PPT05 – Fundamentos de Python
(UFCD 10793)

Sandra Liliana Meira de Oliveira

Financiado pela União Europeia NextGenerationEU

Listas

Listas

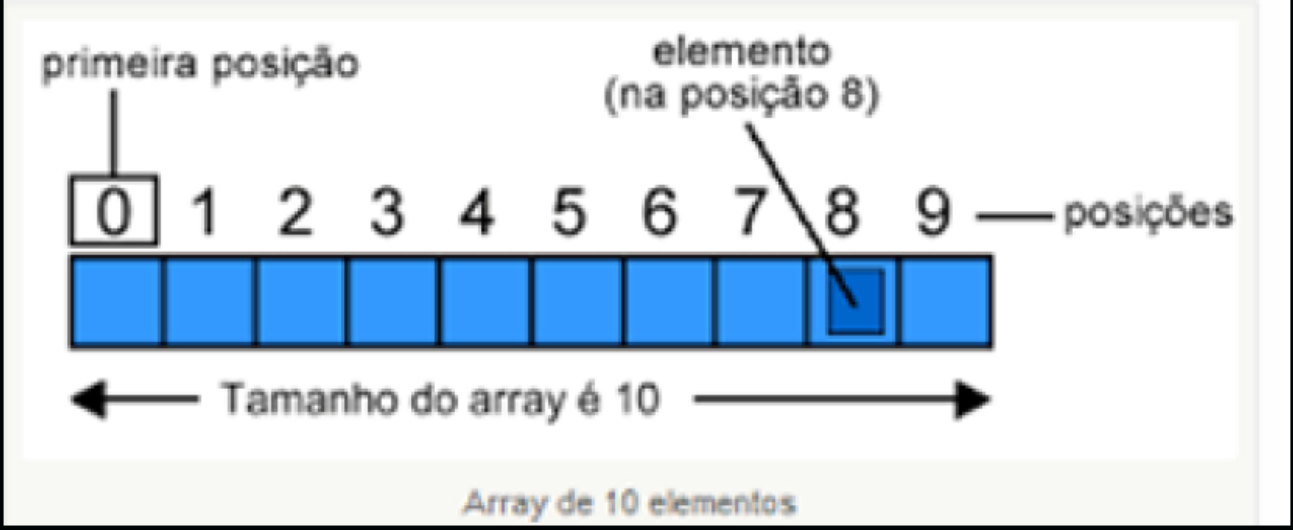
As listas são usadas para armazenar vários itens em uma única variável.

```
nomelista = ["preto", "branco", "cinza","amarelo","verde", "azul", "vermelho"]
#Definir lista através de construtor
outralista = list(("verde", "azul", "vermelho"))
print(nomelista)
print(outralista)
```

As listas podem ser declaradas com parênteses reto ou através do construtor list().

Listas

- Os itens da lista podem ser alterados e aceita valores duplicados.
- Os itens da lista são indexados, o primeiro item tem índice [0]



Listas

LISTAS - Acesso

```
fruta=["pessego", "uva", "figo", "pera", "manga", "cereja"]

print(fruta[2]) #Faz um print da posição 3 da lista

print(fruta[2:4]) #Faz um print da posição 3 e 4 da lista

print(fruta[2:]) #Faz um print da posição 3 até ao fim da lista

print(fruta[:3]) #Faz um print do inicio até à posição 3 lista

print(fruta[-1]) #Faz um print da ultima posição da lista
```

Listas

LISTAS - Acesso

```
fruta=["pessego", "uva", "figo", "pera", "manga", "cereja"]

print(fruta[2]) #Faz um print da posição 3 da lista

print(fruta[2:4]) #Faz um print da posição 3 e 4 da lista

print(fruta[2:]) #Faz um print da posição 3 até ao fim da lista

print(fruta[:3]) #Faz um print do inicio até à posição 3 lista

print(fruta[-1]) #Faz um print da ultima posição da lista
```

Listas

LISTAS – Alterar / Acrescentar valores

```
fruta=["pessego", "uva", "figo", "pera", "manga", "cereja"]

fruta[2]="laranja"# Altera a posição 3 (figo) para "laranja"

fruta[2:4]=["laranja", "roma"] #Altera as posições 3 e 4 para "laranja" e "roma"
```

```
fruta=["pessego", "uva", "figo", "pera", "manga", "cereja"]
legumes=["cenoura", "broculos", "cebolas"]

fruta.insert(2, "laranja") #insere na posição 3 o valor laranja

fruta.append("ananas") # Acrescenta no final da lista o valor "ananas"

fruta.extend(legumes) #adiciona a lista legumes à lista fruta
```

Listas

LISTAS – Remover valores

```
fruta=["pessego", "uva", "figo", "pera", "manga", "cereja"]

fruta.remove("cereja") #Remove o "valor "cereja" da lista

fruta.pop(1) #Remove a posição 2 da lista

fruta.pop() #Remove a ultima posição da lista

del fruta[1] #Remove a posição 2 da lista
del fruta #Apaga a lista inteira

fruta.clear() #Limpa a lista
```

Listas

LISTAS – Analise de listas

```
lista =[1, 2, 3, 4, 5, 6, 7]

print(len(lista)) # Print do tamanho da lista

# Estes comandos só funcionam se os valores da lista
# forem inteiros ou floats
print (sum(lista)) # Print da soma da lista
print(min(lista)) # Print do valor minimo da lista
print(max(lista)) # Print do valor máximo da lista
```

Listas

LISTAS – Loops

```
fruta=["pessego", "uva", "figo", "pera", "manga", "cereja"]

for x in fruta: #x toma o valor de cada item da lista
    print(x)

for i in range (len(fruta)): #i toma o valor do indice da lista
    print(fruta[i])

# verificar se existe um elemento na lista

if "uva" in fruta:
    print("A lista contem a a fruta uva!!")
```

Listas

LISTAS – Classificar

```
fruta=["pessego", "uva", "figo", "pera", "manga", "cereja"]

fruta.sort() #Ordena a lista em ordem alfabética

fruta.sort(reverse = True) #Ordena a lista de forma decrescente
fruta.sort(key=str.lower) #Ordena a lista com letras em minusculas

fruta.reverse() # inverte a ordem dos itens da lista
```

Listas - Síntese

As listas podem ser utilizadas para armazenar valores. Esses valores podem ser strings, tuplos, outras listas, dicionários...

Lista Vazia:

```
lista = []
```

Ou ainda

```
lista = list()
```

LISTA COM ALGUNS VALORES

lista1 = [10, 20, 30, 40]

Ou ainda

lista2 = ['Hello World!', 10, 20, 30, [1, 2, 3], (45, 50)]

Observe que a lista2 armazena uma string ('Hello World!'), números inteiros, outra lista e um tuplo

ACEDER A VALORES DE UMA LISTA

lista3 = ['A', 'B', 'C', 'D']

A string 'A' possui índice 0, já a string 'B' índice 1 e assim sucessivamente até a string 'D', que possui índice 3.

Podemos verificar o tipo de dado da nossa lista3 com a função built-in type().

type(lista3)

Observe que a lista2 armazena uma string ('Hello World!'), números inteiros, outra lista e um tuplo

ACEDER A VALORES DE UMA LISTA

lista = ['A', 'B', 'C’]

lista[0] devolve ‘A’
lista[1] devolve ‘B’
lista[2] devolve ‘C’

Se executarmos lista[3] teremos *IndexError*: *list index out of range*, pois não temos nenhum valor a ocupar este índice.

MODIFICAR OS VALORES DE UMA LISTA

a = [10, 11, 12, 13]

a[0] = 0 muda o elemento de indíce 0, ou seja o 10
a[1] = 0 muda o elemento de indíce 1, ou seja o 11
a[2] = 0 muda o elemento de indíce 2, ou seja o 12
a[3] = 0 muda o elemento de indíce 3, ou seja o 13

Resultado Final: [0, 0, 0, 0]

Outra forma de indexação

O último elemento de uma lista possui índice -1..

a[-1] = 0 muda o elemento de indíce 3, ou seja o 13

INDEXAÇÃO NEGATIVA

a = [10, 11, 12, 13]

a[-4] = 0 muda o elemento de indíce 0, ou seja o 10
a[-3] = 0 muda o elemento de indíce 1, ou seja o 11
a[-2] = 0 muda o elemento de indíce 2, ou seja o 12
a[-1] = 0 muda o elemento de indíce 3, ou seja o 13

Resultado Final: [0, 0, 0, 0]

Outra forma de indexação
podemos aceder aos elementos de uma lista no sentido contrário (indexação negativa), do fim para o início

TAMANHO DE UMA LISTA

```
a = [10, 11, 12, 13]

len(a)
```

Para verificar o tamanho de uma lista usamos a função integrada len().

MÁXIMO, MÍNIMO, SOMA E TAMANHO DE UMA LISTA

```
a = [10, 11, 12, 13]

print(f'Valor máximo: {max(num)}’)
print(f'Valor mínimo: {min(num)}’)
print(f'Número de elementos: {len(num)}’)
print(f'Soma dos elementos: {sum(num)}’)
```

UNIR LISTAS

```
lista1 = ['a', 'b', 'c’]
lista2 = ['d', 'e', 'f', 'g’]

uniao = lista1+lista2
print(uniao)
```

Resultado: ['a', 'b', 'c', 'd', 'e', 'f', 'g’]

Podemos unir duas listas com o operador +

Repetir elementos

h = [0,1,2]
h*4

Resultado: [0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2]

Podemos repetir os elementos de uma lista usando o operador *

CONSTRUTOR LIST()

```
tupla = (1, 2, 3, 4, 5)
lista = list(tupla)
Resultado: [1, 2, 3, 4, 5]
```

Podemos também criar uma lista com uma sequência de números inteiros por meio da função embutida range().

Sintaxe: range(início, fim, passo)

serie1 = list(range(2010, 2021))

Resultado:

[2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020]

CONVERTER UMA STRING NUMA LISTA

Podemos usar a função **split()** dividir uma string numa lista, sendo que cada palavra desta string será um item da lista. Para tanto usamos um delimitador (argumento sep).

Exemplos:

frase = 'Eu amo aprender Python!'

frase.split(sep =' ')
['Eu', 'amo', 'aprender', 'Python!']

```
y = 'python%python%python%python'
y.split(sep = '%')
```

```
z = 'carro,computador,livro,café'
z.split(',')
```

TRANSFORMAR UMA LISTA NUMA STRING

Podemos juntar strings dentro de uma lista, usando um caractere como separador, com a função **join()**

```
lista = ['Eu','amo','python!']
string = ' '.join(lista)
'Eu amo python!'
```

DEFINIR LISTAS A PARTIR DE VARIÁVEIS

num1 = 10
num2 = 20
num3 = 30

lista = [num1, num2, num3]

Resultado: [10, 20, 30]

DESEMPACOTAR LISTAS

```
valores = [350, 400, 900, 120]

 num1, num2, num3, num4 = valore
```

Resultado:

```
num1 =350
num2 =400
num3 =900
num4 =120
```

“FATIAR” LISTAS - SLICING

z = [4, 7, 9, 11, 3, 1]

z[1:4] elementos desde o índice 1 até ao índice 4
[7, 9, 11]

z[1:] elementos deste o indíce 1 até ao final
[7, 9, 11, 3, 1]

LISTA[INÍCIO:FIM]
LISTA[INÍCIO:FIM:PASSO]

z[::2]
[4,9,3]

Iterar uma Lista

Podemos percorrer todos os elementos de uma lista com o **loop for** e efetuar diversas operações. Vejamos um exemplo em que percorremos todos os elementos de uma lista e mostramos no ecrã cada um de seus elementos.

```
lista_carros = ['Uno', 'Gol', 'Argo', 'Celta', 'Palio', 'Mobi', 'Up']
```

```
for carro in lista_carros:
    print(carro)
```

Vamos agora mostrar o quadrado de cada número da lista ***lista_num***:

```
lista_num = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
for numero in lista_num:
    print(numero**2)
```

ITERAR UMA LISTA

```
numeros = [1, 2, 3, 4, 5]
for numero in numeros:
    if numero%2==0:
        print(numero, 'Par')
    else:
        print(numero,'Ímpar')
```

```
1 Ímpar
2 Par
3 Ímpar
4 Par
5 Ímpar
```

ITERAR UMA LISTA

```
lista1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
soma = 0
for num in lista1:
    soma+=num  #equivale a soma = soma+num
print(soma)
```

```
numeros = [10, 8, 7, 9, 7]
mean = sum(numeros)/len(numeros)  #cálculo da média
print(f'Média: {mean}')
```

```
numeros = [10, 8, 7, 9, 7]
soma = 0  #acumulador
x = 0  #contador
while x<len(numeros):  #condição lógica
    soma+=numeros[x]  #acumula cada valor da lista numeros no índice x
    x+=1  #incrementa o contador a cada laço
print(f'Média: {soma/x}')
```

VERIFICAR SE UM ELEMENTO ESTÁ CONTIDO NUMA LISTA

Podemos verificar se um determinado elemento está contido em uma lista com a *keyword* **in**

```
k = ['F', 'M', 'J', 'K']  #declara a lista
```

```
'F' in k  #verifica se o elemento 'F' está contido na lista
```

Devolve: True

```
'F' not in k  #verifica se o elemento 'F' não está contido na lista
```

Devolve: False

VERIFICAR SE UM ELEMENTO ESTÁ CONTIDO EM UMA LISTA

Podemos, por exemplo, criar uma estrutura condicional para verificar se um elemento está contido numa lista e caso seja uma condição verdadeira, mostrar uma determinada mensagem no ecrã.

```
if 'M' in k:
    print('M pertence a lista k.')
else:
    print('M não pertence a lista k.')
```

```
M pertence a lista k.
```

MÉTODOS DE LISTAS

Métodos em List	Descrição
append()	Adicona um elemento ao final da lista.
insert()	Adiciona um elemento em um índice especificado.
extend()	Adiciona os elementos de um iterável no final da lista.
pop()	Remove e retorna um elemento de um índice especificado.
remove()	Remove o primeiro elemento com o valor especificado.
clear()	Remove todos os elementos de uma lista.
copy()	Retorna uma cópia da lista selecionada.
sort()	Ordena a lista.
reverse()	Reverte a ordem dos elementos em uma lista.
index()	Retorna o primeiro índice do elemento especificado.
count()	Conta o número de vezes que o elemento aparece na lista.

APPEND - adiciona um elemento ao final da lista.


```
numeros =[45, 50, 55]  #declara a lista
numeros
```

```
numeros.append(60)  #adiciona o número 60 ao final da lista
numeros
```

```
[45, 50, 55, 60]
```

Se fizermos **numeros.append(60,61)** teremos:

TypeError: append() takes exactly one argument (2 given)

Ou seja, o método ***append*** recebe apenas um argumento. Poderíamos no entanto colocar os números dentro de uma lista ou tuplo e passar como argumento da função append.

```
numeros.append([60,65])
```

```
[45, 50, 55, 60, [60, 65]]
```

APPEND - adiciona um elemento ao final da lista.


```
numeros.append((70,75))
```

```
[45, 50, 55, 60, [60, 65], (70, 75)]
```

extend - adiciona os elementos de um iterável (por exemplo, lista,tupla...) ao final de uma lista.

```
numeros = [0, 0, 0, 0]  #declaramos uma lista
numeros.extend([1, 2, 3])  #adicionamos apenas os números do iterável

numeros [0, 0, 0, 0, 1, 2, 3]
```

EXTEND – adiciona elementos de uma coleção ao final de uma lista


```
numeros.extend((4,5,6))  #adicionamos apenas os números da tupla

numeros
[0, 0, 0, 0, 1, 2, 3, 4, 5, 6]

numeros.extend({7,8,9,10})  #adicionamos apenas os números do set

numeros
[0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 7]
```

EXTEND – adiciona elementos de uma coleção ao final de uma lista

```
numeros.extend('abcdef')  #adicionamos as letras da string

numeros
```

```
[0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 8, 9, 10, 7, 'a', 'b', 'c', 'd',
'e', 'f']
```

INSERT - adiciona um elemento numa posição especificada

lista.insert(index,object) - primeiro passamos o índice e depois o objeto.

```
numeros = [4, 10, 11, 12, 7, 9, 15]  #declara a lista
numeros.insert(1,0)  #insere o número 0 no índice 1
[4, 0, 10, 11, 12, 7, 9, 15]

numeros.insert(0, 11)  #insere o número 11 no índice 0
[11, 4, 0, 10, 11, 12, 7, 9, 15]
```

REMOVE – remove o primeiro item com o valor especificado

lista.insert(index,object) - primeiro passamos o índice e depois o objeto.

```
numeros = [10, 11, 12, 13]
numeros.remove(13)  #removemos o número 13 da lista
```

```
[10, 11, 12]
```

CLEAR – remove todos os elementos da lista


```
letras = ['abc','ab','a']
letras.clear() # aplicando método
[]
```

COUNT – retorna o número de ocorrências de um valor numa lista


```
numeros = [1, 1, 1, 0, 0, 0, 5, 3, 1, 2]
numeros.count(1)  #conta o número de ocorrências do número 1
```

```
4
```

REVERSE – inverte a ordem de uma lista


```
numeros = [10, 11, 12, 13, 14, 15]
numeros.reverse()  #inverte a ordem da lista
```

```
[15, 14, 13, 12, 11, 10]
```

SORT – ordena a lista


```
numeros = [-10, 100, -1, 60, 76, 33, -30, 150]  #declara uma lista não ordenada

numeros.sort()  #ordena a lista

 [-30, -10, -1, 33, 60, 76, 100, 150]

anos = [1950, 1960, 1970, 1980, 1990, 2000, 2010, 2020]  #declara uma lista não
 ordenada

anos.sort(reverse=True)

 [2020, 2010, 2000, 1990, 1980, 1970, 1960, 1950]
```

Index - retorna o índice do primeiro elemento com o valor especificado

index(value, start=0,stop)

```
lista = [5, 6, 7, 8]
```

```
lista.index(5)  #retorna o índice da primeira ocorrência do número 5
```

```
0
```

```
lista.index(7)  #retorna o índice da primeira ocorrência do número 7
```

```
2
```

```
numeros.index(3, 5)  #retorna o índice do valor 3, começando a contar a partir d
o índice 5 da lista
```

```
5
```

Copy - retorna uma cópia da lista. Existem duas formas de cópia em Python: Shallow Copy e Deep Copy.

In [211]:

```
lista_numeros = [4, 9, 11, -1]
nova = lista_numeros  #cria uma cópia de lista_numeros

nova
```

Out[211]:

```
[4, 9, 11, -1]
```

In [212]:

```
nova.append(0)  #acrescenta um número na lista nova
```

In [213]:

```
nova
```

Out[213]:

```
[4, 9, 11, -1, 0]
```

Este número também será acrescentado em *lista_numeros*.

Shallow Copy

Ou seja, adicionamos o número 0 na cópia (lista nova), como também na lista original.

Se acrescentarmos um valor na nossa lista original, teremos o mesmo comportamento.

Copy - retorna uma cópia da lista. Existem duas formas de cópia em Python: Shallow Copy e Deep Copy.

Deep Copy

```
lista_numeros = [1, 2, 3]
nova = lista_numeros.copy()  #deep copy
```

```
nova.append(4)  #adiciona o número 4 na cópia
```

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

A Lista Original não é afetada

```
lista_numeros  #a lista original não é afetada
```

```
Out[221]:

[1, 2, 3]
```