



PARADIGMAS DE LINGUAGENS DE PROGRAMAÇÃO EM PYTHON

TIPOS DE DADOS BÁSICOS EM PYTHON

Prof. Dr. Daniel Caetano

2020 - 2

Compreendendo o problema

- **Situação:** aplicação de registro de temperatura
 - Vários dias/horas, no formato:

Dia/Hora	08:00	12:00	16:00	20:00
01/01/2020	25.0	32.5	30.5	28.0
02/01/2020	26.0	31.6	32.5	29.5

- Tipos primitivos são suficientes? Qual?



<https://www.menti.com/>

Compreendendo o problema

- **Situação:** aplicação de registro de temperatura
 - Vários dias/horas, no formato:

Dia/Hora	08:00	12:00	16:00	20:00
01/01/2020	25.0	32.5	30.5	28.0
02/01/2020	26.0	31.6	32.5	29.5

- Erros de precisão são relevantes?



<https://www.menti.com/>

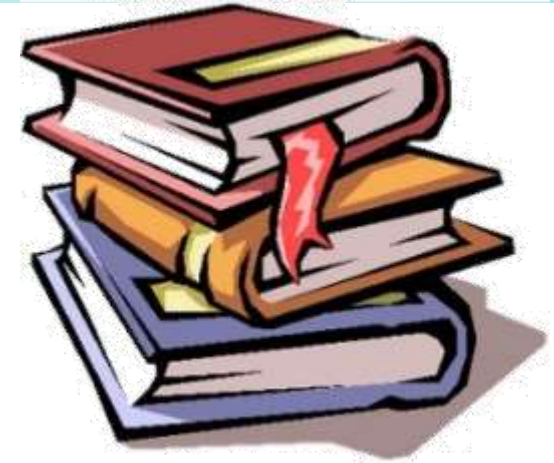
Objetivos

- Conceituar os tipos primitivos
- Compreender as cadeiras de caracteres
- Conhecer e compreender as listas/vetores
- Conhecer e compreender as matrizes
- Compreender as matrizes associativas

- **Desafio Aula 06**



Bibliografia da Aula



Material	Acesso ao Material
Apresentação	https://www.caetano.eng.br/ (Paradigmas de Programação – Aula 6)
Livro Texto	Capítulo 6, páginas 233 a 262
Aprenda Mais!	<ul style="list-style-type: none">• Vídeo: Strings https://www.youtube.com/watch?v=DdhNItkI_hE• Vídeo: Matrizes https://www.youtube.com/watch?v=c9yiwWNiNQw

TIPOS PRIMITIVOS



Mentimeter

<https://www.menti.com/>

Prof. Dr. Daniel Caetano

Tipos Primitivos

- São os tipos básicos de uma linguagem
 - Implementados "nativamente" na linguagem
- São três as categorias de tipos mais comuns:
 - Numéricos
 - Lógicos (Booleanos)
 - Caracteres
- Vejamos detalhes de cada um deles
 - Em C++/C#, o tamanho é dado por **sizeof(x)**
 - Em Python, o tipo é dado por **type(x)**
 - Em Python, o tamanho é dado por **sys.getsizeof(x)**

Tipos Numéricos: Inteiros

- Todas as linguagens, algumas em várias versões
 - Com e sem sinal, mais ou menos memória...
- Nomes comuns: int, long
- Em geral, negativos em complemento de dois
 - $0 - 1 = -1 \rightarrow 00000000b - 00000001b = 11111111b$
 - $0 - 2 = -2 \rightarrow 00000000b - 00000010b = 11111110b$

```
#include <iostream>
using namespace std;
int main() {
    int i = 3000000000;
    unsigned int u = 3000000000;
    cout << i << endl << u << endl;
}
```

```
i = 3000000000
u = 3000000000
print (i)
print (u)
```


Tipos Numéricos: Ponto Flutuante

- Nem todas; quando há, em geral em duas versões
 - float e double
- Em geral, seguem IEEE 754/2008
 - Mantissa + Expoente: $1,011 * 2^{-110}$

Sinal	Sinal Exp.	Expoente	Mantissa
Bit 31	Bit 30	Bit 29 ~ Bit 23	Bit 22 ~ Bit 0
Bit 63	Bit 62	Bit 61 ~ Bit 52	Bit 51 ~ Bit 0

```
#include <iostream>
using namespace std;
int main() {
    float f = 3.141592356589793;
    double d = 3.141592356589793;
    cout.precision(20);
    cout << f << endl << d << endl;
}
```

```
f = 3.141592356589793
d = 3.141592356589793
print (f)
print (d)
```

Tipos Numéricos: Complexos

- Poucas linguagens (foco: engenharia, física)
 - complex
- Em Python têm uma representação bem específica:
 - (real + imaginário j): (20.5+10j)

```
c = (20.5+10j)  
print (c)
```

Tipos Numéricos: Decimais

- Poucas linguagens possuem (comerciais)
 - COBOL, C# e F#
 - decimal
- Armazenados como Binary Coded Decimal
 - 1 ou 2 dígitos por byte
 - Custoso computacionalmente
 - Nem sempre permitem expoentes

```
using System;
class Program {
    static void Main() {
        decimal d = 0.1m;
        Console.WriteLine(d);
    }
}
```

Tipos Numéricos: Decimais

- Exemplo prático em C#

```
using System;
class Program {
    static void Main() {
        decimal total = 0.0m;
        decimal termo = 0.1m;
        for (int i=0; i<1000; i++) {
            total = total + termo;
        }
        Console.WriteLine(total);
    }
}
```

```
using System;
class Program {
    static void Main() {
        float total = 0.0f;
        float termo = 0.1f;
        for (int i=0; i<1000; i++) {
            total = total + termo;
        }
        Console.WriteLine(total);
    }
}
```

Lógicos

- A maioria das linguagens possui: false x true
 - boolean, bool...
- Em geral: resultado de operações lógicas
- Poderiam ocupar um único bit...
 - Mas, por velocidade, costumam ocupar bem mais!

```
b = True  
print (b)
```

```
c = 5 >= 7  
print (c)
```

Caracteres

- A maioria das linguagens possui
 - char
- O tamanho varia bastante
 - 1 byte (ASCII, ISSO 8859-1, UTF-8)
 - 2 bytes (S-JIS, UTF-16)
 - 4 bytes (UTF-32)
- Em Python não há esse tipo primitivo

```
#include <iostream>
using namespace std;
int main() {
    char c = 'A';
    cout << c << endl;
}
```



<https://www.menti.com/>



CADEIAS DE CARACTERES: AS STRINGS

Strings

- Cadeias de caracteres (strings) guardam **textos**
 - char[], String...
- Na memória: apontam início de sequência de chars

	0	1	2	3	4	5	6	7	
8	A	b	a	c	a	x	i	!	\0

- Como sabe se acabou?
 - **Descritor** ou **Terminador**
- Podem ser de três tipos
 - Estáticas/imutáveis (Java, Python, C#...)
 - Comprimento dinâmico limitado (C)
 - Comprimento dinâmico ilimitado (JavaScript, Perl...)

Strings

- Considerando a forma de armazenamento

0	1	2	3	4	5	6	7
A	b	a	c	a	x	i	!

- Algumas linguagens permitem nativamente...
 - O acesso aos caracteres

```
#include <iostream>
using namespace std;
int main() {
    char texto[] = "Abacaxi!";
    cout << texto[3];
}
```

```
texto = "Abacaxi! "
print (texto[3])
print (texto[-1])
```

Strings

- Algumas linguagens "operam" com strings
 - Ex. Python!
- Outras, exigem o uso de funções...
 - Ex.: C!

```
A = "Um "  
B = "texto!"  
print (A+B)
```

```
#include <iostream>  
#include <string.h>  
using namespace std;  
int main() {  
    char A[] = "Um ";  
    char B[] = "texto!";  
    cout << strcat(A,B);  
}
```

LISTAS, VETORES E MATRIZES



Mentimeter

<https://www.menti.com/>

Prof. Dr. Daniel Caetano

Listas, Vetores e Matrizes

- No geral: aglomerados simples de dados
- Dependendo da linguagem, podem ser:
 - Uniformes: C, C++, Java, C#...
 - Mistos: Python, JavaScript, Ruby...
- Cada elemento: acessado por sua posição
 - Em C, a String é, na verdade, um vetor de char!
- Vejamos com mais detalhe em Python!



LISTAS/VETORES EM PYTHON

O que é uma lista?

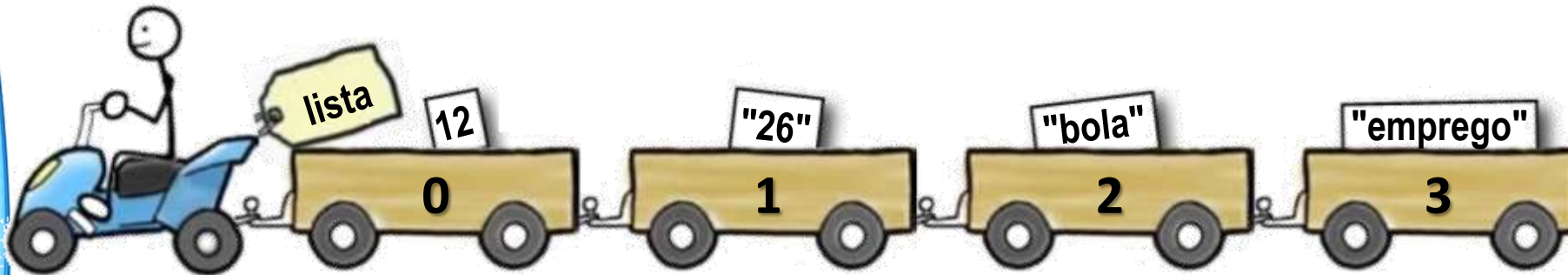
- É um conjunto de dados encadeados
 - Uma lista é como um trem de dados:

Cada vagão guarda uma informação!



O que é uma lista?

- **lista** é uma única variável, com vários dados
 - Os vagões são numerados



```
>>> print( lista[1] )  
26  
>>> print( lista[3] )  
emprego
```

As listas são também os **vetores** do Python

Criando uma lista no console

- Podemos criar uma lista vazia:

Console

```
>>> lista = []
```


Criando uma lista no console

- Podemos criar uma lista vazia:

Console

```
>>> lista = []
```

Qual é o tipo?

Criando uma lista no console

- Podemos criar uma lista já preenchida:

Console

```
>>> lista = [12, "26", "bola", "emprego"]
```

Criando uma lista no console

- Podemos mostrar o conteúdo da lista:

Console

```
>>> print( lista )
```

Usando uma lista no console

- Podemos obter o tamanho da lista

Console

```
>>> N = len( lista )  
>>> print( N )
```

Usando uma lista no console

- Podemos acrescentar um item na lista

Console

```
>>> lista.append( 33 )
```

Usando uma lista no console

- Podemos mudar um valor da lista

Console

```
>>> print ( lista[3] )  
emprego  
>>> lista[3] = "praia"  
>>> print ( lista[3] )  
praia
```

Usando uma lista no console

- Podemos remover um elemento da lista

Console

```
>>> print ( lista )  
[12, '26', 'bola', 'praia', 33]  
>>> lista.remove( "26" )  
>>> print ( lista )  
[12, 'bola', 'praia', 33]
```

Usando uma lista no console

- Podemos remover um elemento da lista

Console

```
>>> print ( lista )  
[12, 'bola', 'praia', 33]  
>>> lista.pop( 1 )  
>>> print ( lista )  
[12, 'praia', 33]
```


Usando uma lista no console

- Podemos juntar listas

Console

```
>>> print ( lista )  
[12, 'praia', 33]  
>>> lista2 = [ "Mais", "Elementos" ]  
>>> lista = lista + lista2  
>>> print ( lista )  
[12, 'praia', 33, 'Mais', 'Elementos' ]
```

Usando uma lista no console

- Podemos percorrer os elementos da lista

Console

```
>>> N = 0
>>> while N < 5:
    print( lista[N] )
    N = N + 1
```

Usando uma lista no console

- Podemos percorrer os elementos da lista

Console

```
>>> for N in range(0,5):  
    print( lista[N] )
```

Usando uma lista no console

- Podemos percorrer os elementos da lista

Console

```
>>> for EL in lista:  
    print( EL )
```

Exemplo

- Vamos completar o programa abaixo para calcular a média da turma:

```
aula06ex01.py
```

```
# Calcula a média das notas
```

```
NOTAS = [ 8.0, 5.5, 7.5, 6.0, 3.2 ]
```

Exemplo

- Vamos completar o programa abaixo para calcular a média da turma:

```
aula06ex01.py
```

```
# Calcula a média das notas

NOTAS = [ 8.0, 5.5, 7.5, 6.0, 3.2 ]

SOMA = 0
for NOTA in NOTAS:
    SOMA = SOMA + NOTA
MEDIA = SOMA / len(NOTAS)
print( "A média é ", MEDIA)
```

Exemplo 2

- Modifiquemos o programa abaixo para que o usuário possa digitar 5 notas:

`aula06ex01.py`

```
# Calcula a média das notas

NOTAS = [ 8.0, 5.5, 7.5, 6.0, 3.2 ]

SOMA = 0
for NOTA in NOTAS:
    SOMA = SOMA + NOTA
MEDIA = SOMA / len(NOTAS)
print( "A média é ", MEDIA)
```

Exemplo 2

- Modifiquemos o programa abaixo para que o usuário possa digitar 5 notas:

aula06ex01.py

```
# Calcula a média das notas

NOTAS = []
for N in range(5):
    NOTA = float( input("Digite uma nota: ") )
    NOTAS.append( NOTA )
SOMA = 0
for NOTA in NOTAS:
    SOMA = SOMA + NOTA
MEDIA = SOMA / len(NOTAS)
print( "A média é ", MEDIA)
```

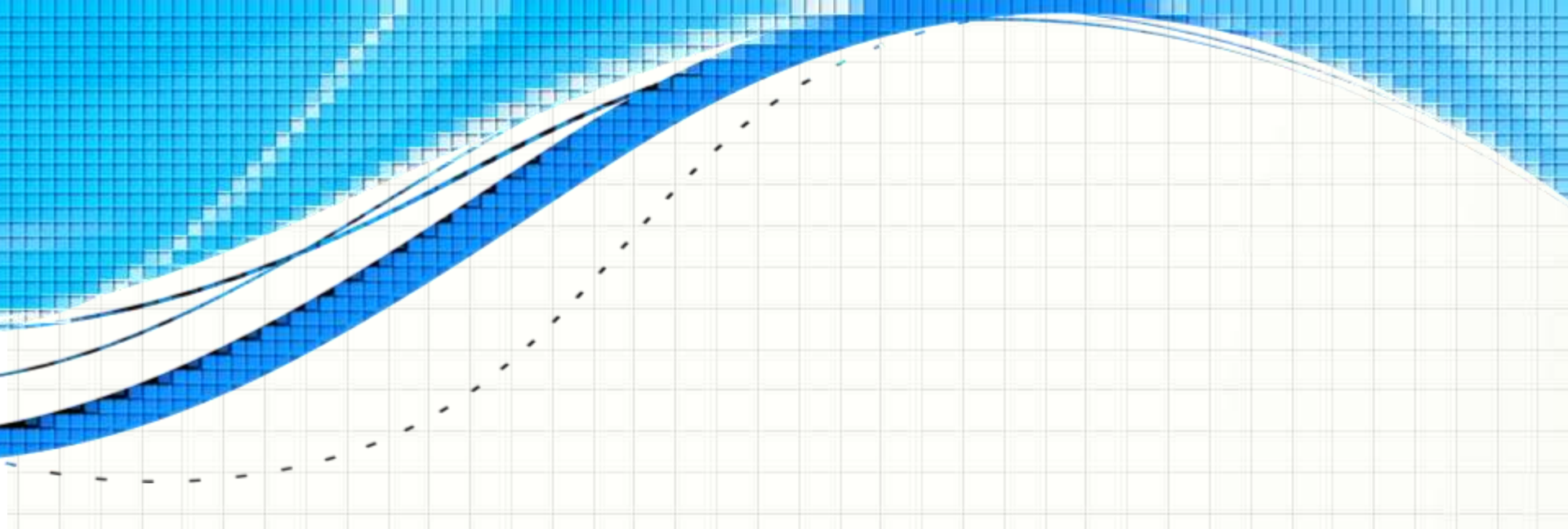



LISTAS ASSOCIATIVAS

Listas Associativas

- Listas e vetores comuns: posição numérica
- Às vezes, é conveniente dar nome às posições
 - Para tradução, por exemplo
- Essas são as listas associativas!
 - Presentes em Python e outras linguagens!

```
cores = { "red" : "vermelho", "green" : "verde", "blue" : "azul" }  
cores_fr = { "red" : "rouge", "green" : "vert", "blue" : "bleu" }  
  
print ( cores["green"] )  
print ( cores_fr["green"] )
```



MATRIZES EM PYTHON

TAMBÉM CONHECIDAS COMO LISTAS DE LISTAS

Matrizes

- Podemos ter uma lista de listas
 - Cada elemento da lista... É outra lista

Console

```
>>> matriz = [ [ 0, 1, 2 ],  
                [ 3, 4, 5 ],  
                [ 6, 7, 8 ] ]  
  
>>> print( matriz )
```

Matrizes

- Podemos imprimir uma linha da lista

Console

```
>>> print( matriz[0] )
```

Matrizes

- Podemos imprimir um elemento da lista

Console

```
>>> print( matriz[0][2] )
```

Matrizes

- Podemos imprimir um elemento da lista
 - **Cuidado!**

Console

```
>>> print( matriz[0,2] )
```

Matrizes

- Podemos percorrer as linhas da matriz

Console

```
>>> for X in matriz:  
        print( X )
```


Matrizes

- Podemos percorrer todos os elementos da matriz

Console

```
>>> for linha in matriz:  
        for valor in linha:  
            print( valor )
```

Matrizes

- Melhorando o visual...
 - Teste essa variação!

Console

```
>>> for linha in matriz:  
        for valor in linha:  
            print( valor )  
        print("----")
```

Exemplo

- Crie a agenda abaixo e faça com que ela seja impressa formatada:

Nome: Daniel

Telefone: 11-5555-1234

Data Nasc.: 10/02/1973

aula14ex01.py

```
# Imprime agenda
```

```
agenda = [ ["Daniel", "11-5555-1234", "10/02/1973"],  
            ["João", "11-5555-2345", "02/10/1980"],  
            ["Alberto", "11-5555-3456", "11/01/1987"]]
```

Exemplo

- Crie a agenda abaixo e faça com que ela seja impressa formatada:

Nome: Daniel
Telefone: 11-5555-1234
Data Nasc.: 10/02/1973

aula14ex01.py

```
# Imprime agenda
```

```
agenda = [ ["Daniel", "11-5555-1234", "10/02/1973"],  
           ["João", "11-5555-2345", "02/10/1980"],  
           ["Alberto", "11-5555-3456", "11/01/1987"]]
```

```
for contato in agenda:
```

```
    print("Nome:", contato[0])  
    print("Telefone:", contato[1])  
    print("Data Nasc.:", contato[2])  
    print("-----")
```

Exemplo

- Cadastro de Lista de Notas
 - Perguntar nome do aluno
 - Perguntar nota
 - Quando nome vazio for digitado, finalizar
 - Ao final, imprimir
 - a média
 - A lista de alunos com nota acima da média
 - Incluir nome e média

Exemplo

- Cadastro de Lista de Notas
 - Nome, nota... Para com nome vazio
 - Média e imprimir alunos acima da média

aula06ex02.py

```
# Cadastro de notas de alunos
TURMA = []
while True:
    NOME = input("Nome: ")
    if len(NOME) == 0 :
        break
    NOTA = float(input("Nota: "))
    TURMA.append([NOME, NOTA])
```

aula06ex02.py (cont)

```
TOTAL = 0
for ALUNO in TURMA :
    TOTAL = TOTAL + ALUNO[1]
MEDIA = TOTAL / len(TURMA)
print("Média:", MEDIA)
for ALUNO in TURMA:
    if ALUNO[1] > MEDIA :
        print(ALUNO[0], " - ", ALUNO[1])
```



ATIVIDADE

Atividade 1

- Individual – 5 minutos
- Faça um Programa que leia DUAS strings, imprima:
 - O conteúdo e o comprimento de cada uma delas
 - Se as duas strings possuem o mesmo tamanho
 - Se as duas strings são iguais ou diferentes.

Atividade 2

- Individual – 5 minutos
- Faça um programa que solicite o nome do usuário e o escreva na vertical. Exemplo:

D
A
N
I
E
L

Atividade 3

- Individual 5 minutos
- Altere o programa da atividade anterior para que ele imprima o nome em uma “escadinha”.
Exemplo:

D
DA
DAN
DANI
DANIE
DANIEL

Atividade 4

- Individual, em Python – 5 minutos
- Faça um programa que leia 5 valores inteiros em um vetor/lista e depois mostre-os.

Atividade 5

- Individual, em Python – 5 minutos
- Modifique o programa anterior para que leia 10 números no vetor, e calcule e mostre a soma dos quadrados dos elementos do vetor.



ENCERRAMENTO

Resumo e Próximos Passos

- Tipos nativos de várias linguagens
 - E seus tamanhos na memória
 - Strings e sua estrutura
 - Listas/Vetores e Listas Associativas
 - Matrizes
 - **Pós Aula:** Aprenda Mais, Pós Aula e Desafio!
 - No padlet: <https://padlet.com/djcaetano/paradigmas>
-
- Aglomerados de dados em geral: registros
 - Structs, Listas, Tuplas e Uniões



PERGUNTAS?