

The background of the slide features a wide-angle photograph of a desert landscape at sunset or sunrise. The sky is a vibrant blue, transitioning from a warm orange near the horizon. In the foreground and middle ground, there are several large, layered rock formations with distinct horizontal sedimentary patterns. The rocks are a reddish-brown color, with some darker shadows and highlights from the low-angle sunlight. A few small green bushes are visible at the base of the rocks.

Introdução à Linguagem **Python**

Listas

Prof. Cláudio A. Fleury
Mai/2021

Conteúdo

- O que são Listas em Python?
- Listas em Python
- Operações com Listas
- Filtrando Listas em Python
- Lista Abrangente
 - Com Condição
 - Com Aninhamento
- Funções Incorporadas
- Métodos da Classe **list**
- Signos do Zodíaco
- Exercícios

O que são Listas em Python?

- 
- ACHOCOLATADO
 - AÇÚCAR
 - ADOÇANTE
 - ARROZ
 - ATUM
 - AZEITE
 - BATATA PALHA
 - BISCOITO
 - BOMBOM
 - CAFÉ
 - CREME DE LEITE
 - ERVILHA
 - FEIJÃO
 - FERMENTO
 - GELATINA
 - LEITE
 - LEITE CONDENSADO
 - MACARRÃO
 - SAL

- São estruturas de dados que apresentam uma sequência ordenada de elementos mutáveis
- Cada elemento da lista é chamado de **item**
- Assim como as strings, definidas por caracteres entre aspas, as **Listas** são definidas por itens separados por vírgulas e delimitados por colchetes []
- Exemplo:

```
lista = ['IFG', 2021, True, 6.863]
```

Operações com Listas

Criando listas

Acessando listas

Fatiando listas

Reatribuindo listas

Apagando itens

Listas Multidimensionais



Python

Listas

Concatenando listas

Operações em listas

Percorrendo uma lista

Lista abrangente

Funções incorporadas

Métodos incorporados

Criando Listas

- Para criar uma lista, indique os itens separados por vírgulas, delimitados por colchetes (syntax Python) e atribua a uma variável: `var = [item1, item2, item3]`
- Tipo de dado nativo (incorporado ao interpretador), do tipo contêiner¹ que pode conter diferentes tipos de valores:
`lista = ['segunda', 'terça', 'quarta', 4, 7.5, True]`
- Exemplo
 - Lista de nomes
 - Gerar uma lista com a quantidade de caracteres de cada nome

```
nomes = ['Ana', 'Olga',
'Roberta', 'Deise', 'Carla',
'Barbara', 'Catarina']

c = []      # lista de comprimentos

for nome in nomes:
    c.append(len(nome))

print(c)

[3, 4, 7, 5, 5, 7, 8]
```

```
c = [len(nome) for nome in nomes]
print(q)
```

Lista abrangente

¹ tipo de dado abstrato: classe ou estrutura de dados que é uma coleção de outros objetos

Criando Listas

- Exemplo

- Gerar uma lista com os possíveis pares de bebida e comida das listas ['água', 'chá', 'suco'] e ['ovos', 'carne', 'arroz'], respectivamente.

```
menu = []
for bebida in ['água', 'chá', 'suco']:
    for comida in ['ovos', 'carne', 'arroz']:
        menu.append([bebida,comida])
print(menu)
[['água', 'ovos'], ['água', 'carne'],
 ['água', 'arroz'], ['chá', 'ovos'],
 ['chá', 'carne'], ['chá', 'arroz'],
 ['suco', 'ovos'], ['suco', 'carne'],
 ['suco', 'arroz']]
```

```
menu = [[b,c] for b in ['água', 'chá', 'suco'] for c in ['ovos', 'carne', 'arroz']]
print(menu)
```

Lista abrangente

Acessando Listas

- Para acessar uma lista Python no todo, basta acessar a variável lista
- Exemplo

- Gerar uma lista com os possíveis pares de bebida e comida das listas ['água', 'chá', 'suco'] e ['ovos', 'carne', 'arroz'], respectivamente.

```
menu = [ ]
for bebida in ['água', 'chá', 'suco']:
    for comida in ['ovos', 'carne', 'arroz']:
        menu.append([bebida, comida])
menu
[[['água', 'ovos'], ['água', 'carne'],
  ['água', 'arroz'], ['chá', 'ovos'],
  ['chá', 'carne'], ['chá', 'arroz'],
  ['suco', 'ovos'], ['suco', 'carne'],
  ['suco', 'arroz']]
```

```
menu = [[b,c] for b in ['água', 'chá', 'suco']
        for c in ['ovos', 'carne', 'arroz'] ]
print(menu)
```

Lista abrangente

Acessando Listas

- Para acessar um único elemento, use o índice do item entre colchetes após o nome da lista
- A indexação começa em 0
- Exemplo

```
linguas = ['Inglês', 'Espanhol', 'Mandarim',  
          'Hindi', 'Japonês', 'Português']  
linguas[1]
```

```
'Espanhol'
```

Fatiando Listas

- Para acessar parte de uma lista use o operador de fatiamento `[:]`
- Os índices podem ser positivos e negativos
- Índice negativo significa acesso do final para o começo da lista
- Exemplo

```
cores = ['Azul', 'Amarela', 'Verde',
'Roxa', 'Vermelha', 'Branca']
cores[1:3], cores[3:4]
cores[3:]
cores[:-2]

(['Amarela', 'Verde'], ['Roxa'])
['Roxa', 'Vermelha', 'Branca']
['Azul', 'Amarela', 'Verde', 'Roxa']
```

Reatribuindo Listas

- Listas são mutáveis
- É possível reatribuir seus itens ou reatribuir uma nova lista
- Exemplos

```
cores = ['bronze', 'ouro', 'prata', 'lilás']
cores[2:] = ['branca', 'preta']
cores
cores[1] = 'vermelha'
cores

['bronze', 'ouro', 'branca', 'preta']
['bronze', 'vermelha', 'branca', 'preta']
```

Apagando Listas

- Para excluir uma lista use o comando **del** e o nome da lista
- Use o operador de fatiamento para excluir um ou mais itens
- Exemplos

```
cores = ['bronze', 'ouro', 'prata', 'lilás']
del cores[1]
cores
del cores
cores

['bronze', 'prata', 'lilás']
NameError: name 'cores' is not defined
```

Listas Multidimensionais

- Lista de listas
- Exemplos

```
mercado = [['farinha', 'feijão', 'arroz'],
['óleo', 'carne']]  
mercado[1]  
a = [[[1,2],[3,4],5],[6,7]]  
a  
a[0][1][1]  
  
['óleo', 'carne']  
[[[1, 2], [3, 4], 5], [6, 7]]  
4
```

Concatenando Listas

- Ajuntamento de duas ou mais listas, com suas ordens preservadas
- O operador de concatenação usado com strings também funciona com as listas
- Exemplo

```
a,b = [3,1,2], [5,4,6]
```

```
a+b
```

```
[3, 1, 2, 5, 4, 6]
```

Operações com Listas

- Multiplicar uma lista por um inteiro faz cópias de seus itens, preservando a ordem
- Verificar se um dado elemento pertence ou não à lista
- Exemplo

```
a = [3,1,2]
a*3
3 in a, 15 in a
3 not in a, 15 not in a

[3, 1, 2, 3, 1, 2, 3, 1, 2]
(True, False)
(False, True)
```

Percorrendo Listas

- A lista pode ser percorrida com um laço **for**
- Exemplo

```
# percorrendo a lista [1,2,3,4,5]
for i in [1,2,3,4,5]:
    if i % 2 == 0:
        print(f"{i} é par\n")
```

2 é par

4 é par

Filtrando Listas

- Exemplo

- Crie uma lista com números inteiros de 0 a 9, cujo quadrado é maior que 5 e menor que 50.

```
q = []          # lista de quadrados
for x in range(10):
    quad = x ** 2
    if quad > 5 and quad < 50:
        q.append(quad)
print(q)
```

```
[9, 16, 25, 36, 49]
```

Lista abrangente

```
q = [x**2 for x in range(10) if x**2 < 50 and x**2 > 5]
print(q)
```

Filtrando Listas

- Exemplo

- A partir de uma lista existente de nomes, crie uma nova lista contendo apenas os nomes que começam com a letra B ou C.

```
nomes = ['Ana', 'Olga', 'Roberta', 'Deise',
         'Carla', 'Barbara', 'Catarina']
nomesBC = []
for nome in nomes:
    if nome.startswith('B') or nome[0] == 'C':
        nomesBC.append(nome)
print(nomesBC)
```

```
['Carla', 'Barbara', 'Catarina']
```

Lista abrangente

```
nomesBC = [n for n in nomes if n[0]=='B' or n[0]=='C']:
print(menu)
```

Lista Abrangente

- Construção sintática disponível em Python para a criação de novas listas, baseadas em outras já existentes, de uma forma concisa e inspirada na teoria dos conjuntos
- Outras linguagens também implementam listas abrangentes: *Clojure*, *Common LISP*, *Haskell*, porém a mais conhecida delas é o comando SELECT da linguagem *SQL*
- Sintaxe:
[«*expressão*» for «*variáveis*» in «*iterável*» if «*condição*»]

Lista Abrangente

- **Teoria dos Conjuntos**

- Exemplo na notação de definição de conjunto:

$$S = \{2 \cdot x \mid x \in \mathbb{N}, x^2 > 3\}$$

- Lido como:

"**S** é o conjunto de todos os números '2 vezes **x**' onde **x** é um item no conjunto dos números naturais **N**, pelo qual **x** ao quadrado é maior que 3"

- **x** é a variável que representa os membros de um conjunto de entrada
 - **N** representa o conjunto de entrada, que neste exemplo é o conjunto dos números naturais
 - **x² > 3** é uma função predicado que age como um filtro nos membros do conjunto de entrada
 - **2.x** é uma função de saída que produz membros do novo conjunto de membros do conjunto de entrada que satisfaz a função predicado
 - **{ }** chaves funcionam como delimitadores da expressão
 - **|** , a barra vertical e a vírgula são separadores

Lista Abrangente

- Exemplo
 - Criação de uma lista com os números inteiros entre zero e nove

Tradicional

```
nint = []          # Lista de números inteiros (vazia)
for i in range(10):
    nint.append(i) # anexando elementos à lista 'nint'
print(nint)        # exibindo a lista 'nint'
```

Com lista
abrangente

```
nint = [i for i in range(10)]
print(nint)
```

Saída dos dois
trechos...

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

Lista Abrangente com Condição

- Exemplo

- É possível usar uma condição de validação dos elementos da lista a ser criada

Tradicional

```
nint_par = []      # Lista de números inteiros (vazia)
for i in range(10):
    if not i % 2:
        nint_par.append(i)
print(nint_par)
```

Com lista
abrangente

```
nint_par = [i for i in range(10) if not i % 2]
print(nint_par)
```

Saída dos dois
trechos...

```
[0, 2, 4, 6, 8]
```

Lista Abrangente com Aninhamento

- Exemplo
 - Podemos gerar listas dentro de uma lista

Tradicional

```
from random import randint
mat = [[],[],[],[],[]]
for i in range(5):
    for j in range(5):
        mat[i].append(randint(0,j+1))
print(mat)
```

Com lista
abrangente

```
from random import randint
mat = [[randint(0, j + 1) for i in range(5)] for j in range(5)]
print(mat)
```

Saída dos dois
trechos...

```
[[0, 0, 1, 0, 1],
 [1, 0, 0, 1, 2],
 [0, 1, 0, 2, 2],
 [2, 4, 4, 3, 3],
 [5, 5, 5, 2, 1]]
```

Funções Incorporadas

```
>>> par = [2*i for i in range(1,11)]  -> [2,4,6,8,10,12,14,16,18,20]
```

- **len()** - calcula o comprimento da lista

```
>>> len(par)           -> 10
```

- **max()** - retorna o item da lista de valor mais alto

```
>>> max(par)           -> 20
```

- **min()** - retorna o item da lista de valor mais baixo

```
>>> min(par)           -> 2
```

- **sum()** - retorna a soma de todos os elementos da lista

```
>>> sum(par)           -> 110
```

Funções Incorporadas

- **sorted()** - retorna uma versão classificada da lista, sem alterar a original

```
>>> sorted([3,1,2])  ->      [1, 2, 3]
```

- **list()** - converte um tipo de dados diferente em uma lista

```
>>> list("abc")      ->      ['a', 'b', 'c']
```

- **any()** - retorna **True** se tem pelo menos um item na lista não **False**, e **False** para iterável vazio

```
>>> any(['', '', '1']) ->      True
```

- **all()** - retorna **True** se todos os itens da lista têm um valor **True**, e **True** para iterável vazio

```
>>> all(['', '', '1']) ->      False
```

Métodos da Classe `list`

- **append()** - adiciona um item ao final da lista

```
>>> [3,1,2].append(4)           -> [3, 1, 2, 4]
```

- **insert()** - insere um item em uma posição especificada

```
>>> [3,1,2,4].insert(3,9)      -> [3, 1, 2, 9, 4]
```

- **remove()** - remove a primeira instância do item da lista

```
>>> [3,1,2,9,4].remove(2)      -> [3, 1, 9, 4]
```

- **pop()** - remove o elemento no índice especificado e o mostra

```
>>> [3,1,9,4].pop(2)          -> 9  
                                [3, 1, 4]
```

Métodos da Classe `list`

- **clear()** - esvazia a lista

```
>>> a=[3,1,2]; a.clear(); a      ->      []
```

- **index()** - retorna o primeiro índice correspondente ao item especificado

```
>>> [3,1,2,4].index(4)        ->      3
```

- **count()** - retorna a contagem do item especificado

```
>>> [4,1,2,9,4].count(4)      ->      2
```

- **reverse()** - inverte a ordem dos elementos na lista

```
>>> a=[4,1,2,9,4]; a.reverse(); a ->      [4, 9, 2, 1, 4]
```

- **sort()** - classifica a lista em ordem crescente

```
>>> a=[3,1,9,4]; a.sort(); a      ->      [1, 3, 4, 9]
```

List \longleftrightarrow String

- **Lista \rightarrow String:** método **join()** da classe *string*

```
meses = ['jan', 'fev',
          'mar', 'abr', 'mai', 'jun']
sep = ';'
strjuntos = sep.join(meses)
print(strjuntos)
```

jan;fev;mar;abr;mai;jun

- **String \rightarrow Lista:** método **split()** da classe *string*

```
meses2 = strjuntos.split(sep)
print(meses2)
```

['jan', 'fev', 'mar', 'abr',
 'mai', 'jun']

Signos do Zodíaco

Script para
mostrar o
signo do
zodíaco:

Abr/2021

main.py

```
1  ''' Informa o signo do Zodíaco a partir da data de nascimento '''
2
3  def main():
4      # Signos do zodíaco e suas faixas de datas
5      signosZodiaco = [
6          ("Capricórnio",0,1,19,1),
7          ("Aquário",20,1,18,2),
8          ("Peixes",19,2,20,3),
9          ("Áries",21,3,19,4),
10         ("Touro",20,4,20,5),
11         ("Gêmeos",21,5,20,6),
12         ("Câncer",21,6,22,7),
13         ("Leão",23,7,22,8),
14         ("Virgem",23,8,22,9),
15         ("Libra",23,9,22,10),
16         ("Escorpião",23,10,21,11),
17         ("Sagitário",22,11,21,12),
18         ("Capricórnio",22,12,31,12)
19     ]
20     dnasc = input("Data de Nascimento (DD/MM): ")
21     diastr, messtr = dnasc.split('/')
22     dia, mes = int(diastr), int(messtr)
23     for signo, dia0, mes0, dia1, mes1 in signosZodiaco:
24         if (mes0,dia0) <= (mes,dia) <= (mes1,dia1):
25             print("Signo do Zodíaco: ", signo)
26             break
27
28 main()
```

Console

Shell

```
Data de Nascimento (DD/MM): 6/2
Signo do Zodíaco: Aquário
>
```

Resumo

👉 Vimos:

- 👉 Como declarar e acessar uma lista
- 👉 Como fatiar listas (tal qual com strings)
- 👉 Como excluir e reatribuir elementos ou uma lista inteira
- 👉 Como usar listas multidimensionais
- 👉 Como criar listas abrangentes
- 👉 Como iterar em listas, concatená-las
- 👉 Como fazer operações com listas
- 👉 Algumas funções e métodos da classe **list**

Exercícios

- 1) Criar uma lista com 10 números reais e depois mostrar a lista em ordem inversa.
- 2) Criar uma lista com 4 notas (números reais), em seguida exibir as notas e a média aritmética delas.
- 3) Criar uma lista com as idades de 10 pessoas e exibir a maior e menor idades.
- 4) Criar uma lista de 20 números inteiros e separar os números pares numa lista PAR e os números ímpares noutra lista IMPAR. Exibir as listas PAR e IMPAR. Nota: use listas abrangentes.

Exercícios

- 5) Fazer um *script* que receba a temperatura média de cada mês do ano e as guarde em uma lista. Em seguida, calcule a média anual das temperaturas e mostre a média calculada juntamente com todas as temperaturas acima da média anual, e em que mês (por extenso: Janeiro, Fevereiro, ...) elas ocorreram.
- 6) Fazer um *script* que leia uma lista com valores informados pelo usuário e gere uma nova lista cujos itens consistam da metade dos valores lidos.
- 7) Fazer um *script* que crie uma matriz $N \times N$ com valores reais escolhidos aleatoriamente. O valor de N deve ser informado pelo usuário.

Exercícios

8) Utilizando listas faça um *script* que apresente 5 perguntas para uma pessoa sobre um crime:

- "Telefonou para a vítima?"
- "Esteve no local do crime?"
- "Mora perto da vítima?"
- "Devia para a vítima?"
- "Já trabalhou com a vítima?"

Exiba a classificação sobre a participação da pessoa no crime.

Se a pessoa responder positivamente a 2 questões ela deve ser classificada como "Suspeita"; entre 3 e 4 como "Cúmplice" e; 5 como "Assassino". Caso contrário, ele será classificado como "Inocente".

Exercícios

- 9) Uma empresa de pesquisas precisa tabular os resultados de uma enquete feita com algumas organizações: "Qual o melhor Sistema Operacional para uso em servidores?" As possíveis respostas foram:
1-Windows XP 2-Unix 3-Linux 4-Netware 5-MacOS 6-Outros

Fazer um *script* em Python para ler as respostas da enquete (*flag* 0 - encerra a entrada dos dados) e informar o resultado da mesma. Use lista(s). Calcular o percentual de cada S.O. e informar o vencedor da enquete, o total de votos do vencedor e o total de organizações.

Fim