Disciplina BCM0505-15
Processamento da Informação

Vetores em Python

Profa. Carla Negri Lintzmayer

carla.negri@ufabc.edu.br

http://professor.ufabc.edu.br/~carla.negri

Centro de Matemática, Computação e Cognição
Universidade Federal do ABC

Agenda

Vetores em Python

Exemplos

Operações básicas

Mais sobre leitura de vetores

Pratique!

Vetores em Python

Vetores em Python

- A forma mais simples de representar um vetor em Python é usando **listas**.

Vetores em Python

- A forma mais simples de representar um vetor em Python é usando **listas**.
- Para criar uma lista, basta adicionar seus elementos entre colchetes e separá-los por vírgula:

```
identificador = [el1, el2, ..., eln]
```

onde $n \geq 0$.

Vetores em Python

- A forma mais simples de representar um vetor em Python é usando **listas**.
- Para criar uma lista, basta adicionar seus elementos entre colchetes e separá-los por vírgula:

```
identificador = [el1, el2, ..., eln]
```

onde $n \geq 0$.

- O acesso se dá usando também colchetes e uma posição, entre 0 e $n - 1$, sendo n a quantidade de elementos na lista:

```
identificador[i]
```

Vetores em Python

```
1  vogais = ["a", "e", "i", "o", "u"]
2  print(vogais)
3  print(vogais[2])
4
5  digitos = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
6  print(digitos)
7
8  digitos = list(range(10))
9  print(digitos)
10 digitos[4] = 19
```

Comando `range`

O comando `range` tem três sintaxes possíveis:

- `range(fim)`: cria uma lista com valores entre 0 e $fim - 1$, de uma em uma unidade;

Comando `range`

O comando `range` tem três sintaxes possíveis:

- `range(fim)`: cria uma lista com valores entre 0 e $fim - 1$, de uma em uma unidade;
- `range(ini, fim)`: cria uma lista com valores entre ini e $fim - 1$, de uma em uma unidade;

Comando `range`

O comando `range` tem três sintaxes possíveis:

- `range(fim)`: cria uma lista com valores entre 0 e $fim - 1$, de uma em uma unidade;
- `range(ini, fim)`: cria uma lista com valores entre ini e $fim - 1$, de uma em uma unidade;
- `range(ini, fim, passo)`: cria uma lista com valores entre ini e $fim - 1$, começando em ini e seguindo a cada $passo$ unidades.

Operações em listas/vetores

- Cada valor armazenado em uma lista é identificado por um **índice**.

Operações em listas/vetores

- Cada valor armazenado em uma lista é identificado por um **índice**.
- Por isso, dizemos que elas são estruturas **indexadas**.

Operações em listas/vetores

- Cada valor armazenado em uma lista é identificado por um **índice**.
- Por isso, dizemos que elas são estruturas **indexadas**.
- O primeiro elemento tem índice 0, o segundo índice 1, e o i-ésimo tem índice $i - 1$.

Operações em listas/vetores

- Cada valor armazenado em uma lista é identificado por um **índice**.
- Por isso, dizemos que elas são estruturas **indexadas**.
- O primeiro elemento tem índice 0, o segundo índice 1, e o i-ésimo tem índice $i - 1$.
- A função `len` devolve o tamanho de uma lista que lhe é passada por parâmetro.

Operações em listas/vetores

- Cada valor armazenado em uma lista é identificado por um **índice**.
- Por isso, dizemos que elas são estruturas **indexadas**.
- O primeiro elemento tem índice 0, o segundo índice 1, e o i-ésimo tem índice $i - 1$.
- A função `len` devolve o tamanho de uma lista que lhe é passada por parâmetro.
- O operador `in` serve para verificar se um valor está contido em uma lista.

Criando vetores

Temos duas formas de criar listas em Python:

```
1  n = int(input())    # ou n = 100
2  vetor = [0] * n
3  for i in range(n):
4      vetor[i] = int(input())
```

```
1  n = int(input())    # ou n = 100
2  vetor = []
3  for i in range(n):
4      num = int(input())
5      vetor.append(num)
```

Percorrendo vetores

Temos algumas formas de percorrer as listas em Python:

```
1 for i in range(n):
2     print(vetor[i])
```

```
1 for i in range(len(vet)):
2     print(vetor[i])
```

```
1 for elem in vet:
2     print(elem)
```

Exemplos

Exemplos

Leia dois vetores de dimensão n, com $n \leq 100$, e compute o produto interno (produto escalar destes).

Exemplos

Leia dois vetores de dimensão n, com $n \leq 100$, e compute o produto interno (produto escalar destes).

```
1  n = int(input())
2  vet1 = [0] * n
3  vet2 = [0] * n
4
5  for i in range(n):
6      vet1[i] = float(input())
7  for i in range(n):
8      vet2[i] = float(input())
9
10 prod_escalar = 0
11 for i in range(n):
12     prod_escalar = prod_escalar + (vet1[i] * vet2[i])
13
14 print(prod_escalar)
```

Exemplos

Leia dois vetores de dimensão n, com $n \leq 100$, e compute o produto interno (produto escalar destes).

Exemplos

Leia dois vetores de dimensão n, com $n \leq 100$, e compute o produto interno (produto escalar destes).

```
1  n = int(input())
2  vet1 = []
3  vet2 = []
4
5  for i in range(n):
6      vet1.append(float(input()))
7  for i in range(n):
8      vet2.append(float(input()))
9
10 prod_escalar = 0
11 for i in range(n):
12     prod_escalar = prod_escalar + (vet1[i] * vet2[i])
13
14 print(prod_escalar)
```

Operações básicas

Busca

Dado um vetor com n elementos **distintos** e um elemento k, descubra se e onde k está armazenado no vetor.

Busca

Dado um vetor com n elementos **distintos** e um elemento k, descubra se e onde k está armazenado no vetor.

```
1  def busca(V: [int], tam: int, chave: int) -> int:
2      for i in range(tam):
3          if V[i] == chave:
4              return i
5      return -1
```

Busca

Dado um vetor com n elementos **distintos** e um elemento k, descubra se e onde k está armazenado no vetor.

```
1  def busca(V: [int], chave: int) -> int:
2      for i in range(len(V)):
3          if V[i] == chave:
4              return i
5      return -1
```

Busca

Dado um vetor com n elementos **distintos** e um elemento k, descubra se e onde k está armazenado no vetor.

```
1  def Busca(V: [int], chave: int) -> int:
2      if chave in V:
3          return V.index(chave)
4      return -1
```

Busca

Dado um vetor com n elementos **distintos** e um elemento k, descubra se k está armazenado no vetor.

```
1  def busca(V: [int], chave: int) -> int:
2      return chave in V
```

Inserção

Dado um vetor V com capacidade m e com n elementos **distintos** armazenados, e dado um elemento x, insira x em V.

Inserção

Dado um vetor V com capacidade m e com n elementos **distintos** armazenados, e dado um elemento x, insira x em V.

```
1  def insere(V: [int], capac: int, tam: int, chave: int) -> int:
2      if tam < capac:
3          i = busca(V, tam, chave)
4          if i == -1:
5              V[tam] = chave
6              tam = tam+1
7      return tam
```

Inserção

Dado um vetor V com n elementos **distintos** e dado um elemento x, insira x em V.

```
1  def Insere(V: [int], chave: int) -> None:
2      if chave not in V:
3          V.append(chave)
```

Inserção

Dado um vetor V com n elementos **distintos** e dado um elemento x, insira x em V.

```
1  def Insere(V: [int], chave: int) -> None:
2      if chave not in V:
3          V.insert(len(V), chave)
```

Remoção

Dado um vetor V com n elementos **distintos** armazenados e dado um elemento x, remova x de V.

Remoção

Dado um vetor V com n elementos **distintos** armazenados e dado um elemento x, remova x de V.

```
1  def remove(V: [int], tam: int, chave: int) -> int:
2      i = busca(V, tam, chave)
3      if i != -1:
4          V[i] = V[tam-1]
5          tam = tam-1
6      return tam
```

Remoção

Dado um vetor V com n elementos **distintos** armazenados e dado um elemento x, remova x de V.

```
1  def remove(V: [int], chave: int) -> None:
2      i = busca(V, chave)
3      if i != -1:
4          V[i] = V[len(V)-1]
```

Remoção

Dado um vetor V com n elementos **distintos** armazenados e dado um elemento x, remova x de V.

```
1  def remove(V: [int], chave: int) -> None:
2      i = busca(V, chave)
3      if i != -1:
4          V[i] = V[len(V)-1]
```

ERRADO!

Remoção

Dado um vetor V com n elementos **distintos** armazenados e dado um elemento x, remova x de V.

```
1  def remove(V: [int], chave: int) -> None:
2      i = busca(V, chave)
3      if i != -1:
4          V.pop(i)
```

Remoção

Dado um vetor V com n elementos **distintos** armazenados e dado um elemento x, remova x de V.

```
1  def Remove(V: [int], chave: int) -> int:
2      if chave in V:
3          V.remove(chave)
```

Resumo das operações

Se `V` é um vetor/lista:

- `x in V` devolve `True` se `x` está em alguma posição de `V` ou `False` caso contrário
- `len(V)` devolve quantos elementos existem em `V`
- `V.index(x)` devolve o índice da primeira ocorrência `x` em `V`, funcionando apenas quando `x in V`
- `V.append(x)` insere `x` ao final de `V`
- `V.insert(i, x)` insere `x` na posição `i` de `V`
- `V.remove(x)` remove a primeira ocorrência de `x` em `V`, funcionando apenas quando `x in V`
- `V.pop(i)` remove o elemento que estiver na posição `i` de `V`

Mais sobre leitura de vetores

Recebendo dados da entrada

"A entrada consiste em uma linha que contém um inteiro n, seguida de n linhas que contêm, cada uma, uma palavra."

Recebendo dados da entrada

"A entrada consiste em uma linha que contém um inteiro n, seguida de n linhas que contêm, cada uma, uma palavra."

```
1  n = int(input())
2  palavras = []
3  for i in range(n):
4      palavras.append(input())
```

Recebendo dados da entrada

"A entrada consiste em duas linhas. A primeira contém um inteiro n e a segunda contém n inteiros, separados por espaço."

Recebendo dados da entrada

"A entrada consiste em duas linhas. A primeira contém um inteiro n e a segunda contém n inteiros, separados por espaço."

```
1  n = int(input())
2  vet = input().split()
3  for i in range(n):
4      vet[i] = int(vet[i])
```

Recebendo dados da entrada

"A entrada consiste em uma linha que contém números reais separados por espaços."

Recebendo dados da entrada

"A entrada consiste em uma linha que contém números reais separados por espaços."

```
1  vet = input().split()
2  for i in range(len(vet)):
3      vet[i] = float(vet[i])
```

Pratique!

Exercício 1

Faça um programa para contar o número de inversões de um vetor. Dois elementos a e b formam uma inversão se $a > b$ e a aparece antes de b no vetor.

Exercício 2

Faça um programa que copia os valores pares de um vetor para outro vetor.

Exercício 3

Faça um programa que realize a apuração dos votos de uma eleição com 5 candidatos e n eleitores. Cada candidato tem um identificador único, que vai de 1 a 5.

Exercício 4

Escreva uma função que recebe dois vetores que representam conjuntos e verifica se o primeiro está contido no segundo.