OMIS 30: Intro to Programming (with Python)

Week 1, Class 2

Introduction to Programming
Instructor: Denis Vrdoljak

Learn programming with Python

Goals for the week

- ~~Cover Intros and Intro Material~~
- ~~Get your tools and environment set up~~
- Get familiar with the command line

By the end of this, week you should:

- ~~Have Python installed and your IDE set up~~
- Be able to write a Hello World program in Python, and run it from the command line

Office Hours

Instructor	Days available
Yuan Wang (our TA)	M 2:30-3:30p, W 9-10a
Mike Davis (other section’s instructor)	Tu 9:30-10:20a, Th 12-1p
Denis Vrdoljak	Tu 3:40-4:40p, W 5-6p

Course Topics

- ~~Computer setup~~
- **Shell - ls, mv (and rename), cp, pwd,cd, '..',mkdir, rm, touch, echo**
- **cat, pipe, output redirect (> and >>), 'python --version' (introduce args)**
- Python Basics - print, input, math.
- Pseudo-code, algorithm design, comments
- iterables (lists, sets, dicts, strings)
- Loops, Nested Loops, Recursion
- Flow Control (If, else, elif, try, except)
- Functions
- Strings, upper(), lower()
- indexing and slicing iterables
- lists, extending, appending
- mutability
- Jupyter Notebooks

Command Line

- Why learn command line?
- Used as the basic interface on servers & virtual machines (cloud)
  - Uses less resources (memory/storage) than a graphical interface
  - Less network traffic transmitted back and forth from the server to your computer
- Important to know to navigate on the servers
- Can do a: **man <command>** or **help <command>** to see the help page on that command
- Dos/Windows vs Linux/Unix: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/4/html/Step_by_Step_Guide/ap-doslinux.html

Command Line Commands

- Navigation
  - **ls** - shows the contents of the present folder (**dir** for windows)
    - **ls -alh** - anything after a hyphen is called an option, flag, or switch & modifies the original command
      - a = all files including hidden ones (ones that start with a .)
      - l = long list
      - h = human readable (so a size of 4096 bytes = 4.0K instead)
  - **cd <dir>** = change directory
    - **cd ..** = back one directory
    - **cd .** = this directory
  - **pwd** = print working directory - the full directory name of your current directory (**cd** for windows)

Command Line Commands

- Making files & directories
  - **touch <file_name>** = make a blank file of that name (N/A on windows)
  - **mkdir <directory_name>** = make a new directory of that name
- Deleting
  - **rm <file or directory name>** = removes / deletes that file or directory
    - CAUTION: if you remove it it is gone (no ‘recycling bin’ etc to recover it from!)
    - **rm -r <directory_name>**= removes all the directories under that directory name

Command Line Commands

- Writing to the command line
  - **echo <message>** = print that message back to the screen
  - **clear** = clears the screen (**cls** in windows)
- Writing to a file
  - **> <file_name>** = push the output of that command to that file_name (and overwrites the file)
    - echo Hi > hi.txt
  - **>> <file_name>** = appends the output of that command to that file_name
    - echo How are you? >> hi.txt
- See contents of a file:
  - **cat <file_name>** = view the contents of that file_name

Command Line Commands

- Moving, renaming & copying
  - **mv <source_file> <destination_file>** = moves that file from one spot to the other
    - The file will no longer be in the source directory
    - Rename a file: use the ‘mv’ command to ‘move’ a file to the same spot with a different name
  - **cp <source_file> <destination_file>** = copies the file to another spot
    - The file will be in both locations

Advanced Command Line Commands

- **grep** = search a file for the matching string
  - grep Hi hi.txt
- **|** (pipe) = string two commands together in a sequence
  - cat hi.txt | grep Hi
- **python --version**
  - **--** = ‘long’ option sent to the python command
  - Some commands use the - and some use the --
  - Usually the -- is more spelled out (hence long)
  - **python -V** = the short version and does the same thing

Advanced Command Line Commands

- **vi (vim)** - file text editors
  - Best to hit the INS key to start typing (hit INS twice to go to replace mode)
  - To exit hit ESC - :wq <enter>
    - w = write
    - q = quit
    - If you don’t want to write the changes type :q! (quit and disregard changes)
- **chmod 755 <file_name>** - modify permission to a file
  - First Number 7 - Read, write, and execute for user
    Second Number 5 - Read and execute for group
    Third Number 5 - Read and execute for other
  - Can do 777 for Read, write, and execute for everyone (but less secure)

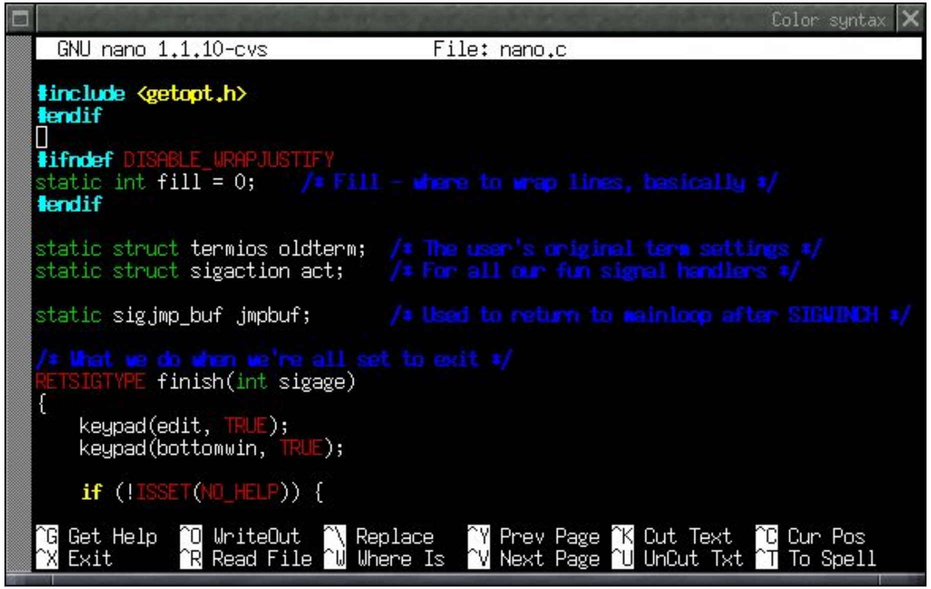
Starting Anaconda Navigator, Spyder, and Jupyter Notebooks

On the command line:

- anaconda-navigator
- jupyter notebook
- spyder

Or, you should be able to find Anaconda Navigator in the Start Menu (PC), or Spotlight search (Mac)

nano (text editor)



```
  GNU nano 1.1.10-cvs                File: nano.c

#include <getopt.h>
#endif

#ifndef DISABLE_WRAPJUSTIFY
static int fill = 0;     /* Fill - where to wrap lines, basically */
#endif

static struct termios oldterm;   /* The user's original term settings */
static struct sigaction act;     /* For all our fun signal handlers */

static sigjmp_buf jmpbuf;        /* Used to return to mainloop after SIGWINCH */

/* What we do when we're all set to exit */
RETSIGTYPE finish(int sigage)
{
    keypad(edit, TRUE);
    keypad(bottomwin, TRUE);

    if (!ISSET(NO_HELP)) {

^G Get Help  ^O WriteOut  ^\ Replace  ^Y Prev Page ^K Cut Text  ^C Cur Pos
^X Exit      ^R Read File ^W Where Is ^V Next Page ^U UnCut Txt ^T To Spell
```

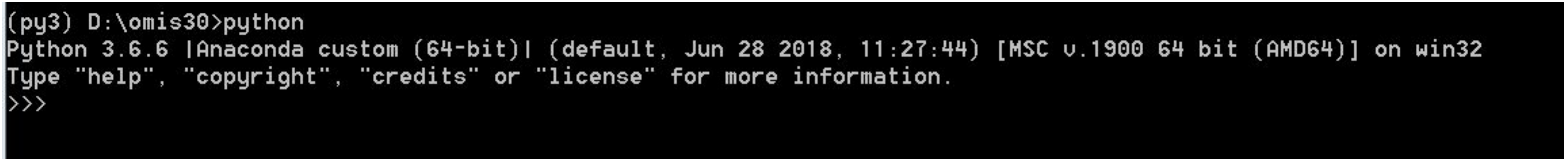
Running Python

3 ways to start Python and input code:

- Command-line interface
- Jupyter Notebook
- Spyder

Python from the command line

- Find and open your terminal, this will be powershell on PC and shell/terminal on mac
- Launch python from the shell, by typing ‘python’
- Should look like this:

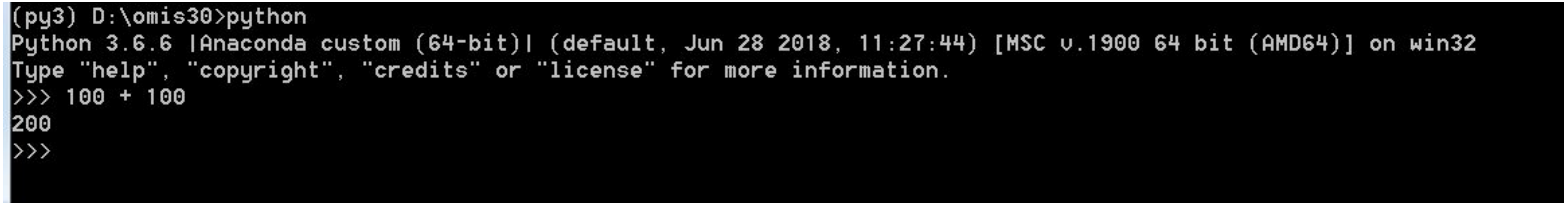


```
(py3) D:\omis30>python
Python 3.6.6 |Anaconda custom (64-bit)| (default, Jun 28 2018, 11:27:44) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

- Make sure the version is Python 3.6

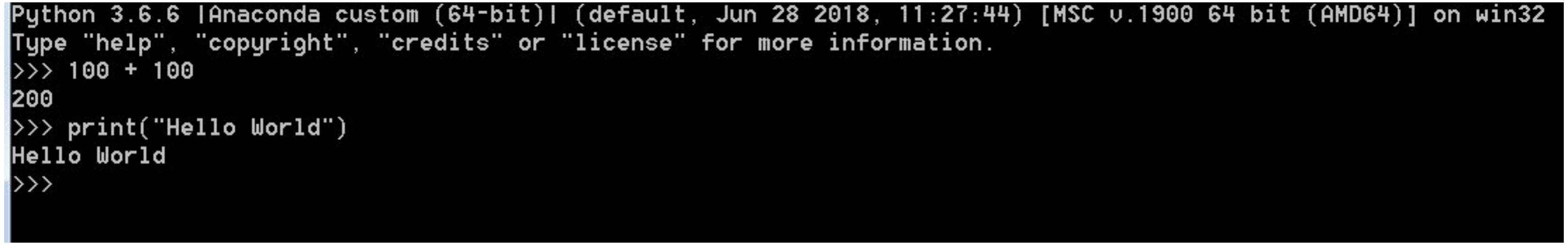
Python from the command line - simple commands

- Type in 100 + 100 and hit enter - should get a response of 200



```
(py3) D:\omis30>python
Python 3.6.6 |Anaconda custom (64-bit)| (default, Jun 28 2018, 11:27:44) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 100 + 100
200
>>>
```

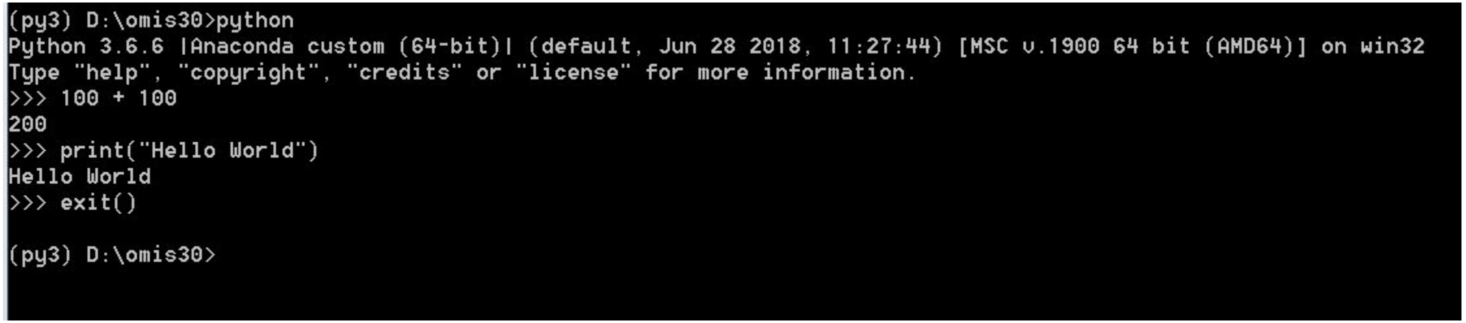
- Next type in a simple print command in Python:
- print(“Hello World”)



```
Python 3.6.6 |Anaconda custom (64-bit)| (default, Jun 28 2018, 11:27:44) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 100 + 100
200
>>> print("Hello World")
Hello World
>>>
```

Python from the command line

- This interpreter is useful for short commands or testing of some code but probably isn't practical for a large program
- To exit: type exit() or CTRL-D



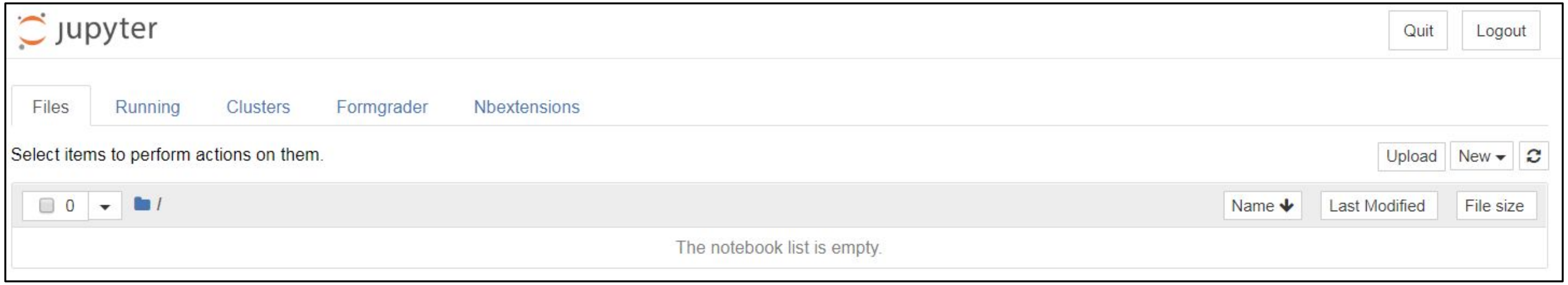
```
(py3) D:\omis30>python
Python 3.6.6 |Anaconda custom (64-bit)| (default, Jun 28 2018, 11:27:44) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 100 + 100
200
>>> print("Hello World")
Hello World
>>> exit()

(py3) D:\omis30>
```

What is a Jupyter notebook (.ipynb)? When do we use them

Opening jupyter notebook

- Type in ‘jupyter notebook’ on your command line
- The notebook server should open in your browser like this:

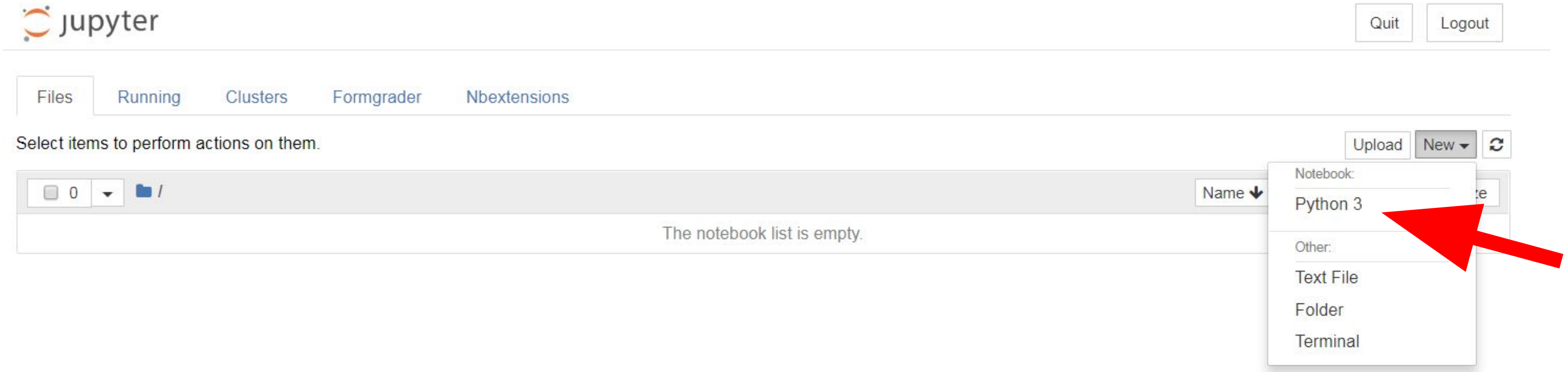


Select items to perform actions on them.

The notebook list is empty.

Opening jupyter notebook

- In the upper right click on New - and then Python 3 from the dropdown menu

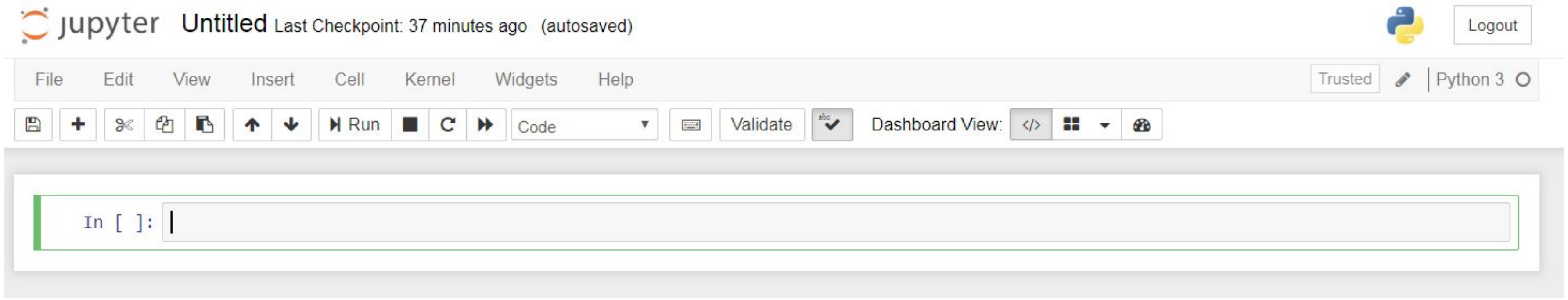


Select items to perform actions on them.

The notebook list is empty.

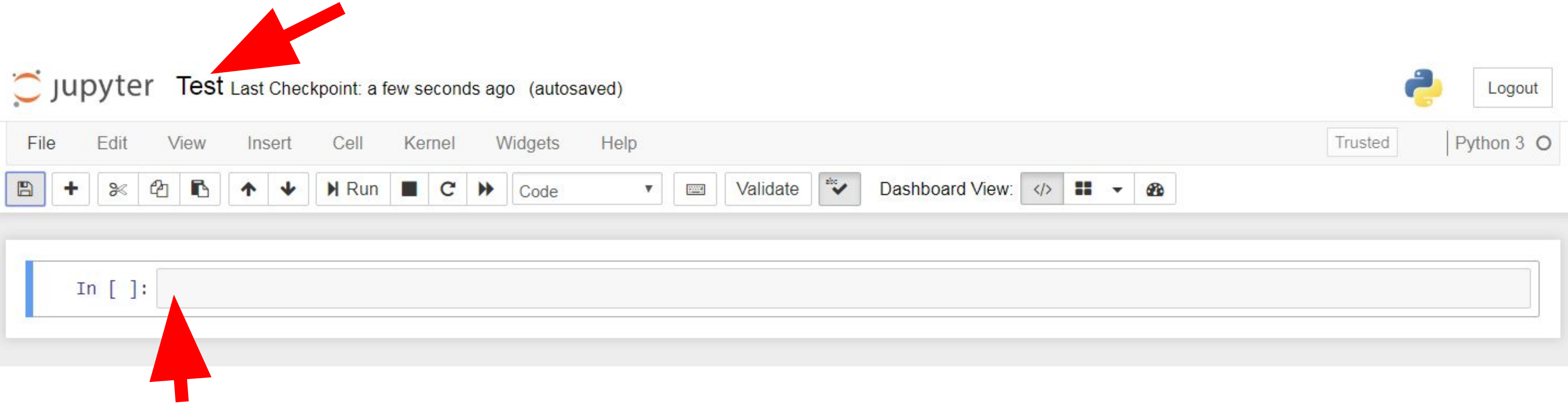
Opening jupyter notebook

- That should bring up a new tab in the browser that looks like this:



Coding in a jupyter notebook

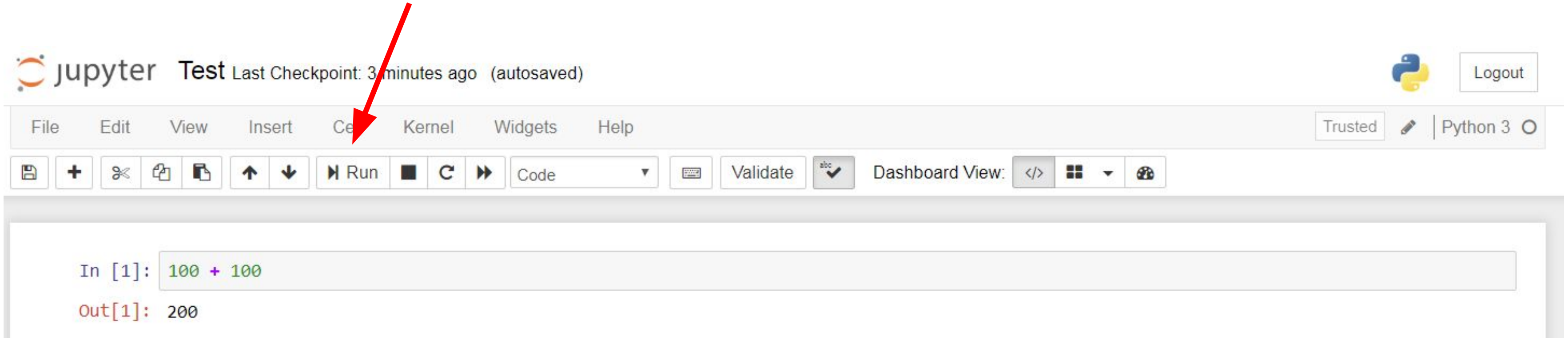
- Change the title: test



Coding Cell

Coding in a jupyter notebook

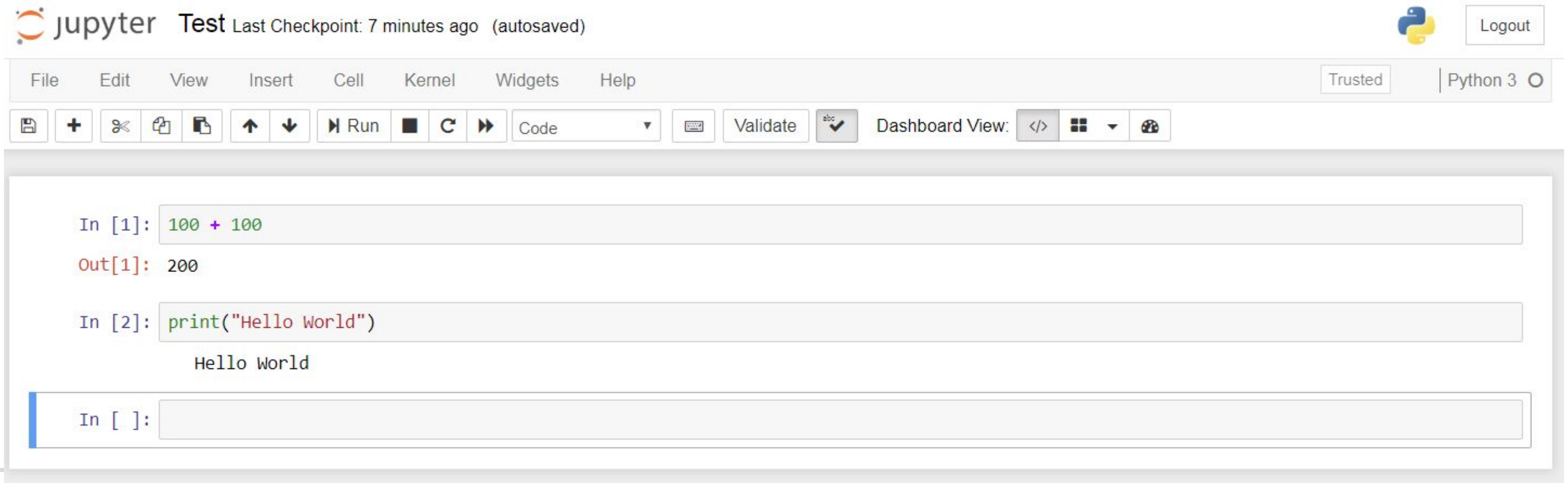
- In the coding cell type in 100 + 100
- Click the 'Run' button (or SHIFT-ENTER) to run the code cell



```
In [1]: 100 + 100
Out[1]: 200
```

Coding in a jupyter notebook

- In the next coding cell type in print("Hello World")
- Again Click the 'Run' button (or SHIFT-ENTER) to run the code cell



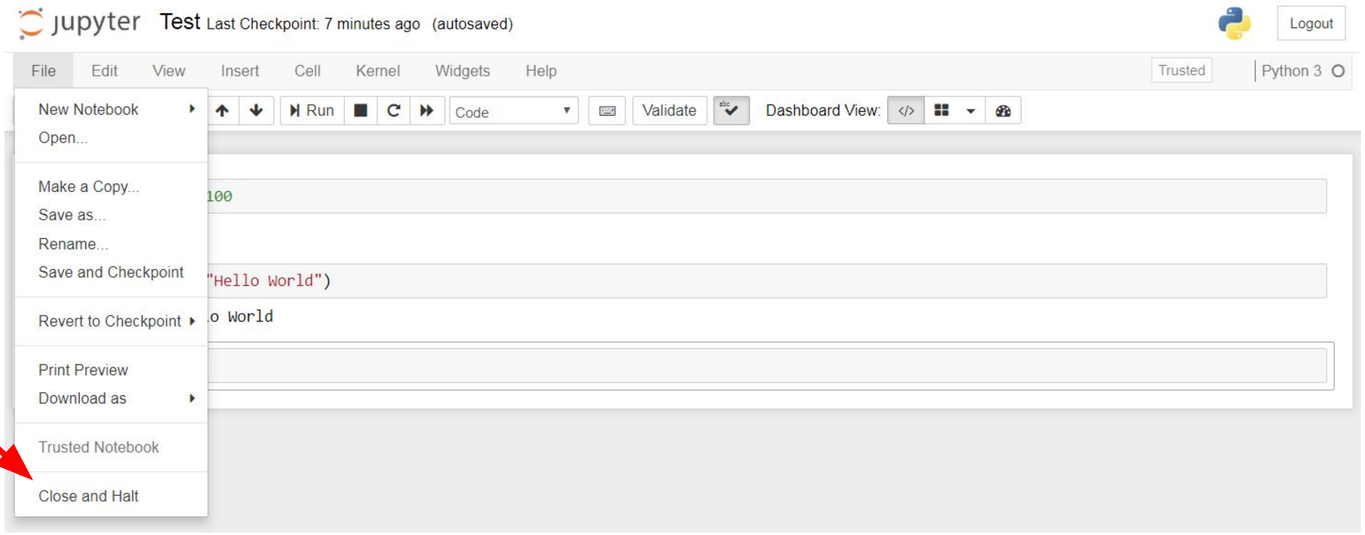
```
In [1]: 100 + 100
Out[1]: 200

In [2]: print("Hello World")
          Hello World

In [ ]:
```

Exiting jupyter notebook

- To exit jupyter notebook - click on the File menu; select Close and Halt
- It is important to exit this way, if you just ‘X’ out of the tab - the notebook will still be running in the background (and this can chew up system resources!)



```
100
"Hello World")
o World
```

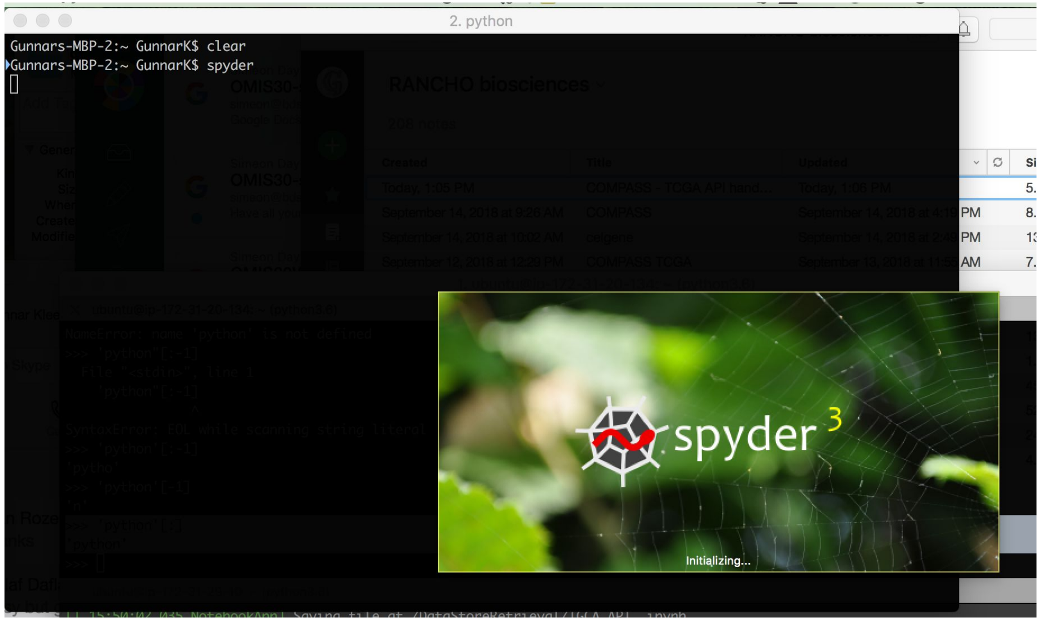
Spyder is an IDE

- What is an IDE and How do we use it?
- Why do we use an IDE vs a notebook?

Opening spyder

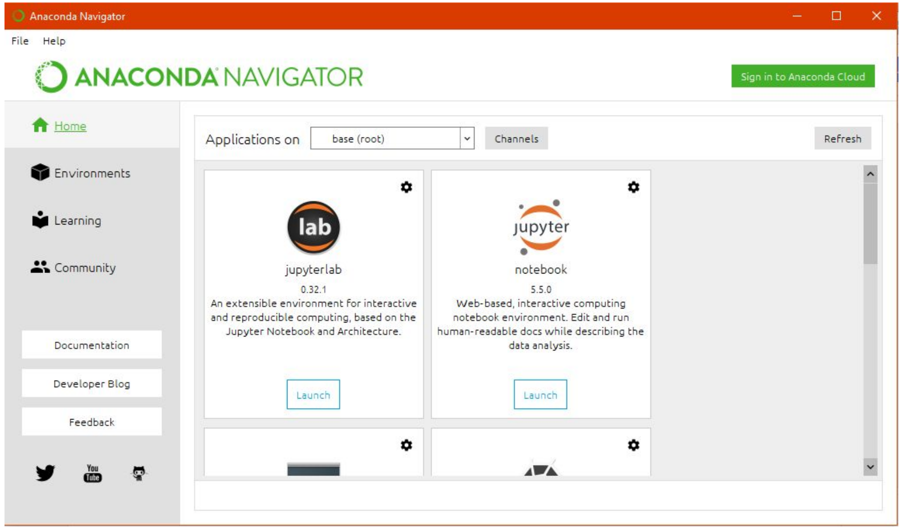
- CMD/Terminal
- Anaconda Navigator

Opening spyder



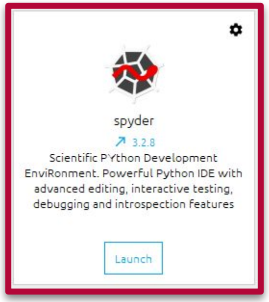
```
Gunnars-MBP-2:~ GunnarK$ clear
Gunnars-MBP-2:~ GunnarK$ spyder
```

Opening spyder

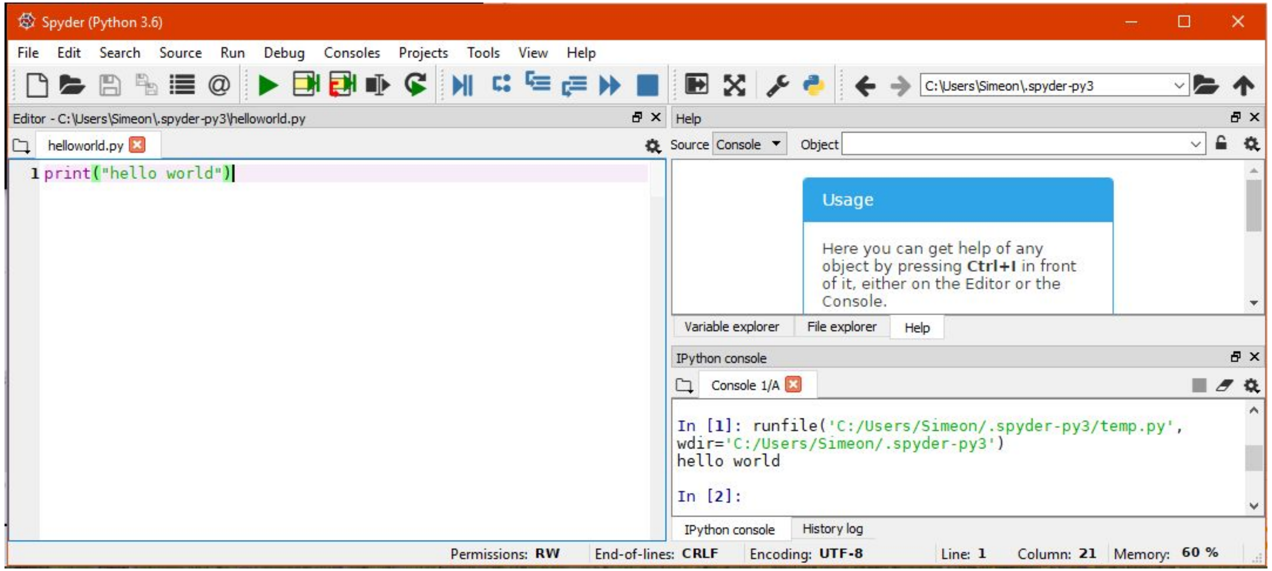


jupyterlab
0.32.1
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

notebook
5.5.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.



spyder
3.2.8
Scientific PYthon Development EnviRonment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features


Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\Simeon\.spyder-py3
Editor - C:\Users\Simeon\.spyder-py3\helloworld.py
helloworld.py
Help
Source Console Object
Variable explorer File explorer Help
IPython console
Console 1/A
IPython console History log

```
print("hello world")
```

Usage

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

```
In [1]: runfile('C:/Users/Simeon/.spyder-py3/temp.py',
wdir='C:/Users/Simeon/.spyder-py3')
hello world

In [2]:
```

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 1 Column: 21 Memory: 60 %

Exiting spyder

- If you're running spyder from your terminal or CMD, you can close the program by typing control+c
- If opened in Anaconda Navigator close by exiting the window

Homework Logistics

- Due: Before the next class
- Turn-in: email (Camino is not setup)
- Extension Policy:
  - With prior approval only.
- Late Grading Policy:
  - Don’t be late! But, if you have, you must have prior approval, or have an excused absence.
- Questions on a grade:
  - Message the Instructor Privately

Homework 1.2

Email this homework to: denisvrdoljak@berkeley.edu

- Hello World, Hello You
  - Write and run Hello World in a Jupyter Notebook
  - Write and run Hello You in a jupyter Notebook (instructions on GitHub)
  - See the wk1hw2.md file in the GitHub REPO for details
  - Email your notebook. (notebooks end with the extension .ipynb)

- Command Line Scripting
  - See the wk1hw2.md file in the GitHub REPO for details
  - Email/submit your script file (file with all your commands)

Full instructions:

- https://github.com/denisvrdoljak/OMIS30_Fall2018/blob/master/wk1/wk1hw2.md

Appendix

- Reserved keywords in Python:
  - https://pentangle.net/python/handbook/node52.html
  - DO NOT USE THESE as VARIABLE NAMES!
- Dos/Windows vs Unix/Linux:
  - https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/4/html/Step_by_Step_Guide/ap-doslinux.html
  - You should be familiar with the basic commands for navigating aroudn the file structure and modifying/creating files/folders. You should be aware of the harder to remember ones (like grep and vi) so you know what to Google when you need them!