

Lógica de Programação II

Vetores

Exemplo Motivacional

- Programa para auxiliar a escrever os nomes dos alunos que tiveram nota acima da média de uma disciplina com 3 alunos
 - Ler os nomes e as notas de 3 alunos
 - Calcular a média da turma
 - Listar a média
 - Listar os alunos que tiveram nota acima da média

Exemplo Motivacional

```
nome1 = input('Informe o nome do aluno 1: ')
nome2 = input('Informe o nome do aluno 2: ')
nome3 = input('Informe o nome do aluno 3: ')

nota1 = eval(input('Informe a nota de ' + nome1 + ':'))
nota2 = eval(input('Informe a nota de ' + nome2 + ':'))
nota3 = eval(input('Informe a nota de ' + nome3 + ':'))

media = (nota1 + nota2 + nota3)/3
print('A media da turma foi', media, '\n' )
print('Alunos que tiveram nota maior que a média: \n')

if nota1 > media:
    print(nome1, '\n')
if nota2 > media:
    print(nome2, '\n')
if nota3 > media:
    print(nome3, '\n')
```

E se fossem 40 alunos?

- É possível definir variáveis que guardam mais de um valor de um mesmo tipo
- Essas variáveis são conhecidas como variáveis compostas, variáveis subscritas, variáveis indexáveis ou arranjos (*array*)
- Em Python existem três tipos principais de variáveis compostas:
 - Listas
 - Tuplas
 - Dicionários

Vetores

- Variável composta **unidimensional**
 - Contém espaço para armazenar diversos valores
 - É acessada via um índice
- A ideia de vetor é comum na matemática, com o nome de variável subscrita
 - Exemplo: x_1, x_2, \dots, x_n
- O que vimos até agora são variáveis com somente um valor
 - Exemplo: $y = 123$
- No caso de vetores, uma mesma variável guarda ao mesmo tempo múltiplos valores
 - Exemplo: $x_1 = 123, x_2 = 456, \dots$
 - $x = [123, 456, \dots]$

Listas

- Em outras linguagens de programação, listas são chamadas de **vetores** e possuem restrições que Python não impõe:
 - Em Python, os valores de uma lista podem ser de qualquer tipo
 - Em outras linguagens, os valores precisam ser do mesmo tipo
- Em Python
 - `lista = ['A', 1, 2, 'Casa', 2.3]`
 - `notas = [10, 5, 6.7, 2, 7.5]`

Utilização de listas

- Para acessar (ler ou escrever) uma posição do vetor, basta informar a posição entre colchetes

```
notas = [8, 5.5, 1.5]  
media = (notas[0] + notas[1] + notas[2]) / 3
```

notas	0	8.0
	1	5.5
	2	1.5
media		5.0

Utilização de listas

- Pode-se iterar por todos os seus valores usando um comando **for**

```
notas = [8, 5.5, 1.5]
for i in range(3):
    print(notas[i])
```


Criação de uma lista a partir de valores lidos do teclado

- Armazenar as notas de 3 alunos em uma lista. A nota de cada aluno será informada pelo teclado.

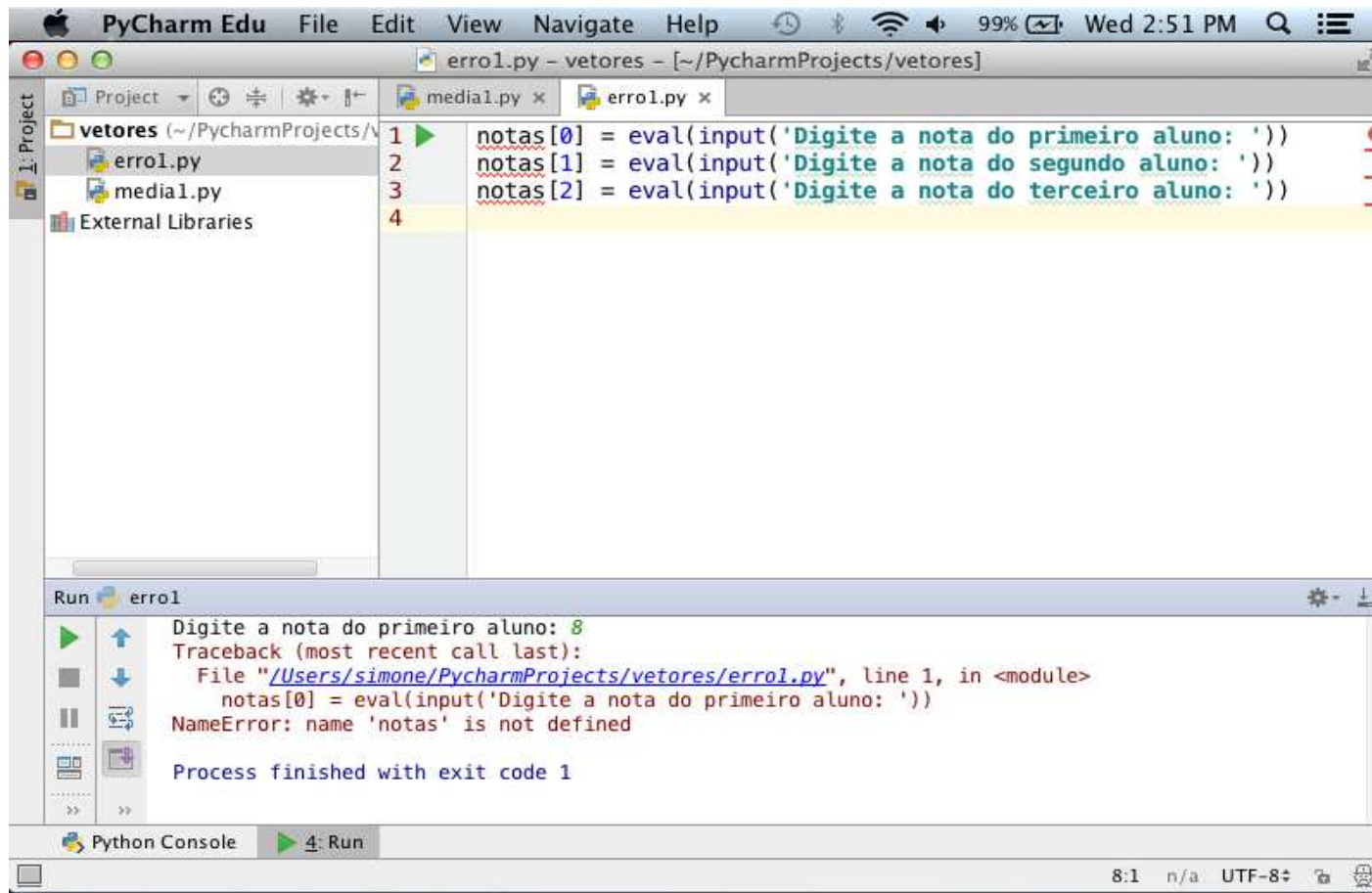
```
notas[0] = eval(input('Digite a nota do primeiro aluno: '))
notas[1] = eval(input('Digite a nota do segundo aluno: '))
notas[2] = eval(input('Digite a nota do terceiro aluno: '))
```

Criação de uma lista a partir de valores lidos do teclado

- Armazenar as notas de 3 alunos em uma lista. A nota de cada aluno será informada pelo teclado.

```
notas[0] = eval(input('Digite a nota do primeiro aluno: '))  
notas[1] = eval(input('Digite a nota do segundo aluno: '))  
notas[2] = eval(input('Digite a nota do terceiro aluno: '))
```

Criação de uma lista a partir de valores lidos do teclado



The screenshot displays the PyCharm Edu IDE interface. The top menu bar includes File, Edit, View, Navigate, and Help. The title bar shows the project name 'vetores' and the file 'erro1.py'. The left sidebar shows the project structure with 'vetores' containing 'erro1.py' and 'medial.py'. The main editor window shows the following code in 'erro1.py':

```
1 notas[0] = eval(input('Digite a nota do primeiro aluno: '))
2 notas[1] = eval(input('Digite a nota do segundo aluno: '))
3 notas[2] = eval(input('Digite a nota do terceiro aluno: '))
4
```

The bottom panel shows the Run console output for 'erro1'. It displays the prompt 'Digite a nota do primeiro aluno: 8' and a traceback error:

```
Traceback (most recent call last):
  File "/Users/simone/PycharmProjects/vetores/erro1.py", line 1, in <module>
    notas[0] = eval(input('Digite a nota do primeiro aluno: '))
NameError: name 'notas' is not defined

Process finished with exit code 1
```

The status bar at the bottom indicates the file encoding is UTF-8.

É preciso primeiro criar a lista...

- Como não sabemos o que colocar em cada posição da lista, vamos criar uma lista vazia

```
notas = []
```

- Depois vamos adicionar valores na lista usando **append**

```
n = eval(input('Digite a nota do primeiro aluno:'))  
notas.append(n)
```

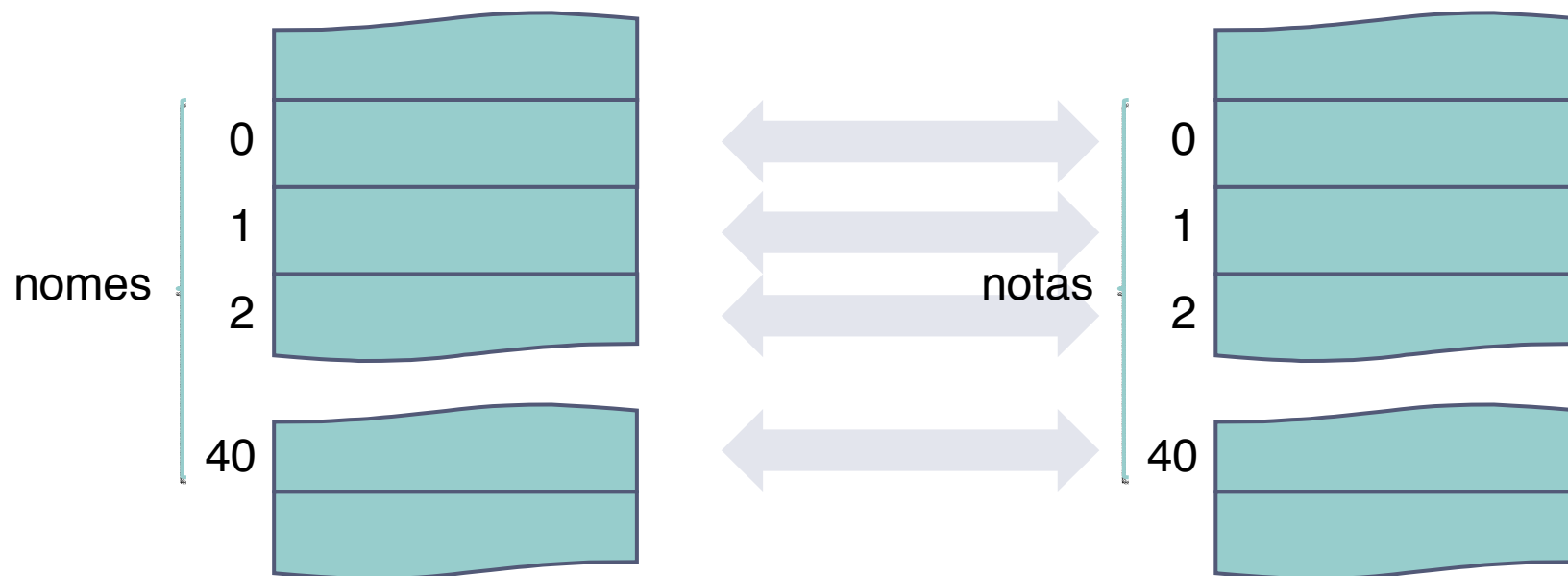
Voltando ao exemplo

- Armazenar as notas de 3 alunos em uma lista. A nota de cada aluno será informada pelo teclado.

```
notas = []
notas.append(eval(input('Digite a nota do primeiro aluno: ')))
notas.append(eval(input('Digite a nota do segundo aluno: ')))
notas.append(eval(input('Digite a nota do terceiro aluno: ')))
print(notas)
```

Retomando: E se fossem 40 alunos?

- Criaríamos dois vetores (nomes e notas) de 40 posições
- Vincularíamos a posição N do vetor de nomes à posição N do vetor de notas



Retomando: E se fossem 40 alunos?

```
num_alunos = 40
nomes = []
notas = []
media = 0

for i in range(num_alunos):
    nomes.append(input('Informe o nome do aluno: '))
    notas.append(eval(input('Informe a nota de ' + nomes[i] +
        ': ')))
    media = media + notas[i]

media = media / num_alunos
print('A media da turma foi ', media)
print('Alunos que tiveram nota maior que a média: \n')

for i in range(num_alunos):
    if notas[i] > media:
        print(nomes[i])
```

Cuidados no uso de listas

- Certifique-se de que não esteja querendo acessar posição da lista que não existe
- Exemplo:

```
alunos = ['Andre', 'Lucas', 'Antonio', 'Maria']  
print(alunos[4])
```


Índices para acesso aos elementos da lista

- Python permite acesso à lista em ordem crescente ou decrescente de posição
 - Primeira posição é 0
 - Última posição é -1

```
>>> c = [-45, 6, 0, 72, 1543]
>>> c[3]
72
>>> c[-2]
72
>>> c[0] == c[-5]
True
```

c[0]	-45	c[-5]
c[1]	6	c[-4]
c[2]	0	c[-3]
c[3]	72	c[-2]
c[4]	1543	c[-1]

Funções de manipulação de listas

- `len(lista)`
 - Retorna o tamanho da lista

```
>>> numeros = [3,1,6,7,10,22,4]  
>>> len(numeros)  
7
```

Exemplo

- Programa que lê uma lista do teclado, soma 1 aos elementos da lista e imprime a lista resultante

```
continua = 's'
lista = []
while (continua == 's' or continua == 'S'):
    n = eval(input('Digite um numero: '))
    lista.append(n)
    continua = input('Deseja continuar? (s/n): ')
print(lista)
for i in range(len(lista)):
    lista[i] = lista[i] + 1
print(lista)
```

Concatenação de listas

- É possível anexar os valores de uma lista em outra usando o operador “+”

```
>>> lista = [1,2,3]
>>> lista + [4]
[1,2,3,4]
>>> lista + [4,5,6]
[1,2,3,4,4,5,6]
```

Exemplo

- Programa que retorna uma lista com todos os números pares entre 2 e um número n, inclusive

```
n = eval(input('Digite um numero: '))
lista = []
for i in range(2,n+1,2):
    lista = lista + [i]
print(lista)
```

Exemplo

- Programa que retorna uma lista com todos os números pares entre 2 e um número n, inclusive, **em ordem reversa**

```
n = eval(input('Digite um numero: '))
lista = []
for i in range(2,n+1,2):
    lista = [i] + lista
print(lista)
```

“Multiplicação” de listas

- O operador “*” repete **n** vezes os elementos que já estão na lista
- **lista * n** equivale a **lista + lista + ... + lista** (n vezes)

```
>>> lista = [1,2,3]
>>> lista * 3
[1,2,3,1,2,3,1,2,3]
```

Inicialização de listas com zero

- Em diversas situações onde já sabemos de antemão qual será o tamanho da lista, é útil inicializar a lista com o valor 0. Isso evita que precisemos usar o append para adicionar valores

```
>>> tamanho = 10
>>> lista = [0] * tamanho
>>> lista
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```


Exemplo

```
# inicializa vetor de notas com 0
notas = [0] * 3
soma = 0
# preenche vetor de notas, sem usar append
for i in range(3):
    notas[i] = eval(input("Digite a nota do
    aluno " + str(i) + ": "))
    soma = soma + notas[i]
print("A media da turma é", soma/3)
```

Referências

- Slides baseados no curso de Programação de Computadores I da Prof. Vanessa Braganholo