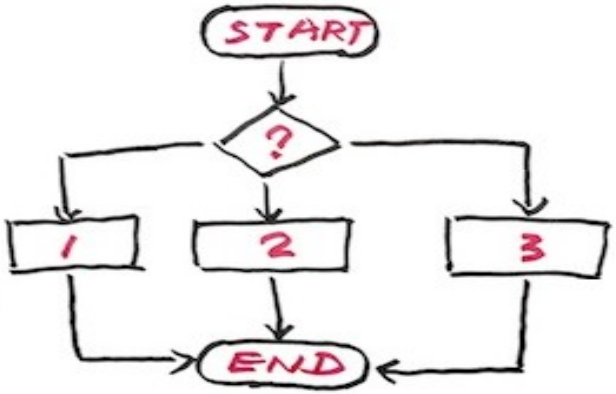
Introdução à Programação



Aula 12

Lista

Prof. Dr. Ivan José dos Reis Filho

Aula anterior

Strings

Aula de hoje

Listas

- Operações

Princípio das Listas

Estrutura de dados constituída por uma sequência ordenada e finita de itens

- As quais podem inclusive, ser outras listas, ditas sub-listas.
- podem ser modificadas com a inserção, exclusão e reordenamento dos itens

Toda lista pode conter 0, *n* ou infinitos elementos.

Contém 4 conceitos fundamentais

- Adição, alteração, exclusão e leitura.

Princípio das Listas

Diversas definições de listas

- Pilha (LIFO)
- Fila (FIFO)
- Conjunto
  - sem a existência de elementos repetidos.
- Array
  - cada elemento contido possui um número inteiro não negativo vinculado
- Tabela

Cada definição possui propriedades e funcionalidades que conferem à estrutura funcionalidades única

Coleções Python (arrays)

Existem quatro tipos de dados (arrays) em Python

- **List** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set** is a collection which is unordered, unchangeable*, and unindexed. No duplicate members.
- **Dictionary** is a collection which is ordered** and changeable. No duplicate members.

A Classe list()

A classe *list*

A classe **list** possui as funcionalidades e propriedades primitivas e pertencentes todas as listas.

Usa-se a estrutura ***list***.

- Declara-se de forma primitiva (automática);
- Python adicionou a sua sintaxe para representar as listas;
- Elementos que estiverem delimitados por um par de colchetes é considerado uma lista (python)

Declaração

A declaração explícita

```
1 #Declaração explícita
2 lst01 = list()
3
4 # Declaração explícita com mesmo resultado do que o código anterior
5 lst02 = list((1, 2, 3, 4, 5))
6 print(lst02)
```

A declaração implícita

```
1 #Declaração implícita
2 lst03 = [1, 2, 3, 4, 5]
3
4 print(lst03)
```

Elementos de uma lista

Todos elementos de uma lista é um objeto

- Conjunto de objetos Strings

```
1 #Conjunto de objetos strings
2
3 lst04 = ['valor1', 'valor2', 'Ivan', 'Corinthians', 'BiMundial']
4 print(lst04)
```

- Objetos mistos

```
1 #Conjunto de objetos mistos
2
3 lst05 = ['valor1', 123, 'Ivan', True, 2]
4 print(lst04)
```

Acessando elementos

No código a seguir, vamos definir uma lista, atribuí-la a uma referência e então, acessar alguns elementos.

```
1 #acessando elementos
2
3 lst06 = ['a', 'b', 'c', 'd', 'e']
4 print(lst06[0]) #acessa elemento cujo texto é igual a 'a'
5 print(lst06[2]) #acessa elemento cujo texto é igual a 'c'
6 print(lst06[-1]) #acessa elemento cujo texto é igual a 'e'
```

manipular vários elementos numa única expressão.

```
1 print(lst06[0] + lst06[1])
```

```
ab
```

Listas de objetos

Atribuir itens para a lista interna (lista de lista/objetos), por exemplo:

```
1 # Atribuir itens para a lista interna (lista de lista)
2
3 lst07 = [[1, 2, 3], ['a', 'b', 'c', 'd']]
```

O acesso a elementos das listas contidas em outras.

```
2 print(lst07[0])
3 print(lst07[0][2])
4 print(lst07[1])
5 print(lst07[1][3])
```

```
[1, 2, 3]
3
['a', 'b', 'c', 'd']
d
```

Definição

Listas

Listas em Python não definem o tipo de seus elementos, como ocorre com os Array em linguagem como Java, C e C++.

O funcionamento das listas é similar a um Array em C, C++, Pascal, numa lista só iremos ter objetos e por isso, o tipo do objeto é indiferente

Todo objeto, indiferente da sua declaração, descende de uma mesma estrutura implicita a linguagem.

Funções da classe *list( )*

Funções da classe *list*

Se tem 2 listas e deseja-se obter uma terceira que seja a junção, podemos utilizar o sinal de adição, por exemplo:

```
1 lst08 = [1, 2, 3]
2 lst09 = ['a', 'b', 'c', 'd']
3
4 lst10 = lst08 + lst09
5 print(lst10)
```

```
[1, 2, 3, 'a', 'b', 'c', 'd']
```

Concatenação entre uma lista e um objeto, será levantado uma mensagem de erro

```
1 lst11 = [1, 2, 3] + 4 # Erro
```

Atividade 01

Crie **duas** listas. **Uma** lista com nomes de 3 cidades e **outra** lista com nomes de 5 cidades. Faça a concatenação das duas listas e exiba na tela.

Atividade 01

Crie **duas** listas. **Uma** lista com nomes de 3 cidades e **outra** lista com nomes de 5 cidades. Faça a concatenação das duas listas e exiba na tela.

```
1 # ATIVIDADE 01
2
3 cidades1 = ['Frutal', 'Uberlândia', 'Uberaba']
4 cidades2 = ['Rio Preto', 'São Carlos', 'Barretos', 'Rio Claro', 'São Paulo']
5
6 cidades = cidades1 + cidades2
7
8 print(cidades)
```

```
['Frutal', 'Uberlândia', 'Uberaba', 'Rio Preto', 'São Carlos', 'Barretos', 'Rio Claro', 'São Paulo']
```

Mudando itens da lista

Mudar valor da lista

Para mudar o valor de um item com posição específica, use o índice:

```
1 frutas = ['Banana', 'maçã', 'morango']
2
3 frutas[1] = 'Melão'
4 print(frutas)
```

```
['Banana', 'Melão', 'morango']
```

Mudar valor da lista

Para mudar o valor de itens dentro de um range específico

```
1 frutas = ['Banana', 'maçã', 'morango', 'kiwi', 'manga']
2
3 frutas[1:3] = ['pera', 'melancia']
4 print(frutas)
```

```
['Banana', 'pera', 'melancia', 'kiwi', 'manga']
```

Mudar valor da lista

Se você inserir mais itens do que substituir, os novos itens serão inseridos onde você especificou e os itens restantes serão movidos de acordo:

```
1 frutas = ['Banana', 'maçã', 'morango', 'kiwi', 'manga']
2
3 frutas[1:2] = ['pera', 'melancia']
4 print(frutas)
```

```
['Banana', 'pera', 'melancia', 'morango', 'kiwi', 'manga']
```

Atividade 02

Crie uma lista com nomes de 5 carros. Substitua os carros das posições 2 e 3 por outros carros. Exiba todos elementos da lista na tela.

Atividade 02

Crie uma lista com nomes de 5 carros. Substitua os carros das posições 2 e 3 por outros carros. Exiba na tela.

```
1 #ATIVIDADE 02
2
3 carros = ['gol', 'fusca', 'onix', 'uno', 'voyage']
4 carros[2:4] = ['toro', 'virtus']
5
6 print(carros)
```

```
['gol', 'fusca', 'toro', 'virtus', 'voyage']
```

Add item na lista

Funções da classe *list*

A função append() adiciona o elemento definido como argumento ao fim da lista.

- Add um item no **final** da lista

```
1 frutas = ["apple", "banana", "cherry"]
2 frutas.append("orange")
3
4 print(frutas)
```

```
['apple', 'banana', 'cherry', 'orange']
```

Funções da classe *list*

Para inserir um item em um índice específico, use a função insert()

- Insere um item em uma **posição específica**

```
1 frutas = ["apple", "banana", "cherry"]
2 frutas.insert(2, "orange")
3
4 print(frutas)
```

```
['apple', 'banana', 'orange', 'cherry']
```

Funções da classe *list*

Para adicionar elementos de outra lista para uma lista atual, use a função extend()

- Add o elemento “tropical” para “estalista”

```
1 estaLista = ["apple", "banana", "cherry"]
2 tropical = ["mango", "pineapple", "papaya"]
3 estaLista.extend(tropical)
4
5 print(estaLista)
```

```
['apple', 'banana', 'cherry', 'mango', 'pineapple', 'papaya']
```

Atividade 03

Crie uma lista com nomes de 4 pessoas. Faça as inclusões abaixo:

- Add mais dois nomes no final da lista
- Add um nome na posição 1
- Use uma outra lista (2 nomes) e extenda para a lista inicial de nomes.

Atividade 03

Crie uma lista com nomes de 4 pessoas. Faça as inclusões abaixo:

- Add mais dois nomes no final da lista
- Add um nome na posição 1
- Use uma outra lista (2 nomes) e extenda para a lista inicial de nomes.

```
 3 nomes = ['Ivan', 'Jose', 'Joao', 'Jorge']
 4
 5 nomes.append('Antonio')
 6 nomes.append('Sebastiao')
 7 nomes.insert(1, 'Tiago')
 8
 9 nomes2 = ['Diego', 'Diogo']
10
11 nomes.extend(nomes2)
12 print(nomes)
```

```
['Ivan', 'Tiago', 'Jose', 'Joao', 'Jorge', 'Antonio', 'Sebastiao', 'Diego', 'Diogo']
```

Iteração de Listas

(Estruturas de repetição)

Iteração de lista

Você pode fazer um loop de items da lista usando o **for**

- Exibir todos itens da lista, “um por um”

```
1 frutas = ["apple", "banana", "cherry"]
2
3 for x in frutas:
4   print(x)
```

```
apple
banana
cherry
```

Iteração de lista

Você pode também usar a lista de items e referenciar o indice

- Use as funções **range()** e **len()** para criar a iteração

```
1 frutas = ["apple", "banana", "cherry"]
2
3 for x in range(len(frutas)):
4   print(frutas[x])
```

```
apple
banana
cherry
```

Iteração de lista

Iteração usando a estrutura **while()**

- Use a função **len()** para determinar o comprimento da lista, então comece em 0 e faça um loop pelos itens da lista consultando seus índices.

```
1 frutas = ["apple", "banana", "cherry"]
2 i = 0
3
4 while i < len(frutas):
5   print(frutas[i])
6   i = i + 1
```

Atividade 04

Crie uma lista com nome de 5 frutas. Use as 3 estruturas para exibir os elementos na tela:

- Estrutura for para lista
- Estrutura for usando o índice: range e len
- Estrutura while

Fatiando Listas

Fatiando Listas

Exibir apenas uma parte da lista

```
1 A = ['a', 'b', 'c', 'd', 'e']
2 B = A[1:3]
3
4 print(B)
```

```
['b', 'c']
```

Fatiando Listas

Intervalo de elementos

```
referência[ <inicio> : <fim> : <intervalo> ]
```

```
1 C = [x for x in range(20)]
2
3 D = C[::3]
4
5 print(D)
```

```
[0, 3, 6, 9, 12, 15, 18]
```

Ordenando listas

Ordenação de lista

Ordene lista alfanuméricamente

- Objetos de lista têm um método **sort()** que classificará a lista em ordem alfanumérica, crescente, por padrão:

```
1 Frutas = ["orange", "mango", "kiwi", "pineapple", "banana"]
2 Frutas.sort()
3
4 print(Frutas)
```

```
['banana', 'kiwi', 'mango', 'orange', 'pineapple']
```

Ordenação de lista

Ordene lista alfanuméricamente

- Ordenação de modo descendente

```
1 Frutas = ["orange", "mango", "kiwi", "pineapple", "banana"]
2 Frutas.sort(reverse=True)
3
4 print(Frutas)
```

```
['pineapple', 'orange', 'mango', 'kiwi', 'banana']
```

Ordenação de lista

Ordene lista numérica

- Lista com números

```
1 Numeros = [100, 50, 65, 82, 23]
2 Numeros.sort()
3
4 print(Numeros)
```

```
[23, 50, 65, 82, 100]
```

Ordenação de lista

Ordene lista numérica

- Ordenação de modo decrecente

```
1 Numeros = [100, 50, 65, 82, 23]
2 Numeros.sort(reverse=True)
3
4 print(Numeros)
```

```
[100, 82, 65, 50, 23]
```

Sequência inversa

A função reverse() inverte a ordem da lista mas sem ordena-la.

```
1 Numeros = [100, 50, 65, 82, 23]
2 Numeros.reverse()
3
4 print(Numeros)
```

```
[23, 82, 65, 50, 100]
```

Copiando listas

Cópia de lista

Cópia de lista

- Faça uma cópia usando o método copy()

```
1 frutas = ["apple", "banana", "cherry"]
2 frutinhas = frutas.copy()
3
4 print(frutinhas)
```

```
['apple', 'banana', 'cherry']
```

Excluindo Listas

Excluindo

A mesma notação utilizada para obtermos o elemento vinculado a determinado índice, pode ser utilizada, para alterarmos o elemento vinculado.

```
1 lista_A = ['a', 'b', 'c']
2 lista_B = ['d', 'e', 'f']
3
4 del(lista_A[2])
5 print(lista_A)
6
7 del(lista_B[0:2])
8 print(lista_B)
```

```
['a', 'b']
['f']
```

Excluindo todos elementos

A função clear() remove todos elementos da lista

```
1 lista_C = ['a', 'b', 'c', 'd', 'e']
2
3 lista_C.clear()
4
5 print(lista_C)
```

Métodos para listas

Métodos

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Exercícios

https://www.w3schools.com/python/exercise.asp?filename=exercise_lists1

Faça esse também

SeuNome_Aula12.py

1. Faça um Programa que recebe do usuário 5 números inteiros, insira em uma lista e depois mostre-os.
2. Faça um Programa que recebe 10 números reais do usuário, insira em uma lista, e mostre-os na ordem inversa.
3. Faça um Programa que leia 4 notas, insira em uma lista, mostre as notas e a média na tela.
4. Faça um Programa que recebe do usuário 10 caracteres, e diga quantas consoantes foram lidas. Imprima as consoantes.
5. Faça um Programa que leia 20 números inteiros e armazene-os em uma lista. Armazene os números pares no vetor PAR e os números IMPARES no vetor impar. Imprima os três vetores.

SeuNome_Aula12.py

6. Em uma competição de salto em distância cada atleta tem direito a cinco saltos.

O resultado do atleta será determinado pela média dos cinco valores restantes. Você deve fazer um programa que receba o nome e as cinco distâncias alcançadas pelo atleta em seus saltos e depois informe o nome, os saltos e a média dos saltos.

O programa deve inserir distâncias em uma lista ser encerrado quando não for informado o nome do atleta.

A saída do programa deve ser conforme o exemplo abaixo:

```
Atleta: Rodrigo Curvêllo

Primeiro Salto: 6.5 m
Segundo Salto: 6.1 m
Terceiro Salto: 6.2 m
Quarto Salto: 5.4 m
Quinto Salto: 5.3 m

Resultado final:
Atleta: Rodrigo Curvêllo
Saltos: 6.5 - 6.1 - 6.2 - 5.4 - 5.3
Média dos saltos: 5.9 m
```

SeuNome_Aula12.py

7. Uma empresa de pesquisas precisa tabular os resultados da seguinte enquete feita a um grande quantidade de organizações:

Você foi contratado para desenvolver um programa que leia o resultado da enquete e informe ao final o resultado da mesma. O programa deverá ler os valores até ser informado o valor 0, que encerra a entrada dos dados. Não deverão ser aceitos valores além dos válidos para o programa (0 a 6). Os valores referentes a cada uma das opções devem ser armazenados num vetor. Após os dados terem sido completamente informados, o programa deverá calcular a percentual de cada um dos concorrentes e informar o vencedor da enquete. O formato da saída foi dado pela empresa, e é o seguinte:

```
"Qual o melhor Sistema Operacional para uso em servidores?"

As possíveis respostas são:

1- Windows Server
2- Unix
3- Linux
4- Netware
5- Mac OS
6- Outro
```

```
Sistema Operacional     Votos   %
-------------------     -----   ---
Windows Server           1500   17%
Unix                     3500   40%
Linux                    3000   34%
Netware                   500    5%
Mac OS                    150    2%
Outro                     150    2%
-------------------     -----
Total                    8800

O Sistema Operacional mais votado foi o Unix, com 3500 votos, correspondendo a 40% dos votos.
```

SeuNome_Aula12.py

7. Faça um programa que simule um lançamento de dados. Lance o dado 100 vezes e armazene os resultados em um vetor. Depois, mostre quantas vezes cada valor foi conseguido. Dica: use uma lista de contadores(1-6) e uma função para gerar numeros aleatórios, simulando os lançamentos dos dados.

Obs: é preciso pesquisar sobre random()

Número aleatórios

List comprehension

(Estrutura de repetição para criação de lista)

List Comprehension

List comprehension oferece uma sintaxe mais curta quando você deseja criar uma nova lista com base nos valores de uma lista existente.

Exemplo:

- Com base em uma lista de frutas, você deseja uma nova lista, contendo apenas as frutas com a letra "a" no nome.
- Sem compreensão de lista, você terá que escrever uma instrução for com um teste condicional dentro:
- Add o elemento tropical para estalista

List Comprehension

Exemplo:

```
1 frutas = ["apple", "banana", "cherry", "kiwi", "mango"]
2 novalista = []
3
4 for x in frutas:
5   if "a" in x:
6     novalista.append(x)
7
8 print(novalista)
```

```
['apple', 'banana', 'mango']
```

List Comprehension

Com list comprehension você pode fazer isso em apenas uma linha de código

```
1 frutas = ["apple", "banana", "cherry", "kiwi", "mango"]
2
3 novalista = [x for x in frutas if "a" in x]
4
5 print(novalista)
```

List Comprehension

Sintaxe

novalista = [expression **for** item **in** iterable **if** condition == **True**]

novaLista = [x for x in fruits if x == "apple"]

O retorno é uma nova lista, deixando a lista mais velha inalterada

List Comprehension

Exemplo

- Apenas aceita itens na nova lista se não for ‘apple’

```
novaLista = [x for x in fruits if x != "apple"]
```

novalista = [expression **for** item **in** iterable **if** condition == **True**]

List Comprehension

Exemplo: Iterable

- You can use the range() function to create an iterable:

novaLista = [x for x in range(10)]

```
1 novaLista = [x for x in range(10)]
2
3 print(novaLista)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

List Comprehension

Veja mais em:

https://www.w3schools.com/python/python_lists_comprehension.asp

Atividades 05

5.1 - Crie uma lista usando comprehension com números de 1 à 10

5.2 - Crie uma lista usando comprehension com números de -50 a 50 com intervalor de 5 em 5.

5.3 – Crie uma lista com números pares e divisíveis por 3 de 1 a 1000.

Atividades 05

5.1 - Crie uma lista usando comprehension com números de 1 à 10

5.2 - Crie uma lista usando comprehension com números de -50 a 50 com intervalor de 5 em 5.

5.3 – Crie uma lista com números pares e divisíveis por 3 de 1 a 1000.

```python
#Atividades 05

lista1 = [x for x in range(11)]
lista2 = [x for x in range(-50, 50, 5)]
lista3 = [x for x in range(1000) if (x % 2 == 0) and (x % 3 == 0)]

print(lista1)
print(lista2)
print(lista3)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[-50, -45, -40, -35, -30, -25, -20, -15, -10, -5, 0, 5, 10, 15, 20, 25,
[0, 6, 12, 18, 24, 30, 36, 42, 48, 54, 60, 66, 72, 78, 84, 90, 96, 102,
```