Programación para la Bioinformática

Módulo 4: Librerías científicas en Python - NumPy - Ejercicios

Ejercicios y preguntas teóricas

Ejercicio 1

Cread una matriz de 8x8 que tenga un patrón 0 (blancas)/1 (negras) como si se tratara de un tablero de ajedrez. El resultado ha de ser:

```
[[0 1 0 1 0 1 0 1]
[1 0 1 0 1 0 1 0]
[0 1 0 1 0 1 0 1]
[1 0 1 0 1 0 1 0]
[0 1 0 1 0 1 0 1]
[1 0 1 0 1 0 1 0]
[0 1 0 1 0 1 0 1]
[1 0 1 0 1 0 1 0]]
```

In [1]:
```
# Respuesta
```

Ejercicio 2

Cread dos matrices de tamaño 6x4 y 4x4 con números reales aleatorios y obtened la matriz resultado de multiplicar la primera por la segunda:

In [2]:
```
# Respuesta
```

Ejercicio 3

Calculad la media y la desviación estándar de un vector aleatorio de 300 elementos:

In [3]:
```
# Respuesta
```

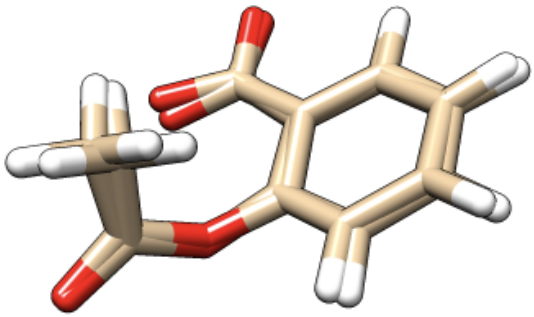
Ejercicio 4

Evaluad las funciones seno y arcoseno en el intervalo [-1,1] y con paso (resolución) de 0.025 y guardadlas en dos arrays (evaluar significa calcular):

In [4]:
```
# Respuesta
```

Ejercicio 5

En química computacional, es muy útil simular mediante la técnica de dinámica molecular cómo pequeños ligandos (por ejemplo, la aspirina) se unen a otras proteínas. Esta unión entre una proteína y un ligando puede bloquear la unión de esa misma proteína junto a otras, potenciarla, etc. En nuestro caso, hemos obtenido dos capturas del movimiento de la molécula de aspirina y deseamos saber cuál es la distancia en términos de RMSD entre ambas, es decir, cuánto se ha movido entre una captura y la otra. Si representáramos ese movimiento, podríamos visualizar algo similar a esto:



A continuación, tenéis parte del código que deberéis completar para calcular la distancia RMSD que está definida de la siguiente forma: https://en.wikipedia.org/wiki/Root-mean-square_deviation_of_atomic_positions (https://en.wikipedia.org/wiki/Root-mean-square_deviation_of_atomic_positions).

$$\begin{aligned}\mathrm{RMSD}(\mathbf{v},\mathbf{w}) &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}\|v_i - w_i\|^2} \\ &= \sqrt{\frac{1}{n}\sum_{i=1}^{n}((v_{ix}-w_{ix})^2+(v_{iy}-w_{iy})^2+(v_{iz}-w_{iz})^2)}\end{aligned}$$

La variable *estructura_1* contiene la información de las coordenadas de los átomos de la primera captura y *estructura_2* la segunda:

In [5]:

```
import numpy as np
import os


# Estructura atómica en formato PDB
estructura_1 = """
HETATM    1  C4  AIN A 141      20.988  20.013   8.918  1.00 32.08           C
HETATM    2  C3  AIN A 141      19.732  19.385   8.857  1.00 32.34           C
HETATM    3  C2  AIN A 141      19.527  18.104   9.636  1.00 33.22           C
HETATM    4  C5  AIN A 141      21.977  19.496   9.752  1.00 31.39           C
HETATM    5  H4  AIN A 141      21.183  20.890   8.320  1.00  0.00           H
HETATM    6  C6  AIN A 141      21.761  18.347  10.492  1.00 31.52           C
HETATM    7  H5  AIN A 141      22.932  19.991   9.830  1.00  0.00           H
HETATM    8  C1  AIN A 141      20.570  17.658  10.434  1.00 32.96           C
HETATM    9  H6  AIN A 141      22.537  17.967  11.139  1.00  0.00           H
HETATM   10  H1  AIN A 141      20.440  16.756  11.015  1.00  0.00           H
HETATM   11  C7  AIN A 141      18.679  19.893   7.921  1.00 32.12           C
HETATM   12  O1  AIN A 141      18.856  21.012   7.231  1.00 28.52           O
HETATM   13  O2  AIN A 141      17.582  19.373   7.818  1.00 32.90           O
HETATM   14  O3  AIN A 141      18.290  17.505   9.519  1.00 35.01           O
HETATM   15  C8  AIN A 141      17.117  17.595  10.410  1.00 35.75           C
HETATM   16  O4  AIN A 141      16.308  16.678  10.381  1.00 37.46           O
HETATM   17  C9  AIN A 141      16.879  18.753  11.344  1.00 36.44           C
HETATM   18  H91 AIN A 141      15.952  18.591  11.895  1.00  0.00           H
HETATM   19  H92 AIN A 141      17.709  18.830  12.046  1.00  0.00           H
HETATM   20  H93 AIN A 141      16.803  19.676  10.769  1.00  0.00           H
"""

estructura_2 = """
HETATM    1  C4  AIN A 141      20.909  19.934   8.896  1.00 32.08           C
HETATM    2  C3  AIN A 141      19.652  19.312   8.766  1.00 32.34           C
HETATM    3  C2  AIN A 141      19.387  18.158   9.565  1.00 33.22           C
HETATM    4  C5  AIN A 141      21.860  19.487   9.821  1.00 31.39           C
HETATM    5  H4  AIN A 141      21.131  20.820   8.331  1.00  0.00           H
HETATM    6  C6  AIN A 141      21.586  18.379  10.634  1.00 31.52           C
HETATM    7  H5  AIN A 141      22.771  20.048   9.926  1.00  0.00           H
HETATM    8  C1  AIN A 141      20.364  17.703  10.480  1.00 32.96           C
HETATM    9  H6  AIN A 141      22.314  18.043  11.358  1.00  0.00           H
HETATM   10  H1  AIN A 141      20.171  16.825  11.079  1.00  0.00           H
HETATM   11  C7  AIN A 141      18.638  19.970   7.867  1.00 32.12           C
HETATM   12  O1  AIN A 141      18.979  20.954   7.165  1.00 28.52           O
HETATM   13  O2  AIN A 141      17.431  19.667   7.937  1.00 32.90           O
HETATM   14  O3  AIN A 141      18.249  17.398   9.458  1.00 35.01           O
HETATM   15  C8  AIN A 141      17.235  17.488  10.335  1.00 35.75           C
HETATM   16  O4  AIN A 141      16.343  16.644  10.397  1.00 37.46           O
HETATM   17  C9  AIN A 141      17.211  18.698  11.261  1.00 36.44           C
HETATM   18  H91 AIN A 141      16.416  18.581  11.994  1.00  0.00           H
HETATM   19  H92 AIN A 141      18.154  18.814  11.795  1.00  0.00           H
HETATM   20  H93 AIN A 141      17.020  19.587  10.663  1.00  0.00           H
"""

def lee_coordenadas_atomo(linea):
    """Obtiene de una línea de texto en formato PDB las coordenadas en f
    ormato numérico"""
    if linea.startswith('HETATM'):
        x = float(linea[30:38])    # Coordenada x
        y = float(linea[38:46])    # Coordenada y
        z = float(linea[46:54])    # Coordenada z
        return [x, y, z]


def obtiene_coordenadas(estructura):
    """Transforma las coordenadas de cada átomo en un array numpy"""
    coordenadas = []
    for linea in estructura.split(os.linesep):
        atomo = lee_coordenadas_atomo(linea)
        if atomo:
            coordenadas.append(atomo)
    return np.array(coordenadas)
```

```
0.0
```

El resultado del cálculo de la RMSD ha de ser de `0.253808096798` , no puede ser un número entero.

Ejercicio 6

Crea un vector de números enteros aleatorios de 30 posiciones y ordénalo:

In [6]:
```
# Respuesta
```

Ejercicio 7

Dado un vector `Z` con números aleatorios entre 0 y 1 de 100 posiciones y un valor `v` aleatorio entre 0 y 1, encuentra cuál es el valor en `Z` más próximo a `v` :

In [7]:
```
import numpy as np

Z = np.random.random(100)
v = np.random.uniform(0,1)

# Respuesta
```

Ejercicio 8

En un vector de 100 posiciones, encuentra cuál es el valor que se repite de forma más frecuente:

In [8]:
```
import numpy as np

Z = np.random.randint(0,10,100)

print(Z)

# Respuesta
```

```
[0 9 5 3 6 9 5 1 3 6 9 6 1 1 3 7 4 7 4 3 2 9 5 0 0 4 0 4 6 7 2 0 6 6 1 5 0
 5 3 8 8 4 7 3 8 6 3 8 3 4 6 7 1 3 6 6 4 5 8 2 8 1 3 1 1 5 2 4 7 2 4 8 5 2
 2 0 5 2 4 9 8 1 4 1 7 1 3 1 0 6 3 6 1 3 5 2 5 4 2 3]
```