DS
IA

2019

Data Science and AI

Module 1
Part 2:

Python for Data Science

Agenda: Module 1 Part 2

- Python Fundamentals
- Advanced Python Programming
- Software Engineering Best Practices
- Using Git & GitHub for Version Control
- Deploying Python Applications

Python Fundamentals

- Version clash: 2.7 vs 3.x
- Installing packages with pip
- Environments
- Installing and using Anaconda
- IPython, Spyder
- Jupyter notebooks
- Data structures in Python
- Writing functions in Python
- Iterating in Python
- numpy, pandas, scikit-learn

Version Clash: 2.7 vs 3.x

- version 2.x
  - large code base
  - last version = 2.7 (no more releases!)

- version 3.x
  - *print* is a function
  - raising & catching exceptions
  - integer division (2.x truncates; 3.x converts to float)
  - short → long integers
  - octal constants: 0*nnn* → 0o*nnn*
  - unicode strings
  - …

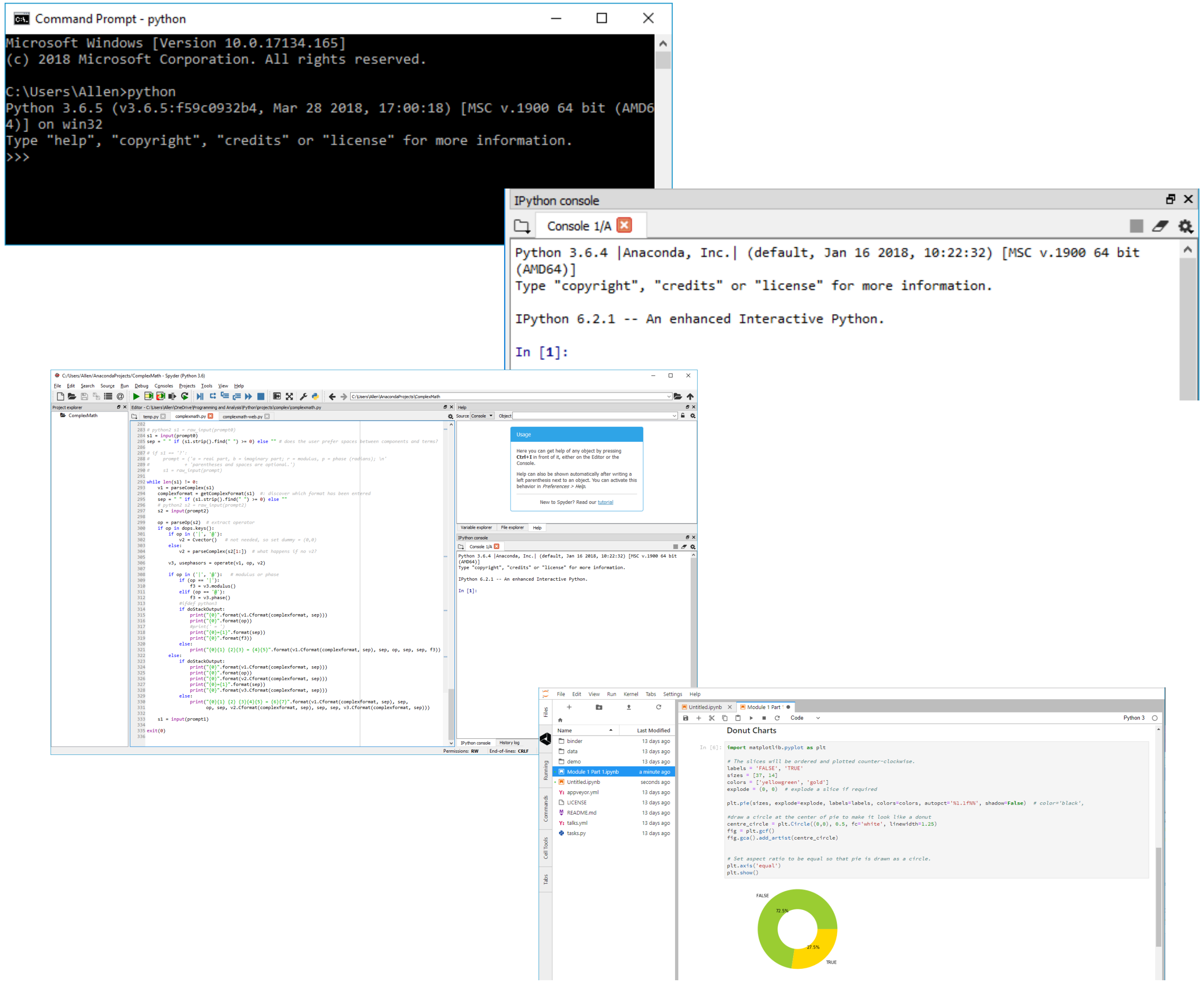
Version Clash: 2.7 vs 3.x

- cheat sheet
  - http://python-future.org/compatible_idioms.html#
- dual-version support
  - pkg future

    https://docs.python.org/3/howto/pyporting.html

Running Python

- command prompt
- Ipython
  - Python IDE
- Spyder
  - IPython-based IDE
  - stand-alone, Anaconda Navigator
- Jupyter notebooks
  - IPython-based, browser-hosted
  - stand-alone, Anaconda Navigator



```
Microsoft Windows [Version 10.0.17134.165]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\Allen>python
Python 3.6.5 (v3.6.5:f59c0932b4, Mar 28 2018, 17:00:18) [MSC v.1900 64 bit (AMD6
4)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

```
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 10:22:32) [MSC v.1900 64 bit
(AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 6.2.1 -- An enhanced Interactive Python.

In [1]:
```

```
In [6]: import matplotlib.pyplot as plt

        # The slices will be ordered and plotted counter-clockwise.
        labels = 'FALSE', 'TRUE'
        sizes = [37, 14]
        colors = ['yellowgreen', 'gold']
        explode = (0, 0)  # explode a slice if required

        plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=False)  # color='black',

        #draw a circle at the center of pie to make it look like a donut
        centre_circle = plt.Circle((0,0), 0.5, fc='white', linewidth=1.25)
        fig = plt.gcf()
        fig.gca().add_artist(centre_circle)


        # Set aspect ratio to be equal so that pie is drawn as a circle.
        plt.axis('equal')
        plt.show()
```

Installing Packages with pip

Install pip …

Windows

1. download get-pip.py to a local folder
2. open a cmd window; navigate to the folder
3. $ python get-pip.py

macOS / OS X

1. $ sudo easy_install pip

Installing Packages with pip – cont’d

- install a package
- upgrade a package
- install a specific version
- install a set of requirements
- install from an alternate index
- install from a local archive

```
$ pip install anypkg
$ pip install --upgrade anypkg
$ pip install anypkg==1.0.4
$ pip install -r reqsfile.txt
$ pip install --index-url http://my.package.repo/simple/ anypkg
$ pip install ./downloads/anypkg-1.0.1.tar.gz
```

https://packaging.python.org/tutorials/installing-packages/
https://docs.python.org/3/installing/index.html

Environments

What is an environment?

> a practical way to deal with Python’s own DLL hell

Issues:

- many packages have not been around long enough to be tested with other packages that you might want to use with them
- packages don’t always get updated quickly in response to updated dependences

solution: virtual environments for hosting isolated projects

- venv
- conda

Environments – cont'd: conda

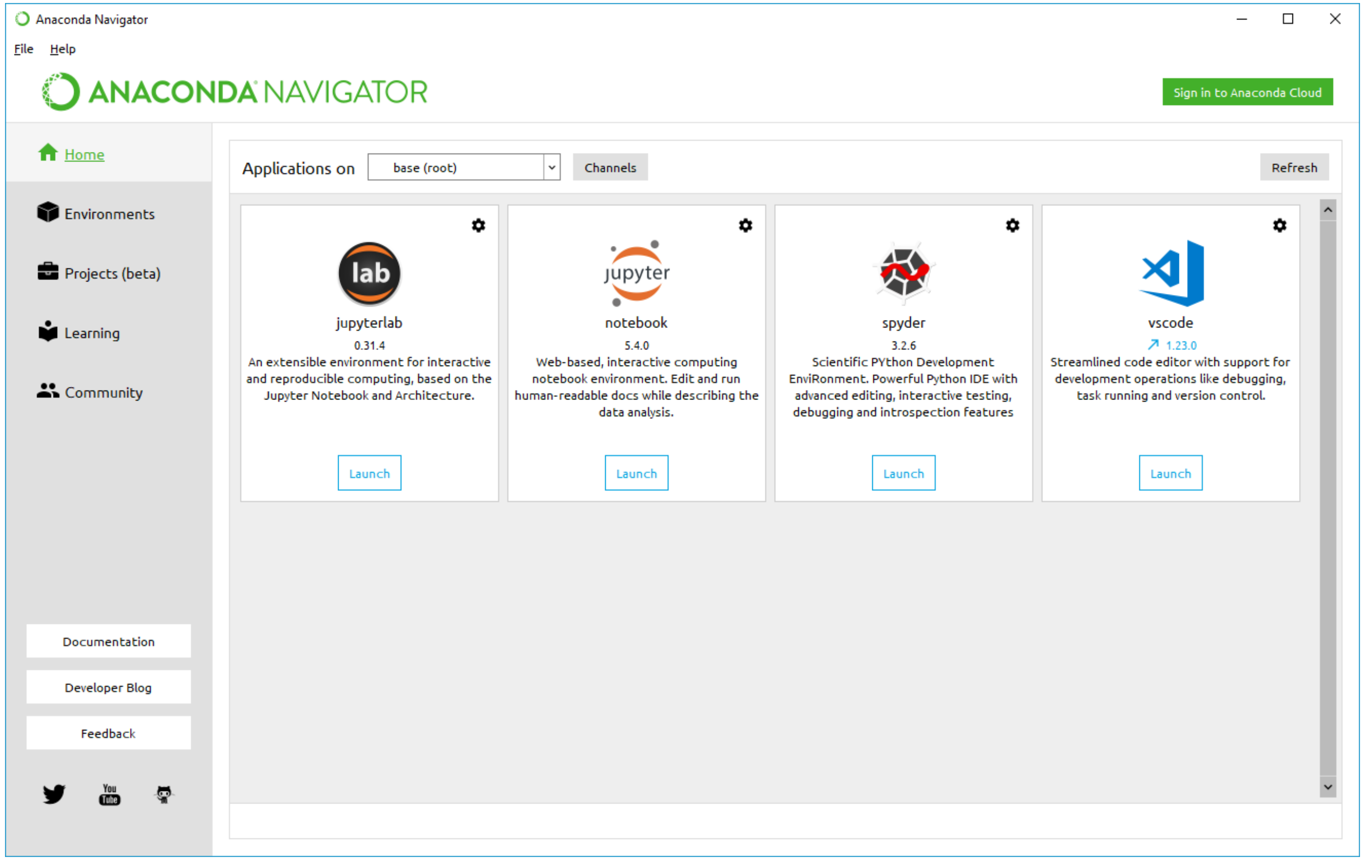
- create an environment
- activate an environment
- deactivate an environment
- install python
- search for available packages
- install a package
- list installed packages

```
$ conda create --name myenv1 python
$ source activate myenv1
$ source deactivate
$ conda install python=version
$ conda search searchterm
$ conda install anypkg
$ conda list --name myenv1
```

https://conda.io/docs/commands.html

Environments – cont'd: Anaconda Navigator

- launcher for a variety of Python (and other) development environments



Applications on base (root)

jupyterlab
0.31.4
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

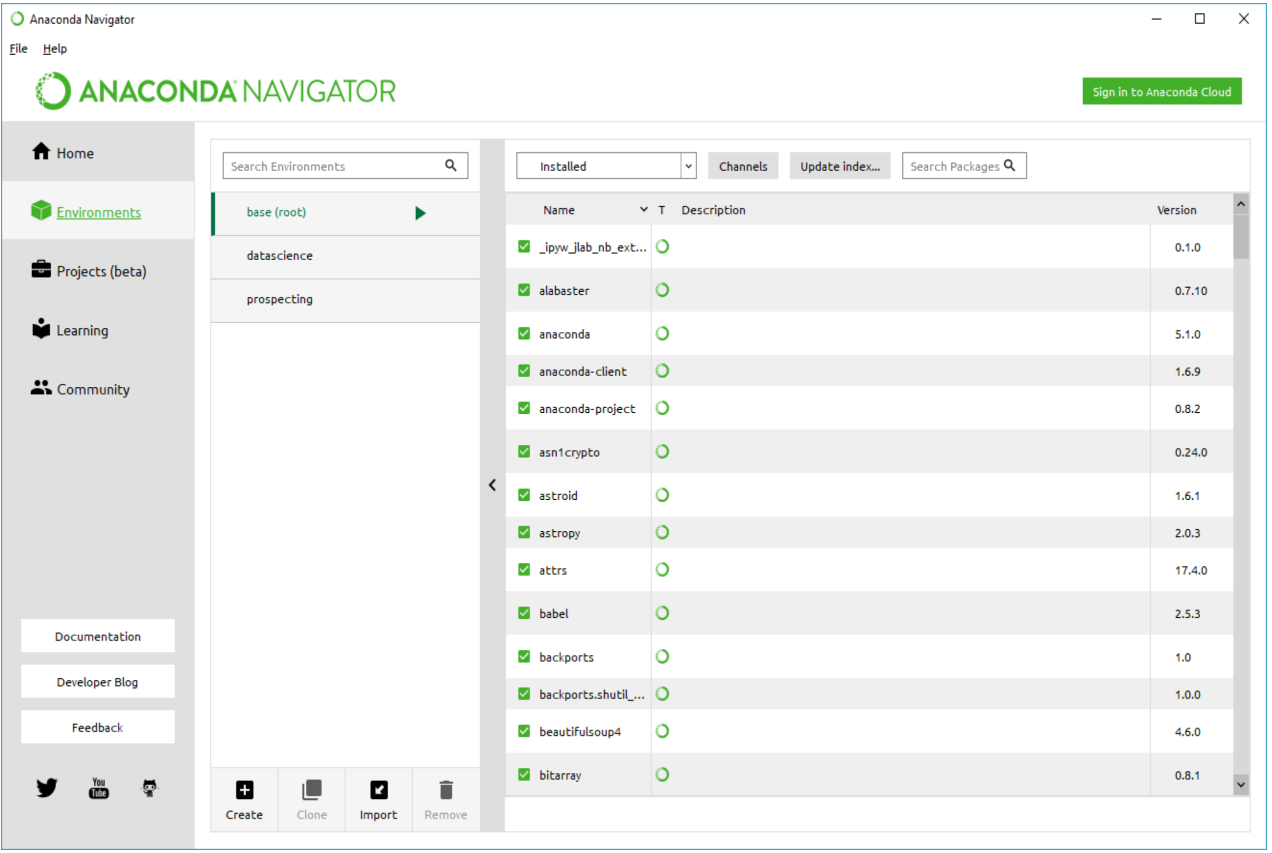
notebook
5.4.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.

spyder
3.2.6
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features

vscode
↗ 1.23.0
Streamlined code editor with support for development operations like debugging, task running and version control.

Environments – cont’d: Anaconda Navigator

- implements conda via a GUI
  - create envs
  - switch between envs
  - list packages in an env
  - search for packages to add to env
- env-specific app instances
  - set env (e.g. Python27)
  - launch Jupyter notebook to run Python 2.7 code



	Name	T	Description	Version
☑	_ipyw_jlab_nb_ext...	○		0.1.0
☑	alabaster	○		0.7.10
☑	anaconda	○		5.1.0
☑	anaconda-client	○		1.6.9
☑	anaconda-project	○		0.8.2
☑	asn1crypto	○		0.24.0
☑	astroid	○		1.6.1
☑	astropy	○		2.0.3
☑	attrs	○		17.4.0
☑	babel	○		2.5.3
☑	backports	○		1.0
☑	backports.shutil_...	○		1.0.0
☑	beautifulsoup4	○		4.6.0
☑	bitarray	○		0.8.1

Installing a Package in an Environment

Option 1: Using Anaconda Navigator

1. Select, clone, or create environment in Environments tab
2. Check for required package with Installed selected in drop-down
3. If not found, select Not Installed to search conda packages available for download/installation
4. If found among available packages:
    - select checkbox
    - click Apply

Installing a Package in an Environment – cont’d

Option 2: Using Conda (in the command window)

1. Check for required package:
   - open Anaconda Prompt (Windows) or Terminal window (Mac, Linux, Unix)
   - activate environment, check installed packages:
     - $ activate your_env
     - $ conda list --n myenv

2. If not found among installed packages:
   - try installing from conda:
     - $ conda install new_pkg

Installing a Package in an Environment – cont'd

If the package is not available in *conda:*

a. try another channel:
   - e.g. *conda-forge* ($ conda config --add channels conda-forge)

b. use *pip* instead:
   (Nb. no option to use Anaconda Navigator, yet):
   - open Anaconda Prompt (Windows) or Terminal window (Mac, Linux, Unix)
   - Windows:

     ```
     $ python -m pip install new_pkg
     ```
   - Mac, Linux, Unix:

     ```
     $ python -m pip install new_pkg
     ```

Notes:

- prefixing the command with python -m ensures that the package gets installed for the active python installation (i.e. the current environment)

Installing a Package in an Environment – cont’d

Option 3: Installing a package from within a Jupyter notebook

(to the current Jupyter kernel):

1. Import the sys package:

```
import sys
```

2. If the package **is** in *conda*:

```
!conda install --yes --prefix {sys.prefix} new_pkg
```

3. If the package is **not** in *conda*:

```
!{sys.executable} -m pip install new_pkg
```

Installing a Package in an Environment – cont'd

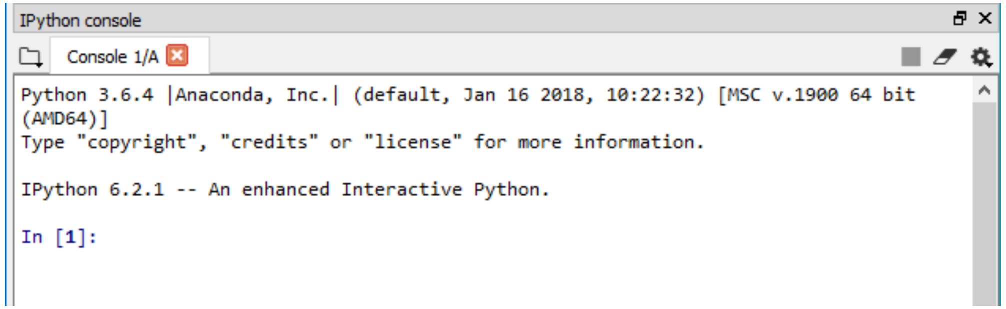
Notes

- always double-check that the target environment is active before installing packages
- the references to sys guarantee that the package installs in the same environment that Jupyter was launched from
- ***never*** use sudo pip install new_pkg
  - this may override objections from the operating system's default Python environment, causing future problems
- installed packages can be upgraded to the most recent version by inserting the --upgrade option
- a specific version of a package can be installed
  - do this when the latest version does not work with other packages in the target environment

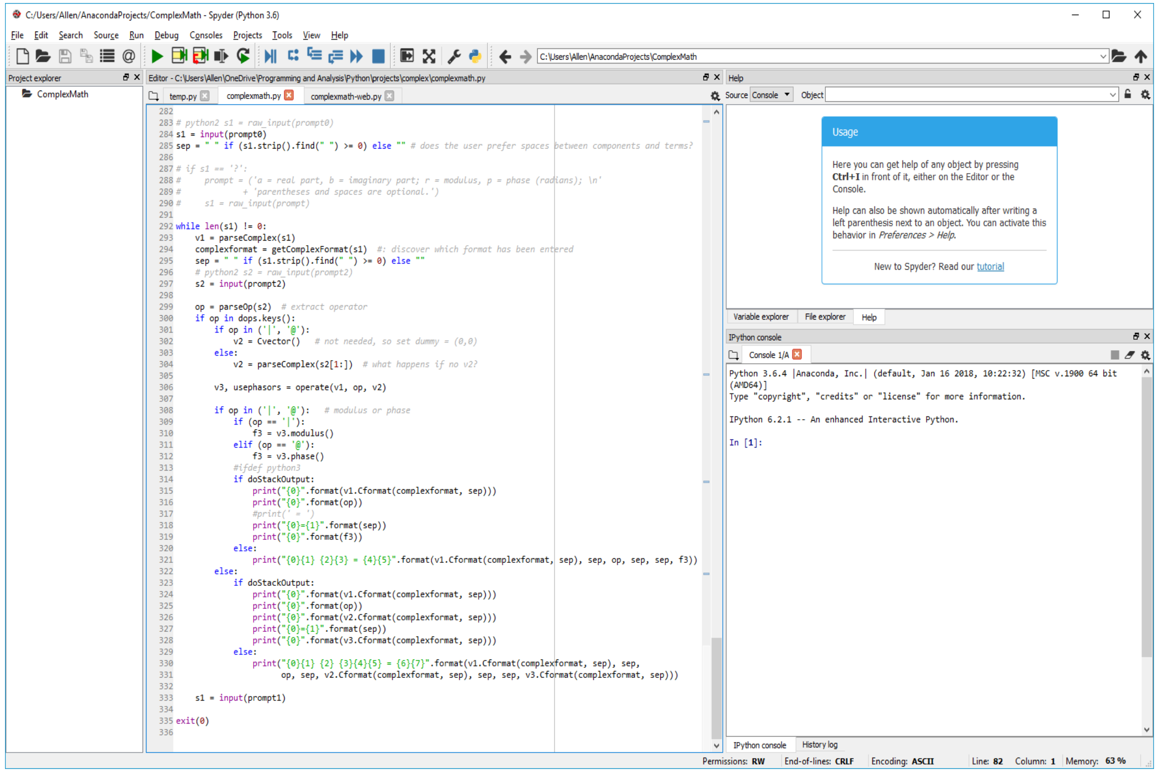
https://conda.io/docs/user-guide/tasks/index.html
https://jakevdp.github.io/blog/2017/12/05/installing-python-packages-from-jupyter/

Spyder

- editor
    - tab completion
        - extend with jedi
    - breakpoints
    - interactive IPython console
- commands
    - line magics %
    - cell magics %%
- graphics (plotting)
    - inline: %matplotlib inline
    - external: %matplotlib



```
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 10:22:32) [MSC v.1900 64 bit
(AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 6.2.1 -- An enhanced Interactive Python.

In [1]:
```



```python

# python2 s1 = raw_input(prompt0)
s1 = input(prompt0)
sep = " " if (s1.strip().find(" ") >= 0) else "" # does the user prefer spaces between components and terms?

# if s1 == '?':
#     prompt = ('a = real part, b = imaginary part; r = modulus, p = phase (radians); \n'
#             + 'parentheses and spaces are optional.')
#     s1 = raw_input(prompt)

while len(s1) != 0:
    v1 = parseComplex(s1)
    complexformat = getComplexFormat(s1)  #: discover which format has been entered
    sep = " " if (s1.strip().find(" ") >= 0) else ""
    # python2 s2 = raw_input(prompt2)
    s2 = input(prompt2)

    op = parseOp(s2)  # extract operator
    if op in dops.keys():
        if op in ('|', '@'):
            v2 = Cvector()   # not needed, so set dummy = (0,0)
        else:
            v2 = parseComplex(s2[1:])  # what happens if no v2?

        v3, usephasors = operate(v1, op, v2)

        if op in ('|', '@'):   # modulus or phase
            if (op == '|'):
                f3 = v3.modulus()
            elif (op == '@'):
                f3 = v3.phase()
            #ifdef python3
            if doStackOutput:
                print("{0}".format(v1.Cformat(complexformat, sep)))
                print("{0}".format(op))
                #print(' = ')
                print("{0}={1}".format(sep))
                print("{0}".format(f3))
            else:
                print("{0}{1} {2}{3} = {4}{5}".format(v1.Cformat(complexformat, sep), sep, op, sep, sep, f3))
        else:
            if doStackOutput:
                print("{0}".format(v1.Cformat(complexformat, sep)))
                print("{0}".format(op))
                print("{0}".format(v2.Cformat(complexformat, sep)))
                print("{0}={1}".format(sep))
                print("{0}".format(v3.Cformat(complexformat, sep)))
            else:
                print("{0}{1} {2} {3}{4}{5} = {6}{7}".format(v1.Cformat(complexformat, sep), sep,
                      op, sep, v2.Cformat(complexformat, sep), sep, sep, v3.Cformat(complexformat, sep)))

    s1 = input(prompt1)

exit(0)

```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

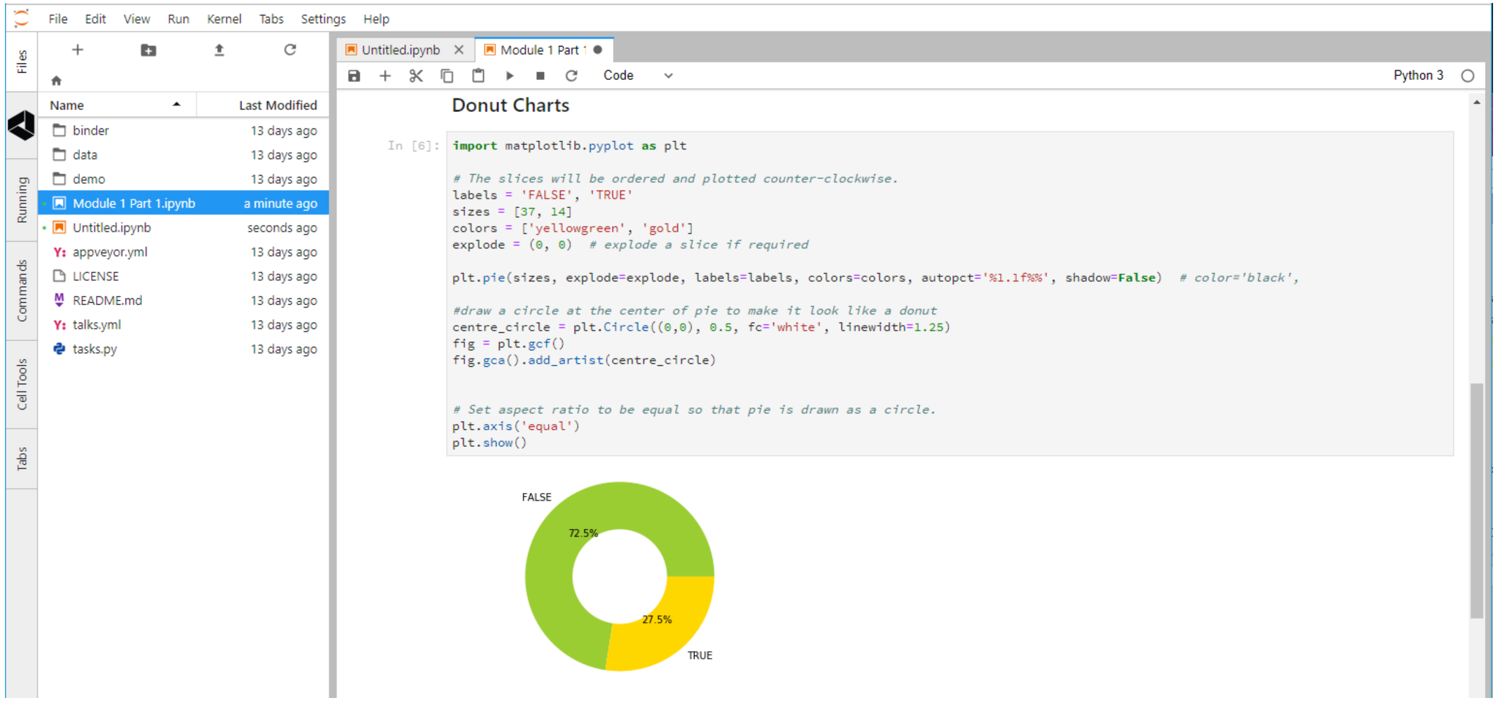
New to Spyder? Read our tutorial

```
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 10:22:32) [MSC v.1900 64 bit
(AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 6.2.1 -- An enhanced Interactive Python.

In [1]:
```

Permissions: RW End-of-lines: CRLF Encoding: ASCII Line: 82 Column: 1 Memory: 63 %

Jupyter Notebooks

- shareable
- environment-based
- interactive or batch execution
- > 40 languages
  - Python, R, Scala, …
- Big Data support
  - Spark



Name	Last Modified
binder	13 days ago
data	13 days ago
demo	13 days ago
Module 1 Part 1.ipynb	a minute ago
Untitled.ipynb	seconds ago
appveyor.yml	13 days ago
LICENSE	13 days ago
README.md	13 days ago
talks.yml	13 days ago
tasks.py	13 days ago

Donut Charts

In [6]:

```
import matplotlib.pyplot as plt

# The slices will be ordered and plotted counter-clockwise.
labels = 'FALSE', 'TRUE'
sizes = [37, 14]
colors = ['yellowgreen', 'gold']
explode = (0, 0)  # explode a slice if required

plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=False)  # color='black',

#draw a circle at the center of pie to make it look like a donut
centre_circle = plt.Circle((0,0), 0.5, fc='white', linewidth=1.25)
fig = plt.gcf()
fig.gca().add_artist(centre_circle)


# Set aspect ratio to be equal so that pie is drawn as a circle.
plt.axis('equal')
plt.show()
```

Generic Data Types

Numeric	Text	Other
integer ● signed, unsigned ● 2, 4 B	character ● 1 B (ASCII) ● 2 B (unicode)	Boolean ● true, false Binary ● 2^n
floating-point (‘float’) ● 4, 8 B ● double = 2 x float	string ● character array ● 0-based *or* 1-based ● null-terminated *or* length-encoded ● usually immutable in OOP	unassigned ● null ● NA undefined ● NA ● +, − infinity
complex ● 2 x double (real, imaginary)	document ● key-value pairs (JSON strings)	BLOB ● images, video ● signals

Data Structures

- lists
  - ordered, mixed-type, mutable
  - append, extend, insert, remove, pop, clear, index, count, sort, reverse, copy
  - comprehensions
- tuples
  - ordered, mixed-type, immutable
  - support packing, unpacking of variables
- sets
  - unordered, no duplicates
- dictionaries
  - key-value pairs (unordered)

Functions

```
def funcName(param1, param2, defArg1 = 0, defArg2 = 100):
        # code here
        return someResult
```

- optional parameters take default arguments if missing from function call
- arguments are assigned to parameters in defined sequence unless named in call
- return statement
  - optional
  - can return multiple items
- scope is inherited from main (but not from a calling function)

Classes

```
class phasor:
    def __init__(self, r=0, p=0):
        self.r = r
        self.p = p
    def real(self):
        return (self.r * math.cos(self.p))
    def imag(self):
        return (self.r * math.sin(self.p))

z = phasor(2.7, 0.4 * math.pi)
```

- 2 underscores before/after init
- the **self** parameter is not explicitly mapped to the function call

Iteration

- while *condition*
- for *iterator* in *list*
- continue
- break
- pass

```
a = ['Mary', 'had', 'a', 'little', 'lamb']
for w in a:
    print(w)
for i in range(len(a)):
    print (i, a[i])
```

Formatted Output

string.format(*args*)

- *string* contains placeholders, formatting codes, and optional annotation text
- *args* are objects (e.g. numerics) to be formatted

Code	Usage
'f'	float
'd'	integer
's'	string

Option	Meaning	Example
'<'	Left-aligned	{0:>20s}
'>'	Right-aligned	{0:<20s}
'^'	centered	{0:^20s}
'0'	sign-aware zero-padding	{:08d}
','	thousands separator	{:,}
'='	space padding after sign	{:=8d}
'+'	include sign for positive numbers	{:+8d}
'-'	skip sign for positive numbers	{:=8d}
' '	leading space for positive numbers	{: 8d}

e.g. print a dictionary of string:float pairs

```
dic = {'a' : 1.23, 'b' : 4.567}
for i in dic:
    print("{0:s} = {1.2f}".format(i, dic[i]))
```

Lab 1.2.1: Python Programming Basics

Purpose:

- To practice basic Python skills

Tools and Resources:

- an IPython console or IDE

Materials:

- ‘lab-1-2-1.py’

SciPy

- ecosystem of open source software for scientific computing in Python
  - numpy, scipy, matplotlib, ipython, jupyter, pandas, sympy, nose
- scipy library
  - numerical algorithms & domain-specific toolboxes

Clustering package (scipy.cluster)
Constants (scipy.constants)
Discrete Fourier transforms (scipy.fftpack)
Integration and ODEs (scipy.integrate)
Interpolation (scipy.interpolate)
Input and output (scipy.io)
Linear algebra (scipy.linalg)
Miscellaneous routines (scipy.misc)
Multi-dimensional image processing (scipy.ndimage)
Orthogonal distance regression (scipy.odr)
Optimization and root finding (scipy.optimize)
Signal processing (scipy.signal)
Sparse matrices (scipy.sparse)
Sparse linear algebra (scipy.sparse.linalg)
Compressed Sparse Graph Routines (scipy.sparse.csgraph)
Spatial algorithms and data structures (scipy.spatial)
Special functions (scipy.special)
Statistical functions (scipy.stats)
Statistical functions for masked arrays (scipy.stats.mstats)
Low-level callback functions

https://www.scipy.org/

NumPy

- the fundamental package for scientific computing with Python
  - a powerful N-dimensional array object
  - sophisticated (broadcasting) functions
  - tools for integrating C/C++ and Fortran code
  - useful linear algebra, Fourier transform, and random number capabilities

import numpy as np

http://www.numpy.org/

Data Types in Python and NumPy

Type	Python	Numpy	Usage
byte byte array	b‘any string’ bytearray()		● immutable ● mutable
integer	int()	● 11 types	● signed, unsigned ● 8, 16, 32, 64 bits, unlimited
floating-point	float()	● 3 types	● 16, 32, 64 bits
complex	complex()	● 2 types	● 64, 128 bits
unassigned	None		● object ● myVar is not None
missing	nan	isnull(), notnull(), isnan()	● float, object

Pandas

- high-performance, easy-to-use data structures and data analysis tools
  - DataFrame class
  - IO tools
  - data alignment
  - handling of missing data
  - manipulating data sets
    - reshaping, pivoting
    - slicing, dicing, subsetting
    - merging, joining
  - time series

import pandas as pd

https://pandas.pydata.org/

Scikit-learn

- biggest library of ML functions for Python
  - classification
  - regression
  - clustering
  - dimensional reduction
  - model selection & tuning
  - preprocessing

```
$ pip install -U scikit-learn
```

or

```
$ conda install scikit-learn
```

http://scikit-learn.org/stable/

Other Python Packages for Data Science

- statsmodels
  - statistical modelling & testing
  - R-style formulae

```
import statsmodels.api as sm
import statsmodels.formula.api as smf
```

- BeautifulSoup
  - reading & parsing XML & HTML data

```
from bs4 import BeautifulSoup
```

- Natural Language Toolkit
  - tokenising, tagging, analysing text

```
import nltk
```

Lab 1.2.2: Scientific Programming

Purpose:

- To introduce techniques and considerations for scientific programming in Python.

Tools and Resources:

- an IPython console or IDE

Materials

- Part 1: High-Performance Programming
  - ‘Lab 1.2.2 Part 1.docx’
- Part 2: Basic Scientific Programming
  - ‘Lab 1.2.2 Part 2.ipynb’

Discussion

- Python Programming
- Jupyter Notebooks
- Evironments
    - *READ THIS:* Installing Python Packages from a Jupyter Notebook (https://jakevdp.github.io/blog/2017/12/05/installing-python-packages-from-jupyter/)

- QUESTIONS

HOMEWORK: Numpy & Pandas

1. Explain the following Numpy methods and create working examples:

- ndim
- shape
- Size
- dtype
- itemsize
- data
- array
- arange
- reshape
- linspace
- random.random
- ones
- sum
- cumsum
- min
- max

HOMEWORK: Numpy & Pandas

2. Explain the following Pandas methods and create working examples:

- DataFrame
- loc
- iloat
- iat
- Index
- sort_index
- MultiIndex
- set_index
- reset_index
- date_range
- sample
- map
- isin
- where
- mask
- copy
- query
- get
- lookup
- difference
- symmetric_difference
- duplicated
- drop_duplicates

Multivariate Integration

3. Answer the following:

If dfmi is a multi-indexed DataFrame, why does

```
dfmi['one']['second'] = value
```

throw a warning while

```
dfmi.loc[:,('one','second')] = value
```

does not?

Advanced Python Programming

- Visualisation in Python
- Parallel Processing in Python
- Performance Optimisation in Python
- Debugging Python programs
- Memory Managing in Python
- Building Python Packages

Visualisation

matplotlib

- histograms
- bars
- curves
- surfaces
- contours
- maps
- legends
- annotations
- primitives

https://matplotlib.org/gallery.html

Seaborn

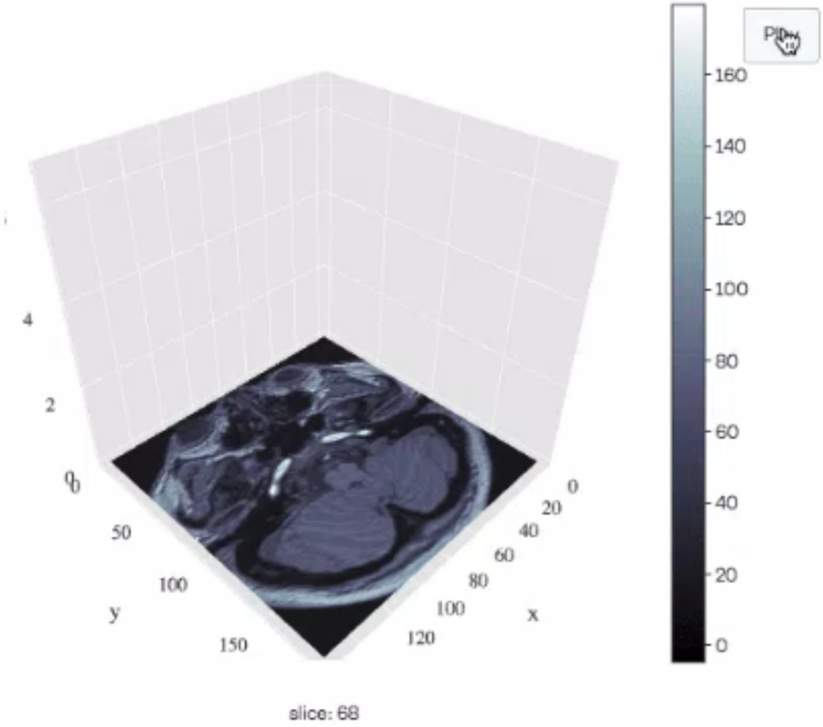
- based on matplotlib
- prettier
- more informative
- more specialised

https://seaborn.pydata.org/examples/index.html

Visualisation – cont'd

Plotly

- open-source (requires sign-up)
- web-based, shareable
- interactive
- online (Jupyter), offline
  - import plotly as py
- + D3, WebGL
  - JavaScript-based



https://plot.ly/d3-js-for-python-and-pandas-charts/

https://d3js.org/

Parallel Processing in Python

multiprocessing module

- uses multiple CPUs (“cores”) to execute sub-tasks in parallel
- submit multiple processes to completely separate memory locations (“distributed memory”)

```
from multiprocessing import Pool
def f(x):
    return x*x
if __name__ == '__main__':
    with Pool(5) as p:
        print(p.map(f, [1, 2, 3]))
```

```
from multiprocessing import Process
def f(name):
    print('hello', name)
if __name__ == '__main__':
    p = Process(target=f, args=('bob',))
    p.start()
    p.join()
```

https://docs.python.org/dev/library/multiprocessing.html

Performance Optimisation in Python

- sort lists using keys and default *sort*() method
- minimise loop nesting, loop computations
  - local variables instead of globals
  - pre-compute the unchanging elements before entering the loop
  - remove function calls
  - remove function references (e.g. set *append = newlist.append* then call *append(w)*)
- stick to built-in methods & algorithms
  - based on optimised C, FORTRAN libraries
- investigate overheads before using non-standard methods

Memory Management in Python

- don't try to access memory management functions in code
  - too fraught!
- use generators with large lists
  - prevents entire list from being loaded at once
- write code that lets the garbage collector do its job:
  - encapsulate temporary data / objects in functions, so that they get destroyed when the function exit

Debugging Python Programs

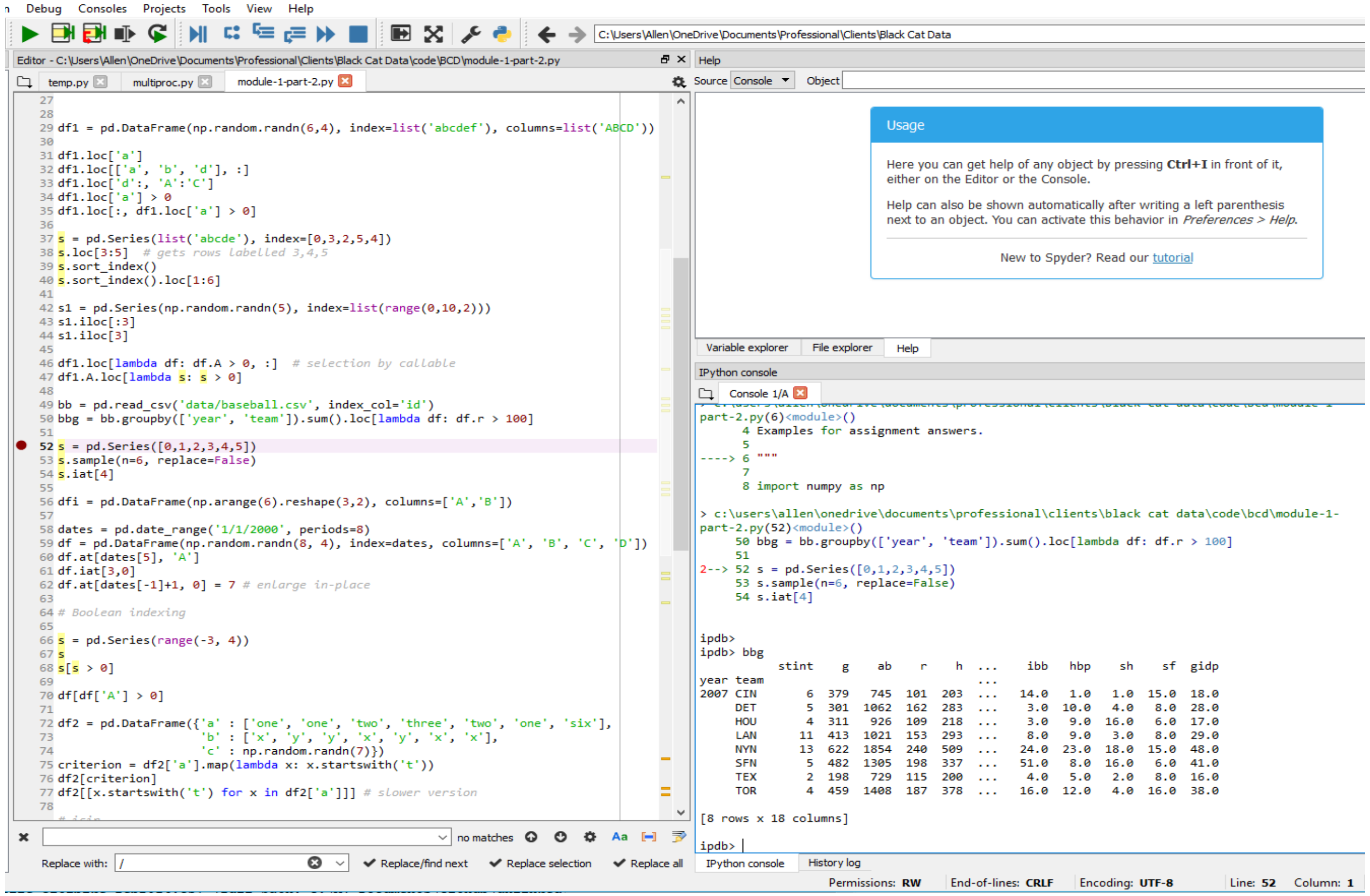
- the pdb module

  import pdb

  import mymodule

  pdb.run('mymodule.test()')

- the IPython debugger



```python


df1 = pd.DataFrame(np.random.randn(6,4), index=list('abcdef'), columns=list('ABCD'))

df1.loc['a']
df1.loc[['a', 'b', 'd'], :]
df1.loc['d':, 'A':'C']
df1.loc['a'] > 0
df1.loc[:, df1.loc['a'] > 0]

s = pd.Series(list('abcde'), index=[0,3,2,5,4])
s.loc[3:5]  # gets rows labelled 3,4,5
s.sort_index()
s.sort_index().loc[1:6]

s1 = pd.Series(np.random.randn(5), index=list(range(0,10,2)))
s1.iloc[:3]
s1.iloc[3]

df1.loc[lambda df: df.A > 0, :]  # selection by callable
df1.A.loc[lambda s: s > 0]

bb = pd.read_csv('data/baseball.csv', index_col='id')
bbg = bb.groupby(['year', 'team']).sum().loc[lambda df: df.r > 100]

s = pd.Series([0,1,2,3,4,5])
s.sample(n=6, replace=False)
s.iat[4]

dfi = pd.DataFrame(np.arange(6).reshape(3,2), columns=['A','B'])

dates = pd.date_range('1/1/2000', periods=8)
df = pd.DataFrame(np.random.randn(8, 4), index=dates, columns=['A', 'B', 'C', 'D'])
df.at[dates[5], 'A']
df.iat[3,0]
df.at[dates[-1]+1, 0] = 7 # enlarge in-place

# Boolean indexing

s = pd.Series(range(-3, 4))
s
s[s > 0]

df[df['A'] > 0]

df2 = pd.DataFrame({'a' : ['one', 'one', 'two', 'three', 'two', 'one', 'six'],
                    'b' : ['x', 'y', 'y', 'x', 'y', 'x', 'x'],
                    'c' : np.random.randn(7)})
criterion = df2['a'].map(lambda x: x.startswith('t'))
df2[criterion]
df2[[x.startswith('t') for x in df2['a']]] # slower version

```

Usage

Here you can get help of any object by pressing **Ctrl+I** in front of it, either on the Editor or the Console.

Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this behavior in *Preferences > Help*.

New to Spyder? Read our tutorial

```
> c:\users\allen\onedrive\documents\professional\clients\black cat data\code\bcd\module-1-
part-2.py(6)<module>()
      4 Examples for assignment answers.
      5
----> 6 """
      7
      8 import numpy as np

> c:\users\allen\onedrive\documents\professional\clients\black cat data\code\bcd\module-1-
part-2.py(52)<module>()
     50 bbg = bb.groupby(['year', 'team']).sum().loc[lambda df: df.r > 100]
     51
2--> 52 s = pd.Series([0,1,2,3,4,5])
     53 s.sample(n=6, replace=False)
     54 s.iat[4]


ipdb>
ipdb> bbg
           stint    g    ab    r    h  ...   ibb   hbp    sh    sf  gidp
year team                              ...
2007 CIN       6  379   745  101  203  ...  14.0   1.0   1.0  15.0  18.0
     DET       5  301  1062  162  283  ...   3.0  10.0   4.0   8.0  28.0
     HOU       4  311   926  109  218  ...   3.0   9.0  16.0   6.0  17.0
     LAN      11  413  1021  153  293  ...   8.0   9.0   3.0   8.0  29.0
     NYN      13  622  1854  240  509  ...  24.0  23.0  18.0  15.0  48.0
     SFN       5  482  1305  198  337  ...  51.0   8.0  16.0   6.0  41.0
     TEX       2  198   729  115  200  ...   4.0   5.0   2.0   8.0  16.0
     TOR       4  459  1408  187  378  ...  16.0  12.0   4.0  16.0  38.0

[8 rows x 18 columns]

ipdb>
```

Permissions: **RW** End-of-lines: **CRLF** Encoding: **UTF-8** Line: **52** Column: **1**

Building Python Packages

- basic steps to create a package
    1. Create a directory and name it after the package
    2. Put the classes in it
    3. Create a **__init__.py** file in the directory

- details
    - lowercase name; unique on PyPi; no hyphens
    - use setuptools.setup to make it installable with pip
    - publish on PyPi
        - http://python-packaging.readthedocs.io/en/latest/minimal.html

Compiling Python Code

- most Python libraries are already based on compiled C, C++, FORTAN code
- useful if custom Python code is computationally intensive

Example: using Numba to compile a function that processes an array parameter

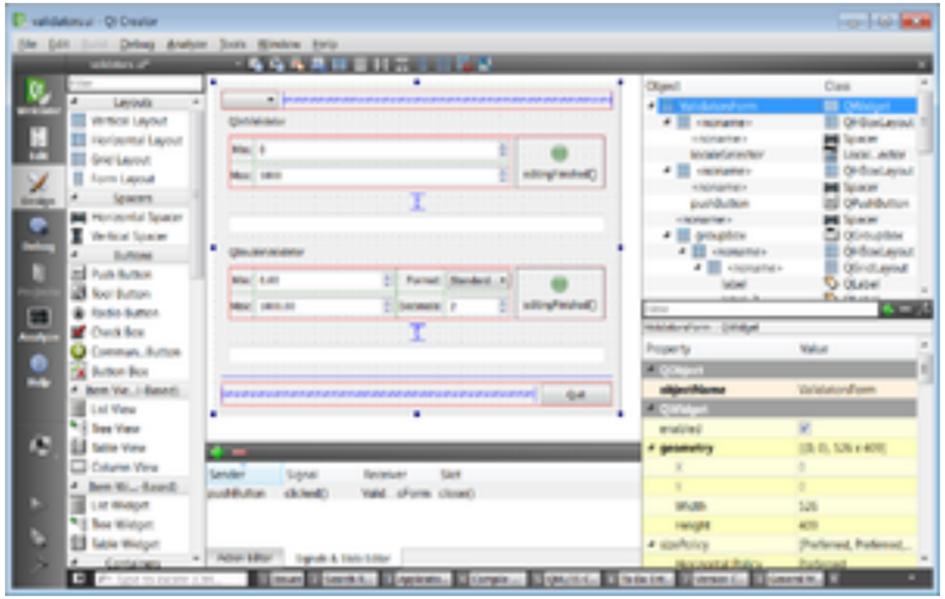
```
from numba import jit
@jit
def sum2d(arr):
    # body
    return myResult
```

https://numba.pydata.org/

http://cython.org/

Creating GUIs for Python Programs

- PyQt
  - a set of C++ libraries and development tools with platform-independent abstractions for:
    - GUIs, networking, OpenGL, XML, user and application settings, location services, Bluetooth, cloud access, etc.
- Glade
- appJar



Qt Creator

Lab 1.2.3: Building a Python Package

Purpose:

- To develop the foundation skills for building packages in Python.

- Tools and Resources:
  - an IPython console or IDE

- Materials
  - ‘lab-1-2-3.py’

Software Engineering Best Practices

- Object-Oriented Programming
- Refactoring
- Coding for readability
- Coding for testability
- Documenting

Object-Oriented Programming

- an *object* encapsulates
  - data (*attributes*)
  - procedures (*methods*)
- a *class* is a prototype for an object
  - *instantiation*: creating an object (in memory) from a class definition

def: **encapsulation**

- attributes of the class should only be accessible by methods of the class
  - get()
  - set()

Creating and Using a Class in Python

```
class myclass:
    def __init__(self, param1, …):
        # initialise class attributes

    def method1(self, ):
        # do something
        return (method1result)



obj1 = myclass(arg1, …)
```

- define class by name
  - initialisation code
    - only **self** is mandatory
    - may use arguments passed from caller
  - define methods
    - only **self** is mandatory
    - may use arguments passed from caller
    - may use attributes
    - may return a value

- invoke class name in assignment to instantiate an object
    - omit **self**

Other OOP Concepts

def: **abstraction**

- data and procedures that do not need to be accessible to the caller should be hidden within the class

def: **inheritance**

- new classes can be based on and extend an existing class

def: **polymorphism**

- a class can implement multiple methods with the same name and function, but which operate on different parameters (type and/or number)

Refactoring

def: Restructuring existing code without changing its behaviour

Examples

- abstract reused code to functions
  - generalise functions (polymorphism?)
- use get, set methods
- simplify structure of nested loops, logic
- minimise use of global variables
  - in Python, this includes all variables defined in main program

Coding for Readability (Maintainability)

Examples

- indent blocks
    - mandatory in Python
- white space
    - between groups of lines
    - between symbols
- comments: inline (to explain logic, return values, etc.)
    - sectional (to explain functional blocks)
    - header (to explain program or module)
        - purpose, authors, date
        - dependences, assumptions

- comments are for coders
    - maintaining or extending your code
- documentation is for users
    - explaining what the application is for and how to use it

Coding for Testability

Examples

- avoid side-effects in functions
- enable testing via compiler flags

```
##define TEST_MODE
#if TEST_MODE
print(“test mode activated”)
#endif
```

- write tests *before* functions
  - specify return type(s) supported
  - test return type(s), validity
  - pass sample data as arguments
  - print result

- test *frequently*
  - avoid marathon coding sessions
- code top-down
  - create wireframe code to test logic, structures
  - fill in the details later

pytest

https://docs.pytest.org/en/latest/

Lab 1.2.4: Object Oriented Programming

Purpose:

- To explore the use of inheritance and polymorphism in Python.

Tools and Resources:

- an IPython console or IDE

Materials

- ‘lab-1-2-4.py’

Discussion

- QUESTIONS

HOMEWORK – Part 1

1. Create a GitHub account (if you don’t already have one).
2. Optional: Install GitHub Desktop

url: https://desktop.github.com

HOMEWORK – Part 2

1. Read Google App Engine product overview

   ref: https://cloud.google.com/appengine/

2. Create a Google Cloud Platform (GCP) account

3. Install the GCP SDK

   ref: https://cloud.google.com/sdk/docs/

4. Read about Google Cloud Shell

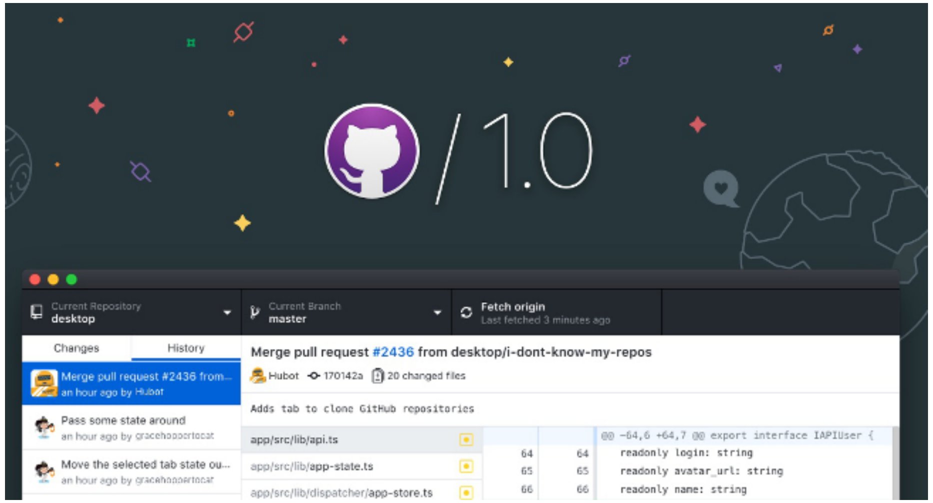
   ref: https://cloud.google.com/shell/docs/features#web_preview

Version Control with Git & GitHub

- Forking
- Cloning
- Communicating issues
- Managing notifications
- Creating branches
- Making commits
- Introducing changes with Pull Requests

Git & GitHub

- web-based, API
- host code, data, resources
- version control
  - integrates with open-source and commercial IDE tools
- share, collaborate
  - branching
- showcase achievements
- command line & desktop versions



Merge pull request #2436 from...
an hour ago by Hubot

Pass some state around
an hour ago by gracehoppertocat

Move the selected tab state ou...
an hour ago by gracehoppertocat

Merge pull request #2436 from desktop/i-dont-know-my-repos

Hubot 170142a 20 changed files

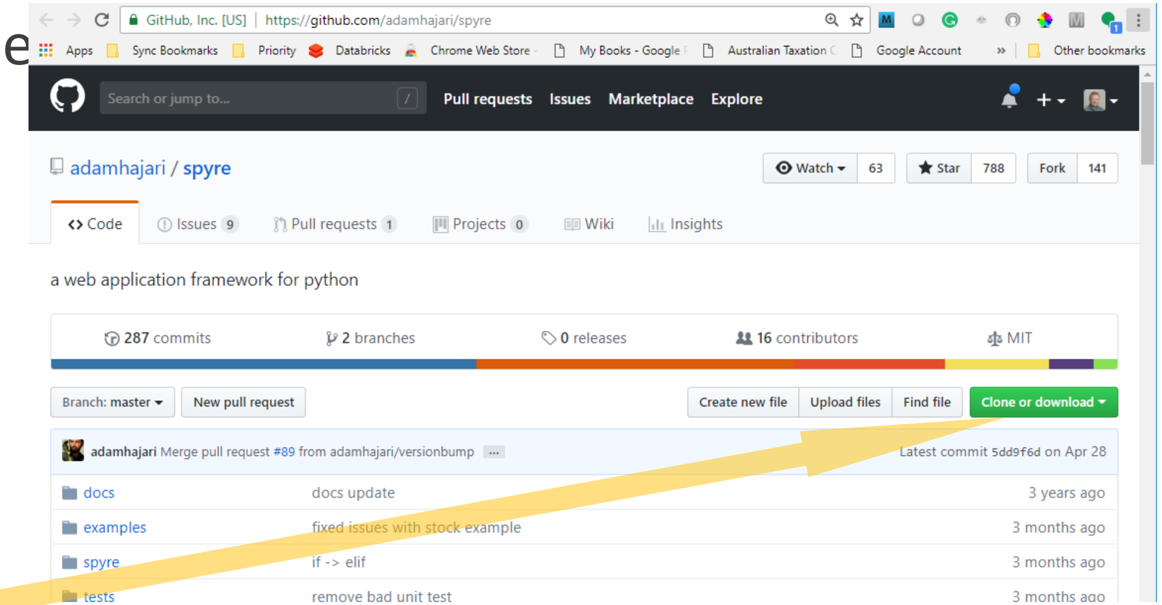
Adds tab to clone GitHub repositories

app/src/lib/api.ts
app/src/lib/app-state.ts
app/src/lib/dispatcher/app-store.ts

```
      @@ -64,6 +64,7 @@ export interface IAPIUser {
64  64   readonly login: string
65  65   readonly avatar_url: string
66  66   readonly name: string
```

GitHub: Forking & Cloning a Repo

- *fork:* make your own copy of someone else’s repo, on GitHub
    1. click <Fork>
- *clone:* create a (working) copy of the repo on your computer



a web application framework for python

adamhajari Merge pull request #89 from adamhajari/versionbump … Latest commit 5dd9f6d on Apr 28

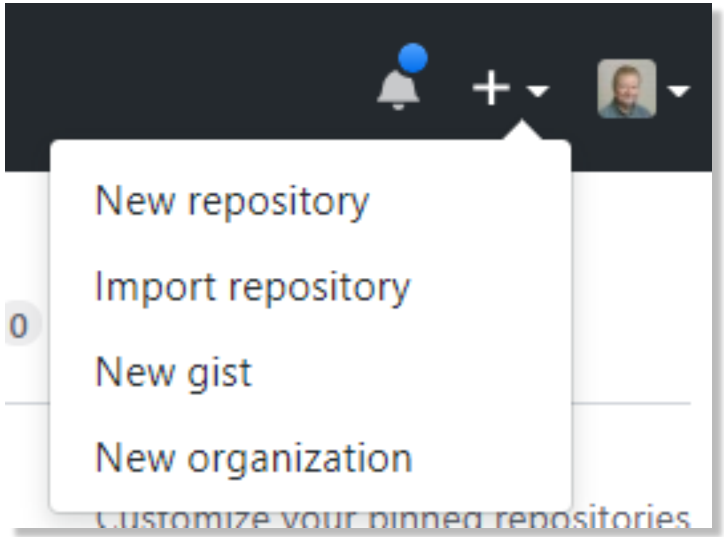
docs	docs update	3 years ago
examples	fixed issues with stock example	3 months ago
spyre	if -> elif	3 months ago
tests	remove bad unit test	3 months ago

- GitHub Desktop procedure:
    1. click <Clone or download>
    2. click <Open in Desktop>
    3. navigate to target (local) folder
    4. click <Clone>

- command-line procedure:
    1. $ cd yourpath
    2. $ git clone https://github.com/yourgithubname/yourgithubrepo

GitHub: Creating a New Repo

- from your GitHub home page
  1. <New repository>
  2. clone the repo to your local drive
  3. copy files, folders into it
  4. commit changes
  5. generate a *pull* request



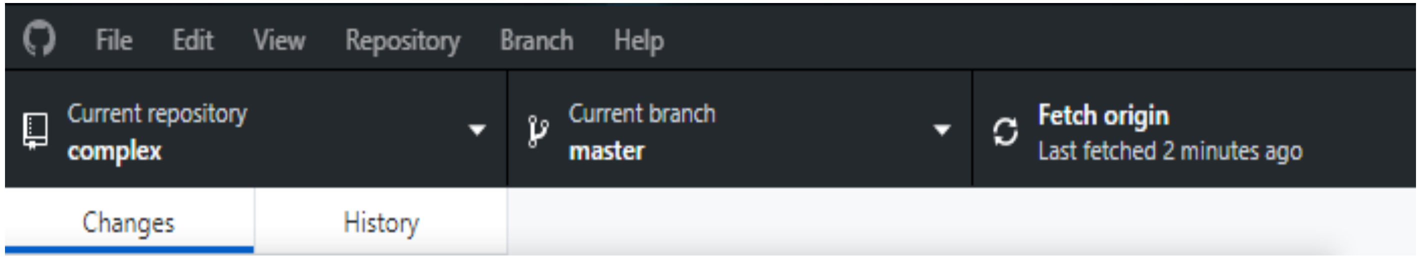
New repository
Import repository
New gist
New organization
Customize your pinned repositories

- Creating a branch
  - to allow development in isolation from source repo
    - protects your changes from changes to source
    - rejoin main branch when ready

GitHub: Refreshing Local Repo from Source

Desktop

- <Fetch origin>



- Ensure you're in the master branch
- Grab the latest changes from the master
- Merge the master changes with your repo

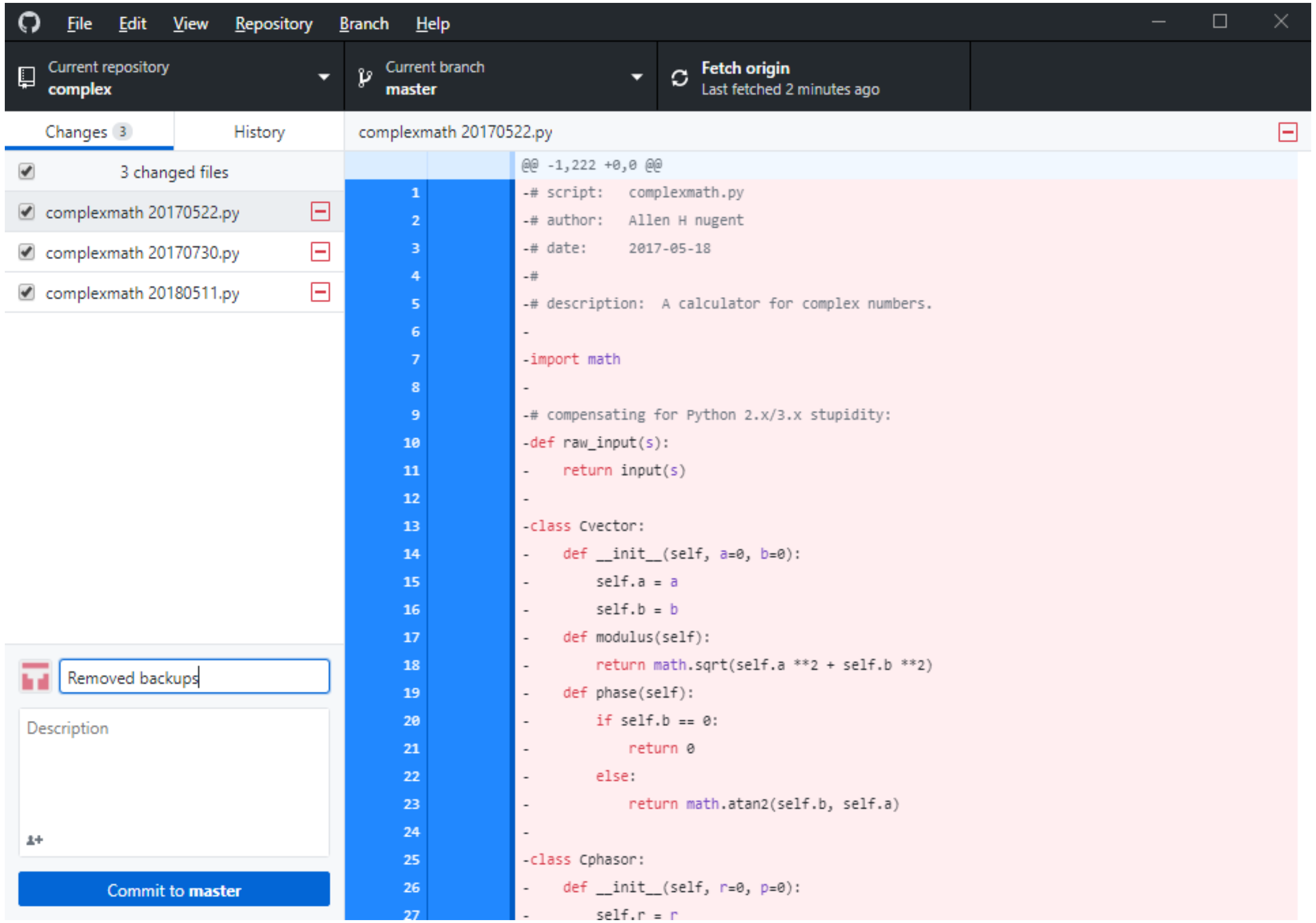
Command-line

```
$ git checkout master
$ git fetch upstream
$ git merge upstream/master
```

GitHub: Commit & Pull Request

Desktop

- enter comments in text box
- <Commit to master>
- Repository > Push
  or
  <Push origin>



☑ 3 changed files

☑ complexmath 20170522.py

☑ complexmath 20170730.py

☑ complexmath 20180511.py

Removed backups

complexmath 20170522.py

```
@@ -1,222 +0,0 @@
-# script:   complexmath.py
-# author:   Allen H nugent
-# date:     2017-05-18
-#
-# description:  A calculator for complex numbers.
-
-import math
-
-# compensating for Python 2.x/3.x stupidity:
-def raw_input(s):
-    return input(s)
-
-class Cvector:
-    def __init__(self, a=0, b=0):
-        self.a = a
-        self.b = b
-    def modulus(self):
-        return math.sqrt(self.a **2 + self.b **2)
-    def phase(self):
-        if self.b == 0:
-            return 0
-        else:
-            return math.atan2(self.b, self.a)
-
-class Cphasor:
-    def __init__(self, r=0, p=0):
-        self.r = r
```

GitHub: Commit & Pull Request

Command-line

- commit

```
$ git status
$ git add filename
$ git add .
$ git commit -m your_comments
$ git status
```

- pull request

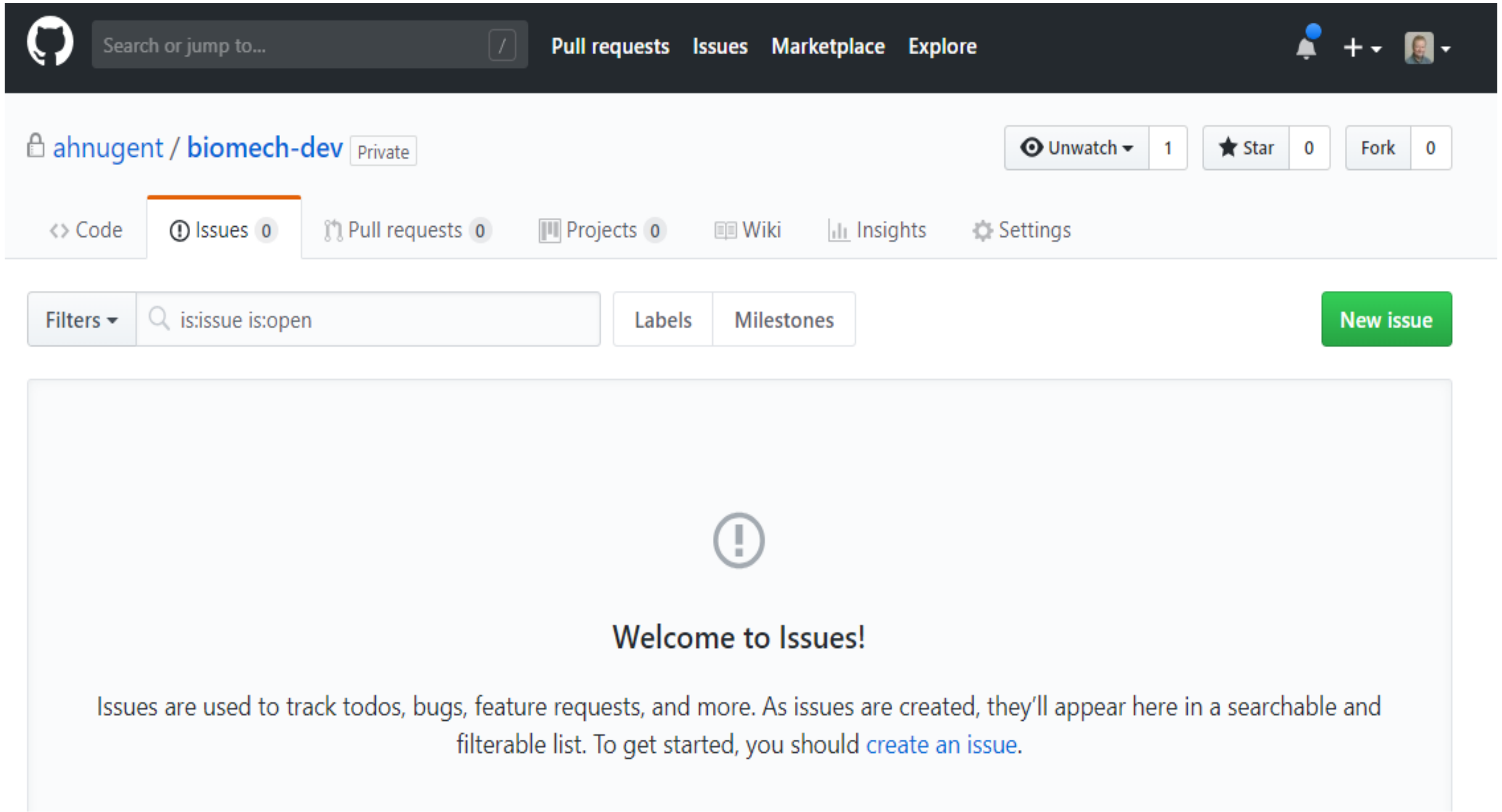
```
$ git push origin master
```

- show changes
- stage one file
- stage all change
- commit file(s), with comments

- origin = your GitHub repo (forked from source repo)
- master = source repo

GitHub: Issues

- track
  - issues / bugs
  - to-do items
  - feature requests
- search
- filter



Welcome to Issues!

Issues are used to track todos, bugs, feature requests, and more. As issues are created, they'll appear here in a searchable and filterable list. To get started, you should create an issue.

GitHub: Notifications

Triggers

- you, a team member, or a parent team are mentioned
- you're assigned to an issue or pull request
- a comment is added in a conversation you're subscribed to
- a commit is made to a pull request you're subscribed to
- you open, comment on, or close an issue or pull request
- a review is submitted that approves or requests changes to a pull request you're subscribed to
- you or a team member are requested to review a pull request
- you or a team member are the designated owner of a file affected by a pull request
- you create or reply to a team discussion

Lab 1.2.5: Setting Up GitHub

Purpose:

- To establish a GitHub repo and develop basic skills for collaborating and maintaining projects.

Tools & Resources:

- GitHub / GitHub Desktop

-

Materials:

- ‘Lab 1.2.5.docx’

Deploying Python Applications to the Web

- Preconfigured virtual machines for machine learning
- Flask, Django, Spyre
- webapp2

Virtual Machines for Machine Learning

- avoids complexity & expense of configuring cloud tools for Deep Learning
  - ML frameworks and tools pre-installed

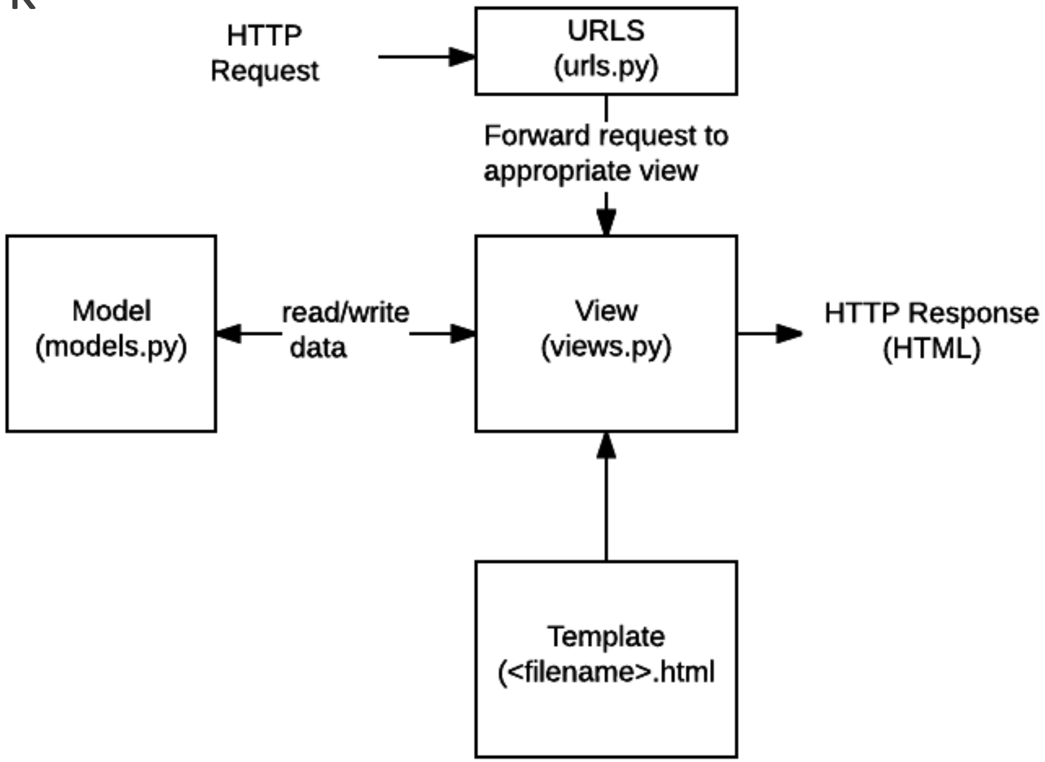
- Google: GCP Deep Learning Virtual Machine Image
  - can be used out of the box on instances with GPUs
  - https://cloud.google.com/deep-learning-vm/docs/

- Amazon: AWS Deep Learning AMIs
  - EC2 GPU or CPU instances
  - https://aws.amazon.com/machine-learning/amis/

Python Web App Frameworks: Django

- fully-featured server-side web framework
- written in Python
- secure
- scalable
- portable

django



https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django

Python Web App Frameworks: Flask

- a microframework for Python

http://flask.pocoo.org/

> create an app:

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"
```

> run app:

```
$ pip install Flask
$ FLASK_APP=hello.py flask run
```

Python Web App Frameworks: Spyre

- a Web Application Framework for providing a simple user interface for Python data projects

> create app:

```
from spyre import server

class SimpleApp(server.App):
    title = "Simple App"
    inputs = [dict( type='text',
                    key='words',
                    label='write words here',
                    value='hello world',
                    action_id='simple_html_output')]

    outputs = [dict( type='html',
                     id='simple_html_output')]

    def getHTML(self, params):
        words = params["words"]
        return "Here's what you wrote in the textbox: <b>%s</b>"
% words

app = SimpleApp()
app.launch()
```

> run app:

http://dataspyre.readthedocs.io/en/latest/

Lab 1.2.6: Deploying Apps to GCP

- Purpose:
  - To practice the deployment of a simple Python app to a web host

- Tools & Resources:
  - Google Cloud Platform (GCP)
    - SDK
    - App Engine (web service)
  - webapp2 framework (contained in GCP App Engine)

- Materials:
  - 'Lab 1.2.6.docx'

Lab: Deploying Apps to GCP – cont’d

1. Bookmark the *Google App Engine Python Standard Environment Documentation*
   ref: https://cloud.google.com/appengine/docs/standard/python/

2. Open App Engine dashboard

3. Select or create a project

Discussion

- Choice of framework vs choice of host
- Resources required by the application
- QUESTIONS

HOMEWORK

1. Explore Mode Analytics
   - create an account:
     https://modeanalytics.com

   - work through the introduction:
     https://help.modeanalytics.com/articles/getting-started-with-mode/