# Arrays

Andrew Reeve
School of Earth and Climate Sciences
University of Maine

# Python Numérico Python (numpy)

- introduce estructura nueva: el 'array'
  - también hay un estructura 'array' en python básico
    - no incluye las matemáticas
  - estructura rejilla multi-dimensional
  - almacena un tipo de datos
- más rápido matemáticas
  - funciones de álgebra lineal

## ¿Quien usa las matrices? ¡Usted las usa!

- Imagines, Video Juegos
- Modelos (Tiempo, Flujo)
- GPS
- Radar, Sonar
- comunicación (teléfono)

## Digresión: Que es un matriz?

- almacena los coeficientes de ecuaciones simultaneas
- presentación más compacto
- bueno para la contabilización
- almacena los tensores (propiedades direccional)

## Las Vectores

- un columna o fila
- dirección y magnitud
- e.g., fuerza, velocidad, flujo

# Indexando y Rebanado Matrices

- Booleanas usan símbolos diferentes
  - and : &
  - or : |
  - not : ~
- diferente formato para cortar y rebanar
  - indexando, lista de coordinadas separado con comas
  - usa una lista de indexes para seleccionar valores
  - puede usar booleanas para seleccionar valores
  - las columnas se indican en la posición a la derecha
    - A[col] o A[row,col] o A[layer,row,col] etc.
    - rebanar matrices es similar a las listas: A[inició, fin, paso]
    - si inicio, fin, o paso está blanco significa primero, ultimo, o uno

```
# typical way numpy imported
import numpy as np
# create and array object
A = np.array([[1,2],[3,4]],'float')
print( A, ' :A')
print(A[0,1],' :A[0,1]') # row,col
print(A[:,0], ' :A[:,0]') # all rows,
      first (0th) column
B=np.array([[1,22],[33,4]],'float')
print(A==B, ' :A==B ')
print(~(A==B),' :~(A==B)')
print((A>1)&(B<10), ' :(A>1)&(B<10)')
print((A>1)|(B<10), ' :(A>1)|(B<10)')
```

Output

```
[[1. 2.]
 [3. 4.]] :A
2.0 :A[0,1]
[1. 3.] :A[:,0]
[[ True False]
 [False True]] :A==B
[[False True]
 [ True False]] :~(A==B)
[[False False]
 [False True]] :(A>1)&(B<10)
[[ True True]
 [ True True]] :(A>1)|(B<10)
```

# Crear Matrices

- Matrices llenada con ceros: `np.zeros([nrow,ncol],dtype)`
- Matrices llenada con unos: `np.ones([nrow,ncol],dtype)`
- Matriz de identidad: `np.identity(nrow)`
- Numpy 'range': `np.arange(start,stop,stride)`
- Cambia una lista a matriz `np.array(a_list)`
- Remodelar una matriz: `np.reshape(Array,(row,col)`
  - no puede cambia el tamaño de una matriz
- Cambiar el tamaño de una matriz: `np.resize(Array,(row, col))`
- Transponer una matriz: `np.transpose(A)`

```
import numpy as np
A=np.ones([2,4],np.float16)
B=np.zeros([3,2],np.int8)
print(A)
print(B)
C=np.identity(3)
D=np.array([[i*j for i in range(1,4)] for j in range(1,3)])
print(C)
print(D)
A=np.reshape(A,[2,2,2])
D=np.transpose(D)
print(A)
print(D)
```

Output

```
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]]
[[0 0]
 [0 0]
 [0 0]]
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
[[1 2 3]
 [2 4 6]]
[[[1. 1.]
  [1. 1.]]

 [[1. 1.]
  [1. 1.]]]
[[1 2]
 [2 4]
 [3 6]]
```

# Metodos Basicos de Matrices

## información sobre la matriz, valores espaciados regularmente

- obtener forma de la matriz: `array.shape`
- obtener el tipo de dato: `array.dtype`
- crear números espaciados regularmente: `np.linspace()`
- crear números con espaciado de log10: `np.logspace()`

## Concatenando

- para combinar las matrices, necesitan tener las mismas tamaños
- `concatenate`, `vstack`, `hstack`, `dstack`
- similar mandas para cortarlas (`split`, `vsplit`, etc.)

## NaN's

- significa no es numero
- entrar a mano: 'np.NaN', se falta un valor
- resultado de:
  - underflows o overflows (NaN or inf)
  - operaciones que no tener sentido (eg. np.sqrt(-2))
- algunas funciones ignoran los valores de NaN
  - np.nansum(array), np.nammean(array)

# Computo con Matrices

- rápido para cálculos
  - operación elemento por elemento
  - funciones universal
- funciones especiales para for aggregadas (sum, std, var, percentile)
- pruebas de booleano (any, all)

```
import numpy as np
A=np.linspace(0,2*np.pi, 6)
B = A*2
B = np.multiply(A,2)
B = np.sin(A)
C = np.add.reduce(A)
D = np.add.outer(A,A)
E = np.add.accumulate(A)
print(f'{A}\n{C}\n{D}\n{E}')
```

Output

```
[0.   1.26 2.51 3.77 5.03 6.28]
18.84955592153876
[[ 0.    1.26  2.51  3.77  5.03  6.28]
 [ 1.26  2.51  3.77  5.03  6.28  7.54]
 [ 2.51  3.77  5.03  6.28  7.54  8.8 ]
 [ 3.77  5.03  6.28  7.54  8.8  10.05]
 [ 5.03  6.28  7.54  8.8  10.05 11.31]
 [ 6.28  7.54  8.8  10.05 11.31 12.57]]
[ 0.    1.26  3.77  7.54 12.57 18.85]
```

# Matemáticas de Matiz

- Producto punto
  - Multiplicativo de matriz
  - Alineación de las matriz (ValueError)
- linalg (biblioteca de álgebra linear)
  - resolver
  - vectores propios (eigenvectors)
  - mínimos cuadrados etc.
- numeros aleatorios
  - de distribuciones
  - elegir de los valores (choice)
  - cambia el orden (shuffle)

```
import numpy as np
v1=np.array([1, 2, 3])
v2=np.array([.2, .4, .8])
A=np.dot(v1,v2) #returns a scalar
v3=v1.reshape([3,1])
v4=v2.reshape([1,3])
B=v3.dot(v4) #both return 2 dimensional mtrx
C=np.dot(v4,v3)

data=np.arange(20)
A=np.random.choice(data,size=9).reshape(3,3)
b=np.random.choice(data,size=3).reshape(3,1)
print(f'{b}\n {A}')
print(np.dot(A,b))
print(np.linalg.solve(A,b))
```

Output

```
[[ 7]
 [13]
 [ 7]]
 [[ 6  1 11]
 [ 9 17 19]
 [13  7 12]]
[[132]
 [417]
 [266]]
[[-0.14]
 [ 0.05]
 [ 0.71]]
```

# Las Booleanas y los indices

- `where`:
  - usa booleana para seleccionar los indices
  - usa booleana para cambiar una parte del matriz
- usa booleanas con las matemáticas
  - `False` es 0, `True` es 1

```
import numpy as np

np.set_printoptions(precision=2)

A = np.random.normal(0, 2, 36).reshape(6, 6)
b = np.arange(6).reshape(1, -1)
C = b * b.transpose() + A

idx = np.where(C > 2)

D = np.where(C > 2, C, 2)

e = np.choose([0, 1, 2, 3, 4, 5], C)

E = (A > 0) * A + (A < 0) * -A
print(E)
```

Output

```
[[2.08e+00 1.88e-03 2.13e+00 1.91e+00 6.06e+00 1.65e-03]
 [6.16e-01 2.92e+00 1.35e+00 1.53e-01 1.50e+00 3.20e+00]
 [2.64e+00 1.04e-01 2.48e+00 9.52e-01 1.71e-01 1.79e+00]
 [7.63e-01 1.88e-01 3.90e-01 1.67e-01 2.21e-01 2.52e+00]
 [8.99e-01 1.52e+00 1.50e+00 2.50e+00 1.67e+00 1.45e+00]
 [1.19e+00 2.60e+00 1.17e+00 1.34e+00 2.57e+00 8.74e-01]]
```

# radiodifusión de matriz (Array broadcasting)

- Se expande los dimensiones de las matrices
- Usa con operaciones de elemento por elemento
- Para upcast a dimensiones basado en las matices que aporte
- Usa matrices que no son alineados
- Almohadillas la dimensión más baja/pequeña

```
import numpy as np

A=np.arange(4).reshape(4,1)
B=np.arange(4).reshape(1,4)

### expanding axis w/ dim = 1
print(A+B)

A=np.arange(8).reshape(4,1,2) #layer, row, col
B=np.arange(4).reshape(4,1) #row, col
### pads missing dimension, based on matching dimensions
print(A+B)
```

Output

```
[[0 1 2 3]
 [1 2 3 4]
 [2 3 4 5]
 [3 4 5 6]]
[[[ 0  1]
  [ 1  2]
  [ 2  3]
  [ 3  4]]

 [[ 2  3]
  [ 3  4]
  [ 4  5]
  [ 5  6]]

 [[ 4  5]
  [ 5  6]
  [ 6  7]
  [ 7  8]]
```

# Otras estructuras de Numpy

## Matrices enmascaramiento (Masked Arrays)

- Para esconder una parte de matriz
- contiene los datos, una matriz booleana, y valor de relleno (fill value)
- biblioteca de `numpy.ma`

## matrices de registros (record array)

- matriz con columnas y filas con nombres
- permite una mezcla de tipos de datos

```
import numpy as np
A = np.array([("a",1,2,3),("b",4,5,6),("c",7,8,9)],
        dtype=[('site','U4'), ('Ca',float), ('c','i4'), ('pulpo',
      'f8')],)
print(A['Ca'])
print(A[2])

B=[[col+row  for col in range(4)] for row in range(4)]
B=np.array(B)
B_bool=B<2
B_ma=np.ma.masked_array(B,B_bool,fill_value=-9999)
print(B_ma)
```

Output

```
[1. 4. 7.]
('c', 7., 8, 9.)
[[-- -- 2 3]
 [-- 2 3 4]
 [2 3 4 5]
 [3 4 5 6]]
```

# Otros Bibliotecas externas de matemáticas

- scipy: lots of computational science modules
  - interpolation
  - statistics
  - numerical integration
  - special functions (e.g. Bessel, exponential, error functions)
  - linear algebra functions
- sympy: symbolic algebra with python
- statsmodel: statistical tests in python
- scikit-learn: machine learning algorithms with python
- and many,many others.