Solução da lista 9

(1) Esse exercício tem duas maneiras de se resolvido:

Primeira maneira:

```python
#Ex1
import numpy as np
n=int(input('n = '))
x=[]
y=[]
for i in range(n):
    a=float(input('x = '))
    x.append(a)

for j in range(n):
    b=float(input('y = '))
    y.append(b)

y.reverse()
x=np.array(x)
y=np.array(y)
z=np.zeros(n)
z=x+y**2
print(z)
```

Segunda maneira:

```python
#Ex1
import numpy as np
n=int(input('n = '))
x=np.zeros(n)
y=np.zeros(n)
for i in range(n):
    x[i]=float(input('x = '))

for j in range(n):
    y[j]=float(input('y = '))
y=y[::-1]
z=x+y**2
print('z = ',z)
```

(2) Duas maneiras de resolver esse exercício:

Primeira maneira:

```python
#Ex2
import numpy as np
n=int(input('n = '))
A=np.zeros((n,n))
cont=0
for i in range(n):
    for j in range(n):
        A[i,j]=float(input('A = '))
        if A[i,j]%5==0:
                cont=cont+1
print('A = ',A)
print('núm. div por 5 = ',cont)
```

Segunda maneira:

```
#Ex2
import numpy as np
n=int(input('n = '))
A=np.zeros((n,n))

for i in range(n):
    for j in range(n):
        A[i,j]=float(input('A = '))

cont=len(A[A % 5 == 0])

print('A = ',A)
print('núm. div por 5 = ',cont)
```

(3)

```
#Ex3
import numpy as np
n=int(input('n = '))
A=np.zeros((n,n))

for i in range(n):
    for j in range(n):
        A[i,j]=float(input('A = '))
ord=0
for i in range(n):
    for j in range(n-1):
        if A[i,j]>A[i,j+1]:
                ord=1
                break
print(ord)
for j in range(n):
    for i in range(n-1):
        if A[i,j]>A[i+1,j]:
            ord=1
            break
if ord==0:
    print('ordenada')
else:
    print('não ordenada')
```

(4)

```
#Ex4
n=int(input('n= '))
print('----------------')
list=[1]
for i in range(n):
    print(list)
    nova=[]
    nova.append(list[0])
    for i in range(len(list)-1):
        nova.append(list[i]+list[i+1])
    nova.append(list[-1])
    list=nova
```

(5) Existem duas possíveis soluções. A primeira com o uso tradicional do “for” e cálculo da média e a segunda com comandos específicos do Python.

Primeira solução:

```
1 #Ex5
2 import numpy as np
3 n=int(input('n = '))
4 A=np.zeros((n,n))
5 s=0
6 c=0
7 for i in range(n):
8     for j in range(n):
9         A[i,j]=float(input('A = '))
10        if i<j:
11            s=s+A[i,j]
12            c=c+1
13 media=s/c
14 print('MATRIZ A:')
15 print(A)
16 print('Média acima da diagonal da Matriz')
17 print('++++++++++++++++++++')
18 print(media)
```

```
MATRIZ A:
[[ 1. 10.  3.]
 [ 2.  5. -1.]
 [ 0.  2.  5.]]
Média acima da diagonal da Matriz
++++++++++++++++++++
4.0
```

Segunda solução:

O Python tem uma função chamada “triu” que significa “triangular superior”, para obter apenas os elementos acima da diagonal principal. A função agregada triu_indices( ) dá os índices dos elementos acima da diagonal. Colocando-se A[ ], significa que queremos apenas os elementos da matriz cujos índices voltaram da triu_indices( ).

Então para esses elementos, se colcarmos *.mean( ) o Python retornará a média dos elementos acima da diagonal principal, como está na linha 9 do código a seguir.

```
1 #Ex5
2 import numpy as np
3 n=int(input('n = '))
4 A=np.zeros((n,n))
5 for i in range(n):
6     for j in range(n):
7         A[i,j]=float(input('A = '))
8
9 media = A[np.triu_indices(3,k=1)].mean()
10
11 print('Média acima da diagonal da Matriz')
12 print('++++++++++++++++++++')
13 print(media)
```

```
Média acima da diagonal da Matriz
++++++++++++++++++++
4.0
```

(6) Existem duas possíveis soluções. A primeira com o uso tradicional do “for” e cálculo da média e a segunda com comandos específicos do Python.

Primeira solução:

```python
#Ex6
import numpy as np
n=int(input('n = '))
A=np.zeros((n,n))
s=0
c=0
for i in range(n):
    for j in range(n):
        A[i,j]=float(input('A = '))
        if i>j:
            s=s+A[i,j]
            c=c+1
media=s/c
print('MATRIZ A:')
print(A)
print('Média abaixo da diagonal da Matriz')
print('++++++++++++++++++++')
print(media)
```

```
MATRIZ A:
[[ 1. 10.  3.]
 [ 2.  5. -1.]
 [ 0.  2.  5.]]
Média abaixo da diagonal da Matriz
++++++++++++++++++++
1.3333333333333333
```

Segunda solução:

O Python tem uma função chamada “tril” que significa “triangular inferior”, para obter apenas os elementos abaixo da diagonal principal. A função agregada tril_indices( ) dá os índices dos elementos abaixo da diagonal. Colocando-se A[ ], significa que queremos apenas os elementos da matriz cujos índices voltaram da tril_indices( ).

Então para esses elementos, se colcarmos *.mean( ) o Python retornará a média dos elementos abaixo da diagonal principal, como está na linha 9 do código a seguir.

```python
#Ex6
import numpy as np
n=int(input('n = '))
A=np.zeros((n,n))
for i in range(n):
    for j in range(n):
        A[i,j]=float(input('A = '))

media = A[np.tril_indices(3,k=-1)].mean()

print('Média acima da diagonal da Matriz')
print('++++++++++++++++++++')
print(media)
```

```
Média acima da diagonal da Matriz
++++++++++++++++++++
1.3333333333333333
```

(7) Existem duas possíveis soluções. A primeira com o uso tradicional do “for” e cálculo da média e a segunda com comandos específicos do Python.

Primeira solução:

```python
#Ex7
import numpy as np
n=int(input('n = '))
A=np.zeros((n,n))
for i in range(n):
    for j in range(n):
        A[i,j]=float(input('A = '))
        if i>j:
            if i==0 and j==1:
                maximo=A[i,j]
            else:
                if A[i,j]>maximo:
                    maximo=A[i,j]

print('+++++ Matriz ++++++')
print(A)
print('Máximo acima da diagonal da Matriz')
print('++++++++++++++++++++')
print(maximo)
```

```
+++++ Matriz ++++++
[[ 1. 10.  3.]
 [ 2.  5. -1.]
 [ 0.  2.  5.]]
Máximo acima da diagonal da Matriz
++++++++++++++++++++
10.0
```

Segunda solução:

O Python tem uma função chamada “tril” que significa “triangular inferior”, para obter apenas os elementos abaixo da diagonal principal. A função agregada tril_indices( ) dá os índices dos elementos abaixo da diagonal. Colocando-se A[ ], significa que queremos apenas os elementos da matriz cujos índices voltaram da tril_indices( ).

Então para esses elementos, se colcarmos *.max( ) o Python retornará o maior elemento acima principal, como está na linha 9 do código a seguir.

```python
#Ex7
import numpy as np
n=int(input('n = '))
A=np.zeros((n,n))
for i in range(n):
    for j in range(n):
        A[i,j]=float(input('A = '))

maximo = A[np.triu_indices(3,k=1)].max()

print('Máximo acima da diagonal da Matriz')
print('++++++++++++++++++++')
print(maximo)
```

```
Máximo acima da diagonal da Matriz
++++++++++++++++++++
10.0
```

(8)

Primeira solução:

```
1 #Ex8
2 import numpy as np
3 n=int(input('n = '))
4 A=np.zeros((n,n))
5 maximo=-1000
6 for i in range(n):
7     for j in range(n):
8         A[i,j]=float(input('A = '))
9         if A[i,j]>maximo:
10            maximo=A[i,j]
11            linha=i
12 minimo=A[linha,1]
13 for j in range(n):
14     if A[linha,j]<minimo:
15         minimo=A[linha,j]
16
17 print(A)
18 print('++++++++++++++++++++')
19 print('MinMax =',minimo)
20
```

```
[[-3. -1.  5.]
 [ 7.  3. 10.]
 [-5.  1.  2.]]
++++++++++++++++++++
MinMax = 3.0
```

Segunda solução:

O Python tem uma função chamada “where” da biblioteca numpy, que significa onde buscar. Quando colocada qual a condição que se deseja buscar, o numpy volta a linha e coluna da busca. Como queremos o maior elemento, “where” retorna a linha e coluna do maior elemento. Com a linha do maior elemento, percorremos toda ela com A[linha,:] e escolhemos a função agregada *.min( ) que retornará o menor valor da linha com o maior valor.

Observe a linha 10 e a linha 11 do código a seguir.

```
1 #Ex8
2 import numpy as np
3 n=int(input('n = '))
4 A=np.zeros((n,n))
5 maximo=-1000
6 for i in range(n):
7     for j in range(n):
8         A[i,j]=float(input('A = '))
9
10 linha,coluna = np.where(A == A.max())
11 minimo=A[linha,:].min()
12
13 print(A)
14 print('++++++++++++++++++++')
15 print('MinMax =',minimo)
```

```
[[-3. -1.  5.]
 [ 7.  3. 10.]
 [-5.  1.  2.]]
++++++++++++++++++++
MinMax = 3.0
```