MC102 — Combinando e Usando as Estruturas de Dados

Rafael C. S. Schouery
rafael@ic.unicamp.br

Universidade Estadual de Campinas

Atualizado em: 2025-02-19 11:36

Estruturas de Dados

Vimos várias formas de organizar e acessar dados:

Estruturas de Dados

Vimos várias formas de organizar e acessar dados:

- Listas

Estruturas de Dados

Vimos várias formas de organizar e acessar dados:

- Listas
- Dicionários

Estruturas de Dados

Vimos várias formas de organizar e acessar dados:

- Listas
- Dicionários
- Conjuntos

Estruturas de Dados

Vimos várias formas de organizar e acessar dados:

- Listas
- Dicionários
- Conjuntos
- Matrizes

Estruturas de Dados

Vimos várias formas de organizar e acessar dados:

- Listas
- Dicionários
- Conjuntos
- Matrizes

São formas diferentes de estruturar os dados

Estruturas de Dados

Vimos várias formas de organizar e acessar dados:

- Listas
- Dicionários
- Conjuntos
- Matrizes

São formas diferentes de estruturar os dados

- Com suas características, vantagens e desvantagens

Estruturas de Dados

Vimos várias formas de organizar e acessar dados:

- Listas
- Dicionários
- Conjuntos
- Matrizes

São formas diferentes de estruturar os dados

- Com suas características, vantagens e desvantagens

Nessa aula veremos alguns conceitos mais avançados

Estruturas de Dados

Vimos várias formas de organizar e acessar dados:

- Listas
- Dicionários
- Conjuntos
- Matrizes

São formas diferentes de estruturar os dados

- Com suas características, vantagens e desvantagens

Nessa aula veremos alguns conceitos mais avançados

- E a combinação de tais estruturas

Compreensão de Listas

É uma sintaxe do Python para criar listas rapidamente

Compreensão de Listas

É uma sintaxe do Python para criar listas rapidamente

- Ex: `[i for i in range(10)]`

Compreensão de Listas

É uma sintaxe do Python para criar listas rapidamente

- Ex: `[i for i in range(10)]`
  - `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`

Compreensão de Listas

É uma sintaxe do Python para criar listas rapidamente

- Ex: `[i for i in range(10)]`
  - `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`
- Ex: `[i ** 2 for i in range(10)]`

Compreensão de Listas

É uma sintaxe do Python para criar listas rapidamente

- Ex: `[i for i in range(10)]`
  - `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`
- Ex: `[i ** 2 for i in range(10)]`
  - `[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]`

Compreensão de Listas

É uma sintaxe do Python para criar listas rapidamente

- Ex: `[i for i in range(10)]`
  - `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`
- Ex: `[i ** 2 for i in range(10)]`
  - `[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]`
- Ex: `[math.factorial(i) for i in range(5)]`

Compreensão de Listas

É uma sintaxe do Python para criar listas rapidamente

- Ex: `[i for i in range(10)]`
  - `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`
- Ex: `[i ** 2 for i in range(10)]`
  - `[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]`
- Ex: `[math.factorial(i) for i in range(5)]`
  - `[1, 1, 2, 6, 24]`

Compreensão de Listas

É uma sintaxe do Python para criar listas rapidamente

- Ex: `[i for i in range(10)]`
  - `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`
- Ex: `[i ** 2 for i in range(10)]`
  - `[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]`
- Ex: `[math.factorial(i) for i in range(5)]`
  - `[1, 1, 2, 6, 24]`
- Ex: `[x.upper() for x in lista]`

Compreensão de Listas

É uma sintaxe do Python para criar listas rapidamente

- Ex: `[i for i in range(10)]`
  - `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`
- Ex: `[i ** 2 for i in range(10)]`
  - `[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]`
- Ex: `[math.factorial(i) for i in range(5)]`
  - `[1, 1, 2, 6, 24]`
- Ex: `[x.upper() for x in lista]`
  - Lista com as strings de `lista` todas em maiúsculas

Compreensão de Listas

É uma sintaxe do Python para criar listas rapidamente

- Ex: `[i for i in range(10)]`
  - `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`
- Ex: `[i ** 2 for i in range(10)]`
  - `[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]`
- Ex: `[math.factorial(i) for i in range(5)]`
  - `[1, 1, 2, 6, 24]`
- Ex: `[x.upper() for x in lista]`
  - Lista com as strings de `lista` todas em maiúsculas

Sintaxe: `[exp for var in iter]`

Compreensão de Listas

É uma sintaxe do Python para criar listas rapidamente

- Ex: `[i for i in range(10)]`
  - `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`
- Ex: `[i ** 2 for i in range(10)]`
  - `[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]`
- Ex: `[math.factorial(i) for i in range(5)]`
  - `[1, 1, 2, 6, 24]`
- Ex: `[x.upper() for x in lista]`
  - Lista com as strings de `lista` todas em maiúsculas

Sintaxe: `[exp for var in iter]`

- `exp` é alguma expressão que pode usar o valor de `var`

Compreensão de Listas

É uma sintaxe do Python para criar listas rapidamente

- Ex: `[i for i in range(10)]`
  - `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`
- Ex: `[i ** 2 for i in range(10)]`
  - `[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]`
- Ex: `[math.factorial(i) for i in range(5)]`
  - `[1, 1, 2, 6, 24]`
- Ex: `[x.upper() for x in lista]`
  - Lista com as strings de `lista` todas em maiúsculas

Sintaxe: `[exp for var in iter]`

- `exp` é alguma expressão que pode usar o valor de `var`
- `iter` é algum objeto que pode ser iterado

Compreensão de Listas

É uma sintaxe do Python para criar listas rapidamente

- Ex: `[i for i in range(10)]`
  - `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`
- Ex: `[i ** 2 for i in range(10)]`
  - `[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]`
- Ex: `[math.factorial(i) for i in range(5)]`
  - `[1, 1, 2, 6, 24]`
- Ex: `[x.upper() for x in lista]`
  - Lista com as strings de `lista` todas em maiúsculas

Sintaxe: `[exp for var in iter]`

- `exp` é alguma expressão que pode usar o valor de `var`
- `iter` é algum objeto que pode ser iterado
  - listas, dicionários, conjuntos, ranges, entre outros

Compreensão de Listas

É uma sintaxe do Python para criar listas rapidamente

- Ex: `[i for i in range(10)]`
  - `[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]`
- Ex: `[i ** 2 for i in range(10)]`
  - `[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]`
- Ex: `[math.factorial(i) for i in range(5)]`
  - `[1, 1, 2, 6, 24]`
- Ex: `[x.upper() for x in lista]`
  - Lista com as strings de `lista` todas em maiúsculas

Sintaxe: `[exp for var in iter]`

- `exp` é alguma expressão que pode usar o valor de `var`
- `iter` é algum objeto que pode ser iterado
  - listas, dicionários, conjuntos, ranges, entre outros

Estamos mapeando os valores do iterável em uma lista

Compreensão de Listas — Versões com `if`

Podemos também filtrar os valores antes de mapear

Compreensão de Listas — Versões com `if`

Podemos também filtrar os valores antes de mapear

- Ex: `[x for x in lista if x % 2 == 0]`

Compreensão de Listas — Versões com `if`

Podemos também filtrar os valores antes de mapear

- Ex: `[x for x in lista if x % 2 == 0]`
  - Lista dos números pares de `lista`

Compreensão de Listas — Versões com `if`

Podemos também filtrar os valores antes de mapear

- Ex: `[x for x in lista if x % 2 == 0]`
  - Lista dos números pares de `lista`
- Ex: `[i ** 2 for i in range(10) if i % 2 == 0]`

Compreensão de Listas — Versões com `if`

Podemos também filtrar os valores antes de mapear

- Ex: `[x for x in lista if x % 2 == 0]`
  - Lista dos números pares de `lista`
- Ex: `[i ** 2 for i in range(10) if i % 2 == 0]`
  - `[0, 4, 16, 36, 64]`

Compreensão de Listas — Versões com `if`

Podemos também filtrar os valores antes de mapear

- Ex: `[x for x in lista if x % 2 == 0]`
  - Lista dos números pares de `lista`
- Ex: `[i ** 2 for i in range(10) if i % 2 == 0]`
  - `[0, 4, 16, 36, 64]`
- Sintaxe: `[exp for var in iter if cond]`

Compreensão de Listas — Versões com `if`

Podemos também filtrar os valores antes de mapear

- Ex: `[x for x in lista if x % 2 == 0]`
  - Lista dos números pares de `lista`
- Ex: `[i ** 2 for i in range(10) if i % 2 == 0]`
  - `[0, 4, 16, 36, 64]`
- Sintaxe: `[exp for var in iter if cond]`

E podemos usar `if` na hora de mapear o valor:

Compreensão de Listas — Versões com `if`

Podemos também filtrar os valores antes de mapear

- Ex: `[x for x in lista if x % 2 == 0]`
  - Lista dos números pares de `lista`
- Ex: `[i ** 2 for i in range(10) if i % 2 == 0]`
  - `[0, 4, 16, 36, 64]`
- Sintaxe: `[exp for var in iter if cond]`

E podemos usar `if` na hora de mapear o valor:

- `[x if x > 0 else -x for x in range(-3, 4)]`

Compreensão de Listas — Versões com `if`

Podemos também filtrar os valores antes de mapear

- Ex: `[x for x in lista if x % 2 == 0]`
  - Lista dos números pares de `lista`
- Ex: `[i ** 2 for i in range(10) if i % 2 == 0]`
  - `[0, 4, 16, 36, 64]`
- Sintaxe: `[exp for var in iter if cond]`

E podemos usar `if` na hora de mapear o valor:

- `[x if x > 0 else -x for x in range(-3, 4)]`
  - `[3, 2, 1, 0, 1, 2, 3]`

Compreensão de Listas — Versões com `if`

Podemos também filtrar os valores antes de mapear

- Ex: `[x for x in lista if x % 2 == 0]`
  - Lista dos números pares de `lista`
- Ex: `[i ** 2 for i in range(10) if i % 2 == 0]`
  - `[0, 4, 16, 36, 64]`
- Sintaxe: `[exp for var in iter if cond]`

E podemos usar `if` na hora de mapear o valor:

- `[x if x > 0 else -x for x in range(-3, 4)]`
  - `[3, 2, 1, 0, 1, 2, 3]`
- Sintaxe: `[exp1 if cond else exp2 for var in iter]`

Compreensão de Listas — Versões com `if`

Podemos também filtrar os valores antes de mapear

- Ex: `[x for x in lista if x % 2 == 0]`
  - Lista dos números pares de `lista`
- Ex: `[i ** 2 for i in range(10) if i % 2 == 0]`
  - `[0, 4, 16, 36, 64]`
- Sintaxe: `[exp for var in iter if cond]`

E podemos usar `if` na hora de mapear o valor:

- `[x if x > 0 else -x for x in range(-3, 4)]`
  - `[3, 2, 1, 0, 1, 2, 3]`
- Sintaxe: `[exp1 if cond else exp2 for var in iter]`
- Na verdade, `exp1 if cond else exp2` funciona por si só

Compreensão de Listas — Versões com `if`

Podemos também filtrar os valores antes de mapear

- Ex: `[x for x in lista if x % 2 == 0]`
  - Lista dos números pares de `lista`
- Ex: `[i ** 2 for i in range(10) if i % 2 == 0]`
  - `[0, 4, 16, 36, 64]`
- Sintaxe: `[exp for var in iter if cond]`

E podemos usar `if` na hora de mapear o valor:

- `[x if x > 0 else -x for x in range(-3, 4)]`
  - `[3, 2, 1, 0, 1, 2, 3]`
- Sintaxe: `[exp1 if cond else exp2 for var in iter]`
- Na verdade, `exp1 if cond else exp2` funciona por si só
  - É um `if` in-line...

Compreensão de Listas — Versões com `if`

Podemos também filtrar os valores antes de mapear

- Ex: `[x for x in lista if x % 2 == 0]`
  - Lista dos números pares de `lista`
- Ex: `[i ** 2 for i in range(10) if i % 2 == 0]`
  - `[0, 4, 16, 36, 64]`
- Sintaxe: `[exp for var in iter if cond]`

E podemos usar `if` na hora de mapear o valor:

- `[x if x > 0 else -x for x in range(-3, 4)]`
  - `[3, 2, 1, 0, 1, 2, 3]`
- Sintaxe: `[exp1 if cond else exp2 for var in iter]`
- Na verdade, `exp1 if cond else exp2` funciona por si só
  - É um `if` in-line...

Dá até para combinar tudo...(mas talvez não seja tão legível)

Compreensão de Listas — Versões com `if`

Podemos também filtrar os valores antes de mapear

- Ex: `[x for x in lista if x % 2 == 0]`
  - Lista dos números pares de `lista`
- Ex: `[i ** 2 for i in range(10) if i % 2 == 0]`
  - `[0, 4, 16, 36, 64]`
- Sintaxe: `[exp for var in iter if cond]`

E podemos usar `if` na hora de mapear o valor:

- `[x if x > 0 else -x for x in range(-3, 4)]`
  - `[3, 2, 1, 0, 1, 2, 3]`
- Sintaxe: `[exp1 if cond else exp2 for var in iter]`
- Na verdade, `exp1 if cond else exp2` funciona por si só
  - É um `if` in-line...

Dá até para combinar tudo...(mas talvez não seja tão legível)

```
[x if x > 0 else -x for x in range(-3,4) if x%2 == 0]
```

Compreensão de Conjuntos e Dicionários

Você também pode criar conjuntos:

Compreensão de Conjuntos e Dicionários

Você também pode criar conjuntos:

- `{x if x > 0 else -x for x in range(-3, 4)}`

Compreensão de Conjuntos e Dicionários

Você também pode criar conjuntos:

- `{x if x > 0 else -x for x in range(-3, 4)}`
  - `{0, 1, 2, 3}`

Compreensão de Conjuntos e Dicionários

Você também pode criar conjuntos:

- `{x if x > 0 else -x for x in range(-3, 4)}`
  - `{0, 1, 2, 3}`

E dicionários:

Compreensão de Conjuntos e Dicionários

Você também pode criar conjuntos:

- `{x if x > 0 else -x for x in range(-3, 4)}`
  - `{0, 1, 2, 3}`

E dicionários:

- `{i: math.factorial(i) for i in range(5)}`

Compreensão de Conjuntos e Dicionários

Você também pode criar conjuntos:

- `{x if x > 0 else -x for x in range(-3, 4)}`
  - `{0, 1, 2, 3}`

E dicionários:

- `{i: math.factorial(i) for i in range(5)}`
  - `{0: 1, 1: 1, 2: 2, 3: 6, 4: 24}`

Exercícios

Use compreensão de listas e dicionários para:

1. Converter uma lista de temperaturas dadas em Fahrenheit para Celsius. Lembre-se que $C = ((F - 32) * 5)/9$.
2. Fazer uma lista dos números primos menores ou iguais a um `n` dado.
3. Pegar uma lista de strings que inclui palavras e números, e obter uma lista de inteiros apenas das strings que representam números não negativos.
   - Dica: use o método `isdecimal` de `str`.
   - Repita o exercício anterior para números inteiros quaisquer.
4. Fazer um dicionário que representa um histograma de uma lista (Dica: use o método `count` de `list`)
5. Criar uma matriz `n` por `m` com valor zero em cada entrada.
6. Criar uma matriz `n` por `m` com valores sequencias começando em `1` da esquerda para direita, de cima para baixo.

Listas, Dicionários e Conjuntos como Valores

Podemos ter listas, dicionários e conjuntos como valores de listas e dicionários

Listas, Dicionários e Conjuntos como Valores

Podemos ter listas, dicionários e conjuntos como valores de listas e dicionários

- Ex: Matrizes são representadas como lista de listas

Listas, Dicionários e Conjuntos como Valores

Podemos ter listas, dicionários e conjuntos como valores de listas e dicionários

- Ex: Matrizes são representadas como lista de listas
- Ex: Dicionário que, dado o RA, nos dá a lista de notas

Listas, Dicionários e Conjuntos como Valores

Podemos ter listas, dicionários e conjuntos como valores de listas e dicionários

- Ex: Matrizes são representadas como lista de listas
- Ex: Dicionário que, dado o RA, nos dá a lista de notas
- Ex: Dicionário que, dado o RA, nos dá um dicionário com chaves sendo a disciplina cursada e valor sendo a nota final.

Listas, Dicionários e Conjuntos como Valores

Podemos ter listas, dicionários e conjuntos como valores de listas e dicionários

- Ex: Matrizes são representadas como lista de listas
- Ex: Dicionário que, dado o RA, nos dá a lista de notas
- Ex: Dicionário que, dado o RA, nos dá um dicionário com chaves sendo a disciplina cursada e valor sendo a nota final.
- Ex: Uma lista de notas de alunos, onde cada elemento é um dicionário disciplina/nota.

Listas, Dicionários e Conjuntos como Valores

Podemos ter listas, dicionários e conjuntos como valores de listas e dicionários

- Ex: Matrizes são representadas como lista de listas
- Ex: Dicionário que, dado o RA, nos dá a lista de notas
- Ex: Dicionário que, dado o RA, nos dá um dicionário com chaves sendo a disciplina cursada e valor sendo a nota final.
- Ex: Uma lista de notas de alunos, onde cada elemento é um dicionário disciplina/nota.
- Ex: Um dicionário de disciplinas, onde cada valor é o conjunto de RAs que cursou a disciplina

Chaves de Dicionário/Conjuntos

Digamos que queremos associar informações a pontos no espaço:

Chaves de Dicionário/Conjuntos

Digamos que queremos associar informações a pontos no espaço:

- Temos um mapa e o nome da cidade em alguns pontos

Chaves de Dicionário/Conjuntos

Digamos que queremos associar informações a pontos no espaço:

- Temos um mapa e o nome da cidade em alguns pontos
- Ou temos sensores espalhados que medem temperatura

Chaves de Dicionário/Conjuntos

Digamos que queremos associar informações a pontos no espaço:

- Temos um mapa e o nome da cidade em alguns pontos
- Ou temos sensores espalhados que medem temperatura
- etc...

Chaves de Dicionário/Conjuntos

Digamos que queremos associar informações a pontos no espaço:

- Temos um mapa e o nome da cidade em alguns pontos
- Ou temos sensores espalhados que medem temperatura
- etc...

Poderíamos então tentar fazer o seguinte:

Chaves de Dicionário/Conjuntos

Digamos que queremos associar informações a pontos no espaço:

- Temos um mapa e o nome da cidade em alguns pontos
- Ou temos sensores espalhados que medem temperatura
- etc...

Poderíamos então tentar fazer o seguinte:

- `temperatura = {}`

Chaves de Dicionário/Conjuntos

Digamos que queremos associar informações a pontos no espaço:

- Temos um mapa e o nome da cidade em alguns pontos
- Ou temos sensores espalhados que medem temperatura
- etc...

Poderíamos então tentar fazer o seguinte:

- `temperatura = {}`
- `temperatura[[3, 7]] = 27`

Chaves de Dicionário/Conjuntos

Digamos que queremos associar informações a pontos no espaço:

- Temos um mapa e o nome da cidade em alguns pontos
- Ou temos sensores espalhados que medem temperatura
- etc...

Poderíamos então tentar fazer o seguinte:

- `temperatura = {}`
- `temperatura[[3, 7]] = 27`
- E tomar uma `TypeError: unhashable type: 'list'`...

Chaves de Dicionário/Conjuntos

Digamos que queremos associar informações a pontos no espaço:

- Temos um mapa e o nome da cidade em alguns pontos
- Ou temos sensores espalhados que medem temperatura
- etc...

Poderíamos então tentar fazer o seguinte:

- `temperatura = {}`
- `temperatura[[3, 7]] = 27`
- E tomar uma `TypeError: unhashable type: 'list'`...

Listas e conjuntos não podem ser chaves de dicionários

Chaves de Dicionário/Conjuntos

Digamos que queremos associar informações a pontos no espaço:

- Temos um mapa e o nome da cidade em alguns pontos
- Ou temos sensores espalhados que medem temperatura
- etc...

Poderíamos então tentar fazer o seguinte:

- `temperatura = {}`
- `temperatura[[3, 7]] = 27`
- E tomar uma `TypeError: unhashable type: 'list'`...

Listas e conjuntos não podem ser chaves de dicionários

- Nem chaves de conjuntos

Chaves de Dicionário/Conjuntos

Digamos que queremos associar informações a pontos no espaço:

- Temos um mapa e o nome da cidade em alguns pontos
- Ou temos sensores espalhados que medem temperatura
- etc...

Poderíamos então tentar fazer o seguinte:

- `temperatura = {}`
- `temperatura[[3, 7]] = 27`
- E tomar uma `TypeError: unhashable type: 'list'`...

Listas e conjuntos não podem ser chaves de dicionários

- Nem chaves de conjuntos
  - Ex: `{[1, 2], [3, 4]}` também dá `TypeError`

Chaves de Dicionário/Conjuntos

Digamos que queremos associar informações a pontos no espaço:

- Temos um mapa e o nome da cidade em alguns pontos
- Ou temos sensores espalhados que medem temperatura
- etc...

Poderíamos então tentar fazer o seguinte:

- `temperatura = {}`
- `temperatura[[3, 7]] = 27`
- E tomar uma `TypeError: unhashable type: 'list'`...

Listas e conjuntos não podem ser chaves de dicionários

- Nem chaves de conjuntos
  - Ex: `{[1, 2], [3, 4]}` também dá `TypeError`

Mas eu ainda preciso associar pontos a valores...

Hash

Listas, conjuntos e dicionários não podem ser chaves de dicionários e conjuntos pois podem ser alterados

Hash

Listas, conjuntos e dicionários não podem ser chaves de dicionários e conjuntos pois podem ser alterados

- São *unshashable*: não têm o método `__hash__`

Hash

Listas, conjuntos e dicionários não podem ser chaves de dicionários e conjuntos pois podem ser alterados

- São *unshashable*: não têm o método `__hash__`
- Eles não têm tal método porque seu conteúdo pode ser alterado

Hash

Listas, conjuntos e dicionários não podem ser chaves de dicionários e conjuntos pois podem ser alterados

- São *unshashable*: não têm o método `__hash__`
- Eles não têm tal método porque seu conteúdo pode ser alterado

Exemplo de um código com erro:

Hash

Listas, conjuntos e dicionários não podem ser chaves de dicionários e conjuntos pois podem ser alterados

- São *unshashable*: não têm o método `__hash__`
- Eles não têm tal método porque seu conteúdo pode ser alterado

Exemplo de um código com erro:

```
1 d = {}
2 l = [1, 2]
3 d[l] = 10
4 l.append(3)
5 print(d[[1, 2, 3]])  # o que deveria ser impresso?
6 print(d[[1, 2]])     # o que deveria ser impresso?
```

Hash

Listas, conjuntos e dicionários não podem ser chaves de dicionários e conjuntos pois podem ser alterados

- São *unshashable*: não têm o método `__hash__`
- Eles não têm tal método porque seu conteúdo pode ser alterado

Exemplo de um código com erro:

```
1 d = {}
2 l = [1, 2]
3 d[l] = 10
4 l.append(3)
5 print(d[[1, 2, 3]])  # o que deveria ser impresso?
6 print(d[[1, 2]])     # o que deveria ser impresso?
```

Os tipos `int`, `float` e `str` são *hashable*

Hash

Listas, conjuntos e dicionários não podem ser chaves de dicionários e conjuntos pois podem ser alterados

- São *unshashable*: não têm o método `__hash__`
- Eles não têm tal método porque seu conteúdo pode ser alterado

Exemplo de um código com erro:

```
1 d = {}
2 l = [1, 2]
3 d[l] = 10
4 l.append(3)
5 print(d[[1, 2, 3]])  # o que deveria ser impresso?
6 print(d[[1, 2]])     # o que deveria ser impresso?
```

Os tipos `int`, `float` e `str` são *hashable*

- Por isso podem ser usados como chave

Hash

Listas, conjuntos e dicionários não podem ser chaves de dicionários e conjuntos pois podem ser alterados

- São *unshashable*: não têm o método `__hash__`
- Eles não têm tal método porque seu conteúdo pode ser alterado

Exemplo de um código com erro:

```
1 d = {}
2 l = [1, 2]
3 d[l] = 10
4 l.append(3)
5 print(d[[1, 2, 3]])  # o que deveria ser impresso?
6 print(d[[1, 2]])     # o que deveria ser impresso?
```

Os tipos `int`, `float` e `str` são *hashable*

- Por isso podem ser usados como chave
- Lembre-se que uma string não pode ser alterada...

Hash

Listas, conjuntos e dicionários não podem ser chaves de dicionários e conjuntos pois podem ser alterados

- São *unshashable*: não têm o método `__hash__`
- Eles não têm tal método porque seu conteúdo pode ser alterado

Exemplo de um código com erro:

```
1 d = {}
2 l = [1, 2]
3 d[l] = 10
4 l.append(3)
5 print(d[[1, 2, 3]])  # o que deveria ser impresso?
6 print(d[[1, 2]])     # o que deveria ser impresso?
```

Os tipos `int`, `float` e `str` são *hashable*

- Por isso podem ser usados como chave
- Lembre-se que uma string não pode ser alterada...

Veremos mais sobre hashs em MC202 — Estruturas de Dados

Mas ainda precisamos resolver o problema...

- O Python tem um tipo parecido com lista chamado `tuple`

Mas ainda precisamos resolver o problema...

- O Python tem um tipo parecido com lista chamado `tuple`
  - O acesso é por índice e pode guardar vários tipos de dados

Mas ainda precisamos resolver o problema...

- O Python tem um tipo parecido com lista chamado `tuple`
  - O acesso é por índice e pode guardar vários tipos de dados
  - Mas, após a sua criação, não pode ser alterada

Mas ainda precisamos resolver o problema...

- O Python tem um tipo parecido com lista chamado `tuple`
  - O acesso é por índice e pode guardar vários tipos de dados
  - Mas, após a sua criação, não pode ser alterada
  - É *hashable*!

Mas ainda precisamos resolver o problema...

O Python tem um tipo parecido com lista chamado `tuple`

- O acesso é por índice e pode guardar vários tipos de dados
- Mas, após a sua criação, não pode ser alterada
- É *hashable*!

Criação:

Mas ainda precisamos resolver o problema...

O Python tem um tipo parecido com lista chamado `tuple`

- O acesso é por índice e pode guardar vários tipos de dados
- Mas, após a sua criação, não pode ser alterada
- É *hashable*!

Criação:

- Com zero elementos: `t = tuple()`

Mas ainda precisamos resolver o problema...

O Python tem um tipo parecido com lista chamado `tuple`

- O acesso é por índice e pode guardar vários tipos de dados
- Mas, após a sua criação, não pode ser alterada
- É *hashable*!

Criação:

- Com zero elementos: `t = tuple()`
- Com um elemento: `t = (1, )`

Mas ainda precisamos resolver o problema...

O Python tem um tipo parecido com lista chamado `tuple`

- O acesso é por índice e pode guardar vários tipos de dados
- Mas, após a sua criação, não pode ser alterada
- É *hashable*!

Criação:

- Com zero elementos: `t = tuple()`
- Com um elemento: `t = (1, )`
- Com mais elementos: `t = (1, 2)`

Mas ainda precisamos resolver o problema...

O Python tem um tipo parecido com lista chamado `tuple`

- O acesso é por índice e pode guardar vários tipos de dados
- Mas, após a sua criação, não pode ser alterada
- É *hashable*!

Criação:

- Com zero elementos: `t = tuple()`
- Com um elemento: `t = (1, )`
- Com mais elementos: `t = (1, 2)`
- É possível converter listas, conjuntos, etc...:
  `t = tuple(lista)`

Mas ainda precisamos resolver o problema...

O Python tem um tipo parecido com lista chamado `tuple`

- O acesso é por índice e pode guardar vários tipos de dados
- Mas, após a sua criação, não pode ser alterada
- É *hashable*!

Criação:

- Com zero elementos: `t = tuple()`
- Com um elemento: `t = (1, )`
- Com mais elementos: `t = (1, 2)`
- É possível converter listas, conjuntos, etc...:
  `t = tuple(lista)`

Acesso é por índice: `t[i]`

Mas ainda precisamos resolver o problema...

O Python tem um tipo parecido com lista chamado `tuple`

- O acesso é por índice e pode guardar vários tipos de dados
- Mas, após a sua criação, não pode ser alterada
- É *hashable*!

Criação:

- Com zero elementos: `t = tuple()`
- Com um elemento: `t = (1, )`
- Com mais elementos: `t = (1, 2)`
- É possível converter listas, conjuntos, etc...:
  `t = tuple(lista)`

Acesso é por índice: `t[i]`

Tem também métodos `count`, `index` e alguns outros métodos

Usando Tuplas

Podemos escrever:

Usando Tuplas

Podemos escrever:

- `dicionario[(1, 2)] = 10` para usar a tupla `(1, 2)` como chave

Usando Tuplas

Podemos escrever:

- `dicionario[(1, 2)] = 10` para usar a tupla `(1, 2)` como chave
- `d = {(1, 2): 10, (3, 4): 15}` para criar um dicionário

Usando Tuplas

Podemos escrever:

- `dicionario[(1, 2)] = 10` para usar a tupla `(1, 2)` como chave
- `d = {(1, 2): 10, (3, 4): 15}` para criar um dicionário
- `c = {(1, 2), (3, 4, 5)}` para criar um conjunto

Usando Tuplas

Podemos escrever:

- `dicionario[(1, 2)] = 10` para usar a tupla `(1, 2)` como chave
- `d = {(1, 2): 10, (3, 4): 15}` para criar um dicionário
- `c = {(1, 2), (3, 4, 5)}` para criar um conjunto

Podemos devolver uma tupla e atribuir para várias variáveis

Usando Tuplas

Podemos escrever:

- `dicionario[(1, 2)] = 10` para usar a tupla `(1, 2)` como chave
- `d = {(1, 2): 10, (3, 4): 15}` para criar um dicionário
- `c = {(1, 2), (3, 4, 5)}` para criar um conjunto

Podemos devolver uma tupla e atribuir para várias variáveis

- `x, y, z = f()`

Exercícios

1. Faça uma função que, dada uma lista de elementos comparáveis entre si (ex: números), devolve o menor e o maior elemento.
2. Faça uma função que, dados um dicionário onde as chaves são pontos bidimensionais e os valores são números e um ponto `(x, y)`, encontra o valor do ponto mais próximo (distância Euclidiana) de `(x, y)` que é chave do dicionário.
3. Faça uma função que, dado um dicionário onde as chaves são nomes e os valores são listas de notas, calcula a média aritmética de cada nome, devolvendo um dicionário com os resultados.
4. Faça uma função que, dado um tipo (fire, water, grass, etc) de Pokémon, imprime todos os Pokémon[1] com número menor ou igual a 151. Dica: Use a PokéAPI: `https://pokeapi.co`

[1] Pokémon é incontável, então não tem plural.