



Laboratório de Programação Competitiva

Profa. Silvia Brandão

2024.2

Aula de hoje ...

- Revisão e Aprofundamento em Python.
- Introdução às estruturas de dados em Python.
- Estruturas de dados em Python: listas, tuplas, dicionários e conjuntos.
- Implementação prática de algoritmos simples em Python.

Plano de Ensino

SISTEMA DE AVALIAÇÃO

- Momento N1: 25 pontos + (Uniube+)
 - **1ª PROVA: T17 e T11** - 25/09, **T12** - 26/09, 10pts;
 - Avaliação contínua/Trabalhos/Beecrowd: 15pts;
- Momento N2: 25 pontos + (Uniube+)
 - **2ª PROVA: T17 e T11** - 06/11, **T12** - 07/11, 10pts;
 - Avaliação contínua/Trabalhos/Beecrowd: 15pts;
 - 15ª Maratona de Programação: 23/11 - **pontuação extra**
- Momento N3: 25 pontos + (Uniube+ e Avaliação Institucional)
 - PROJETO PRÁTICO: **T17 e T11** - 04/12, **T12** - 05/12, 20pts;
 - Avaliação contínua/Trabalhos/Beecrowd: 5pts;
- **2ª OPORTUNIDADE dos Momentos N1 e N2:**
 - PROVA: **T17 e T11** - 11/12, **T12** - 12/12, CADA UMA 10pts; (CONTEÚDO DE TODO O SEMESTRE)
- **RECUPERAÇÃO DE NOTAS para atingir 60pts:**
 - VALOR: 20pts de prova (substituirá 1ª e 2ª avaliações); porém o aluno não poderá ter sua nota do Uniube+ zerada ou ter menos de 40pts no total do semestre.

Notas de trabalhos e projetos não são recuperáveis.

Tirando dúvidas

Assuntos bastante interessantes:

- o conceito de pilha
- a recursividade de funções.



```
def fatorial(n):  
    if n == 0 or n == 1:  
        return 1  
    else:  
        return n * fatorial(n - 1)  
  
for valor in [0,1,2,3,4,5,6]:  
    print(fatorial(valor))
```



```
1  
1  
2  
6  
24  
120  
720
```



```
def fatorial(n):  
    if n == 0 or n == 1:  
        return 1  
    else:  
        return n * fatorial(n - 1)  
  
for valor in [0,1,2,3,4,5,6]:  
    print(fatorial(valor))
```



```
1  
1  
2  
6  
24  
120  
720
```

Tabela 4 – Demonstração de uso da pilha com o armazenamento da chamada de função recursiva.

CÓDIGO	PILHA	COMENTÁRIO
fatorial(3)	fatorial(3)	A chamada da função fatorial com parâmetro 3 é colocada na pilha
return 3 * fatorial(2)	fatorial(2) fatorial(3)	O código de fatorial(3) está na pilha, porém, deve esperar a resposta de fatorial(2)
return 2 * fatorial(1)	fatorial(1) fatorial(2) fatorial(3)	Agora, estão na pilha: fatorial(3) e fatorial(2). Só que fatorial(2) exige a solução de fatorial(1) que também é colocada na pilha
return 1	fatorial(1) #Retorna 1 fatorial(2) fatorial(3)	No caso de fatorial 1, n é 1, então ele retornará 1. Assim, fatorial(1) deve sair da pilha.
return 2 * 1	fatorial(2) #Retorna 2 fatorial(3)	Como fatorial(1) foi resolvido, agora fatorial(2) pode ser resolvida também, utilizando o retorno de fatorial(1). Nesse caso, a resposta de fatorial(2) é 2.
return 3 * 2	fatorial(3) #Retorna 6	Por fim, fatorial(3) que aguardava na pilha, recebe a resposta de fatorial(2) e consegue calcular a resposta final, que no caso é 6.

Tirando dúvidas

```
import math
def distance(x1, y1, x2, y2):
    return math.sqrt( (x2-x1)**2 + (y2-y1)**2 )

distance(1, 2, 4, 6)
```

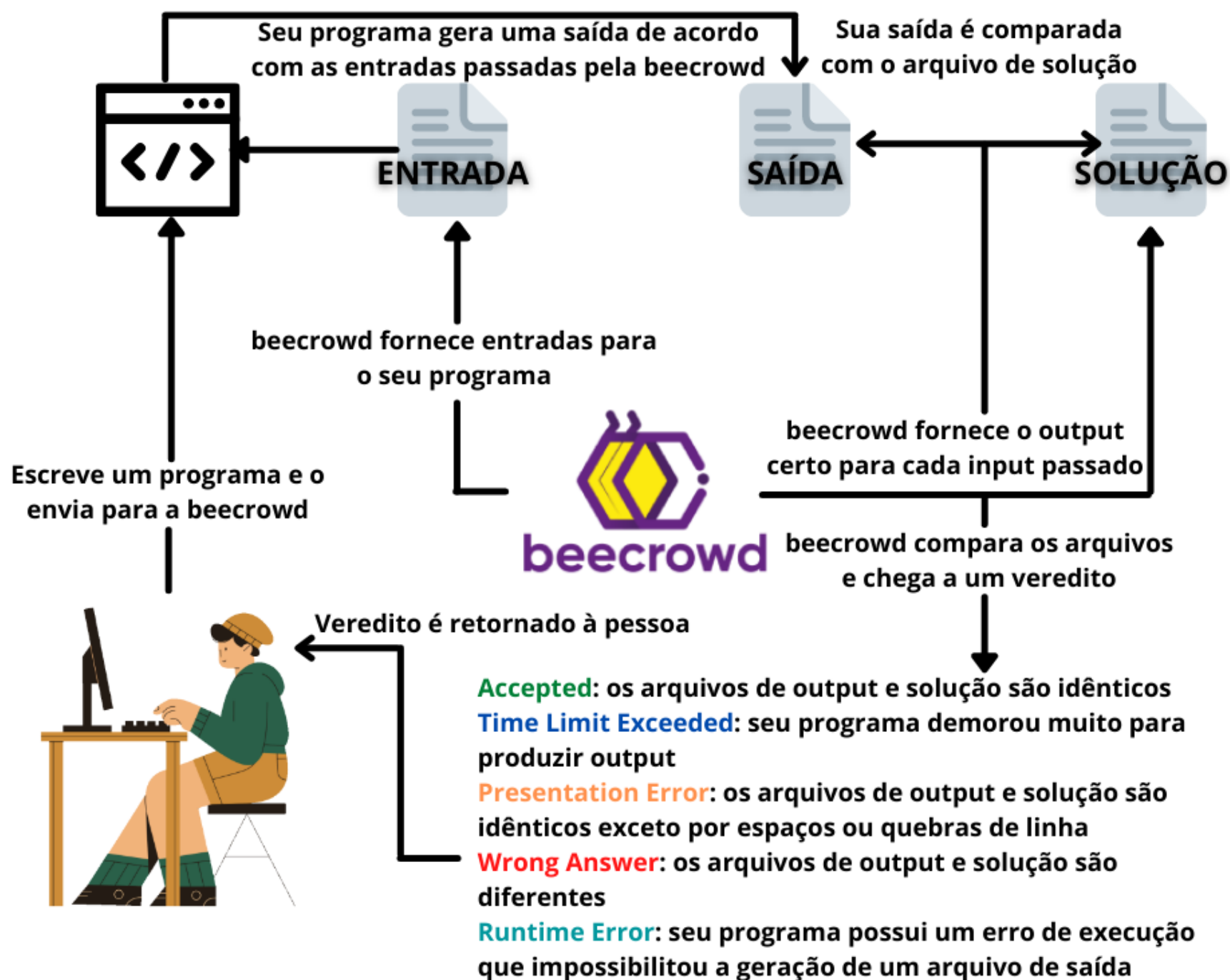
```
n1,n2 = map(float, input("Digite dois números ").split())
m = n1 * n2
print("Multiplicação = ", m)
```

```
lista1 = [x ** 2 for x in range(10)]
print(lista1)

lista2 = []
lista2 = [x for x in range(1,20) if x % 2 == 0]
print(lista2)

lista3 = [i for i in "HACKATHON" if i in ["A","E","I","O","U"]]
print(lista3)

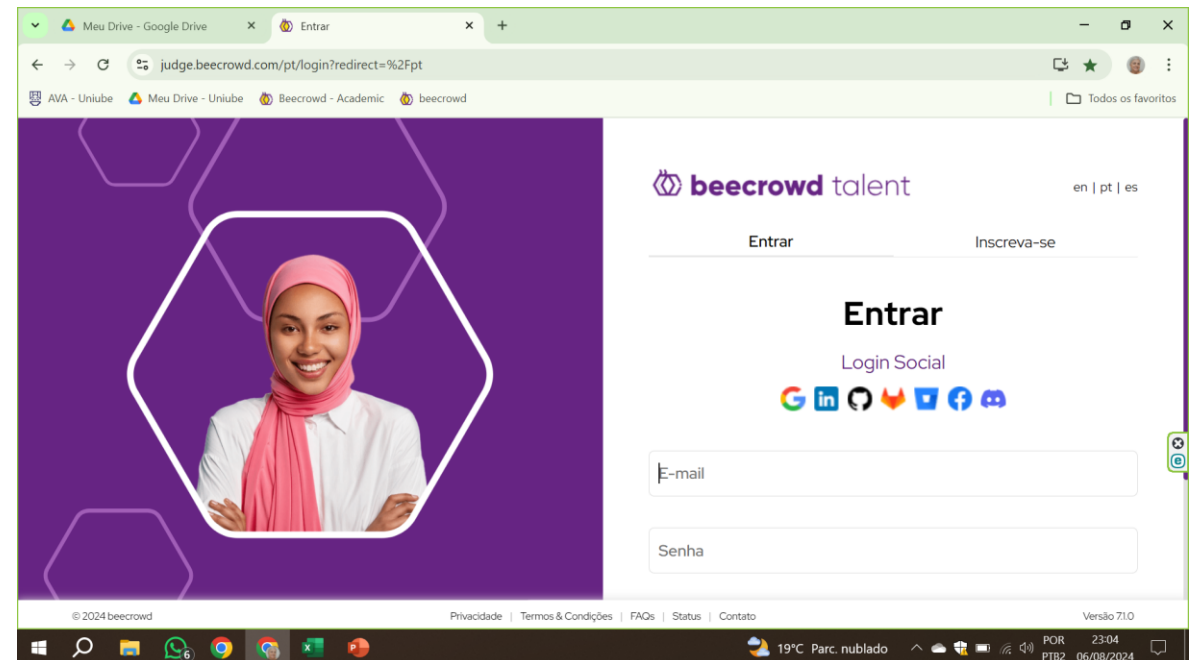
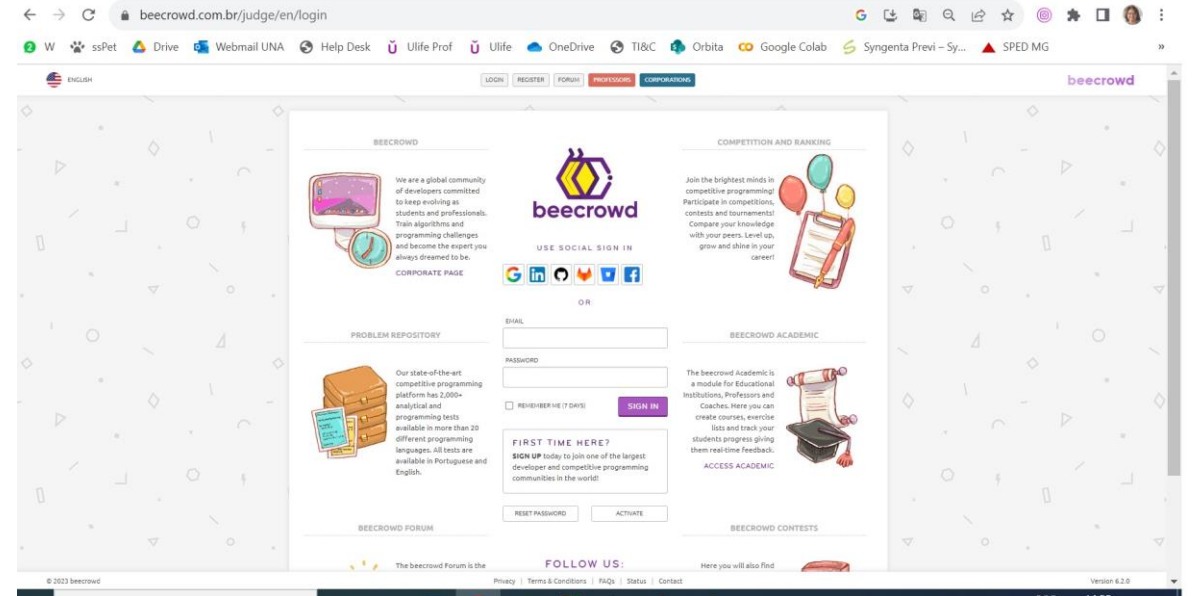
lista4 = [1,2,3]
lista4 = [i**3 for i in lista4]
```



Desafio

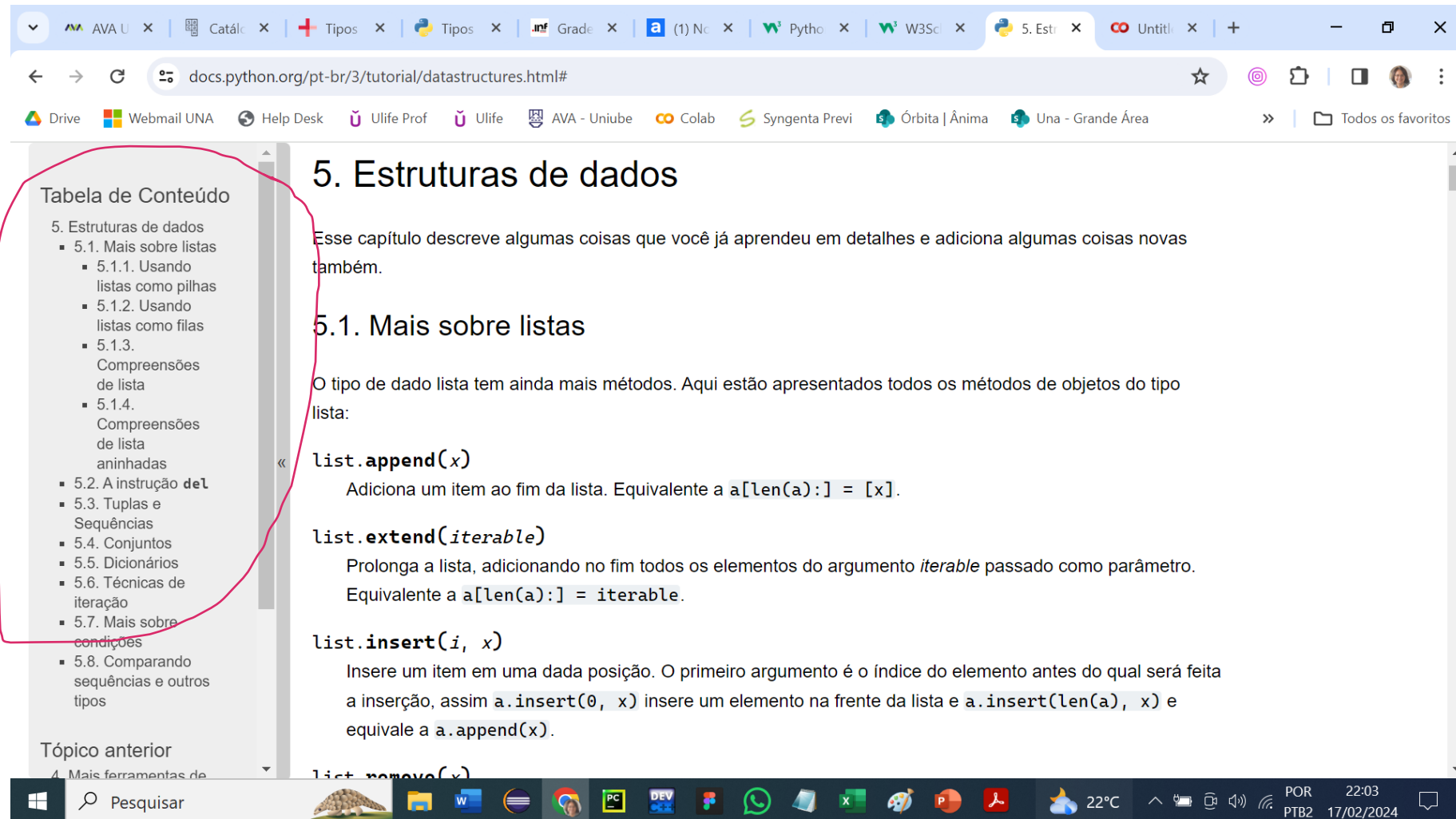
1. Inscreva-se na plataforma Beecrowd:
<https://judge.beecrowd.com/en/login>

2. Resolva os exercícios de números:
- 1073, 1095



Estruturas de dados em Python

- Acesse o link e estude:
<https://docs.python.org/pt-br/3/tutorial/datastructures.html>



The screenshot shows a web browser displaying the Python documentation page for data structures. The browser's address bar shows the URL `docs.python.org/pt-br/3/tutorial/datastructures.html#`. The page title is "5. Estruturas de dados". The left sidebar contains a "Tabela de Conteúdo" (Table of Contents) with a red circle highlighting the section "5. Estruturas de dados" and its sub-items: "5.1. Mais sobre listas", "5.2. A instrução `del`", "5.3. Tuplas e Sequências", "5.4. Conjuntos", "5.5. Dicionários", "5.6. Técnicas de iteração", "5.7. Mais sobre condições", and "5.8. Comparando sequências e outros tipos". The main content area shows the introduction to the chapter, stating that it describes things you've already learned in detail and adds some new things. It then introduces the "5.1. Mais sobre listas" section, explaining that the list type has more methods and that all methods of list objects are presented. The first method shown is `list.append(x)`, which adds an item to the end of the list, equivalent to `a[len(a):] = [x]`. The second method is `list.extend(iterable)`, which prolongs the list by adding all elements of the `iterable` argument at the end, equivalent to `a[len(a):] = iterable`. The third method is `list.insert(i, x)`, which inserts an item at a given position, with the first argument being the index of the element before which the insertion is made. The fourth method is `list.remove(x)`.

Estruturas de dados em Python

Listas e Tuplas

- As **listas são mutáveis**, o que permite adicionar, remover e modificar elementos. Assim, podemos, por exemplo, alterar o valor de um elemento específico em uma lista, adicionar um novo elemento ao final ou remover um elemento existente.
- Já **as tuplas são imutáveis** e não permitem tais operações. Veja o exemplo:

```
# Lista (mutável)
lista = [1, 2, 3]
lista[0] = 4
lista.append(5)
lista.remove(2)
print(lista) # Output: [4, 3, 5]

# Tupla (imutável)
tupla = (1, 2, 3)
tupla[0] = 4 # Erro: As tuplas são imutáveis e não podem ser modificadas
```

[4, 3, 5]

```
TypeError                                Traceback (most recent call last)
<ipython-input-7-638788d43a93> in <cell line: 10>()
      8 # Tupla (imutável)
      9 tupla = (1, 2, 3)
--> 10 tupla[0] = 4 # Erro: As tuplas são imutáveis e não podem ser modificadas

TypeError: 'tuple' object does not support item assignment
```

Desafio: Gerenciamento de Biblioteca Virtual

Atividade prática: Para o desenvolvimento da biblioteca virtual, vamos optar por utilizar a linguagem de programação Python , com os recursos de listas e tuplas. Por exemplo:

```
livros = ["Dom Casmurro", "O Pequeno Príncipe", "1984", "O Senhor dos Anéis"]
```

```
autores = [("Machado de Assis", 1839), ("Antoine de Saint-Exupéry", 1900), ("George Orwell", 1903), ("J.R.R. Tolkien", 1892)]
```

Assim, é possível adicionar novos livros ao acervo, inserindo títulos na lista livros e informações sobre os autores na tupla autores:

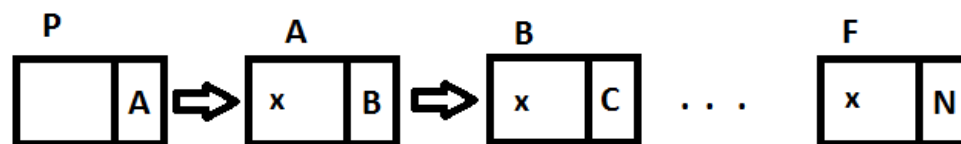
```
livros.append("Harry Potter")  
autores.append(("J.K. Rowling", 1965))
```

- **Crie** uma função de cadastro de livros (no mínimo 5 títulos) em uma lista e armazene os dados sobre os autores desses livros em uma tupla. Em seguida, **crie** uma função de busca que percorra a lista de livros e retorne as informações do livro correspondente, caso este exista.

```
# Exemplo de utilização da função  
titulo_busca = "1984"  
informacoes_livro = buscar_livro(titulo_busca)  
print(f"As informações do livro '{titulo_busca}' são: {informacoes_livro}")
```

Leituras para a próxima aula:

- As três principais estruturas de dados utilizadas em ordenação e alocação de dados são as estruturas de dados dinâmicas: lista, fila e pilha. Essas estruturas são fundamentais para o desenvolvimento de aplicações; assim como as árvores.
- <https://docs.python.org/pt-br/3/tutorial/datastructures.html>



P: primeiro elemento (nó) da lista
x: dado armazenado
F: último elemento (nó) da lista
N: NULL

Tarefa para a próxima aula:

- Qual a forma de acesso aos dados de uma fila?
- Qual a forma de acesso aos dados de uma pilha?
- Vamos explorar os conceitos de listas, tuplas, dicionários e conjuntos?