Programming in Python
Today...

We will look at:

- Caesar’s Cipher
- Classes, objects and encapsulation.
- Instance variable vs local variables.
- Constructors and methods.
- GUI Design Tools.
Re-cap
Editing Files

To Edit Files: **TextWrangler.**

‘Source code’ is stored in .py files, e.g. “program.py”
Create one file per program.

To Compile Files: **Terminal.**

“cd” – change directory, e.g. cd Desktop
“ls” – list files, e.g. ls Desktop
“python” – to run python, e.g. python myProg.py

Notes:

• **PyTHon iS CaSe SeNsItiVe.**
• **Comments:** # comment.
Last Week: Lists

**Syntax**

<table>
<thead>
<tr>
<th>To create a list:</th>
<th>[ value_1, value_2, \ldots ]</th>
</tr>
</thead>
<tbody>
<tr>
<td>To access an element:</td>
<td>listReference[index]</td>
</tr>
</tbody>
</table>

**Example**

- Name of list variable: `moreValues = []`
  - Creates an empty list
- Initial values: `values = [32, 54, 67, 29, 35, 80, 115]`  
  - Creates a list with initial values

**Use brackets to access an element.**

- `values[i] = 0`
- `element = values[i]`
Many Ways to Traverse Lists

# First version (list index used)
for i in range(10):
    print(i, values[i])

# Better version (list index used)
for i in range(len(values)):
    print(i, values[i])

# Third version: index values not needed (traverse list elements)
for element in values:
    print(element)
Reading Files

To read an entire line python provides:

```python
line = infile.readline()
```

Can be used within a loop to read the entire file!

```python
line = infile.readline()
while line != "":
    # Process the line.
    line = infile.readline()
infile.close()
```
Writing to Files

Writing to files is pretty similar:

To open the file for writing:

```python
outfile = open("output.txt", "w")
```

To write a string to the file:

```python
outfile.write("Hello, World!\n")
outfile.close()
```
Caesar's Cipher
The Setup

(Plaintext) Hello World!

(Plaintext) Hello World!

(plaintext) %#giuyrkwn,m,s:?

Encryption

Decryption

(Shared Secret Key)
Caesar’s Shift

A very well known form of encryption is the shift cipher.

Idea: Shift each character n positions. For example, n=3:

Encryption:
Plaintext....: ABCDEFGHIJKLMNOPQRSTUVWXYZ
Ciphertext.: DEFGHIJKLMNOPQRSTUVWXYZABC

Decryption: Easy its just the reverse :)

Diagram of the Caesar's Shift cipher showing the shift of three positions.
### Character Representations

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hx Oct</th>
<th>Char</th>
<th>Hx Oct</th>
<th>Htmlm</th>
<th>Chr</th>
<th>Hx Oct</th>
<th>Htmlm</th>
<th>Chr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000</td>
<td>NUL</td>
<td>32 20</td>
<td>040</td>
<td>&lt; #32; Space</td>
<td>64 40 100</td>
<td>0064</td>
<td>&lt;</td>
</tr>
<tr>
<td>1</td>
<td>0001</td>
<td>SOH</td>
<td>33 21</td>
<td>041</td>
<td>!</td>
<td>65 41 101</td>
<td>065</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>0002</td>
<td>STX</td>
<td>34 22</td>
<td>042</td>
<td>&quot;</td>
<td>66 42 102</td>
<td>066</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>0003</td>
<td>ETX</td>
<td>35 23</td>
<td>043</td>
<td>#</td>
<td>67 43 103</td>
<td>067</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>0004</td>
<td>EOT</td>
<td>36 24</td>
<td>044</td>
<td>$</td>
<td>68 44 104</td>
<td>068</td>
<td>D</td>
</tr>
<tr>
<td>5</td>
<td>0005</td>
<td>ENQ</td>
<td>37 25</td>
<td>045</td>
<td>%</td>
<td>69 45 105</td>
<td>069</td>
<td>E</td>
</tr>
<tr>
<td>6</td>
<td>0006</td>
<td>ACK</td>
<td>38 26</td>
<td>046</td>
<td>&lt;</td>
<td>70 46 106</td>
<td>070</td>
<td>F</td>
</tr>
<tr>
<td>7</td>
<td>0007</td>
<td>BEL</td>
<td>39 27</td>
<td>047</td>
<td>!</td>
<td>71 47 107</td>
<td>071</td>
<td>G</td>
</tr>
<tr>
<td>8</td>
<td>0010</td>
<td>BS</td>
<td>40 28</td>
<td>050</td>
<td>(</td>
<td>72 48 110</td>
<td>072</td>
<td>H</td>
</tr>
<tr>
<td>9</td>
<td>0011</td>
<td>HTB</td>
<td>41 29</td>
<td>051</td>
<td>)</td>
<td>73 49 111</td>
<td>073</td>
<td>I</td>
</tr>
<tr>
<td>10</td>
<td>0012</td>
<td>LF</td>
<td>42 2A</td>
<td>052</td>
<td>*</td>
<td>74 4A 112</td>
<td>074</td>
<td>J</td>
</tr>
<tr>
<td>11</td>
<td>0013</td>
<td>VT</td>
<td>43 2B</td>
<td>053</td>
<td>+</td>
<td>75 4B 113</td>
<td>075</td>
<td>K</td>
</tr>
<tr>
<td>12</td>
<td>0014</td>
<td>FF</td>
<td>44 2C</td>
<td>054</td>
<td>-</td>
<td>76 4C 114</td>
<td>076</td>
<td>L</td>
</tr>
<tr>
<td>13</td>
<td>0015</td>
<td>CARR</td>
<td>45 2D</td>
<td>055</td>
<td>:</td>
<td>77 4D 115</td>
<td>077</td>
<td>M</td>
</tr>
<tr>
<td>14</td>
<td>0016</td>
<td>SO</td>
<td>46 2E</td>
<td>056</td>
<td>;</td>
<td>78 4E 116</td>
<td>078</td>
<td>N</td>
</tr>
<tr>
<td>15</td>
<td>0017</td>
<td>SI</td>
<td>47 2F</td>
<td>057</td>
<td>/</td>
<td>79 4F 117</td>
<td>079</td>
<td>O</td>
</tr>
<tr>
<td>16</td>
<td>0020</td>
<td>DLE</td>
<td>48 30</td>
<td>060</td>
<td>0</td>
<td>80 50 120</td>
<td>080</td>
<td>P</td>
</tr>
<tr>
<td>17</td>
<td>0021</td>
<td>DC1</td>
<td>49 31</td>
<td>061</td>
<td>1</td>
<td>81 51 121</td>
<td>081</td>
<td>Q</td>
</tr>
<tr>
<td>18</td>
<td>0022</td>
<td>DC2</td>
<td>50 32</td>
<td>062</td>
<td>2</td>
<td>82 52 122</td>
<td>082</td>
<td>R</td>
</tr>
<tr>
<td>19</td>
<td>0023</td>
<td>DC3</td>
<td>51 33</td>
<td>063</td>
<td>3</td>
<td>83 53 123</td>
<td>083</td>
<td>S</td>
</tr>
<tr>
<td>20</td>
<td>0024</td>
<td>DC4</td>
<td>52 34</td>
<td>064</td>
<td>4</td>
<td>84 54 124</td>
<td>084</td>
<td>T</td>
</tr>
<tr>
<td>21</td>
<td>0025</td>
<td>NAK</td>
<td>53 35</td>
<td>065</td>
<td>5</td>
<td>85 55 125</td>
<td>085</td>
<td>U</td>
</tr>
<tr>
<td>22</td>
<td>0026</td>
<td>SYM</td>
<td>54 36</td>
<td>066</td>
<td>6</td>
<td>86 56 126</td>
<td>086</td>
<td>V</td>
</tr>
<tr>
<td>23</td>
<td>0027</td>
<td>ETB</td>
<td>55 37</td>
<td>067</td>
<td>7</td>
<td>87 57 127</td>
<td>087</td>
<td>W</td>
</tr>
<tr>
<td>24</td>
<td>0030</td>
<td>CAN</td>
<td>56 38</td>
<td>070</td>
<td>8</td>
<td>88 58 130</td>
<td>088</td>
<td>X</td>
</tr>
<tr>
<td>25</td>
<td>0031</td>
<td>EM</td>
<td>57 39</td>
<td>071</td>
<td>9</td>
<td>89 59 131</td>
<td>089</td>
<td>Y</td>
</tr>
<tr>
<td>26</td>
<td>0032</td>
<td>SUB</td>
<td>58 3A</td>
<td>072</td>
<td>:</td>
<td>90 5A 132</td>
<td>090</td>
<td>Z</td>
</tr>
<tr>
<td>27</td>
<td>0033</td>
<td>ESC</td>
<td>59 3B</td>
<td>073</td>
<td>;</td>
<td>91 5B 133</td>
<td>091</td>
<td>[</td>
</tr>
<tr>
<td>28</td>
<td>0034</td>
<td>FS</td>
<td>60 3C</td>
<td>074</td>
<td>&lt;</td>
<td>92 5C 134</td>
<td>092</td>
<td>\</td>
</tr>
<tr>
<td>29</td>
<td>0035</td>
<td>GS</td>
<td>61 3D</td>
<td>075</td>
<td>=</td>
<td>93 5D 135</td>
<td>093</td>
<td>]</td>
</tr>
<tr>
<td>30</td>
<td>0036</td>
<td>RS</td>
<td>62 3E</td>
<td>076</td>
<td>&gt;</td>
<td>94 5E 136</td>
<td>094</td>
<td>^</td>
</tr>
<tr>
<td>31</td>
<td>0037</td>
<td>US</td>
<td>63 3F</td>
<td>077</td>
<td>?</td>
<td>95 5F 137</td>
<td>095</td>
<td>_</td>
</tr>
</tbody>
</table>

Source: www.LookupTables.com
Hint: Character Representations

The `ord` and `chr` functions in python may be useful!

Example: `ord` gives ascii value for a character.

```
ord("a") gives 97
ord("A") gives 65
```

Also note: `chr` gives character represented by the given number.

```
chr(65) gives "a"
```

Thinking point: Can we use this to compute Caesar’s cipher?
GCSE Task

Task 1:
In groups discuss how you can implement Caesar’s cipher (for any shift) using the `ord` and `chr` functions.

Task 2:
Write a program that computes the Caesar shift cipher for a lower case string.

Break your solution down into the following:
- Ask the user to input a message (within a main function).
- Ask the user to input the offset (within a main function).
- Write a “encode” function, that take a message and an offset returns the ciphertext.
- Write a “decode” function, that take a ciphertext and an offset returns the plaintext.

Challenge Task:
- Extend your solution to respect capital letters.
Object Oriented Programming
What We Have Covered

You have learnt structured programming:

• Basic Constructs
• Breaking tasks into subtasks
• Writing re-usable functions to handle tasks

We will now study Objects and Classes:

• To build larger and more complex programs
• To model objects we use in the world
Classes and Objects

A class describes a set of objects with the same behavior.

Each object then has its own set of data, together with a set of methods that act upon the data.

Such a situation is usually captured using a Class Diagram:
A Concrete Class

The str class describes the behavior of all strings.

This class specifies how a string stores its characters, which methods can be used with strings, and how the methods are implemented.

For example, when you have a str object, you can invoke the upper method:

```
"Hello, World".upper()
```

String object

Method of class String
Objects/Instances

What would be a typical object for the string class?
Objects/Instances

What would be a typical object for the string class?

Any string!

Examples:
  “Hello”
  “3+5”
  “My name is...”
Public Interfaces

When you work with an object of a class, you do not (need to) know how the object stores its data, or how the methods are implemented.

All you need to know is the public interface—–which methods you can apply, and what these methods do.

**Idea**: Encapsulation!

**Result**: When the implementation is hidden, the improvements do not affect the programmers who use the objects.

What is the public interface of the String class?
7.1. string — Common string operations

The `string` module contains a number of useful constants and classes, as well as some deprecated legacy functions that are also available as methods on strings. In addition, Python’s built-in string classes support the sequence type methods described in the `Sequence Types — str, unicode, list, tuple, bytearray, buffer, xrange` section, and also the string-specific methods described in the `String Methods` section. To output formatted strings use template strings or the `%` operator described in the `String Formatting Operations` section. Also, see the `re` module for string functions based on regular expressions.

7.1.1. String constants

The constants defined in this module are:

- `string.ascii_letters`
  - The concatenation of the `ascii_lowercase` and `ascii_uppercase` constants described below. This value is not locale-dependent.

- `string.ascii_lowercase`
  - The lowercase letters `\'abcdefghijklmnopqrstuvwxyz\'`. This value is not locale-dependent and will not change.

- `string.ascii_uppercase`
  - The uppercase letters `\'ABCDEFGHIJKLMNOPQRSTUVWXYZ\'`. This value is not locale-dependent and will not change.

- `string.digits`
  - The string `\'0123456789\'`.

- `string.hexdigits`
  - The string `\'0123456789abcdefABCDEF\'`.

- `string.letters`
  - The concatenation of the strings `lowercase` and `uppercase` described below. The specific value is locale-dependent, and will be updated when `locale.setlocale()` is called.

- `string.lowercase`
  - A string containing all the characters that are considered lowercase letters. On most systems this is the string `\'abcdefghijklmnopqrstuvwxyz\'`. The specific value is locale-dependent, and will be updated when `locale.setlocale()` is called.

- `string.octdigits`
  - The string `\'01234567\'`.

- `string.punctuation`
  - String of ASCII characters which are considered punctuation characters in the `c` locale.

- `string.printable`
Class Concepts by Example (Counter)
The Concept

Tally Counter: A class that models a mechanical device that is used to count people.

What should it do?

-- Be able to create counters
-- Increment the tally
-- Get the current total
-- Reset the counter to zero
The Class Diagram

```
Driver
+main()

Counter
+_value: int
+click()
+reset()
+getCounter(): int
```
Group Task :)

Logo: technocamps
Object Orientated Tasks

Task 1:
As a group discuss the class diagram for the following example:

The Head of History has asked you to write an application that will help her store and analyze the test marks for her Year 11 pupils.

Your task is to write an application to:
- Read in data from a file each line contains a student and a mark.
- Calculate the grade (based on some grade table – see next slide).
- Store the data (hint: list of pupils?)
- Output the number of pupils who achieve each grade
- Output the names of pupils who achieve an A or A* grade.
Example: Counter Class Code

class Counter :
    def __init__(self) :
        self._value = 0

    ## Gets the current value of this counter.
    def getValue(self) :
        return self._value

    ## Advances the value of this counter by 1.
    def click(self) :
        self._value = self._value + 1

    ## Resets the value of this counter to 0.
    def reset(self) :
        self._value = 0
Creating Objects: Constructors

A constructor is a method that initializes an object.

Consider this call:

```python
tally = Counter()  # Creates an instance
```

This creates a new Counter instance. Within the counter class we have:

```python
def __init__(self):
    self._value = 0
```

Python uses the special name `__init__` for the constructor — its purpose is to initialize an instance of the class.
Data in Objects: Instance Variables

An object stores its data in so-called instance variables.

In our example, each Counter object has a single instance variable named _value.

Now, if concertCounter and boardingCounter are two objects of the Counter class, then each object has its own _value variable.
Methods for Objects

A method definition is very similar to a function with these exceptions:

1. A method is defined as part of a class definition.
2. The first parameter variable of a method is (usually) called self.

For example:
The click() method advances the _value instance variable by 1.

```python
def click(self):
    self._value = self._value + 1
```
“Setters” and “Getters”

It is good practice to provide a setter and getter method for every instance variable:

Example: Getter

```python
# Gets the current value of this counter.
def getValue(self):
    return self._value
```

Example: Setter

```python
# Sets the current value of this counter.
def setValue(self, val):
    self._value = val
```
Invoking Methods Within the Class

A small difference to functions, is the way that methods are called. They need the calling object!

From within the class:

```python
self.click()
```

From another class:

```python
c = Counter()
c.click()
```
Complete Counter Class Code

class Counter:
    def __init__(self):
        self._value = 0

    # Gets the current value of this counter.
    def getValue(self):
        return self._value

    # Advances the value of this counter by 1.
    def click(self):
        self._value = self._value + 1

    # Resets the value of this counter to 0.
    def reset(self):
        self._value = 0

# Notice I have excluded a setter!
Using The Counter

```python
from counter import Counter

tally = Counter()
tally.reset()
tally.click()
tally.click()

result = tally.getValue()
print("Value:", result)

tally.click()
result = tally.getValue()
print("Value:", result)
```
Some Important Notes!

Some things that **should** be remembered:

- All methods in a class have a first parameter named `self`.
- All instance variables should start with a `_`
- All classes should have a `__init__` method.
- Usually there is a single “Driver” class containing the `main` method.
Object Orientated Tasks (Cont.)

Task 2:
Re-read the slides on the Counter example. Then implement the counter class.

Task 3:
From another file, create two different Counter’s (Maybe concertCounter and busCounter) and call various methods on them.

Challenge Task:
Consider the counter class, extend the code so that the counter stops counting at 100. If the limit is reached, then the user should be informed.
GUI Programming and Useful Bits
Lots of Tools!

There are lots of tools to help generate code for Python GUIs. Including:

wxPython
-- Standalone, cross platform GUI designer
IronPython Studio

IronPython studio – Visual Studio Plugin (Much like VB):
Further Reading

All of the course notes and coding examples are available online at:

http://cs.swansea.ac.uk/~cspj/teaching/pythonCPD

Remember code academy:

http://www.codecademy.com/tracks/python

Many examples have been taken from “Python for Everyone” by Cay Horstmann and Rance Necaise.


Lots of teaching material for this book online!
Summary

We have explored classes and objects on the conceptual level.

You now have all the tools you need to program just about anything!

Just remember:
1. Libraries and the internet are your friend.
2. Explaining your program to others will help your own understanding.

Next Week: We will solve 2 more GCSE examples using the concept of classes.