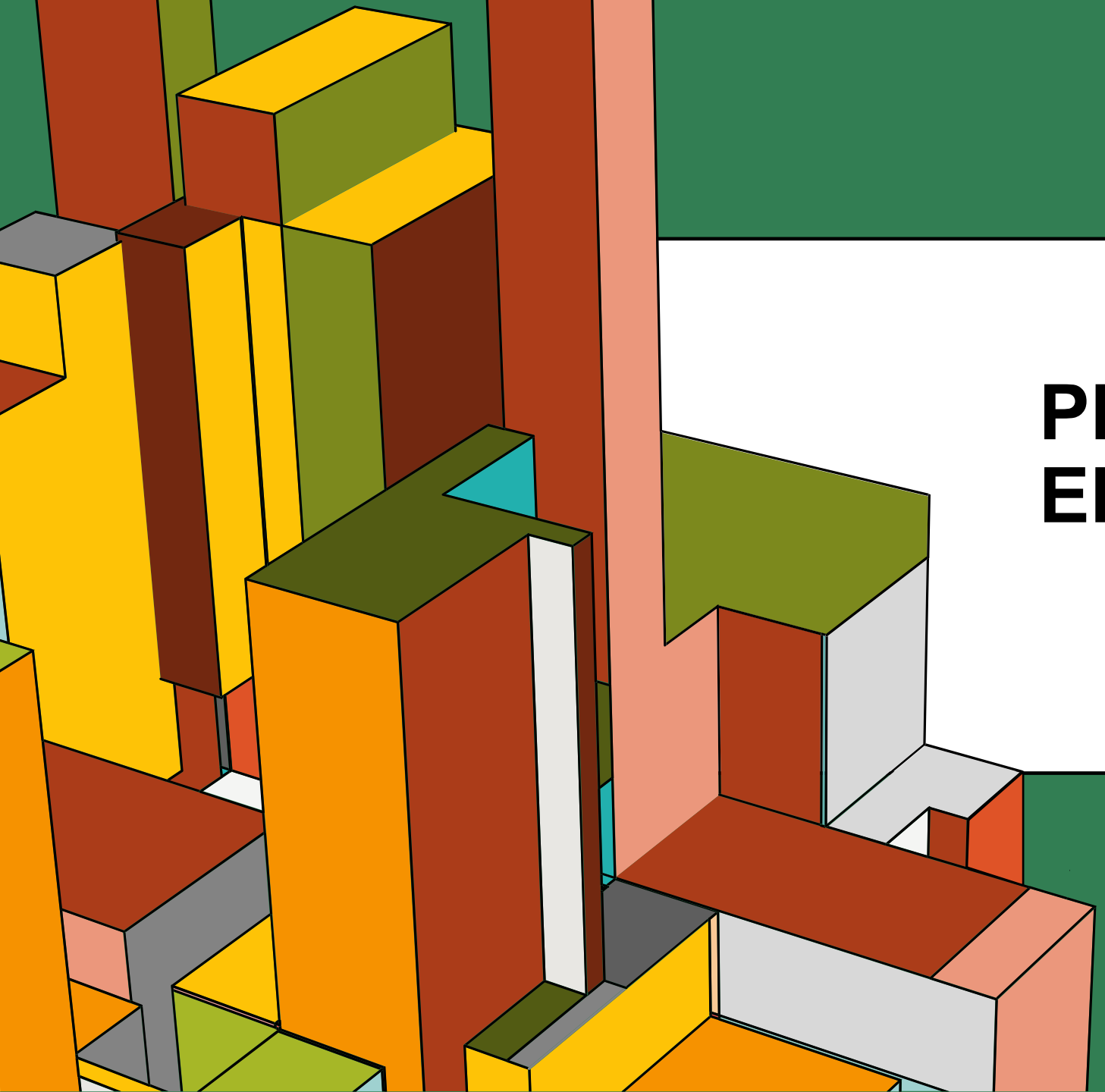


PROGRAMAÇÃO EM PYTHON

python

SENAI

AULA 07

TUPLAS

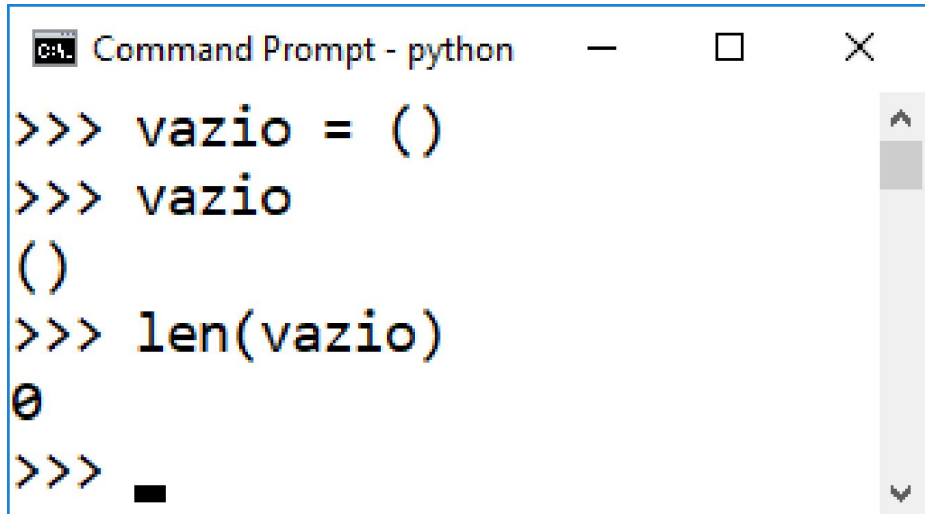
Tuplas são semelhantes às listas, porém, são imutáveis.
Não podemos acrescentar, apagar ou fazer atribuições aos seus itens
As tuplas são criadas usando se parênteses em vez de colchetes, porém, os parênteses não são obrigatórios

Veja a criação de tuplas com e sem parênteses linguagens

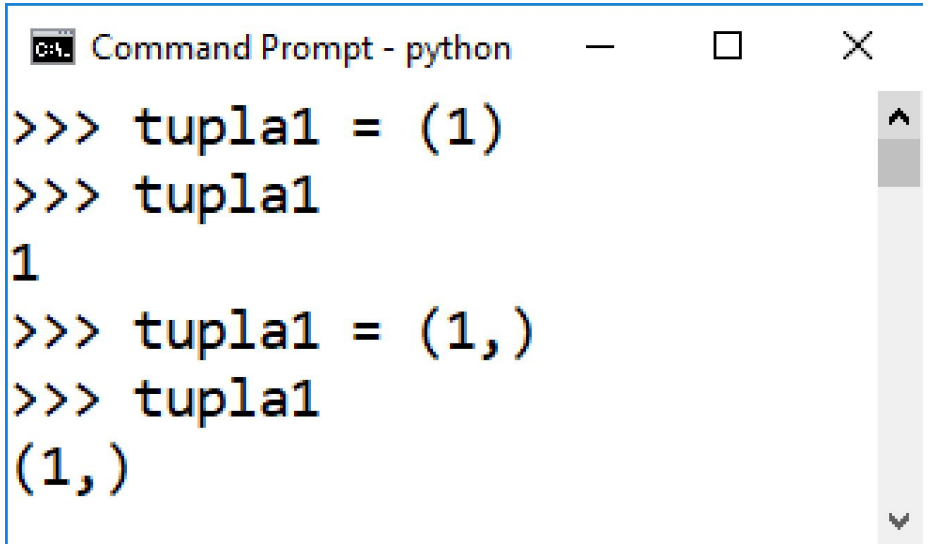
```
linguagens = ("Assembly", "Cobol", "C", "C++")
print(linguagens)
#Resultado: ('Assembly', 'Cobol', 'C', 'C++')

linguagens = "Python", "Java", "Go", "C#"
print(linguagens)
#Resultado: ('Python', 'Java', 'Go', 'C#')
```

TUPLAS

É possível criar uma tupla vazia



Criando uma tupla com um único elemento



ACESSANDO TUPLAS

Podemos acessar os elementos da tupla pelo índice e usar fatiamento países

```
paises = "Brasil", "Paraguai", "Uruguai", "México"
pais = paises[0]
print(pais) # Brasil

fatia = paises[1:3]
print(fatia) # ('Paraguai', 'Uruguai')
```

ALTERAR UMA TUPLA

Se tentarmos alterar um item da tupla, é gerado o erro “O objeto não suporta a atribuição de itens”

```
paises = "Brasil", "Paraguai", "Uruguai", "México"
paises[1] = "Colômbia"
```

```
Traceback (most recent call last):
  File “.../tupla.py", line 2, in <module>
    paises[1] = "Colômbia"
TypeError: 'tuple' object does not support item assignment
```

IMPRIMIR TUPLAS

Podemos imprimir os elementos de uma tupla usando o comando for

```
paises = "Brasil", "Paraguai", "Uruguai", "México"

for pais in paises:
    print(pais)
```

Como eu já disse, melhor não usar acentuação. Nem todas linguagens aceitam....

Resultado:
Brasil
Paraguai
Uruguai
México

CONVERTENDO UMA LISTA EM UMA TUPLA

```
lista_carros = ["Gol", "Corolla", "Ranger", "Kadett",
                "Fusca", "Clio"]
tupla_carros = tuple(lista_carros)
print(f"Tupla carros: {tupla_carros}")
```

Resultado:

Tupla carros: ('Gol', 'Corolla', 'Ranger', 'Kadett', 'Fusca', 'Clio')

CONVERTENDO UMA TUPLA EM UMA LISTA

```
tupla_carros = "Gol", "Corolla", "Ranger", "Kadett",
               "Fusca", "Clio"
lista_carros = list(tupla_carros)
print(f"Lista carros: {lista_carros}")
```

Resultado:

Lista carros: ['Gol', 'Corolla', 'Ranger', 'Kadett', 'Fusca', 'Clio']

DESEMPACOTANDO ELEMENTOS DA TUPLA

```
tupla_carros = "Golf", "Corolla", "Civic"
carro1, carro2, carro3 = tupla_carros
print(f"Carro1: {carro1}")
print(f"Carro2: {carro2}")
print(f"Carro3: {carro3}")
```

Resultado:
Carro1: Golf
Carro2: Corolla
Carro3: Civic

DESEMPACOTANDO ELEMENTOS DA TUPLA USANDO ATRIBUIÇÃO MÚLTIPLA

```
tupla_carros = "Golf", "Corolla", "Civic", "Opala", \
               "Tucson", "Elantra"
carro1, *carros = tupla_carros
print(f"Carro1: {carro1}")
print(f"Carros: {carros}") # Uma lista com os itens restantes
```

Resultado:
Carro1: Golf
Carros: ['Corolla', 'Civic', 'Opala', 'Tucson', 'Elantra']

Barra invertida indica que a linha continua.

ATRIBUIÇÃO MÚLTIPLA NÃO PRECISA ESTAR NO FIM DA SEQUÊNCIA

```
tupla_carros = "Golf", "Corolla", "Civic", "Opala", \
               "Tucson", "Elantra"
*carros, tucson, elantra = tupla_carros
print(f"Carros: {carros}")
print(f"Tucson: {tucson}")
print(f"Elantra: {elantra}")
```

Resultado:
Carros: ['Golf', 'Corolla', 'Civic', 'Opala']
Tucson: Tucson
Elantra: Elantra

EXERCÍCIO 1

Dada uma lista L de n valores inteiros, escreva um programa que remova os números pares da lista

EXERCÍCIO 2

Faça um programa que crie uma tupla com o que você quiser dentro e depois altere o último elemento para o número 3.

Dicionários

- Segundo Borges (2010), um dicionário é uma lista de associações composta por uma chave (de tipo imutável) e estruturas correspondentes às chaves que podem ser mutáveis ou não. Isso caracteriza os Dicionários como sendo mutáveis (pois os valores em si são mutáveis, exceto as chaves).

Dicionários

Dicionários são estruturas de dados similares às listas, porém, com propriedades de acesso diferentes Um dicionário consiste em um conjunto de chaves e valores.
Sintaxe:

dicionario = {‘a’: a, ‘b’: b, ..., ‘z’:z}
dicionario = {chave: valor, chave: valor}

Assim como as listas, os dicionários são mutáveis. Mas suas chaves devem ser de um tipo imutável
Para criar listas usamos colchetes “[]” para criar tuplas usamos parênteses “()” para criar dicionários usamos chaves “{}”

DICIONÁRIOS

Chave	Valor
Curso	Python
Aula	Aula07
Professor	Peterson

Listas: Acessamos os elementos pelo número do índice
Dicionários: Acessamos um elemento utilizando a chave

```
1 dic = {'curso': 'Python', 'aula': 'Aula07', 'Professor': 'Peterson' }
2 print(f'Professor {dic["Professor"]} que ministra o curso de {dic["curso"]}')
```

Dicionários

Para adicionar/alterar elementos ao dicionário usamos dicionário [“ valor”] = valor
Para apagar um elemento de um dicionário usamos: dicionario.clear()
Para apagar o dicionário usamos: del dicionário, deste modo não existe mais.

```
dic["aula"] = "Aula07 - Dicionários"
dic.clear()
print(dic)
del dic
```

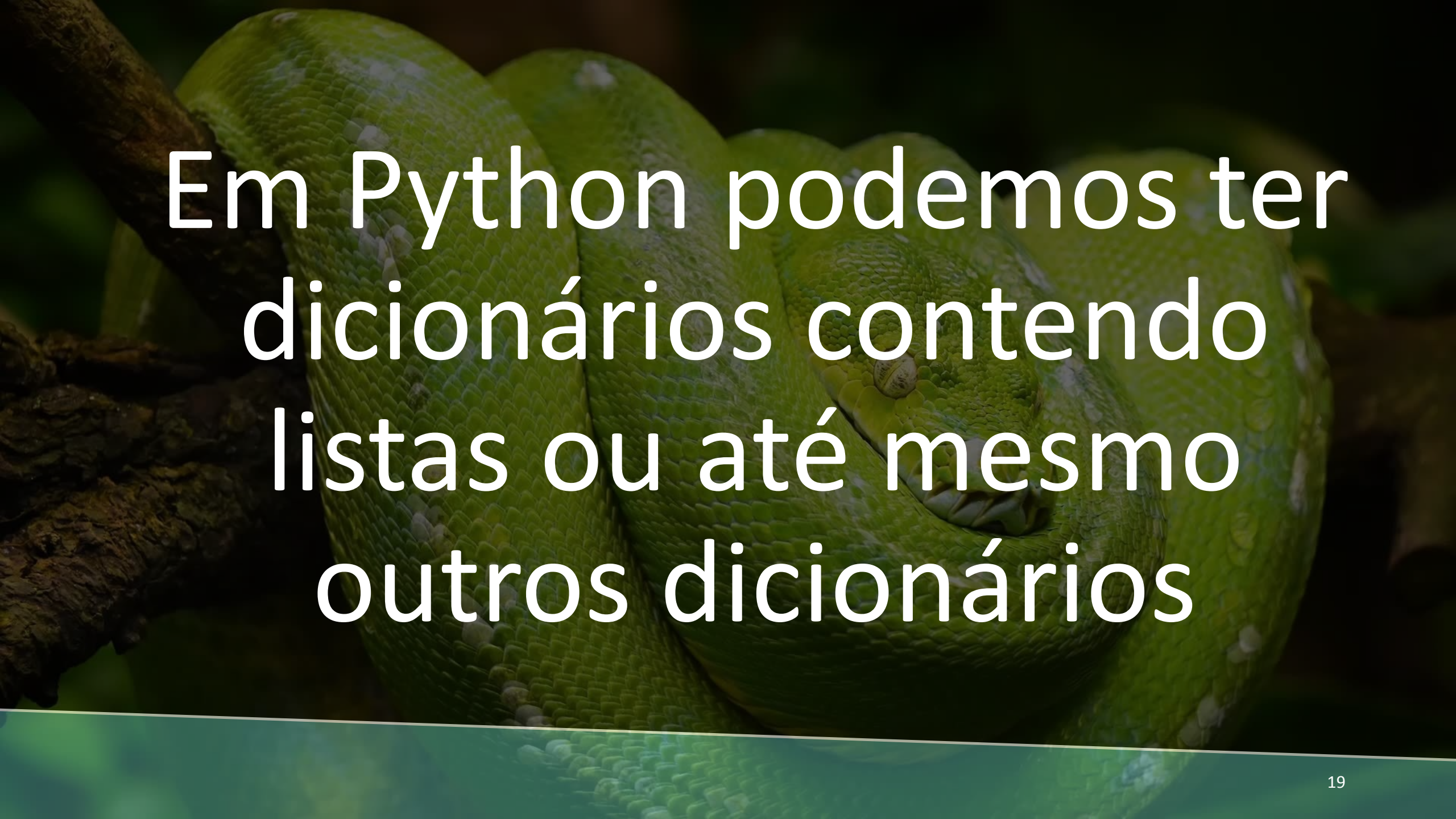
OBTENDO ITENS, CHAVES E VALORES DE UM DICIONÁRIO

key(), items() e values() retornam views (visualizações), que são iteradores dos tipos dict_keys dict_items e dict_values que devolvem um elemento de cada vez.

```
carros = {'marca': ' VW', 'modelo': ' Gol', 'ano_modelo' : 2016}
print(f'Itens do dicionário carros : {carros.items()}')
print(f'Chaves do dicionário carros : {carros.keys()}')
print(f'Valores do dicionário carros : {carros.values()}')
```

EXEMPLO

```
produtos = { "Mouse": 98.75,
             "Teclado": 125.65,
             "Monitor": 134.78,
             "Gabinete": 170.00,
             "HD Externo": 510.50,
             "Headset": 125.45 }
while True:
    produto = input("Informe o produto a pesquisar o preço ou fim para sair: ")
    if produto == "fim":
        break
    if produto in produtos:
        print(f"Produto {produto} custa {produtos[produto]}.")
    else:
        print(f"Produto {produto} não encontrado.")
```



Em Python podemos ter dicionários contendo listas ou até mesmo outros dicionários

OUTRAS FUNÇÕES

- Copiando um dicionário para outro
- Adicionando conteúdo de um dicionário em outro dicionário
- Para retornar quantos conjuntos chave/valor existem no dicionário usamos a função len().

```
dic = {'nome':"Fulano", 'sobrenome':'de Tal'}
local = {'UF':"SP", 'cidade':'São Carlos'}
dic2 = dic.copy()
dic.update(local)
print(len(dic))
print(f'dic: {dic}')
print(f'dic2: {dic2}')
```

EXEMPLO 2

```
1 aluno = {}
2 aluno['nome'] = input('Digite o nome do aluno: ')
3 aluno['media'] = float(input(f'Digite a média do aluno {aluno["nome"]}: '))
4 if aluno['media'] >= 7:
5   aluno['situacao'] = 'Aprovado'
6 elif 5 <= aluno['media'] <7:
7   aluno['situacao'] = 'Recuperação'
8 else:
9   aluno['situacao'] = 'Reprovado'
10 print('=='*30)
11 for k, v in aluno.items():
12   print(f'{k} = {v}')
```

EXERCÍCIOS

1. Faça um dicionário com as 5 pessoas mais perto de você, tendo o nome como chave e a cor da camisa que está usando como valor.

2. Crie um dicionário vazio semana = {} e o complete com uma chave para cada dia da semana, tendo como seu valor uma lista com as aulas que você tem nesse dia (sábado e domingo recebem listas vazias, ou você tem aula?).

3. Crie um dicionário vazio filmes = {}. Utilize o nome de um filme como chave. E, como valor, outro dicionário contendo o vilão e o ano em que o filme foi lançado. Preencha 3 filmes.