Python Intro

Lesson Objectives

- Get a brief introduction to Python, it’s history, differences from other languages and use cases
- Install Python
- Learn about Python Integrated Development Environments (IDEs)
- Learn, Install and Use Jupyter Notebooks

Introduction

About Python

- Python is a general-purpose, object-oriented, dynamic programming language
- Python is also **language & environment** for data science computing and graphics
- Open source
- Rich ecosystem (lots of libraries)
- Great for modeling, machine learning, ad-hoc analytics
- Used by app developers, web developers, but also popular among scientists and now data scientists
- Python Website - https://www.python.org/

Why Python

- Comprehensive
  - Pretty much any development can be done in Python
- State-of-the-art graphics capabilities
  - Because picture IS worth a thousands words
- Designed for interactive analysis
  - Most analysis is done this way
  - No time consuming edit / compile / run cycle
  - Can support scripting too
- Open source
  - Commercial packages costs thousands

Why Python

- Python can import from variety of formats (csv, excel, db, …)
- Python is extensible
  - Thousands of libraries in PyPI (open source)
- Python usually gets ‘bleeding edge’ routines before other commercial packages!!
  - Power of open source
- Free IDEs (Spyder, Pycharm) are available
  - Easy to use / program
- Runs on multiple platforms (Mac, Windows, Linux)

The Zen of Python

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Flat is better than nested.
- Sparse is better than dense.
- Readability counts.
- Special cases aren't special enough to break the rules.
- Although practicality beats purity.
- Errors should never pass silently.
- Unless explicitly silenced.
- In the face of ambiguity, refuse the temptation to guess.

Speaker notes

Source : https://www.python.org/dev/peps/pep-0020/

The Zen of Python

- There should be one -- and preferably only one -- obvious way to do it.
- Although that way may not be obvious at first unless you're Dutch.
- Now is better than never.
- Although never is often better than *right* now.
- If the implementation is hard to explain, it's a bad idea.
- If the implementation is easy to explain, it may be a good idea.
- Namespaces are one honking great idea -- let's do more of those!

Speaker notes

Source: https://www.python.org/dev/peps/pep-0020/

Python History

- Created By Guido Von Rossom in 1991
- Designed as an alternative to ”scripting languages” like Perl
  - Fully OOP (Object Oriented Programming)
- Dynamically Typed Language
  - “Duck Typing” – if it walks like a duck..
  - Automatic type conversion
- JIT (Just in Time)
  - Code compiles and runs in real time
  - No compile – package – deploy – cycle
- REPL Shell
  - Real time shell for analysis.
  - Read-Evaluate-Print-Loop Shell

Speaker notes

The language was called Python because of Guido’s fascination with the Monty Python TV show. Python’s association with snakes gave rise to snake-like names, such as Anaconda.

Python Versions

- Python v1.x : Not used, (1994-2000)
- Python v.2.x: Still Used! (2000-2007+)
  - Still some older libraries are Python 2.x only
  - Lots of older legacy code
  - Most new features back-ported into Python 2.x
- Python x.3.x: Current Version(2007-Present)
  - Most new features
  - Most libraries support python 3 now

Python 2 vs Python 3 Incompatibilities

	Python 2	Python 3
print semantics (important)	Statement print x # no parenthesis	Built-in function: print(x) #note parenthesis
input vs raw_input() (important)	input() evaluates input as code. `raw_input` returns data as `str`	input() returns data as `str`; There is no `raw_input`.
reduce	Statement	Built-in function in functools namespace
Annotations	Specified annotation only	User-defined annotations
Unicode	separate `str` (ascii) vs `unicode` types	All strings are internally stored as Unicode
Integer Division, (important)	`5 / 2` is `2`	`5 / 2` is `2.5` # int to float auto-conversion
		`5 // 2` is 2 # note the // for integer division

Should you use Python 2 or Python 3?

- You should use python 2 **only** if you have the following:
  - You inherit a large codebase of legacy python 2 code
  - You need a python package that only supports python 2
    - (These are becoming increasingly rare)
  - The community will continue to support Python 2 indefinitely
    - But new features will always show up in Python 3 first
- Python 3 is the future (and present) of Python
  - New development done here

Python Versus R

Python	R
General Purpose Language	Specialized for Statistics and Analytics
Object-Oriented Approach	Mixed Paradigm: Procedural/Functional
Large Package Repository: PyPI	Huge Package Repository; CRAN
Dynamic Language	Dynamic Language
Uses various editors, IDEs	Rstudio is highly integrated IDE for R
Favored by Computer Scientists	Favored By Statisticians
Summary: General Purpose Language now used widely in analytics	Summary: Specialized Language for Statistical Programming now used widely in analytics

Python Versus Java

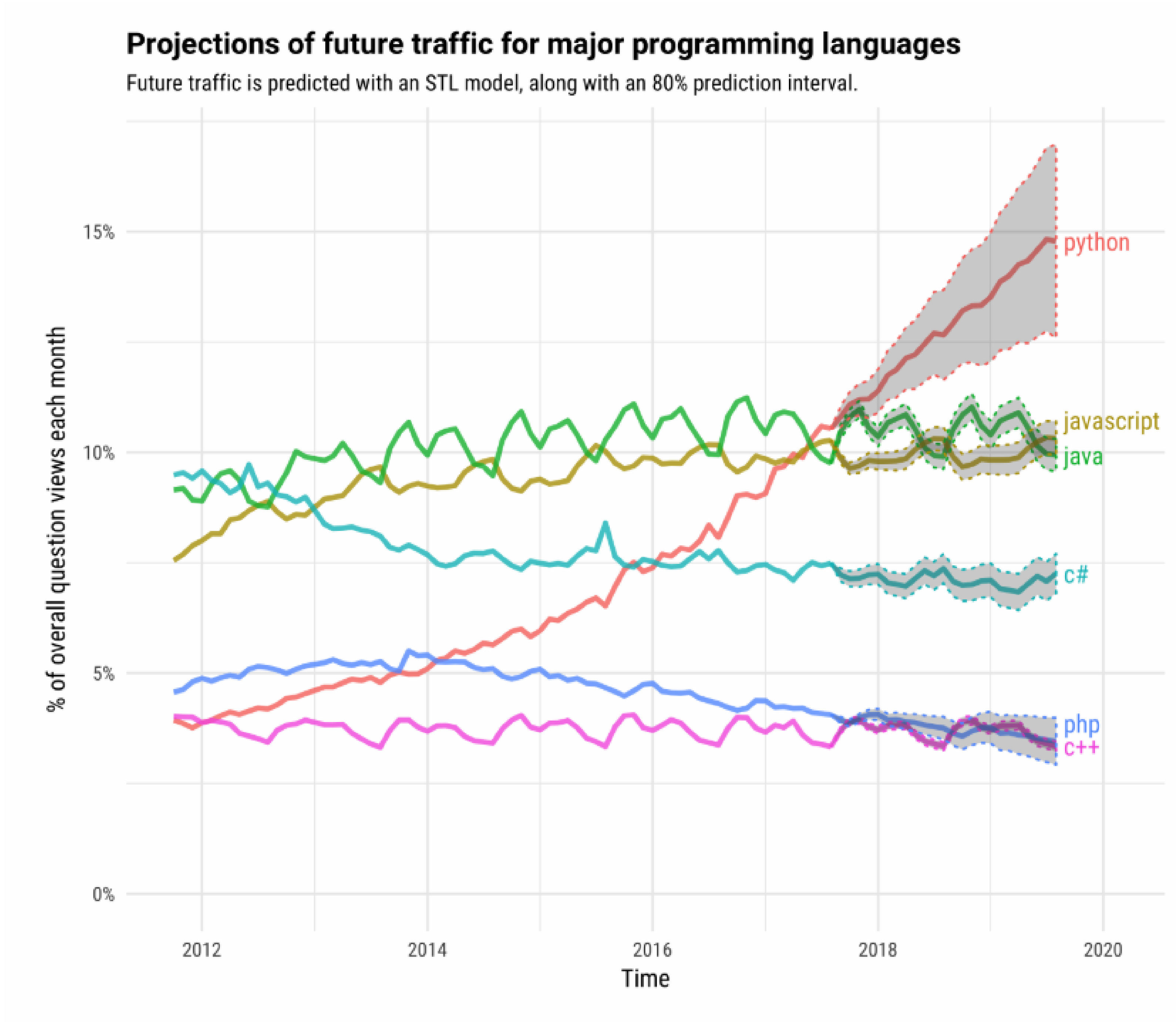
Python	Java
Dynamically Typed Language	Statically Typed Language
Interactive REPL Shell	No REPL
Can’t build dependencies into object	Can build dependencies into a FAT JAR.
Good for interactive analytics	Good for “productionizing” analytics.
Relatively slow	Faster (but not as fast as native code).
Interpreted (from bytecode)	Usually JIT compiled.

Python vs Javascript/Node

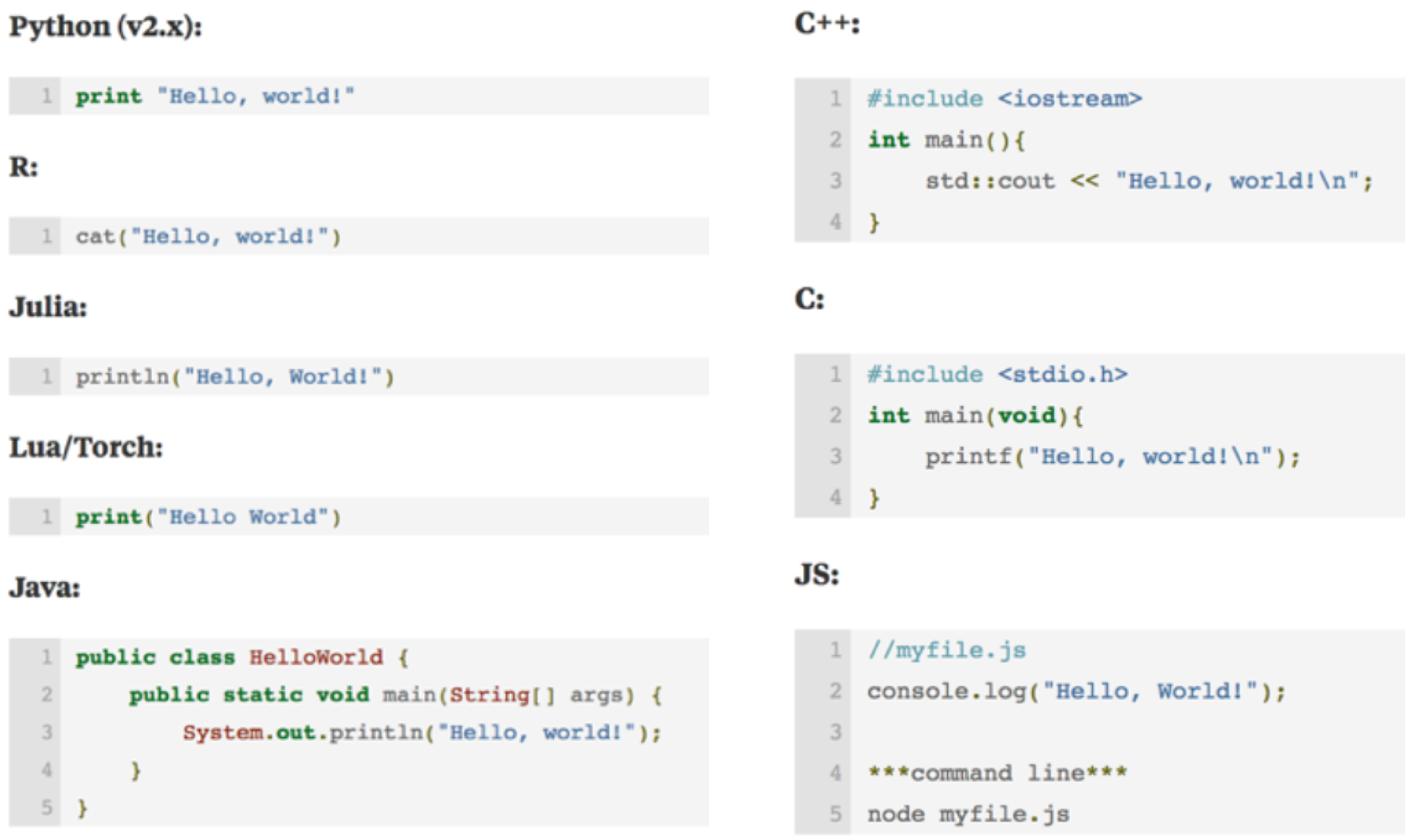
	Python	Javascript
Typing	Dynamic	Dynamic
Code Execution	Interpreted from Bytecode	V8: Interpreted from Bytecode
Runs client-side code in browser	No	Yes
Runs server side code	Yes	Yes (using node.js)
Standard Library	Extensive	Minimalistic
Package Management	via pip	via npm
Code Legibility	Generally good	Can be very unreadable
Native Code	Usually use native code in packages	Not common.
Summary	Good for science, data science, AI/machine learning, data munging	Good for web services and back-end for web development

Python Popularity

- Python Most popular language in 2018 on Stack Overflow



A simple “Hello, world!” comparison



Python (v2.x):

```python
print "Hello, world!"
```

R:

```r
cat("Hello, world!")
```

Julia:

```julia
println("Hello, World!")
```

Lua/Torch:

```lua
print("Hello World")
```

Java:

```java
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, world!");
    }
}
```

C++:

```cpp
#include <iostream>
int main(){
    std::cout << "Hello, world!\n";
}
```

C:

```c
#include <stdio.h>
int main(void){
    printf("Hello, world!\n");
}
```

JS:

```
//myfile.js
console.log("Hello, World!");

***command line***
node myfile.js
```

Python Use Cases

- Who uses Python?
- Python is very commonly used in the following areas:
  - Web Programming
  - Web Scraping
  - Microservices
  - System Automation Tasks
  - Scientific Programming
  - Data Analysis / Data Science
  - Machine Learning/Deep Learning and AI

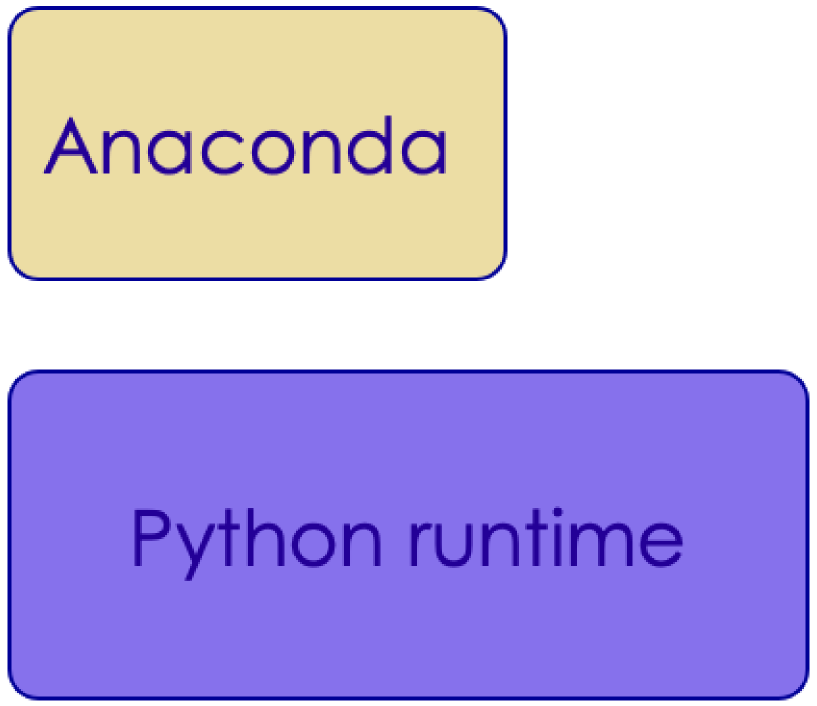
Is Python Interpreted?

- Standard Python (CPython) is **byte-code** interpreted
  - This means it will first just-in-time (JIT) compile the code to bytecode. (.pyc code)
  - Then, it will interpret (rather than compile) the bytecode
  - This allows for dynamic code and environment.
- Cython (C + Python) is **compiled** much like C/C++
  - Cython will first translate the Python code into C/C++
  - It will then combine that code with custom C/C++ code
  - It will then compile it into native code.

Is Python Fast?

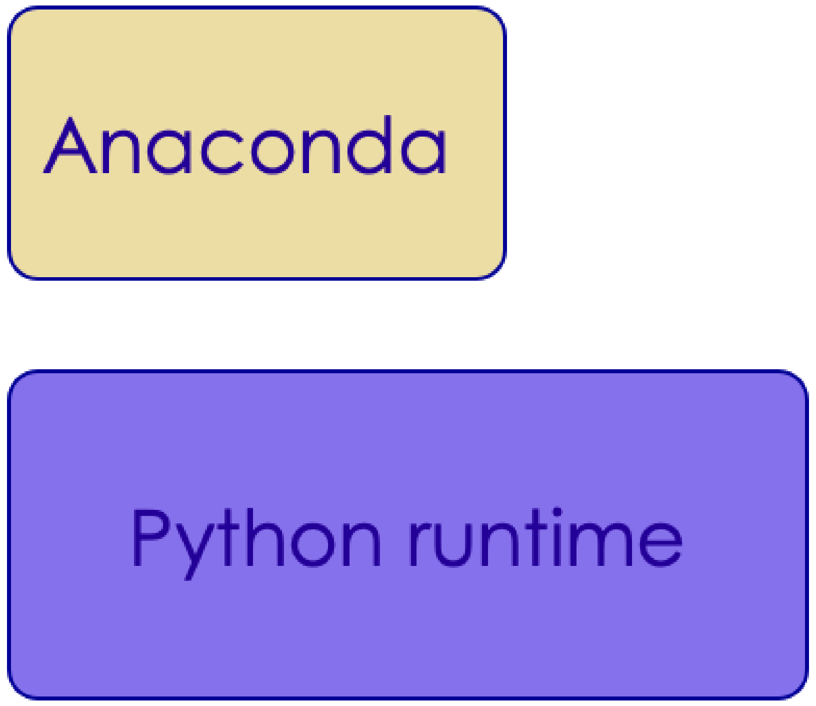
- Pure python code is fairly slow
  - Faster than purely interpreted languages (LISP, Smalltalk, shell scripts)
  - But slower than managed languages like Java, .NET
  - And much slower than pure native code like C/C++/Fortran.
- Python does allow users to write native code
  - Python integrates with C, C++, Fortran, etc.
  - Typically performance sensitive code is written in native code (usually C/C++)
  - Cython compiler: allows you to mix C, C++ and python code
- Native code is Fast!
- Write Native code (C/C++) for performance sensitive parts,
  - Everything else, pure python! Much easier

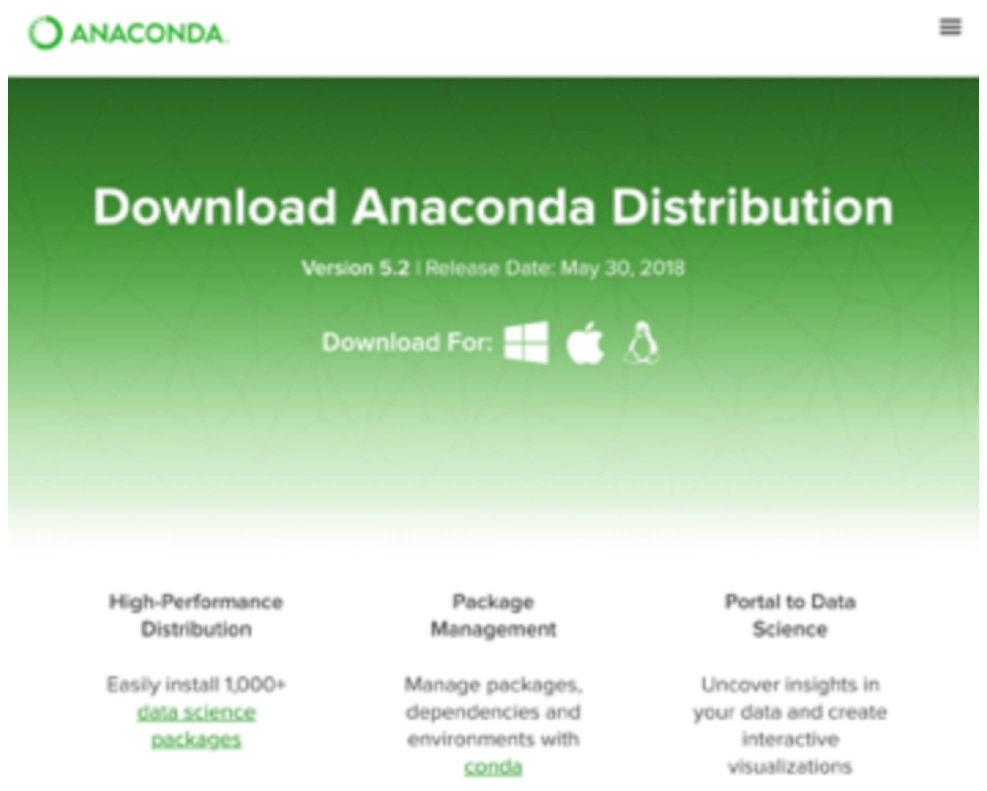
Installation



Installing Python

- We recommend downloading Anaconda from Continuum Analytics
  - [https://www.anaconda.com/]
- Download the free version of Anaconda Distribution for Windows / Mac / Linux at [http://www.anaconda.com/download]
- Ensure to get the Python `3.*` version




ANACONDA

Download Anaconda Distribution

Version 5.2 | Release Date: May 30, 2018

Download For:

High-Performance Distribution

Easily install 1,000+ data science packages

Package Management

Manage packages, dependencies and environments with conda

Portal to Data Science

Uncover insights in your data and create interactive visualizations

Why Anaconda?

- Anaconda has hundreds of the most commonly used DS packages already built!
  - No need for C/C++ compilers
  - Good for Windows Users! (Hard to build on Windows)
- Easy to Install Bundle
  - Platform Native bundling (MSI: Windows, DMG: Mac, etc)
- Commercially Available Support
  - Good for Enterprise Users
  - Easy for IT Services to “Certify” entire distribution including packages
- Separate from System Python
  - Anaconda is separate from your system python
  - So it won’t break anything else you may be doing on your machine in Python

Do I really need Anaconda?

- No, using your system python is fine (if you have one)
- Most Mac and Linux users already have Python
  - However, it may not be the latest version
  - On Linux, python `3.x` is often called python3
- You may want to use ”virtualenv” to create a virtual environment for your data science work
- You will have to download and install your own packages as-needed

Lab: Install Anaconda

- Overview:
  - Install Anaconda for Python 3.x
- Approximate time:
  - 10 mins
- Instructions:
  - **01-helloworld /**

Python Console

- We can also use Python from console

- Though Spyder is a much better interface

```
# Type “python“ to get into the python console
$ python

# Now we are in Python Shell. Type help()
>>> help()

# Now, we are in the help prompt. Type “for“.
[help> for

# To exit from the help prompt,
[help> control + D

# To exit python, back to the Linux prompt,
>>> control + D
```

Lab: Python REPL Shell

- Overview:
  - Learn Python REPL – Read Evaluate Print Loop Shell
- Approximate time:
  - 10 mins
- Instructions:
  - **02__pythonIntro | 2.1-repl.md**

Getting Started With Python REPL (Lab)

```
# prints to screen
print("hello world")

# create variables (= and = are equivalent)
a = 10
b = 3.1
c = "hello world"
```

- Lab: 01-helloworld/1.1-REPL.md

Running Python

- To run Python, we can just run the from the command line.

```
$ nano myprogram.py
#!/usr/env/python3

print(”Hello World!”)
```

```
$ python myprogram.py
Hello World!
```

Lab: Run a Script

- Overview:
  - Quick intro to running a script.
- Approximate time:
  - 10 mins
- Instructions:
  - **01-helloworld / 1.2-Script.md**

Lab: Writing a Python Script

- Overview::
  - Write a python script
- Approximate time:
  - 10 mins
- Instructions:
  - **02__pythonIntro | 2.2-script.md**
  - **02__pythonIntro | helloworld.py**

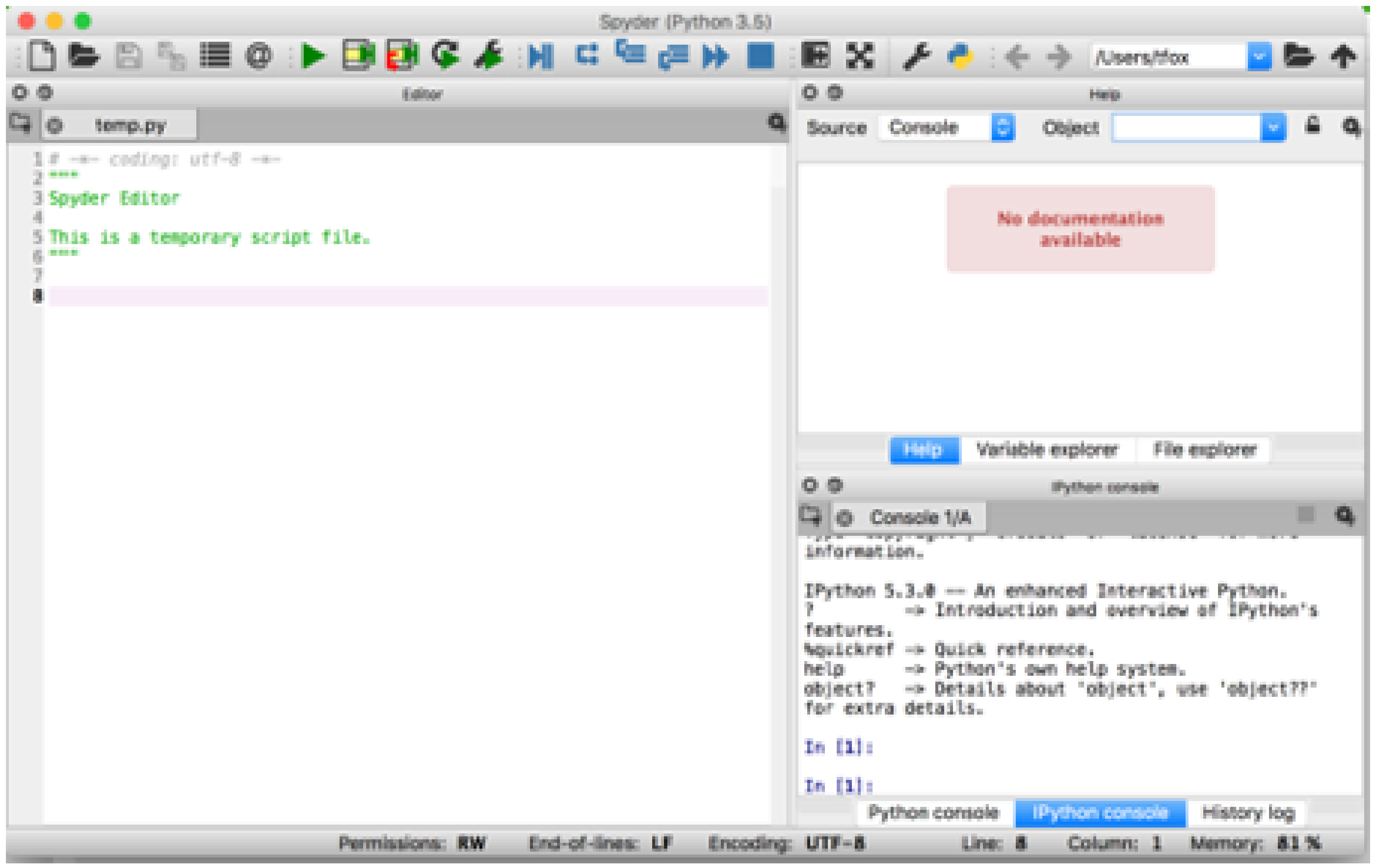
IDEs

IDEs

- Python Has Many IDEs
- We will discuss some common ones:
  - Spyder
  - PyCharm
  - PyDev
  - Visual Studio Code
  - Sublime Text / Other Text Editors.

Spyder

- Rstudio-like IDE for data science and scientific programming
- Lightweight
- Included with Anaconda
- Website - https://github.com/spyder-ide/spyder



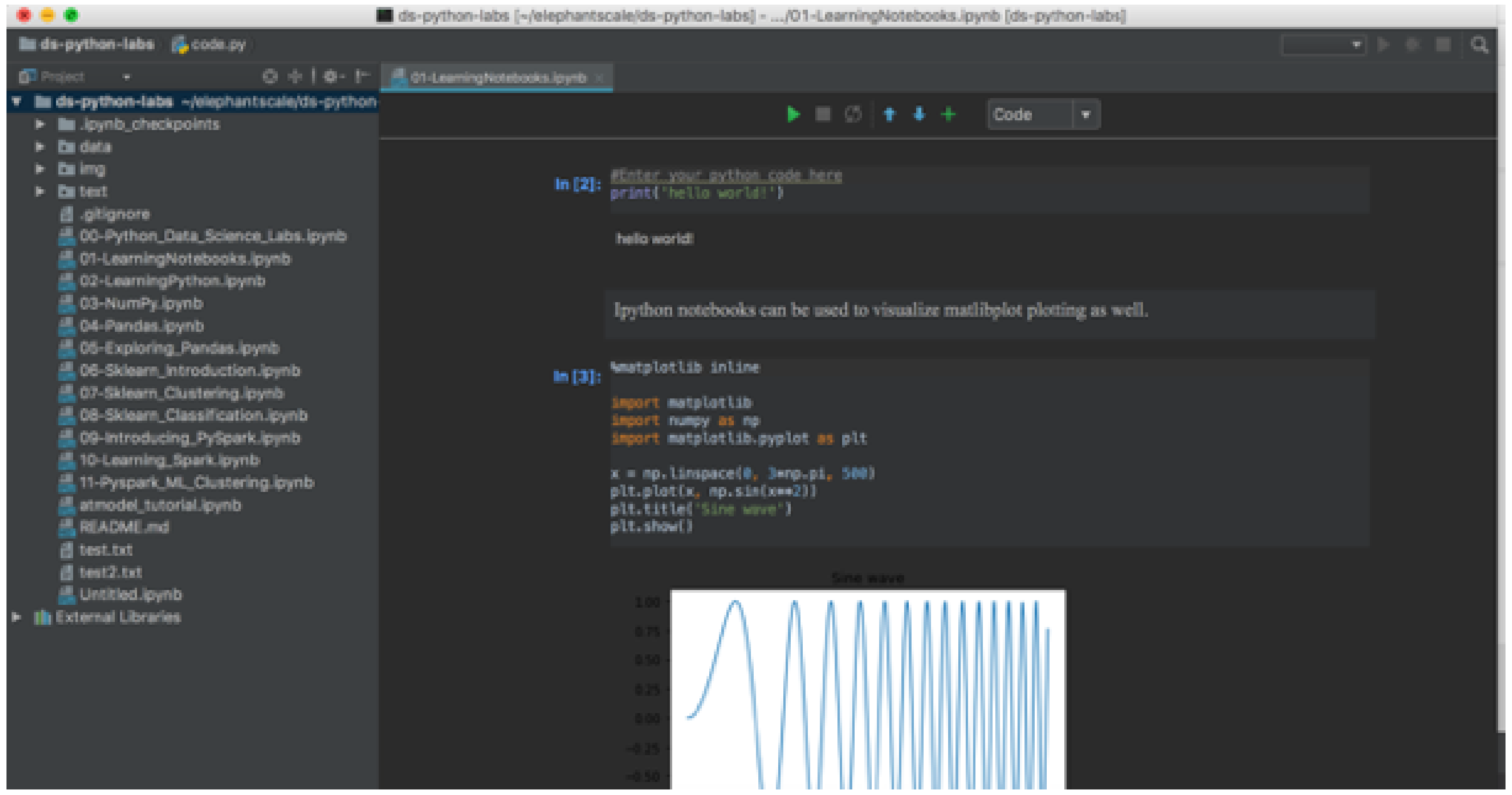
```
# -*- coding: utf-8 -*-
"""
Spyder Editor

This is a temporary script file.
"""
```

```
information.

IPython 5.3.0 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's
features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object', use 'object??'
for extra details.

In [1]:

In [1]:
```

Permissions: RW   End-of-lines: LF   Encoding: UTF-8   Line: 8   Column: 1   Memory: 81 %

Pycharm

- There are many other IDEs:
- Example: PyCharm from JetBrains
- Pretty Heavyweight


ds-python-labs [~/elephantscale/ds-python-labs] - .../01-LearningNotebooks.ipynb [ds-python-labs]
ds-python-labs
code.py
Project
01-LearningNotebooks.ipynb
Code
ds-python-labs ~/elephantscale/ds-python
.ipynb_checkpoints
data
img
text
.gitignore
00-Python_Data_Science_Labs.ipynb
01-LearningNotebooks.ipynb
02-LearningPython.ipynb
03-NumPy.ipynb
04-Pandas.ipynb
05-Exploring_Pandas.ipynb
06-Sklearn_Introduction.ipynb
07-Sklearn_Clustering.ipynb
08-Sklearn_Classification.ipynb
09-Introducing_PySpark.ipynb
10-Learning_Spark.ipynb
11-Pyspark_ML_Clustering.ipynb
atmodel_tutorial.ipynb
README.md
test.txt
test2.txt
Untitled.ipynb
External Libraries
Sine wave
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50

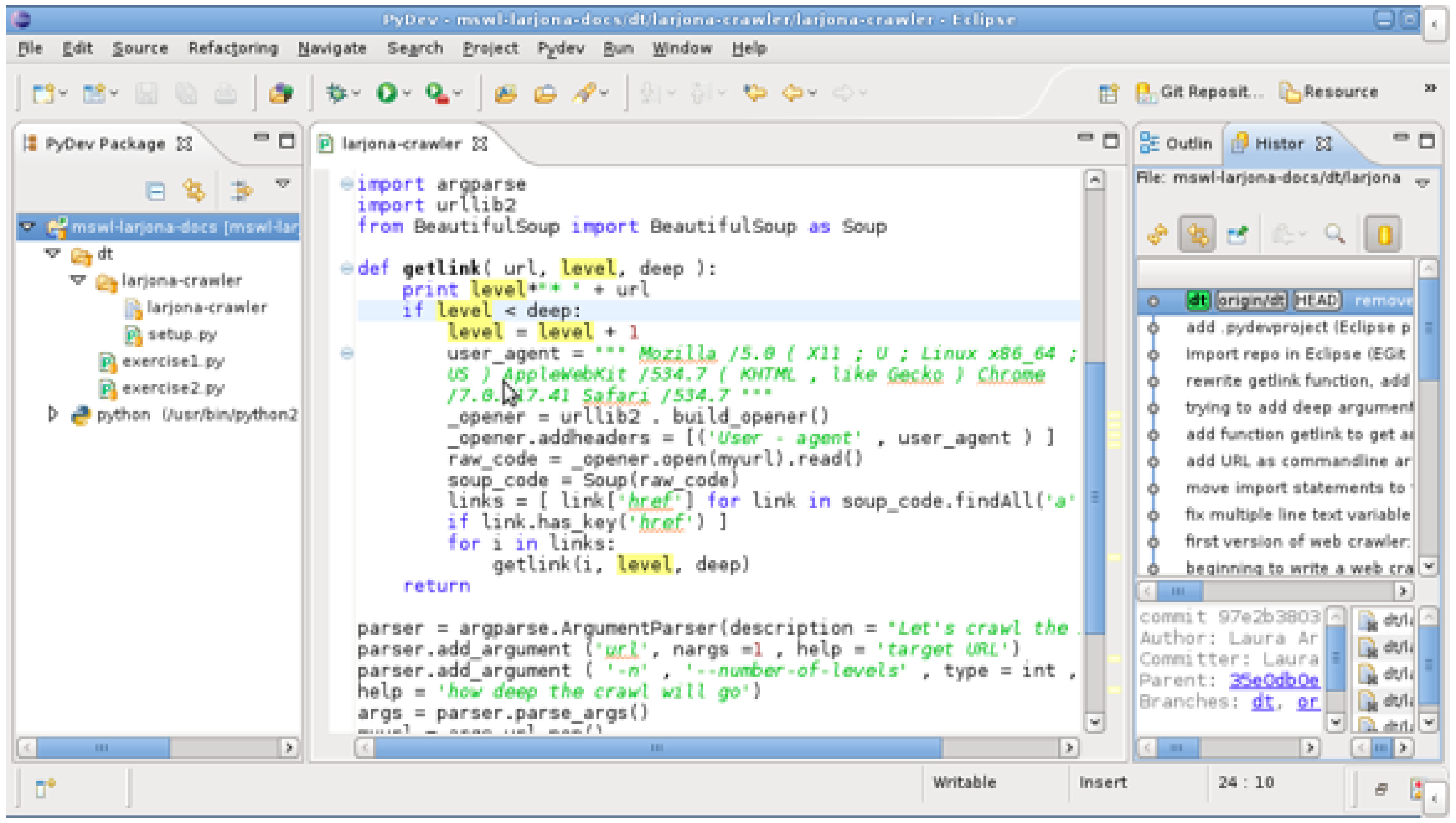
```
In [2]: #Enter your python code here
        print('hello world!')

hello world!
```

Ipython notebooks can be used to visualize matlibplot plotting as well.

```
In [3]: %matplotlib inline

        import matplotlib
        import numpy as np
        import matplotlib.pyplot as plt

        x = np.linspace(0, 3*np.pi, 500)
        plt.plot(x, np.sin(x**2))
        plt.title('Sine wave')
        plt.show()
```

PyDev / Eclipse

- PyDev is an extension to eclipse
- Website - http://www.pydev.org/



PyDev - mswl-larjona-docs/dt/larjona-crawler/larjona-crawler - Eclipse
File Edit Source Refactoring Navigate Search Project Pydev Run Window Help
Git Reposit... Resource
PyDev Package
mswl-larjona-docs [mswl-lar
dt
larjona-crawler
larjona-crawler
setup.py
exercise1.py
exercise2.py
python (/usr/bin/python2
larjona-crawler
Outlin
Histor
File: mswl-larjona-docs/dt/larjona
dt
origin/dt
HEAD
remove
Writable
Insert
24 : 10

```
import argparse
import urllib2
from BeautifulSoup import BeautifulSoup as Soup

def getlink( url, level, deep ):
    print level*"* " + url
    if level < deep:
        level = level + 1
        user_agent = """ Mozilla /5.0 ( X11 ; U ; Linux x86_64 ;
        US ) AppleWebKit /534.7 ( KHTML , like Gecko ) Chrome
        /7.0.517.41 Safari /534.7 """
        _opener = urllib2 . build_opener()
        _opener.addheaders = [('User - agent' , user_agent ) ]
        raw_code = _opener.open(myurl).read()
        soup_code = Soup(raw_code)
        links = [ link['href'] for link in soup_code.findAll('a'
        if link.has_key('href') ]
        for i in links:
            getlink(i, level, deep)
    return

parser = argparse.ArgumentParser(description = 'Let's crawl the
parser.add_argument ('url', nargs =1 , help = 'target URL')
parser.add_argument ( '-n' , '--number-of-levels' , type = int ,
help = 'how deep the crawl will go')
args = parser.parse_args()
```

```
add .pydevproject (Eclipse p
Import repo in Eclipse (EGit
rewrite getlink function, add
trying to add deep argument
add function getlink to get a
add URL as commandline ar
move import statements to
fix multiple line text variable
first version of web crawler:
beginning to write a web cra
```

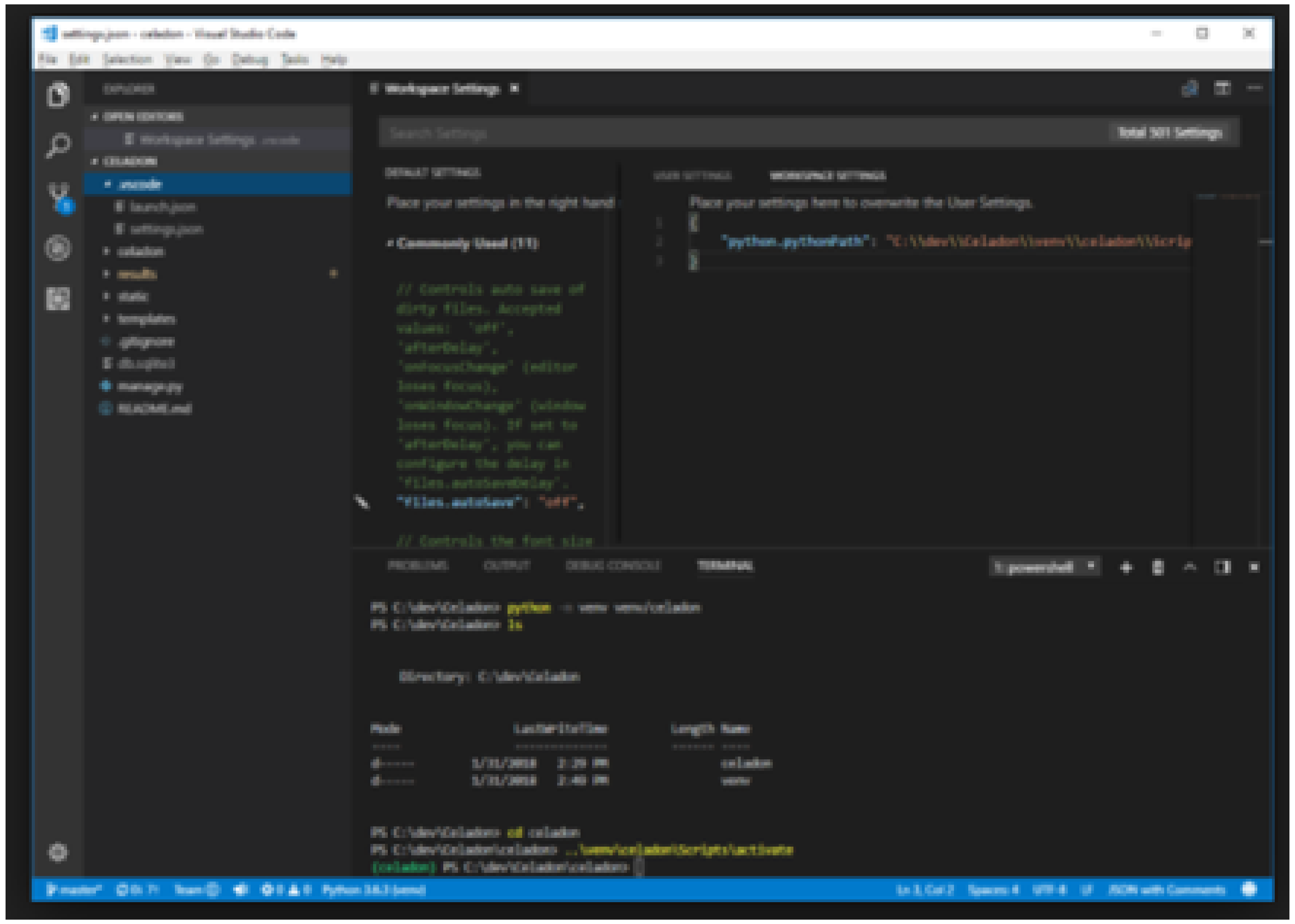
```
commit 97e2b3803
Author: Laura Ar
Committer: Laura
Parent: 35e0db0e
Branches: dt, or
```

PyDev / Eclipse

- PyDev is an extension to eclipse

Visual Studio Code

- For all the Microsoft Fans out there:
- Surprisingly lightweight, but fully-featured IDE!
- Website - https://code.visualstudio.com/



Place your settings in the right hand

Place your settings here to overwrite the User Settings.

```
// Controls auto save of
dirty files. Accepted
values:  'off',
'afterDelay',
'onFocusChange' (editor
loses focus),
'onWindowChange' (window
loses focus). If set to
'afterDelay', you can
configure the delay in
'files.autoSaveDelay'.
"files.autoSave": "off",

// Controls the font size
```

```
{
    "python.pythonPath": "C:\\dev\\Celadon\\venv\\celadon\\Scrip
}
```

```
PS C:\dev\Celadon> python -m venv venv/celadon
PS C:\dev\Celadon> ls


    Directory: C:\dev\Celadon


Mode                LastWriteTime         Length Name
----                -------------         ------ ----
d-----        1/31/2018   2:39 PM                celadon
d-----        1/31/2018   2:40 PM                venv


PS C:\dev\Celadon> cd celadon
PS C:\dev\Celadon\celadon> ..\venv\celadon\Scripts\activate
(celadon) PS C:\dev\Celadon\celadon>
```

Other options

- Some just like text editors
    - Vi(m)
    - Sublime Text
- Advantages:
    - Lightweight and agile
    - Fast
- Limited interactive debugging / breakpoints

Lab: Introducing Spyder

- Overview:
    - Quick intro lab to Spyder IDE
- Approximate time:
    - 10 mins
- Instructions:
    - **01-helloworld / 1.3-Spyder.md**

Lab: Introducing Spyder

- Overview:
  - Introduction to Spyder IDE
- Approximate time:
  - 10 mins
- Instructions:
  - **02__pythonIntro | 2.3-spyder.md**

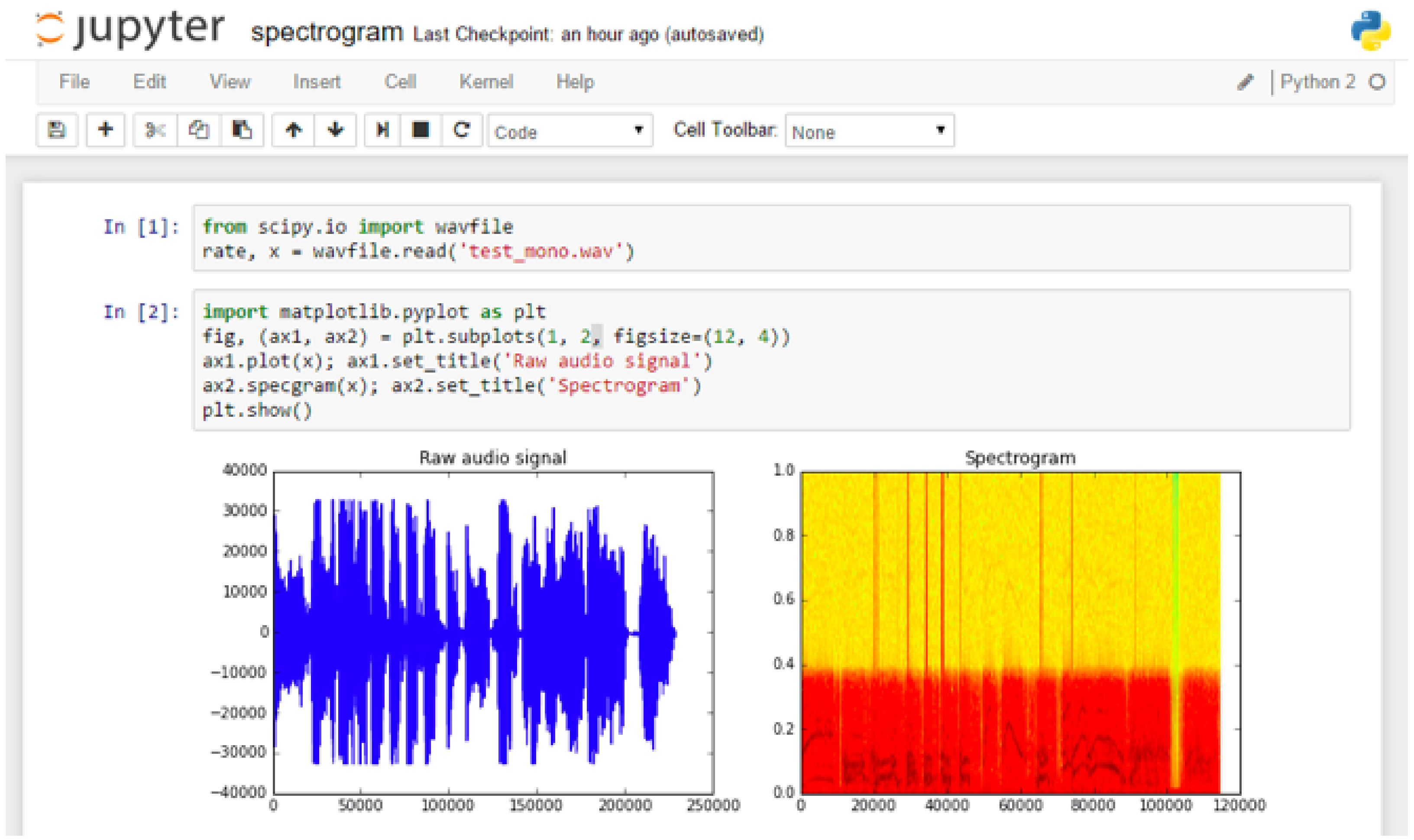
Jupyter Notebook

Jupyter Notebooks

- Jupyter notebooks are a great way to showcase working python code.
- Jupyter Notebooks allow us to combine:
  - Text
  - HTML / Images
  - Visualizations
  - Runnable Python Code
- Document Centric View
- We do Jupyter notebooks in this class!
- NOT an IDE.
  - Good for exploratory data analysis
  - Showcasing code

Jupyter Notebooks

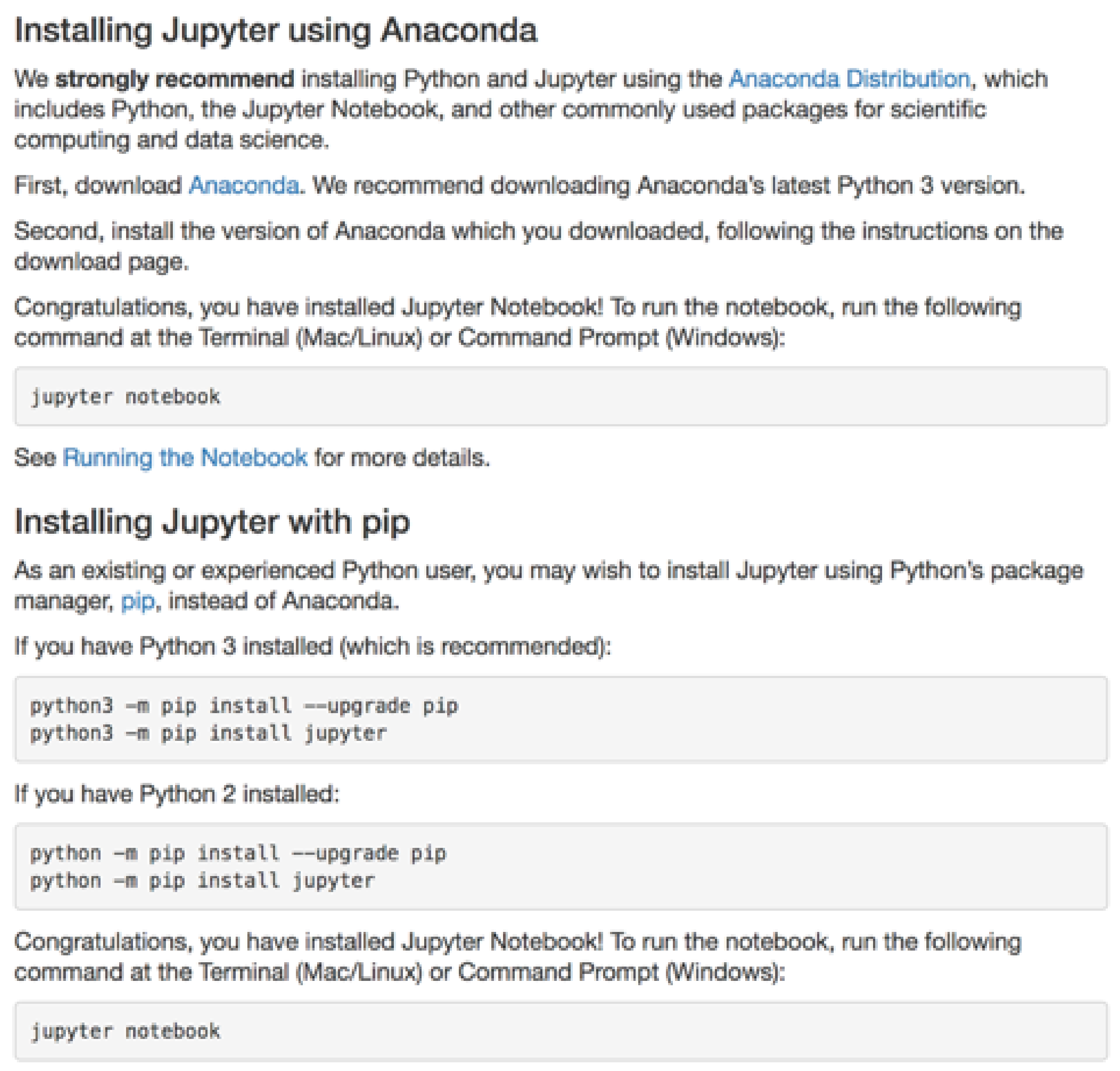
- Example:



```
In [1]: from scipy.io import wavfile
        rate, x = wavfile.read('test_mono.wav')

In [2]: import matplotlib.pyplot as plt
        fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 4))
        ax1.plot(x); ax1.set_title('Raw audio signal')
        ax2.specgram(x); ax2.set_title('Spectrogram')
        plt.show()
```

Jupyter Notebook Installation

- Project Jupyter Website - https://jupyter.org/
- Installation recommended using Anaconda, but Jupyter can be installed individually too - https://jupyter.org/install



Installing Jupyter using Anaconda

We **strongly recommend** installing Python and Jupyter using the Anaconda Distribution, which includes Python, the Jupyter Notebook, and other commonly used packages for scientific computing and data science.

First, download Anaconda. We recommend downloading Anaconda's latest Python 3 version.

Second, install the version of Anaconda which you downloaded, following the instructions on the download page.

Congratulations, you have installed Jupyter Notebook! To run the notebook, run the following command at the Terminal (Mac/Linux) or Command Prompt (Windows):

```
jupyter notebook
```

See Running the Notebook for more details.

Installing Jupyter with pip

As an existing or experienced Python user, you may wish to install Jupyter using Python's package manager, pip, instead of Anaconda.

If you have Python 3 installed (which is recommended):

```
python3 -m pip install --upgrade pip
python3 -m pip install jupyter
```

If you have Python 2 installed:

```
python -m pip install --upgrade pip
python -m pip install jupyter
```

Congratulations, you have installed Jupyter Notebook! To run the notebook, run the following command at the Terminal (Mac/Linux) or Command Prompt (Windows):

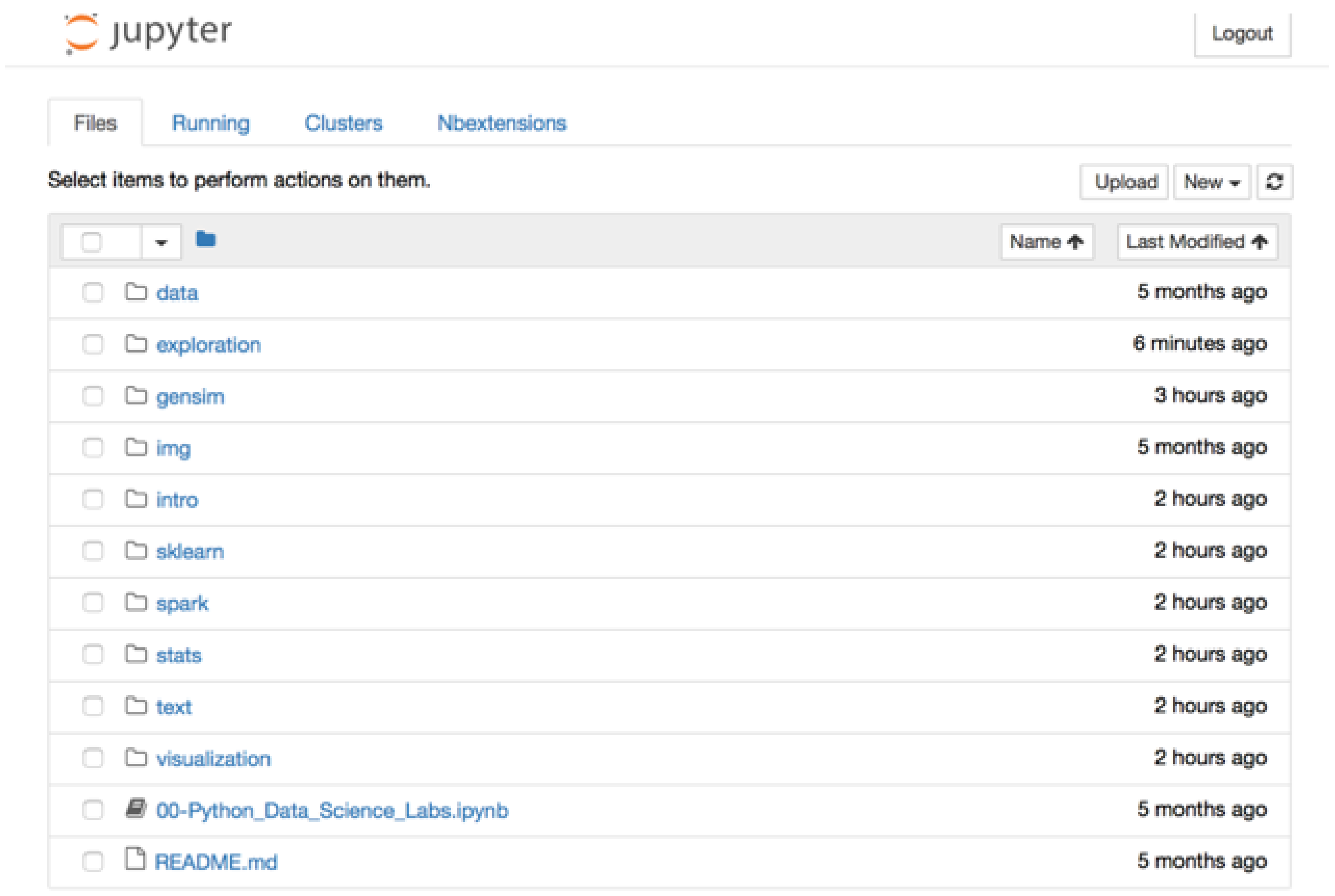
```
jupyter notebook
```

Using Jupyter Notebooks

- Go to directory and type jupyter notebook
- The Jupyter notebook by default starts at http://localhost:8888

```
$ cd /path/to/my/notebooks

$ jupyter notebook

[I 22:28:59.637 NotebookApp] The Jupyter Notebook is
running at: http://localhost:8888/?token=YOURTOKEN

[I 22:28:59.637 NotebookApp] Use Control-C to
stop this server and shut down all kernels
(twice to skip confirmation).

[C 22:28:59.639 NotebookApp]    Copy/paste this URL into
your browser when you connect for the first time,
to login with a token:
http://localhost:8888/?token=YOURTOKEN
```

Browser for Jupyter Notebook



jupyter

Logout

Files Running Clusters Nbextensions

Select items to perform actions on them.

Upload New

Name	Last Modified
data	5 months ago
exploration	6 minutes ago
gensim	3 hours ago
img	5 months ago
intro	2 hours ago
sklearn	2 hours ago
spark	2 hours ago
stats	2 hours ago
text	2 hours ago
visualization	2 hours ago
00-Python_Data_Science_Labs.ipynb	5 months ago
README.md	5 months ago

Lab: Running Jupyter

- Overview:
  - Run Jupyter Notebook
- Approximate time:
  - 10 mins
- Instructions:
  - **02-notebooks / 2.1-install-jupyter.md**

Lab: Introducing Notebooks

- Overview:
  - Quick intro lab to Jupyter Notebook
- Approximate time:
  - 10 mins
- Instructions:
  - **02-notebooks / 2.2-LearningNotebooks.ipynb**

Lab: Introducing Jupyter Notebook

- Overview:
  - Introduction to Jupyter Notebook
- Approximate time:
  - 10 mins
- Instructions:
  - **02__pythonIntro | 2.4-startingJupyter.md**
  - **02__pythonIntro | learningJupyter.ipynb**