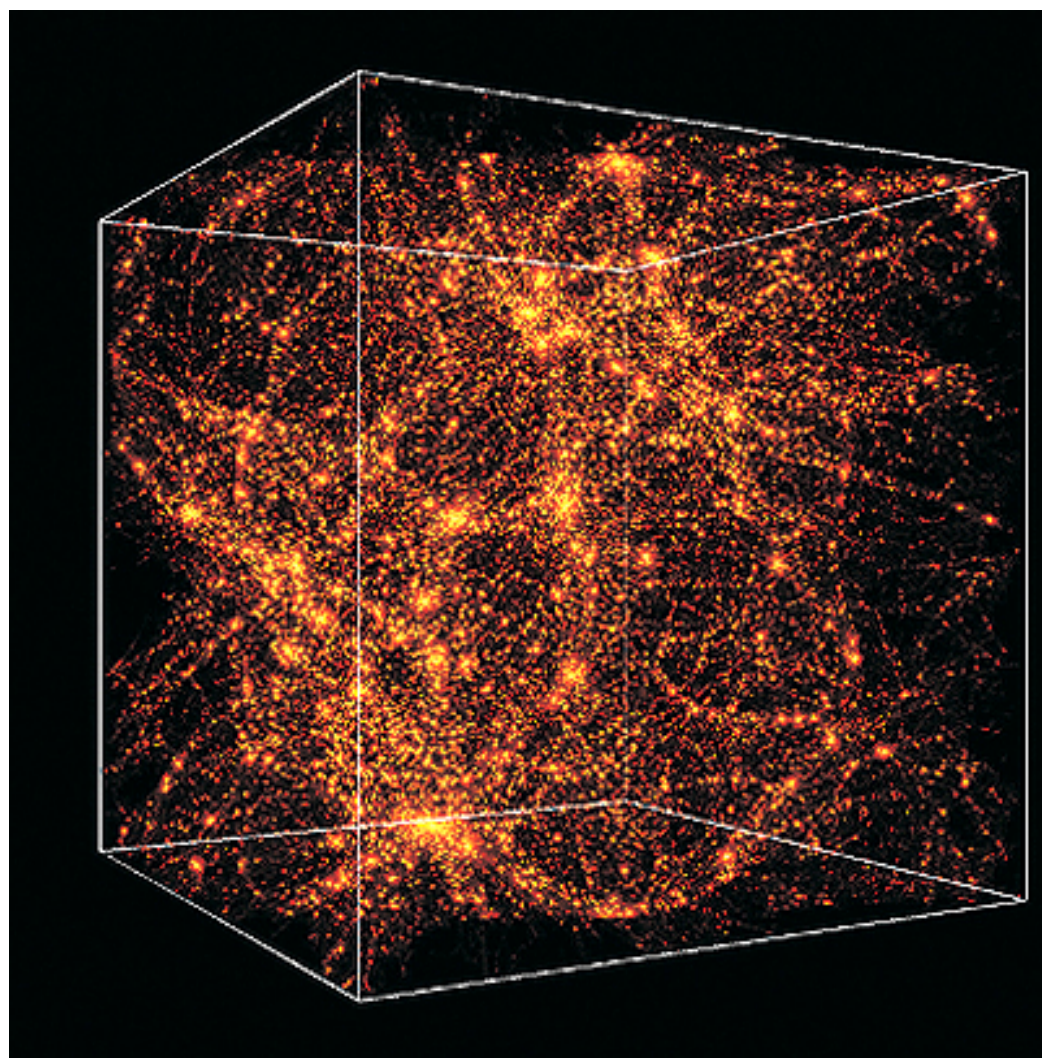


# Course Introduction

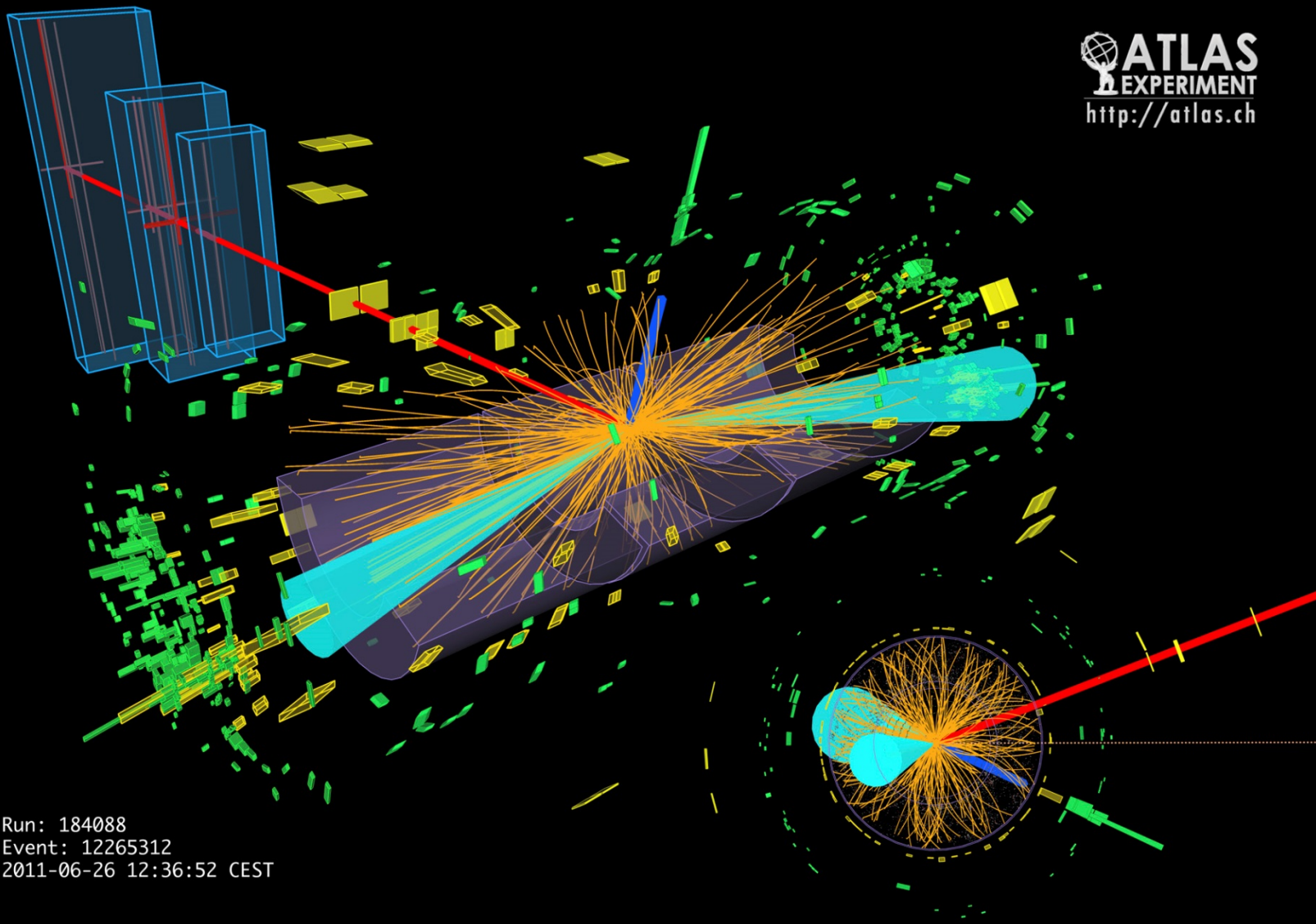
Computational Physics

3 September 2014

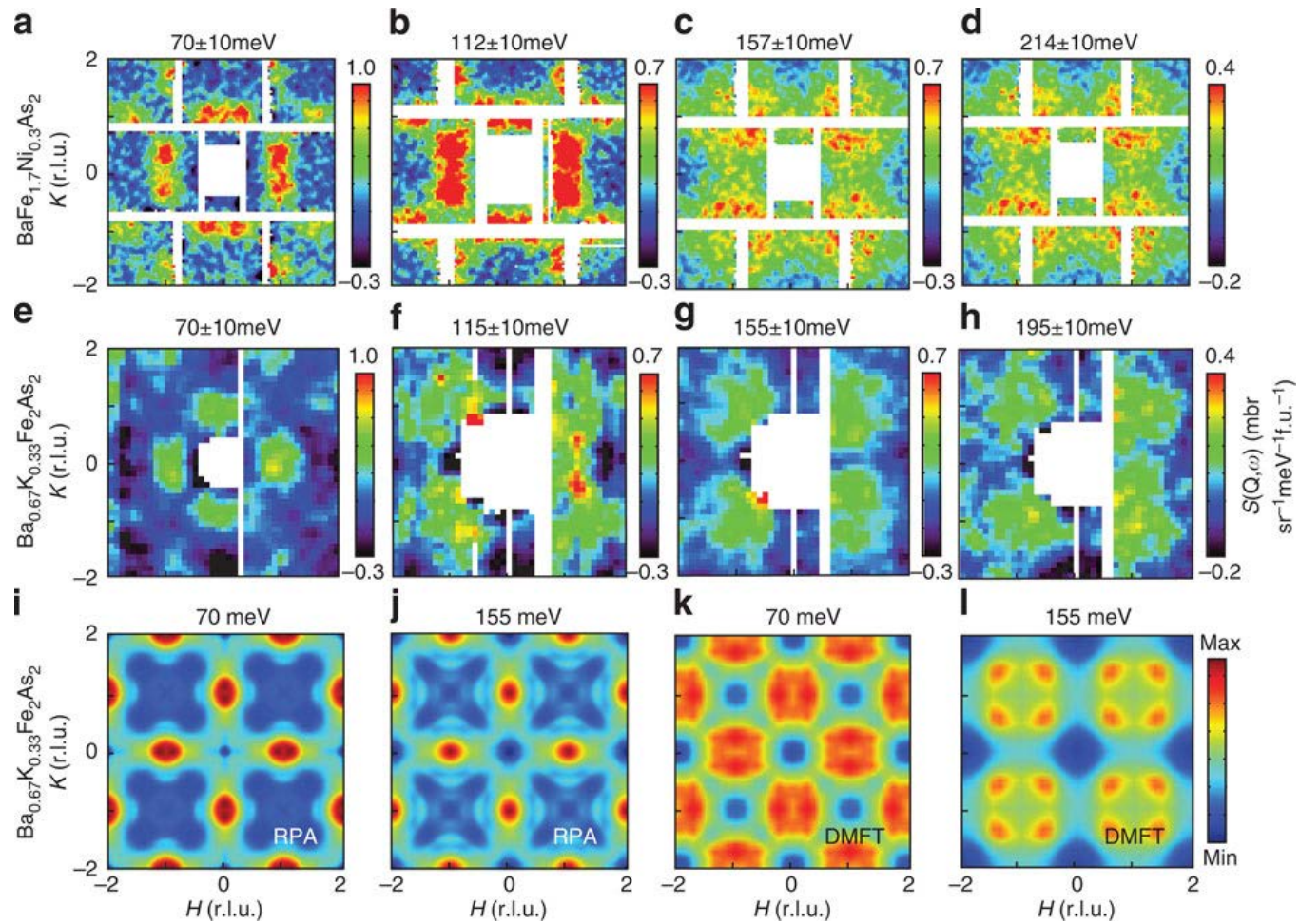




Run: 184088  
Event: 12265312  
2011-06-26 12:36:52 CEST







From “Doping dependence of spin excitations and its correlations with high-temperature superconductivity in iron pnictides” Nature Communications, 4, 2874, 2013

# Why are we using Python?

# Why are we using Python?

because it's free and it always will be free.

# Today's Activities

- Course Description
- Course Information
  - Instructors
  - Moodle
  - Other Administrative Items
- Getting started with computers
- Getting started with Python



# Course Description

- Introduction to computational methods for simulating physical systems and solving problems numerically.
- Sophomore level
- Assumes no previous experience with programming or computational methods
- “Lab Course”

# Course Description

## *The Learning Curve*

- We need to learn many new things
  - Proper use of the Lab computers
    - Linux
    - X-windows
    - Text editors and X windows
    - Report Writing
  - Using a computing language (Python)
    - The Python environment
    - Arrays and Matrices
    - Scripting and building programs
    - Plotting and displaying data

# Course Description

## *The Learning Curve*

- Once we have this under our belts we get to
  - Easily perform calculations that are very difficult to do analytically
  - Simulate physical systems (including random ones!)
  - Solve ordinary differential equations
  - Solve partial differential equations

# Course Information

## *Instructors*

- Grant Wilson
  - 644 LGRT-B – 545-0460
  - Email: [wilson@astro.umass.edu](mailto:wilson@astro.umass.edu)
- Teaching Assistants:
  - Yuping Tang (yupingt@astro.umass.edu)
  - Seunghwan Lim (shlim1206@gmail.com)
- Computer Lab Supervisor: Dan Popowich
  - Report problems to the TAs

# Textbook



# Course Information

- Everything about the course is posted on Moodle
  - [Moodle.umass.edu](http://Moodle.umass.edu)

# How this class works

- Every class begins with a short **Quiz** on the previously assigned readings/viewings
- During most classes I will provide a short lecture to describe some technique and its uses.
- During every class you will work on **Exercises**
- Every few weeks I will assign a **Project**
- The final exam will be a short **Oral exam**



# The Projects

- Every few weeks I will assign a different project for you to do. Projects consist of
  - Calculation of a real physics problem
  - Exploring the behavior of a physical system
  - Answering questions posed about the system
  - Writing a Project Report

# Some Words to the Wise

- Attend Class
  - Class attendance is mandatory.
  - There is not always a lecture but we always do something important.
  - Group participation is a part of your assessment.
- Don't fall behind
  - Exercises and Projects take longer than you think.
  - Deadlines are strict.
- Learn Python
  - Get comfortable with Python early on and life will be much much easier down the road.

# Tentative Schedule

- Python
- Interpolation; Numerical Integration
- Random Numbers and Simulation
- Model Fitting
- Differential Equations
  - Ordinary
  - Partial

# Getting Started with the Computers

# General Lab Guidelines

- See the document on your desktop for rules.
- Lab is open 24/7 except when other classes are meeting.
- All computers are the same.
- Ubuntu is our operating system.
- Report problems to me or the TAs

# Lab Do's

- Use the lab!
- Respect the equipment.
- You are responsible for your own backups.
- Move computers in their sliders gently.
- Log off when you are done.
- Last one out, turn off lights and lock door.

# Lab Don'ts

- Don't share combination of door lock.
- Don't unplug any cable or power cord.
- No public internet connections in the lab. Do not connect your personal computer to the lab network!
- No food or drink in the lab!



# Big Rule

- No non-class use of the computers during class time. This includes:
  - email
  - facebook
  - etc.
- Why not?
  - it's distracting
  - it's disrespectful

# Login/Logout

- Login Procedure
  - Login with your assigned username
  - Enter your assigned password
    - See instructions on desktop to change password.
- Logout Procedure
  - Click mouse on the menu bar at top of screen
  - Select “Log Out”
  - Select “Log Out” when dialog box appears.

# Set up your environment

- Open “terminal” window  
(you are in your “home” directory)
  - › *mkdir class01*
  - › *cd class01*
- Click on the terminal window again.
  - › *ipython*

# Let's write our first Python program!

- The “command line” way:
  - In ipython window type:
    - *print “Hello World”*
  - Now try:
    - *x = “Hello Cruel World”*
    - *print x*
  - Now try:
    - *x = Hello Cruel World*

# Let's write our first Python program!

Now with an editor so we can save our work.

- Double-click: home->class01
- Right click and create a new leafpad document called `hello_world.py`
- Turn on line numbering in "Options"
- Open `hello_world.py` and type:  

```
print "Hello Cruel World"  
x = "... hello? ..."  
print x
```
- Save your work by pressing *Ctrl+s*.
- In the terminal window (with ipython running) type:  

```
run hello_world.py
```

# So how do you learn Python?

- The web is rife with examples.
- I still use Google every time I write a Python program.
- Example:
  - Write a Python program to calculate the number of characters in your full name, not counting the white spaces between your three names.

# Python Classes and Objects


- Everything in Python is a “thing”
  - things have characteristics (attributes)
  - some things can do things
- The superset of what something is is its class.
  - ex. What I drove to work in.
    - Vehicle
      - automobile
        - » Honda Fit
          - MY Honda Fit



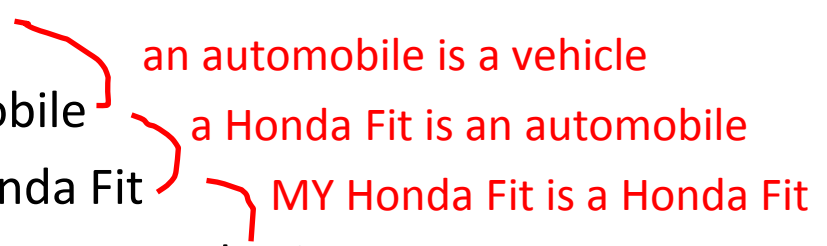
# Python Classes and Objects

- Everything in Python is a “thing”
  - things have characteristics (attributes)
  - some things can do things
- The superset of what something is is its class.
  - ex. What I drove to work in.
    - Vehicle
      - automobile
        - » Honda Fit
          - MY Honda Fit

This is not a class. This is a particular “instance” of a class which is called an “object”



# Python Classes and Objects

- Everything in Python is a “thing”
    - things have characteristics (attributes)
    - some things can do things
  - The superset of what something is is its class.
    - ex. What I drove to work in.
      - Vehicle
        - automobile
          - » Honda Fit
            - MY Honda Fit
- 
- The diagram illustrates a class hierarchy for vehicles. It starts with 'Vehicle' at the top, which is the superclass for 'automobile'. A red bracket connects 'Vehicle' to 'automobile' with the annotation 'an automobile is a vehicle'. 'automobile' is the superclass for 'Honda Fit', with a red bracket and the annotation 'a Honda Fit is an automobile'. Finally, 'Honda Fit' is the superclass for 'MY Honda Fit', with a red bracket and the annotation 'MY Honda Fit is a Honda Fit'.

# Python Classes and Objects

- Creation of things can be implicit or explicit. It's your job to keep track of what is being created.
- Once created, objects have properties (nouns) and their own methods (verbs).
- For Example
  - MY Honda Fit = Honda Fit()
    - print MY Honda Fit.wheel
    - print MY Honda Fit.seat
    - MY Honda Fit.drive()
    - My Honda Fit.turn()

# Python Classes and Objects

- Creation of things can be implicit or explicit. It's your job to keep track of what is being created.
- Once created, objects have properties (nouns) and their own methods (verbs).
- For Example
  - MY Honda Fit = Honda Fit()
    - print MY Honda Fit.wheel ← inspect the wheel of my car (note that the wheel is another thing so it will have its own set of properties eg, wheel.pressure and wheel.tread.)
    - print MY Honda Fit.seat
    - MY Honda Fit.drive()
    - My Honda Fit.turn()

# Python Classes and Objects

- Creation of things can be implicit or explicit. It's your job to keep track of what is being created.
- Once created, objects have properties (nouns) and their own methods (verbs).
- For Example
  - MY Honda Fit = Honda Fit()
    - print MY Honda Fit.wheel
    - print MY Honda Fit.seat
    - MY Honda Fit.drive()
    - MY Honda Fit.turn() ← This is called a “method” and it makes the object do something. The parens are required! After this method is executed, the values of some of the other properties of MY Honda Fit will have changed.

# Python Classes and Objects

- Let's try this with the “float” class.
- Open ipython
- Create an floating point number x
  - `x = 36.25`
- Check out its properties and methods
  - `x.<press tab>`
- Now create the complex number y
  - `y = 4.0 + 6.4j`
- Check out its properties and methods
  - `y.<press tab>`

# Python Functions

- A function is a self-contained piece of code.
  - Some functions return calculated quantities.
  - Some functions do things (like make a plot).
  - Some functions take inputs.
- Example: a function to calculate the Fibonacci series up to n

```
def fib(n): # write Fibonacci series up to n
    "Print a Fibonacci series up to n"
    a, b = 0, 1
    while b < n:
        print b,
        a, b = b, a+b
```

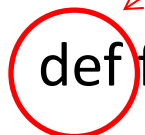


# Python Functions

```
def fib(n):  
    a = 0  
    b = 1  
    while b < n:  
        print b,  
        a0 = a  
        a = b  
        b = a0+b
```

# Python Functions

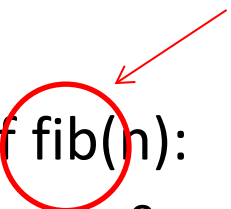
“def” says that we are defining a function



```
def fib(n):  
    a = 0  
    b = 1  
    while b < n:  
        print b,  
        a0 = a  
        a = b  
        b = a0+b
```

# Python Functions

“fib” is the function name



```
def fib(n):  
    a = 0  
    b = 1  
    while b < n:  
        print b,  
        a0 = a  
        a = b  
        b = a0+b
```

# Python Functions

the inputs to the function are in parentheses, you can have as many inputs as you want. Ex. (n,m,x,y,z...)

```
def fib(n):
```

```
    a = 0
```

```
    b = 1
```

```
    while b < n:
```

```
        print b,
```

```
        a0 = a
```

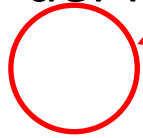
```
        a = b
```

```
        b = a0+b
```

# Python Functions

```
def fib(n):  
    a = 0  
    b = 1  
    while b < n:  
        print b,  
        a0 = a  
        a = b  
        b = a0+b
```

The function body is always tabbed one tab in.  
This is how Python knows you're still in the function.



# Python Functions

```
def fib(n):
```

```
    a = 0
```

```
    b = 1
```

```
    while b < n:
```

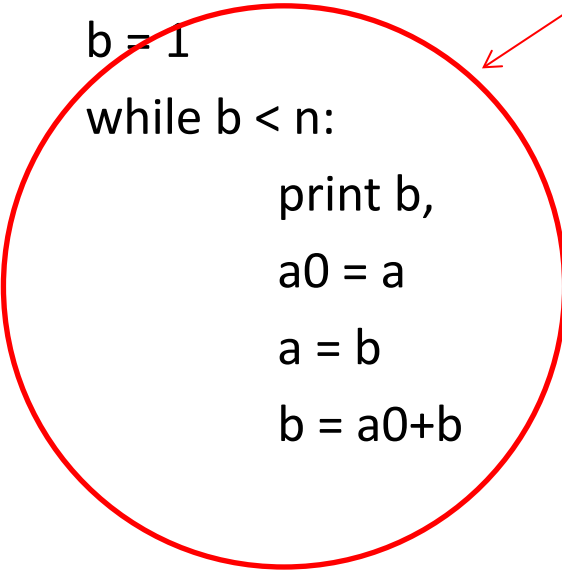
```
        print b,
```

```
        a0 = a
```

```
        a = b
```

```
        b = a0+b
```

This is a “while” loop. We’ll talk about these later. In this example, everything that is tabbed is looped over until  $b \geq n$



# Python Functions

```
def fib(n):
```

```
    a = 0
```

```
    b = 1
```

```
    while b < n:
```

```
        print b,
```

```
        a0 = a
```

```
        a = b
```

```
        b = a0+b
```

Always end a function with a blank line.



# Exercise

- Write a simple function that takes your first and last names as two separate inputs and then prints your whole name to the screen.



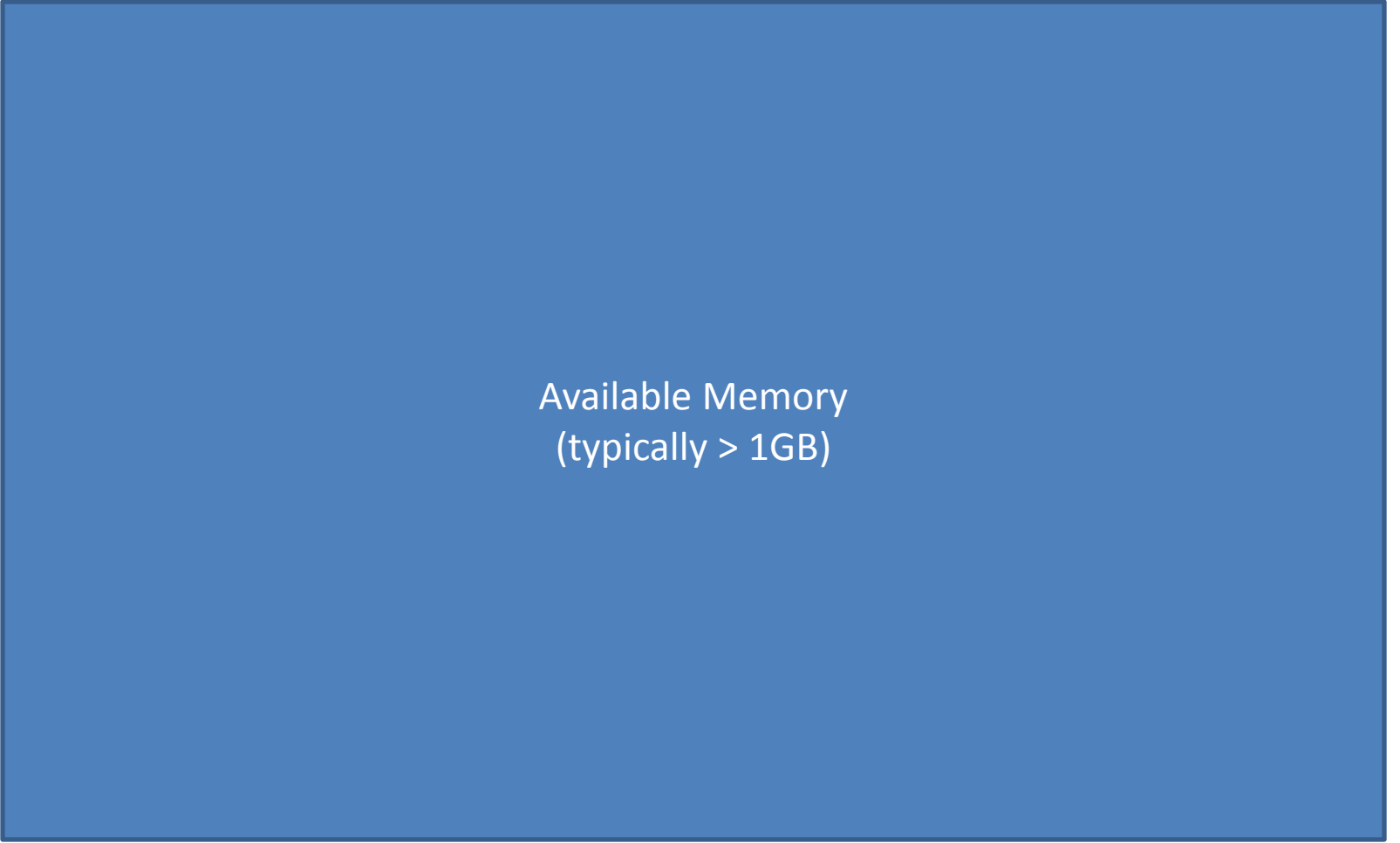
# Python Modules and Packages

- Modules are collections of functions that exist outside of your main program body.
  - Ex., the “os” module contains functions that interact with the underlying operating system.
  - When you start ipython, only a bare-bones set of modules are available.
- Modules must be imported before they can be used. Try this:
  - *import os*
  - *print os.getcwd()*
- Now quit ipython and start it again, try “*print os.getcwd()*” without importing os to see what happens.
- By convention, all imports should happen at the top of a written Python program.
- Packages are collections of Modules

# Python Modules and Packages

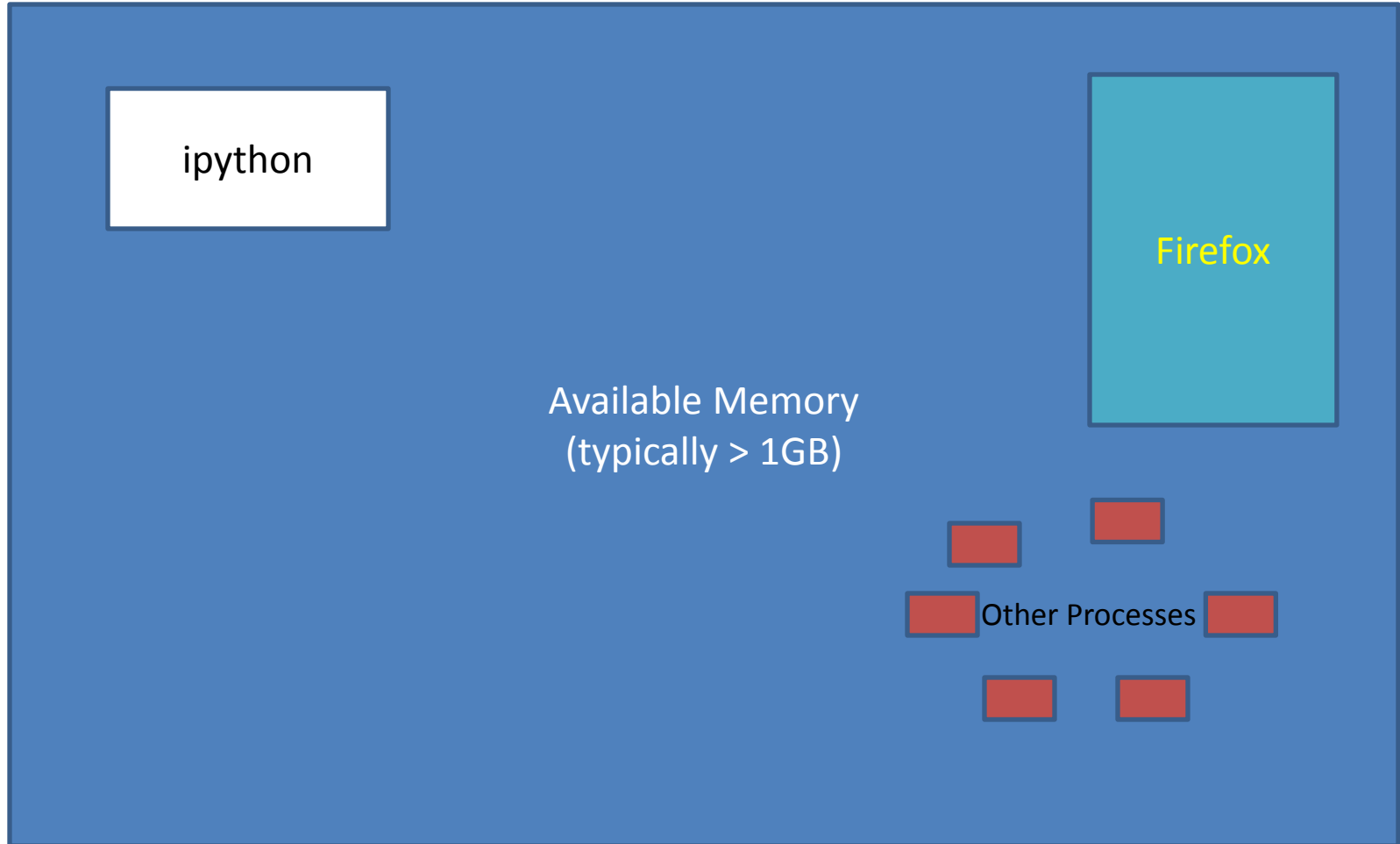
- The most common Packages that we will be using are:
  - NumPy – the fundamental package for scientific computing (see Friday's class)
  - Matplotlib – the most widely used plotting package (see next Monday's class)

# How Computer Memory Works

A large blue rectangle with a thin dark blue border, representing a memory pool. Centered within this rectangle is white text.

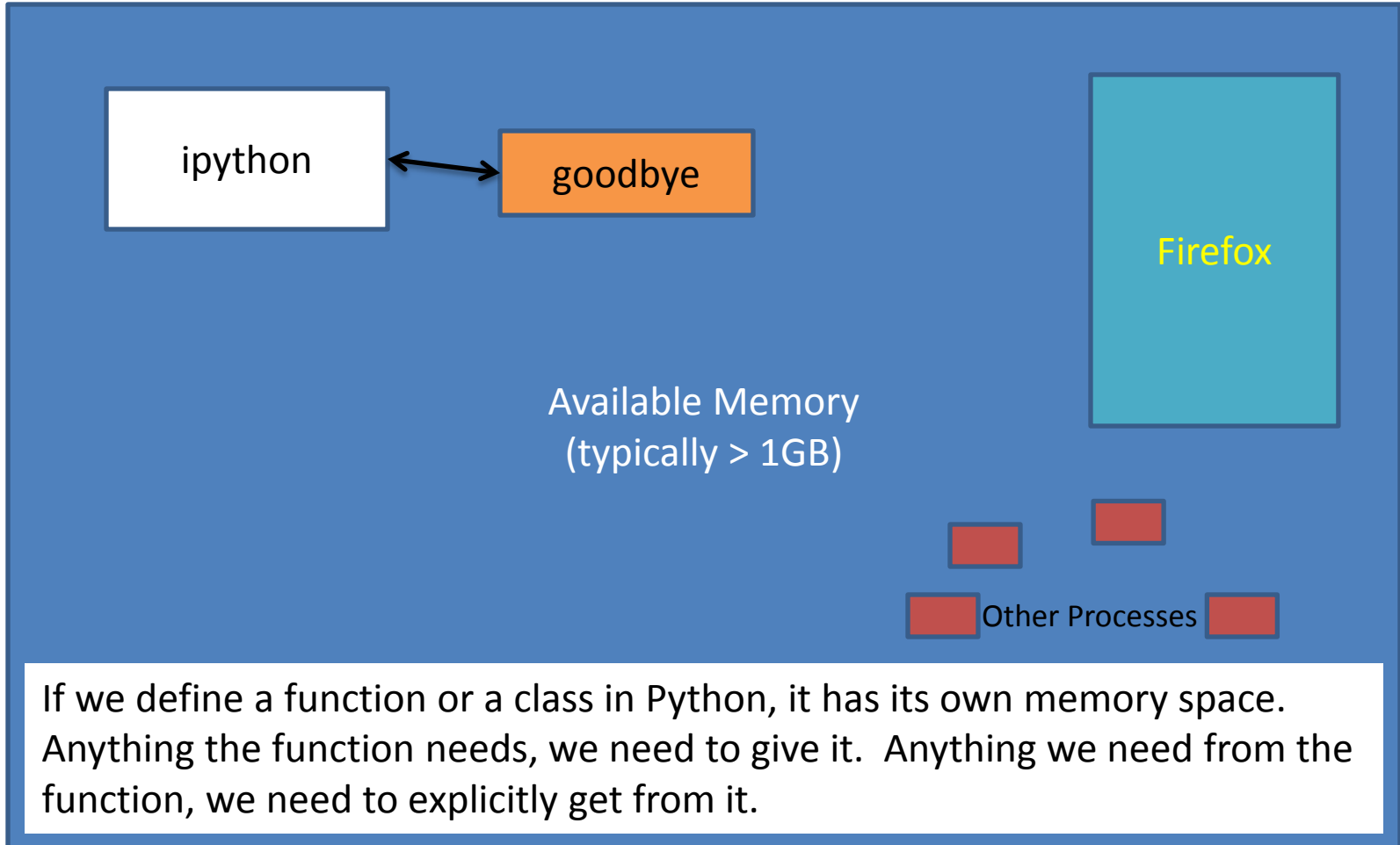
Available Memory  
(typically > 1GB)

# How Computer Memory Works

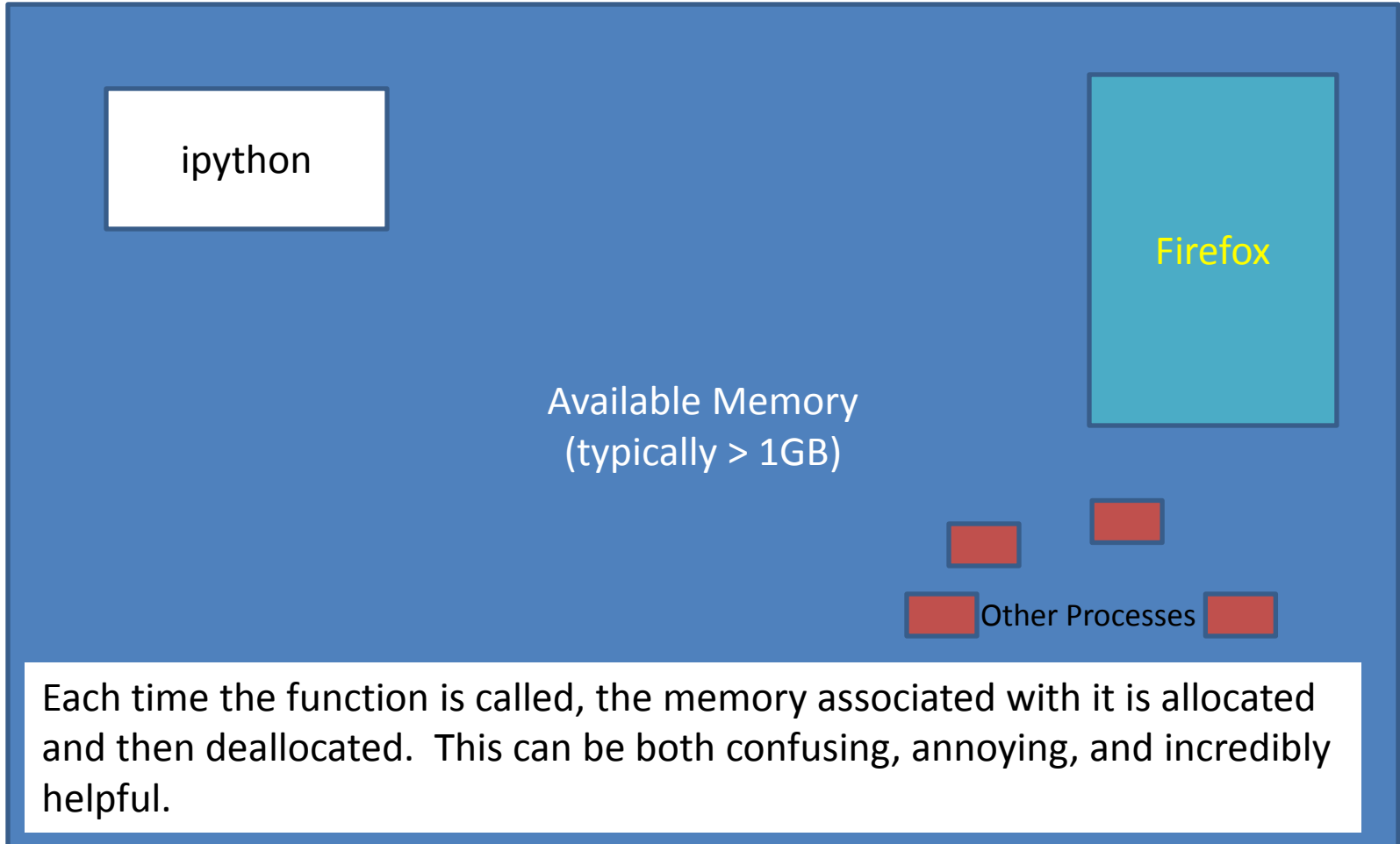


Programs “allocate” their own piece of memory from the available pool.

# How Computer Memory Works



# How Computer Memory Works



# Get Python Running on your Home Computer!

- A company called “Enthought” has a nice version that works on PCs and Macs.
  - see <https://store.enthought.com/#canopy-academic>
- This is free!