centro universitário facvest
unifacvest

Fábrica de Software

Aula 05

Python

TUPLAS

Tupla é um tipo de estrutura de dados utilizada em Python que funciona de modo semelhante a uma lista, entretanto, com a característica principal de ser imutável. Isso significa que quando uma tupla é criada não é possível adicionar, alterar ou remover seus elementos. Sua característica de imutabilidade oferece **segurança nas informações** armazenadas. Por isso, uma das finalidades da tupla é armazenar uma sequência de dados que não será modificada em outras partes do código.

Tuplas são sequências de valores de qualquer tipo. Elas podem ser indexadas por números inteiros.

Como declarar tuplas?

A declaração de tuplas em Python é muito simples, basta adicionar seus elementos entre parênteses e separados por vírgula, conforme o exemplo a seguir:

```
tupla = (20,30,40,50,60)
tupla
```

```
(20, 30, 40, 50, 60)
```

Caso você não coloque os parênteses, ele será uma tupla também!

```
tupla2 = 20,30,40,50,60
tupla2
```

```
(20, 30, 40, 50, 60)
```

```
type(tupla2)
```

```
tuple
```

PS: É importante dizer que **a utilização dos parênteses não é obrigatória.** No entanto, é considerada uma boa prática. Você pode fazer uma tupla vazia ou, ainda, usar a função tuple().

```
#Tupla Vazia

tupla3 = ()
```

```
# Tupla com a função tuple()
tupla4 = tuple("Teste")
tupla4
```

```
('T', 'e', 's', 't', 'e')
```

Para criar uma tupla com um único elemento, é preciso incluir uma vírgula final. Um valor entre parênteses não é uma tupla.

```
valor_unico = (3,)
valor_unico
```

```
(3,)
```

```
type(valor_unico)
```

```
tuple
```

```
#Se não colocar a vírgula,não é uma tupla!!
nao_tupla = (3)
type(nao_tupla)
```

```
int
```

A maior parte dos operadores de lista também funciona com as tuplas. O operador colchetes indexa um elemento.

```
#Tupla p
p = ('a','b','c','d','e')
p
```

```
('a', 'b', 'c', 'd', 'e')
```

```
# Primeiro elemento da tulpa (index zero)
p[0]
```

```
'a'
```

```
# Elementos 0, 1 e 2
p[0:3]
```

```
('a', 'b', 'c')
```

Ao tentar alterar uma tupla, recebemos um erro:

```
p[2] = 'z'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-14-104f0fef12f7> in <module>
----> 1 p[2] = 'z'

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

DICIONÁRIOS

Dicionário é um tipo diferente de coleção. Ele é um tipo de mapeamento nativo do Python. Um dicionário se parece com uma lista, mas é mais geral. A associação, ou mapeamento, é feita a partir de uma chave, que pode ser qualquer tipo imutável, para um valor, que pode ser qualquer objeto de dados do Python.

Se nas listas, os elementos são acessados por meio de uma posição ou índice, nos dicionários isso ocorre de forma diferente. O acesso às informações acontece por meio de chaves.

Para criar um dicionário em Python, basicamente, as informações precisam ser passadas entre chaves {}, que por sua vez precisam seguir a estrutura de pares “chave: valor” - denotada na literatura pelos termos em inglês key:value.

A função dict() cria um novo dicionário vazio.

```
# Dicionário com a função dict()

p = dict()
p
```

```
{}
```

```
type(p)
```

```
dict
```

Podemos criar o dicionário usando as chaves também:

```
# Dicionário com chaves {}
p = {'Nome':'Paulo', 'Sobrenome': 'Martins', 'Idade': 15}
p
```

```
{'Nome': 'Paulo', 'Sobrenome': 'Martins', 'Idade': 15}
```

Como acessar os itens de um dicionário?

O acesso aos itens ou elementos de um dicionário Python pode ser realizado de modo similar ao método de acesso das listas. No entanto, é necessário passar a chave para a qual se deseja acessar o valor. Ou ainda, através da função get(), que pode ter o retorno da função atribuída a uma variável ou ser chamada diretamente na função print.

```
# Dicionário com chaves {}
p = {'Nome':'Paulo', 'Sobrenome': 'Martins', 'Idade': 15}
p
```

```
{'Nome': 'Paulo', 'Sobrenome': 'Martins', 'Idade': 15}
```

```
# Acessar o nome:
p['Nome']
```

```
'Paulo'
```

```
#Acessar Sobrenome
p['Sobrenome']
```

```
'Martins'
```

```
#Usando get
idade = p.get('Idade')
print(idade)
```

```
15
```

Como adicionar elementos ao dicionário?

É possível inserir um item em um dicionário dict com chave key e valor value simplesmente fazendo a operação direta:

dict[key] = value

```
# Adicionar um valor para sexo

p['Sexo'] = 'masculino'
```

```
p
```

```
{'Nome': 'Paulo', 'Sobrenome': 'Martins', 'Idade': 15, 'Sexo': 'masculino'}
```

Como remover itens do dicionário?

Se for necessário excluir um determinado item do dicionário, basta usarmos o método pop(), que irá receber a key de deletar o item com a chave indicada.

```
#Retirando o Sexo
p.pop('Sexo')
p
```

```
{'Nome': 'Paulo', 'Sobrenome': 'Martins', 'Idade': 15}
```

Métodos dos Dicionários

keys

keys() retorna uma lista com todas as chaves do dicionário.

```
p.keys()
```

```
dict_keys(['Nome', 'Sobrenome', 'Idade'])
```

values

values() retorna uma lista com todos os valores do dicionário.

```
p.values()
```

```
dict_values(['Paulo', 'Martins', 15])
```

items

items() retorna uma lista com todos os pares chave/conteúdo do dicionário.

```
p.items()
```

```
dict_items([('Nome', 'Paulo'), ('Sobrenome', 'Martins'), ('Idade', 15)])
```