Laborator 7 – Bioinformatică

Alinierea de secvenţe - continuare

A. Biblioteca NumPy

NumPy este o bibliotecă dezvoltată pentru manipularea de tablouri multidimensionale (arrays).

Tablourile sunt obiecte mutabile. Tipurile de date conţinute în tablouri pot fi: *int*, *float*, *complex*, *double*, *byte*, *long*, *string*, etc.

Să se testeze următoarele exemple. Ce afişează?

Observaţie. Ca orice bibliotecă, *NumPy* trebuie inclusă în scriptul Python care o utilizează:

```
from numpy import *
```

1.1.1 Funcţii de bază pentru manipularea de tablouri

Exemplul 1

```
# crearea unui vector dintr-o lista

a = array([1,2,3,4,5])

print a

# crearea unui vector de lungime 4 cu elementele 0.0

b = zeros(4)

print b

# crearea unui vector de lungime 4 cu elementele 1.0

c = ones(4)

print c

# crearea unei matrici

d = array([[1,2],[3,4]])
```

```
print d

# matrice de 4x3 cu toate elementele 0.0

f = zeros((4,3))

print f

# matrice de 4x3 cu toate elementele nr intregi 0

g = zeros((4,3),int)

print g

# matrice de 4x3 cu toate elementele nr complexe

h = zeros((4,3), complex)

print h
```

1.1.2 Funcţii matematice de bază pe tablouri

Exemplul 2

```
a = array([1,5,4.3])

b = array([-0.9,3,4])

print a + b

print a-b

print a*b

print a/b
```

Exemplul 3

```
a = array([[1,2,3,4],[5,6,7,8],[9,10,11,12]])

print a
```

```
print a[0]

print a[-1]

print a[0][0]

print a[1][2]

print a[:2]

print a[:,1]

print a.shape #dimensiunea unei matrici

c = array([[1,2,3,4],[5,6,7,8],[3]) #eroare, de ce?
```

Exemplul 4

```
a = array([1,2,3,4])

b = array([5,6,7,8])

# inmultire pe componente

print a*b

# produs scalar - inmultire vector linie a cu vector colana b

print dot(a,b)

# inmultire un vector coloana a cu un vector linie b

print outer (a,b)
```

Exemplul 5

```
from numpy.random import *

a = ranf((4,3))

print a
```

```
b = random_integers(0,5,(4,3))

print b
```

Exemplul 6

```
b = random_integers(0,5,(4,3))

print sqrt(b)

print b**2

print exp(b)

print log(b)

print log2(b)
```

1.1.3 Copieri, comparări şi *slicing*

Exemplul 7

Variabilele conţin referinţe către obiectele din memorie. Tablourile sunt tot obiecte. Ce afişează codul de mai jos şi de ce?

```
b = array ([5.6, 4.2, 3.8])

c = b

c[0] = 0

print b
```

Exemplul 8

Pentru a copia locaţia de memorie se utilizează funcţia *copy*.

```
from copy import *

b = array([5.6, 4.2, 3.8])

c=copy(b)

c[0]=0
```

```
print b
```

Copia se mai poate realiza prin adunarea lui 0 la vectorul *b* care creează alt obiect, la altă locaţie de memorie.

```
b = array([5.6, 4.2, 3.8])

c=b+0

c[0]=0

print b
```

Exemplul 9

```
a = ranf(5)

b = ranf(5)

print a

print b

print greater(a,b)

print greater(a,b).astype(int)
```

Alte funcţii de comparare sunt: *less*, *less_equal*, *greater_equal*, *equal*.

Exemplul 10

```
a = ranf(5) - 0.5

print a

print sign(a)
```

Exemplul 11

```
a = [True, False, False,True]

b = [True, False, True,False]

print logical_and(a,b)
```

```
print logical_or(a,b)
```

Exemplul 12

- Conversii:

```
c = array([[1,2,3],[4,5,6],[7,8,9]])

print c

c[0,2] = 10.5

c[0][1] = 0.5

print c

c = c + 0.0

c[0,2] = 10.5

print c

c = c.astype(int)

print c
```

Exemplul 13

```
a = arange(15) #intoarce un array de la 0 la 14

print a

a = a/4

print a

a = arange(15)

a = a/4.

print a

b = [3,5,14,9,2]
```

```
print a[b]
```

Exemplul 14

- Sortare:

```
a = ranf(5)

print a

a.sort()

print a

b = a[::-1]

print b

a = ranf(5)

print a

print a.argsort() #indicii sevcventei sortate
```

Exemplul 15

```
a = ranf((3,2))

print a

print ravel(a)

print

print a.reshape((2,3))

print

print a

print

b = ranf((2,2))
```

```
print b

print

c = ranf((1,2))

print c

print

print concatenate((a,b,c))
```

Exemplul 16

- Permutarea random a unui tablou:

```
a = ranf(5)

print a

shuffle(a)

print a
```

Exemplul 17

- Funcţii statistice:

```
a = ranf((4,3))

print a

# maximul din tot tabloul

print a.max()

# intoarce un vector cu maximele de pe fiecare coloana

print a.max(0)

# intoarce un vector cu maximele de pe fiecare linie

print a.max(1)
```

```
print a.min()

print a.min(0)

print a.min(1)

print a.sum()

print a.sum(0)

print a.sum(1)

print a.mean()

print a.mean(0)

print a.mean(1)
```

1.1.4 Conversii de la tablouri la alte tipuri

Exemplul 18

```
a = ranf(5)

print a.tolist()

a = ranf((4,3))

print a.tolist()
```

Exemplul 19

```
a = ranf(5)

print str(a)+'este string'

print map(str,a)
```

B. Altă modalitate de calculare a scorului în alinierea de secvenţe:

Pentru calcularea scorului alinierilor secvenţelor se iau în calcul şi alţi factori biologici precum tipurile de aminoacizi (polari, nepolari, ionizaţi). Astfel, când doi aminoacizi diferă, contează în calculul scorului final şi dacă ambii sunt încărcaţi pozitiv, de exemplu, sau dacă unul este încărcat, iar celălalt nu. Astfel, scorul se poate calcula ţinând cont şi de probabilitatea înlocuirii unui aminoacid cu altul în cadrul aceleiaşi familii de proteine.

Să se implementeze o funcţie, *blosum_score(seq1, seq2, gap = -8)*, care calculează scorul a doua secvenţe aliniate astfel:

- când se întâlneşte "–", se adună la scor valoarea *gap = -8*;
- se adună ponderile perechilor de aminoacizi aliniaţi;
- în matricea *BLOSUM*, elementul *(i,j)* reprezintă valoarea care se adună la scor dacă se întalneşte aliniată perechea *PBET[i] – PBET[j]*, unde *PBET* este un vector care conţine codurile aminoacizilor în urmatoarea ordine:
  PBET ='ARNDCQEGHILKMFPSTWYV'
- se observă că pe diagonala principală a matricii *BLOSUM* se găsesc valori pozitive deoarece corespund alinierii aceluiaşi aminoacid în ambele secvenţe.
- se pot utiliza funcţia *sir.index('caracter')* care întoarce poziţia caracterului în şirul *sir*.
- să se utilizeze următoarea matrice *BLOSUM* de scoruri:

```
BLOSUM = array([
[ 5,-2,-1,-2,-1,-1,-1, 0,-2,-1,-2,-1,-1,-3,-1, 1, 0,-3,-2, 0],
[-2, 7,-1,-2,-1, 1, 0,-3, 0,-4,-3, 3,-2,-3,-3,-1,-1,-3,-1,-3],
[-1,-1, 7, 2,-2, 0, 0, 0, 1,-3,-4,-0,-2,-4,-2,-1, 0,-4,-2,-3],
[-2,-2, 2, 8,-4, 0, 2,-1,-1,-4,-4,-1,-4,-5,-1, 0,-1,-5,-3,-4],
[-1,-4,-2,-4,13,-3,-3,-3,-3,-2,-2,-3,-2,-2,-4,-1,-1,-5,-3,-1],
[-1,-1, 0, 0,-3, 7, 2,-2, 1,-3,-2, 2, 0,-4,-1,-0,-1,-1,-1,-3],
[-1, 0, 0, 2,-3, 2, 6,-3, 0,-4,-3, 1,-2,-3,-1,-1,-1,-3,-2,-3],
[ 0,-3, 0,-1,-3,-2,-3, 8,-2,-4,-4,-2,-3,-4,-2, 0,-2,-3,-3,-4],
[-2, 0, 1,-1,-3, 1, 0,-2,10,-4,-3, 0,-1,-1,-2,-1,-2,-3,-1, 4],
[-1,-4,-3,-4,-2,-3,-4,-4,-4, 5, 2,-3, 2, 0,-3,-3,-1,-3,-1, 4],
[-2,-3,-4,-4,-2,-2,-3,-4,-3, 2, 5,-3, 3, 1,-4,-3,-1,-2,-1, 1],
[-1, 3, 0,-1,-3, 2, 1,-2, 0,-3,-3, 6,-2,-4,-1, 0,-1,-3,-2,-3],
[-1,-2,-2,-4,-2, 0,-2,-3,-1, 2, 3,-2, 7, 0,-3,-2,-1,-1, 0, 1],
[-3,-3,-4,-5,-2,-4,-3,-4,-1, 0, 1,-4, 0, 8,-4,-3,-2, 1, 4,-1],
[-1,-3,-2,-1,-4,-1,-1,-2,-2,-3,-4,-1,-3,-4,10,-1,-1,-4,-3,-3],
[ 1,-1, 1, 0,-1, 0,-1, 0,-1,-3,-3, 0,-2,-3,-1, 5, 2,-4,-2,-2],
[ 0,-1, 0,-1,-1,-1,-1,-2,-2,-1,-1,-1,-1,-2,-1, 2, 5,-3,-2, 0],
[-3,-3,-4,-5,-5,-1,-3,-3,-3,-3,-2,-3,-1, 1,-4,-4,-3,15, 2,-3],
[-2,-1,-2,-3,-3,-1,-2,-3, 2,-1,-1,-2, 0, 4,-3,-2,-2, 2, 8,-1],
[0,-3,-3,-4,-1,-3,-3,-4,-4, 4, 1,-3, 1,-1,-3,-2, 0,-3,-1, 5]])
```