Datasheet Comandos de Numpy

Estudiantes:

- Daniel Alejandro Albarracín Vargas
- Juan Sebastián Garzón Gómez
- Nicolás López Sánchez
- Lina Mariana Pinzón Pinzón

CADI: Profundización II (Machine Learning)

Ejercicio 9

9. Evaluación de Desempeño en Diferentes Tareas

Tarea	Evaluación 1	Evaluación 2	Evaluación 3
1	85	90	88
2	80	85	83
3	75	80	78

Tabla de Contenido

- *np.divide*
- *np.power*
- *np.sin*
- *np.cos*
- *np.exp*
- *np.log*

4. **Estadísticas**

- *np.mean*
- *np.median*
- *np.std*
- *np.var*
- *np.min*
- *np.max*
- *np.sum*
- *np.cumsum*
- *np.percentile*

5. **Álgebra Lineal**

- *np.dot*
- *np.matmul*
- *np.linalg.det*
- *np.linalg.eig*
- *np.linalg.svd*

6. **Funciones de Indexacion y Seleccion**

- *np.where*
- *np.take*
- *np.choose*
- *np.nonzero*
- *np.extract*

---

1. Creación de Arrays

- *np.array*
  - *Descripción:* Crea un array Numpy.
  - *Ejemplos*:

- ```
  import numpy as np
   Tarea_1 = np.array([85, 90, 88,])
   Tarea_2 = np.array([80, 85, 83])
   Tarea_3 = np.array([75, 80, 78])
   print(Tarea_1)
   print(Tarea_2)
   print(Tarea_3)
   #
  ```

```
Output:
[85 90 88]
[80 85 83]
[75 80 78]
```

- ```
  import numpy as np
   Datos_Evaluacion = np.array([Tarea_1, Tarea_2, Tarea_3])
   print(Datos_Evaluacion)
  ```

```
Output:
[[85 90 88]
 [80 85 83]
 [75 80 78]]
```

---

2. Manipulación de Arrays

- *np.reshape*
  - *Descripción:* Cambia la forma de un array.
  - *Ejemplo*: En este caso lo cambio a forma vertical.
  - ```
    np.reshape(Datos_Evaluacion, (9, 1))
    ```

```
Output:
array([[85],
       [90],
       [88],
       [80],
       [85],
       [83],
       [75],
       [80],
       [78]])
```

- *np.transpose*
```

- *Descripción:* Transpone un array.
- *Ejemplo*:
- ```
  np.transpose(Datos_Evaluacion)
  #
  ```

```
Output:
array([[85, 80, 75],
       [90, 85, 80],
       [88, 83, 78]])
```

- *np.ravel*
  - *Descripción:* Aplana un array.
  - *Ejemplo*:
  - ```
    np.ravel(Datos_Evaluacion)
    #
    ```

```
Output:
array([85, 90, 88, 80, 85, 83, 75, 80, 78])
```

- *np.concatenate*
  - *Descripción:* Concatena arrays a lo largo de un eje.
  - *Ejemplo*: En este caso lo concatenó de forma horizontal.
  - ```
    np.concatenate((Tarea_1, Tarea_2, Tarea_3), axis=0)
    #
    ```

```
Output:
array([85, 90, 88, 80, 85, 83, 75, 80, 78])
```

- *np.hstack*
  - *Descripción:* Concatena arrays horizontalmente.
  - *Ejemplo*:
  - ```
    np.hstack((Tarea_1, Tarea_2, Tarea_3))
    #
    ```

```
Output:
array([85, 90, 88, 80, 85, 83, 75, 80, 78])
```

- *np.vstack*
  - *Descripción:* Concatena arrays verticalmente.
  - *Ejemplo*:
  - ```
    np.vstack((Tarea_1, Tarea_2, Tarea_3))
    #
    ```
    ```

```
Output:
array([[85, 90, 88],
       [80, 85, 83],
       [75, 80, 78]])
```

- *np.split*
  - *Descripción:* Divide un array en sub-arrays.
  - *Ejemplo*:
  - ```
    np.split (Datos_Evaluacion, 3)
    #
    ```

```
Output:
[array([[85, 90, 88]]), array([[80, 85, 83]]), array([[75, 80, 78]])]
```

- *np.hsplit*
  - *Descripción:* Divide un array horizontalmente.
  - *Ejemplo*:
  - ```
    np.hsplit(Datos_Evaluacion, 3)
    #
    ```

```
Output:
[array([[85],
        [80],
        [75]]),
 array([[90],
        [85],
        [80]]),
 array([[88],
        [83],
        [78]])]
```

- *np.vsplit*
  - *Descripción:* Divide un array verticalmente.
  - *Ejemplo*:
  - ```
    np.vsplit (Datos_Evaluacion, 3)
    #
    ```

```
Output:
[array([[85, 90, 88]]), array([[80, 85, 83]]), array([[75, 80, 78]])]
```

---

3. Funciones Matemáticas

- *np.add*
```

- *Descripción:* Suma elementos de arrays.
- *Ejemplo*: En este caso se sumaron las 3 tareas.
- ```
  a= np.add(Tarea_1, Tarea_2)
  suma= np.add(a, Tarea_3)
  print(suma)
  #
  ```

```
Output:
[240 255 249]
```

- *np.subtract*
  - *Descripción:* Resta elementos de arrays.
  - *Ejemplos*:
  - ```
    np.subtract(Tarea_1, Tarea_2)
    #
    ```

```
Output:
array([5, 5, 5])
```

  - ```
    np.subtract(Tarea_2, Tarea_3)
    #
    ```

```
Output:
array([5, 5, 5])
```

  - ```
    np.subtract(Tarea_1, Tarea_3)
    #
    ```

```
Output:
array([10, 10, 10])
```

- *np.multiply*
  - *Descripción:* Multiplica elementos de arrays.
  - *Ejemplo*:
  - ```
    b=np.multiply(Tarea_1, Tarea_2)
    multiplicacion=np.multiply(b, Tarea_3)
    print(multiplicacion)
    #
    ```

```
Output:
[510000 612000 569712]
```

- *np.divide*
  - *Descripción:* Divide elementos de arrays.
```

- *Ejemplos*:
- ```
  np.divide(Tarea_1, Tarea_2)
  #
  ```

```
Output:
array([0.94444444, 0.94117647, 0.9375])
```

- ```
  np.divide(Tarea_2, Tarea_3)
  #
  ```

```
Output:
array([1.06666667, 1.0625, 1.06410256])
```

- ```
  np.divide(Tarea_1, Tarea_3)
  #
  ```

```
Output:
array([1.13333333, 1.125, 1.12820513])
```

- *np.power*
  - *Descripción:* Eleva elementos a una potencia.
  - *Ejemplo*:
  - ```
    np.power(Datos_Evaluacion,2)
    #
    ```

```
Output:
array([[7225, 8100, 7744],
       [6400, 7225, 6889],
       [5625, 6400, 6084]])
```

- *np.sin*
  - *Descripción:* Calcula el seno de cada elemento.
  - *Ejemplo*:
  - ```
    np.sin(Datos_Evaluacion)
    #
    ```

```
Output:
array([[-0.17607562,  0.89399666,  0.0353983 ],
       [-0.99388865, -0.17607562,  0.96836446],
       [-0.38778164, -0.99388865,  0.51397846]])
```

- *np.cos*
  - *Descripción:* Calcula el coseno de cada elemento.
  - *Ejemplo*:
```

- ```
  np.cos(Datos_Evaluacion)
  #
  ```

```
Output:
array([[-0.98437664, -0.44807362,  0.99937328],
       [-0.11038724, -0.98437664,  0.24954012],
       [ 0.92175127, -0.11038724, -0.85780309]])
```

- *np.exp*
  - *Descripción:* Calcula la exponencial de cada elemento.
  - *Ejemplo*:
  - ```
    np.exp(Datos_Evaluacion)
    #
    ```

```
Output:
array([[8.22301271e+36, 1.22040329e+39, 1.65163625e+38],
       [5.54062238e+34, 8.22301271e+36, 1.11286375e+36],
       [3.73324200e+32, 5.54062238e+34, 7.49841700e+33]])
```

- *np.log*
  - *Descripción:* Calcula el logaritmo natural de cada elemento.
  - *Ejemplo*:
  - ```
    np.log(Datos_Evaluacion)
    #
    ```

```
Output:
array([[4.44265126, 4.49980967, 4.47733681],
       [4.38202663, 4.44265126, 4.41884061],
       [4.31748811, 4.38202663, 4.35670883]])
```

---

4. Estadísticas

- *np.mean*
  - *Descripción:* Calcula la media.
  - *Ejemplos*: Se calculó la media de calificaciones de cada evaluación por tarea y luego en general.
  - ```
    np.mean(Tarea_1)
    #
    ```

```
Output:
(87.66666666666667)
```

- ```
  np.mean(Tarea_2)
  #
  ```

```
Output:
(82.66666666666667)
```

- ```
  np.mean(Tarea_3)
  #
  ```

```
Output:
(77.66666666666667)
```

- ```
  np.mean(Datos_Evaluacion)
  #
  ```

```
Output:
(82.66666666666667)
```

- *np.median*
  - *Descripción:* Calcula la mediana.
  - *Ejemplos*: Se calculó la mediana de calificaciones de cada evaluación por tarea y luego en general.
  - ```
    np.median(Tarea_1)
    #
    ```

```
Output:
(88.0)
```

- ```
  np.median(Tarea_2)
  #
  ```

```
Output:
(83.0)
```

- ```
  np.median(Tarea_3)
  #
  ```

```
Output:
(78.0)
```

- ```
  np.median(Datos_Evaluacion)
  #
  ```

```
Output:
(83.0)
```
```

- *np.std*
  - *Descripción:* Calcula la desviación estándar.
  - *Ejemplos*: Se calculó la desviación estándar presente en las calificaciones de cada evaluación por tarea y luego en general.
  - ```
    np.std(Tarea_1)
    #
    ```

```
Output:
(2.0548046676563256)
```

  - ```
    np.std(Tarea_2)
    #
    ```

```
Output:
(2.0548046676563256)
```

  - ```
    np.std(Tarea_3)
    #
    ```

```
Output:
(2.0548046676563256)
```

  - ```
    np.std(Datos_Evaluacion)
    #
    ```

```
Output:
(4.570436400267363)
```

- *np.var*
  - *Descripción:* Calcula la varianza.
  - *Ejemplos*: Se calculó la varianza presente en las calificaciones de cada evaluación por tarea y luego en general.
  - ```
    np.var(Tarea_1)
    #
    ```

```
Output:
(4.222222222222222)
```

  - ```
    np.var(Tarea_2)
    #
    ```

```
Output:
(4.222222222222222)
```

- ```
  np.var(Tarea_3)
  #
  ```

```
Output:
(4.222222222222222)
```

- ```
  np.var(Datos_Evaluacion)
  #
  ```

```
Output:
(20.88888888888889)
```

- *np.min*
  - *Descripción:* Encuentra el valor mínimo.
  - *Ejemplos*: Se encontró la calificación mínima de cada evaluación por tarea y luego en general.
  - ```
    np.min(Tarea_1)
    #
    ```

```
Output:
(85)
```

- ```
  np.min(Tarea_2)
  #
  ```

```
Output:
(80)
```

- ```
  np.min(Tarea_3)
  #
  ```

```
Output:
(75)
```

- ```
  np.min(Datos_Evaluacion)
  #
  ```

```
Output:
(75)
```

- *np.max*
  - *Descripción:* Encuentra el valor máximo.
  - *Ejemplos*: Se encontró la calificación máxima de cada evaluación por tarea y luego en general.

- `np.max(Tarea_1)`
  `#`

```
Output:
(90)
```

- `np.max(Tarea_2)`
  `#`

```
Output:
(85)
```

- `np.max(Tarea_3)`
  `#`

```
Output:
(80)
```

- `np.max(Datos_Evaluacion)`
  `#`

```
Output:
(90)
```

- *np.sum*
  - *Descripción:* Suma los elementos.
  - *Ejemplos*: Se sumaron las calificaciones de cada evaluación por tarea y luego en general.
  - `np.sum(Tarea_1)`
    `#`

```
Output:
(263)
```

- `np.sum(Tarea_2)`
  `#`

```
Output:
(248)
```

- `np.sum(Tarea_3)`
  `#`

```
Output:
(233)
```

```
np.sum(Datos_Evaluacion)
#
```

```
Output:
(744)
```

- *np.cumsum*
  - *Descripción:* Suma acumulada de elementos.
  - *Ejemplos*: Se realizó la suma acumulada de las calificaciones de cada evaluación por tarea y luego en general.
  - ```
    np.cumsum(Tarea_1)
    #
    ```

```
Output:
array([ 85, 175, 263])
```

- ```
  np.cumsum(Tarea_2)
  #
  ```

```
Output:
array([ 80, 165, 248])
```

- ```
  np.cumsum(Tarea_3)
  #
  ```

```
Output:
array([ 75, 155, 233])
```

- ```
  np.cumsum(Datos_Evaluacion)
  #
  ```

```
Output:
array([ 85, 175, 263, 343, 428, 511, 586, 666, 744])
```

- *np.percentile*
  - *Descripción:* Calcula el percentil.
  - *Ejemplo*:
  - ```
    np.percentile(Datos_Evaluacion, 25)
    #
    ```

```
Output:
(80)
```

```

5. Álgebra Lineal

- *np.dot*
  - *Descripción:* Calcula el producto punto de dos arrays.
  - *Ejemplos*:
  - ```
    np.dot(Tarea_1, Tarea_2)
    #
    ```

```
Output:
(21754)
```

  - ```
    np.dot(Tarea_2, Tarea_3)
    #
    ```

```
Output:
(19274)
```

  - ```
    np.dot(Tarea_1, Tarea_3)
    #
    ```

```
Output:
(20439)
```

- *np.matmul*
  - *Descripción:* Calcula el producto matricial.
  - *Ejemplos*:
  - ```
    np.matmul(Tarea_1, Tarea_2)
    #
    ```

```
Output:
(21754)
```

  - ```
    np.matmul(Tarea_2, Tarea_3)
    #
    ```

```
Output:
(5394992)
```

  - ```
    np.matmul(Tarea_1, Tarea_3)
    #
    ```

```
Output:
(5721302)
```

- *np.linalg.det*
  - *Descripción:* Calcula el determinante de una matriz.
  - *Ejemplo*:
  - ```
    np.linalg.det (Datos_Evaluacion)
    #
    ```

```
Output:
(0.0)
```

- *np.linalg.eig*
  - *Descripción:* Calcula los valores propios y vectores propios.
  - *Ejemplo*:
  - ```
    np.linalg.eig (Datos_Evaluacion)
    #
    ```

```
Output:
EigResult(eigenvalues=array([ 2.47818415e+02,  1.81584568e-01, -7.70095880e-15]),
eigenvectors=array([[-0.61154985, -0.71558811, -0.32444284],
       [-0.57664649, -0.00855904, -0.48666426],
       [-0.54174312,  0.69847004,  0.81110711]]))
```

- *np.linalg.svd*
  - *Descripción:* Descomposición en valores singulares.
  - *Ejemplo*:
  - ```
    np.linalg.svd (Datos_Evaluacion)
    #
    ```

```
Output:
SVDResult(U=array([[-0.61150482,  0.67778698,  0.40824829],
       [-0.57664838, -0.02846017, -0.81649658],
       [-0.54179194, -0.73470731,  0.40824829]]), S=array([2.48378557e+02,
3.03964824e-01, 3.44845063e-16]), Vh=array([[-0.55859965, -0.59342442,
-0.57949451],
       [ 0.76335003, -0.64109699, -0.07931818],
       [ 0.32444284,  0.48666426, -0.81110711]]))
```

---

6. Funciones de Indexación y Selección

- *np.where*
  - *Descripción:* Selecciona elementos según una condición.
  - *Ejemplo*:
```

```
np.where (Datos_Evaluacion > 0)
#
```

```
Output:
(array([0, 0, 0, 1, 1, 1, 2, 2, 2]), array([0, 1, 2, 0, 1, 2, 0, 1, 2]))
```

- *np.take*
  - *Descripción:* Selecciona elementos por índices.
  - *Ejemplo*: En este caso los índices escogidos fueron 1,4,8
  - ```
    np.take (Datos_Evaluacion, [1,4,8])
    #
    ```

```
Output:
array([90, 85, 78])
```

- *np.choose*
  - *Descripción:* Construye un array seleccionando de varias opciones.
  - *Ejemplo*:
  - ```
    opciones = [0] * 16
    opciones[10:] = [1] * 6
    resultado = np.choose(Datos_Evaluacion - 75, opciones)  # Restamos 75
    para que los índices empiecen en 0
    print(resultado)
    ```

```
Output:
[[1 1 1]
 [0 1 0]
 [0 0 0]]
```

- *np.nonzero*
  - *Descripción:* Encuentra índices de elementos no cero.
  - *Ejemplo*:
  - ```
    np.nonzero (Datos_Evaluacion)
    #
    ```

```
Output:
(array([0, 0, 0, 1, 1, 1, 2, 2, 2]), array([0, 1, 2, 0, 1, 2, 0, 1, 2]))
```

- *np.extract*
  - *Descripción:* Extrae elementos según una condición.
  - *Ejemplo*: Se extrajeron las calificaciones mayores a 80.
  - ```
    np.extract (Datos_Evaluacion > 80, Datos_Evaluacion)
    #
    ```

```
Output:
array([85, 90, 88, 85, 83])
```

Conclusiones

- Numpy proporciona un conjunto de herramientas muy eficientes para la creación y manipulación de arrays, permitiendo realizar transformaciones y operaciones complejas de manera sencilla y rápida.
- La gran cantidad de funciones disponibles en Numpy, desde operaciones básicas hasta herramientas avanzadas de álgebra lineal, estadística e indexación son de gran ayuda en tareas de análisis de datos, donde la exactitud es crucial, especialmente en el manejo de grandes volúmenes de información.