

Universidade Federal de Viçosa
Departamento de Informática
Centro de Ciências Exatas e Tecnológicas



INF 100 – Introdução à Programação

Arranjos Simples
(listas, matrizes)

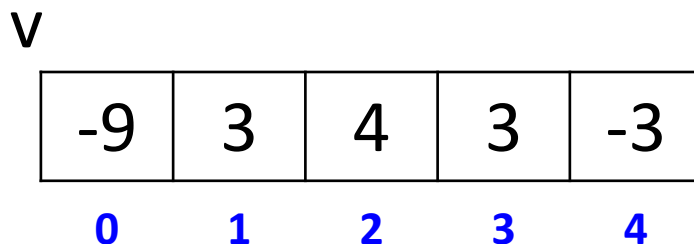
Arranjos (listas / vetores / *arrays*)

- São agregados de elementos (conjunto de variáveis) do mesmo tipo.
- Esse conjunto é sempre indexado por um número inteiro. Os índices dos elementos de um arranjo começam sempre de 0.
- Um arranjo pode ser uni-, bi-, tri- ou n -dimensional.



Arranjos

- Podemos representar esse esquema em memória como um diagrama, onde cada posição é identificada por um índice e pode armazenar um dado de um determinado tipo.
- Por exemplo, a estrutura chamada de *v* contendo 5 números inteiros em memória pode ser representada em um diagrama como:



Arranjos

- Nesse exemplo, o acesso ao elemento armazenado na 5ª posição do arranjo pode ser por meio de $v[4]$ e vale -3, assim como $v[1]$ vale 3 e assim por diante.

v

-9	3	4	3	-3
0	1	2	3	4

- Note que não há problema algum em armazenar um mesmo valor em várias posições do arranjo, mas os índices do arranjo são sempre únicos.



Arranjos em Python

- Para criar e manipular arranjos numéricos (vetores e matrizes) em Python, usaremos a biblioteca **numpy**. Motivos:
 - Maior facilidade para criar e inicializar os arranjos.
 - Maior semelhança com outras linguagens de programação.



Arranjos em Python

- Para isso, é preciso importar a biblioteca:

```
import numpy
```

- Podemos fazer dando também um 'apelido' para a biblioteca, só para podermos usar um nome menor (você entenderá isso melhor mais adiante:

```
import numpy as np
```



Arranjos em Python

- Criação de um arranjo de n números reais, não inicializado:

```
identificador = numpy.empty( n )
```

- **identificador** determina o nome da variável que identifica o conjunto de dados.
- Exemplo:

```
import numpy
```

```
notas = numpy.empty( 5 )  
alturas = numpy.empty( 10 )
```



Arranjos em Python

- Usando o 'apelido' para **numpy**, teríamos:

```
import numpy as np
```

```
notas = np.empty( 5 )
```

```
alturas = np.empty( 10 )
```



Arranjos em Python

- Criação de um arranjo de n números reais, já inicializado com zeros:

```
import numpy as np
```

```
notas = numpy.zeros( 5 )
```

```
alturas = numpy.zeros( 10 )
```



Arranjos em Python

Operações e acesso sobre elementos:

```
v = np.empty( 5 ) # Cria um vetor v de 5 elementos

v[0] = 2          # Atribui valor na primeira posição
v[1] = 4.5        # idem, na segunda posição
v[2] = -1         # idem, na terceira posição
v[3] = 0.5        # idem, na quarta posição
v[4] = 4.5        # idem, na quinta posição

print( v[4] + v[1] ) # exibe 9.0 (4.5 + 4.5) em tela
v[0] = v[2] = -10   # armazena -10 nas 3ª e 1ª posições
v[3] = float( input('') ) # Lê, do usuário, novo valor para v[3]

v[4] = v[3] + v[2]*v[2]

# Erros de acesso (índices inválidos)
print( v[-1], v[5], v[1.5] )
```

Cuidado!



Arranjos em Python

- Inicialização de arranjos na criação:

```
A = np.asarray([4, 3, 2, 1])
```

```
notas = np.asarray([70.5, 80, 54.3, 77.8])
```



Percorrendo Arranjos

- Ao usarmos arranjos, uma necessidade frequente é percorrer todos os elementos do arranjo, fazendo alguma tarefa com cada elemento. Exemplos:

**Para cada elemento i do arranjo A :
Escreva $A[i]$ na tela**

**Para cada elemento i do arranjo A :
Multiplique $A[i]$ por 2**



Percorrendo Arranjos

- Podemos fazer isso facilmente usando o comando **while**:

Para cada elemento i do arranjo A :
Escreva $A[i]$ na tela



```
i = 0
while i < len( A ):
    print( A[i] )
    i = i + 1
```



Percorrendo Arranjos

- No entanto, o comando **for** é bem mais apropriado para essas situações:

```
para i = 0 até n-1:  
    Escreva A[i]  
fim_para  
(onde n = tamanho de A)
```

(algoritmo)

```
for i in range(0, len( A )):  
    print( A[i] )
```

(Python)



Percorrendo Arranjos

- O comando **for** do Python permite também outra forma conveniente de iterar pelos elementos do arranjo:

```
para cada  $x \in A$ :  
    Escreva  $x$   
fim_para
```

(algoritmo)

```
for x in A:  
    print( x )
```

(Python)



Percorrendo Arranjos (mais exemplos)

```
import numpy as np

n = int( input('Número de elementos: '))
A = np.empty( n ) # Criação do arranjo

# Leitura do arranjo pelo teclado
for i in range(0, n):
    s = 'Informe o %dº elemento: ' % (i+1)
    A[i] = float( input( s ))

# Multiplicando o arranjo por 2
for i in range(0, n):
    A[i] = A[i] * 2

# Escrita do arranjo na tela
for x in A:
    print( x )
```



Percorrendo Arranjos

(mais exemplos)

```
import numpy as np
```

```
n = int( input('Número de elementos: '))
```

```
A = np.zeros( n ) # Criação do arranjo, preenchido com 0.0
```

```
# Escrita do arranjo na tela, elementos separados por espaço
```

```
for x in A:  
    print( x, end=' ' )  
print()
```

```
# Escrita do arranjo na tela, elementos separados por tabulação
```

```
for x in A:  
    print( x, end='\t' )  
print()
```

```
# Escrita do arranjo na tela, elementos formatados
```

```
for x in A:  
    print( '%7.3f' % (x), end=' ' )  
print()
```



Exercício #1

- Faça um programa em Python que leia as notas de uma turma de n alunos de um curso de idiomas e exiba em tela:

a) A média das notas:
$$\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$$

- b) O desvio padrão das notas:

$$s = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}}$$

- c) A maior nota
d) A menor nota
e) A quantidade de notas menor que a média



```

import numpy as np

n = int( input('Número de alunos: '))
print('Entre com a nota de cada aluno (uma em cada linha):')
notas = np.empty( n )
# input dos valores e cálculo da média, menor e maior
soma = 0
menor = 1000
maior = -1
for i in range(0,n):
    x = float( input(''))
    notas[i] = x
    soma = soma + x
    if x < menor:
        menor = x
    if x > maior:
        maior = x
media = soma / n

# Calcula o desvio padrão
soma = 0
for i in range(0,n):
    soma = soma + (notas[i] - media) ** 2
desvio = (soma / (n-1)) ** 0.5

print('\nMédia das notas: ', media )
print('Desvio padrão: ', desvio )
print('Maior nota: ', maior )
print('Menor nota: ', menor )

# det. qtd. de notas < que a média
n_menor = 0
for x in notas:
    if x < media:
        n_menor = n_menor + 1
print('Notas menores que a média: ', n_menor )

```



```

import numpy as np

n = int( input('Número de alunos: '))
print('Entre com a nota de cada aluno (uma em cada linha):')
notas = np.empty( n )
# input dos valores e cálculo da média
soma = 0
for i in range(0,n):
    notas[i] = float( input(''))
    soma = soma + notas[i]
media = soma / n

# Calcula o desvio padrão
soma = 0
for i in range(0,n):
    soma = soma + (notas[i] - media) ** 2
desvio = (soma / (n-1)) ** 0.5

print('\nMédia das notas: ', media )
print('Desvio padrão: ', desvio )

# Determinar menor e maior notas
menor = maior = notas[0]
for i in range(1,n):
    if notas[i] < menor:
        menor = notas[i]
    if notas[i] > maior:
        maior = notas[i]

print('Maior nota: ', maior )
print('Menor nota: ', menor )

# det. qtd. de notas < que a média
n_menor = 0
for x in notas:
    if x < media:
        n_menor = n_menor + 1
print('Notas menores que a média: ', n_menor )

```



Exercício #2

- Faça um programa em Python que leia as notas de uma turma de n alunos de um curso de idiomas e exiba em tela:

a) A média geométrica das notas:

$$GM(x) = \left(\prod_{i=1}^n x_i \right)^{1/n} = \sqrt[n]{x_1 x_2 \dots x_n}$$

b) A quantidade de notas maiores que a média geométrica



```
import numpy as np
```

```
n = int( input('Número de alunos: '))
```

```
print('Entre com a nota de cada aluno (uma em cada linha):')
```

```
notas = np.empty( n )
```

```
# input dos valores e cálculo da média geométrica
```

```
prod = 1
```

```
for i in range(0,n):
```

```
    notas[i] = float( input(''))
```

```
    prod = prod * notas[i]
```

```
mg = prod ** (1/n)
```

```
print('\nMédia geométrica das notas: ', mg )
```

```
# det. qtd. de notas > que a média geométrica
```

```
n_maior = 0
```

```
for x in notas:
```

```
    if x > mg:
```

```
        n_maior = n_maior + 1
```

```
print('Notas maiores que a média geométrica: ', n_maior )
```



Exercício #3

Faça um programa que:

- a) Leia uma sequência de n valores inteiros quaisquer.
- b) Escreva na tela a relação entre cada elemento 'vizinho'.

Exemplo:

n: 5

Entre com os valores (uma em cada linha):

0

4

-1

-1

5

Relações: 0 < 4 > -1 = -1 < 5



```
import numpy as np
```

```
n = int( input('n: '))
```

```
# Criar um arranjo de n elementos inteiros
```

```
v = np.empty( n, dtype=int )
```

```
print('Entre com os valores (uma em cada linha):')
```

```
for i in range(0,n):
```

```
    v[i] = int( input(''))
```

```
print('Relações: ', end='')
```

```
print(v[0], end='')
```

```
for i in range(1,n):
```

```
    if (v[i] > v[i-1]):
```

```
        print(' < ', end='')
```

```
    elif (v[i] < v[i-1]):
```

```
        print(' > ', end='')
```

```
    else:
```

```
        print(' = ', end='')
```

```
    print(v[i], end='')
```

```
print()
```



Exercício #4

Num arquivo existem resultados de experimentos genéticos, onde os dados consistem de letras A, T, G e C, representando as bases adenina, citosina, guanina e timina das cadeias de DNA. Exemplo:

```
TCGCCGAATAAACTCCTCTCGAGAGACT  
ACACGCGCAACATCCGTATAATTTACGG  
ACA...
```



Exercício #4

Deseja-se ler esse arquivo e produzir um histograma com a frequência de cada base encontrada nos dados. Por exemplo:

```
A :***** (101 = 25.2%)
T :***** (75 = 18.8%)
G :***** (114 = 28.5%)
C :***** (110 = 27.5%)
```



Exercício #4

Considere uma largura máxima da tela fixa (e.g. 80 caracteres) de modo a produzir um 'gráfico' em escala independente da quantidade de bases lidas.

Dica: inicialmente, em vez de trabalhar com o arquivo de entrada, gere aleatoriamente um número n de bases para testar o algoritmo de contagem e produção do histograma.



```

import numpy as np
import random

random.seed()

cbases = ['A', 'T', 'G', 'C']
conta = [ 0, 0, 0, 0 ] # contagem de cada base

n = int( input("Entre com o número de bases: "))
bases = np.empty( n, dtype=str )
for i in range(0,n):
    j = random.randint( 0, 3 ) # sorteia número de 0 a 3
    bases[i] = cbases[j]
    conta[j] = conta[j] + 1

"""
# Escreve as bases sorteadas na tela
for i in range(0,n):
    print( bases[i], end='')
print()
#"""
print( conta )
print()

maxcol = 110 # escala máxima da coluna de asteriscos
for i in range(0,4):
    print( cbases[i], ': ', end='')
    ci = conta[i]
    # calcula número de asteriscos
    na = int( ci/n * maxcol )
    # imprime asteriscos
    for j in range(0,na):
        print('*', end='')
    # imprime % da base
    print('\t(%d = %4.1f%%)' % ( ci, ci/n*100 ))

```

