Mixing C and Python

Comp-206 : Introduction to Software Systems
Lecture 19

Alexandre Denault
Computer Science
McGill University
Fall 2006

Lets talk about the final

- Wednesday, December 13th, 2006
- Somewhere in the Arts building.
- Covers topics from all year.
- 18 questions.
  - 3 of them require you to write some form of code.
- Somewhat similar to the midterm.

Assignment 2

- You now have an extension for Assignment 2.
- You can now handin your assignment until Thursday, November 16th without penalty.
- However, if you handin your assignment on Friday, it will be considered 3 days late.

Plagiarism

Representing the work or ideas of another person as your own in any academic writing, essay, thesis, research, report, project or assignment submitted in a course or program of study or represent as your own an entire essay or work of another, whether the material so represented constitutes a part of the entirety of the work submitted.

Article 15 (a) Code of Student Conduct Disciplinary Procedures, McGill University

Bringing Ideas Togheter

I don’t expect you to do original research. Instead, I expect you to read about the research of others, and to bring together their ideas in such a way that makes sense to you and will make sense to me. Therefore, it’s essential for you to cite your sources in any research paper you write. … So don’t feel you need to hide the fact that you’re drawing from one of your sources. That’s what it’s all about.

Professor Bill Taylor’s A Letter to My Students

Why use references?

- It enables the interested reader to go further into the subject by accessing the references.
- It allows the reader to check their own interpretation of the idea or data source.
- It tells the reader the depth and breadth of materials accessed.

Cheating

- in the course of an examination, obtaining or attempting to obtain information from another student or unauthorized source or giving or attempting to give information to another student or possessing, using or attempting to use any unauthorized material;
- submitting in any course or program of study, without both the knowledge and approval of the person to whom it is submitted, all or a substantial portion of any academic writing, essay, thesis, research report, project or assignment for which you have previously obtained credit or which you are submitting in another course or program of study in the University or elsewhere;
- submitting in any course or program of study any academic writing, essay, thesis, research report, project or assignment containing a statement of fact know by the student to be false or a reference to a source which reference or source has been fabricated.

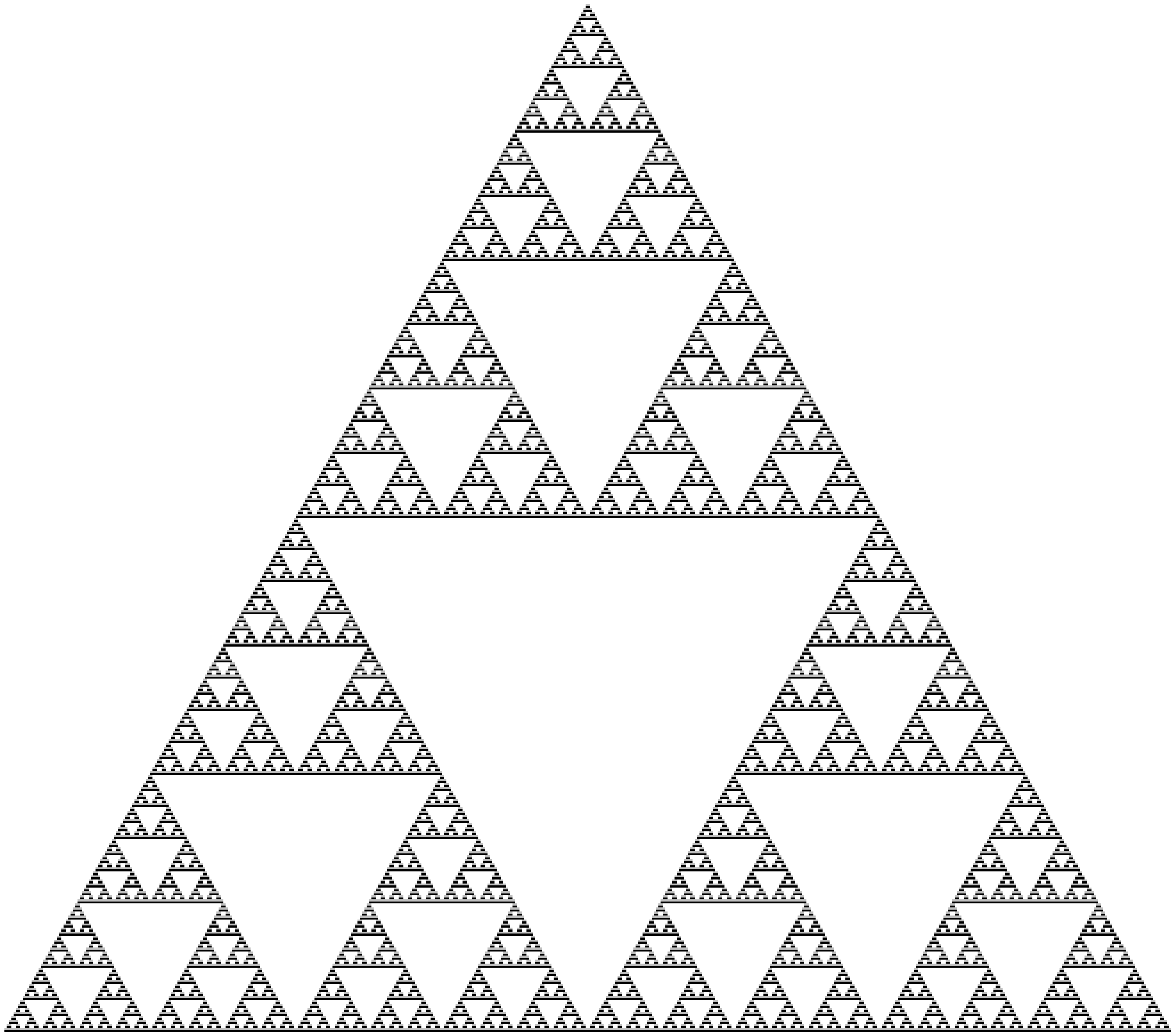
Article 16 (a,c,d), Code of Student Conduct and Disciplinary Procedures, McGill University

Resources

- www.mcgill.ca/integrity
- Library research workshops
- Library EndNote™ and Reference Manager™ reference citation sessions and software
- English for academic purposes through the center for English and French language
- Tutorial Services, Student Services, term paper tutors
- Doing Honest Work in College: How to Prepare Citations, Avoid Plagiarism and Achieve Real Academic Success by C. Lipson (2004), University of Chicago Press

Source : ACADEMIC INTEGRITY

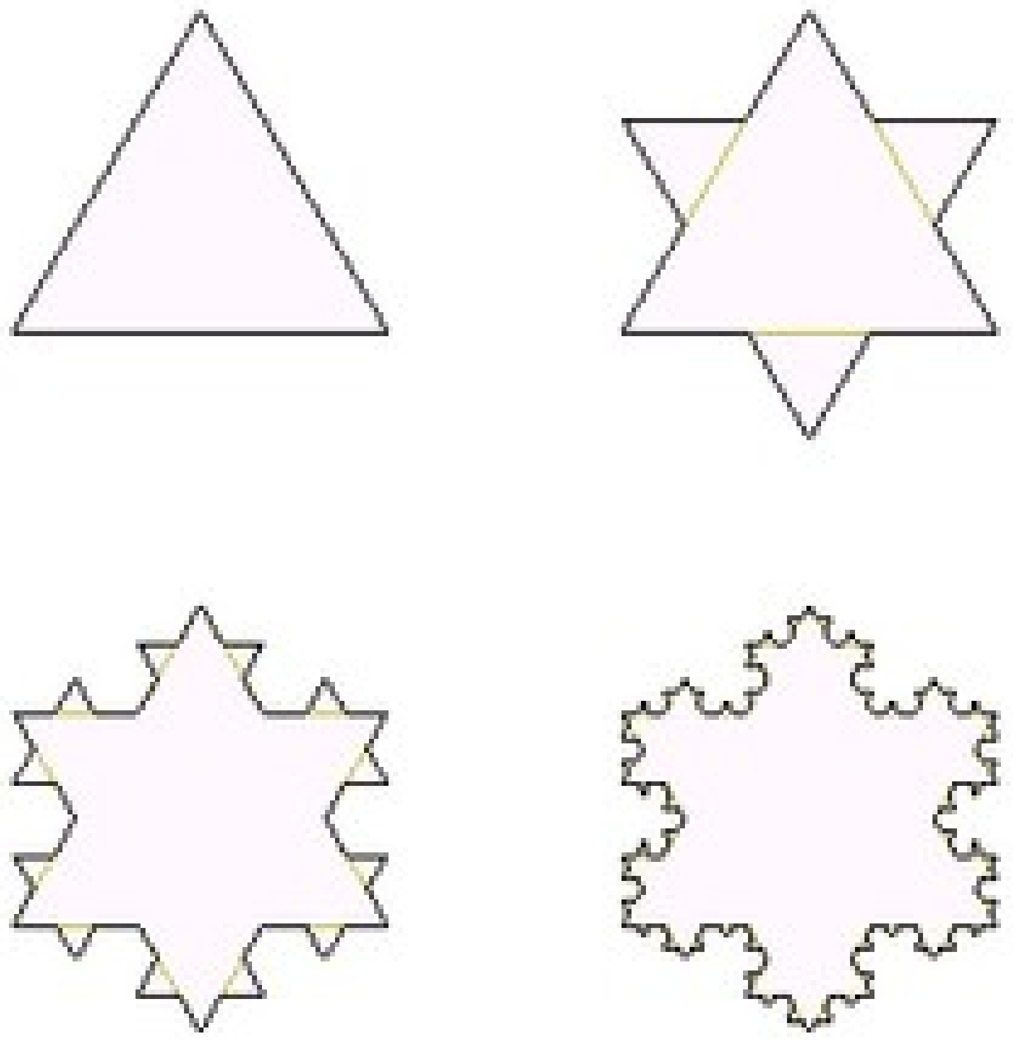
The Sierpinski Triangle



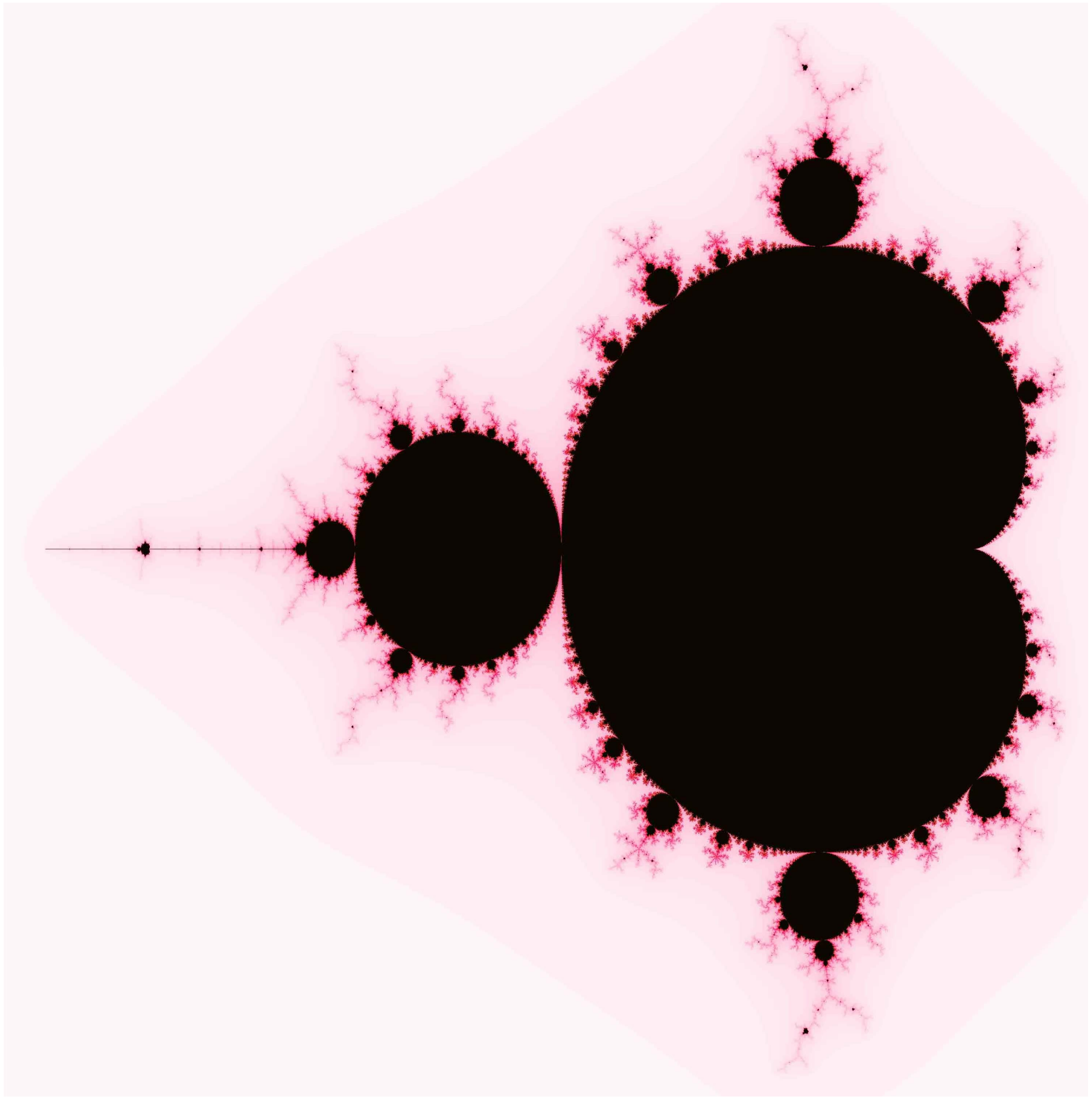
Introduction to Fractal

- Fractals are geometric figures, just like rectangles, circles and squares, but fractals have special properties that those figures do not have.
  - Self-similarity
  - Fractional dimension
  - Formation by iteration

Koch Snowflake



Mandelbrot



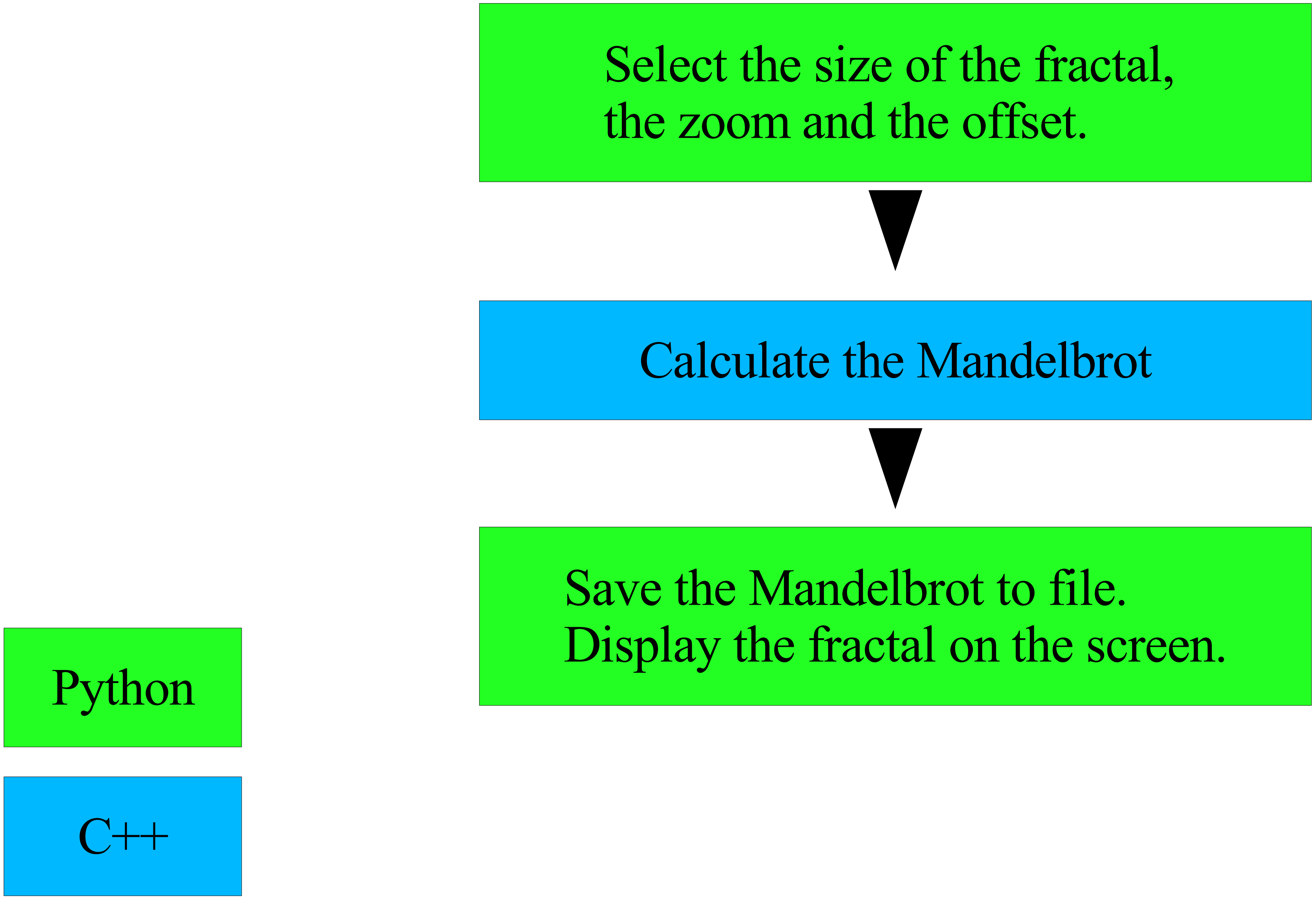
Pseudo code for Mandelbrot

```
For each pixel on the screen do:
{
  x = x0 = x co-ordinate of pixel (x image – width/2) / (width/2)
  y = y0 = y co-ordinate of pixel (y image – height/2) / (height/2)

  x2 = x*x
  y2 = y*y

  iteration = 0
  maxiteration = 1000
```

Pseudo code for Mandelbrot (cont.)

```
  while ( x2 + y2 < (2*2)  AND  iteration < maxiteration ) {

    y = 2*x*y + y0
    x = x2 - y2 + x0

    x2 = x*x
    y2 = y*y

    iteration = iteration + 1
  }

  if ( iteration == maxiteration )
    colour = black
  else
    colour = iteration
}
```

Assignment 3



🟩 Select the size of the fractal, the zoom and the offset.

🟦 Calculate the Mandelbrot

🟩 Save the Mandelbrot to file. Display the fractal on the screen.

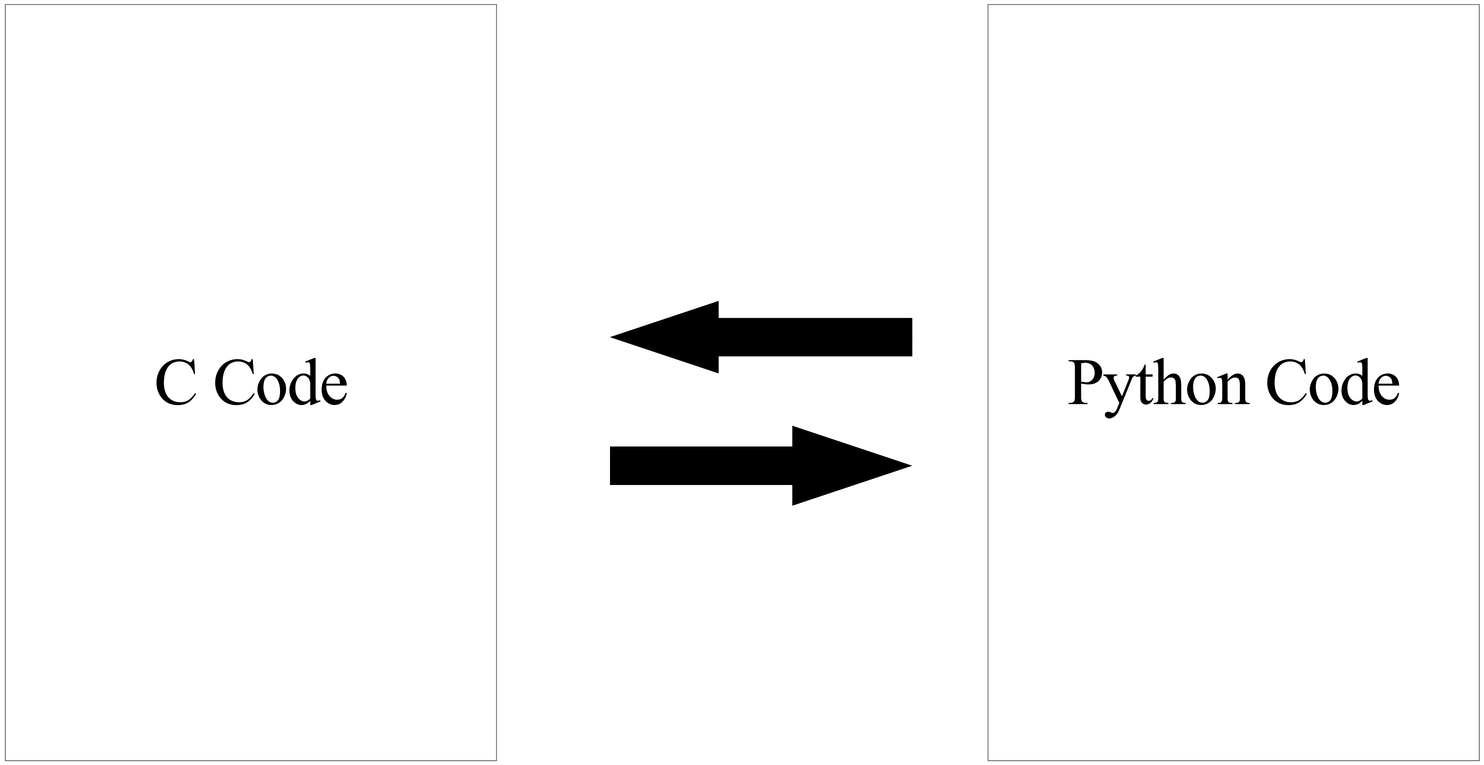
🟩 Python

🟦 C++

Translation / Scale

- What do you need to change in the algorithm ...
    - to zoom in on the fractal
    - to focus on a specific part of the fractal

C / Python API



Why?

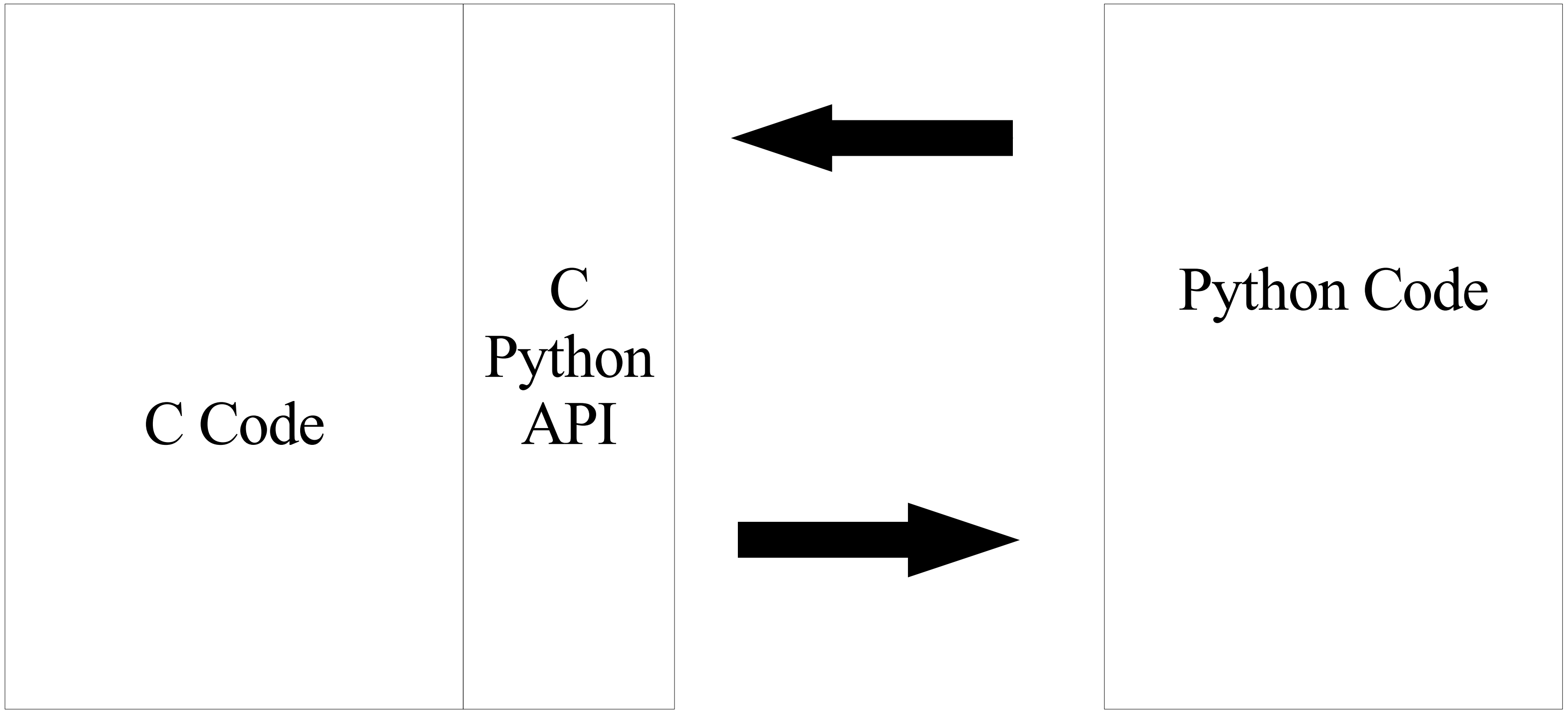
Why?

- C code is faster to execute.
- C is really good at manipulating memory and binary data.

- Python is Object Oriented.
- Python is easy to use to build prototypes.
- Python has many libraries and built-in data structures.

Translation through API

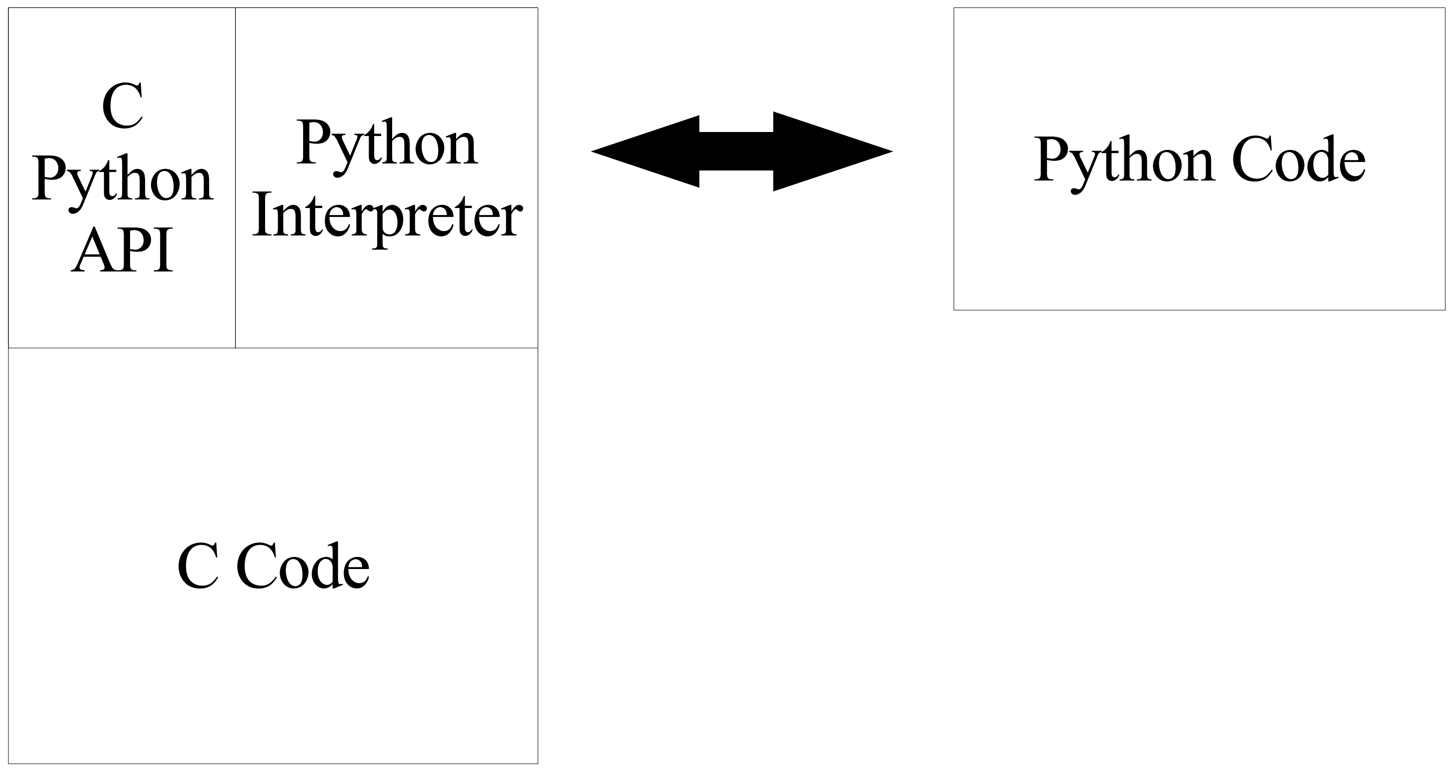
- Python code calls the C module just has if it were regular python code.
  - The python code doesn't know the module is written in C.
  - The python API takes care of the translation.



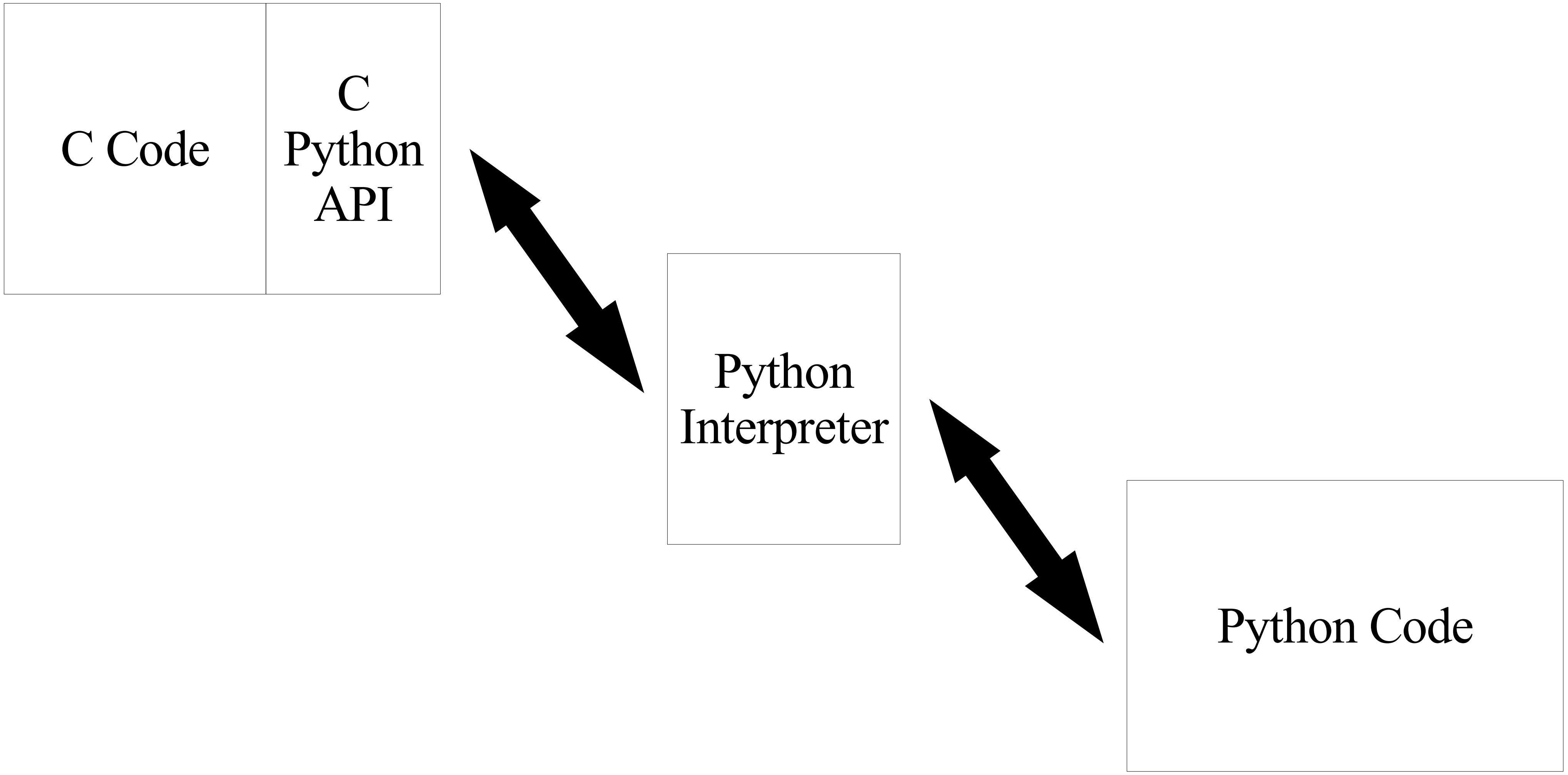
Implementation

- There are two ways to link C and Python code:
  - You can build your C code a as python module.
  - You can embedded a python interpreter in your application.
- Good reading material on this:
  - Extending Python with C or C++
    - http://docs.python.org/ext/intro.html
  - Python/C API Reference Manual
    - http://docs.python.org/api/api.html

Embedded Solution



Module Solution



Challenges

- Python is Object-Oriented, C is not.
- Python has many data types that C doesn't
  - Strings, lists, dictionaries, tuples, etc
- Python does automated memory management, C doesn't.

C/Python API

- The C/Python API is declare in <Python.h>.
- To compile your library, you'll also need to include the Python libraries.
    - We will look at the needed compile options latter.
- All the functions and data types used by API has the “Py” prefix.

Translation Exercise

Data types problems

- To use Python data types in C, we to wrap them in a C structure.
- Since everything in Python is an object, we need mostly need a structure for these objects : PyObject *
- We can then manipulate objects using special functions.

Reference Counts

- As already mentioned, Python does its own memory management.
- An object is considered alive as long as at least one other object is pointing to it.
  - When no objects are pointing to an object, it can be garbage collected.
- To determine how many objects are pointing to an object, Python uses reference counting.
  - An object's reference count is the number of other objects pointing to it.
  - When an object's reference count reaches zero, it can be garbage collected.
- Python's garbage collector is smart, it can detected cyclic dependencies.

Managing the Reference Count

- Two macros, Py_INCREF(x) and Py_DECREF(x), allows the incrementing and decrementing of the reference count.
  - Py_DECREF() also frees the object when the count reaches zero.
  - Forgetting to dispose of an owned reference creates a memory leak.
- You only need to do this if you create your own data type.

Integer

- The following functions allow you to manipulate integer objects:
  - **int PyInt_Check(PyObject *o)** : Return true if o is of type PyInt_Type or a subtype of PyInt_Type.
  - **PyObject* PyInt_FromString(char *str, char **pend, int base)** : Return a new PyIntObject or PyLongObject based on the string value in str, which is interpreted according to the radix in base.
  - **PyObject* PyInt_FromLong(long ival)** : Create a new integer object with a value of ival.
  - **long PyInt_AsLong(PyObject *io)** : Will first attempt to cast the object to a PyIntObject, if it is not already one, and then return its value.

Float

- The following functions allow you to manipulate float objects:
  - **int PyFloat_Check(PyObject *p)** : Return true if its argument is a PyFloatObject or a subtype of PyFloatObject.
  - **PyObject* PyFloat_FromString(PyObject *str, char **pend)** : Create a PyFloatObject object based on the string value in str, or NULL on failure.
  - **PyObject* PyFloat_FromDouble(double v)** : Create a PyFloatObject object from v, or NULL on failure.
  - **double PyFloat_AsDouble(PyObject *pyfloat)** : Return a C double representation of the contents of pyfloat.

String

- The following functions allow you to manipulate float objects:
  - **int PyString_Check(PyObject *o)** : Return true if the object o is a string object or an instance of a subtype of the string type.
  - **PyObject* PyString_FromString(const char *v)** : Return a new string object with the value v on success, and NULL on failure.
  - **PyObject* PyString_FromStringAndSize(const char *v, Py_ssize_t len)** : Return a new string object with the value v and length len on success, and NULL on failure. If v is NULL, the contents of the string are uninitialized.
  - **PyObject* PyString_FromFormat(const char *format, ...)** : Take a C printf()-style format string and a variable number of arguments, calculate the size of the resulting Python string and return a string with the values formatted into it.
  - **Py_ssize_t PyString_Size(PyObject *string)** :Return the length of the string in string object string.
  - **char* PyString_AsString(PyObject *string)** : Return a NUL-terminated representation of the contents of string.

Booleans

- The following functions allow you to manipulate boolean objects:
  - **int PyBool_Check(PyObject *o)** : Return true if o is of type PyBool_Type.
  - **PyObject* Py_False** : The Python False object. This object has no methods.
  - **PyObject* Py_True** : The Python True object. This object has no methods.
  - **PyObject* PyBool_FromLong(long v)** : Return a new reference to Py_True or Py_False depending on the truth value of v.
- Note that the Python API considers 0 to be false and 1 to be true.

Tuples

- The following functions allow you to manipulate tuples:
  - **int PyTuple_Check(PyObject *p)** : Return true if p is a tuple object or an instance of a subtype of the tuple type.
  - **PyObject* PyTuple_New(Py_ssize_t len)** : Return a new tuple object of size len, or NULL on failure.
  - **int PyTuple_Size(PyObject *p)** : Take a pointer to a tuple object, and return the size of that tuple.
  - **PyObject* PyTuple_GetItem(PyObject *p, Py_ssize_t pos)** : Return the object at position pos in the tuple pointed to by p.
  - **int PyTuple_SetItem(PyObject *p, Py_ssize_t pos, PyObject *o)** : Insert a reference to object o at position pos of the tuple pointed to by p.

Lists

- The following functions allow you to manipulate lists:
  - **int PyList_Check(PyObject *p)** : Return true if p is a list object or an instance of a subtype of the list type.
  - **PyObject* PyList_New(Py_ssize_t len)** : Return a new list of length len on success, or NULL on failure.
  - **Py_ssize_t PyList_Size(PyObject *list)** : Return the length of the list object in list; this is equivalent to "len(list)" on a list object.
  - **PyObject* PyList_GetItem(PyObject *list, Py_ssize_t index)** : Return the object at position pos in the list pointed to by p.

Lists

- The following functions allow you to manipulate lists:
  - **int PyList_SetItem(PyObject *list, Py_ssize_t index, PyObject *item)** : Set the item at index index in list to item.
  - **int PyList_Insert(PyObject *list, Py_ssize_t index, PyObject *item)** Insert the item item into list list in front of index index.
  - **int PyList_Append(PyObject *list, PyObject *item)** : Append the object item at the end of list list.
  - int PyList_Sort(PyObject *list) : Sort the items of list in place.
  - **PyObject* PyList_AsTuple(PyObject *list)** : Return a new tuple object containing the contents of list; equivalent to "tuple(list)".