Aula 3 - Listas

June 10, 2021

1 Necessidade de se trabalhar com listas

A necessidade se dá pela repetição constante de varios valores do mesmo tipo, onde precisamos de uma maneira mais fácil de trabalhar com tais valores

[4]:
```
def main():
    num1 = 10
    num2 = 11
    num3 = 12
    num4 = 13
    num5 = 14
    soma = num1 + num2 + num3 + num4 + num5
    print(soma)

main()
```

```
60
```

[5]:
```
def main():
    num1 = 10
    num2 = 11
    num3 = 12
    num4 = 13
    num5 = 14
    num6 = 15
    num7 = 16
    num8 = 17
    num9 = 18
    num10 = 19
    soma = num1 + num2 + num3 + num4 + num5 + num6 + num7 + num8 + num9 + num10
    print(soma)

main()
```

```
145
```

[6]:
```
def main():
    aluno1 = 'Felipe'
    aluno2 = 'Alisson'
```

```
    aluno3 = 'Haynes'
    prova1_1 = 8
    prova2_1 = 10
    prova1_2 = 10
    prova2_2 = 7
    prova1_3 = 8
    prova2_3 = 9
    media1 = (prova1_1 + prova2_1) / 2
    media2 = (prova1_2 + prova2_2) / 2
    media3 = (prova1_3 + prova2_3) / 2
    print(f"O aluno1 ({aluno1}) ficou com a média de {media1}")
    print(f"O aluno2 ({aluno2}) ficou com a média de {media2}")
    print(f"O aluno3 ({aluno3}) ficou com a média de {media3}")

main()
```

```
O aluno1 (Felipe) ficou com a média de 9.0
O aluno2 (Alisson) ficou com a média de 8.5
O aluno3 (Haynes) ficou com a média de 8.5
```

Motivação das listas Trabalhar com vários valores relacionados em uma só variável

[7]:
```
def main():
    # Todos os possiveis alunos
    # Felipe, Alisson, Haynes, ByLearner
    # As notas respectivas dos alunos
    # 8 e 10, 10 e 7, 8 e 9, 10 e 10

    # Media de [Primeiro Aluno] = [Notas do 1º] / 2
    # Para todos -> Media do [Atual] = [Notas do Atual] /2
    pass

main()
```

2 Criando listas

Listas são tipos de dados que armazenam vários valores em uma variável só.
Em listas, nós trabalhos com índices.
índices começam sempre por zero

[8]:
```
primeira_lista = ['Felipe', 'Alisson', 'Haynes', 'ByLearner']
#------Posição------ 1º ------ 2º ------- 3º ------ 4º ------
#------Índice--------0---------1----------2---------3--------
print(primeira_lista[0])
print(primeira_lista[1])
print(primeira_lista[2])
print(primeira_lista[3])
```

```
Felipe
Alisson
Haynes
ByLearner
```

[10]:
```
print(primeira_lista[4]) # Erro devido a essa posição não existir
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-10-c3fab2670192> in <module>
----> 1 print(primeira_lista[4]) # Erro devido a essa posição não existir

IndexError: list index out of range
```

[11]:
```
primeira_lista[2] = 'Guilherme'
```

[12]:
```
print(primeira_lista[0])
print(primeira_lista[1])
print(primeira_lista[2])
print(primeira_lista[3])
```

```
Felipe
Alisson
Guilherme
ByLearner
```

[13]:
```
print(primeira_lista)
```

```
['Felipe', 'Alisson', 'Guilherme', 'ByLearner']
```

2.1 Tentativa de adicionar na lista

[15]:
```
primeira_lista[4] = 'Novo Aluno' # Não é assim que acrescentamos dados, pois a␣
↪posição não existe
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-15-53143b759cf9> in <module>
----> 1 primeira_lista[4] = 'Novo Aluno' # Não é assim que acrescentamos dados,␣
↪pois a posição não existe

IndexError: list assignment index out of range
```

[16]:
```
primeira_lista.append('Novo Aluno')
```

[17]:
```
print(primeira_lista)
```

```
['Felipe', 'Alisson', 'Guilherme', 'ByLearner', 'Novo Aluno']
```

Conclusão: Adicionamos itens na lista através do método append

2.2 Métodos alternativos de criar listas

[21]:
```
lista = [1,2,3,4]

lista2 = ['a','b','c']

lista3 = ['a','b',2,3]

lista_alunos = []
```

Funções built in são funções externas

[23]:
```
lista_built_in = list() # equivale a lista = []
variavel_para_lista = 12
lista_built_in_cheia = list(['a',2,variavel_para_lista])
```

2.3 Copiando uma lista para outra (atribuindo o valor)

A lista B passa a Referenciar a lista A, então, uma alteração na B também altera a A (e vice-versa)

[24]:
```
lista_a = [1,2,3]
lista_b = lista_a
lista_b.append(4)
print(lista_a)
print(lista_b)
```

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

2.4 Clonar uma lista (passar os valores dela) para outra

Quando trabalhamos com colchetes ([]) trabalhamos com os elementos da lista, e não com o objeto dela

[25]:
```
lista_a = [1,2,3]
lista_b = lista_a[:]
lista_b.append(4)
print(lista_a)
print(lista_b)
```

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

2.5 Recuperando elementos da lista

Para pegar um valor da lista, usamos o índice do elemento que queremos

[26]:
```
lista_a = [1,2,3]
elemento1 = lista_a[0]
elemento2 = lista_a[1]
elemento3 = lista_a[2]
print(elemento1)
print(elemento2)
print(elemento3)
```

```
1
2
3
```

Para recuperarmos com mais de um elemento, vamos pegar uma **fatia** da lista
Fatia funciona da seguinte maneira:
- Nós queremos da posição X até a posição Y - Então nós queremos [X:Y] - Ou seja... [Inicial : Final]

- Caso não tenha um dos valores na fatia, pegaremos TUDO daquele 'lado' Exemplo:
- Para lista[1:4] -> Pegamos elementos entre 1 e 4
- Para listas[:4] -> Pegamos elementos até 4
- Para listas[2:] -> Pegamos elementos a partir do 2

[27]:
```
lista = [0,1,2,3,4,5,6,7,8,9,10]
primeiro_teste = lista[1:4] # Entre 1 e 4
segundo_teste = lista[:4] # Até 4
terceiro_teste = lista[2:] # A partir de 2
```

[28]:
```
print(primeiro_teste)
print(segundo_teste)
print(terceiro_teste)
```

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

2.5.1 Sempre que trabalhamos com Intervalos (range) o limite inferior é INCLUSIVO e o superior é EXCLUSIVO

De maneira informal: O primeiro valor (menor) entra no intervalo, já o ultimo valor (maior) não entra

[29]:
```
# 1 até 4
# 1, 2 e 3
# 2 até o fim (5 elementos)
# 2, 3, 4 e 5
# até 5
# 0, 1, 2, 3, 4
```

[31]:
```
lista_a = [1,2,3]
lista_b = lista_a[:] # De tudo : Até tudo = Todos
```

A fatia [:] significa 'pegue tudo da lista'

2.6 Juntando listas (join)

Serve para 'somar' (juntar) duas ou mais listas

[39]:
```
ids_instrutores = [1,2,3]
ids_alunos_py = [10,9]
ids_alunos_csharp = [8,10]

ids_bylearners = ids_instrutores + ids_alunos_py
print(ids_bylearners)
```

```
[1, 2, 3, 10, 9, 8, 10]
```

[38]:
```
ids_bylearners += ids_alunos_csharp
print(ids_bylearners)
```

```
[1, 2, 3, 10, 9, 8, 10]
```

[40]:
```
ids_instrutores = [1,2,3]
ids_alunos_py = [10,9]
ids_alunos_csharp = [8,10]

ids_bylearners = ids_instrutores + ids_alunos_py + ids_alunos_csharp
print(ids_bylearners)
```

```
[1, 2, 3, 10, 9, 8, 10]
```

2.7 Funções Nativas para listas

Primeira função: append() => Insere um elemento no final na lista

[41]:
```
numeros = ['um']
numeros.append("dois")
numeros.append("tres")
numeros.append("quatro")

print(numeros)
```

```
['um', 'dois', 'tres', 'quatro']
```

Segunda função: index() => Retorna o índice de um determinado elemento

[43]:
```
bylearners = ['Felipe','Alisson','Haynes','ByLearner <3']
#--Indices------ 0 ------- 1 ------ 2 -------- 3 --------
indice_haynes = bylearners.index('Haynes')

print(indice_haynes)
print(bylearners.index('ByLearner <3'))
```

```
2
3
```

Extra: In => Indica se o elemento está ou não na lista
Checa a existência (ou não existência) do elemento

[48]:
```
bylearners.index('Aluno')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-48-7fb48c99ae71> in <module>
----> 1 bylearners.index('Aluno')

ValueError: 'Aluno' is not in list
```

[51]:
```
print('Aluno' in bylearners)
print('Felipe' in bylearners)
print('ByLearner <3' in bylearners)
```

```
False
True
True
```

Terceira função: insert() => Inseri um elemento na lista em uma determinada posição

[45]:
```
animais = ['Gato','Cachorro','Hamster']
animais.append('Furão')
print(animais)
```

```
['Gato', 'Cachorro', 'Hamster', 'Furão']
```

[57]:
```
animais = ['Gato','Cachorro','Hamster']
#------------0--------1---------2------
animais.insert(1,'Furão')
print(animais)
```

```
['Gato', 'Furão', 'Cachorro', 'Hamster']
```

Quarta Função: Remove => Remove um determinado elemento

[58]:
```
animais.remove('Cachorro')
print(animais)
```

```
['Gato', 'Furão', 'Hamster']
```

Quinta Função: Pop => Remove um elemento em um determinado indice
Pop interage com a lista, apenas removendo o elemento do indice enviado por parâmetro

[63]:
```
animais = ['Gato','Furão','Cachorro','Hamster']
animais.pop(2)
print(animais)
```

```
['Gato', 'Furão', 'Hamster']
```

[64]:
```
animais.pop(5) # Temos que tomar cuidado com o limite dos indices
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-64-5e7af14a4f5a> in <module>
----> 1 animais.pop(5) # Temos que tomar cuidado com o limite dos indices

IndexError: pop index out of range
```

Sexta Função: Del => Remove um elemento em um determinado indice
Remove o elemento passado por parâmetro (envio um elemento)

[66]:
```
animais = ['Gato','Furão','Cachorro','Hamster']
del(animais[2])
print(animais)
```

```
['Gato', 'Furão', 'Hamster']
```

[67]:
```
animais = ['Gato','Furão','Cachorro','Hamster']
animais[2]
```

[67]:
```
'Cachorro'
```

Sétima Função: Sort => Ordena os nosos números
Também temos a função inter Sorted

[68]:
```
lista = [2, 1, 4]
print(lista)
lista.sort()
print(lista)
```

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

[70]:
```
lista = [2, 1, 4]
print(sorted(lista))
```

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

[74]:
```
lista = [2, 1, 4]
print(lista.sort()) # Interage diretamente na lista... Não retorna nada
lista2 = lista.sort() # Não da para atribuir
print(lista2)
```

```
None
None
```

[77]:
```
lista = ["c",'b','a']
print(sorted(lista)) # Sorted me retorna uma lista já ordenada
lista2 = sorted(lista) # Consigo atribuir a lista
print(lista2)
```

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

[80]:
```
lista = ["c",'b','a',str(1),str(2)]
print(sorted(lista)) # Sorted me retorna uma lista já ordenada
```

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

Oitava Função: Len => Retorna o tamanho da lista

[110]:
```
ls = ['t','e','s','t','e']
print(len(s))
```

```
5
```

2.8 Manipulação de String

[81]:
```
# Caractere é uma letra só
char = 'c'
# String é um 'conjunto' de carecteres
string_palavra = 'Felipe'
string_frase = 'Seja um ByLearner'

# Vetor (Array) => Sequencia de elementos do mesmo tipo
# String é um Vetor (array) de caracteres
```

[85]:
```
string = "ByLearn"
#---------0123456
print(string[0])
print(string[1])
print(string[2])
print(string[3])
print(string[4])
print(string[5])
print(string[6])
```

```
B
y
L
e
a
r
n
```

[86]:
```
string = "ByLearn"
#---------0123456|7
print(string[7]) # Não existe
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-86-4058738c01cb> in <module>
      1 string = "ByLearn"
      2 #---------0123456|7
----> 3 print(string[7]) # Não existe

IndexError: string index out of range
```

Strings são imutáveis

[88]:
```
palavra = "teste"
palavra[2] = 'z'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-88-2f5dc3c312e6> in <module>
      1 palavra = "teste"
----> 2 palavra[2] = 'z'

TypeError: 'str' object does not support item assignment
```

String **podem** ser fatiadas, como por exemplo, para mudar o valor de algum índice

[90]:
```
nova_palavra = palavra[:2] + 'z' + palavra[3:]
nova_palavra
```

[90]:
```
'tezte'
```

[96]:
```
palavra = 'abcdef'
print(palavra[-1])
print(palavra[-2])
print(palavra[:]) # print(palavra)
```

```
f
e
abcdef
```

Pesquisar em Strings

[97]:
```
'b' in 'abc'
```

[97]:
```
True
```

[99]:
```
'd' in 'abc'
```

[99]: False

[100]:
```
'b' not in 'abc'
```

[100]: False

[101]:
```
'd' not in 'abc'
```

[101]: True

Concatenar Strings (somar)

[102]:
```
string = 'Fe' + 'Li' + 'Pe'
```

[103]:
```
string
```

[103]: 'FeLiPe'

[104]:
```
print('Fe'+'li'+'pe')
```

```
Felipe
```

Alterar entre minúsculo e maiúsculo

[105]:
```
string = 'Fe' + 'Li'.lower() + 'Pe'.lower()
string
```

[105]: 'Felipe'

[106]:
```
string = 'Felipe'.upper()
string
```

[106]: 'FELIPE'

Tamanho da string

[107]:
```
s = 'teste'
print(len(s))
```

```
5
```

Checar se todos os caracteres são letras

[112]:
```
print("abc".isalpha())
print("1fg".isalpha())
print("123".isalpha())
print("++;-/".isalpha())
```

```
True
False
```

```
False
False
```

Remover espaços em branco tanto inicio quanto no fim

[113]:
```
" sobrando espaços ".strip()
```

[113]:
```
'sobrando espaços'
```

[114]:
```
'   sobrando espaços   '.strip()
```

[114]:
```
'sobrando espaços'
```

Juntar os itens da string através de um delimitador

[115]:
```
",".join("abc") # Aqui faz sentido
```

[115]:
```
'a,b,c'
```

[116]:
```
" , ".join("abc") # Aqui faz sentido
```

[116]:
```
'a , b , c'
```

[118]:
```
'|letra: '.join("abc") # Não faz sentido
```

[118]:
```
'a|letra: b|letra: c'
```

Separar uma string através de um delimitador

[119]:
```
s = "n o m e"
```

[121]:
```
s.split()
```

[121]:
```
['n', 'o', 'm', 'e']
```

[122]:
```
s = "n,o,m,e"
s.split(",")
```

[122]:
```
['n', 'o', 'm', 'e']
```

[126]:
```
s = 'nome'
s.split()
```

[126]:
```
['nome']
```

2.9 Listas dentro de listas

Podemos criar quantas listas quisermos dentro de outras listas, bantando o elemento ser também uma lista (Utilizando as barras [])

[3]:
```
lista_dentro_de_lista = [[1,2],[3,4]]
```

[4]:
```
lista_dentro_de_lista[0]
```

[4]:
```
[1, 2]
```

[5]:
```
lista_dentro_de_lista[0][0]
```

[5]:
```
1
```

[6]:
```
lista_dentro_de_lista[1][0]
```

[6]:
```
3
```

[ ]:
```

```