

PROGRAMAÇÃO I

Curso Técnico Subsequente em Informática
Lucas Sampaio Leite



Tuplas

- Tuplas são muito semelhantes às listas, mas são imutáveis.
- Isso significa que, depois de criar uma tupla:
 - Não é possível alterar elementos existentes;
 - Não é possível adicionar ou remover elementos.

```
lista = [1, 2, 3]
lista[0] = 100      # permitido
lista.append(4)     # permitido
print(lista)        # [100, 2, 3, 4]
```

```
tupla = (1, 2, 3)
tupla[0] = 100      # erro
tupla.append(4)     # erro
```

Tuplas

- Tuplas são muito semelhantes às listas, mas são imutáveis.
- Isso significa que, depois de criar uma tupla:
 - Não é possível alterar elementos existentes;
 - Não é possível adicionar ou remover elementos.

```
lista = [1, 2, 3]
lista[0] = 100      # permitido
lista.append(4)     # permitido
print(lista)        # [100, 2, 3, 4]
```

```
tupla = (1, 2, 3)
tupla[0] = 100      # erro
tupla.append(4)     # erro
```

De forma geral as tuplas são mais eficientes que as listas pois consomem menos recursos computacionais (memória), por serem estruturas mais simples (Borges, 2014).

Tuplas

- Criando uma Tupla:

```
tupla = (0, 1, 2, "muito parecido com listas")  
print(tupla)
```

```
tupla = 0, 1, 2, "muito parecido com listas"  
print(tupla)
```



```
(0, 1, 2, 'muito parecido com listas')  
(0, 1, 2, 'muito parecido com listas')
```

Tuplas

- Acessando elementos pelo índice:
 - Cada elemento de uma lista ou tupla pode ser acessado usando sua posição (índice), começando em 0.

```
x = (1, 2, 3)
print(x)
print(x[0])
print(x[0:2])
print(len(x))
x[0] = 2
```

Tuplas

- Acessando elementos pelo índice:
 - Cada elemento de uma lista ou tupla pode ser acessado usando sua posição (índice), começando em 0.

```
x = (1, 2, 3)
print(x)
print(x[0])
print(x[0:2])
print(len(x))
x[0] = 2
```



```
(1, 2, 3)
1
(1, 2)
3
Traceback (most recent call last):
  File "/home/lucas/Documents/vscode-projects/lucas.py", line 37,
in <module>
    x[0]=2
TypeError: 'tuple' object does not support item assignment
```

Tuplas

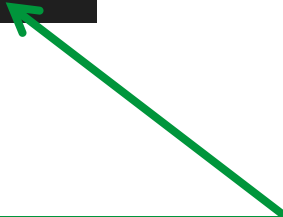
- Tupla com apenas um elemento:
 - Para criar, é necessário adicionar uma vírgula após o elemento:

```
tupla = (42,)
```

Tuplas

- Tupla com apenas um elemento:
 - Para criar, é necessário adicionar uma vírgula após o elemento:

```
tupla = (42,)
```



A vírgula é obrigatória. Sem ela, o interpretador entende que tupla é apenas uma variável comum do mesmo tipo do elemento, e não uma tupla.

Tuplas

```
variavel = 1,  
print(variavel)  
print(type(variavel))  
  
variavel = 1  
print(variavel)  
print(type(variavel))
```



```
(1,)  
<class 'tuple'>  
1  
<class 'int'>
```

Tuplas

- Tupla vazia:
 - Pode ser criada de duas formas equivalentes:

```
tupla_vazia = ()  
tupla_vazia = tuple()
```

Uma tupla vazia é útil para inicializar uma variável de forma imutável ou como retorno padrão de funções, garantindo que o conteúdo não será alterado e mantendo compatibilidade com operações que esperam tuplas..

Tuplas

- Os parênteses são opcionais em tuplas quando não geram ambiguidade.
- Qual seria o resultado impresso?

```
tupla = ()  
print(tupla)  
print(type(tupla))
```

```
tupla = 3*(10+3)  
print(tupla)  
print(type(tupla))
```

```
tupla = 3*(10+3,)  
print(tupla)  
print(type(tupla))
```

```
tupla = 3*10+3,  
print(tupla)  
print(type(tupla))
```

Tuplas

- Os parênteses são opcionais em tuplas quando não geram ambiguidade.
- Qual seria o resultado impresso?

```
()  
<class 'tuple'>  
39  
<class 'int'>  
(13, 13, 13)  
<class 'tuple'>  
(33,)  
<class 'tuple'>
```

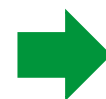


```
tupla = ()  
print(tupla)  
print(type(tupla))  
  
tupla = 3*(10+3)  
print(tupla)  
print(type(tupla))  
  
tupla = 3*(10+3,)  
print(tupla)  
print(type(tupla))  
  
tupla = 3*10+3,  
print(tupla)  
print(type(tupla))
```

Tuplas

- É possível converter listas em tuplas utilizando a função `tuple()`.

```
lista = [0, 1, 2, 3, 4]
print(lista)
tupla = tuple(lista)
print(tupla)
lista = list(tupla)
print(lista)
```



```
[0, 1, 2, 3, 4]
(0, 1, 2, 3, 4)
[0, 1, 2, 3, 4]
```

Tuplas

- O mesmo procedimento também pode ser aplicado a uma string, utilizando a função `str()`.

```
tupla = (0, 1, 2, "muito parecido com listas")  
print(tupla)  
string = str(tupla)  
print(string)  
print(type(string))
```



```
(0, 1, 2, 'muito parecido com listas')  
(0, 1, 2, 'muito parecido com listas')  
<class 'str'>
```


Tuplas

- Embora uma tupla seja imutável, ela pode conter elementos mutáveis (como listas), que podem ser modificados mesmo dentro da tupla.

```
tupla = [0, 1], "a", "b"  
print(tupla)  
tupla[0].append(2)  
print(tupla)  
tupla[0] = [0, 1, 2]
```



```
([0, 1], 'a', 'b')  
([0, 1, 2], 'a', 'b')  
Traceback (most recent call last):  
  File "/home/lucas/Documents/vscode-projects/lucas.py", line 36,  
    in <module>
```

Tuplas

- É possível concatenar tuplas utilizando o operador +:

```
tupla = (1, 2, 3)
print(tupla)
tupla = tupla + (4, 5, 6)
print(tupla)
tupla += (7, 8)
print(tupla)
```



```
(1, 2, 3)
(1, 2, 3, 4, 5, 6)
(1, 2, 3, 4, 5, 6, 7, 8)
```


Tuplas

- Desempacotamento de tuplas (*tuple unpacking*) é a técnica de atribuir elementos de uma tupla a variáveis individuais de forma direta, sem precisar acessar cada elemento pelo índice:

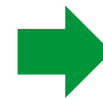
```
coordenadas = (10, 20, 30)
```

```
x, y, z = coordenadas
```

```
print(x)
```

```
print(y)
```

```
print(z)
```



```
10  
20  
30
```

Tuplas

- Desempacotamento de tuplas (*tuple unpacking*) é a técnica de atribuir elementos de uma tupla a variáveis individuais de forma direta, sem precisar acessar cada elemento pelo índice:

```
coordenadas = (10, 20, 30)
```

```
x, y, z = coordenadas
```

```
print(x)
```

```
print(y)
```

```
print(z)
```



```
10  
20  
30
```

O número de variáveis deve coincidir com o número de elementos da tupla.

Tuplas

- Desempacotamento de tuplas (*tuple unpacking*) é a técnica de atribuir elementos de uma tupla a variáveis individuais de forma direta, sem precisar acessar cada elemento pelo índice:

```
def pontos():  
    return 1, 2, 3  
  
a, b, c = pontos()
```

Tuplas

- Operadores básicos sobre tuplas:

| Descrição | Expressão | Resultado |
|--|---|----------------------------|
| Número de elementos que a tupla contém | <code>len((1,2,3))</code> | 3 |
| Concatenação | <code>(1,2,3) + (4,5,6)</code> | <code>(1,2,3,4,5,6)</code> |
| Repetição | <code>(1,) * 4</code> | <code>(1,1,1,1)</code> |
| Pertencimento | <code>3 in (1,2,3)</code> | True |
| variavel teste | Conter espaço | Não |
| Iteração | <code>for x in (1,2,3): print(x)</code> | 1 2 3 |

Tuplas

- Os parênteses são opcionais em tuplas quando não geram ambiguidade.
Qual seria o resultado impresso?

```
cliente_1 = (1, "Lucas", "999.999.999-99", "9.999.999")  
cliente_2 = (2, "Maria", "888.888.888-88", "8.888.888")  
clientes = [cliente_1, cliente_2]
```

Quais são as vantagens dessa abordagem?

Tuplas

- Em geral, tuplas podem ser substituídas por listas, oferecendo flexibilidade em vez de eficiência.
- No entanto, algumas construções em Python exigem tuplas ou sequências imutáveis, por exemplo:
 - Chaves de dicionários: apenas tuplas (e outros tipos imutáveis) podem ser usadas como chave;
 - Funções com número variável de argumentos: os argumentos são acessados por meio de tuplas;
 - Representação de coleções imutáveis: útil para coordenadas geográficas, informações fixas de uma entidade, entre outros casos em que os elementos não devem ser alterados.

Exercícios

1. Crie uma tupla chamada aluno contendo o nome, idade e nota final de um aluno. Imprima o nome e a nota final usando indexação.
2. Crie uma função chamada valores que retorna três números quaisquer. Atribua o retorno da função a uma tupla chamada resultado. Imprima o resultado e seu tipo.
3. Dadas as tuplas: $t1 = (1, 2)$ e $t2 = (3, 4)$. Crie uma nova tupla $t3$ concatenando $t1$ e $t2$. Crie outra tupla $t4$ repetindo $t1$ três vezes. Imprima $t3$ e $t4$.

Exercícios

4. Dada uma lista de números, escreva um programa em Python que contenha uma função capaz de receber essa lista e retornar uma nova lista de tuplas, em que o primeiro elemento de cada tupla seja o número original e o segundo elemento seja o seu cubo. Em seguida, imprima a lista de entrada e o resultado, conforme o exemplo abaixo:

```
Entrada: lista = [1, 2, 3]
Resultado: [(1, 1), (2, 8), (3, 27)]
Entrada: lista = [9, 5, 6]
Resultado: [(9, 729), (5, 125), (6, 216)]
```


Exercícios

5. Dado um texto armazenado em uma string, escreva um programa em Python que conte a quantidade de ocorrências de cada palavra. O programa deve imprimir uma lista de tuplas, em que cada tupla contenha a palavra e o número de vezes que ela aparece no texto.

Exemplo de entrada: “A casa é bonita, a casa é azul”

Saída: [('A', 1), ('casa', 2), ('é', 2), ('bonita,', 1), ('a', 1), ('azul', 1)]

6. Crie um programa que registre os dados dos alunos de uma turma, incluindo nome, idade e notas. Utilize uma lista de tuplas para armazenar os registros dos alunos e permita a busca de um aluno pelo nome.

Exercícios

7. Escreva um programa com uma função em que dada uma tupla, remova os valores duplicados.

Exemplo de entrada: tupla = (1, 3, 5, 2, 3, 5, 1, 1, 3)

Resultado: A tupla original é: (1, 3, 5, 2, 3, 5, 1, 1, 3)

A tupla após a remoção de duplicatas: (1, 2, 3, 5)

Obs: Não converter a tupla para nenhum outro tipo.

Dúvidas



PROGRAMAÇÃO I

Curso Técnico Subsequente em Informática
Lucas Sampaio Leite

