02-numpy

March 21, 2020

1 NumPy

```
In [1]: import numpy as np
```

```
In [2]: L1 = [1,2,3,4,5,6,7,8]# Crear una lista
```

```
In [5]: x1 = np.array(L1)
        x1

Out[5]: array([1, 2, 3, 4, 5, 6, 7, 8])
```

```
In [44]: x2 = np.array(L1, dtype='float32')
         x2

Out[44]: array([1., 2., 3., 4., 5., 6., 7., 8.], dtype=float32)
```

- bool_
- int_, intc, intp, int8, int16, int32, int64
- uint8, uint16, uint32, uint64
- float_, float16, float32, float64 (+/-999.9999999e99999)
- complex_, complex64, complex128 (a+bi, a,b ∈ float_)

```
In [8]: np.zeros((3,4))

Out[8]: array([[0., 0., 0., 0.],
               [0., 0., 0., 0.],
               [0., 0., 0., 0.]])
```

```
In [6]: np.ones((4,3))

Out[6]: array([[1., 1., 1.],
               [1., 1., 1.],
               [1., 1., 1.],
               [1., 1., 1.]])
```

```
In [9]: np.arange(10)

Out[9]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
In [10]: np.arange(3,12,dtype=np.float)

Out[10]: array([ 3.,  4.,  5.,  6.,  7.,  8.,  9., 10., 11.])

In [13]: np.arange(4,5,0.1)

Out[13]: array([4. , 4.1, 4.2, 4.3, 4.4, 4.5, 4.6, 4.7, 4.8, 4.9])

In [12]: np.linspace(1., 7., 12)

Out[12]: array([1.        , 1.54545455, 2.09090909, 2.63636364, 3.18181818,
                3.72727273, 4.27272727, 4.81818182, 5.36363636, 5.90909091,
                6.45454545, 7.        ])

In [11]: np.eye(5)

Out[11]: array([[1., 0., 0., 0., 0.],
                [0., 1., 0., 0., 0.],
                [0., 0., 1., 0., 0.],
                [0., 0., 0., 1., 0.],
                [0., 0., 0., 0., 1.]])

In [52]: x = np.arange(24)
         x

Out[52]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
                17, 18, 19, 20, 21, 22, 23])

In [29]: x.reshape((6,4))

Out[29]: array([[ 0,  1,  2,  3],
                [ 4,  5,  6,  7],
                [ 8,  9, 10, 11],
                [12, 13, 14, 15],
                [16, 17, 18, 19],
                [20, 21, 22, 23]])

In [53]: x.reshape(3,8)

Out[53]: array([[ 0,  1,  2,  3,  4,  5,  6,  7],
                [ 8,  9, 10, 11, 12, 13, 14, 15],
                [16, 17, 18, 19, 20, 21, 22, 23]])

In [51]: x = np.array([[1,2,3,4], [5,6,7,8]])
         x

Out[51]: array([[1, 2, 3, 4],
                [5, 6, 7, 8]])

In [32]: np.ravel(x)
```

```
Out[32]: array([1, 2, 3, 4, 5, 6, 7, 8])
```

```
In [33]: x.flatten()
```

```
Out[33]: array([1, 2, 3, 4, 5, 6, 7, 8])
```

```
In [36]: np.transpose(x)
```

```
Out[36]: array([[1, 5],
                [2, 6],
                [3, 7],
                [4, 8]])
```

```
In [37]: x
```

```
Out[37]: array([[1, 2, 3, 4],
                [5, 6, 7, 8]])
```

```
In [23]: np.resize(x, (5, 3))
```

```
Out[23]: array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 1],
                [2, 3, 4],
                [5, 6, 7]])
```

1.0.1 Ejercicios

1. Crear un array de datos con valores entre 5 y 120.
2. Crear una matriz 4x4 con los valores desde 0 hasta 15.
3. Crear la identidad 7x7
4. Crear un array de 20 elementos y transformarlos en una matrix 5x4
5. Crear un array con 20 números con los valores entre 0 y 5 espaciados de forma uniforme

```
In [40]: a = np.arange(5,121)
         a
```

```
Out[40]: array([  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])
```

1.0.2 Ejemplo uso de *LaTeX*:

La solucion de la ecuacion

$$ax^2 - bx + c = 0$$

es

$$x = \frac{-b \pm \sqrt{b^2 - 4 \cdot a \cdot c}}{2 \cdot a}$$

```
In [49]: np.arange(0,16).reshape(4,4)

Out[49]: array([[ 0,  1,  2,  3],
                [ 4,  5,  6,  7],
                [ 8,  9, 10, 11],
                [12, 13, 14, 15]])
```

```
In [56]: np.eye(7) #ó tambien np.identity(7)

Out[56]: array([[1., 0., 0., 0., 0., 0., 0.],
                [0., 1., 0., 0., 0., 0., 0.],
                [0., 0., 1., 0., 0., 0., 0.],
                [0., 0., 0., 1., 0., 0., 0.],
                [0., 0., 0., 0., 1., 0., 0.],
                [0., 0., 0., 0., 0., 1., 0.],
                [0., 0., 0., 0., 0., 0., 1.]])
```

```
In [68]: np.arange(1,21).reshape(5,4)

Out[68]: array([[ 1,  2,  3,  4],
                [ 5,  6,  7,  8],
                [ 9, 10, 11, 12],
                [13, 14, 15, 16],
                [17, 18, 19, 20]])
```

```
In [73]: np.linspace(0,5,20)

Out[73]: array([0.        , 0.26315789, 0.52631579, 0.78947368, 1.05263158,
                1.31578947, 1.57894737, 1.84210526, 2.10526316, 2.36842105,
                2.63157895, 2.89473684, 3.15789474, 3.42105263, 3.68421053,
                3.94736842, 4.21052632, 4.47368421, 4.73684211, 5.        ])
```

1.1 Propiedades de los arrays

```
In [27]: x = np.arange(12)
         x = x.reshape((3,4))
         x

Out[27]: array([[ 0,  1,  2,  3],
                [ 4,  5,  6,  7],
                [ 8,  9, 10, 11]])
```

```
In [29]: x.ndim

Out[29]: 2

In [30]: x.shape

Out[30]: (3, 4)

In [31]: x.size

Out[31]: 12

In [32]: x.dtype

Out[32]: dtype('int64')

In [33]: x.itemsize

Out[33]: 8

In [34]: x.data

Out[34]: <memory at 0x10fa2b2d0>

In [35]: x[2]

Out[35]: array([ 8,  9, 10, 11])

In [36]: x[2,1]

Out[36]: 9

In [37]: x.shape = (4,3)
         x

Out[37]: array([[ 0,  1,  2],
                [ 3,  4,  5],
                [ 6,  7,  8],
                [ 9, 10, 11]])

In [38]: x[1:3, 0:2]

Out[38]: array([[3, 4],
                [6, 7]])

In [39]: y = np.arange(12)
         y

Out[39]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11])

In [40]: y[3:8]
```

```
Out[40]: array([3, 4, 5, 6, 7])

In [41]: y[1:7:2]

Out[41]: array([1, 3, 5])

In [42]: z = np.arange(10, 6, -1)
         z

Out[42]: array([10,  9,  8,  7])

In [43]: y[z]

Out[43]: array([10,  9,  8,  7])

In [44]: x = np.arange(50)
         x

Out[44]: array([ 0,  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])

In [45]: x[x>30]

Out[45]: array([31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47,
                48, 49])

In [47]: cond = (x < 25)
         cond

Out[47]: array([ True,  True,  True,  True,  True,  True,  True,  True,  True,
                 True,  True,  True,  True,  True,  True,  True,  True,  True,
                 True,  True,  True,  True,  True,  True,  True, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False, False, False, False, False,
                False, False, False, False, False])

In [48]: x[cond]

Out[48]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
                17, 18, 19, 20, 21, 22, 23, 24])

In [49]: x[12:24] = 1
         x

Out[49]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11,  1,  1,  1,  1,  1,
                 1,  1,  1,  1,  1,  1,  1, 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])

In [50]: x[13:16] = [6,9,12]
         x
```

```
Out[50]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11,  1,  6,  9, 12,  1,
                 1,  1,  1,  1,  1,  1,  1, 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])
```

```
In [51]: x.dtype
```

```
Out[51]: dtype('int64')
```

```
In [52]: x[5] = 3.1415
         x
```

```
Out[52]: array([ 0,  1,  2,  3,  4,  3,  6,  7,  8,  9, 10, 11,  1,  6,  9, 12,  1,
                 1,  1,  1,  1,  1,  1,  1, 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])
```

```
In [53]: x[8] = 1.2j
```

```
        ---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call last)

        <ipython-input-53-a93a954583e2> in <module>()
    ----> 1 x[8] = 1.2j


        TypeError: can't convert complex to int
```

1.2 Copias y vistas de arrays

```
In [54]: x = np.arange(10)
```

```
In [55]: y = x
```

```
In [56]: y.shape = (2,5)
         y
```

```
Out[56]: array([[0, 1, 2, 3, 4],
                [5, 6, 7, 8, 9]])
```

```
In [57]: x
```

```
Out[57]: array([[0, 1, 2, 3, 4],
                [5, 6, 7, 8, 9]])
```

```
In [58]: z = x.copy()
         z
```

```
Out[58]: array([[0, 1, 2, 3, 4],
                [5, 6, 7, 8, 9]])
```

```
In [59]: z is x

Out[59]: False

In [60]: y is x

Out[60]: True

In [61]: x

Out[61]: array([[0, 1, 2, 3, 4],
                [5, 6, 7, 8, 9]])

In [62]: t = x.view()
         t

Out[62]: array([[0, 1, 2, 3, 4],
                [5, 6, 7, 8, 9]])

In [63]: t.shape = (5,2)
         t

Out[63]: array([[0, 1],
                [2, 3],
                [4, 5],
                [6, 7],
                [8, 9]])

In [64]: x

Out[64]: array([[0, 1, 2, 3, 4],
                [5, 6, 7, 8, 9]])

In [65]: s = x[0:2, 1:3]
         s

Out[65]: array([[1, 2],
                [6, 7]])

In [66]: s[:] = 5
         s

Out[66]: array([[5, 5],
                [5, 5]])

In [67]: x

Out[67]: array([[0, 5, 5, 3, 4],
                [5, 5, 5, 8, 9]])

In [68]: y

Out[68]: array([[0, 5, 5, 3, 4],
                [5, 5, 5, 8, 9]])

In [69]: z

Out[69]: array([[0, 1, 2, 3, 4],
                [5, 6, 7, 8, 9]])
```

2 Funciones Univeresales (ufunc)

- Unarias: sqrt, sin, cos, **2
- Binarias: maximum, minimum

```
In [70]: x = np.arange(10)
         x
```

```
Out[70]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
In [71]: x+3
```

```
Out[71]: array([ 3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
```

```
In [72]: x-3
```

```
Out[72]: array([-3, -2, -1,  0,  1,  2,  3,  4,  5,  6])
```

```
In [73]: x*3
```

```
Out[73]: array([ 0,  3,  6,  9, 12, 15, 18, 21, 24, 27])
```

```
In [74]: x/3
```

```
Out[74]: array([0.        , 0.33333333, 0.66666667, 1.        , 1.33333333,
                1.66666667, 2.        , 2.33333333, 2.66666667, 3.        ])
```

```
In [75]: alpha = np.linspace(0,2*np.pi, 4)
         alpha
```

```
Out[75]: array([0.        , 2.0943951 , 4.1887902 , 6.28318531])
```

```
In [76]: np.sin(alpha)
```

```
Out[76]: array([ 0.00000000e+00,  8.66025404e-01, -8.66025404e-01, -2.44929360e-16])
```

```
In [77]: np.cos(alpha)
```

```
Out[77]: array([ 1. , -0.5, -0.5,  1. ])
```

```
In [78]: np.tan(alpha)
```

```
Out[78]: array([ 0.00000000e+00, -1.73205081e+00,  1.73205081e+00, -2.44929360e-16])
```

```
In [79]: np.exp(x)
```

```
Out[79]: array([1.00000000e+00, 2.71828183e+00, 7.38905610e+00, 2.00855369e+01,
                5.45981500e+01, 1.48413159e+02, 4.03428793e+02, 1.09663316e+03,
                2.98095799e+03, 8.10308393e+03])
```

```
In [80]: np.exp2(x)
```

```
Out[80]: array([  1.,   2.,   4.,   8.,  16.,  32.,  64., 128., 256., 512.])
```

```
In [81]: np.power(3,x)
```

```
Out[81]: array([    1,     3,     9,    27,    81,   243,   729,  2187,  6561,
                19683])
```

```
In [82]: np.power(x, 2)
```

```
Out[82]: array([ 0,  1,  4,  9, 16, 25, 36, 49, 64, 81])
```

```
In [83]: np.log(x)
```

```
/anaconda3/lib/python3.5/site-packages/ipykernel_launcher.py:1: RuntimeWarning: divide by zero
  """Entry point for launching an IPython kernel.
```

```
Out[83]: array([      -inf, 0.        , 0.69314718, 1.09861229, 1.38629436,
                1.60943791, 1.79175947, 1.94591015, 2.07944154, 2.19722458])
```

```
In [84]: np.log2(x)
```

```
/anaconda3/lib/python3.5/site-packages/ipykernel_launcher.py:1: RuntimeWarning: divide by zero
  """Entry point for launching an IPython kernel.
```

```
Out[84]: array([      -inf, 0.        , 1.        , 1.5849625 , 2.        ,
                2.32192809, 2.5849625 , 2.80735492, 3.        , 3.169925  ])
```

```
In [85]: np.log10(x)
```

```
/anaconda3/lib/python3.5/site-packages/ipykernel_launcher.py:1: RuntimeWarning: divide by zero
  """Entry point for launching an IPython kernel.
```

```
Out[85]: array([      -inf, 0.        , 0.30103   , 0.47712125, 0.60205999,
                0.69897   , 0.77815125, 0.84509804, 0.90308999, 0.95424251])
```

$$\sum_{i=1}^{n} x_i$$

```
In [86]: np.sum(x)
```

```
Out[86]: 45
```

```
In [87]: np.nansum(x)
```

```
Out[87]: 45
```

```
In [88]: np.prod(x)
```

```
Out[88]: 0

In [89]: np.mean(x)

Out[89]: 4.5

In [90]: np.median(x)

Out[90]: 4.5

In [91]: np.min(x)

Out[91]: 0

In [92]: np.max(x)

Out[92]: 9

In [93]: np.std(x)

Out[93]: 2.8722813232690143

In [94]: np.var(x)

Out[94]: 8.25

In [95]: np.argmin(x)

Out[95]: 0

In [96]: np.argmax(x)

Out[96]: 9

In [99]: np.percentile(x, q=0.95)

Out[99]: 0.08549999999999999

In [105]:  y = np.array([True, True, True, True])

In [106]: np.any(y)

Out[106]: True

In [107]: np.all(y)

Out[107]: True

In [116]: np.random.seed(2019)
          z = np.random.random((3,5))
          z
```

```
Out[116]: array([[0.90348221, 0.39308051, 0.62396996, 0.6378774 , 0.88049907],
                 [0.29917202, 0.70219827, 0.90320616, 0.88138193, 0.4057498 ],
                 [0.45244662, 0.26707032, 0.16286487, 0.8892147 , 0.14847623]])
```

```
In [117]: z.sum()
```

```
Out[117]: 8.550690063213514
```

```
In [118]: z.sum(axis=0)
```

```
Out[118]: array([1.65510085, 1.3623491 , 1.69004099, 2.40847402, 1.43472509])
```

```
In [119]: z.sum(axis=1)
```

```
Out[119]: array([3.43890915, 3.19170818, 1.92007274])
```

```
In [ ]:
```