Algorítmica y Programación

Enero - Mayo 2020

Dr. Iván S. Razo Zapata
(ivan.razo@itam.mx)

ITAM

Numpy

Introducción

Numpy

- Numerical Python
- Computo científico y análisis de datos
- Características relevantes
  - Ndarray (built-in data type)
    - Arreglo/vector multi-dimensional
  - Operaciones matemáticas avanzadas
    - P.ej., algebra lineal, transformada de fourier

ndarray – creación de arreglos

Nuevo código en **editor**

```python

import numpy as np

# Creando arreglo introduciendo datos directamente
arreglo1 = np.array([6, 7, 8, 9, 10])

# Creando arreglo a partir de una lista
lista1 = [1, 2, 3, 4, 5]
arreglo2 = np.array(lista1)

# Creando arreglo con numeros aleatorios
arreglo3 = np.array(np.random.rand(1,5))

# Creando arreglo con numeros aleatorios y tres dimensiones
arreglo4 = np.array(np.random.rand(3,5))
```

Ejecutar código

ndarray – creación de arreglos

Terminal de **IPython**

```
In [13]: arreglo1
Out[13]: array([ 6,  7,  8,  9, 10])

In [14]: arreglo2
Out[14]: array([1, 2, 3, 4, 5])

In [15]: arreglo3
    ...:
Out[15]: array([[0.30751852, 0.52015935, 0.57811619, 0.39254117, 0.87551713]])

In [16]: arreglo4
Out[16]:
array([[0.70225365, 0.89518874, 0.75407839, 0.81743783, 0.15803613],
       [0.38783419, 0.5383055 , 0.99547266, 0.66423277, 0.8497358 ],
       [0.12045907, 0.44381888, 0.31651033, 0.61306665, 0.93390252]])
```

ndarray – creación de arreglos

```
In [17]:

In [18]: arreglo1.dtype
Out[18]: dtype('int32')

In [19]: arreglo2.dtype
Out[19]: dtype('int32')

In [20]: arreglo3.dtype
Out[20]: dtype('float64')

In [21]: arreglo4.dtype
Out[21]: dtype('float64')
```

ndarray – creación de arreglos

Editor

```
7
8 import numpy as np
9
10 # Creando arreglo con ceros
11 arreglo5 = np.array(np.zeros(5))
12
13 # Creando arreglo con valores en un rango
14 arreglo6 = np.array(np.arange(0,10,2))
15
16
17
```

Terminal de **IPython**

```
In [27]: arreglo5
Out[27]: array([0., 0., 0., 0., 0.])

In [28]: arreglo6
Out[28]: array([0, 2, 4, 6, 8])

In [29]:
```

ndarray – operaciones con escalares y arreglos

Broadcasting

```
In [38]: arreglo1
Out[38]: array([ 6,  7,  8,  9, 10])

In [39]: arreglo1 * 25
Out[39]: array([150, 175, 200, 225, 250])

In [40]: arreglo2
Out[40]: array([1, 2, 3, 4, 5])

In [41]: arreglo2 / 2
Out[41]: array([0.5, 1. , 1.5, 2. , 2.5])

In [42]: arreglo1 * arreglo2
Out[42]: array([ 6, 14, 24, 36, 50])

In [43]: arreglo2 / arreglo1
Out[43]: array([0.16666667, 0.28571429, 0.375     , 0.44444444, 0.5       ])
```

ndarray – operaciones con escalares y arreglos

Diferentes resultados con listas

```
In [55]: lista1
Out[55]: [1, 2, 3, 4, 5]

In [56]: lista1 * 2
Out[56]: [1, 2, 3, 4, 5, 1, 2, 3, 4, 5]
```

ndarray – operaciones con escalares y arreglos

```
In [45]: arreglo4
Out[45]:
array([[0.70225365, 0.89518874, 0.75407839, 0.81743783, 0.15803613],
       [0.38783419, 0.5383055 , 0.99547266, 0.66423277, 0.8497358 ],
       [0.12045907, 0.44381888, 0.31651033, 0.61306665, 0.93390252]])

In [46]: arreglo4 * 5.25
Out[46]:
array([[3.68683164, 4.69974089, 3.95891157, 4.29154862, 0.82968966],
       [2.0361295 , 2.82610388, 5.22623147, 3.48722202, 4.46111296],
       [0.6324101 , 2.33004913, 1.66167924, 3.21859991, 4.90298822]])

In [47]: arreglo4 * arreglo1
Out[47]:
array([[4.21352188, 6.26632119, 6.03262716, 7.35694048, 1.58036125],
       [2.32700514, 3.7681385 , 7.96378129, 5.9780949 , 8.49735802],
       [0.7227544 , 3.10673217, 2.53208265, 5.51759985, 9.33902518]])
```

ndarray – operaciones con escalares y arreglos

```
In [48]: arreglo7 = np.array(np.random.rand(2,4))

In [49]: arreglo7
Out[49]:
array([[0.93976231, 0.05861595, 0.68583992, 0.81741521],
       [0.77383874, 0.13602761, 0.35577954, 0.68985691]])

In [50]: arreglo4 * arreglo7
```

ndarray – operaciones con escalares y arreglos

```
In [48]: arreglo7 = np.array(np.random.rand(2,4))

In [49]: arreglo7
Out[49]:
array([[0.93976231, 0.05861595, 0.68583992, 0.81741521],
       [0.77383874, 0.13602761, 0.35577954, 0.68985691]])

In [50]: arreglo4 * arreglo7
Traceback (most recent call last):

  File "<ipython-input-50-a93f2afd4eca>", line 1, in <module>
    arreglo4 * arreglo7

ValueError: operands could not be broadcast together with shapes (3,5) (2,4)
```

ndarray – operaciones con escalares y arreglos

Multiplicación de matrices (arreglos bidimensionales)

```
In [66]: arreglo4.dot(arreglo7)
Traceback (most recent call last):

  File "<ipython-input-66-682f43723dbf>", line 1, in <module>
    arreglo4.dot(arreglo7)

ValueError: shapes (3,5) and (2,4) not aligned: 5 (dim 1) != 2 (dim 0)
```

```
In [67]:

In [67]: arreglo8 = np.array(np.random.rand(5,2))

In [68]: arreglo8
Out[68]:
array([[0.73212953, 0.88825618],
       [0.14838998, 0.74925624],
       [0.45093798, 0.79275557],
       [0.41390713, 0.70645349],
       [0.66979552, 0.25045549]])

In [69]: arreglo4.dot(arreglo8)
Out[69]:
array([[1.4312155 , 2.50936956],
       [1.65680035, 2.21906192],
       [1.17605282, 1.35745199]])

In [70]: |
```

Arreglos bidimensionales (matrices)

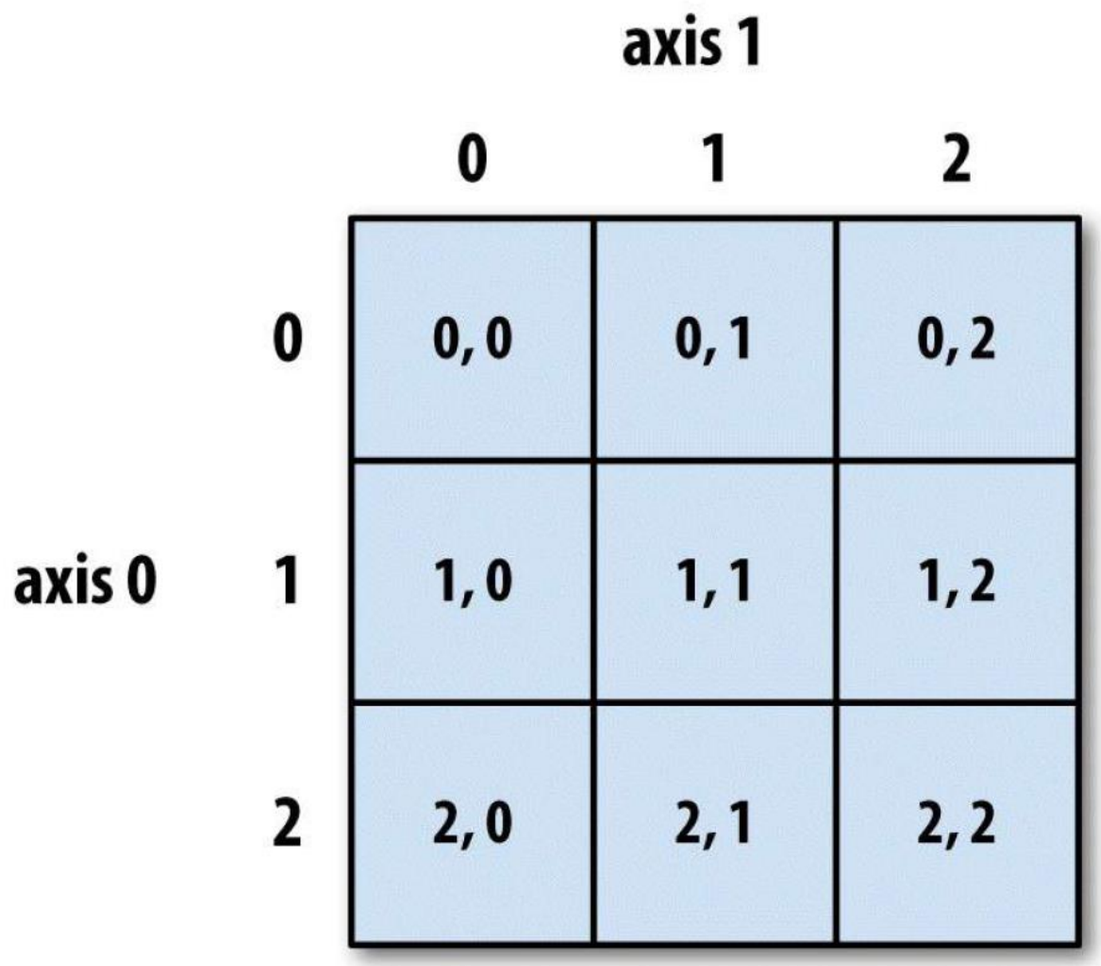
```python
import numpy as np

# Creando arreglo 2D con zeros
arreglo2D_1 = np.zeros((3,3))
arreglo2D_2 = np.array(np.zeros((3,3)))


# Creando arreglo 2D con valores aleatorios
arreglo2D_3 = np.array(np.random.rand(3,5))
arreglo2D_4 = np.random.rand(3,5)
```

```
In [81]: arreglo2D_1
Out[81]:
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])

In [82]: arreglo2D_2
Out[82]:
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])

In [83]: arreglo2D_3
Out[83]:
array([[0.86002484, 0.72267042, 0.68832863, 0.74688985, 0.00540337],
       [0.57634786, 0.67612725, 0.97589886, 0.02346288, 0.12413947],
       [0.41026754, 0.34467031, 0.35732899, 0.34797135, 0.53211073]])

In [84]: arreglo2D_4
Out[84]:
array([[0.4812903 , 0.55013223, 0.10819807, 0.31470377, 0.44045086],
       [0.64031164, 0.22416327, 0.12135196, 0.37688389, 0.66804157],
       [0.64779095, 0.38711557, 0.99141894, 0.22336247, 0.09163455]])
```

Arreglos bidimensionales (matrices)

Accediendo al contenido

Columna



Renglón

Arreglos bidimensionales (matrices)

Accediendo al contenido

```
In [84]: arreglo2D_4
Out[84]:
array([[0.4812903 , 0.55013223, 0.10819807, 0.31470377, 0.44045086],
       [0.64031164, 0.22416327, 0.12135196, 0.37688389, 0.66804157],
       [0.64779095, 0.38711557, 0.99141894, 0.22336247, 0.09163455]])

In [85]: arreglo2D_4[1][2]
Out[85]: 0.12135195853418468

In [86]: arreglo2D_4[1][5]
Traceback (most recent call last):

  File "<ipython-input-86-502efeb61adc>", line 1, in <module>
    arreglo2D_4[1][5]

IndexError: index 5 is out of bounds for axis 0 with size 5
```

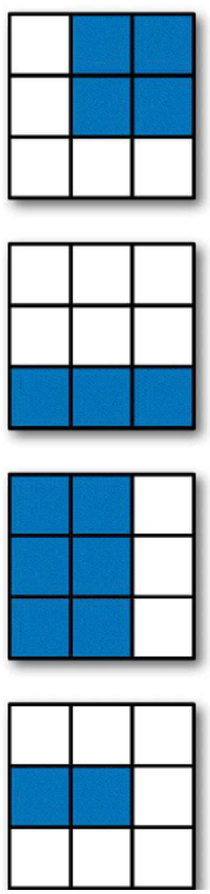
Arreglos bidimensionales (matrices)

Accediendo al contenido

```
In [87]: arreglo2D_4[1,2]
Out[87]: 0.12135195853418468

In [88]: arreglo2D_4[1,:]
Out[88]: array([0.64031164, 0.22416327, 0.12135196, 0.37688389, 0.66804157])

In [89]: arreglo2D_4[:,2]
Out[89]: array([0.10819807, 0.12135196, 0.99141894])

In [90]:
```

Arreglos bidimensionales (matrices)

Accediendo al contenido



Expression	Shape
arr[:2, 1:]	(2, 2)
arr[2]	(3,)
arr[2, :]	(3,)
arr[2:, :]	(1, 3)
arr[:, :2]	(3, 2)
arr[1, :2]	(2,)
arr[1:2, :2]	(1, 2)

```
In [126]: arreglo2D_5 = np.random.rand(3,3)

In [127]: arreglo2D_5[0] = np.arange(1,4)

In [128]: arreglo2D_5[1] = np.arange(4,7)

In [129]: arreglo2D_5[2] = np.arange(7,10)

In [130]: arreglo2D_5
Out[130]:
array([[1., 2., 3.],
       [4., 5., 6.],
       [7., 8., 9.]])

In [131]:
```

Cambiando el contenido de un renglón

Arreglos bidimensionales (matrices)

Filtrando contenido

```
In [156]: arreglo2D_5[arreglo2D_5 > 2]
Out[156]: array([3., 4., 5., 6., 7., 8., 9.])

In [157]: arreglo2D_5[arreglo2D_5 % 2 == 0]
Out[157]: array([2., 4., 6., 8.])

In [158]: arreglo2D_5[arreglo2D_5 % 2 != 0]
Out[158]: array([1., 3., 5., 7., 9.])

In [159]:
```

Conservando impares

```
In [164]: arreglo2D_5[arreglo2D_5 % 2 == 0] = 0

In [165]: arreglo2D_5
Out[165]:
array([[1., 0., 3.],
       [0., 5., 0.],
       [7., 0., 9.]])
```

ITAM

Procesamiento básico de datos con arreglos

```python
import numpy as np

arreglo1 = np.random.randn(1,5)

arreglo2 = np.random.rand(1,5)

arreglo3 = np.random.randn(3,5)

arreglo4 = np.random.rand(3,5)
```

```
In [9]: arreglo1.mean()
Out[9]: -0.09578874096773601

In [10]: arreglo2.mean()
Out[10]: 0.5487579838737476

In [11]: arreglo3.mean()
Out[11]: -0.35181296464060857

In [12]: arreglo4.mean()
Out[12]: 0.3825316629779574

In [13]: arreglo1
Out[13]: array([[-0.05866434, -1.2970198 ,  1.13533382, -0.29999122,  0.04139784]])

In [14]: arreglo2
Out[14]: array([[0.05107957, 0.74463768, 0.4986265 , 0.6223728 , 0.82707337]])

In [15]: arreglo3
Out[15]:
array([[-0.84212715,  0.38829324, -0.68547111,  0.62645273, -0.96299672],
       [ 0.89442269, -1.07228604,  0.5167779 , -0.15226598,  0.12015435],
       [-0.39385749, -0.09238875, -0.7732848 , -1.35540106, -1.49321627]])

In [16]: arreglo4
Out[16]:
array([[0.87955394, 0.07443135, 0.16218781, 0.25921036, 0.27557284],
       [0.32887577, 0.36378571, 0.28820946, 0.99823064, 0.33746803],
       [0.13936676, 0.1796533 , 0.28543321, 0.268742  , 0.89725378]])
```

ITAM

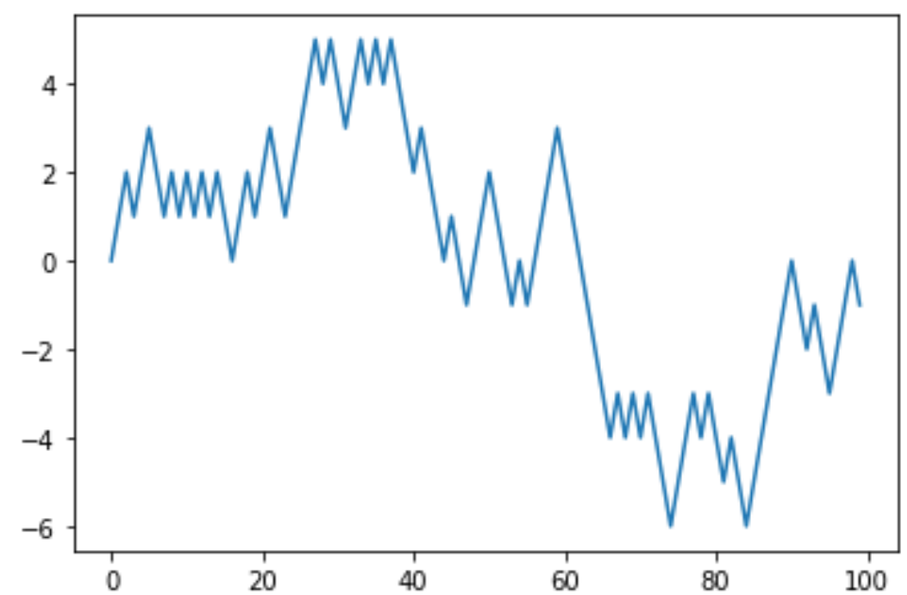
Procesamiento básico de datos con arreglos

Method	Description
`sum`	Sum of all the elements in the array or along an axis; zero-length arrays have sum 0
`mean`	Arithmetic mean; zero-length arrays have `NaN` mean
`std, var`	Standard deviation and variance, respectively, with optional degrees of freedom adjustment (default denominator `n`)
`min, max`	Minimum and maximum
`argmin, argmax`	Indices of minimum and maximum elements, respectively
`cumsum`	Cumulative sum of elements starting from 0
`cumprod`	Cumulative product of elements starting from 1

ITAM

Procesamiento básico de datos con arreglos

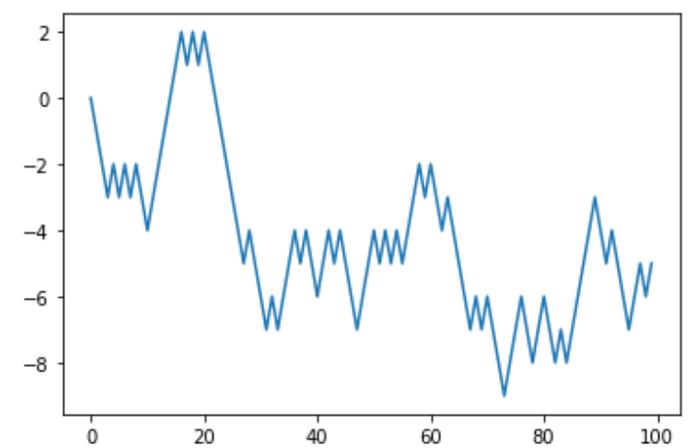
```python
import random
import numpy as np
import matplotlib.pyplot as plt

posicion = 0
listaTmp = [posicion]
pasos = 1000

for i in range(pasos):
    paso = 1 if random.randint(0, 1) else -1
    posicion += paso
    listaTmp.append(posicion)

caminata = np.array(listaTmp)
plt.plot(caminata[:100])

```



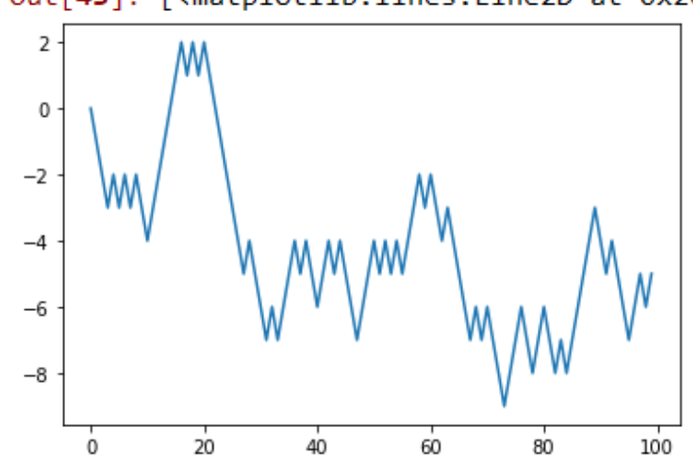
Guardar y leer datos de archivos – Ejemplos básicos

```python

import random
import numpy as np
import matplotlib.pyplot as plt

posicion = 0
listaTmp = [posicion]
pasos = 1000

for i in range(pasos):
    paso = 1 if random.randint(0, 1) else -1
    posicion += paso
    listaTmp.append(posicion)

caminata = np.array(listaTmp)
plt.plot(caminata[:100])

np.savetxt("Resultados.txt", caminata, delimiter=',')

nuevoArray = np.loadtxt("Resultados.txt", delimiter=',')

```

```
In [42]: runfile('C:/Users/irazoz/Documents/ITAM/Cursos
ITAM/Cursos/2020/AyP-2020-Enero-Mayo/Ejemplos')
```




```
In [43]: plt.plot(nuevoArray[:100])
Out[43]: [<matplotlib.lines.Line2D at 0x26051dc5b00>]
```




```
In [44]:
```