



**CENTRO UNIVERSITÁRIO  
INSTITUTO DE EDUCAÇÃO SUPERIOR DE BRASÍLIA – IESB**



# **Aprendizagem de Máquina**

**PABLO COELHO FERREIRA**

Antes de  
começarmos...

.

REVISAR  
é  
Importante

# Resumo

Nome	Tipo	Descrição
Inteiro	int	Todos os números inteiros: 2 300 100000
Ponto Flutuante	float	Todos números com decimais: 2.102 403.98
Strings	str	Sequências de caracteres: "oi" "Pablo" "Janeiro"
Listas	list	Sequência ordenada de objetos:
Dicionários	dict	Sequência de pares sem ordem
Tuplas	tup	Sequência fixa de objetos
Conjuntos	set	Coleção de objetos sem ordem
Booleanos	bool	Verdadeiro ou falso

# Operações com Strings

- Pulando linha: `\n`
  - `print ( "Hello \n World")`
- SLICE
  - `[start:stop:step]`, traduzindo `[começo:fim:passos]`
  - START = Início do slice
  - STOP = é até onde vai o slice (mas não inclui o próprio. Para um elemento antes)
  - STEP = é o tamanho do passo que você vai dar (um pulo)

# Montando uma lista a partir de uma string

- `strMinhaString = 'Pablo é professor de ADS'`
- `strMinhaString = strMinhaString.split()`
- `print (strMinhaString)` #vai imprimir ['Pablo', 'é', 'professor', 'de', 'ADS']
- O resultado é uma lista (vetor) com os itens separados pelo caracter dentro dos parêntesis do split. Se não tiver nenhum utilizará o espaço.

# Outro exemplo de SPLIT

- `strMinhaString = 'calabreza, portuguesa, marguerita, frango catupiri, bacon com frango, pepperoni'`
- `strMinhaString = strMinhaString.split(',')`
- `print (strMinhaString)`
- `['calabreza', ' portuguesa', ' marguerita', ' frango catupiri', ' bacon com frango', ' pepperoni']`

# f-strings

- Insere variáveis em uma string
- Sintaxe: (f'TEXTO QLQ {<nome variável> } ')
- Exemplo:
- `strMinhaString = 'Pablo'`
- `print (f'Meu nome é: {strMinhaString}.')` #vai Imprimir: Pablo

# Tuples

- São idênticas as listas (vetores) mas seus valores são constantes (imutáveis).
- Uma tupla é feita com parêntesis ao invés de colchete.
- Dois métodos: count e index



# Aula 03

# Aula de hoje

- Sets
- I/O files
- Operadores de comparação
- Estruturas de decisão e repetição (for)

# SETS

- São coleções não ordenadas de elementos únicos.

```
setMeuConjunto = set()
```

```
setMeuConjunto.add('a')
```

```
setMeuConjunto.add('d')
```

```
setMeuConjunto.add('b')
```

```
setMeuConjunto.add(2)
```

```
print (setMeuConjunto) # imprime: {'d', 'b', 2, 'a'} em qualquer ordem.
```

# SETS

```
setMeuConjunto = set()
```

```
setMeuConjunto.add('a')
```

```
setMeuConjunto.add('d')
```

```
setMeuConjunto.add('a')
```

```
print (setMeuConjunto) # {'d', 'a'} – em qualquer ordem.
```

# Sets e listas

```
lisMinhaLista = [1,1,1,1,2,2,21,3,4,51,23,1,2,3,5,8]
```

```
print (lisMinhaLista)
```

```
print (set(lisMinhaLista))
```

```
#Lista -> [1, 1, 1, 1, 2, 2, 21, 3, 4, 51, 23, 1, 2, 3, 5, 8]
```

```
#Set -> {1, 2, 3, 4, 5, 8, 51, 21, 23}
```

# Arquivos – Criar e gravar

- **1º passo – criar um arquivo.**

Se tentar criar um arquivo que já existe vai dar pau!

```
f = open ("meuArquivo.txt","x")
```

- **2º Passo – abrir o arquivo para escrever algo nele (append)**

```
f = open ("meuArquivo.txt","a")
```

```
f.write ("Minha primeira linha.")
```

```
f.write ("Minha segunda linha.\nTerceira linha")
```

- **3º Passo – fechar o arquivo**

```
f.close()
```

# Arquivos - ler

```
f = open("meuArquivo.txt","r")  
arqMeuArquivo = f.read()  
print (arqMeuArquivo)  
f.close()
```

```
#Minha primeira linha.Minha segunda linha.  
#Terceira linha
```

Por que da primeira para a segunda linha o arquivo não “pulou” a linha?

# Exercícios

- 01 – Repetir os códigos
- 02 – Criar uma lista com os seguintes elementos, em ordem alfabética (Atenção as letras maiúsculas e minúsculas e aos números e strings):
  - Carro, Abajur, Moto, Barco, Ferro de passar roupa, “2”, Barco, moto
  - Gravar a lista em um arquivo APENAS com elementos únicos (sem estarem repetidos).
  - DESAFIO: Garantir que os dados no arquivo estejam ordenados.



# Operadores de comparação

**a = 3 b = 4 c=3**

Operador	Descrição	Exemplo
==	Se os dois lados da operação forem iguais, então a condição é verdadeira	(a==b) é falso (a==a) é verdadeiro
!=	Não igual. Retorna verdadeiro se os lados da operação forem diferentes.	(a==b) é verdadeiro (a==a) é falso
>	Maior que. Retorna verdadeiro se o lado esquerdo da operação for maior que o direito.	(a>b) é falso.
<	Menor que. Retorna verdadeiro se o lado direito da operação for maior que o esquerdo.	(a>b) é verdadeiro.
>=	Similar ao ">" porém retorna verdadeiro se o valor da esquerda e direita foram iguais.	(a>=c) é verdadeiro (a>=b) é falso
<=	Similar ao "<" porém retorna verdadeiro se o valor da esquerda e direita foram iguais.	(a<=c) é verdadeiro (a<=b) é falso

# Cadeia de comparações

- Você pode realizar várias comparações ao mesmo tempo para obter um único resultado (V/F).
- Os operadores são AND e OR, equivalentes a “E” e “OU”
- Se uma condição é encadeada com AND, ambas devem ser verdadeiras para o resultado ser verdadeiro. Se uma delas for falsa, o resultado é falso.
- Se uma condição é encadeada com OR, basta uma ser verdadeira para o resultado ser verdadeiro. Se TODAS forem falsas, o resultado é falso.

# Exemplos

a = 1

b = 2

c = 3

print (a == b) #False

print (a == b and a == a) #False

print (a == b or a == a) #True

print (a == a or a == a and a==c) #True

print (a == a and a == a or a==c) #True

print (a == a and a == c or a==a) #True

**print (a == c and (a == c or a==a)) #False**

# Cuidado!

```
>>> True and True and (3 or True)
3
```

Para evitar utilize o encapsulamento:

```
>>> bool(True and 3)
True
```

# Estruturas de decisão

# Estrutura de decisões

if condição:

    código2 (executa todas as linhas iniciadas na mesma coluna – indentação)

else:

    código2 (executa todas as linhas iniciadas na mesma coluna – indentação)

código 3 (fora da condição if)

# Estrutura de decisões

if condição:

    código1 (executa todas as linhas iniciadas na mesma coluna – indentação)

elif:

    código2 (executa todas as linhas iniciadas na mesma coluna – indentação)

else:

    código3 (executa todas as linhas iniciadas na mesma coluna – indentação)

Código 4 (fora da condição if)

# Sem codificar... O que sairá deste código?

```
a = 1
b = 2
c = 3
if (a == b):
    print ("Primeiro teste é verdadeiro.")
if (a == b and a == a):
    print ("Segundo teste é verdadeiro.")
if (a == b or a == a):
    print ("Terceiro teste é verdadeiro.")
if (a == a or a == a and a==c):
    print ("Quarto teste é verdadeiro.")
```

```
if (a == a and a == a or a==c):
    print ("Quinto teste é verdadeiro.")
if (a == a and a == c or a==a):
    print ("Sexto teste é verdadeiro.")
if (a == c and (a == c or a==a)):
    print ("Sétimo teste é verdadeiro.")
print ("Saindo do IF.")
```



# Resultado

Terceiro teste é verdadeiro.

Quarto teste é verdadeiro.

Quinto teste é verdadeiro.

Sexto teste é verdadeiro.

Saindo do IF.

# Sem codificar... O que sairá deste código?

```

a = 1
b = 2
c = 3
if (a == b):
    print ("Primeiro teste é verdadeiro.")
if (a == b and a == a):
    print ("Segundo teste é verdadeiro.")
if (a == b or a == a):
    print ("Terceiro teste é verdadeiro.")
if (a == a or a == a and a==c):
    print ("Quarto teste é verdadeiro.")

```

```

if (a == a and a == a or a==c):
    print ("Quinto teste é verdadeiro.")
if (a == a and a == c or a==a):
    print ("Sexto teste é verdadeiro.")
if (a == c and (a == c or a==a)):
    print ("Sétimo teste é verdadeiro.")
print ("Saindo do IF.")

```

# Exercícios

- 01 – Repetir os códigos
- 02 – Criar um programa que leia duas listas (use o comando INPUT para montar cada uma das listas\*) e imprima no console se eles são iguais ou não.

\*Considere o código “#9” para interromper a carga de uma lista e passar para a outra.

# Estruturas de Repetição

# FOR

- No Python a repetição: “for x = 1; x<5; x=x+1” é escrita: “for \_ in range(x)”

```
for i in range (0,10,2):  
    print (i)
```

# saída:

# 0

# 2

# 4

# 6

# 8

# FOR

- O for também é utilizado para percorrer listas.

```
lstMinhaLista = [1,2,3,4,5]
```

```
for qlqNome in lstMinhaLista: # O primeiro parâmetro é uma variável  
    # que pode ter qualquer nome.  
    # o segundo parâmetro (depois do IN)  
    # é o nome da lista que você quer percorrer.
```

```
print (qlqNome) # Vai imprimir 'de 1 a 5, sendo um número embaixo do  
                # outro.
```

# FOR

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

```
for numero in lstMinhaLista:
```

```
    # Verificar se é um número par
```

```
    if (numero % 2 == 0):
```

```
        print (f'O número {numero} é par.')
```

```
    else:
```

```
        print (f'O número {numero} é ímpar.')
```

# FOR

```
lstMinhaLista = [(1,2),(3,4),(4,5),(5,6),(7,8)]
```

```
print (len(lstMinhaLista))
```

```
for item in lstMinhaLista:  
    print (item)
```



# FOR

```
lstMinhaLista = [(1,2),(3,4),(4,5),(5,6),(7,8)]
```

```
print (len(lstMinhaLista))
```

```
for a,b in lstMinhaLista:
```

```
    print (f'Este é o primeiro item da tupla: {a} e o segundo é: {b}')
```

# ZIP

- A função ZIP junta itens de listas em uma única lista com os itens compostos dentro da lista.

```
lista1 = [1, 2, 3]
```

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

```
resultado = zip(lista1, lista2)
```

```
for x,y in resultado:
```

```
    print(f'Primeiro elemento: {x} e segundo: {y}')
```

```
# resultado = [(1,"a"),(2,"b"),(3,"c")]
```

```
# saída:
```

```
# Primeiro elemento: 1 e segundo: a
```

```
# Primeiro elemento: 2 e segundo: b
```

```
# Primeiro elemento: 3 e segundo: c
```

# list e zip

```
lstMinhaLista1 = [1,2,3]
```

```
lstMinhaLista2 = ['a','b','c']
```

```
lstMinhaLista3 = [100,200,300]
```

```
lstMinhaLista4 = list(zip(lstMinhaLista1,lstMinhaLista2,lstMinhaLista3))
```

```
print (lstMinhaLista4)
```

```
# saída: [(1, 'a', 100), (2, 'b', 200), (3, 'c', 300)]
```

# in

- in busca se um elemento está na lista

```
lstMinhaLista1 = [1,2,3]
```

```
print (1 in lstMinhaLista1)
```

```
# saída: True
```

# max e min

- max trás o maior elemento da lista e min o menor.

```
lstMinhaLista1 = [1,2,3,4,5]
```

```
print (max(lstMinhaLista1))
```

```
print (min(lstMinhaLista1))
```

```
# saída:
```

```
# 5
```

```
# 1
```

# random

- Cria números randômicos (aleatórios)

```
import random
```

```
# Cria uma lista com 10 números aleatórios entre 1 e 100
```

```
lstNumeros = [random.randint(1, 100) for _ in range(10)]
```

```
# Cria uma lista de 10 números distintos entre 1 e 100
```

```
lstUnica = random.sample(range(1, 101), 10)
```

# Exercícios

- 01 – Repetir os códigos
- 02 – Crie uma lista com 10 itens randômicos e encontre o maior deles.

# Experiências e dicas

- Startups – Ideias inovadoras e escaláveis.



Empreendedor potencial  
lapidando uma ideia de  
negócio



Uma startup iterando em  
direção ao encaixe entre  
Problema e Solução,  
validando MVPs enquanto  
busca clientes pagantes



Empresa com um portfólio  
estável de clientes pagantes,  
consistentemente  
melhorando sua operação em  
direção à previsibilidade de  
receita e o encaixe entre  
Produto e Mercado



Empresa em crescimento  
consistente após o encaixe  
entre Produto e Mercado,  
otimizando a gestão para  
acelerar vendas e escalar o  
negócio



Scale-up crescendo 100% ou  
mais ao ano, buscando  
diversificação e consolidaçã

- <https://sebrae.com.br/sites/PortalSebrae/sebraestartups>



# Subvenção de startup

- <https://sebraeforstartups.sebraesp.com.br/fapesp/>



Linhas de subvenção

## Convênio Fapesp

# R\$ 150M

**R\$ 75 M (Sebrae)** Acesso ao mercado  
+ **R\$ 75 M (Fapesp)** Desenvolvimento tecnológico

## Linhas

PIPE Sebrae	PIPE Startup empresa
<b>Tecnologia e mercado</b> até R\$ 1.800.000   24 meses	<b>Tecnologia e mercado</b> até R\$ 500.000   12 meses
Empresas que precisam desenvolver uma nova tecnologia e já realizaram a análise de viabilidade técnico-científica	Empresas que querem validar uma tecnologia já desenvolvida dentro de um ambiente de uma empresa parceira
<a href="#">Saiba mais</a>	<a href="#">Saiba mais</a>

# Dever de casa

1. Por meio de código, crie um arquivo **dados.csv** contendo:

Nome, Idade

Ana,25

Bruno,30

Carla,22

Daniel,28

Eduardo,35

Exemplo de saída esperada:

Digite um nome: Carla

Carla tem 22 anos, não é a pessoa mais velha da lista.

ou

Digite um nome: João

2. Ler o arquivo **dados.csv** e armazene os dados em uma lista.

Peça ao usuário para digitar um nome.

Verifique se o nome digitado está na lista, exiba a idade correspondente e se é a pessoa mais velha ou não da lista.

Caso o nome não esteja na lista, exiba uma mensagem informando isso.

Nome não encontrado.

