escreva uma função que recebe uma lista **vet** e devolve o maior e o menor elementos. Obs: Não é permitido o uso de mint(), max() e list.sort()

    def find_min_max(vet: list) -> tuple[int, int]

13. Escreva uma função que recebe uma lista **vet** e devolve seus 3 maiores elementos. Obs: Não é permitido o uso de max() e list.sort()

    def get_max3(vet: list) -> tuple[int, int, int]

14. Escreva uma função que recebe uma lista **vet** e um elemento **elem**. A função deve remover todas as ocorrências de **elem** de **vet**. Dica: list.remove().

    def remove_all(vet: list, elem: int)

15. Escreva uma função que recebe uma lista **vet** e um elemento **elem**. A função deve remover todas as ocorrências de **elem** de **vet**. Obs: Não é permitido o uso de list.index() e list.remove()

    def remove_all(vet: list, elem: int)

16. Escreva uma função que recebe uma lista **vet** e inverte os seus elementos na própria lista. Obs: Não é permitido o uso de list.reverse().

    def reverse_list(vet: list)

17. Escreva uma função que recebe uma lista **vet** preenchido com inteiros positivos. A função deve imprimir as ocorrências (contagem) de cada número na lista. Dica: utilize uma lista **count** para armazenar a contagem de cada elemento na lista **vet**, relacionando as posições de **count** aos valores em **vet**. A lista count precisa ter tamanho max(vet)+1.

    def count_elements(vet: list)

18. Escreva uma função que recebe um valor inteiro **value** e uma lista **notes** com as possíveis cédulas. A função deve imprimir a quantidade mínima de cédulas equivalente ao valor.
    Dica: use uma lista auxiliar **count** para armazenar a contagem de cada cédula.

    def min_bills(value: int, notes: list)

    ```
    Exemplo:    min_bills(209, [1,5,10,50,100])   # lista de tipos de cédulas
                # Saída:
                # 2 x $100
                # 1 x $5
                # 4 x $1
    ```

19. Escreva uma função que recebe uma lista **days** de tamanho 7 que exibe um histograma da variação da temperatura durante a semana. A lista **days** armazena as temperaturas para cada dia da semana, iniciando por **domingo**. Por exemplo, se as temperaturas em **days** forem [19, 21, 25, 22, 20, 17 e 15] graus celsius, a função deverá exibir:

    ```
    D: >>>>>>>>>>>>>>>>>>
    S: >>>>>>>>>>>>>>>>>>>>
    T: >>>>>>>>>>>>>>>>>>>>>>>>
    Q: >>>>>>>>>>>>>>>>>>>>
    Q: >>>>>>>>>>>>>>>>>>>
    S: >>>>>>>>>>>>>>>>>
    S: >>>>>>>>>>>>>>>>
    ```

    def histogram(days: list)

20. Chamando a função desenvolvida no exercício anterior, escreva uma função que recebe a lista **days** e exibe:
    a) As temperaturas média, mínima, máxima;
    b) O histograma.

    def temperature_report(days: list)

21. Escreva uma função que recebe duas listas e concatena o conteúdo de ambas em uma nova lista, a ser retornada. Obs: Não é permitido o uso de list.extend() e do operador +.
    def list_concat(v1: list, v2: list) -> list

    ```
    Exemplo:
    v3 = list_concat([1,2,3,4], [5,6,7])      # v3 = [1,2,3,4,5,6,7]
    ```

22. Escreva uma função que recebe duas listas e realiza a união entre o conteúdo de ambas, colocando o resultado em uma nova lista, a ser retornada.
    def list_union(v1: list, v2: list) -> list

    ```
    Exemplo:
    v3 = list_union([1,2,3,4,5], [8,2,4,9])        # v3 = [1,2,3,4,5,8,9]
    ```

23. Escreva uma função que recebe duas listas e realiza a interseção entre o conteúdo de ambas, colocando o resultado em uma nova lista, a ser retornada.
    def list_intersection(v1: list, v2: list) -> list

    ```
    Exemplo:
    v3 = list_intersection([1,2,3,4,5], [8,2,4,9])  # v3 = [2,4]
    ```

24. Escreva uma função para imprimir o número de acertos de cada aluno em uma prova. A prova possui 20 questões e cada questão tem cinco alternativas (A, B, C, D, E). Para isso são dados:
    a. lista **check**, de 20 elementos, com o gabarito;
    b. lista **answers,** com as respostas dos alunos, contendo as 20 respostas de cada aluno, em sequência.
        i. A lista contém as respostas de todos os alunos (20 para cada aluno). Dessa forma, o tamanho da lista é múltiplo de 20. Se forem 10 alunos, **answers** terá tamanho 200.

    def check_tests(check: list, answers: list)

25. Escreva uma função que recebe duas listas de inteiros (**bin1** e **bin2**) de mesmo tamanho, que devem ser interpretadas como dois números binários de n algarismos. A função deve calcular a sequência de números que representa a soma dos dois binários, colocando-a em uma nova lista a ser retornada.
    def binary_sum(bin1: list, bin2: list)

```
carry        1    1    1    1    1    1    0    0    0
bin1:             1    1    0    0    1    1    0    1
bin2:        +    0    1    1    1    0    1    1    0
            -----------------------------------------
return:      1    0    1    0    0    0    0    1    1
```

26. Escreva uma função que recebe duas listas **v1** e **v2,** ordenadas crescentemente. A função deve mesclar ordenadamente os conteúdos de **v1** e **v2**, colocando-os em uma nova lista, a ser retornada. OBS: não é permitido o uso de list.sort() e do operador +.
    def list_merge(v1: list, v2: list)

```
Entrada:    v1 = [1, 3, 4, 7, 9, 10]
            v2 = [2, 3, 5, 7, 7, 14]
Saída:      v3 = [1, 2, 3, 3, 4, 5, 7, 7, 7, 9, 10, 14]
```

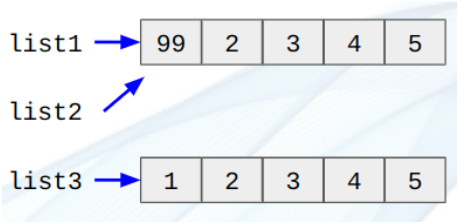
27. Escreva uma função que recebe uma lista contendo números inteiros. A função deve determinar o segmento de soma máxima, iniciando na posição 0. Ao final, deve imprimir a soma máxima.
    def max_sum_slice(v: list)

```
Exemplo: Na lista [5, 2,-2,-7, 3,14,10,-3, 9,-6, 4, 1], a soma máxima é 31.
```

GUIA RÁPIDO DE CONSULTA: LISTAS EM PYTHON

```
# Criando e acessando elementos
-----------------------------------
# definição de lista
numbers = [2,4,6,8,10]
tens = []        # lista vazia
n = len(numbers) # comprimento

x = numbers[0]   # leitura do 1o elem
numbers[2] = 55  # escrita no 3o elem

# Verificando elementos e posições
-----------------------------------
idx = 15
try:
      print(numbers[idx])
except IndexError:
      print(f'Não há índice {idx}')

if 10 in numbers:
      print('10 esta na lista.')
else:
      print('10 não está na lista.')

elem = 20
try:
      #list.index(elem, pos_inicial=0)
      idx = numbers.index(elem)
except ValueError:
      print(f'Não há elemento {elem}')

# Varredura de lista em laço
-----------------------------------
# varredura da lista por elementos
for elem in numbers:
      print(elem, end=', ')
print('\b\b ')

# varredura da lista por índices
for i in range(len(numbers)):
      print(numbers[i], end=', ')
print('\b\b ')
```

```
# Tipo Referência
-----------------------------------
list1 = [1,2,3,4,5]
list2 = list1
list3 = list1.copy()
list1[0] = 99
```




```
# Utilitários: Funções e Operadores
-----------------------------------
list1 = [10,11,12]   #[10,11,12]
list1.append(4)      #[10,11,12,4]
list1.insert(0, 99)  #[99,10,11,12,4]

list1.pop()           #[99,10,11,12]
list1.pop(0)#pos      #[10,11,12]
list1.remove(11)#elem #[10,12]

list1.extend([55,-66])#[10,12,55,-66]

print(max(numbers))  # 55
print(min(numbers))  # -66

numbers = [3,1,2]
print(numbers == [3,1,2]) # True
numbers += [9,7,5,1] # [3,1,2,9,7,5,1]
numbers.reverse()    # [1,5,7,9,2,1,3]
numbers.sort()       # [1,1,2,3,5,7,9]
```