1. Importa el paquete NUMPY bajo el nombre np.

```
#[tu código aquí]
import numpy as np
```

2. Imprime la versión de NUMPY y la configuración.

```
#[tu código aquí]
print(f"La version de numpy es: {np.__version__}")
```

3. Genera un array tridimensional de 2x3x5 con valores aleatorios. Asigna el array a la variable "a"

Desafío: hay al menos tres maneras fáciles que usan numpy para generar arrays aleatorios. ¿Cuántas formas puedes encontrar?

```
#[tu código aquí]
#a=np.random.random((2,3,5))
#a=np.random.randint(5,size=(2,3,5))
#a=np.random.permutation(np.linspace(0,1,30).reshape(2,3,5))
a=np.random.permutation(np.arange(30).reshape(2,3,5))
```

4. Imprime a.

```
#[tu código aquí]
print(f"El valor de 'a' es: \n {a}")
```

5. Crea un array tridimensional de 5x2x3 con todos los valores igual a 1.

Asigna el array a la variable "b"

```
#[tu código aquí]
b=np.ones((5,2,3))
```

6. Imprime b.

```
#[tu código aquí]
print(f"El valor de 'b' es: \n {b}")
```

7. ¿Tienen a y b el mismo tamaño? ¿Cómo lo demuestras en código Python?

```
#[tu código aquí]
print("Tienen 'a' y 'b' el mismo tamaño?: " +
      "Sí" if a.size==b.size else "No")
```

8. ¿Es posible sumar a y b? ¿Por qué sí o por qué no?

```
#[tu código aquí]
print("Es posible sumar 'a' y 'b'?: " +
      "No, ya que no tienen la misma forma" if a.shape!=b.shape
      else "Sí, ya que tienen la misma forma")
```

9. Transpone b para que tenga la misma estructura que a (es decir, se convierta en un array de 2x3x5). Asigna el array transpuesto a la variable "c".

```
#[tu código aquí]
c=np.transpose(b,(1,2,0))
```

10. Intenta sumar a y c. Ahora debería funcionar. Asigna la suma a la variable "d". Pero, ¿por qué funciona ahora?

```
#[tu código aquí]
d=a+c
```

11. Imprime a y d. ¿Notas la diferencia y la relación entre los dos arrays en términos de los valores? Explica.

```
#[tu código aquí]
print(f"El valor de 'a' es: \n {a} \n")
print(f"El valor de 'd' es: \n {d}")
```

12. Multiplica a y c. Asigna el resultado a e.

```
#[tu código aquí]
e=a*c
```

13. ¿Es e igual a a? ¿Por qué sí o por qué no?

```
#[tu código aquí]
print("Es 'e' igual a 'a'?: " +
      "No, ya que 'c' no es el elemento neutro del producto" if not
np.array_equal(e,a)
         else "Si, ya que 'c' es el elemento neutro del producto")
```

14. Identifica los valores máximos, mínimos y medios en d. Asigna esos valores a las variables "d_max", "d_min" y "d_mean"

```
#[tu código aquí]
d_max=np.max(d)
d_min=np.min(d)
d_mean=np.mean(d)
```

15. Ahora queremos etiquetar los valores en d. Primero crea un array vacío "f" con la misma forma (es decir, 2x3x5) que d usando `np.empty`.

```
#[tu código aquí]
f=np.empty((2,3,5))
```

16. Rellena los valores en f. Para cada valor en d, si es mayor que d_min pero menor que d_mean, asigna 25 al valor correspondiente en f.

- Si un valor en d es mayor que d_mean pero menor que d_max, asigna 75 al valor correspondiente en f.
- Si un valor es igual a d_mean, asigna 50 al valor correspondiente en f.
- Asigna 0 al valor correspondiente(s) en f para d_min en d.
- Asigna 100 al valor correspondiente(s) en f para d_max en d.
- Al final, f debería tener solo los siguientes valores: 0, 25, 50, 75 y 100.

```
#[tu código aquí]
def fill_f_using_d(f,d):
    d_max=np.max(d)
    d_min=np.min(d)
    d_mean=np.mean(d)

    f[(d>d_min)&(d<d_mean)]=25
    f[(d>d_mean)&(d<d_max)]=75
    f[d==d_mean]=50
    f[d==d_min]=0
    f[d==d_max]=100
```

17. Imprime d y f. ¿Tienes el f esperado?

Por ejemplo, si tu d es:

```
array([[[1.85836099, 1.67064465, 1.62576044, 1.40243961, 1.88454931],
        [1.75354326, 1.69403643, 1.36729252, 1.61415071, 1.12104981],
        [1.72201435, 1.1862918 , 1.87078449, 1.7726778 , 1.88180042]],
       [[1.44747908, 1.31673383, 1.02000951, 1.52218947, 1.97066381],
        [1.79129243, 1.74983003, 1.96028037, 1.85166831, 1.65450881],
        [1.18068344, 1.9587381 , 1.00656599, 1.93402165,
1.73514584]]])
```

Tu f debería ser:

```
array([[[ 75.,  75.,  75.,  25.,  75.],
        [ 75.,  75.,  25.,  25.,  25.],
        [ 75.,  25.,  75.,  75.,  75.]],
       [[ 25.,  25.,  25.,  25., 100.],
```

```
        [ 75.,  75.,  75.,  75.,  75.],
        [ 25.,  75.,   0.,  75.,  75.]]])

#[tu código aquí]

d_test=np.array([[[1.85836099, 1.67064465, 1.62576044, 1.40243961,
1.88454931],
        [1.75354326, 1.69403643, 1.36729252, 1.61415071, 1.12104981],
        [1.72201435, 1.1862918 , 1.87078449, 1.7726778 , 1.88180042]],
       [[1.44747908, 1.31673383, 1.02000951, 1.52218947, 1.97066381],
        [1.79129243, 1.74983003, 1.96028037, 1.85166831, 1.65450881],
        [1.18068344, 1.9587381 , 1.00656599, 1.93402165,
1.73514584]]])

f_solution=np.array([[[ 75.,  75.,  75.,  25.,  75.],
        [ 75.,  75.,  25.,  25.,  25.],
        [ 75.,  25.,  75.,  75.,  75.]],
       [[ 25.,  25.,  25.,  25., 100.],
        [ 75.,  75.,  75.,  75.,  75.],
        [ 25.,  75.,   0.,  75.,  75.]]])


fill_f_using_d(f,d_test)

print("f tiene el valor esperado" if np.array_equal(f,f_solution) else
"f no tiene el valor esperado")

f tiene el valor esperado
```

18. Pregunta de bonificación: en lugar de usar números (es decir, 0, 25, 50, 75 y 100), ¿cómo usar valores de cadena ("A", "B", "C", "D" y "E") para etiquetar los elementos del array? Esperas el resultado sea:

```
array([[[ 'D',  'D',  'D',  'B',  'D'],
        [ 'D',  'D',  'B',  'B',  'B'],
        [ 'D',  'B',  'D',  'D',  'D']],
       [[ 'B',  'B',  'B',  'B',  'E'],
        [ 'D',  'D',  'D',  'D',  'D'],
        [ 'B',  'D',   'A',  'D', 'D']]])

#[tu código aquí]
def fill_f_with_str_using_d(f,d):
    d_max=np.max(d)
    d_min=np.min(d)
    d_mean=np.mean(d)

    f[(d>d_min)&(d<d_mean)]="B"
    f[(d>d_mean)&(d<d_max)]="D"
    f[d==d_mean]="C"
    f[d==d_min]="A"
```

```
    f[d==d_max]="E"


f_solution_str=np.array([[[ 'D',  'D',  'D',  'B',  'D'],
        [ 'D',  'D',  'B',  'B',  'B'],
        [ 'D',  'B',  'D',  'D',  'D']],
       [[ 'B',  'B',  'B',  'B',  'E'],
        [ 'D',  'D',  'D',  'D',  'D'],
        [ 'B',  'D',   'A',  'D', 'D']]])

f_str=np.empty((2,3,5))
fill_f_with_str_using_d(f_str,d_test)
# print(f_str)

print("f_str tiene el valor esperado" if
np.array_equal(f_str,f_solution_str) else "f_str no tiene el valor
esperado")
```

```
---------------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
Cell In[15], line 22
     14 f_solution_str=np.array([[[ 'D',  'D',  'D',  'B',  'D'],
     15         [ 'D',  'D',  'B',  'B',  'B'],
     16         [ 'D',  'B',  'D',  'D',  'D']],
     17        [[ 'B',  'B',  'B',  'B',  'E'],
     18         [ 'D',  'D',  'D',  'D',  'D'],
     19         [ 'B',  'D',   'A',  'D', 'D']]])
     21 f_str=np.empty((2,3,5))
---> 22 fill_f_with_str_using_d(f_str,d_test)
     23 # print(f_str)
     25 print("f_str tiene el valor esperado" if
np.array_equal(f_str,f_solution_str) else "f_str no tiene el valor
esperado")

Cell In[15], line 7, in fill_f_with_str_using_d(f, d)
      4 d_min=np.min(d)
      5 d_mean=np.mean(d)
----> 7 f[(d>d_min)&(d<d_mean)]="B"
      8 f[(d>d_mean)&(d<d_max)]="D"
      9 f[d==d_mean]="C"

ValueError: could not convert string to float: 'B'
```