



UNIVERSIDADE DE  
**VASSOURAS**



**UNIVERSIDADE DE VASSOURAS**  
Curso de Graduação em Engenharia de Software

---

**Aula 4**  
**11 MAR 2022**

## **Laboratório de Programação de Interfaces com o Usuário**

**Prof. André Saraiva**

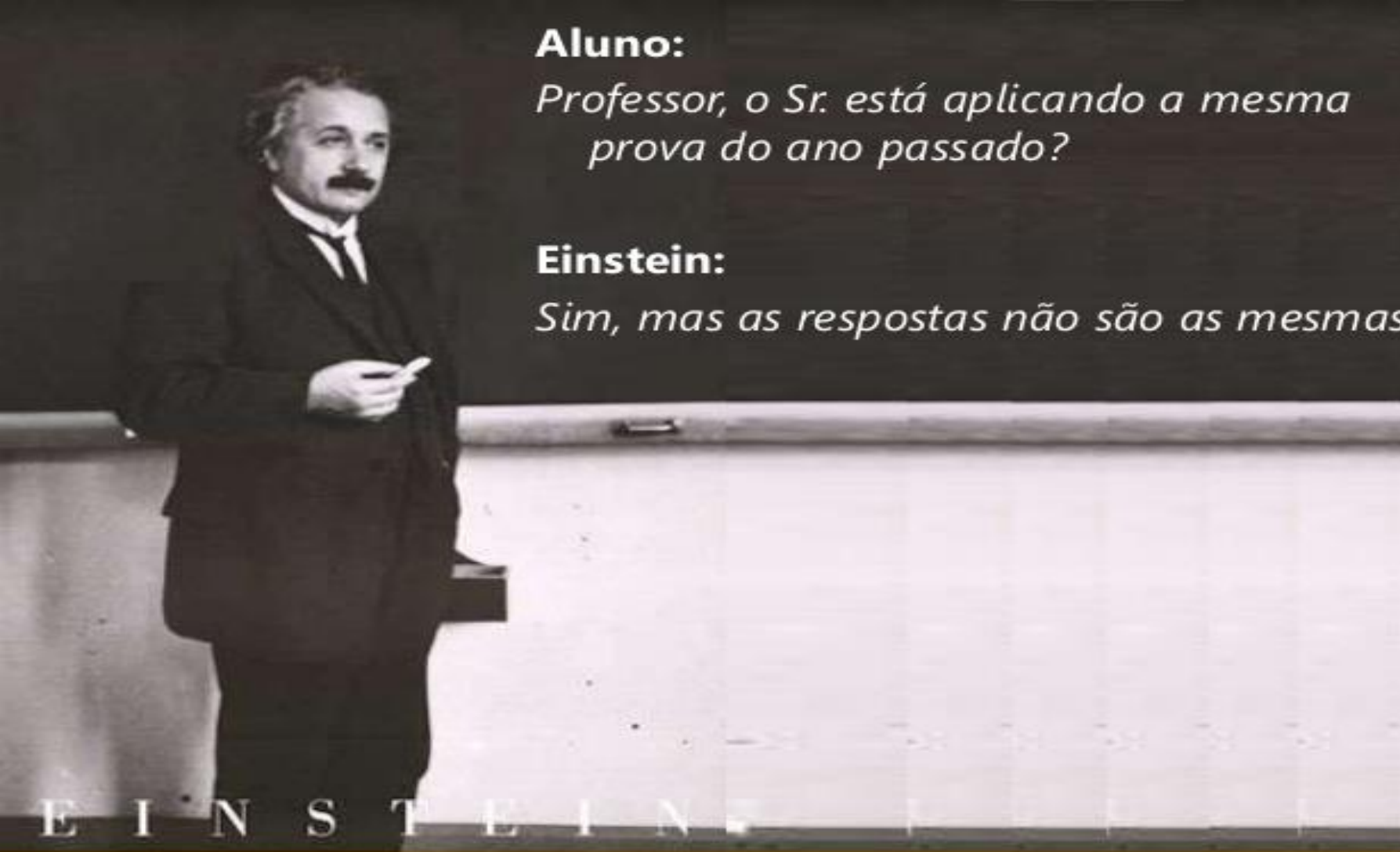
Mestre em Sistemas de Computação  
Especialista em Arquitetura e Projeto de Cloud Computing  
Analista Blue Team em Cibersegurança pela Kimoshiro  
Tutor EaD pela Universidade Federal Fluminense - UFF



Bibliografia

BARTIÉ, Alexandre. **Garantia da qualidade de software**. Gulf Professional Publishing, 2002.

Capítulo 12



**Aluno:**

*Professor, o Sr. está aplicando a mesma prova do ano passado?*

**Einstein:**

*Sim, mas as respostas não são as mesmas!*

E I N S T E I N

## Tópicos

### Revisão - Python

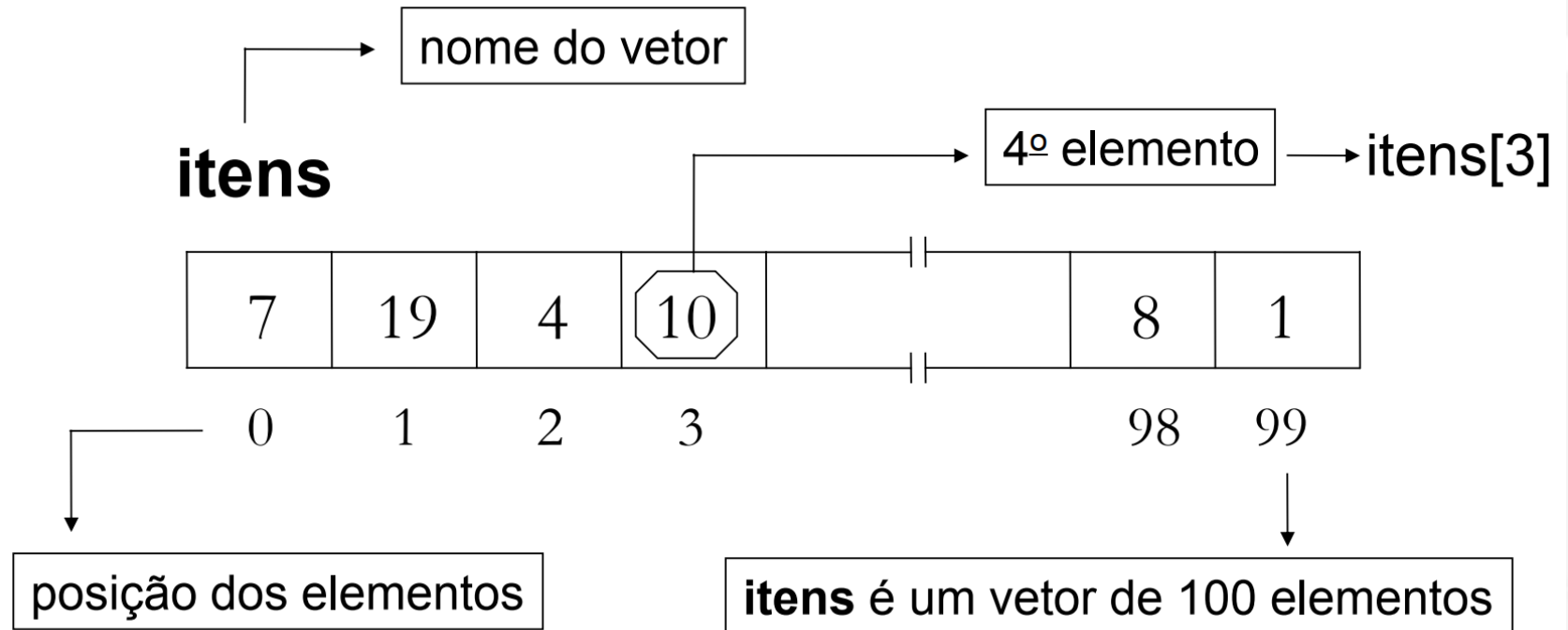
- Vetores;
  - Matrizes;
  - Strings.
- 
- Atividade Avaliativa para P1

## Vetor

Um vetor (array) é um agregado de elementos (valores) de um mesmo tipo.

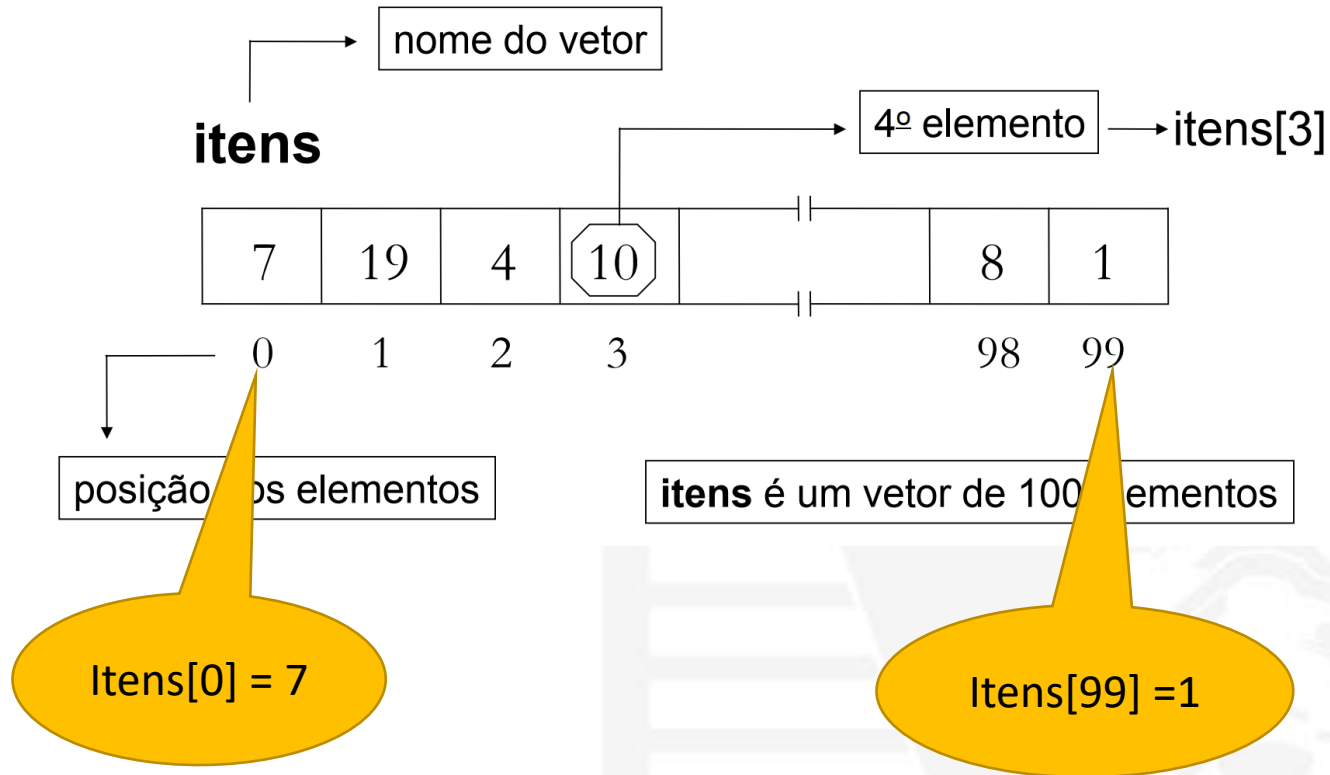
Trata-se de uma estrutura homogênea, isto é, formada por elementos de um mesmo tipo, chamado tipo base;

Todos os elementos da estrutura são igualmente acessíveis, ou seja, o tipo de procedimento para acessar qualquer elemento é igual;

**Vetor**

Cada elemento da estrutura tem um nome próprio, composto pelo nome do vetor e pelo índice.

## Vetor





## Metodologia para a Solução:

1. Caso não exista, crie um projeto do qual seu programa fará parte;

2.

### # Solução 1

```
soma = 0
```

```
for indice in range(0,100):
```

```
    soma = soma + itens[indice]
```

### # Solução 2

```
soma = 0
```

```
for item in itens:
```

```
    soma = soma + item
```

## Vetor

- ✓ A função **len**(*nome do vetor*) que retorna o tamanho de um vetor.

### # Solução 3

```
soma = 0
```

```
for indice in range(len(itens)):
```

```
    soma = soma + itens[indice]
```

- ✓ Todas as informações podem ser lidas de uma vez, como uma linha de caracteres, sobre a qual se aplica a operação **split** para separá-las.

## Atividade Prática

### ✓ Especificação do Problema:

Faça um programa para ler, do teclado, dez números **reais**. Após ***a inserção dos números, deverá ser escrito***, na saída padrão (vídeo), a listagem de entrada e em ordem crescente.

## Atividade Prática

### ✓ Metodologia para a Solução:

1. Caso não exista, crie um projeto do qual seu programa fará parte;
2. Dentro do projeto, crie um nome para o programa: atividade\_vetor.py;
3. Identifique a “necessidade” de constantes, declare as variáveis necessárias, identifique “macro” operações a serem realizadas como: ler, ordenar e escrever;

Um vetor (array) de vetores.

A declaração a seguir corresponde a uma matriz  
(de duas dimensões)

```
celulas = [[True, False, False, True, True], [True, True,  
False, True, False]]
```

## Matriz

```
celulas = [[True, False, False, True, True],  
            [True, True, False, True, False]]
```

		colunas				
		0	1	2	3	4
linhas	0	True	False	False	True	True
	1	True	True	False	True	False

O primeiro índice representa a linha e o segundo a coluna

**Matriz**

**celulas**

**colunas**

**linhas**

	0	1	2	3	4
0	True	False	False	True	True
1	True	True	False	True	False

**celulas[1][2]**

O primeiro índice representa a linha e o segundo a coluna

## Atividade Prática

### ✓ Especificação do Problema:

Imprima na saída padrão (vídeo) os resultados de uma turma de cinco alunos com três provas cada. Sua aplicação deverá imprimir a matriz de entrada “alunos” e a matriz de entrada ‘resultados’. Em seguida deverá imprimir se o aluno(a) foi aprovado ou não.

**alunos = [“Maria”, “Lucas”, “Ana”, “Juca”, “Carlos”]**

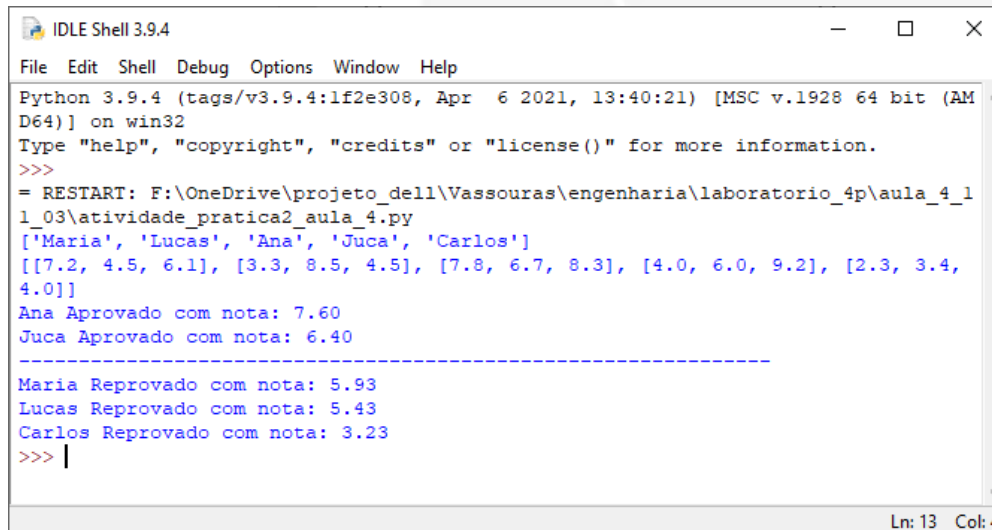
**resultados = [ [7.2, 4.5, 6.1], [3.3, 8.5, 4.5], [7.8, 6.7, 8.3], [4.0, 6.0, 9.2],  
[2.3, 3.4, 4.0] ]**



# Atividade Prática

## ✓ Metodologia para a Solução:

1. Caso não exista, crie um projeto do qual seu programa fará parte;
2. Dentro do projeto, crie um nome para o programa: atividade\_matriz.py;
3. Sua saída deve ser conforme a saída abaixo:



```
IDLE Shell 3.9.4
File Edit Shell Debug Options Window Help
Python 3.9.4 (tags/v3.9.4:1f2e308, Apr 6 2021, 13:40:21) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: F:\OneDrive\projeto_dell\Vassouras\engenharia\laboratorio_4p\aula_4_1_03\atividade_pratica2_aula_4.py
['Maria', 'Lucas', 'Ana', 'Juca', 'Carlos']
[[7.2, 4.5, 6.1], [3.3, 8.5, 4.5], [7.8, 6.7, 8.3], [4.0, 6.0, 9.2], [2.3, 3.4, 4.0]]
Ana Aprovado com nota: 7.60
Juca Aprovado com nota: 6.40
-----
Maria Reprovado com nota: 5.93
Lucas Reprovado com nota: 5.43
Carlos Reprovado com nota: 3.23
>>> |
```

Ln: 13 Col: 4

## String

Um objeto do tipo **str** representa uma cadeia de caracteres, de tamanho e valor imutáveis.

**nome = "Ana"**

nome	A	n	a
	0	1	2

## String

Um objeto do tipo **str** representa uma cadeia de caracteres, de tamanho e valor imutáveis.

**nome = "Ana"**

nome[0] = 'A'

nome	A	n	a
	0	1	2

## String

Um objeto do tipo **str** representa uma cadeia de caracteres, de tamanho e valor imutáveis.

**nome = "Ana"**

nome[1] = 'n'

nome	A	n	a
	0	1	2

## String

Um objeto do tipo **str** representa uma cadeia de caracteres, de tamanho e valor imutáveis.

**nome = "Ana"**

nome

A	n	a
---	---	---

0

1

2

nome[2] = 'a'

## String

**# programa comparando lexicograficamente Strings**

x = "ABCD"

y = "ABCZ"

z = "ABCDEFGG"

**# As seguintes comparações retornam True**

print(x < y)

print(x != "DCBA")

print(x < z)

print(y > z)

## String

O operador **+**, quando aplicado a dois operandos Strings *x* e *y* retorna uma String que é a concatenação das Strings *x* e *y*.

- Considere as declarações e atribuições:

```
nome = "Andre "  
sobrenome = "Saraiva"  
nomeCompleto = nome+sobrenome
```

- Temos então que:  
nomeCompleto = "Andre Saraiva"

## String

Retornando uma “nova” sub-String de uma String, via fatiamento:

`nomeString[posição inicial : posição final + 1]`

- Considere as declarações o trecho de programa:  
    `nomeCompleto = “Andre Saraiva”`  
    `a = nomeCompleto[6:12]`
  - Temos então que:  
    `a = “Saraiva”`



## String

Método **find**(*subStringProcurada*) → Retorna a posição do índice da primeira ocorrência da *subStringProcurada* na String sendo consultada. Caso não encontre, retorna menos um (-1).

Considere as declarações e atribuições abaixo:

```
nome = "Andre Saraiva"  
i = nome.find("Saraiva")
```

Temos então que:  
i = 6

## String

### 1. **replace**(*subStringProcurada*, *subStringNova*)

Retorna uma cópia da String sendo consultada, substituindo todas as ocorrências da *subStringProcurada* pela *subStringNova*.

### 2. **count**(*subStringProcurada*)

Retorna a quantidade de ocorrências da *subStringProcurada* na String.

### 3. **upper**()

Retorna uma cópia da String, convertendo os eventuais caracteres alfabéticos minúsculos para caracteres maiúsculos correspondentes.

## String

### 4. **lower()**

Idem ao anterior, convertendo os maiúsculos para minúsculos.

### 5. **strip()**

Retorna uma cópia da String, removendo todos os eventuais caracteres brancos do início e do final.

### 6. **split()**

Retorna uma lista de todas as palavras da String.

### 7. **split(subStringSeparadora)**

Retorna uma lista de todas as palavras da String, sendo o delimitador procurado entre palavras aquele especificado em *subStringSeparadora*.

## Tupla

Uma tupla é uma sequência ordenada de zero ou mais referências a objetos.

Suportam o mesmo fatiamento, o mesmo acesso por iteradores e o mesmo desempacotamento que Vetores e Strings.

Assim como Strings, tuplas são imutáveis e a tupla pode ser vazia.

```
valores = ("aula", 19.00, 22.15, 3)
```

## Tupla

### 1. Concatenação: $a + b$

Operador infixado que gera uma nova tupla a partir do conteúdo da primeira (a), seguido do conteúdo da segunda (b).

### 2. Replicação: $a * n$

Operador que gera uma nova tupla a partir do conteúdo da primeira (a), seguido pela repetição do mesmo conteúdo  $n - 1$  vezes.

### 3. Fatiamento: $a[\textit{posição inicial} : \textit{posição final} + 1]$

Operador que gera uma nova tupla a partir de um subconjunto de elementos contidos na tupla original (a).

**4. Operadores de atribuição incremental:**  $a += b$  ou  $a *= n$

Equivale aos operadores de concatenação e replicação, porém é atribuído à variável (a) a referência para a nova tupla gerada.

**5. Operadores de comparação:**  $<$ ,  $<=$ ,  $==$ ,  $!=$ ,  $>$  ou  $>=$

Comparação item a item e, recursivamente, para itens aninhados.

**6. Teste de associação:**  $in$  e  $not\ in$

Verifica a pertinência de um valor em uma tupla.

## Próximos Tópicos

### Revisão de Python

- Lista
- Conjunto (Set)
- Dicionário (dict)

## Atividade Avaliativa

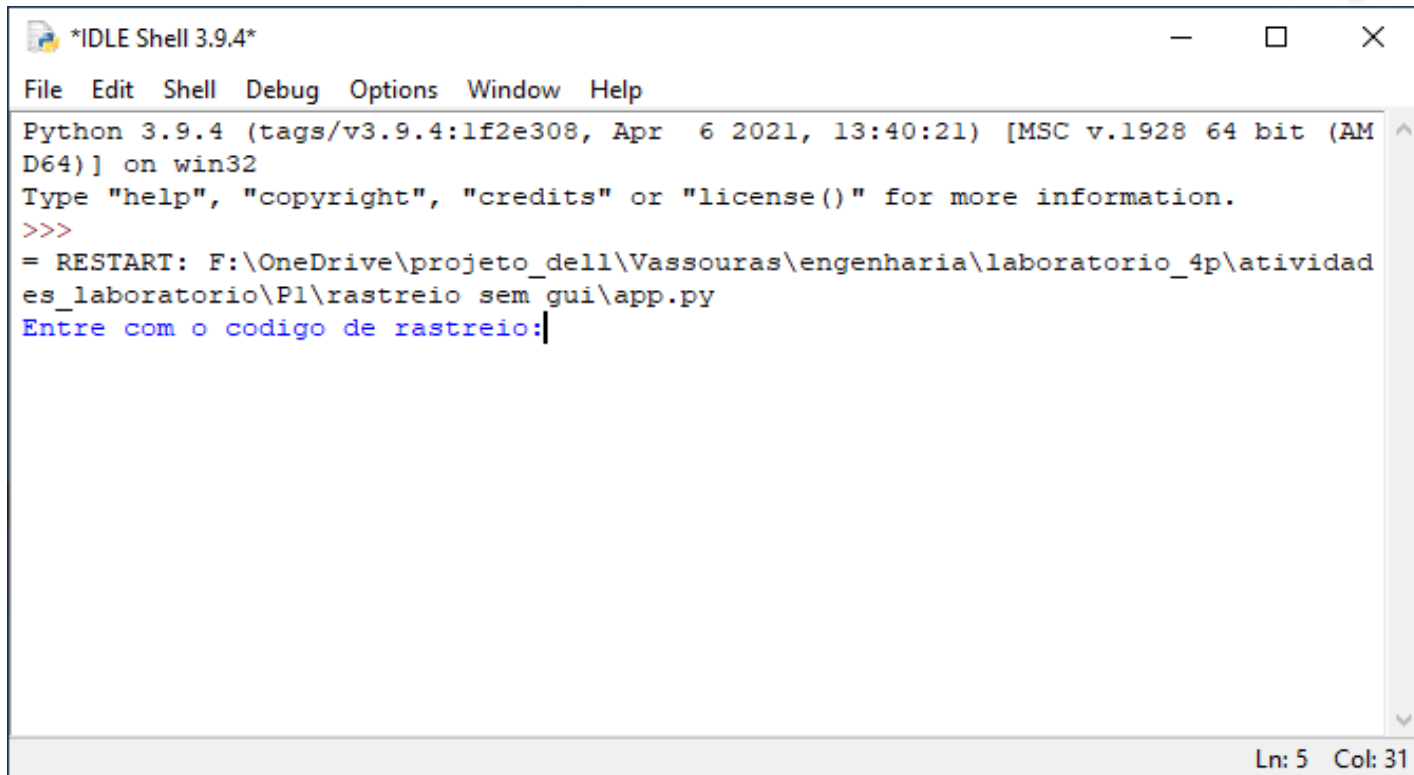
### Objetivos

Neste projeto será implementado em python um rastreamento de pacotes da ECT (Empresa Brasileira de Correios e Telegrafo). O rastreamento será um scraping realizado no site "<https://www.linkcorreios.com.br/?id={codigo}>", retornando os dados de rastreamento daquele pacote em uma saída no terminal.

Assim, ao ser executado o arquivo app.py, o usuário irá ver uma janela do terminal com onde será digitado o código de rastreio para o scraping, conforme figura no próximo slide:



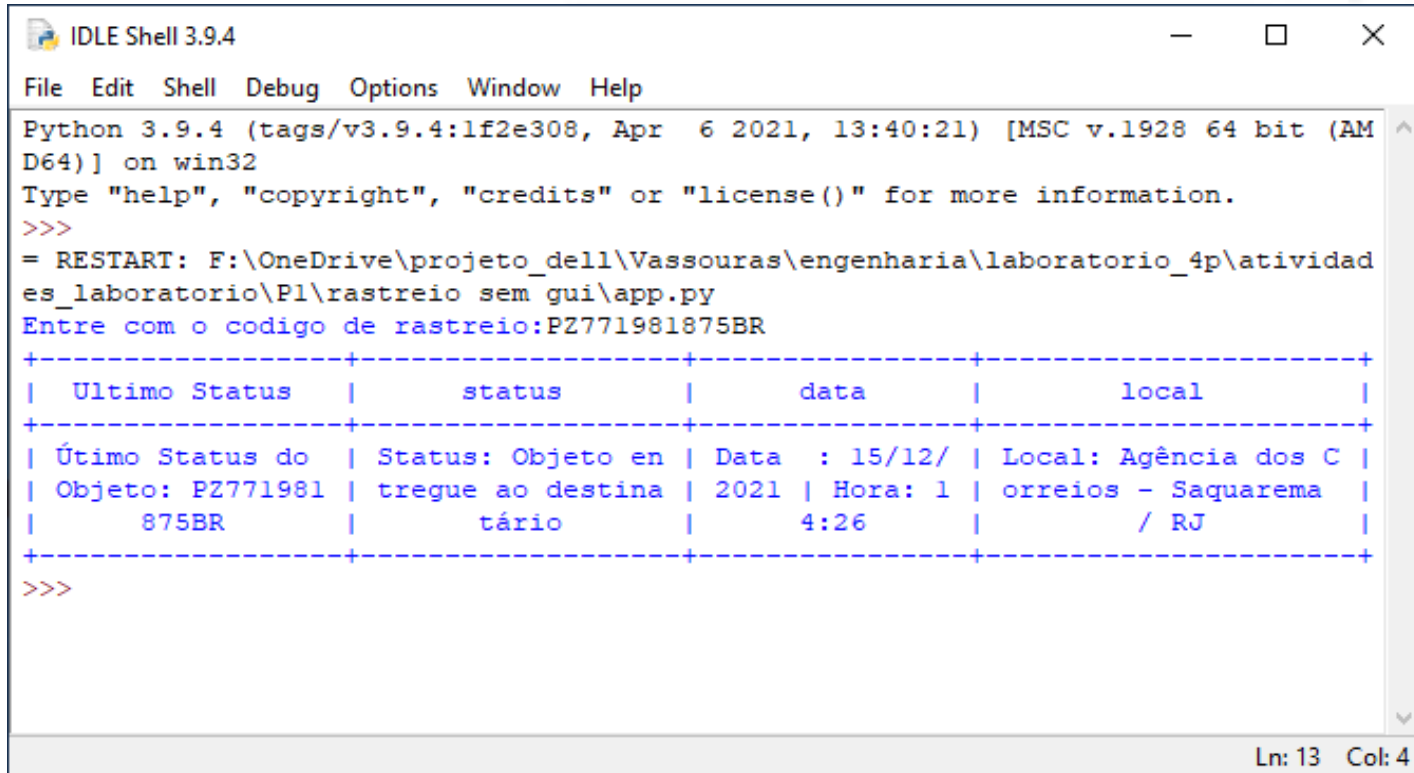
## Atividade Avaliativa



```
*IDLE Shell 3.9.4*
File Edit Shell Debug Options Window Help
Python 3.9.4 (tags/v3.9.4:1f2e308, Apr  6 2021, 13:40:21) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: F:\OneDrive\projeto_dell\Vassouras\engenharia\laboratorio_4p\atividades_laboratorio\Pl\rastreio sem gui\app.py
Entre com o codigo de rastreio:|
```

Ln: 5 Col: 31

## Atividade Avaliativa



```
Python 3.9.4 (tags/v3.9.4:1f2e308, Apr 6 2021, 13:40:21) [MSC v.1928 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: F:\OneDrive\projeto_dell\Vassouras\engenharia\laboratorio_4p\atividades_laboratorio\Pl\rastreo sem gui\app.py
Entre com o codigo de rastreo:PZ771981875BR
+-----+-----+-----+-----+
| Ultimo Status | status | data | local |
+-----+-----+-----+-----+
| Último Status do | Status: Objeto em | Data : 15/12/ | Local: Agência dos C | |
| Objeto: PZ771981 | tregue ao destina | 2021 | Hora: 1 | orreios - Saquarema |
| 875BR | tário | 4:26 | / RJ |
+-----+-----+-----+-----+
>>>
```

Ln: 13 Col: 4

## Atividade Avaliativa

### Usabilidade

Ao ser preenchido o código de rastreamento correto, o arquivo `api.py` fará a consulta aos correios, através do link acima, passando o código de rastreio fornecido. Utilizando de da biblioteca BeautifulSoup, o resultado será tratado para recuperar o status do rastreio que deverá ser apresentado no terminal, conforme figura anterior.

## Atividade Avaliativa

### Bibliotecas Necessárias

Beautifultable, requests e bs4

### Dicas

A biblioteca requests precisa ser configurada para 'utf-8' e o BeautifulSoup trabalha muito bem localizando tags como <div>, <ul>, <li> e etc.

Neste projeto serão apenas 3 arquivos python obrigatórios (api.py, app.py e constants.py). O arquivo constants só deverá ter uma única linha de código, onde uma variável recebe a url principal (sem o código).

### Atividades Complementares (Obrigatória)

Todos os métodos e seus argumentos devem ser documentados de um modo compatível ao Doxygen (<http://www.doxygen.nl/manual/>).

O Doxygen, além de ser um código aberto, é capaz de produzir documentação para programas escritos em diversas linguagens, como C, C++, Python, Perl, PHP, etc.

Para tanto, baixe o pacote disponível para Linux, MacOS ou Windows, no próprio site do desenvolvedor (<http://www.doxygen.nl/download.html>), e disponibilize no diretório do projeto, um arquivo de configuração chamado Doxyfile (pode ser criado pelo comando **doxygen -g**).

## Atividade Avaliativa

### O que será entregue

Um arquivo compactado (zip, rar, 7-zip, etc) contendo todos os arquivos .py e todos os arquivos da documentação.

Para testar utilize códigos de rastreio oficiais dos correios, como:  
QB525544792BR e PZ771981875BR

**Contato**



**Professor:**

André Saraiva, MSc

**E-mail:**

[andre.saraiva@universidadedevassouras.edu.br](mailto:andre.saraiva@universidadedevassouras.edu.br)