

# Human Computer Interaction

## Python and Kivy Laboratory

Prof. Andrew D. Bagdanov

Dipartimento di Ingegneria dell'Informazione  
Università degli Studi di Firenze  
`andrew.bagdanov AT unifi.it`

October 4, 2017

- 1 Overview
- 2 Installing Anaconda
- 3 Using Anaconda
- 4 Installing Kivy
- 5 Summary
- 6 Homework

# Overview

This will be a very informal laboratory. We are mostly interested in getting our Anaconda installations up and running, and I am here to help. Things we will do today:

- **Anaconda installation:** the first thing we will do is install an Anaconda distribution on each of our laptops.
- **Anaconda invocation:** next, I will show you some of the ways we can execute Python and Python programs using Anaconda.
- **Kivy installation:** after discussing the Anaconda package manager Conda, we will install Kivy.
- **Some Python and Kivy exercises:** now that we have working Python and Kivy installation (hopefully), we can work through some exercises in IPython notebook, IPython consoles, or the Spyder IDE.

# Installing Anaconda

- **Anaconda** is a Python distribution that includes more than 400 of the most popular Python packages for science, math, engineering, and data analysis.
- Though it targets the **scientific** programming community, we will use it because of some of its packaging features:
  - Anaconda has an excellent package manager called **Conda** which allows you to install and upgrade packages.
  - Anaconda has tools for building **virtual Python environments** that contain exactly the packages (and package versions) you need.
  - Anaconda's environments can be used to **isolate** a working environment from others.
- In particular, I have already verified that there are working **Kivy** packages (at least for Windows and Linux).

All installers, graphical and console, are available here:

<https://www.continuum.io/downloads>

## Linux

- There is a shell script installer for Linux systems.
- Download it and execute it:

```
bash Anaconda3-5.0.0.1-Linux-x86_64.sh
```

- **NOTE:** you do not need to run this with **sudo**.

## OSX

- **Should** be the same as for the Linux version.
- There is also a **graphical installer**, which might be more to your taste.

## Windows (ugh)

- Windows has **only** the graphical installer.
- Download it and execute it.

## Linux

- The Anaconda installation will ask you if you want to modify your `.bashrc` file to add the Anaconda directory to your `PATH`.
- If your `PATH` is configured correctly, you should be able to run `ipython` from a console and get an IPython console (check for the "Anaconda" build string).

## OSX

- **Presumably** the same as Linux (from console).
- Alternatively, there might be an "Anaconda" application that you can run that will put you in shell with the `PATH` setup correctly.

## Windows (ugh)

- In Windows, under the start menu (or whatever equivalent there is now), there will be an "Anaconda" application.
- If you run this, it will put you in a shell with the `PATH` set correctly; you should be able to run `ipython` from here.



## Using Anaconda

Try these now to verify that both work and that both are calling the correct Python interpreter (Anaconda).

## Very simple invocation

- The simplest way to invoke the Python interpreter is with the `python` command.
- This gives you an interactive console, but not a very full-featured one.

## Running the IPython console

- A better way to invoke Python is with the IPython interactive console.
- The `ipython` command will put you in an interactive console.
- The IPython console has a number of **magic commands** that do very useful things.

- Quick rundown of **magic commands**:
  - `ls`: lists files in current directory (globbing works)
  - `cd`: change current working directory
  - `run`: loads and executes a Python file
  - `who`: list objects in current session
  - `%pdb`: toggles automatic calling of PDB debugger on errors.
- Everything you type in the IPython console (e.g. **definitions**) takes place in the toplevel, “global” namespace (i.e. not in any module).
- You can input (nearly) any **valid Python expression** (even multi-line) at the IPython console.
- Console interaction is excellent when you want to poke around at things.
- Use the **Python debugger** (PDB) to see what’s going on.

## Scripts to play with right now

- Basic **execution** of programs: `basics.py`
- **Scripts** versus **modules** (best of both): `script.py`
- **Explicit** debugger invocation: `debugme.py`
- **Implicit** debugger invocation: `buggy.py`

## Things to remember

- PDB (`ipdb`) is a **powerful, console-based** debugging environment. It's worth familiarizing yourselves with it.
- **Implicit** invocation with `%pdb` gives you a way to figure out how to fix what went wrong.
- Think about **compartmentalizing** executable commands in a `__main__` guard so that you can **import** stuff from other places.

## Running IPython in notebook mode

- In addition to console mode (which we just experimented with), IPython can be started in **notebook mode**.
- When you invoke `jupyter-notebook`, IPython starts an interactive webservice and opens a browser window.
- From this interface you can select existing notebooks to load, or create new ones.
- When you create or load a notebook, IPython starts a new **IPython kernel** (basically an IPython session that will be used for evaluation of Python code).
- These notebooks support a **cell-based** user interface.
- There are several types of cells supported, including code and **markdown** (useful for textual documentation and annotations).

- Open a shell (on Windows an Anaconda console) and change to a temporary working directory.
- Start up a notebook with the `jupyter notebook` command.
- **Note:** if the notebook starts up OK, but a new browser window isn't created, you can point your browser to:

<http://localhost:8888>

to access the notebook interface.

- Quick rundown:
  - On the toplevel page is the interface for administering notebooks: there are controls for seeing what **kernels** are already running (top tab) and to create new files and notebooks ("new" menu at the top right).
  - Creating a new notebook will start a new kernel and open the notebook window in a new browser tab.
  - In the new notebook, go to the menu item Help --> User Interface Tour

## Overview

- In the toolbar, the Kernel menu has functions to interrupt or restart the kernel, and the Cell menu has functions to control the cell currently being edited.
- Many of these functions are duplicated in buttons directly on the toolbar.
- On cells: pressing **ENTER** goes to the next line (does not execute cell), pressing **Ctrl-ENTER** will **execute** the cell contents, and pressing **Shift-ENTER** will execute the current cell AND advance the the next cell (creating one if it didn't exist).
- You can insert new cells using the '+' button, and move cells up or down using the up- and down-arrows.

## Try it yourself

- If you run `jupyter notebook` in the directory containing the files from this lab, you will see a notebook already defined (the file with the `.ipynb` extension).
- Start up a kernel running this notebook

## A full-blown IDE

- Included in the Anaconda distribution is the Spyder IDE.
- Run it and change the working directory to where you put the files from this class.
- Now open one of the scripts (`script.py`, for example).
- Pressing the 'run' button will execute the script, showing the results in the console.
- This IDE seems to give a nice mix of IPython consoles and notebooks and editor.

## Other options

- There are other Python IDEs (like PyCharm), so feel free to experiment to see what you like.
- **My preference:** Emacs for editing, IPython consoles for running, IPython notebooks for didactic purposes and sharing.



## Installing Kivy

## Conda test drive

- The Anaconda package manager is called **Conda**.
- It nicely wraps all functionality for downloading and installing packages.
- It also manages **virtual Python environments** for you.
- **AND** it plays well with other Python package managers like pip and easy\_install.
- Life is just better with it.
- For more info, follow this tutorial for Conda:

<http://conda.pydata.org/docs/test-drive.html>

- The conda package manager has **many** features.
- Have a look at the Test Drive tutorial on the previous slide for much more info.
- For now, an overview of some useful commands:

<code>conda env list</code>	<i># List existing conda environments.</i>
<code>conda create --name flowers &lt;pkgs&gt;</code>	<i># Create a new virtual env.</i>
<code>source activate flowers</code>	<i># Activate virtual env.</i>
<code>source deactivate</code>	<i># Deactivate virtual env.</i>
<code>conda install &lt;pkgs&gt;</code>	<i># Install packages.</i>
<code>conda search --full-name python</code>	<i># Search for packages.</i>
<code>conda info</code>	<i># Info about Anaconda install.</i>
<code>conda list</code>	<i># List packages in current env.</i>
<code>pip search &lt;search terms&gt;</code>	<i># Search for a pip package.</i>
<code>pip install &lt;pkg&gt;</code>	<i># Install pip package.</i>

## Linux

- This works on Linux (Ubuntu 16.04), **should** work on OSX:

```
conda create -n kivy python=3 ipython jupyter \  
    cython pillow numpy matplotlib  
source activate kivy  
pip install Kivy Pygame
```

## OSX

- This might work on OSX:

```
conda create --name kivy python=3 ipython jupyter \  
    cython pillow numpy matplotlib  
source activate kivy  
USE_OSX_FRAMEWORKS=0 pip install -I --no-cache-dir --no-binary all kivy
```

## Windows

- First: try using the graphical installer for Anaconda.
- Search for a pre-packaged Kivy (1.9 or 1.10).
- Detailed Windows installation instructions:

<https://kivy.org/docs/installation/installation-windows.html>

## OSX

- Detailed OSX installation instructions:

<https://kivy.org/docs/installation/installation-osx.html>

## Simple test

```
from kivy.app import App
from kivy.uix.button import Button

class TestApp(App):
    def build(self):
        return Button(text='Hello World')

TestApp().run()
```

## Try it out

- Put this code in a .py file.
- And then run it from IPython, or Spyder.
- Try running it from an IPython notebook.

## Summary

- This lab should have given you the basic tools you need to start working seriously with Python.
- At the very least you should be in a good position to now **learn** everything you might need to know.
- Some students asked about reference books on Python, but I don't know of any that I can personally recommend.
- But, this seems to be a well-curated list of resources:

<http://www.fullstackpython.com/best-python-resources.html>

- And this page seems like an excellent (and entertaining) reference:

<http://docs.python-guide.org/en/latest/>



# Homework

## Exercise 4.1: Kivy

Spend some time looking at the [Kivy Gallery](#) on the website. This will give you some ideas about what is possible with the Kivy toolkit. Look at the source code for some of the gallery examples to get a feel for what Kivy applications look like “under the hood.”

## Exercise 4.2: Anaconda environment management

Spend some time working through the [Conda Test Drive](#) (if you haven't already done so). Experiment with making Anaconda virtual environments with various dependencies. Experiment with activating, deactivating, and deleting environments.