

Pensamento Computacional

Estruturas de Dados (Listas)



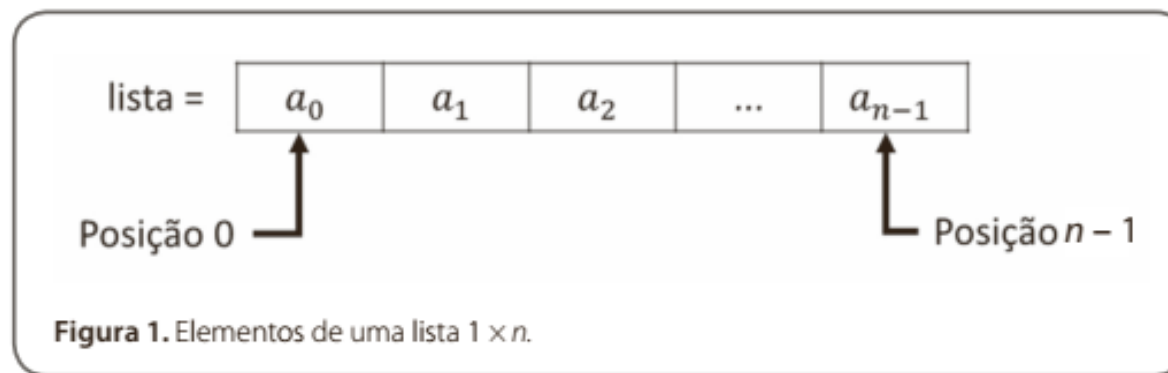
UNIVERSIDADE
CANDIDO
MENDES

EAD ■

■ Estruturas de dados

■ Listas:

- São estruturas utilizadas para armazenamento de vários tipos de dados;
- Também conhecidas como vetores;
- Listas podem ser compreendidas como um conjunto de dados, números e/ou símbolos que são estruturados em uma dimensão (RAMALHO, 2015);
- Além dos dados já citados, listas podem armazenar inclusive objetos (instâncias de classes);
- Dados organizados em um vetor $1 \times n$ (1 linha e n colunas);



■ Estruturas de dados

■ Listas:

- Para criar uma lista utilize colchetes []
- Exemplos:

```
lista01 = [1, 2, 3, 4, 5]
```

```
cesta_de_frutas = ['banana', 'maça', 'pera', 'abacaxi']
```

```
notas_aluno = ['Ricardo', 9.5, 10.0, 8.0]
```

```
lista_heterogenea = ['Pensamento computacional', True, 5.0, None, ['lista', 'dentro', 'outra lista']]
```

■ Estruturas de dados

■ Listas:

- Listas armazenam seus dados organizados sequencialmente e permitem o acesso individual a cada um de seus elementos (RAMALHO, 2015);
- A localização dos elementos (a_j) em uma lista é indicada pelo índice j (coluna);
- Primeira posição possui valor 0 (zero), o segundo elemento está na posição coluna 1 (um), e assim por diante, o último estará na posição $j - 1$;

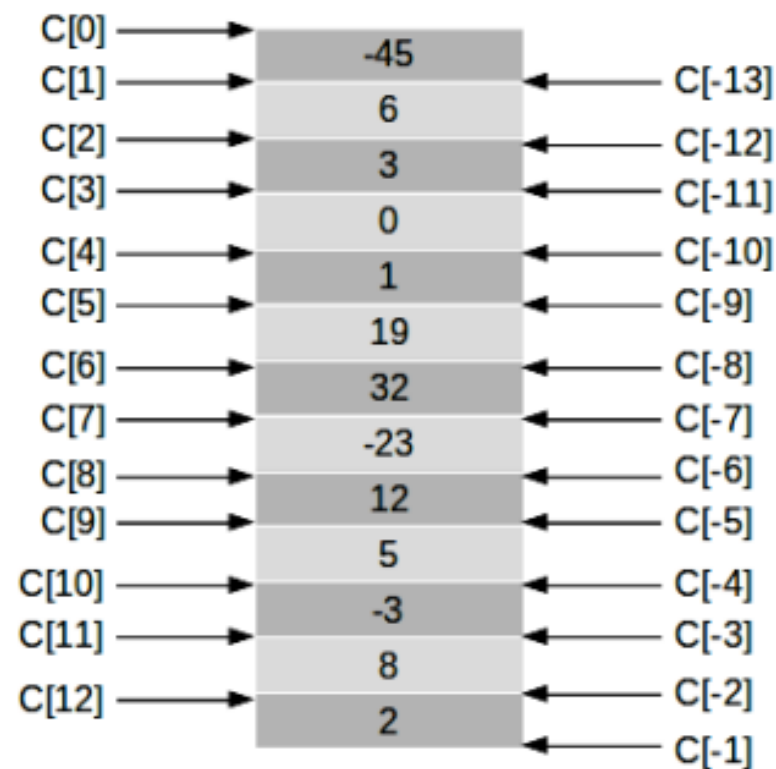
```
notas_aluno = ['Ricardo', 9.5, 10.0, 8.0]
```

Como recuperar valores dentro de uma lista:

```
nome = notas_aluno[0]  
nota_1 = notas_aluno[1]  
nota_3 = notas_aluno[-1]  
notas = notas_aluno[1:4]
```

■ Estruturas de dados

- `lista[0]`
- `lista[1:10]`
- `lista[-13]`
- `lista[-12:-3]`
- `copy()` -> faz uma cópia da lista original
- `append()` -> adiciona um item na lista na última posição;
- `pop()` -> remove um item da lista, de acordo com sua posição, e o retorna;
- `remove()` -> remove um item da lista , de acordo com sua posição;
- `len(lista)` -> retorna o tamanho da lista (quantidade de elementos)



■ Estruturas de dados

■ Listas:

■ Operadores matemáticos com listas:

- O operador de adição “+” é utilizado para concatenar listas, ou seja, esse operador vai unir o conteúdo de uma lista a outra;
- O operador multiplicativo “*” é utilizado para multiplicar x vezes a quantidade de elementos de uma lista; ou seja, esse operador produz uma nova lista com repetições da lista original;
- Operadores matemáticos não soma ou multiplica os elementos pelo valor informado, para isso devemos utilizar estruturas de repetição;

■ Estruturas de dados

```
lista1 = [5, 6, 7, 8, 9]
print(lista1)
lista1 = lista1 + 10
print(lista1)
```

[5, 6, 7, 8, 9]

```
-----
TypeError                                 Traceback (most recent call last)
<ipython-input-32-eadb877aa4a> in <module>
      1 lista1 = [5, 6, 7, 8, 9]
      2 print(lista1)
----> 3 lista1 = lista1 + 10
      4 print(lista1)
```

TypeError: can only concatenate list (not "int") to list

```
lista1 = [5, 6, 7, 8, 9]
print(lista1)
lista1 = lista1 + [10]
print(lista1)
```

[5, 6, 7, 8, 9, 10]



■ Estruturas de dados

```
lista1 = [5, 6, 7, 8, 9]
print(lista1)
lista1 = lista1 * 10
print(lista1)
```

```
[5, 6, 7, 8, 9]
[5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9]
```

```
lista1 = [5, 6, 7, 8, 9]
print(lista1)
lista1 = lista1 * [10]
print(lista1)
```

```
[5, 6, 7, 8, 9]
```

```
-----
TypeError                                Traceback (most recent call last)
<ipython-input-34-459ba41e4c83> in <module>
      1 lista1 = [5, 6, 7, 8, 9]
      2 print(lista1)
----> 3 lista1 = lista1 * [10]
      4 print(lista1)
```

TypeError: can't multiply sequence by non-int of type 'list'



■ Estruturas de dados

```
lista1 = [5, 6, 7, 8, 9]
print(lista1)
lista1 = lista1 * 10
print(lista1)
```

```
[5, 6, 7, 8, 9]
[5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9]
```

```
lista1 = [5, 6, 7, 8, 9]
print(lista1)
lista1 = lista1 * [10]
print(lista1)
```

```
[5, 6, 7, 8, 9]
```

```
-----
TypeError                                Traceback (most recent call last)
<ipython-input-34-459ba41e4c83> in <module>
      1 lista1 = [5, 6, 7, 8, 9]
      2 print(lista1)
----> 3 lista1 = lista1 * [10]
      4 print(lista1)
```

TypeError: can't multiply sequence by non-int of type 'list'



■ Estruturas de dados

■ Listas:

■ Operadores matemáticos com listas:

- Utilizando o for para somar ou multiplicar valores a lista existente

```
# Adicionar o valor 10 em cada elemento da lista  
lista1 = [5, 6, 7, 8, 9]  
lista2 = []  
for item in lista1:  
    lista2.append(item + 10)  
print(lista1)  
print(lista2)
```

```
[5, 6, 7, 8, 9]  
[15, 16, 17, 18, 19]
```

```
lista1 = [5, 6, 7, 8, 9]  
print(lista1)  
lista1 = [item+10 for item in lista1]  
print(lista1)
```

```
[5, 6, 7, 8, 9]  
[15, 16, 17, 18, 19]
```

```
# Multiplica pelo valor 10 em cada elemento da lista  
lista1 = [5, 6, 7, 8, 9]  
lista2 = []  
for item in lista1:  
    lista2.append(item * 10)  
print(lista1)  
print(lista2)
```

```
[5, 6, 7, 8, 9]  
[50, 60, 70, 80, 90]
```

```
lista1 = [5, 6, 7, 8, 9]  
print(lista1)  
lista1 = [item*10 for item in lista1]  
print(lista1)
```

```
[5, 6, 7, 8, 9]  
[50, 60, 70, 80, 90]
```



■ Estruturas de dados

■ Listas:

■ Operadores lógico in:

- Com o operador lógico in é possível verificar se um determinado conteúdo pertence a uma lista. Esse operador retorna True, caso o valor procurado pertença à lista, e False, caso não pertença.

```
cesta_de_frutas = ['banana', 'maça', 'pera', 'abacaxi']  
print('banana está contida na cesta de frutas?')  
print('banana' in cesta_de_frutas)
```

```
banana está contida na cesta de frutas?  
True
```

```
cesta_de_frutas = ['banana', 'maça', 'pera', 'abacaxi']  
print('uva está contida na cesta de frutas?')  
print('uva' in cesta_de_frutas)
```

```
uva está contida na cesta de frutas?  
False
```

```
lista1 = [5, 6, 7, 8, 9]  
print('O valor 7 está contido na lista1?')  
print(7 in lista1)
```

```
O valor 7 está contido na lista1?  
True
```

```
lista1 = [5, 6, 7, 8, 9]  
print('O valor 10 está contido na lista1?')  
print(10 in lista1)
```

```
O valor 10 está contido na lista1?  
False
```



■ Estruturas de dados

■ Biblioteca Numpy

- A biblioteca numpy é um pacote de ferramentas utilizado para executar cálculos matemáticos com arrays multidimensionais (listas e matrizes) em Python (RAMALHO, 2015);
- Contém funções e métodos que auxiliam realizar operações matemáticas em listas e matrizes

```
lista1 = [5, 6, 7, 8, 9]
print(lista1)
print(type(lista1))
lista1 = lista1 + 10
```

```
[5, 6, 7, 8, 9]
<class 'list'>
```

```
-----
TypeError                                 Traceback (most recent call last)
<ipython-input-52-b08656cc756b> in <module>
      2 print(lista1)
      3 print(type(lista1))
----> 4 lista1 = lista1 + 10
```

TypeError: can only concatenate list (not "int") to list

```
import numpy as np
lista1 = np.array([5, 6, 7, 8, 9])
print(lista1)
print(type(lista1))
lista1 = lista1 + 10
print(lista1)
```

```
[5 6 7 8 9]
<class 'numpy.ndarray'>
[15 16 17 18 19]
```



■ Estruturas de dados

```
lista1 = [5, 6, 7, 8, 9]
print(lista1)
print(type(lista1))
lista1 = lista1 * 10
print(lista1)
```

```
[5, 6, 7, 8, 9]
<class 'list'>
[5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9, 5, 6, 7, 8, 9]
```

```
import numpy as np
lista1 = np.array([5, 6, 7, 8, 9])
print(lista1)
print(type(lista1))
lista1 = lista1 * 10
print(lista1)
```

```
[5 6 7 8 9]
<class 'numpy.ndarray'>
[50 60 70 80 90]
```



■ Estruturas de dados

■ Biblioteca Numpy

- Transformar uma lista numpy numa lista padrão:

```
import numpy as np
lista1 = np.array([5, 6, 7, 8, 9])
print(lista1)
print(type(lista1))
lista1 = lista1 * 10
print(lista1)
lista2 = lista1.tolist()
print(type(lista2))
lista2 = lista2 + 10
```

```
[5 6 7 8 9]
<class 'numpy.ndarray'>
[50 60 70 80 90]
<class 'list'>
```

```
-----
TypeError                                Traceback (most recent call last)
<ipython-input-58-23d8d86a1564> in <module>
      7 lista2 = lista1.tolist()
      8 print(type(lista2))
----> 9 lista2 = lista2 + 10
```

TypeError: can only concatenate list (not "int") to list



■ Estruturas de dados

■ Biblioteca Matplotlib

- a biblioteca matplotlib permite trabalhar com as informações registradas em listas e transformá-las em gráficos:

```
1 #Importa a biblioteca matplotlib
2 import matplotlib.pyplot as plt
3 #Cria as listas que representam cada um dos eixos
4 lista_eixo_x = [8,10,12,14,16]
5 lista_eixo_y = [1,9,4,15, 12]
6 #Gera o gráfico na memória do computador
7 plt.plot(lista_eixo_x, lista_eixo_y)
8 #Coloca título e rótulos dos eixos no gráfico
9 plt.title('Vendas no dia DD/MM/AAAA')
10 plt.ylabel('Números de vendas')
11 plt.xlabel('Horas do dia')
12 #Mostra as linhas de grade
13 plt.grid(True)
14 #Mostra o gráfico na tela
15 plt.show()
```



Visite a galeria da Biblioteca para ver vários tipos de gráficos:

<https://matplotlib.org/3.1.1/gallery/index.html>



UNIVERSIDADE
CANDIDO
MENDES

EAD

Pensamento Computacional

Estruturas de Dados (Matrizes)



UNIVERSIDADE
CANDIDO
MENDES

EAD ■

■ Estruturas de dados

■ Matrizes:

- São estruturas utilizadas para armazenamento de vários tipos de dados;
- Matrizes podem ser compreendidas como um conjunto de dados, números e/ou símbolos que são estruturados em duas ou mais dimensões;
- Elas podem ser obtidas a partir da construção de uma lista de listas;
- O índice i (linha) começa no valor 0 e vai até o valor $m - 1$, e o índice j vai de 0 até o $n - 1$

	Coluna 0	Coluna 1	...	Coluna n-2	Coluna n-1
linha 0	$a_{0\ 0}$	$a_{0\ 1}$...	$a_{0\ n-2}$	$a_{0\ n-1}$
linha 1	$a_{1\ 0}$	$a_{1\ 1}$...	$a_{1\ n-2}$	$a_{1\ n-1}$
	\vdots	\vdots	\ddots	\vdots	\vdots
linha m-2	$a_{m-2\ 0}$	$a_{m-2\ 1}$...	$a_{m-2\ n-2}$	$a_{m-2\ n-1}$
linha m-1	$a_{m-1\ 0}$	$a_{m-1\ 1}$...	$a_{m-1\ n-2}$	$a_{m-1\ n-1}$

índice i

índice j



■ Estruturas de dados

■ Matrizes:

- Criação de matrizes:
- `matriz = [lista 1, lista 2, ..., lista N]`
- `matriz = [[0, 1], [2, 3]]`
- O acesso individual a um elemento específico de uma matriz é realizado pelo nome dado à matriz na sua criação e pelos índices, especificando a linha (i) e a coluna (j) entre colchetes.
- `matriz[i][j]`

```
contatos = [['Pedro', '(21) 99900-9900'], ['Maria', '(27) 98765-4321'], ['Joao', '(32) 98989-8989']]  
print(contatos)
```

```
 [['Pedro', '(21) 99900-9900'], ['Maria', '(27) 98765-4321'], ['Joao', '(32) 98989-8989']]
```

```
contatos = [['Pedro', '(21) 99900-9900'], ['Maria', '(27) 98765-4321'], ['Joao', '(32) 98989-8989']]  
print('O número do {} é {}'.format(contatos[0][0], contatos[0][1]))
```

```
O número do Pedro é (21) 99900-9900
```



■ Estruturas de dados

■ Matrizes:

- Utilizando estruturas de repetição para percorrer matrizes:

```
contatos = [['Pedro', '(21) 99900-9900'], ['Maria', '(27) 98765-4321'], ['Joao', '(32) 98989-8989']]  
for linha in contatos:  
    print(linha)
```

```
['Pedro', '(21) 99900-9900']  
['Maria', '(27) 98765-4321']  
['Joao', '(32) 98989-8989']
```

```
contatos = [['Pedro', '(21) 99900-9900'], ['Maria', '(27) 98765-4321'], ['Joao', '(32) 98989-8989']]  
for linha in contatos:  
    print('O número do(a) {} é {}'.format(linha[0], linha[1]))
```

```
O número do(a) Pedro é (21) 99900-9900  
O número do(a) Maria é (27) 98765-4321  
O número do(a) Joao é (32) 98989-8989
```



■ Estruturas de dados

■ Matrizes:

- Utilizando estruturas aninhadas de repetição para percorrer matrizes:

```
contatos = [['Pedro', '(21) 99900-9900'], ['Maria', '(27) 98765-4321'], ['Joao', '(32) 98989-8989']]  
for linha in range(len(contatos)):  
    for coluna in range(len(contatos[linha])):  
        print(contatos[linha][coluna], end = ' ')  
    print('\n')
```

Pedro (21) 99900-9900

Maria (27) 98765-4321

Joao (32) 98989-8989



■ Estruturas de dados

■ Matrízes:

■ Operadores matemáticos com matrizes:

- O operador de adição “+” é utilizado para concatenar matrizes, ou seja, esse operador vai unir o conteúdo de uma matriz a outra;
- O operador multiplicativo “*” é utilizado para multiplicar x vezes a quantidade de linhas de uma matriz, ou seja, esse operador produz uma nova linha com repetições do conteúdo da matriz original;
- Operadores matemáticos não soma ou multiplica os elementos pelo valor informado, para isso devemos utilizar estruturas de repetição;

■ Estruturas de dados

```
(a) 1 #Matriz 2x3 com valores numéricos
    2 num = [[1,2,3], [4,5,6]]
    3 num + num
    4 num + [[7,8,9]]
    5
    6 #Matriz 2x2 com valores literais
    7 mat = [['A','B'], ['C','D']]
    8 mat + mat
    9 mat + [['E','F']]

(b) Out[1]: [[1, 2, 3], [4, 5, 6], [1, 2, 3], [4, 5, 6]]
    Out[2]: [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
    Out[3]: [['A', 'B'], ['C', 'D'], ['A', 'B'], ['C', 'D']]
    Out[4]: [['A', 'B'], ['C', 'D'], ['E', 'F']]
```

■ Estruturas de dados

(a)

```
1 #Matriz 2x3 com valores numéricos
2 num = [[1,2,3], [4,5,6]]
3 num * 3 #triplica as listas dentro da matriz
4 teste = [[0]]*10 #cria um matriz 10x1 com zeros
5 teste_2 = [[2, 3]]*2 #cria um matriz 2x2 com a lista [2, 3]
```

(b)

```
[1]: [[1, 2, 3], [4, 5, 6], [1, 2, 3], [4, 5, 6], [1, 2, 3], [4, 5, 6]]
[2]: [[0], [0], [0], [0], [0], [0], [0], [0], [0], [0]]
[3]: [[2, 3], [2, 3]]
```



■ Estruturas de dados

- `append()` -> adiciona uma linha ou item na linha de uma matriz;
- `pop()` -> remove uma linha ou item de uma linha de uma matriz, retornando o valor;
- `len(matriz)` -> retorna o número de linhas de uma matriz;
- `len(matriz[linha])` -> retorna o número de elementos de uma linha de uma matriz;

■ Estruturas de dados

```
contatos = [['Pedro', '(21) 99900-9900'], ['Maria', '(27) 98765-4321'], ['Joao', '(32) 98989-8989']]
print(contatos)
contatos.append(['Marcelo', '(00) 00000-0000'])
print(contatos)
```

```
[['Pedro', '(21) 99900-9900'], ['Maria', '(27) 98765-4321'], ['Joao', '(32) 98989-8989']]
[['Pedro', '(21) 99900-9900'], ['Maria', '(27) 98765-4321'], ['Joao', '(32) 98989-8989'], ['Marcelo', '(00) 00000-0000']]
```

```
contatos = [['Pedro', '(21) 99900-9900'], ['Maria', '(27) 98765-4321'], ['Joao', '(32) 98989-8989']]
print(contatos)
contatos.pop(1)
print(contatos)
```

```
[['Pedro', '(21) 99900-9900'], ['Maria', '(27) 98765-4321'], ['Joao', '(32) 98989-8989']]
[['Pedro', '(21) 99900-9900'], ['Joao', '(32) 98989-8989']]
```

```
contatos = [['Pedro', '(21) 99900-9900'], ['Maria', '(27) 98765-4321'], ['Joao', '(32) 98989-8989']]
print(contatos)
contatos[1].pop(1)
print(contatos)
```

```
[['Pedro', '(21) 99900-9900'], ['Maria', '(27) 98765-4321'], ['Joao', '(32) 98989-8989']]
[['Pedro', '(21) 99900-9900'], ['Maria'], ['Joao', '(32) 98989-8989']]
```



■ Estruturas de dados

```
: contatos = [['Pedro', '(21) 99900-9900'], ['Maria', '(27) 98765-4321'], ['Joao', '(32) 98989-8989']]  
print('Número de linhas:', len(contatos))  
print('Número de elementos:', len(contatos[2]))
```

Número de linhas: 3

Número de elementos: 2

Pensamento Computacional

Leitura de Arquivos



UNIVERSIDADE
CANDIDO
MENDES

EAD ■

■ Leitura de Arquivos

- Para abrir um arquivo, a linguagem Python conta com o comando `open`, combinado com o modo de abertura desse arquivo:

MODO DE ABERTURA	DESCRIÇÃO
'r'	Abre o arquivo somente para a leitura do seu conteúdo
'w'	Abre o arquivo para escrita a partir do início do arquivo
'a'	Abre o arquivo para escrita a partir do final do arquivo
't'	Abre o arquivo como um arquivo de texto
'b'	Abre o arquivo como um arquivo binário

```
arquivo = open("ArquivoExemplo.txt", 'rt')
```



■ Leitura de Arquivos

- Erro ao não encontrar o arquivo no diretório indicado:

```
arquivo = open("ArquivoInexistente.txt", 'rt')
```

```
-----  
FileNotFoundError                                Traceback (most recent call last)  
<ipython-input-1-c55e352c21bb> in <module>  
----> 1 arquivo = open("ArquivoInexistente.txt", 'rt')  
  
FileNotFoundError: [Errno 2] No such file or directory: 'ArquivoInexistente.txt'
```

- É possível indicar o diretório ou subdiretório:

- Exemplo:

```
arquivo_texto = open("\\<caminho_do_subdiretorio>\\<nome_do_arquivo>", 'rt')
```



■ Leitura de Arquivos

- `readlines()` -> irá retornar uma lista contendo onde cada elemento corresponde uma linha do arquivo texto;
- `readline()` -> irá retornar uma string contendo uma linha do arquivo texto. Essa linha irá mudar de acordo com a quantidade de vezes que o comando for acionado;

```
arquivo = open('arquivo_texto.txt', 'rt')
texto = arquivo.readlines()
print(texto)
print(type(texto))
```

```
['Bem vindo a aula de Pensamento Computacional\n', 'Universidade: UCAM\n', 'Modalidade: EAD\n', 'Curso: Desenvolvimento de Software\n']
<class 'list'>
```

■ Leitura de Arquivos

```
arquivo = open('arquivo_texto.txt', 'rt')
texto = arquivo.readline()
print(texto)
print(type(texto))
texto = arquivo.readline()
print(texto)
print(type(texto))
texto = arquivo.readline()
print(texto)
print(type(texto))
texto = arquivo.readline()
print(texto)
print(type(texto))
```

Bem vindo a aula de Pensamento Computacional

<class 'str'>
Universidade: UCAM

<class 'str'>
Modalidade: EAD

<class 'str'>
Curso: Desenvolvimento de Software

<class 'str'>



UNIVERSIDADE
CANDIDO
MENDES

EAD

■ Leitura de Arquivos

- Os arquivos separados por vírgulas (csv) são arquivos texto com vários dados separados, como se fossem uma tabela;
- A linguagem Python possui o módulo csv, que facilita o processo de leitura e interpretação dos campos separados por vírgulas;
- Utilize o comando import csv;
- leitorCSV = csv.reader(arquivoCSV)

```
import csv

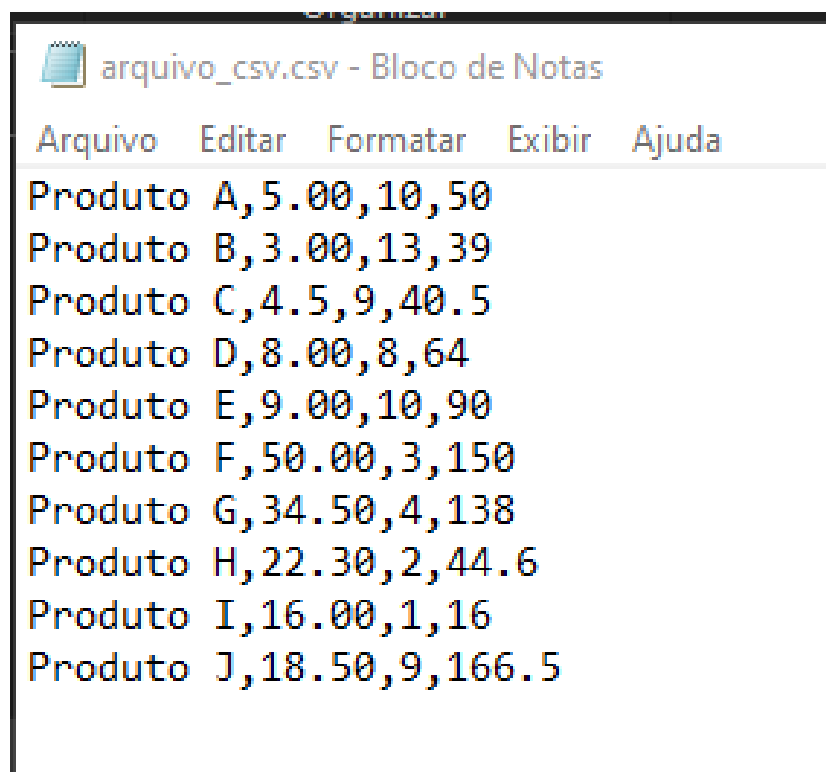
arquivoCSV = open("ArquivoCSVExemplo.csv", 'rt')
reader = csv.reader(arquivoCSV, delimiter=',')
for linha in reader:
    print(linha)

['ColunaA', ' ColunaB', ' ColunaC']
['campo 1', ' 123.01', ' Texto um pouco mais longo']
['Campo 2', ' -123.99', ' Texto curto']
[' ', ' 0', ' Linha com a coluna A em branco']
```



■ Leitura de Arquivos

- Utilizando o for para ler os valores do arquivo CSV:



```
arquivo_csv.csv - Bloco de Notas
Arquivo  Editar  Formatar  Exibir  Ajuda
Produto A,5.00,10,50
Produto B,3.00,13,39
Produto C,4.5,9,40.5
Produto D,8.00,8,64
Produto E,9.00,10,90
Produto F,50.00,3,150
Produto G,34.50,4,138
Produto H,22.30,2,44.6
Produto I,16.00,1,16
Produto J,18.50,9,166.5
```

```
import csv
arquivo_csv = open('pasta_csv\\arquivo_csv.csv', 'rt')
leitor_csv = csv.reader(arquivo_csv)
soma_total = 0
for linha in leitor_csv:
    print(float(linha[3]))
    soma_total += float(linha[3])
print('O valor total das vendas foi {}'.format(soma_total))
```

50.0
39.0
40.5
64.0
90.0
150.0
138.0
44.6
16.0
166.5
O valor total das vendas foi 798.6





UNIVERSIDADE
CANDIDO
MENDES

EAD ■

