Python lab 5: The Internet and 3D visualisation

Dr Ben Dudson

Department of Physics, University of York

25th February 2011

`http://www-users.york.ac.uk/∼bd512/teaching.shtml`

Python libraries

- One of the great strengths of Python is the huge number of libraries which are available for it
- It's used all over the place in many different projects
- This means that many individuals and companies have written and released free libraries to make creating programs in Python quicker and easier
- So far we've seen some of the things Python can do for scientific problems
- This lab we're going to look at just some of the other things Python can be made to do

Networking

- In addition to reading from files, Python makes it easy to read data across a network, or from sites on the Internet

Networking

- In addition to reading from files, Python makes it easy to read data across a network, or from sites on the Internet
- For example, the electronics department has a weather station and makes the data available online

  `http://weather.elec.york.ac.uk/archive.html`

Networking

- In addition to reading from files, Python makes it easy to read data across a network, or from sites on the Internet
- For example, the electronics department has a weather station and makes the data available online

  http://weather.elec.york.ac.uk/archive.html

Opening web addresses can be done in a similar way to files:

```
import urllib2

addr="http://weather.elec.york.ac.uk/data/vaisala/
                                    archives/1110.txt"
f = urllib2.urlopen(addr)
line = f.readline()
f.close()
print line
```

Reading network addresses

Internet address	File
import urllib2	
f = urllib2 .urlopen("http ://... ")	f = open("...", "r")
line = f. readline ()	line = f. readline ()
f . close ()	f . close ()

Reading network addresses

Internet address	File
import urllib2	
f = urllib2.urlopen("http://...")	f = open("...", "r")
line = f.readline()	line = f.readline()
f.close()	f.close()

- To open an address, we can use the urllib2 module, then just use urllib2.urlopen() instead of open()
- To get the data, the same methods can be used
- When we're done, both should be closed

Reading weather data

Reading from

`http://weather.elec.york.ac.uk/data/vaisala/archives/1110.txt`

gives the York weather data from November 2010:

```
         Temp  Temp  Temp  Wind  Wind  Dir   Press.
         Ave   Min   Max   Ave   Max   Ave   Min
Date     (C)   (C)   (C)   (MPH) (MPH) (Deg) (hPa)
01_11_10 10.3  8.9   12.4  9.2   30.6  227   999.1
02_11_10 12.1  10.7  14.3  14.7  38.1  220   992.1
```

Reading weather data

Reading from

`http://weather.elec.york.ac.uk/data/vaisala/archives/1110.txt`

gives the York weather data from November 2010:

```
          Temp  Temp  Temp  Wind  Wind  Dir   Press.
          Ave   Min   Max   Ave   Max   Ave   Min
Date      (C)   (C)   (C)   (MPH) (MPH) (Deg) (hPa)
01_11_10  10.3  8.9   12.4  9.2   30.6  227   999.1
02_11_10  12.1  10.7  14.3  14.7  38.1  220   992.1
```

To turn this data into something useful, we can use the same methods we did last time for files

This lab has tasks on reading and plotting this data

Websites

You can of course also read data from other websites, so

```
import urllib2
f = urllib2.urlopen("http://www.google.com")
line = f.readline()
f.close()
print line
```

Websites

You can of course also read data from other websites, so

```
import urllib2
f = urllib2.urlopen("http://www.google.com")
line = f.readline()
f.close()
print line
```

```
<!doctype html><html><head><meta http-equiv=
"content-type" content="text/html; charset=
ISO-8859-1"><title>Google</title><script>
window.google={kEI:"WLfyTOHWJs-G4gaZOuGDCw",
...
```

This is HTML, which tells browsers how to draw web pages. It's a little more work to get data out of this, but still possible

Websites

You can of course also read data from other websites, so

```
import urllib2
f = urllib2.urlopen("http://www.google.com")
line = f.readline()
f.close()
print line
```

```
<!doctype html><html><head><meta http-equiv=
"content-type" content="text/html; charset=
ISO-8859-1"><title>Google</title><script>
window.google={kEI:"WLfyTOHWJs-G4gaZOuGDCw",
...
```

This is HTML, which tells browsers how to draw web pages. It's a little more work to get data out of this, but still possible
Fortunately there is a lot more to the internet than just websites...

Twitter feeds

- Twitter makes its data available publicly not only through the web, but also in more programmer-friendly ways to allow other programs to access the data

Twitter feeds

- Twitter makes its data available publicly not only through the web, but also in more programmer-friendly ways to allow other programs to access the data
- In particular, it allows us to download tweets in JSON format which has a module for Python. The public timeline data is available from

  http://twitter.com/statuses/public_timeline.json

Twitter feeds

- Twitter makes its data available publicly not only through the web, but also in more programmer-friendly ways to allow other programs to access the data
- In particular, it allows us to download tweets in JSON format which has a module for Python. The public timeline data is available from

  http:// twitter .com/statuses/ public_timeline .json
- If you have an account then you can also access it through Python, but here we'll just look at the publicly available data

Reading the data

Reading data from is made easy with urllib2

```
import urllib2
addr="http://twitter.com/statuses/public_timeline.js
f = urllib2.urlopen(addr)
data = f.read()  # Read all the data
f.close()
```

Reading the data

Reading data from is made easy with urllib2

```
import urllib2
addr="http://twitter.com/statuses/public_timeline.js
f = urllib2.urlopen(addr)
data = f.read()  # Read all the data
f.close()
```

data now contains the tweet information as JSON text:

```
[{"in_reply_to_status_id":null,"truncated":false,"created_a
"Sun Nov 28 16:32:21 +0000 2010","geo":null,"favorited":fal
"source":"web","in_reply_to_status_id_str":null,"id_str":
"892114724454400O","contributors":null,"coordinates":null,
"in_reply_to_screen_name":null,"in_reply_to_user_id_str":
```

To convert this into something more useful, we need to decode (**parse**) it. We could write some code to do this, but luckely someone's already done all this work - the json module

Processing Twitter data

Using the json module to process the data:

```
import json
a = json.loads(data)
```

'a' now contains the data in a more useful form. We can find out what it is using

```
print type(a)
<type 'list'>
```

so 'a' is list of tweets (len(a) = 20): a[0] contains the first, a[1] the second.

Processing Twitter data

Using the json module to process the data:

```
import json
a = json.loads(data)
```

'a' now contains the data in a more useful form. We can find out what it is using

```
print type(a)
<type 'list'>
```

so 'a' is list of tweets (len(a) = 20): a[0] contains the first, a[1] the second.
To find out how each tweet is stored, try getting the type again

```
print type(a[0])
<type 'dict'>
```

This says that a[0] is a 'dict' or dictionary type.

Dictionaries

Dictionaries are like arrays, except rather than using numbers (0,1,2,...) to refer to the individual values they can use text:

```
d = dict([])
d['monday'] = 1
d['tuesday'] = 2
print d['monday']        -> 1
```

Dictionaries

Dictionaries are like arrays, except rather than using numbers (0,1,2,...) to refer to the individual values they can use text:

```
d = dict([])
d['monday'] = 1
d['tuesday'] = 2
print d['monday']       -> 1
```

The names are called **keys**. To see a list of keys, we can use

```
print a[0].keys()
[u'favorited', u'in_reply_to_user_id',
u'contributors', u'truncated', u'text',
u'created_at', u'retweeted', ...]
```

Dictionaries

Dictionaries are like arrays, except rather than using numbers (0,1,2,...) to refer to the individual values they can use text:

```
d = dict([])
d['monday'] = 1
d['tuesday'] = 2
print d['monday']      -> 1
```

The names are called **keys**. To see a list of keys, we can use

```
print a[0].keys()
[u'favorited', u'in_reply_to_user_id',
u'contributors', u'truncated', u'text',
u'created_at', u'retweeted', ...]
```

One of these is called “text”, so is probably the actual tweet text:

```
print a[0]['text']
"AJAX 2-0 VVV, eindelijk weer 3 punten"
```

And we get the result - a Dutch football match score

Small Twitter client

We can put this together to make a program which prints out the last 20 public tweets:

```
import urllib2, json
addr="http://twitter.com/statuses/public_timeline.js
f = urllib2.urlopen(addr)
data = f.read()
f.close()
a = json.loads(data)
for tweet in a:
    print "At "+tweet['created_at']+ \
          " user "+tweet['user']['screen_name']
    print "      " + tweet['text']
```

Small Twitter client

We can put this together to make a program which prints out the last 20 public tweets:

```
import urllib2, json
addr="http://twitter.com/statuses/public_timeline.js
f = urllib2.urlopen(addr)
data = f.read()
f.close()
a = json.loads(data)
for tweet in a:
    print "At "+tweet['created_at']+ \
          " user "+tweet['user']['screen_name']
    print "      " + tweet['text']
```

```
At Sun Nov 28 16:32:21 +0000 2010 user Niek_vl
    AJAX 2-0 VVV, eindelijk weer 3 punten
At Sun Nov 28 16:32:14 +0000 2010 user KachingDeals
    *Please RT* How To Make Money Online Fast...
```

Other sources of data

There are many sources of data online which can be used in your Python programs. How easily you can extract the data will vary

- Infochimps is a big catalogue of online data sources on all sorts of data

  http://infochimps.com/

- Research Pipeline, variety of datasets

  http://www.researchpipeline.com/datasets.html

Other sources of data

There are many sources of data online which can be used in your Python programs. How easily you can extract the data will vary

- Infochimps is a big catalogue of online data sources on all sorts of data

  http://infochimps.com/

- Research Pipeline, variety of datasets

  http://www.researchpipeline.com/datasets.html

- Many sites can be automated, for example Google translate can be automated to detect language and translate text

  http://www.halotis.com/2009/09/15/google-translate-api-python-script/

Other sources of data

There are many sources of data online which can be used in your Python programs. How easily you can extract the data will vary

- Infochimps is a big catalogue of online data sources on all sorts of data

  http://infochimps.com/

- Research Pipeline, variety of datasets

  http://www.researchpipeline.com/datasets.html

- Many sites can be automated, for example Google translate can be automated to detect language and translate text

  http://www.halotis.com/2009/09/15/google-translate-api-python-script/

- Government organisations often make their data available, for example the World Bank has lots of economic data:

  http://data.worldbank.org/

Visualisation

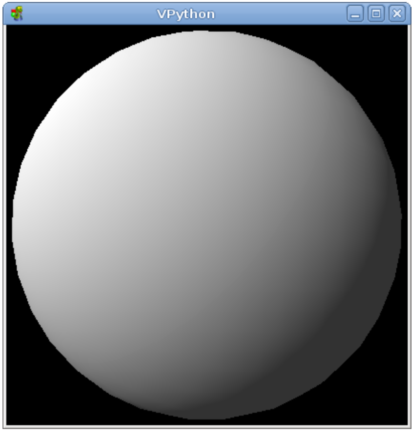
A very nice module is Visual Python (VPython), which makes creating 3D visualisations much easier

```
from visual import *
ball = sphere()        # Draw a sphere
```

Visualisation

A very nice module is Visual Python (VPython), which makes creating 3D visualisations much easier

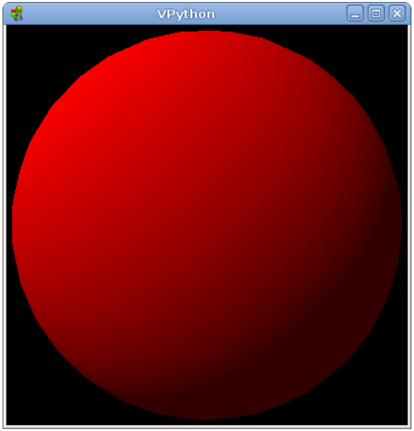
```
from visual import *
ball = sphere()        # Draw a sphere
```



Visualisation

We can change the size and color of the sphere

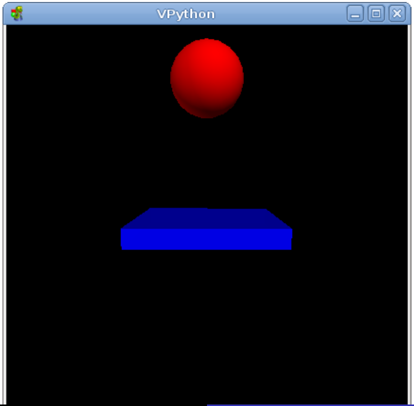
```
from visual import *
ball = sphere(radius=1, color=color.red)
```



Visualisation

We can add many different objects, such as boxes

```
from visual import *
ball = sphere(pos=(0,4,0),radius=1,color=color.red)
floor = box(pos=(0,0,0),size=(4,0.5,4),
            color=color.blue)
```



3D animations

The change in position in a small time δt is given by $\delta \underline{x} = \underline{v}\delta t$ for a velocity $\underline{v}$. We can use VPython vector to represent the velocity:

```
ball.velocity = vector(0,-1,0)
dt = 0.01
```

3D animations

The change in position in a small time δt is given by $\delta \underline{x} = \underline{v} \delta t$ for a velocity $\underline{v}$. We can use VPython vector to represent the velocity:

```
ball.velocity = vector(0,-1,0)
dt = 0.01
```

To change the position of the ball, we can just modify `ball.pos`:

```
ball.pos = ball.pos + ball.velocity*dt
```

3D animations

The change in position in a small time δt is given by $\delta \underline{x} = \underline{v}\delta t$ for a velocity $\underline{v}$. We can use VPython vector to represent the velocity:

```
ball.velocity = vector(0,-1,0)
dt = 0.01
```

To change the position of the ball, we can just modify `ball.pos`:

```
ball.pos = ball.pos + ball.velocity*dt
```

In the same way, the change in velocity in a small time δt is given by $\delta \underline{v} = \underline{a}\delta t$ where a is the acceleration. In this case we want to accelerate downwards (in y) due to gravity:

```
ball.velocity.y = ball.velocity.y - 9.8*dt
```

3D animations

The change in position in a small time δt is given by $\delta \underline{x} = \underline{v}\delta t$ for a velocity $\underline{v}$. We can use VPython vector to represent the velocity:

```
ball.velocity = vector(0,-1,0)
dt = 0.01
```

To change the position of the ball, we can just modify ball.pos:

```
ball.pos = ball.pos + ball.velocity*dt
```

In the same way, the change in velocity in a small time δt is given by $\delta \underline{v} = \underline{a}\delta t$ where $\underline{a}$ is the acceleration. In this case we want to accelerate downwards (in y) due to gravity:

```
ball.velocity.y = ball.velocity.y - 9.8*dt
```

This will just change the position and velocity once. To create an animation, we can move the ball repeatedly in a loop.

3D animations

The change in position in a small time δt is given by $\delta \underline{x} = \underline{v}\delta t$ for a velocity $\underline{v}$. We can use VPython vector to represent the velocity:

```
ball.velocity = vector(0,-1,0)
dt = 0.01

while True:      # Loop forever
    rate (100)   # Limit to 100 times per second
    ball.pos = ball.pos + ball.velocity*dt
    ball.velocity.y = ball.velocity.y - 9.8*dt
```

Each time this loops it updates the ball position and velocity. Unfortunately the ball will just go straight through the floor and keep going downwards. We need to detect when the ball hits the floor

3D animations

The change in position in a small time δt is given by $\delta \underline{x} = \underline{v}\delta t$ for a velocity $\underline{v}$. We can use VPython vector to represent the velocity:

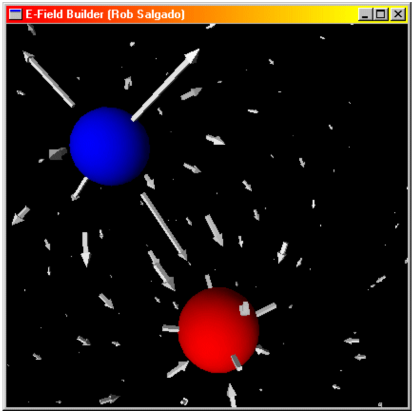
```
ball.velocity = vector(0,-1,0)
dt = 0.01

while True:      # Loop forever
    rate (100)   # Limit to 100 times per second
    ball.pos = ball.pos + ball.velocity*dt
    if ball.y < ball.radius:
        ball.velocity.y = -ball.velocity.y
    else:
        ball.velocity.y = ball.velocity.y - 9.8*dt
```

When the ball reaches the floor, this reverses the direction of the velocity so the ball bounces back up again.

Visual Python

Using VPython, it is possible to produce visualisations and animations of physical situations. You can add arrows and plot curves



Summary

- Python can do many more things than just calculations
- It is ideal for tasks involving quickly putting together programs to manipulate and visualise data
- Reading from internet sources is not much more complicated than from files. Built-in modules such as urllib2 and json simplify the process
- Visual Python makes it straightforward to visualise and animate physical situations
- Many drawing functions including arrow, box, cone, curve, cylinder, ellipsoid, helix, points, pyramid, ring, and sphere.

`http://www-users.york.ac.uk/~bd512/teaching.shtml`