Crear un array: Crea un array de 10 elementos enteros aleatorios entre 1 y 100. Imprime el array generado.

```
import numpy as np

# Generamos el array de 10 enteros aleatorios entre 1 y 100
array = np.random.randint(1, 101, size=10)
print("Array generado:", array)
```

```
Array generado: [63 44 91 90 35 73 13 56 60 38]
```

Utilizamos np.random.randint(1, 101, size=10) para generar 10 números enteros aleatorios entre 1 y 100 (el límite superior es exclusivo, por eso ponemos 101).

Reshape: A partir del array anterior, convierte este array en una matriz de 2x5.

```
# Convertimos el array en una matriz de 2x5
matriz = array.reshape(2, 5)
print("Matriz 2x5:\n", matriz)
```

```
Matriz 2x5:
 [[63 44 91 90 35]
 [73 13 56 60 38]]
```

Usamos reshape(2, 5) para cambiar la forma del array a una matriz de 2 filas y 5 columnas.

Indexación y slicing: De la matriz obtenida, extrae la primera fila y la última columna. Imprime ambos resultados por separado.

```
# Extraemos la primera fila
primera_fila = matriz[0, :]
print("Primera fila:", primera_fila)

# Extraemos la última columna
ultima_columna = matriz[:, -1]
print("Última columna:", ultima_columna)
```

```
Primera fila: [63 44 91 90 35]
Última columna: [35 38]
```

Para la primera fila, usamos matriz[0, :], que selecciona la fila en el índice 0. Para la última columna, usamos matriz[:, -1], que selecciona todas las filas en la última columna.

Modificación: Cambia el valor de los elementos en la segunda fila a ceros.

```
# Cambiamos los valores de la segunda fila a ceros
matriz[1, :] = 0
print("Matriz después de modificar la segunda fila a ceros:\n", matriz)
```

```
Matriz después de modificar la segunda fila a ceros:
 [[63 44 91 90 35]
 [ 0  0  0  0  0]]
```

Crea dos arrays de 5 elementos enteros aleatorios entre 1 y 50.

```
# Creamos dos arrays de 5 elementos enteros aleatorios entre 1 y 50
array1 = np.random.randint(1, 51, size=5)
array2 = np.random.randint(1, 51, size=5)
print("Array 1:", array1)
print("Array 2:", array2)
```

```
Array 1: [15 31 47 41 47]
Array 2: [23 17 36 25 31]
```

Suma ambos arrays y luego multiplícalos entre sí.

```python
# Sumamos ambos arrays
suma = array1 + array2
print("Suma de los arrays:", suma)

# Multiplicamos ambos arrays elemento a elemento
producto = array1 * array2
print("Producto de los arrays:", producto)
```

```
Suma de los arrays: [38 48 83 66 78]
Producto de los arrays: [ 345  527 1692 1025 1457]
```

Calcula el promedio y la desviación estándar de los resultados obtenidos.

```
# Calculamos el promedio y desviación estándar de la suma
promedio_suma = np.mean(suma)
desviacion_suma = np.std(suma)
print("Promedio de la suma:", promedio_suma)
print("Desviación estándar de la suma:", desviacion_suma)

# Calculamos el promedio y desviación estándar del producto
promedio_producto = np.mean(producto)
desviacion_producto = np.std(producto)
print("Promedio del producto:", promedio_producto)
print("Desviación estándar del producto:", desviacion_producto)
```

```
Promedio de la suma: 62.6
Desviación estándar de la suma: 17.22324011328879
Promedio del producto: 1009.2
Desviación estándar del producto: 517.8163380968198
```

Calculamos las estadísticas para ambos resultados por separado para obtener una mejor comprensión de los datos

- **Reemplazar los valores menores a 0.5 con 0.**
- **Reemplazar los valores mayores o iguales a 0.5 con 1.**

```
# Generamos un array de 20 números aleatorios entre 0 y 1
array_random = np.random.rand(20)
print("Array aleatorio original:", array_random)

# Reemplazamos los valores menores a 0.5 con 0
array_random[array_random < 0.5] = 0

# Reemplazamos los valores mayores o iguales a 0.5 con 1
array_random[array_random >= 0.5] = 1

print("Array después de aplicar condiciones lógicas:", array_random)
```

```
Array aleatorio original: [0.07161956 0.39433941 0.55991428 0.64782354 0.02953628 0.34104955
 0.61799201 0.15519551 0.23925863 0.43323882 0.83648995 0.02421738
 0.37696188 0.63303863 0.32099636 0.4154565  0.08126568 0.67101271
 0.05633721 0.67891072]
Array después de aplicar condiciones lógicas: [0. 0. 1. 1. 0. 0. 1. 0. 0. 0. 1. 0. 0. 1. 0. 0. 0. 1. 0. 1.]
```

Producto escalar: Crea dos arrays de tamaño 3x3. Calcula el producto escalar entre ambos.

```
# Creamos dos matrices de tamaño 3x3 con números aleatorios entre 1 y 10
matriz1 = np.random.randint(1, 10, size=(3, 3))
matriz2 = np.random.randint(1, 10, size=(3, 3))
print("Matriz 1:\n", matriz1)
print("Matriz 2:\n", matriz2)

# Calculamos el producto escalar (producto matricial) entre ambas matrices
producto_escalar = np.dot(matriz1, matriz2)
print("Producto escalar de las matrices:\n", producto_escalar)
```

```
Matriz 1:
 [[3 3 4]
 [4 1 4]
 [9 7 5]]
Matriz 2:
 [[8 8 2]
 [6 1 7]
 [5 6 7]]
Producto escalar de las matrices:
 [[ 62  51  55]
 [ 58  57  43]
 [139 109 102]]
```

En el contexto de matrices, el "producto escalar" se refiere al producto matricial. Usamos np.dot para realizar esta operación.

Matriz de datos: Crea una matriz de 5 filas y 4 columnas que contenga datos aleatorios entre 1 y 20. Cada fila representa a una persona, y cada columna representa una medición diferente (como altura, peso, edad, presión arterial).

```
# Creamos la matriz de datos
datos = np.random.randint(1, 21, size=(5, 4))
print("Matriz de datos (Personas x Mediciones):\n", datos)
```

```
Matriz de datos (Personas x Mediciones):
 [[ 8  4 13 17]
 [18  1  8 15]
 [13 10  8 19]
 [ 6  3 18 14]
 [ 7  4 10 18]]
```

Calcula la media de cada columna (medición).

```
# Calculamos la media de cada columna
media_columnas = np.mean(datos, axis=0)
print("Media de cada medición (columna):", media_columnas)
```

```
Media de cada medición (columna): [10.4  4.4 11.4 16.6]
```

Usamos axis=0 en np.mean para calcular la media a lo largo de las filas, es decir, para cada columna.

Encuentra el valor máximo y mínimo de cada fila.

```
# Encontramos el valor máximo y mínimo de cada fila
maximo_filas = np.max(datos, axis=1)
minimo_filas = np.min(datos, axis=1)
print("Valor máximo de cada persona (fila):", maximo_filas)
print("Valor mínimo de cada persona (fila):", minimo_filas)
```

```
Valor máximo de cada persona (fila): [17 18 19 18 18]
Valor mínimo de cada persona (fila): [4 1 8 3 4]
```

Usamos axis=1 para calcular a lo largo de las columnas, es decir, para cada fila.

Ordena las filas según los valores de la segunda columna (peso).

```python
# Obtenemos los índices que ordenarían la matriz según la segunda columna (peso)
indices_ordenados = np.argsort(datos[:, 1])

# Ordenamos la matriz de datos usando estos índices
datos_ordenados = datos[indices_ordenados]
print("Matriz ordenada según la segunda columna (peso):\n", datos_ordenados)
```

```
Matriz ordenada según la segunda columna (peso):
 [[18  1  8 15]
 [ 6  3 18 14]
 [ 8  4 13 17]
 [ 7  4 10 18]
 [13 10  8 19]]
```

np.argsort(datos[:, 1]) devuelve los índices que ordenarían la segunda columna. Al aplicar estos índices a datos, obtenemos la matriz ordenada según el peso.