

```
In [1]: # presentado por Yoan Esteban Lopez Garcia - 1112787002  
# se importa la libreria numpy  
import numpy as np  
  
# se crea un vector con 6 elementos  
a = np.array([0,1,2,3,4,5])  
  
# Se imprime el array  
print(a, '\n')  
  
# numero de dimensiones del array  
print(a.ndim, '\n')  
  
# numero de elementos del array  
print(a.shape)
```

[0 1 2 3 4 5]

1

(6,)

```
In [2]: # se cambia la estructura del array  
b = a.reshape((3,2))  
print(b, '\n')  
  
# se verifican os cambios  
print(b.ndim, '\n')  
print(b.shape)
```

[[0 1]  
 [2 3]  
 [4 5]]

2

(3, 2)

```
In [4]: # Se modifica el primer elemento de la segunda fila  
b[1][0] = 77  
  
# se verifica el cambio  
print(b)
```

[[ 0 1]  
 [77 3]  
 [ 4 5]]

```
In [5]: # debido a que el array b se construyo con base en el array a, el cam  
bio afecta tambien al array a  
print(a)
```

[ 0 1 77 3 4 5]

```
In [6]: # se realiza una copia del array
c = a.reshape((3,2)).copy()
print(c, '\n')

#se cambio el primer valor de c
c[0][0] = -99

# el array a no se modifica
print(a, '\n')

# el array c queda modificado
print(c)
```

```
[[ 0  1]
 [77  3]
 [ 4  5]]

[ 0  1 77  3  4  5]

[[-99  1]
 [ 77  3]
 [  4  5]]
```

```
In [7]: # las operaciones se propagan a lo largo del array
d = np.array([1,2,3,4,5])

# s multiplican los elementos por 2
print(d*2, '\n')

# se elevan al cuadrado los elementos del array
print(d**2)
```

```
[ 2  4  6  8 10]

[ 1  4  9 16 25]
```

```
In [9]: # nueva definicion para el array a
a = np.array([1,2,3,4,5])

# itera sobre todos los elementos del array
print(a>4, '\n')

# se ejecuta la instruccion para los elementos que cumplan la condici
on: " elemento > 4"
# se combinan el contenido de los elementos mayores a 4
a[a>4] = 1

print(a, '\n')

# los elementos cuyo contenido es igual a 1, rciben como nuevo valor
el numero 777
a[a==1] = 777
print(a, '\n')
```

```
[False False False False  True]
```

```
[1 2 3 4 1]
```

```
[777  2  3  4 777]
```

```
In [11]: # control de valores erroneos
c = np.array([1,2,np.NAN,3,4])
print(c, '\n')

# se verifica la existencia de los valores nan
print(np.isnan(c), '\n')

# se eligen todos los valores que no son nan
print(c[~np.isnan(c)], '\n')

# se calcula el promedio de los valores que no son nan
print(np.mean(c[~np.isnan(c)]))
```

```
[ 1.  2. nan  3.  4.]
```

```
[False False  True False False]
```

```
[1.  2.  3.  4.]
```

```
2.5
```

```
In [ ]: # PROYECTO machine learning
```

Enunciado

```
# Enunciado: Una empresa vende el servicio de proporcionar algoritmos
# de aprendizaje automático a través de HTTP.
# Con el éxito creciente de la empresa, aumenta la demanda de una mej
# or infraestructura para atender todas las
# solicitudes web entrantes. No queremos asignar demasiados recursos,
# ya que sería demasiado costoso.
# Por otro lado, perderemos dinero si no hemos reservado suficientes
# recursos para atender todas las solicitudes entrantes.
# Ahora, la pregunta es, ¿cuándo alcanzaremos el límite de nuestra in
# fraestructura actual, que se estima en
# 100.000 solicitudes por hora?. Nos gustaría saberlo de antemano cua
# ndo tenemos que solicitar servidores adicionales
# en la nube para atender todas las solicitudes con éxito sin pagar p
# or las no utilizadas.
```

```
In [12]: # Vamos a desarrollar un programa de machine learning (básico)
# El siguiente es un paquete de datos a ser procesados:
# La primera columna es: Número de horas
# La segunda columna es: Número de tareas ejecutadas
```

```
data = np.genfromtxt("web_traffic.tsv", delimiter="\t")
print(data[:10], '\n')
```

```
# numero de datos
print(data.shape)
```

```
[[1.000e+00 2.272e+03]
 [2.000e+00      nan]
 [3.000e+00 1.386e+03]
 [4.000e+00 1.365e+03]
 [5.000e+00 1.488e+03]
 [6.000e+00 1.337e+03]
 [7.000e+00 1.883e+03]
 [8.000e+00 2.283e+03]
 [9.000e+00 1.335e+03]
 [1.000e+01 1.025e+03]]
```

```
(743, 2)
```

```
In [24]: # se divide el array en dos vectores columna: x, y
x = data[:,0]
y = data[:,1]

# se muestran los valores en x, y
print(x, '\n')
print(y, '\n')
```

```
[ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10. 11. 12. 13. 14.
 15. 16. 17. 18. 19. 20. 21. 22. 23. 24. 25. 26. 27. 28.
 29. 30. 31. 32. 33. 34. 35. 36. 37. 38. 39. 40. 41. 42.
 43. 44. 45. 46. 47. 48. 49. 50. 51. 52. 53. 54. 55. 56.
 57. 58. 59. 60. 61. 62. 63. 64. 65. 66. 67. 68. 69. 70.
 71. 72. 73. 74. 75. 76. 77. 78. 79. 80. 81. 82. 83. 84.
 85. 86. 87. 88. 89. 90. 91. 92. 93. 94. 95. 96. 97. 98.
 99. 100. 101. 102. 103. 104. 105. 106. 107. 108. 109. 110. 111. 112.
 113. 114. 115. 116. 117. 118. 119. 120. 121. 122. 123. 124. 125. 126.
 127. 128. 129. 130. 131. 132. 133. 134. 135. 136. 137. 138. 139. 140.
 141. 142. 143. 144. 145. 146. 147. 148. 149. 150. 151. 152. 153. 154.
 155. 156. 157. 158. 159. 160. 161. 162. 163. 164. 165. 166. 167. 168.
 169. 170. 171. 172. 173. 174. 175. 176. 177. 178. 179. 180. 181. 182.
 183. 184. 185. 186. 187. 188. 189. 190. 191. 192. 193. 194. 195. 196.
 197. 198. 199. 200. 201. 202. 203. 204. 205. 206. 207. 208. 209. 210.
 211. 212. 213. 214. 215. 216. 217. 218. 219. 220. 221. 222. 223. 224.
 225. 226. 227. 228. 229. 230. 231. 232. 233. 234. 235. 236. 237. 238.
 239. 240. 241. 242. 243. 244. 245. 246. 247. 248. 249. 250. 251. 252.
 253. 254. 255. 256. 257. 258. 259. 260. 261. 262. 263. 264. 265. 266.
 267. 268. 269. 270. 271. 272. 273. 274. 275. 276. 277. 278. 279. 280.
 281. 282. 283. 284. 285. 286. 287. 288. 289. 290. 291. 292. 293. 294.
 295. 296. 297. 298. 299. 300. 301. 302. 303. 304. 305. 306. 307. 308.
 309. 310. 311. 312. 313. 314. 315. 316. 317. 318. 319. 320. 321. 322.
 323. 324. 325. 326. 327. 328. 329. 330. 331. 332. 333. 334. 335. 336.
 337. 338. 339. 340. 341. 342. 343. 344. 345. 346. 347. 348. 349. 350.
 351. 352. 353. 354. 355. 356. 357. 358. 359. 360. 361. 362. 363. 364.
 365. 366. 367. 368. 369. 370. 371. 372. 373. 374. 375. 376. 377. 378.
 379. 380. 381. 382. 383. 384. 385. 386. 387. 388. 389. 390. 391. 392.
 393. 394. 395. 396. 397. 398. 399. 400. 401. 402. 403. 404. 405. 406.]
```

```

6.
407. 408. 409. 410. 411. 412. 413. 414. 415. 416. 417. 418. 419. 42
0.
421. 422. 423. 424. 425. 426. 427. 428. 429. 430. 431. 432. 433. 43
4.
435. 436. 437. 438. 439. 440. 441. 442. 443. 444. 445. 446. 447. 44
8.
449. 450. 451. 452. 453. 454. 455. 456. 457. 458. 459. 460. 461. 46
2.
463. 464. 465. 466. 467. 468. 469. 470. 471. 472. 473. 474. 475. 47
6.
477. 478. 479. 480. 481. 482. 483. 484. 485. 486. 487. 488. 489. 49
0.
491. 492. 493. 494. 495. 496. 497. 498. 499. 500. 501. 502. 503. 50
4.
505. 506. 507. 508. 509. 510. 511. 512. 513. 514. 515. 516. 517. 51
8.
519. 520. 521. 522. 523. 524. 525. 526. 527. 528. 529. 530. 531. 53
2.
533. 534. 535. 536. 537. 538. 539. 540. 541. 542. 543. 544. 545. 54
6.
547. 548. 549. 550. 551. 552. 553. 554. 555. 556. 557. 558. 559. 56
0.
561. 562. 563. 564. 565. 566. 567. 568. 569. 570. 571. 572. 573. 57
4.
575. 576. 577. 578. 579. 580. 581. 582. 583. 584. 585. 586. 587. 58
8.
589. 590. 591. 592. 593. 594. 595. 596. 597. 598. 599. 600. 601. 60
2.
603. 604. 605. 606. 607. 608. 609. 610. 611. 612. 613. 614. 615. 61
6.
617. 618. 619. 620. 621. 622. 623. 624. 625. 626. 627. 628. 629. 63
0.
631. 632. 633. 634. 635. 636. 637. 638. 639. 640. 641. 642. 643. 64
4.
645. 646. 647. 648. 649. 650. 651. 652. 653. 654. 655. 656. 657. 65
8.
659. 660. 661. 662. 663. 664. 665. 666. 667. 668. 669. 670. 671. 67
2.
673. 674. 675. 676. 677. 678. 679. 680. 681. 682. 683. 684. 685. 68
6.
687. 688. 689. 690. 691. 692. 693. 694. 695. 696. 697. 698. 699. 70
0.
701. 702. 703. 704. 705. 706. 707. 708. 709. 710. 711. 712. 713. 71
4.
715. 716. 717. 718. 719. 720. 721. 722. 723. 724. 725. 726. 727. 72
8.
729. 730. 731. 732. 733. 734. 735. 736. 737. 738. 739. 740. 741. 74
2.
743.]

[2272.    nan 1386. 1365. 1488. 1337. 1883. 2283. 1335. 1025. 1139. 14
77.
1203. 1311. 1299. 1494. 1159. 1365. 1272. 1246. 1071. 1876.    nan 14
10.
925. 1533. 2104. 2113. 1993. 1045. 2090. 2227. 1413. 1718. 1721. 12
91.

```

```

1838. 2540. 1608. 2455. 1929. 1767. 1203. 1761. 1723. 2160. 808.
nan
1324. 1809. 1933. 1351. 2013. 1207. 2170. 1700. 1899. 1757. 1475. 19
21.
1971. 1809. 1365. 1775. 1687. 1706. 1353. 1316. 1512. 2430. 1788. 13
80.
1357. 990. 1586. 2057. 1690. 1458. 1201. 1949. 1493. 1653. 1217. 14
57.
1179. 1484. 2730. 1414. 1060. 1573. 1260. 1216. 981. 1345. nan 16
67.
730. 1034. 1628. 1155. 1305. 1444. 2242. 1842. 1210. 1384. 1313. 15
08.
1796. 1265. 1090. 2159. 1167. 1391. 1445. 1196. 1049. 1999. 472. 12
85.
1737. 1534. 2636. 1372. 1325. 833. 1200. 2431. 1740. 2121. 1726. 13
44.
1072. 1386. 1054. 1051. 1270. 1857. 1437. 2016. 1352. 909. 1761. 10
09.
2035. 1534. 1708. 733. 1455. 1332. 1606. 1065. 1291. nan 1495. 19
28.
2249. 987. 1023. 875. 1569. 1032. 1079. 1087. 1152. 961. 1232. 21
88.
1179. 1475. 1612. 921. 2432. 1650. 1077. 823. 1578. 1872. 1669.
nan
nan 1407. 1619. 894. 1948. 2298. 2163. 1108. 1731. 1601. 1684. 20
25.
1688. 1736. 1474. 1770. 1348. 1570. 1861. 1458. 2282. 1553. 2323. 12
02.
1768. 2184. 1329. 1781. 1242. nan 1454. 1501. 875. 1521. 2611. 19
48.
1707. 1335. 2211. 1358. 2501. 1764. 1527. 1421. 1949. 2156. 1503. 16
58.
1032. 1536. 1345. 2022. 2035. 2109. 1587. 1666. 1064. 1457. 2399. 14
49.
2406. 1831. 1423. 1754. 1641. 1428. 1928. 1618. 1361. 1273. 1300. 9
97.
1163. 1480. 2131. 1833. 1161. 1168. 1570. 1675. 966. 1395. 1638. 17
13.
1799. 1917. 1894. 1009. 1003. 1962. 1730. 731. 2166. 1059. 1520. 17
08.
1227. 1085. 1045. 1720. 1495. 960. 1420. 1318. 740. 878. 1357. 23
18.
1544. 1583. 1693. 1153. 1469. 2004. 1114. 1281. 1500. 1409. 942. 7
92.
704. 1584. 1004. 795. 1000. 2156. 639. 1391. 1644. 1398. 967. 15
78.
1068. 1419. 1784. 1952. 996. 1485. 1419. 1534. 1633. 1013. 2085. 31
02.
1859. 983. 2169. 2086. 2204. 1578. 1526. 1725. 936. 1678. 1573. 11
87.
1535. 1333. 1701. 1925. 1651. 1491. 1800. 1976. 1246. 2141. 1351. 15
05.
1377. 2386. 1304. 1424. 1881. 2393. 1599. 1444. 1985. 1158. 2098. 15
40.
1410. 2115. 1278. 2039. 2021. 1901. 1139. 1903. 2074. 3661. 1799. 24
31.
1499. 1040. 1825. 1733. 1727. 1076. 1598. 1146. 1534. 1514. 1540. 14

```



45.  
 1248. 1710. 2114. 1816. 1759. 2173. 1791. 1710. 1930. 1803. 1879. 22  
 89.  
 1839. 1641. 1374. 1524. 1360. 1303. 1654. 1928. 1558. 1736. 1752. 10  
 42.  
 1201. 1498. 2101. 2389. 1326. 1285. 1413. 1970. 1242. 1920. 1163. 16  
 51.  
 1300. 1850. 1799. 1703. 1627. 1522. 1409. 2631. 1647. 1536. 1433. 17  
 49.  
 1274. 1658. 1579. 1607. 1382. 1322. 1168. 1067. 1890. 1659. 1064. 8  
 68.  
 1288. 2166. 1382. 1417. 2018. 1777. 1596. 1420. 1324. 1899. 1513. 16  
 83.  
 1369. 1266. 1034. 2045. 1498. 1607. 1331. 1132. 1238. 2298. 1241. 20  
 39.  
 1177. 1220. 1746. 1917. 1165. 860. 1830. 1170. 1229. 1274. 1900. 18  
 67.  
 1610. 1963. 1669. 1291. 1751. 1335. 1323. 1652. 2086. 1437. 1731. 19  
 50.  
 2203. 2260. 1580. 1562. 1860. 1793. 1000. 1912. 2475. 2105. 1732. 23  
 09.  
 1874. 1816. 1097. 2015. 2241. 2772. 1320. 2738. 1389. 2251. 2167. 20  
 28.  
 1590. 2341. 2011. 1613. 1671. 1999. 2894. 2637. 1884. 2404. 2255. 19  
 60.  
 1847. 1558. 1559. 2040. 1996. 2051. 1803. 1969. 1937. 2082. 1408. 27  
 31.  
 2220. 2330. 2437. 1915. 1986. 2145. 2276. 2157. 2626. 1536. 1558. 30  
 44.  
 2246. 2383. 2009. 1972. 2145. 2102. 2327. 1732. 2640. 1992. 2199. 23  
 93.  
 2190. 2495. 2390. 2435. 1737. 2052. 2034. 1834. 3005. 1429. 2215. 19  
 02.  
 2284. 1993. 2059. 2169. 1981. 2098. 2506. 1911. 2560. 1301. 1859. 22  
 86.  
 1734. 2156. 2402. 2404. 3244. 1977. 2412. 2007. 2014. 1564. 2022. 17  
 72.  
 2582. 1845. 1621. 1770. 2021. 2355. 1996. 2127. 2113. 1935. 2125. 17  
 86.  
 2276. 2978. 2542. 2112. 1968. 2368. 2241. 2073. 2122. 2166. 2575. 25  
 00.  
 2181. 1967. 2072. 2027. 2345. 2024. 2249. 2455. 2265. 2425. 2851. 19  
 97.  
 3298. 2366. 1853. 2896. 2537. 2300. 2849. 2974. 1931. 3009. 2538. 27  
 82.  
 2491. 2408. 2003. 2752. 2576. 2818. 2683. 2628. 2994. 2303. 2771. 26  
 07.  
 2704. 2839. 3256. 3025. 2684. 3006. 3310. 3183. 2523. 3401. 2840. 31  
 93.  
 2969. 3337. 3464. 3264. 3535. 3089. 2935. 3007. 4000. 3488. 2814. 33  
 82.  
 2901. 4260. 3785. 4139. 3588. 3343. 3118. 3456. 4150. 3827. 3992. 46  
 67.  
 3301. 3931. 4496. 3402. 3672. 3550. 4230. 3805. 3352. 3602. 4015. 35  
 48.  
 3316. 3932. 3596. 5289. 3561. 3990. 3889. 3636. 3799. 4188. 5248. 41  
 76.

4829. 4346. 4224. 4813. 3997. 4357. 4322. 4156. 4630. 4415. 4410. 4724.  
4363. 4798. 4749. 5143. 4906. 4309. 4970. 4813. 5392. 5906. 4881.]

```
In [25]: # dimension de los vectores x, y
print(x.ndim, '\n')
print(y.ndim, '\n')

# elemtos contenidos en los vectores x, y
print(x.shape, '\n')
print(y.shape)
```

1

1

(743,)

(743,)

```
In [15]: # investigamos el numero de valores nan que contiene el vector y
print(np.sum(np.isnan(y)))
```

8

```
In [27]: # numero de elementos x, y antes de ser comprimidos
print(x.shape, '\n')
print(y.shape, '\n')

# se eliminan los elementos nan tanto de x como de y
x = x[~np.isnan(x)]
y = y[~np.isnan(y)]

# se cuenta el numero de elementos tanto de x como de y
print(x.shape, '\n')
print(y.shape, '\n')
```

(735,)

(735,)

(735,)

(735,)

```
In [29]: # se importa la libreria para graficar
import matplotlib.pyplot as plt

# dibuja los puntos (x, y) con circulos de tamaño 10
plt.scatter(x, y, s=10)

# titulos de la grafica
plt.title("trafico web en el ultimo mes")
plt.xlabel("tiempo")
plt.ylabel("Accesos/Hora")

plt.xticks([w*7*24 for w in range(10)], ['semana %i' % w for w in range(10)])
plt.autoscale(tight=True)

# dibuja una cuadrícula punteada ligeramente opaca
plt.grid(True, linestyle = '-', color='0.75')

# Muestra el grafico
plt.show()
```

