Seção 4 - Estrutura de dados

Essa aula é inspirada em:

- https://docs.python.org/3/tutorial/datastructures.html (https://docs.python.org/3/tutorial/datastructures.html)
- LEE, Kent D. Python Programming Fundamentals. Second Edition. Springer - Verlag London 2014.

4.1 Métodos para listas

Método	Descrição
.append(*variável*)	Adiciona a variável no final
.extend(*outra_lista*)	Adiciona os elementos ao final
.remove(*variável*)	Remove o primeiro elemento com valor *variável*
.count(*variável*)	Conta o número de elementos com o valor *variável*
.sort()	Reordena elementos em ordem numérica ou alfabética
.reverse()	Inverte a ordem dos elementos
.copy()	Retorna uma cópia da lista
.index(*variável*)	Retorna o índice da primeira *variável* da lista

In [27]:

```
lista_inteiros = [1,2,3,4,5]
lista_2 = [6,7,8]
lista_inteiros.append(lista_2)
len(lista_inteiros)
```

Out[27]:

```
6
```

In [29]:

```
lista_letras = ['a','b','c','a','a']
n_vezes = lista_letras.count('a')
n_vezes
```

Out[29]:

```
3
```

In [31]:

```
lista_3 = [1,10,5,3,8]
lista_3.sort()
lista_3
```

Out[31]:

```
[1, 3, 5, 8, 10]
```

In [34]:

```
lista_4 = ['a','b','c','a','a','10','1','5']
lista_4.sort()
lista_4
```

Out[34]:

```
['1', '10', '5', 'a', 'a', 'a', 'b', 'c']
```

4.2 Formas de trabalhar com listas

Função	Descrição
range(*inteiro*)	retorna um iterável do tipo range
lambda x: *função_x*	Retorna uma função
map(*função*,*lista*)	Retorna um iterável do tipo map
list(*iterável*)	Cria uma lista a partir de um iterável

Atenção, este não é um tutorial sobre programação funcional! Quem se interessar pode pesquisar por: *Functional Programming* ou Programação Funcional.

In [36]:

```
lista_quadrados = []

for i in range(5):
    lista_quadrados.append(i**2)
lista_quadrados
```

Out[36]:

```
[0, 1, 4, 9, 16]
```

In [39]:

```
a = list(range(10))
a
```

Out[39]:

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

In [47]:

```
numeros = [1,2,3,4,5]
f = lambda x:x**2

lista_5 = list(map(f,numeros))
lista_5
```

Out[47]:

```
[1, 4, 9, 16, 25]
```

In [48]:

```
quadrados = [x**2 for x in range(6)]
quadrados
```

Out[48]:

```
[0, 1, 4, 9, 16, 25]
```

4.3 Tuplas, conjuntos e dicionários

Tipo	Descrição	Sintaxe
tuple	tupla: uma sequência de dados que é imutável	vogais = ('a','e','i','o','u')
set	conjunto: elementos não possuem ordem e não se repetem	alg_decimais = {0,1,2,3,4,5,6,7,8,9}
dict	dicionário: são indexaxos por uma chave *keys*	alg_romanos = {'I':1,"II":2,'III':3,'IV':4,'V':5,'X':10}

In [54]:

```
vogais = ('a','e','i','o','u')
vogais[0]

len(vogais)
```

Out[54]:

```
5
```

In [59]:

```
vogais_list = ['a','e','i','o','u']

for i in enumerate(vogais_list):
    print(i[1])
```

```
a
e
i
o
u
```

In [64]:

```
set_variaveis = {'e','i','o','a'}
set_variaveis
```

Out[64]:

```
{'a', 'e', 'i', 'o'}
```

In [69]:

```
dict_1 = {'dia':10,'mes':5}
dict_1['mes']
```

Out[69]:

```
5
```

4.4 Métodos para dicionários

Método	Descrição
.items()	Retorna os itens do dicionário
.keys()	Retorna as chaves do dicionário
.values()	Retorna as chaves do dicionário
.pop(*key*)	Remove a chave especificada e retorna o valor do item
.copy()	Retorna uma cópia
.clear()	Remove os itens
.get(*key*)	Retornas a váriável em *key*

Disponível em: https://link.springer.com/chapter/10.1007%2F978-1-4471-6642-9_12 (https://link.springer.com/chapter/10.1007%2F978-1-4471-6642-9_12)

In [26]:

```
alg_romanos = {'I':1,'II':2,'III':3,'IV':4}

a = alg_romanos.pop('IV')
alg_romanos
```

Out[26]:

```
{'I': 1, 'II': 2, 'III': 3}
```

In [33]:

```
list(alg_romanos.items())
```

Out[33]:

```
[('I', 1), ('II', 2), ('III', 3)]
```

In [35]:

```
list(alg_romanos.keys())
```

Out[35]:

```
['I', 'II', 'III']
```

In [37]:

```
list(alg_romanos.values())
```

Out[37]:

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

In [44]:

```
a = alg_romanos.get('I','II')
```

4.5 Exercício 1 -

Dada a lista ['P','A','Y','A','T','A','H','O','N'], conte o número de variáveis 'A' e utilize um loop para remover todos os 'A' excedente.

In [ ]:

```

```

4.6 Exercício 2 -

Utilizando somente uma linha de programação, crie uma lista que contenha os números ímpares de 1 a 51.

In [ ]:

```

```

4.7 Exercício 3 -

Crie um dicionário que correlacione as seguintes listas:

- valores = [1,2,3,4,5]
- keys = ['a','b','c','d','e']

Utilize um loop.

In [ ]:

```

```

4.8 Exercício 4 -

A partir do dicionário criado no exercício anterior, recrie as listas keys e valores.

In [ ]:

```

```

In [ ]:

```
```

4.9 Preparação para o desafio 1 (MATPLOTLIB)

https://matplotlib.org/tutorials/introductory/usage.html#sphx-glr-tutorials-introductory-usage-py (https://matplotlib.org/tutorials/introductory/usage.html#sphx-glr-tutorials-introductory-usage-py)

In [108]:

```
import matplotlib.pyplot as plt
%matplotlib notebook

x = [x for x in range(101)]
y = [y**2 for y in x]

#plt.plot(x,y,'+')
#plt.xlabel('x')
#plt.ylabel('y')
```

In [111]:

```
from math import sqrt

a = sqrt(4)
a
```

Out[111]:

```
2.0
```

4.10 Desafio 1 - Equação da circunferência

Dada a equação da cirunferência, faça o gráfico.

$$(x-a)^2 + (y-b)^2 = r^2, r = 100, a = 100, b = 100$$

In [ ]:

```
```

4.11 Preparação para o desafio 2 (LENDO ARQUIVOS)

In [121]:

```
file_path = 'PIB_100_maiores_cidades_2017.txt'

with open(file_path) as f:
    f_data = f.readlines()
    f.closed

lista_colunas = f_data[0].split(',')
lista_colunas
```

Out[121]:

```
['Municípios',
 'Estado',
 'Posição',
 'PIB (1 000 R$)',
 'Participação (%)',
 'Participação acumulada (%)\n']
```

4.12 Desafio 2 - Dados do IBGE

Importe a tabela dos 100 maiores municípios em relação ao PIB e responda as seguintes perguntas:

- Quantos municípios estão no estado de São Paulo?
- Qual a participação acumulada desses municípios?

Dicas e informações:

- Fonte dos dados: https://www.ibge.gov.br/estatisticas/downloads-estatisticas.html (https://www.ibge.gov.br/estatisticas/downloads-estatisticas.html)
- Arquivo para importar: ./IBGE-PIB_Municipios_2017/PIB_100_maiores_cidades_2017.txt
- Para ler arquivos acesse o tópico 7.2 do link: https://docs.python.org/3/tutorial/inputoutput.html (https://docs.python.org/3/tutorial/inputoutput.html)

Municípios	Estado	Posição	PIB (1 000 R$)	Participação (%)	Participação acumulada (%)
São Paulo	SP	1º	699288352	10.6	10.6
Rio de Janeiro	RJ	2º	337594462	5.1	15.8
Brasília	DF	3º	244682756	3.7	19.5
Belo Horizonte	MG	4º	88951167.8	1.4	20.8
Curitiba	PR	5º	84702356.7	1.3	22.1
Osasco	SP	6º	77910495.9	1.2	23.3
Porto Alegre	RS	7º	73862306	1.1	24.4
Manaus	AM	8º	73201651.2	1.1	25.5
Salvador	BA	9º	62717483.4	1.0	26.5
Fortaleza	CE	10º	61579403.2	0.9	27.4
Campinas	SP	11º	59053563	0.9	28.3
Guarulhos	SP	12º	55743650	0.8	29.2
Recife	PE	13º	51859618.3	0.8	29.9
Goiânia	GO	14º	49023142.4	0.7	30.7

In [ ]:

```
```

4.13 Desafio 2 - Dados do IBGE (usando PANDAS)

https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html)

In [169]:

```
import pandas as pd
file_path = 'PIB_100_maiores_cidades_2017.txt'

df = pd.read_csv(file_path, engine = 'python')
df.head()
```

Out[169]:

	Municípios	Estado	Posição	PIB (1 000 R$)	Participação (%)	Participação acumulada (%)
0	São Paulo	SP	1º	6.992884e+08	10.622125	10.622125
1	Rio de Janeiro	RJ	2º	3.375945e+08	5.128028	15.750153
2	Brasília	DF	3º	2.446828e+08	3.716708	19.466861
3	Belo Horizonte	MG	4º	8.895117e+07	1.351160	20.818021
4	Curitiba	PR	5º	8.470236e+07	1.286621	22.104642

In [175]:

```
Contador_estados = df['Estado'].value_counts()
Contador_estados

Is_SP = df['Estado'] == 'SP'

df_SP = df[Is_SP]
df_SP.head()

pib_SP = df_SP['Participação (%)'].sum()

print(pib_SP)
```

```
23.195820665883666
```

In [ ]:

```
```