

Python - Vetores

Vetores

- Algumas vezes necessitamos armazenar muitos dados de um mesmo tipo em memória (dezenas, centenas, milhares,)
- Até o momento, a única maneira para se fazer isso seria a utilização de dezenas (centenas, ...) variáveis
 - Ineficiente e trabalhoso
- Uma maneira para resolver esse problema é utilizar os vetores

Vetores

- Um exemplo de quando utilizar vetores seria a leitura das notas dos 40 alunos de uma turma e a impressão da quantidade de alunos acima da média da turma
- Com vetores é possível solucionar esse problema sem ter que criar 40 variáveis distintas◀◀
- VETORES
 - Estrutura homogênea – formada por elementos do mesmo tipo
 - Corresponde a várias posições de memória, acessadas através de um único nome e um índice

Vetores

- Semanticamente, no Python a diferença entre vetores e listas é que vetores não permitem tipos diferentes de dados, ao passo que em listas não existe esta restrição.
- Para efeitos práticos deste curso, usaremos LISTAS para trabalhar como VETORES.
- Assim, neste curso não faremos diferença para tratar listas e vetores livremente.

Vetores

Para acessar um elemento usamos o índice: `Nota[2]`

Nota	8	5.7	4	2.5	7.5	9.1	1
	0	1	2	3	4	5	6

Vetores

- Para criar um vetor com tamanho fixo usamos o operador `*`

```
>>> Nota = [0.0] * 4
```

```
>>> print Nota
```

```
[0.0, 0.0, 0.0, 0.0]
```

```
>>> len(Nota)
```

```
4
```

Vetores

>>> # Podemos também iniciar cada elemento do vetor

>>> Nota = [7.0, 5.5, 8, 9.2, 3.4, 1]

>>> Nota

[7.0, 5.5, 8, 9.2, 3.4, 1]

>>> len(Nota)

6

>>> Nota[2]=0.0

>>> Nota

[7.0, 5.5, 0.0, 9.2, 3.4, 1]

Programa

- Fazer um programa para ler as notas de uma turma de 40 alunos, imprimir a qtd dos alunos ACIMA da média da turma

Programa

- Fazer um programa para ler as notas de uma turma de 40 alunos, imprimir a qtd dos alunos ACIMA da média da turma

```
soma = 0
cont = 0
notas = [0.0]*40
for i in range(0,40):
    nota = float(input("nota:"))
    notas[i] = nota
    soma += notas[i]
media = soma/40
print "media:", media
for j in range(0,40):
    if notas[j] > media:
        cont += 1
print "qtd:", cont
```

Vetor - Lista

- Acrescentar um elemento ao final da lista

```
>>> Nota.append(6.0)
```

```
>>> Nota
```

```
[7.0, 5.5, 0.0, 9.2, 3.4, 1, 6.0]
```

```
>>>
```

Lista

- Em um VETOR tradicional, EM REGRA, não é possível acrescentar ou modificar valores de tipos DISTINTOS
- Porém, a LISTA do Python permite que armazenemos qualquer valores

```
>>> Nota.append('texto')
>>> Nota
>>> [7.0, 5.5, 0.0, 9.2, 3.4, 1, 6.0, 'texto']
```

Programa

- Refazer o programa anterior usando o método append

Programa

- Refazer o programa anterior usando o método append

```
soma = 0
cont = 0
notas = []
for i in range(0,40):
    notas.append( float(input("nota:")) )
    soma += notas[i]
media = soma/40
print "media:", media
print "lista:", notas
for j in range(0,40):
    if notas[j] > media:
        cont += 1
print "QTD:", cont
```

Programa

- Fazer um programa para ler 10 dados para um Vetor e imprimir a soma dos elementos pares menos a soma dos elementos de índice ímpar

Programa

- Fazer um programa para ler 10 dados para um Vetor e imprimir a soma dos elementos pares menos a soma dos elementos de índice ímpar

```
soma = 0
notas = []
for i in range(0,10):
    notas.append( int(input("nota:")) )
print "VETOR:", notas
```

```
for i in range(0,10):
    if notas[i] % 2 == 0:
        soma += notas[i]
    if i % 2 == 1:
        soma -= notas[i]
```

```
print "SOMA:", soma
```

Programa

- Fazer um programa para ler números inteiros para dentro do vetor até que seja digitado um número negativo que não faz parte da lista. Depois, ler um número inteiro e imprimir a posição que ele se encontra no vetor ou uma mensagem caso contrário.

Programa

- Fazer um programa para ler números inteiros para dentro do vetor até que seja digitado um número negativo que não faz parte da lista. Depois, ler um número inteiro e imprimir a posição que ele se encontra no vetor ou uma mensagem caso contrário.

```
vetNum = []  
num = input("N:")  
while num >= 0:  
    vetNum.append(num)  
    num = input("N:")  
N=input("Num Buscado:")  
pos = -1  
for i in range(0, len(vetNum)):  
    if N == vetNum[i]:  
        pos = i  
  
if pos<0:  
    print 'ERRO'  
else:  
    print 'POS:',pos
```



Programa

- Fazer um programa para ler dados para dois vetores e imprimir a interseção desses vetores

Programa

- Fazer um programa para ler dados para dois vetores e imprimir a interseção desses vetores

```
vet1 = []
vet2 = []
num = int(input("Num:"))
while num >= 0:
    vet1.append( num )
    num = int(input("Num:"))
print vet1
num = int(input("Num:"))
while num >= 0:
    vet2.append( num )
    num = int(input("Num:"))
print vet2
for i in range(0,len(vet1)):
    for j in range(0,len(vet2)):
        if vet1[i] == vet2[j]:
            print vet1[i]
```



Lista

- O método **count** funciona igual ao método count da classe String
- O método **insert** é similar ao append, porém acrescenta em uma posição específica

```
>>> Nota= [7, 0, 5, 0, 2, 3, 1, 6, 0, 3]
>>> Nota.count(0)
3
>>> Nota.insert(1,16)
>>> Nota
[7, 16, 0, 5, 0, 2, 3, 1, 6, 0, 3]
>>>
```

VETOR.insert(POSIÇÃO, ELEMENTO)

Programa

- Fazer um programa para ler dados para um vetor. Criar um outro vetor onde os elementos de ordem par são multiplicados por 2 e os de ordem ímpar multiplicados por 3 e diminuídos de 1. No final imprimir os dois vetores

Programa

- Fazer um programa para ler dados para um vetor. Criar um outro vetor onde os elementos de ordem par são multiplicados por 2 e os de ordem ímpar multiplicados por 3 e diminuídos de 1. No final imprimir os dois vetores

```
vet1 = []  
vet2 = []  
num = int(input("Num:"))  
while num >= 0:  
    vet1.append( num )  
    num = int(input("Num:"))  
  
for i in range(0,len(vet1)):  
    if i % 2 == 0:  
        vet2.append( vet1[i]*2 )  
    else:  
        vet2.append( vet1[i]*3 - 1)  
  
print vet1  
print vet2
```

Lista

- O método **pop** remove e retorna um item da lista
 - `pop()` – remove o último elemento
 - `Pop(pos)` – remove o elemento da posição `pos`
- O método **remove**, elimina a primeira ocorrência de um elemento na lista

```
>>> Nota= [7, 0, 2, 3, 1, 6, 0, 3]
>>> Nota.pop()
3
>>> Nota
[7, 0, 2, 3, 1, 6, 0]
>>>
>>> Nota.remove(0)
>>> Nota
[7, 2, 3, 1, 6, 0]
>>>
```

Programa

- Fazer um programa para acrescentar um elemento em uma lista se for digitado a letra “A”, remover se “R”, imprimir se “I” e sair se “F”.

Programa

- Fazer um programa para acrescentar um elemento em uma lista se for digitado a letra “A”, remover se “R”, imprimir se “I” e sair se “F”.

```
vetor = []
opcao = raw_input("entre com A para adicionar um elemento, \
R para remover, I para Imprimir e F para terminar")
while opcao.upper() != 'F':
    if opcao.upper() == 'A':
        elem = input("entre com elemento para armazenar no vetor")
        vetor.append(elem)
    elif opcao.upper() == 'R':
        vetor.pop()
    elif opcao.upper() == 'I':
        print vetor
    else:
        print "Opção Inválida"
    opcao = raw_input("entre com A para adicionar um elemento, \
R para remover e F para terminar")
```

Ordenação

- Para ordenar um vetor basta usar o método **sort**, que ordena o próprio vetor

```
>>> vet = [3, 2, 4, 1]
```

```
>>> vet.sort()
```

```
>>> vet
```

```
[1, 2, 3, 4]
```

- Porém, vocês tem que aprender a FAZER o método sort
 - Ou seja, aprender como se ordena um vetor

Ordenação – Método da Bolha

- Método simples para ordenar um vetor que compara a posição i com todos os posteriores a i

Ordenação – Método da Bolha

Ordenação – Método da Bolha

```
soma = 0  
vetor = []  
for i in range(0,5):  
    vetor.append( int(input("nota:")) )  
print "VETOR:", vetor
```

```
for i in range(0,4):  
    for j in range(i+1,5):  
        if vetor[i] > vetor[j]:  
            aux = vetor[i]  
            vetor[i] = vetor[j]  
            vetor[j] = aux  
print "VETOR ORDENADO:", vetor
```

Busca Binária

- Existe uma maneira eficiente de buscar elementos em um vetor
- Porém, é necessário que o vetor esteja ORDENADO
- Compara-se o elemento procurado com o elemento do meio do vetor
 - caso seja igual termina a busca
 - senão se o elemento for menor a busca irá ser feita apenas na metade inferior do vetor

Busca Binária

Busca Binária

```
vetor = []  
for i in range(0,5):  
    vetor.append( int(input("elem vetor:")) )  
vetor.sort()  
elem = int(input("elem vetor:"))
```


Busca Binária

```
vetor = []  
for i in range(0,5):  
    vetor.append( int(input("elem vetor:")) )  
vetor.sort()  
elem = int(input("elem vetor:"))
```

```
inicio = 0  
fim = len(vetor)-1  
achou = False  
while inicio <= fim and not achou:  
    meio = (inicio + fim)//2  
    if elem == vetor[meio]:  
        pos = meio  
        achou = True  
    elif elem < vetor[meio]:  
        fim = meio-1  
  
    else:  
        inicio = meio+1  
  
if achou:  
    print "ENCONTRADO POSIÇÃO:", pos  
else:  
    print "NÃO ENCONTRADO"
```