Numerical Python

modeling, state

CS101 Lecture #16

Administrivia

Administrivia 1/22

Administrivia

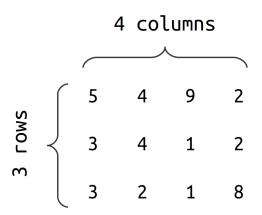
- ▶ Homework #7 is due Friday, Nov. 25.
- ▶ Homework #8 is due Friday, Dec. 2.

Administrivia 2/22

More numpy

More numpy 3/22

Indexing arrays



▶ Reminder: numpy indexes by array[row][col].

More numpy 4/22

Question

$$x = \begin{pmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \end{pmatrix}$$

What will produce this array?

A np.array([[1,2,3],[1,2,3]])

B np.array([2,3])

C np.array([3,2])

D np.array([[1,1],[2,2],[3,3]])

More numpy 5/22

Question

$$x = \begin{pmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \end{pmatrix}$$

What will produce this array?

A np.array([[1,2,3],[1,2,3]])

B np.array([2,3])

C np.array([3,2])

D np.array([[1,1],[2,2],[3,3]])

*

More numpy 6/22

Data types

- numpy supports many possible data types:
 - bool
 - int16, int32
 - float16, float32, float64
 - complex64, complex128
- For the most part, stick with bool, int32, and float64 (most accurate).
- Specify (and query) with dtype:

```
import numpy as np
a = [ 3,2,4 ]
x = np.array( a,dtype=np.float64 )
x.dtype
```

More numpy 7/22

Other arrays

```
x = np.zeros([2,3]) # zeroes
y = np.ones([4,1]) # ones
```

Produce arrays of zeros or ones with specified dimensions.

More numpy 8/22

Other arrays

```
z = np.eye(5) # identity
```

 Produces identity matrix of specified square dimension.

More numpy 9/22

Other arrays

```
w = np.linspace( 0,10,101 )
v = np.linspace( start, finish, n)
```

- Produce arrays from start to finish of n points (not spacing!).
- Excellent for grids and coordinates.
- May also see arange: [start, stop), but I recommend avoiding its use:

```
u = np.arange( 0,10,0.1 ) # tricky!
u == array( [ 0, 0.1, 0.2, ..., 9.9 ] )
```

More numpy 10/22

The punchline: Why?

Plot sin(x) for $x \in [0, 2\pi]$ using pure Python.

```
import matplotlib.pyplot as plt
%matplotlib inline
from math import pi
x = [] # can't use range!
for i in range(100):
    x.append( 2*pi*i/100 )
from math import sin
y = []
for j in range(100):
    y.append( sin(x[j]) )
plt.plot( x,y,'k-' )
plt.xlim(0,2*pi)
plt.ylim(-1,1)
plt.show()
```

More numpy 11/22

The punchline: Why?

```
Plot sin(x) for x \in [0, 2\pi] using numpy.
import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
x = np.linspace(0,2*np.pi,101)
y = np.sin(x)
plt.plot( x,y,'k-' )
plt.xlim(0,2*pi)
plt.ylim(-1,1)
plt.show()
See options for plot at http:
//stackoverflow.com/questions/8376926/
plotting-many-graphs-with-matplotlib
```

More numpy 12/2

Modeling 13/22

Consider a cup falling from the edge of a table. Describe its path and time until it hits the ground. Two approaches:

- Use analytical equation (if available).
- Use finite difference equation otherwise.

Modeling 14/22

Use analytical equation (if available).

$$y(t) = y_0 + v_0 t + \frac{a}{2}t^2$$

$$y_0 = 1$$

$$v_0 = 0$$

$$a = -9.8$$

$$y(t) > 0$$

subject to

$$y(t) \ge 0$$

15/22 Modeling

import numpy as np

```
# Parameters of simulation
n = 100 # number of data points to plot
start = 0.0 # start time, s
end = 1.0 # ending time, s
a = -9.8 # acceleration, m*s**-2
# State variable initialization
t = np.linspace(start,end,n+1) # time, s
v = 1.0 + a/2 * t**2
for i in range(1,n+1):
    if y[i] <= 0: # glass has hit the ground
       y[i] = 0
```

Modeling 16/22

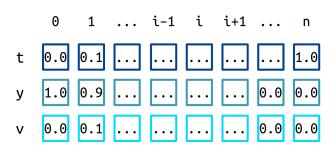
Use finite difference equation otherwise.

$$\frac{dy}{dt} = v(t) \approx \frac{y^{n+1} - y^n}{t^{n+1} - t^n} \to y^{n+1} = y^n + v(t^{n+1} - t^n)$$

$$\frac{dv}{dt} = a \approx \frac{v^{n+1} - v^n}{t^{n+1} - t^n} \to v^{n+1} = v^n + a(t^{n+1} - t^n)$$

$$v^{n=0} = 0 \qquad y^{n=0} = 1 \qquad a = -9.8$$
subject to
$$y(t) \ge 0$$

Modeling 17/2:



Modeling 18/22

import numpy as np

```
# Parameters of simulation
n = 100 # number of data points to plot
start = 0.0 # start time, s
end = 1.0 # ending time, s
a = -9.8 # acceleration, m*s**-2
# State variable initialization
t = np.linspace(start,end,n+1) # time, s
y = np.zeros(n+1)
                              # height, m
v = np.zeros(n+1)
                              # velocity, m*s**-1
v[0] = 1.0
                              # initial condition, m
for i in range(1,n+1):
   v[i] = v[i-1] + a*(t[i]-t[i-1])
   y[i] = y[i-1] + v[i-1] * (t[i]-t[i-1])
    if y[i] <= 0: # glass has hit the ground
       v[i] = 0
       y[i] = 0
```

Modeling

- ▶ How would you make the cup bounce?
- ▶ How would you include lateral motion?

Modeling 20/22

Reminders

Reminders 21/22

Reminders

- ▶ Homework #7 is due Friday, Nov. 25.
- ▶ Homework #8 is due Friday, Dec. 2.

Reminders 22/2