**INTRODUCCION A LA PROGRAMANACION**

**PYTHON**

**RETO 7-8**

**CAREN DAYANA ROMERO**

**JUAN CAMILO TABORDA**

**UNIVERSIDAD DE ANTIOQUIA**

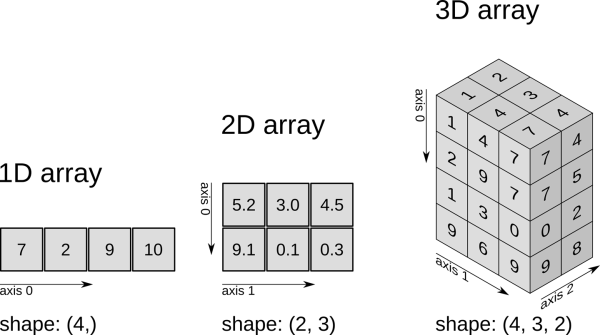
**FACULTAD DE INGENIERIA**

**BOGOTA 9 DE OCTUBRE DE 2023**

**RETO 7-8**

1. Preguntas
2. ¿Qué hace la librería Numpy?

La librería [NumPy](https://www.numpy.org/) es una librería de Python especializada en el cálculo numérico y el análisis de datos, especialmente para Big data. Su objeto principal son los llamados **arrays** que permite representar colecciones de datos de un mismo tipo en varias dimensiones, y funciones muy eficientes para su manipulación, lo cual al procesamiento de datos con mayor rapidez, (hasta 50 veces más) que las listas habituales en Python, lo cual la hace ideal para el procesamiento de vectores y matrices de grandes dimensiones.

Un array es una estructura de datos de un mismo tipo organizada en forma de tabla o cuadricula de distintas dimensiones

Los arrays de NumPy tienen algunas ventajas sobre las listas tradicionales de Python. En primer lugar, utilizan menos memoria y espacio de almacenamiento, que es su principal punto fuerte.Además, NumPy ofrece un mejor rendimiento en términos de velocidad de ejecución. Sin embargo, es más fácil y cómodo de usar. Además, es una herramienta de código abierto, que puede utilizarse de forma totalmente gratuita.

1. ¿En qué situaciones se debe usar la librería Numpy?

NumPy es muy útil para realizar cálculos lógicos y matemáticos sobre vectores y matrices. La usaremos principalmente porque nos permite crear y modificar matrices, y hacer operaciones sobre ellas con facilidad, Esta librería es muy útil para el análisis de datos, data scienie y machine learning, ya que en este tipo de situaciones la información se basa en gran parte en vectores y en matrices. Esto es así porque cualquier representación de datos estructurados o cualquier fichero, se puede observar como si fuera una matriz, es decir, cada una de las filas de la matriz podría ser cada una de las observaciones de ese conjunto de datos, y cada una de las columnas de la matriz podría ser cada una vez las columnas de ese conjunto de datos.

En Python tendría que realizar un ciclo for en Numpy es mas fácil.

Tiene una estructura de datos especial que te permite representar elementos complejos como imágenes, videos, canciones, modelos 3D, etc. Si tienes datos por procesar, NumPy será tu mejor amigo. Además, podrás realizar operaciones numéricas de manera muy sencilla, ahorrándote tener que programar ciclos o cosas extrañas.

1. ¿En qué situaciones no debe usarse la librería Numpy?

numpy ofrece varias formas muy eficientes de crear vectores y manipular datos numéricos. Mientras que una lista de Python puede contener diferentes tipos de datos en su interior, los elementos de un vector numpy serán todos del mismo tipo. Por ello cuando tenemos diferentes tipos de datos no es recomendable utilizar la librería de numpay

1. ¿Qué se necesita para usar la librería Numpy?

Si vas a trabajar con [Notebooks](https://platzi.com/cursos/jupyter-notebook/) en la nube, NumPy ya estará instalado. Pero si vas a trabajar en local, puedes instalarla desde la consola en tu máquina directamente o en un ambiente virtual de la siguiente manera:

pip install numpy

1. ¿Cómo se usa la librería Numpy?

Para comenzar a usar numpy, debemos importar la librería. En este caso la vamos a importar, y darle un sobrenombre al mismo tiempo, np, para que el código sea más legible

import numpy as np

Ahora que tenemos la librería, empecemos creando un vector

La manera más común de crear una un vector o matriz ya inicializada es con la función np.array, que toma una lista (o lista de listas) como parámetro y devuelve una matriz de numpy.

In [3]:

lista=[25,12,15,66,12.5]

vector=np.array(lista)

print(vector)

[ 25. 12. 15. 66. 12.5]

Reto 6

Una compañía de ciencia de datos necesita hacer una prueba de concepto, en la que se verifique la esperanza de tratamiento de información bajo un entorno controlado, usando la librería numpy y su objeto ndarray. Por lo anterior, se solicita lo siguiente:

1. Crear un objeto de tipo ndarray, con números enteros entre 64 hasta 1024.
2. # Crear un objeto de tipo ndarray, con números enteros entre 64 hasta 1024.
3. import numpy as np
4. arreglo = np.arange(64 , 1025)
5. arreglo
6. array([ 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74,
7. 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85,
8. 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96,
9. 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107,
10. 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118,
11. 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 129,
12. 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140,
13. 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151,
14. 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162,
15. 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173,
16. 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184,
17. 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195,
18. 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206,
19. 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217,
20. 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228,
21. 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239,
22. 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250,
23. 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261,
24. 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272,
25. 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283,
26. 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294,
27. 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305,
28. 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316,
29. 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327,
30. 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338,
31. 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349,
32. 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360,
33. 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371,
34. 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382,
35. 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393,
36. 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404,
37. 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415,
38. 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426,
39. 427, 428, 429, 430, 431, 432, 433, 434, 435, 436, 437,
40. 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448,
41. 449, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459,
42. 460, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470,
43. 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481,
44. 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492,
45. 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503,
46. 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514,
47. 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525,
48. 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536,
49. 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547,
50. 548, 549, 550, 551, 552, 553, 554, 555, 556, 557, 558,
51. 559, 560, 561, 562, 563, 564, 565, 566, 567, 568, 569,
52. 570, 571, 572, 573, 574, 575, 576, 577, 578, 579, 580,
53. 581, 582, 583, 584, 585, 586, 587, 588, 589, 590, 591,
54. 592, 593, 594, 595, 596, 597, 598, 599, 600, 601, 602,
55. 603, 604, 605, 606, 607, 608, 609, 610, 611, 612, 613,
56. 614, 615, 616, 617, 618, 619, 620, 621, 622, 623, 624,
57. 625, 626, 627, 628, 629, 630, 631, 632, 633, 634, 635,
58. 636, 637, 638, 639, 640, 641, 642, 643, 644, 645, 646,
59. 647, 648, 649, 650, 651, 652, 653, 654, 655, 656, 657,
60. 658, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668,
61. 669, 670, 671, 672, 673, 674, 675, 676, 677, 678, 679,
62. 680, 681, 682, 683, 684, 685, 686, 687, 688, 689, 690,
63. 691, 692, 693, 694, 695, 696, 697, 698, 699, 700, 701,
64. 702, 703, 704, 705, 706, 707, 708, 709, 710, 711, 712,
65. 713, 714, 715, 716, 717, 718, 719, 720, 721, 722, 723,
66. 724, 725, 726, 727, 728, 729, 730, 731, 732, 733, 734,
67. 735, 736, 737, 738, 739, 740, 741, 742, 743, 744, 745,
68. 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756,
69. 757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767,
70. 768, 769, 770, 771, 772, 773, 774, 775, 776, 777, 778,
71. 779, 780, 781, 782, 783, 784, 785, 786, 787, 788, 789,
72. 790, 791, 792, 793, 794, 795, 796, 797, 798, 799, 800,
73. 801, 802, 803, 804, 805, 806, 807, 808, 809, 810, 811,
74. 812, 813, 814, 815, 816, 817, 818, 819, 820, 821, 822,
75. 823, 824, 825, 826, 827, 828, 829, 830, 831, 832, 833,
76. 834, 835, 836, 837, 838, 839, 840, 841, 842, 843, 844,
77. 845, 846, 847, 848, 849, 850, 851, 852, 853, 854, 855,
78. 856, 857, 858, 859, 860, 861, 862, 863, 864, 865, 866,
79. 867, 868, 869, 870, 871, 872, 873, 874, 875, 876, 877,
80. 878, 879, 880, 881, 882, 883, 884, 885, 886, 887, 888,
81. 889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899,
82. 900, 901, 902, 903, 904, 905, 906, 907, 908, 909, 910,
83. 911, 912, 913, 914, 915, 916, 917, 918, 919, 920, 921,
84. 922, 923, 924, 925, 926, 927, 928, 929, 930, 931, 932,
85. 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943,
86. 944, 945, 946, 947, 948, 949, 950, 951, 952, 953, 954,
87. 955, 956, 957, 958, 959, 960, 961, 962, 963, 964, 965,
88. 966, 967, 968, 969, 970, 971, 972, 973, 974, 975, 976,
89. 977, 978, 979, 980, 981, 982, 983, 984, 985, 986, 987,
90. 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998,
91. 999, 1000, 1001, 1002, 1003, 1004, 1005, 1006, 1007, 1008, 1009,
92. 1010, 1011, 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1019, 1020,
93. 1021, 1022, 1023, 1024])
94. Redimensionar a una matriz bidimensional que tenga las 1024 observaciones, pero que su dimensión tenga tanto el mismo número de filas como de columnas, con la finalidad de obtener una cuadrada.
95. # Redimensionar a una matriz bidimensional que tenga las 1024 observaciones, pero que su dimensión tenga tanto el mismo número de filas como de columnas, con la finalidad de obtener una cuadrada.
96. arreglo = np.arange(64 , 1025).reshape((31,31))
97. arreglo

array([[ 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, 407, 408, 409, 410, 411, 412, 413, 414, 415,

416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426,

427, 428, 429, 430, 431, 432, 433, 434, 435],

[ 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446,

447, 448, 449, 450, 451, 452, 453, 454, 455, 456, 457,

458, 459, 460, 461, 462, 463, 464, 465, 466],

[ 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477,

478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488,

489, 490, 491, 492, 493, 494, 495, 496, 497],

[ 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508,

509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519,

520, 521, 522, 523, 524, 525, 526, 527, 528],

[ 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539,

540, 541, 542, 543, 544, 545, 546, 547, 548, 549, 550,

551, 552, 553, 554, 555, 556, 557, 558, 559],

[ 560, 561, 562, 563, 564, 565, 566, 567, 568, 569, 570,

571, 572, 573, 574, 575, 576, 577, 578, 579, 580, 581,

582, 583, 584, 585, 586, 587, 588, 589, 590],

[ 591, 592, 593, 594, 595, 596, 597, 598, 599, 600, 601,

602, 603, 604, 605, 606, 607, 608, 609, 610, 611, 612,

613, 614, 615, 616, 617, 618, 619, 620, 621],

[ 622, 623, 624, 625, 626, 627, 628, 629, 630, 631, 632,

633, 634, 635, 636, 637, 638, 639, 640, 641, 642, 643,

644, 645, 646, 647, 648, 649, 650, 651, 652],

[ 653, 654, 655, 656, 657, 658, 659, 660, 661, 662, 663,

664, 665, 666, 667, 668, 669, 670, 671, 672, 673, 674,

675, 676, 677, 678, 679, 680, 681, 682, 683],

[ 684, 685, 686, 687, 688, 689, 690, 691, 692, 693, 694,

695, 696, 697, 698, 699, 700, 701, 702, 703, 704, 705,

706, 707, 708, 709, 710, 711, 712, 713, 714],

[ 715, 716, 717, 718, 719, 720, 721, 722, 723, 724, 725,

726, 727, 728, 729, 730, 731, 732, 733, 734, 735, 736,

737, 738, 739, 740, 741, 742, 743, 744, 745],

[ 746, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756,

757, 758, 759, 760, 761, 762, 763, 764, 765, 766, 767,

768, 769, 770, 771, 772, 773, 774, 775, 776],

[ 777, 778, 779, 780, 781, 782, 783, 784, 785, 786, 787,

788, 789, 790, 791, 792, 793, 794, 795, 796, 797, 798,

799, 800, 801, 802, 803, 804, 805, 806, 807],

[ 808, 809, 810, 811, 812, 813, 814, 815, 816, 817, 818,

819, 820, 821, 822, 823, 824, 825, 826, 827, 828, 829,

830, 831, 832, 833, 834, 835, 836, 837, 838],

[ 839, 840, 841, 842, 843, 844, 845, 846, 847, 848, 849,

850, 851, 852, 853, 854, 855, 856, 857, 858, 859, 860,

861, 862, 863, 864, 865, 866, 867, 868, 869],

[ 870, 871, 872, 873, 874, 875, 876, 877, 878, 879, 880,

881, 882, 883, 884, 885, 886, 887, 888, 889, 890, 891,

892, 893, 894, 895, 896, 897, 898, 899, 900],

[ 901, 902, 903, 904, 905, 906, 907, 908, 909, 910, 911,

912, 913, 914, 915, 916, 917, 918, 919, 920, 921, 922,

923, 924, 925, 926, 927, 928, 929, 930, 931],

[ 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942,

943, 944, 945, 946, 947, 948, 949, 950, 951, 952, 953,

954, 955, 956, 957, 958, 959, 960, 961, 962],

[ 963, 964, 965, 966, 967, 968, 969, 970, 971, 972, 973,

974, 975, 976, 977, 978, 979, 980, 981, 982, 983, 984,

985, 986, 987, 988, 989, 990, 991, 992, 993],

[ 994, 995, 996, 997, 998, 999, 1000, 1001, 1002, 1003, 1004,

1005, 1006, 1007, 1008, 1009, 1010, 1011, 1012, 1013, 1014, 1015,

1016, 1017, 1018, 1019, 1020, 1021, 1022, 1023, 1024]])

3. Crear una columna de datos aleatorios de 30 observaciones o registros, en la que se escojan dichos registros desde una lista de 10 nombres. Se debe repetir el proceso anterior para crear un set de números de cédulas aleatorio para cada uno de los nombres.

Lista=[‘Andres’,’Maria’,’Manuel’,’Daniel’,’Sarah’,’Cristian’,’Violetta’,’Lucia’,’Jackson’,’Jose’]

# crear columna de datos aleatorios de 30 observaciones

Lista = ["Andres" , "Maria" , "Manuel" , "Daniel" , "Sarah" , "Cristian" , "Violetta" , "Lucia" , "Jackson" , "Jose"]

import numpy as np

arreglo = np.array(Lista)

arreglo

array(['Andres', 'Maria', 'Manuel', 'Daniel', 'Sarah', 'Cristian', 'Violetta', 'Lucia', 'Jackson', 'Jose'], dtype='<U8')

# creando cedulas con numeros aleatorios

cedulas = np.random.rand(10)

cedulas

array([0.49946496, 0.70464538, 0.8744333 , 0.57659795, 0.13765054, 0.96854183, 0.59228618, 0.74178292, 0.98612519, 0.76219334])

4. Concatenar un nuevo vector con las observaciones de los nombres y cédulas.

# concatenar vector con las observaciones de nombres y cedulas

np.ma.concatenate(([arreglo], [cedulas]), axis=0)

0 s

# concatenar vector con las observaciones de nombres y cedulas

np.ma.concatenate(([arreglo], [cedulas]), axis=0)

output

masked\_array(

data=[['Andres', 'Maria', 'Manuel', 'Daniel', 'Sarah', 'Cristian',

'Violetta', 'Lucia', 'Jackson', 'Jose'],

['0.4222703949951756', '0.844409583506493', '0.5108754549543203',

'0.407695341857944', '0.39162107428256077',

'0.5635199224827401', '0.04311998265173744',

'0.036202469299125295', '0.7781272079642793',

'0.9455931317281825']],

mask=False,

fill\_value='N/A',

dtype='<U32')

1. Mostrar la posición del número más grande de las cédulas generadas del vector creado en el punto 4.
2. np.amax(cedulas)

0.971655530551743