SENAI-SP

Lógica de programação e algoritmo

AULA 05

SENAI
SÃO PAULO

Nessa Aula

SENAI-SP

- Listas
- Dicionários
- Exemplos
- Exercícios

Listas

- Tipo de variável que permite o armazenamento de vários valores, acessados por um índice.
- Pode conter zero ou mais elementos de um mesmo tipo ou de tipos diversos, inclusive outras listas.
- Possui tamanho igual à quantidade de elementos que ela contém.

Exemplo 01

- Lista de um prédio de seis andares.

```
# -*- coding: UTF-8 -*-

predio = ["térreo", "primeiro andar", "segundo andar", "terceiro andar", "quarto andar", "quinto andar"]
print (predio[0])
print (predio[1])
print (predio[2])
print (predio[3])
print (predio[4])
print (predio[5])
```

Onde:

- predio: nome da lista.
- índice: número entre colchetes ([0][1]...[5]).

Listas

- Lista vazia

```
L = []
```

- Lista com três elementos

```
L = [15, 8, 9]
```

- Acesso a uma lista

```
print (L[0])
print (L[1])
print (L[2])
```

OU

```
print (L)
```

- Modificação de uma lista

```
L[0] = 7
```

Exemplo 02

- Cálculo da média aritmética de 5 notas de um aluno.

```
# -*- coding: UTF-8 -*-

notas = [6, 7, 5, 8, 9]
soma = 0
x = 0
while x < 5:
    soma += notas[x]
    x += 1
print ("Média: %5.2f" % (soma / x))
```

Exemplo 03

- Cálculo da média aritmética com notas digitadas pelo usuário.

```
# -*- coding: UTF-8 -*-

notas = [0, 0, 0, 0, 0]
soma = 0
x = 0
while x < 5:
    notas[x] = float(input("Nota %d: " % x))
    soma += notas[x]
    x += 1
x = 0
while x < 5:
    print ("Nota %d: %6.2f" % (x, notas[x]))
    x += 1
print ("Média: %5.2f" % (soma / x))
```

Trabalhando com índices

- **Exemplo 04:** Programa que lê cinco números, armazena-os em uma lista e depois solicita que o usuário escolha um número a mostrar.

```
# -*- coding: UTF-8 -*-

numeros = [0, 0, 0, 0, 0]
x = 0
while x < 5:
    numeros[x] = int(input("Número %d: " % (x + 1)))
    x += 1
while True:
    escolhido = int(input("Que posição você quer imprimir (0 para sair): "))
    if escolhido == 0:
        break
    print ("Você escolheu o número: %d" % (numeros[escolhido - 1]))
```

Cópia de Listas

SENAI-SP

- **Exemplo 05:** Tentativa de copiar listas.

```
# -*- coding: UTF-8 -*-

L = [1, 2, 3, 4, 5]
V = L
print (L)
print (V)
V[0] = 6
print (L)
print (V)
```

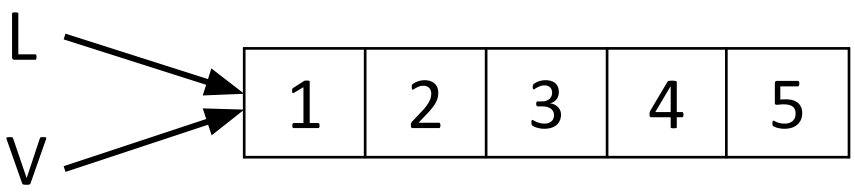
Cópia de Listas

- Uma lista em Python é um objeto.
- Atribuição de objeto:

  V = L

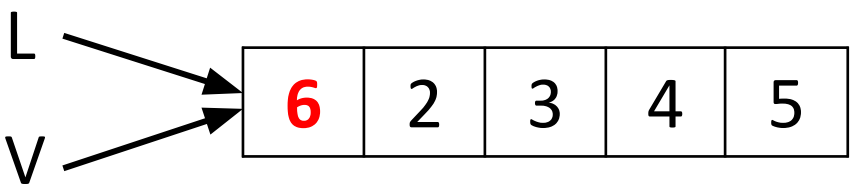
  Copia a mesma referência da lista na memória e não os seus dados.

  Duas variáveis (V e L) referenciam a mesma lista na memória.



Cópia de Listas

- Alteração de valor:
  V[0] = 6
  L[0] = 6



Cópia de Listas

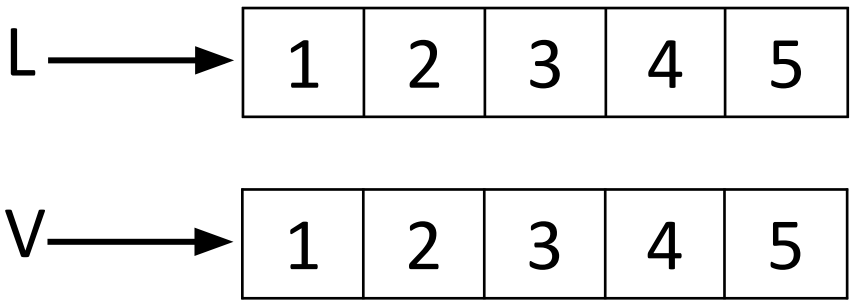
- **Exemplo 06:** Cópia de listas.

```
# -*- coding: UTF-8 -*-

L = [1, 2, 3, 4, 5]
V = L[:]
V[0] = 6
print (L)
print (V)
```

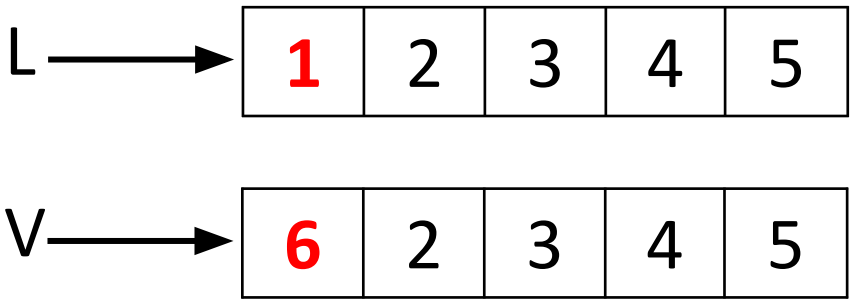
Cópia de Listas

- Uma nova cópia da lista:
  `V = L[:]`
- As listas, L e V, se referem a áreas diferentes na memória.



Cópia de Listas

- É possível alterar as listas, L e V, de forma independente.
  V[0] = 6



Fatiamento de listas

- **Exemplo 07:** Fatiamento de listas.

```
# -*- coding: UTF-8 -*-

L = [1, 2, 3, 4, 5]
print (L[0:5]) # [1, 2, 3, 4, 5]   da posição 0 até a posição 5, sem incluí-la
print (L[:5])  # [1, 2, 3, 4, 5]   do início até a posição 5, sem incluí-la
print (L[:-1]) # [1, 2, 3, 4]      do início até o último, sem incluí-lo
print (L[1:3]) # [2, 3]           da posição 1 até a posição 3, sem incluí-la
print (L[1:4]) # [2, 3, 4]        da posição 1 até a posição 4, sem incluí-la
print (L[3:])  # [4, 5]           da posição 3 até o final
print (L[:3])  # [1, 2, 3]        do início até a posição 3, sem incluí-la
print (L[-1])  # 5                último elemento
print (L[-2])  # 4                penúltimo elemento
```

Tamanho de listas

- Função **len()**, que retorna o número de elementos da lista.
- **Exemplo 08:** Tamanho de listas.

```
# -*- coding: UTF-8 -*-

L = [12, 9, 5]
print (len(L))
V = []
print (len(V))
```

Tamanho de listas

- **Exemplo 09:** Repetição com tamanho da lista usando a função **len()**.

```
# -*- coding: UTF-8 -*-

L = [1, 2, 3]
x = 0
while x < len(L):
    print (L[x])
    x += 1
```

Adição de elementos

- Método **append()** é usado para adicionar um elemento ao fim da lista.
- **Exemplo 10:** Adição de elementos à lista.

```
# -*- coding: UTF-8 -*-

L = []
L.append("a")
print (L)
L.append("b")
print (L)
L.append("c")
print (L)
print (len(L))
```

Adição de elementos

- **Exemplo 11:** Adição de elementos à lista, até que 0 seja digitado.

```
# -*- coding: UTF-8 -*-

L = []
while True:
    n = int(input("Digite um número (0 sai): "))
    if n == 0:
        break
    L.append(n)
x = 0
while x < len(L):
    print (L[x])
    x = x + 1
```

Adição de elementos

- Método **extend()** prolonga a lista, adicionando no fim todos os elementos de uma lista passada como argumento.
- **Exemplo 12:** Adição de elementos e listas.

```
# -*- coding: UTF-8 -*-

L = ["a"]
L.append("b")
print (L)
L.extend(["c"])
print (L)
L.append(["d", "e"])
print (L)
L.extend(["f", "g", "h"])
print (L)
```

Remoção de elementos da lista

- Instrução **del** é usada para remover elementos da lista.
- **Exemplo 13:** Remoção de elementos.

```
# -*- coding: UTF-8 -*-

L = ["a", "b", "c"]
del L[1]
print (L)
del L[0]
print (L)
```

Remoção de elementos da lista

- **Exemplo 14:** Remoção de fatias.

```
# -*- coding: UTF-8 -*-

L = list(range(101))
print (L)
del L[1:99]
print (L)
```

Usando for

- Estrutura de repetição projetada, especialmente, para percorrer listas.
- A cada repetição utiliza um elemento diferente da lista.
- **Exemplo 15:** Impressão de todos os elementos da lista com for.

```
# -*- coding: UTF-8 -*-

L = [8, 9, 15]
for e in L:
    print (e)
```

Ordenação de lista

- Método **sort()** pode ser utilizado para ordenar as listas de valores numéricos ou de strings.
- **Exemplo 16:** Ordenação de lista.

```
# -*- coding: UTF-8 -*-

numeros = [2, 5, 3.14, 1, -7]
numeros.sort()
print (numeros)
animais = ["macacos", "gatos", "cachorros", "ursos",
"elefantes"]
animais.sort()
print (animais)
```

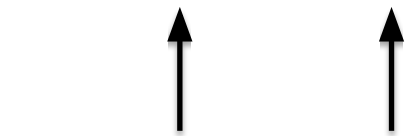
Dicionários

- Consistem em uma estrutura de dados similar às listas, mas com propriedades de acesso diferentes.
- São criados com a utilização de chaves ({}).
- Um dicionário é composto por um conjunto de chaves e valores.
- Utilizam suas chaves como índices e não números como as listas.

Dicionários

- Criação de um dicionário com preços de mercadorias.

```
tabela = {"Alface": 0.45,
          "Batata": 1.20,
          "Tomate": 2.30,
          "Feijão": 1.50}
```



Onde:

- tabela: nome do dicionário.

Dicionários

- **Exemplo 17:** Funcionamento do dicionário.

```
# -*- coding: UTF-8 -*-

tabela = {"Alface": 0.45,
          "Batata": 1.20,
          "Tomate": 2.30,
          "Feijão": 1.50}
print(tabela["Tomate"])
print(tabela)
tabela["Tomate"] = 2.50
print(tabela["Tomate"])
tabela["Cebola"] = 1.20
print(tabela)
```

Dicionários

- **Exemplo 18:** Acesso a uma chave inexistente.

```
# -*- coding: UTF-8 -*-

tabela = {"Alface": 0.45,
          "Batata": 1.20,
          "Tomate": 2.30,
          "Feijão": 1.50}
print(tabela["Manga"])
```

Dicionários

- **Exemplo 19:** Verificação da existência de uma chave.

```
# -*- coding: UTF-8 -*-

tabela = {"Alface": 0.45,
          "Batata": 1.20,
          "Tomate": 2.30,
          "Feijão": 1.50}
print("Manga" in tabela)
print("Batata" in tabela)
```

Dicionários

- **Exemplo 20:** Obtenção de uma lista de chaves e valores.

```
# -*- coding: UTF-8 -*-

tabela = {"Alface": 0.45,
          "Batata": 1.20,
          "Tomate": 2.30,
          "Feijão": 1.50}
print(tabela.keys())
print(tabela.values())
```

Dicionários

- **Exemplo 21:** Obtenção do preço com dicionário.

```
# -*- coding: UTF-8 -*-

tabela = {"Alface": 0.45,
          "Batata": 1.20,
          "Tomate": 2.30,
          "Feijão": 1.50}
while True:
    produto = input("Digite o nome do produto, fim para terminar: ")
    if produto == "fim":
        break
    if produto in tabela:
        print("Preço: %5.2f" % tabela[produto])
    else:
        print("Produto não encontrado!")
```

Dicionários

- **Exemplo 22:** Exclusão de uma associação do dicionário.

```
# -*- coding: UTF-8 -*-

tabela = {"Alface": 0.45,
          "Batata": 1.20,
          "Tomate": 2.30,
          "Feijão": 1.50}
print(tabela)
del tabela["Tomate"]
print(tabela)
```

Exercícios

1) Faça um programa que leia um vetor de 5 números inteiros e mostre-os na tela.

2) Faça um programa que leia um vetor de 10 números reais e mostre-os na ordem inversa.

3) Faça um programa que leia 4 notas, mostre as notas e a média na tela.

4) Faça um programa que leia um vetor de 10 caracteres minúsculos e diga quantas consoantes foram lidas.

5) Faça um programa que percorra duas listas e gere uma terceira com os elementos das duas primeiras.

Exercícios

6) A lista de temperaturas de Mons, na Bélgica, foi armazenada na lista T = [-10, -8, 0, 1, 2, 5, -2, -4]. Faça um programa que imprima a menor e a maior temperatura, assim como a temperatura média.

7) Faça um programa para selecionar os elementos de uma lista, de forma a copiá-los para outras duas listas. Nesse caso, considere que, inicialmente, os valores estão na lista V = [9, 8, 7, 12, 0, 13, 21], mas que devem ser copiados para a P, se forem pares; ou para a I, se forem ímpares.

Exercícios

8) Faça um programa que peça a idade e a altura de 5 pessoas, armazene cada informação no seu respectivo vetor. Imprima a idade e a altura na ordem inversa à ordem lida.

9) Faça um programa que peça as 4 notas de 10 alunos, calcule e armazene em um vetor a média de cada aluno, imprima o número de alunos com média maior ou igual a 7.0.

10) Faça um programa que imprima a lista L = [9, 8, 7, 12, 0, 13, 21], de forma ordenada.

Referências

- MENEZES, N. N. C. **Introdução à Programação com Python:** algoritmos e lógica de programação para iniciantes. 2ª ed. São Paulo: Novatec, 2014.

- PYTONSOFTWAREFOUNDATION. **Download the latest version for Windows:** Python 3.6.4. Disponível em: https://www.python.org/downloads/. Acesso em: 21 fev. 2018.

- Em homenagem à Janaine Arantes

Referências

Bibliografia Complementar

- MENEZES, N. N. C. **Introdução à Programação com Python:** algoritmos e lógica de programação para iniciantes. 2ª ed. São Paulo: Novatec, 2014.
- SWEIGART, Al. **Automatize Tarefas Maçantes com Python** - Programação Prática para Verdadeiros Iniciantes. São Paulo: Novatec, 2015.
- BORGES, L. E. **Python para Desenvolvedores**. 3ª ed. São Paulo: Novatec, 2014.