NumPy

Téllez Gerardo Rubén

1/5/2021

Uso de array

Requiere usar datos homogeneos Amplia los modulos de python

Importar

```
import numpy as np
```

Puede crear objetos a partir de listas

```
L1 = [1, 2, 3, 4, 5, 6, 7, 8, 9]

x1 = np.array(L1)
print(x1)
```

```
## [1 2 3 4 5 6 7 8 9]
```

Tipos de datos

Especificar con *dtype = 'tipo'*

tipos

'float32': Permite el generar la lista con valor decimal

```
x2 = np.array(L1, dtype = 'float32')
print(x2)
```

```
## [1. 2. 3. 4. 5. 6. 7. 8. 9.]
```

'bool': Listas de valores booleanos

```
L2 = []
for n in L1:
  if n > 5:
      L2.append(True)
  else:
```

```
      L2.append(False)

x3 = np.array(L2, dtype = 'bool')
print(x3)
```

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

'int': Lista de enteros variantes: 'intc', 'intp', 'int8', 'int16', 'int32', 'int64'

```
x4 = np.array(L1, dtype='int')
print(x4)
```

```
## [1 2 3 4 5 6 7 8 9]
```

'uint': Enteros sin signo variantes: 'uint', 'int8', 'uint16', 'uint32', 'uint64'

```
x5 = np.array(L1, dtype='uint')
print(L1)
```

```
## [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

'float': De punto flotante variantes: 'float16/32/64" = half/omision/double

```
x6 = np.array(L1, dtype='float')
print(x6)
```

```
## [1. 2. 3. 4. 5. 6. 7. 8. 9.]
```

complex: Números complejos La mitd va para la real y la mitad para la imaginaria variantes: complex64/128

```
x7 = np.array([3+4j, 8+1j], dtype='complex128')
print(x7)
```

```
## [3.+4.j 8.+1.j]
```

Matrices

Matrices de ceros

```
filas = 3
columnas = 4
n0 = np.zeros((filas, columnas))
print(n0)
```

```
## [[0. 0. 0. 0.]
##  [0. 0. 0. 0.]
##  [0. 0. 0. 0.]]
```

Matrices de unos

```
n1 = np.ones((filas, columnas))
print(n1)
```

```
## [[1. 1. 1. 1.]
##  [1. 1. 1. 1.]
##  [1. 1. 1. 1.]]
```

Reordenar matrices constantes

```
print(n0.reshape(4, 3))
```

```
## [[0. 0. 0.]
##  [0. 0. 0.]
##  [0. 0. 0.]
##  [0. 0. 0.]]
```

Generar matrices a partir de repetir array

```
m1 = np.array([L1, L1, L1])
print(m1)
```

```
## [[1 2 3 4 5 6 7 8 9]
##  [1 2 3 4 5 6 7 8 9]
##  [1 2 3 4 5 6 7 8 9]]
```

Aplanar un array, osea hacerlo una lista

```
print(np.ravel(m1))
#Similar a np.flatten
```

```
## [1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9]
```

Cambiar el orden de filas y columnas

```
print(np.transpose(m1))
```

```
## [[1 1 1]
##  [2 2 2]
##  [3 3 3]
##  [4 4 4]
##  [5 5 5]
##  [6 6 6]
##  [7 7 7]
##  [8 8 8]
##  [9 9 9]]
```

Reescala las matrices, pero empieza a llenar espacios vacíos de ser necesario

```
print(np.resize(m1, (6, 12)))
```

```
## [[1 2 3 4 5 6 7 8 9 1 2 3]
##  [4 5 6 7 8 9 1 2 3 4 5 6]
##  [7 8 9 1 2 3 4 5 6 7 8 9]
##  [1 2 3 4 5 6 7 8 9 1 2 3]
##  [4 5 6 7 8 9 1 2 3 4 5 6]
##  [7 8 9 1 2 3 4 5 6 7 8 9]]
```

Rangos

Listas de rangos Similar a seq()

```
r = np.arange(10, 101, 5, dtype='float')
print(r)
```

```
## [ 10.  15.  20.  25.  30.  35.  40.  45.  50.  55.  60.  65.  70.  75.
##   80.  85.  90.  95. 100.]
```

np.linspace: crea array con inicio y fin, de longitud específica. Similar a seq(x, y, length=z)

```
rl = np.linspace(0, 5, num=10)
print(rl)
```

```
## [0.         0.55555556 1.11111111 1.66666667 2.22222222 2.77777778
##  3.33333333 3.88888889 4.44444444 5.        ]
```

Ejercicios

Crear un array de datos con los calores entre 5 y 120

```
a = np.arange(5, 120, dtype='int')
print(a)
```

```
## [  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]
```

Crear una matriz 4*4 con los valores desde 0 hasta 15

```
La = np.arange(0, 3)
Lb = np.arange(4, 7)
Lc = np.arange(8, 11)
Ld = np.arange(12, 15)
Ma = np.array([La, Lb, Lc, Ld])
print(Ma)
```

```
## [[ 0  1  2]
##  [ 4  5  6]
##  [ 8  9 10]
##  [12 13 14]]
```

Crear la matríz identidad 7 * 7

```
Mi = np.identity(7)
print(Mi)
```

```
## [[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.]]
```

Crear un array de 20 elementos y transformarlos en 5*5 matriz

```
Lx = np.arange(1, 21)
Ma = np.array( [Lx[0:5], Lx[5:10], Lx[10:15], Lx[15:20], [0,0,0,0,0] ])
print(Ma)
```

```
## [[ 1  2  3  4  5]
##  [ 6  7  8  9 10]
##  [11 12 13 14 15]
##  [16 17 18 19 20]
##  [ 0  0  0  0  0]]
```

Crear un array con 20 números con los valores entre 0 y 5, espaciados de forma uniforme

```
u = np.linspace(0,5, num=20)
print(u)
```

```
## [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.        ]
```