Advanced Programming

Advanced Programming

Mike Smith

Email: msmith@cit.ac.ug

Office hours: ??

Find me in the AI Lab: Level 6 – block B

Also why not come to the AI Lab's weekly presentations? Thursdays at 10am

Getting python working...

Guido van Rossum,
the creator of Python
image: wikipedia

- Before we get started, let's get **python** installed and working.
- Python is a computer programming language which supports:
  - Object Orientation
  - Functional Programming
  - Automatic Memory Management
- To get it working we need the **python interpreter** and a way of **editing** and running python code.
- So need to install **idle,** an **integrated development environment** (IDE) for python.
- Idle's quite basic, but is quick and easy to install and use.

Installing **idle,** for **ubuntu**

Things are easier in **ubuntu** than in windows (no malware, easier installs, runs quicker, free, etc).

1. open a terminal and type:

```
:~$ sudo apt-get install idle
[sudo] password for lionfish:
Reading package lists... Done
Building dependency tree
```

Type in your password here

... etc ...

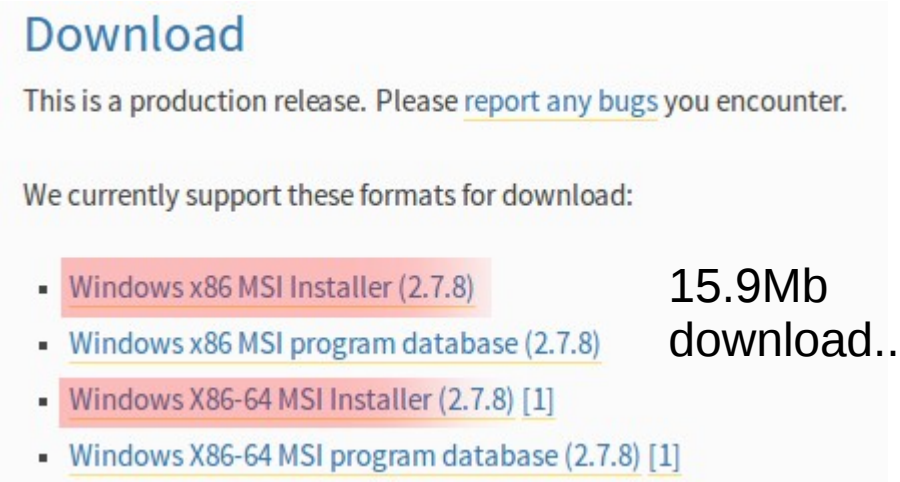
```
The following NEW packages will be installed
  idle idle-python2.7
0 to upgrade, 2 to newly install, 0 to remove ... etc ...
After this operation, 1,165 kB of additional disk
space will be used.
Do you want to continue? [Y/n]
```

2. Enter Y
3. Once installed, click on the ubuntu button and type in **idle**
4. Click on 'IDLE'

Installing **idle,** for **Windows**

1. Go to **www.python.org/download**
2. Click the 2.7.x version

3. Click either on x86 MSI Installer or X86-64 MSI Installer:

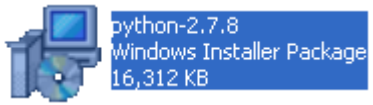


15.9Mb download...

Am I running 32 or 64 bit windows?

- ***Computers running Windows XP***
  - Click Start, right-click My Computer, and then click Properties.
  - The edition of Windows XP you're running is displayed under System near the top of the window.
  - If it doesn't say "x64 edition" then it is probably running 32 bit windows.
- ***Computers running Windows Vista or Windows 7***
  - Click the Start button, right-click Computer, and then click Properties.
  - The version is listed next to "System type"

Installing idle (for windows, continued)



Check it works

- Once idle is open, try the hello world program:

```
Python 2.7.6 (default, Mar 22 2014, 22:59:56)
[GCC 4.8.2] on linux2
Type "copyright", "credits" or "license()" for more
information.
>>> print "Hello World"
Hello World
>>>
```

About the Course

- Getting started with Python
- Object orientation
- Text processing (regular expressions)
- Databases
- Web Programming
  - Security/Vulnerabilities
  - Content Management Systems (maybe)
- Version Control (maybe)
- Using Linux/unix (maybe)
- Coding for microprocessors (maybe)
- XML (maybe)

This might change depending on the speed we can get through the content...

About the Course

- Assessment:
  - 40% coursework
  - 60% final exam
- The course will be mostly practical, with a few mini-lectures, every so often.

About the Course

The course covers **diverse topics**. Throughout the course we should be thinking about how to make our code:

Robust	Secure
Efficient	Maintainable

Robustness

"The ability of a computer system to cope with errors during execution"

- Ariane 5: $500M - 1996
- Used code from Ariane 4 – that wasn't retested properly...
- Concentrated on Efficiency – so removed a check on the size of a number (the horizontal speed).
- Tried to store this value in a space that was too small (2 bytes of memory).

Robustness

- Expect the unexpected.
- Test your code.
- Get someone else to test it some more.

Security

“Defending information from unauthorised access, disruption or destruction”

- Vulnerabilities come in **many forms**.
- Don't underestimate 'social engineering', i.e. **tricking** the administrator or user.
- Largest recent vulnerability was **Heart Bleed**.

Security

- 17.5% of SSL servers had this bug – this includes many 'big' sites (e.g. google, yahoo, dropbox...)
- Easily exploited.
- It allowed an attacker to gain huge amounts of data (including potentially the server's own keys)
- Attacks don't leave a trace (this type of packet typically isn't logged).
- The vulnerability was there for two years before it was fixed...

SSL = Secure Sockets Layer
Used eg. When you go to any https:// sites (eg your email)
OpenSSL is an implementation of this protocol.

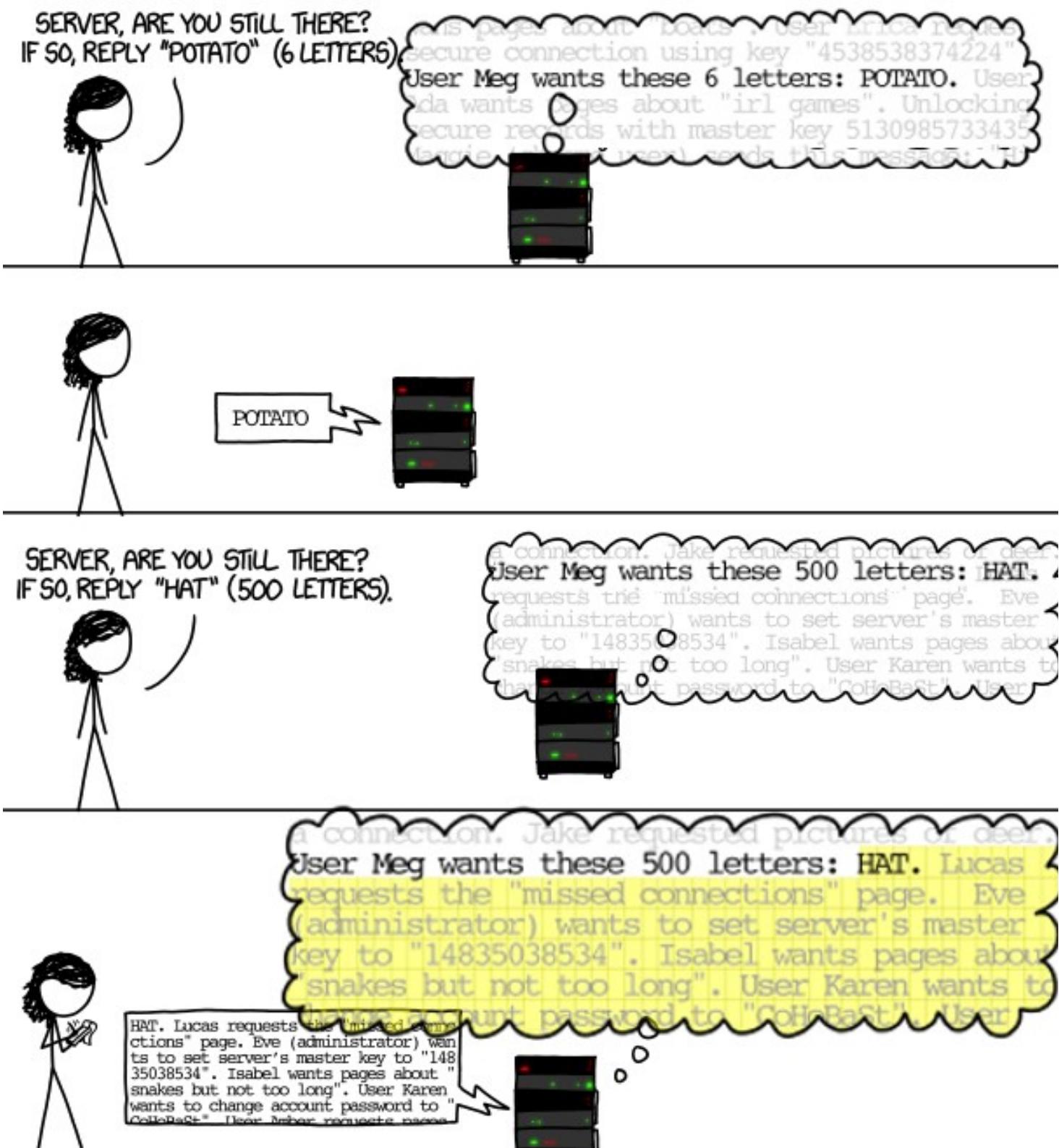
Security

“In the worst-case scenario, criminal enterprises, intelligence agencies, and state-sponsored hackers have known about Heartbleed for **more than two years**, and have used it to systematically access **almost everyone’s encrypted data**. If this is true, then anyone who does anything on the Internet has likely been affected by the bug.”

http://www.newyorker.com/tech/elements/the-internets-telltale-heartbleed

Security

- The Heartbeat extension allows a client to check the connection by sending a message which the server echos back.
- The client tells the server how long the message is...
- ...the problem is the server believes them.



xkcd.com

Security

```
1462        /* Read type and payload length first */
1463        hbtype = *p++;
1464        n2s(p, payload);
1465        pl = p;
 :
1477                /* Allocate memory for the response, size is 1 byte
1478                 * message type, plus 2 bytes payload length, plus
1479                 * payload, plus padding
1480                 */
1481                buffer = OPENSSL_malloc(1 + 2 + payload + padding);
1482                bp = buffer;
1483
1484                /* Enter response type, length and copy payload */
1485                *bp++ = TLS1_HB_RESPONSE;
1486                s2n(payload, bp);
1487                memcpy(bp, pl, payload);
1488                bp += payload;
 :
1492        r = dtls1_write_bytes(s,
                    TLS1_RT_HEARTBEAT, buffer, 3 + payload + padding);
```

Get the size of the payload from the client's packet (store its size in 'payload')

Copy the payload into the packet to send back...

Code from the **dtls1_process_heartbeat** function, in **ssl/d1_both.c**

Maintainable

- **Plan** before you start coding
  - Work out the components and what needs access to what: Avoid “build-then-plan”
- Structured
  - Split into **smaller** functions and classes.
- Consistent
  - Indentation, variable names, etc – use a **rule** and stick to it.
- Understandable?
  - Use **comments**: Explain what chunks of code do & use **good names**
- Don't duplicate code
  - If you find you've written similar code twice, you should have put it in a function and called it twice.
- Share/discuss code
  - If you're working with someone else, you have to make your code good for them.

Back to **python** and **idle**

- We've run 'hello world', but now to run it from a source file.
  - Create a folder to put your python code in
  - Click File → New File (in **idle**)
  - Type into the new file:

```
print “hello world”;
```

  - Save as helloworld.py
  - Click Run → Run Module

Work through Chapter 3

- 3.1.1 **Numbers**
- 3.1.2 **Strings**
- 3.1.4 **Lists**
- 3.2 **Programming**

3.1.1 Numbers

- #Comments
- Numbers,
  - Integer division? 7/3
  - Floating point? 7.0/3
  - Assignment: area = width*height;
- Define before use

Hint: **Tab complete!**
If a variable's been defined, start typing it and then press the '**tab**' button to let idle finish it for you.

3.1.2 Strings

Also see **unicode** for future reference

- Can use 'single' or “double” quotes
- Escaping quotes `'They\'re simple'`
- Other features: `r“r\a\w\” “““ ”””`
- Concatenate: `'hello' + ' ' + 'world'`
- `' test '.strip()` removes white space
- Accessing elements of an array, use **slice notation**:

```
        #01234567 (from start)
        #76543210 (from end)
msg = 'Advanced'
print msg[2:5] #gets items 2,3,4
print msg[2:] #gets items 2,3,4,5,6,7
print msg[-6:-3] #gets -5,-4,-3
```

3.1.4 Lists

- `Breakfast = ['eggs','toast','fruit',5]`
- Same instructions can happen to lists as strings
- `Breakfast[0]`
  - **`'Eggs'`**
- `Breakfast[-1]`
  - **`5`**
- `Breakfast[0] = 'honey'; #can change`
- `Breakfast[-1:] = [] #delete last item`
- `Breakfast[0:2] = ['muesli','juice']`
- `Breakfast`
  - **`['muesli', 'juice', 'fruit']`**

3.1.4 Lists (cont)

- `len(Breakfast)`
  - **`3`**
- `Morning = [Breakfast,'wash','go out']`
- `Morning`
  - **`[['muesli', 'juice', 'fruit'], 'wash', 'go out']`**
- `Breakfast[-1:] = [] #remove last (fruit)`
- `Breakfast`
  - **`['muesli', 'juice']`**
- `Morning`
  - **`[['muesli', 'juice'], 'wash', 'go out']`**
- `#`*`fruit`* `is missing from Morning too now`
- `Morning.append('dance') #etc...`

3.2 Programming: **fibonacci**

- See `tutorial.pdf` (page 17)
- While loop (indentation!)

```
while b < 10:
    print b
    a, b = b, a+b
```

Write your program in a new file, **`fibonacci.py`**

Advantages and disadvantages of this way of writing the two assignments?

- Multiple assignment

```
a, b = 1, 2
```

- Printing
  - difference between

```
print x
print x,
```

4.1. **if** statement...

- Problem: Write a program that works out a patient's BMI (body mass index) for a nutritionist to use.
- The program needs to ask for the patient's height and weight, e.g. using something like: `x = int(raw_input("Please enter an integer: "))`
- Then can use the equation:

$$\mathrm{BMI} = \frac{\mathrm{mass(kg)}}{(\mathrm{height(m)})^2}$$

  (to square something, use: `x**2`)
- Finally print if the person is:
  - Underweight (bmi<18.5)
  - normal range (18.5<bmi<25)
  - overweight (bmi>25)
  - You'll need to use the **if** statement,

```
if (x<=10):
  print “x is less than or equal to ten.”
elif (x==12):
  print “x is twelve.”
else:
  print “x is more than ten and not twelve.”
```

Robust?

- Is your program robust?
  - Try entering a floating point?
  - What about a height of zero?
  - What about text instead of numbers?

Exceptions

- A quick detour to chapter 8.3: **Exceptions**.
- You need to decide how to deal with exceptions,

```
try:
    bmi = weight/(height**2)
except ZeroDivisionError:
    print "Need a non-zero height"
```

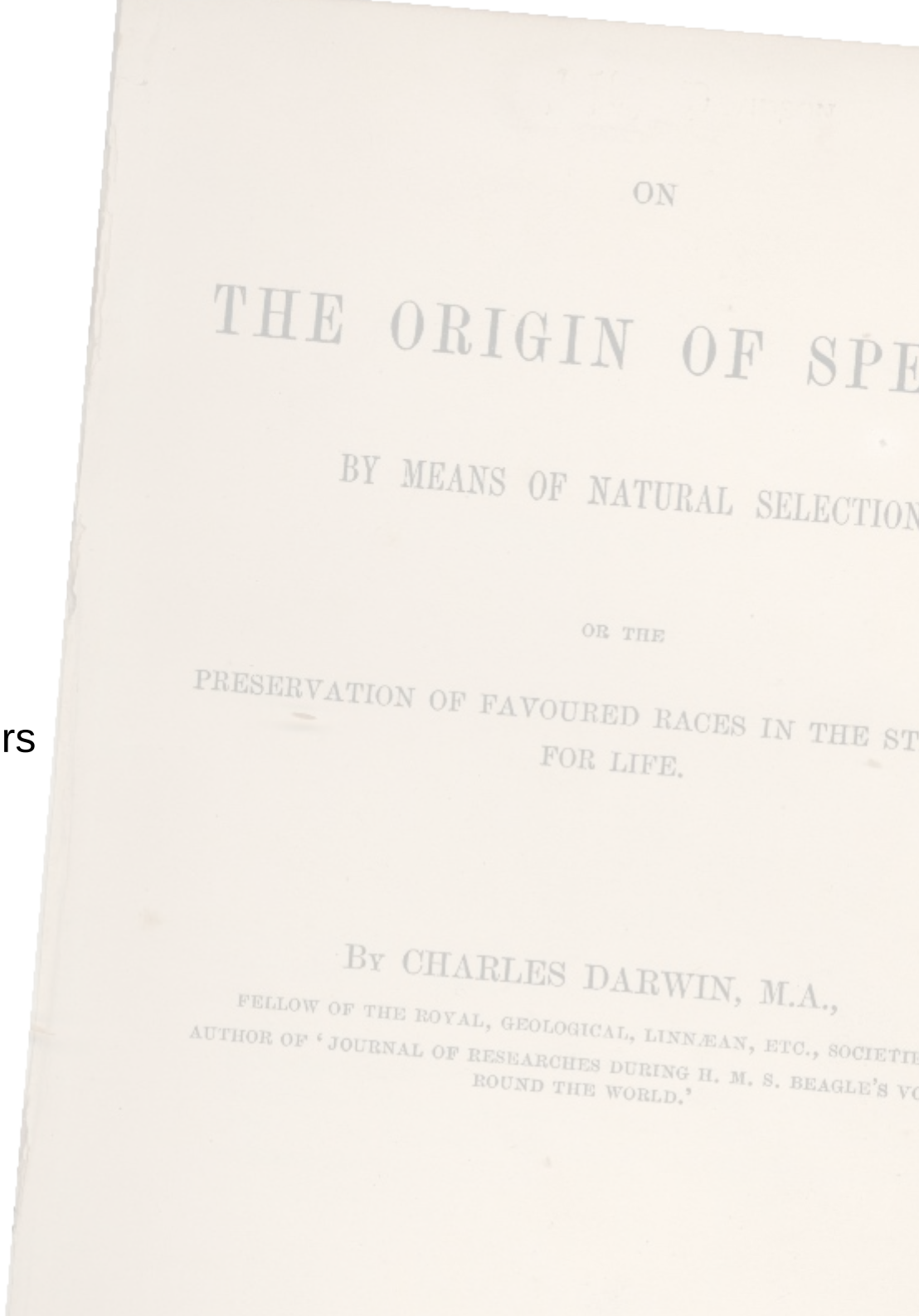
Exceptions

```
try:
    msg = input("Height: ");
    height = float(msg);
    bmi = weight/(height**2);
    print bmi;
except ZeroDivisionError:
    print "Need a non-zero height"
except NameError:
    print "Need a number"
```

4.2. **loops**

- Loop through a list...
  - `For x in [3,1,4]:`
- Loop through a list (making a copy)...
  - `For x in data[:]:`
- Problem:
  - Count the number of times the letters **e** and **q** occur in a text corpus.
  - Download **`darwin.txt`** (the introduction to *On The Origin of Species*) from muele or from:

`http://129.215.142.66/darwin.txt`



4.2. **loops**

- To read the file use:
  - **`content = open('darwin.txt','r').read()`**
- Maybe set a variable to zero and use it to count if a letter is 'e'
- To output you can use this format:

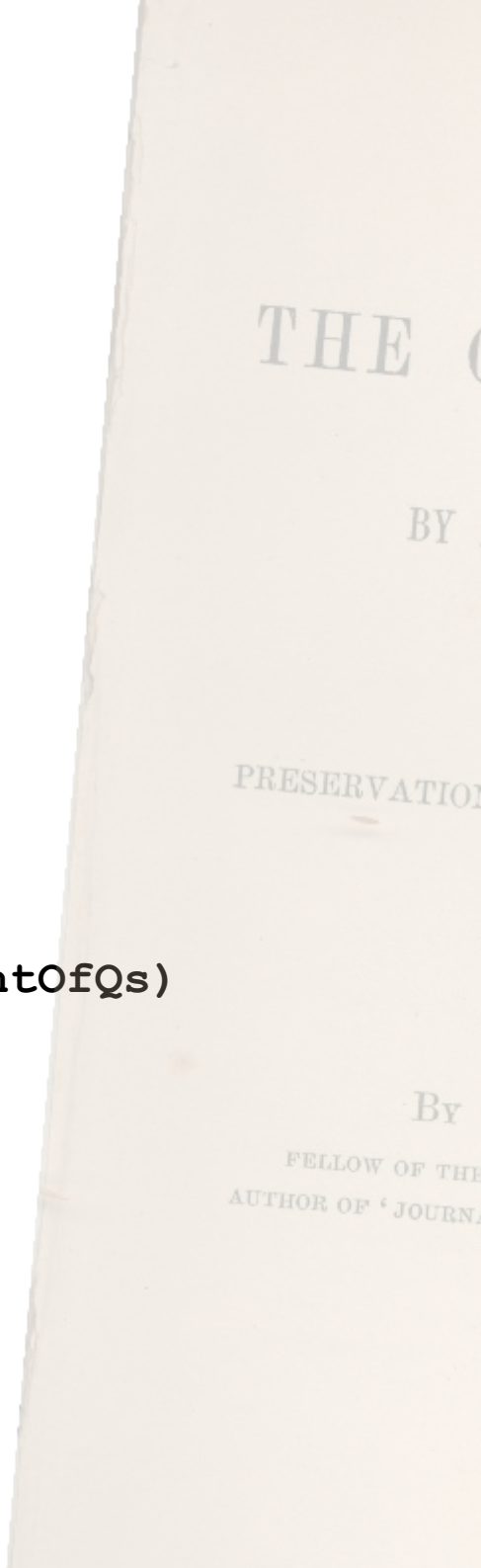
`print "This is an integer: %d" % (number)`

4.2. **loops**

```
content = open('darwin.txt','r').read()

countOfEs = 0;
countOfQs = 0;
for letter in content:
    if (letter=='e'):
        countOfEs+=1;
    if (letter=='q'):
        countOfQs+=1;

print "There were %d Es and %d Qs" % (countOfEs,countOfQs)
```



4.2.-4.5. **loops**

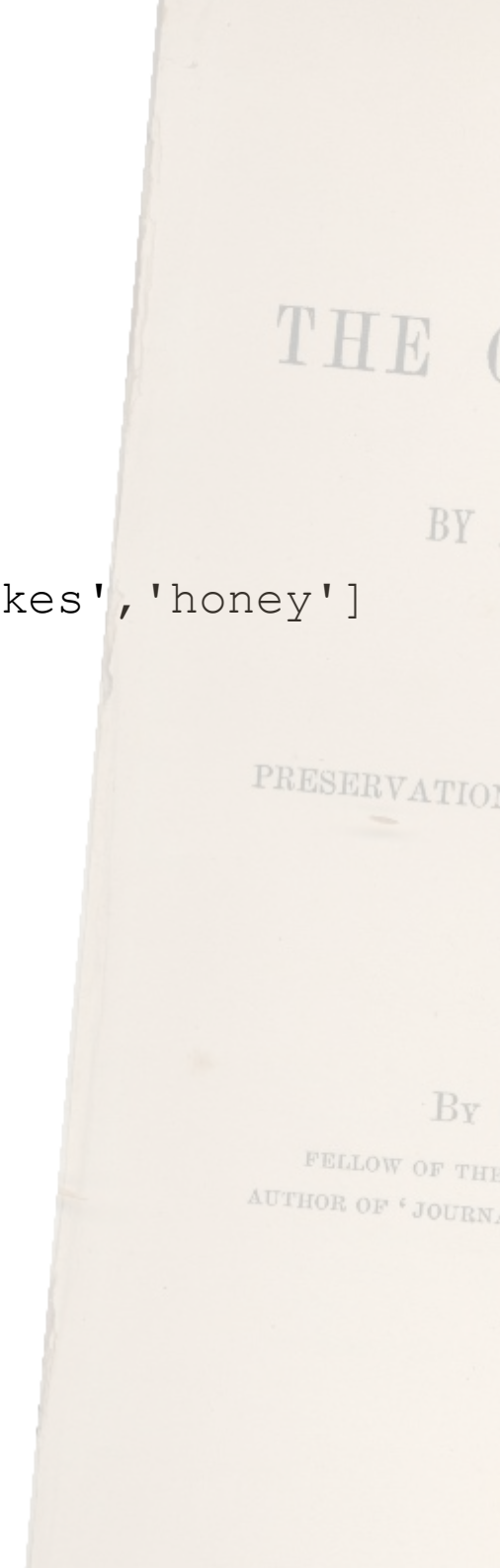
```
for x in range(10):
   print x

Breakfast = ['fruit','bun','juice','toast','cornflakes','honey']
for i, food in enumerate(Breakfast):
    if (len(food)<5):
        continue;
    if (len(food)>5):
        break;
    print i, food
```

0 fruit
2 juice
3 toast

Other features:

- **else**
- **pass**



4.6. Functions

- Look at section 4.6 in the tutorial (page 21pp).

```
def myFunction(n):
    stuffInFunctionHere();
```

- Problem:
  - Alter your bmi and darwin programs to use functions.

Next week

- **Object Orientation** and **Text Processing** (regexp)