The if statement and files
The if statement

Do a code block only when something is True

if test:
    print "The expression is true"
Example

if "GAATTC" in "ATCTGGGAATTCCATCG":
    print "EcoRI site is present"
if the test is true...

```python
if "GAATTC" in "ATCTGGAATTCCATCG":
    print "EcoRI site is present"
```

The test is: "GAATTC" in "ATCTGGAATTCCATCG"
Then **print** the message

```python
if "GAATTC" in "ATCTGGAATTCATCG":  
    print "EcoRI site is present"
```

Here is it done in the Python shell

```python
>>> if "GAATTC" in "ATCTGGAATTCATCG":  
    print "EcoRI is present"
...
EcoRI is present
>>>```
What if you want the false case?

There are several possibilities; here’s two

1) Python has a `not in` operator

   ```python
   if "GAATTC" not in "AAAAAAAAAAA":
       print "EcoRI will not cut the sequence"
   ```

2) The `not` operator switches true and false

   ```python
   if not "GAATTC" in "AAAAAAAAAAA":
       print "EcoRI will not cut the sequence"
   ```
In the Python shell

```python
>>> x = True
>>> x
True
>>> not x
False
>>> not not x
True
>>> if "GAATTC" not in "AAAAAAAAAA":
...     print "EcoRI will not cut the sequence"
...  
EcoRI will not cut the sequence
>>> if not "GAATTC" in "ATCTGGAATTCATCG":
...     print "EcoRI will not cut the sequence"
...  
...  
>>> if not "GAATTC" in "AAAAAAAAAA":
...     print "EcoRI will not cut the sequence"
...  
EcoRI will not cut the sequence

>>> 
```
else:

What if you want to do one thing when the test is true and another thing when the test is false?

Do the first code block (after the if:) if the test is true

```python
if "GAATTC" in "ATCTGGGAATTCATCG":
    print "EcoRI site is present"
else:
    print "EcoRI will not cut the sequence"
```

Do the second code block (after the else:) if the test is false
Examples with else

```python
>>> if "GAATTC" in "ATCTGGAATTCATCG":
...     print "EcoRI site is present"
... else:
...     print "EcoRI will not cut the sequence"
...
EcoRI site is present
>>> if "GAATTC" in "AAAAACTCGT":
...     print "EcoRI site is present"
... else:
...     print "EcoRI will not cut the sequence"
...
EcoRI will not cut the sequence
>>> 
```
Where is the site?

The ‘find’ method of strings returns the index of a substring in the string, or -1 if the substring doesn’t exist.

```python
>>> seq = "ATCTGGAATTCATCG"
>>> seq.find("GAATTC")
5
>>> seq.find("GGCGC")
-1
```

There is a GAATTC at position 5.
But there is no GGCGCG in the sequence.

```python
```
But where is the site?

```python
>>> seq = "ATCTGGGAATTCCATCG"
>>> pos = seq.find("GAATTC")
>>> if pos == -1:
...    print "EcoRI does not cut the sequence"
... else:
...    print "EcoRI site starting at index", pos
EcoRI site starting at index 5
```
Start by creating the string “ATCTGGAATTCCATCG” and assigning it to the variable with name ‘seq’

```python
seq = "ATCTGGAATTCCATCG"
pos = seq.find("GAATTC")
if pos == -1:
    print "EcoRI does not cut the sequence"
else:
    print "EcoRI site starting at index", pos
```
Using the `seq` string, call the method named `find`. This looks for the string “GAATTC” in the `seq` string.

```python
seq = "ATCTGGAATTCATCG"
pos = seq.find("GAATTC")
if pos == -1:
    print "EcoRI does not cut the sequence"
else:
    print "EcoRI site starting at index", pos
```
The string “GAATC” is at position 5 in the seq string. Assign the 5 object to the variable named pos.

```python
seq = "ATCTGGAATTCCATCG"
pos = seq.find("GAATTC")
if pos == -1:
    print "EcoRI does not cut the sequence"
else:
    print "EcoRI site starting at index", pos
```

The variable name “pos” is often used for positions. Common variations are “pos1”, “pos2”, “start_pos”, “end_pos”
Do the test for the if statement

Is the variable pos equal to -1?

seq = "ATCTGGAATTTCATCG"
pos = seq.find("GAATTC")
if pos == -1:
    print "EcoRI does not cut the sequence"
else:
    print "EcoRI site starting at index", pos
Since pos is 5 and 5 is not equal to -1, this test is false.

```python
seq = "ATCTGGAATTCATCG"
pos = seq.find("GAATTC")
if pos == -1:
    print "EcoRI does not cut the sequence"
else:
    print "EcoRI site starting at index", pos
```

The test is False
Skip the first code block
(that is only run if the test is True)
Instead, run the code block after the else:

```python
seq = "ATCTGGGAATTTCATCG"
pos = seq.find("GAATTC")
if pos == -1:
    print "EcoRI does not cut the sequence"
else:
    print "EcoRI site starting at index", pos
```
This is a print statement.
Print the index of the start position

seq = "ATCTGGGAATTCCATCG"
pos = seq.find("GAATTC")
if pos == -1:
    print "EcoRI does not cut the sequence"
else:
    print "EcoRI site starting at index", pos

This prints

EcoRI site starting at index 5
There are no more statements so Python stops.

```python
seq = "ATCTGGAATTCATCG"
pos = seq.find("GAATTC")
if pos == -1:
    print "EcoRI does not cut the sequence"
else:
    print "EcoRI site starting at index", pos
```
A more complex example

Using if inside a for

```python
restriction_sites = ['GAATTC',    # EcoRI
                   'GGATCC',    # BamHI
                   'AAGCTT',    # HindIII
                ]

seq = raw_input("Enter a DNA sequence: ")

for site in restriction_sites:
    if site in seq:
        print site, "is a cleavage site"
    else:
        print site, "is not present"
```
Nested code blocks

```python
restriction_sites = [
    "GAATTC",    # EcoRI
    "GGATCC",    # BamHI
    "AAGCTT",    # HindIII
]

seq = raw_input("Enter a DNA sequence: ")

for site in restriction_sites:
    if site in seq:
        print site, "is a cleavage site"
    else:
        print site, "is not present"
```

This is the code block for the for statement
restriction_sites = [
    "GAATTC",    # EcoRI
    "GGATCC",    # BamHI
    "AAGCTT",    # HindIII
]

seq = raw_input("Enter a DNA sequence: ")

for site in restriction_sites:
    if site in seq:
        print site, "is a cleavage site"
    else:
        print site, "is not present"

This is the code block for the True part of the if statement
restriction_sites = [
    "GAATTC",    # EcoRI
    "GGATCC",    # BamHI
    "AAGCTT",    # HindIII
]

seq = raw_input("Enter a DNA sequence: ")

for site in restriction_sites:
    if site in seq:
        print site, "is a cleavage site"
    else:
        print site, "is not present"
The program output

Enter a DNA sequence: AATGAATTCTCTGGAAGCTTTA
GAATTC is a cleavage site
GGATCC is not present
AAGCTT is a cleavage site
Read lines from a file

• `raw_input()` asks the user for input
• Most of the time you’ll get data from a file. (Or would you rather type in the sequence every time?)
• To read from a file you need to tell Python to open that file.
The open function

```python
>>> infile = open("/usr/coursehome/dalke/10_sequences.seq")
>>> print infile
<open file '/usr/coursehome/dalke/10_sequences.seq', mode 'r' at 0x817ca60>

open returns a new object of type file

A file can’t be displayed like a number or a string. It is useful because it has methods for working with the data in the file.
the readline() method

```python
>>> infile = open("/usr/coursehome/dalke/10_sequences.seq")
>>> print infile
<open file '/usr/coursehome/dalke/10_sequences.seq', mode 'r' at 0x817ca60>
>>> infile.readline()
'CCTGTATTAGCAGCAGATTCGATTAGCTTTACACAAATTTCAATAAAATAGCTTTCGCGCTAA\n'
```
When there are no more lines, readline returns the empty string.
Using `for` with a file

A simple way to read lines from a file

```python
>>> filename = "\"/usr/coursehome/dalke/10_sequences.seq\""
>>> for line in open(filename):
...     print line[:10]
...
CCTGTATTAG
ATTTTTAACT
TTGTTTCTGC
CACACCCAAAT
TCTTCTCCAA
CCACACCAAA
GTGCTCTCTT
CCACACCAAA
TCTGAAAAGT
GTATTGGTCG

>>> for starts with the first line in the file ...
then the second ...
then the third ...
...
and finishes with the last line.
```
A more complex task

List the sequences starting with a cytosine

```python
>>> filename = "~/usr/coursehome/dalke/10_sequences.seq"
>>> for line in open(filename):
...     line = line.rstrip()  # Use rstrip to get rid of the "\n"
...     if line.startswith("C"):
...         print line
...
CCTGTATTAGCAGCAGATTCTAGCTTTACAAACATCTCTTACATCTTAATAATAGCTTCGCGCTAA
CACACCCAATAAGTTAGAGAGTACTTTGACTTGGAGCTGGAGGAATTTGACATAGTCGAT
CCACACCAAAAAACTTTCCACGTGAACCGAAACGAAAGTCTTTGGTTTTAATCAATAA
CCACACCAAAAAACTTTCCACGTGTGAACATATACTCCAAAAACGAAAGTATTGGTTTATCAAA
>>>```
Exercise I

Get a sequence from the user. If there is an A in the sequence, print the number of times it appears in the sequence. Do the same for T, C and G. If a base does not exist, don’t print anything.

Enter a sequence: ACCAGGCA

Test input #1:

A count: 3
C count: 3
G count: 2

Enter a sequence: TTTTTTGGGG

Test input #2:

T count: 5
G count: 4
Excercise 2

Get a sequence from the user. If there is an A in the sequence, print the number of times it appears in the sequence. If it does not exist, print “A not found”. Do the same for T, C and G.

Enter a sequence: ACCAGGCA
A count: 3
T not found
C count: 3
G count: 2

Test input #1:

Enter a sequence: TTTTGGGG
A not found
T count: 5
C not found
G count: 4
Exercise 3

number lines in a file

Read the file /usr/coursehome/dalke/10_sequences.seq . Print out the line number (starting with 1) then the line. Remember to use rstrip() to remove the extra newline.

The output should look like this

1  CCTGTATTAGCAGCAGATTCGATTAGCTTTACAAACATTTTTACAAATTTATAGCTTTCCGCCTAA
2  ATTTTTAACCTTTTCCTCTGTGTCGTCGACCAATCGACTTTTCCTCTGTGTTTCTGGTTTACCAGAA
3  TTGGTTTCTGCTGCCATGCATGTATTGCTCGTCAGCCTGAGGCTGAAAATAAAATCCGTGGTT
4  CACACCAAATAAGTTAGAGAGGCTTTTGACTTGAGCTGGAGGATTTGAACATAGTGAT
5  TCTTCTCAAAGACGCATCCACGTGAGGCTTTGAACATATGTTCTCTGTGCT
6  CCACACCAAAAAACCTTTCCACGTGAACCCGAAAAACGAAAGCTTCTTGGTTTTAATCAAATA
7  GTGCTCTCCTTCGAGAGGAGGCTGCTGCTCTGTGACAGCTGTACTTTCAAAATCCAAATA
8  CCACACCAAAAAACCTTTCCACGTGAACCTATATATCTCCAAAAACGAAATTTGTTTATCATA
9  TCTGAAAAGTGCAAFAQAGGATGATGATGATGATAGAGGAACCTGAGCAGCCATGCTGAACCTATAGC
10  GTATTGGTCGTGTCGACTAAATGTTAATAAAAGTAGTTCTAAGAGATTTTGATGATTCAATGCAAAGTTCTTTATCGTTCAATTG
Exercise 4

List the sequences in /usr/coursehome/dalke/10_sequences.seq which have the pattern CTATA.

Hint: You should find two of them.

Once that works, print the index of the first time that pattern is found.
Exercise 5 - Filtering

Using /usr/coursehome/dalke/sequences.seq

A. How many sequences are in that file?
B. How many have the pattern CTATA?
C. How many have more than 1000 bases?
D. How many have over 50% GC composition?
E. How many have more than 2000 bases and more than 50% GC composition?

Note: for %GC use float to convert the counts into floats before doing the division for percentage.