Wrap Up

Traditional coding

interactive coding
line-by-line
console or through
the command line

batch coding (scripts)
written in a text editor
run through the
command line

Coding choices

interactive coding
line-by-line
console or through the command line

batch coding (scripts)
written in a text editor
run through the command line

notebooks
code and markdown through
Jupyter, Colab, Rstudio, or others

Coding choices

interactive coding
line-by-line
console or through
the command line

batch coding (scripts)
written in a text editor
run through the
command line

notebooks
code and markdown through
Jupyter, Colab, Rstudio, or others

Learn more about scripts at tomorrow's BONUS LEVEL workshop (or register to get the recording)

Coding choices – what are they good for?

interactive coding
debugging
short tasks that you won't repeat
boots up quickly

notebooks
exploring data
data visualization
working out code
sharing code
teaching code
data science
code that requires human feedback
can also run on Quest for shorter jobs

batch coding (scripts)
long tasks
memory intensive tasks
run in the background
parallel processing
running on external servers like Quest
computational pipelines
writing software

Coding choices – what are they bad at?

interactive coding
hard to see past code
nothing saved

notebooks
can be memory intensive/slow
can get messy
can't be combined in pipelines
don't run in the background

batch coding (scripts)
viewing visualizations
output isn't immediate
comments aren't pretty
harder to debug

Tools mentioned this week

Python

Jupyter Notebook (object)

GitHub

Anaconda

Jupyter Notebook (GUI)

pip

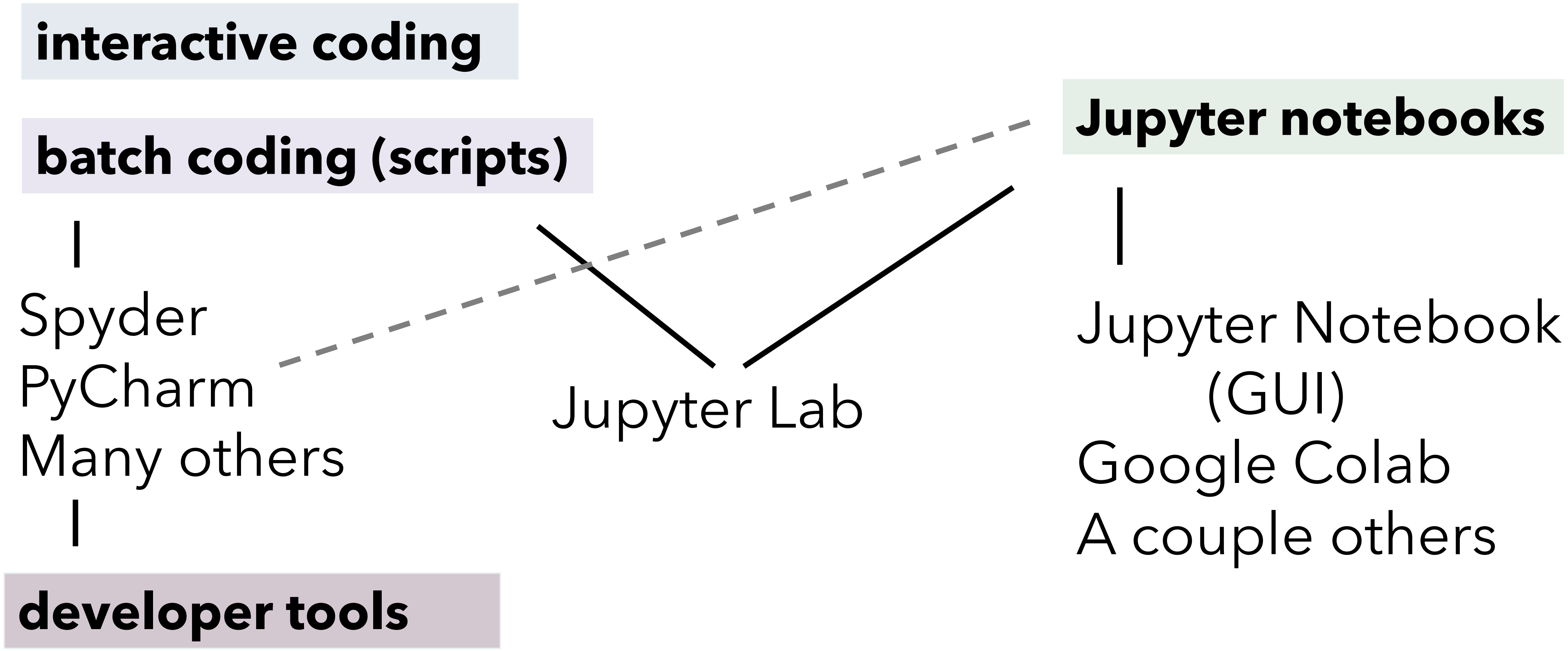
Spyder

Jupyter Lab

Google Colab

PyCharm

Python IDEs


interactive coding
batch coding (scripts)
Spyder
PyCharm
Many others
developer tools
Jupyter Lab
Jupyter notebooks
Jupyter Notebook (GUI)
Google Colab
A couple others

Advantages of Google Colab

More memory than your local machine
Access to GPUs for deep learning
Faster start to open a notebook

But will it always be free?

What is Anaconda?

Anaconda is a company.
They provide many free, open-source tools as well as selling add-on and enterprise-level products.

What is the Anaconda distribution of Python?

Python installation
Software bundle (Jupyter Lab, Notebook, Spyder, etc.)
Links Python to all the software behind the scenes
Python package manager "conda"
"conda" environment manager

What is the Anaconda distribution of Python?

Python installation
Software bundle (Jupyter Lab, Notebook, Spyder, etc.)
Links Python to all the software behind the scenes
Python package manager "conda"
"conda" environment manager

This has greatly improved your life, even though you don't realize it

What is the Anaconda distribution of Python?

Python installation
Software bundle (Jupyter Lab, Notebook, Spyder, etc.)
Links Python to all the software behind the scenes
Python package manager "conda"
"conda" environment manager

You will soon appreciate access to the conda package repository

What is the Anaconda distribution of Python?

Python installation
Software bundle (Jupyter Lab, Notebook, Spyder, etc.)
Links Python to all the software behind the scenes
Python package manager "conda"
"conda" environment manager

You will eventually be really happy about these, too.

Python package repositories

Online collections of free Python modules that are written and vetted by Python users. Each repository is connected with a package manager that handles installations, updates, and dependencies on your local computer.

PyPI/pip (Pip Installs Packages - is associated with Python.org)
conda (uses conda)
conda-forge (uses conda)

Python package repositories

Online collections of free Python modules that are written and vetted by Python users. Each repository is connected with a package manager that handles installations, updates, and dependencies on your local computer.

PyPI/pip (Pip Installs Packages - is associated with Python.org)
conda (uses conda)
conda-forge (uses conda)

If you installed Anaconda, you have pip and conda

You have now been coding in Python for 3 days!
Things to remember:

- 3 days is a very short time
- Python is a very large language
- You can solve a lot of problems using the objects you know combined with loops and if statements
- Your code is already better than you think
- If it works, it works
- You will only get better with practice
- Google, google, google

What we've covered

Objects

- integers
- floats
- booleans
- strings
- lists
- dictionaries
- files

Object concepts

- variables
- indexing strings, lists, dictionaries
- looping through strings, lists, dictionaries
- filtering using if/elif/else statements and booleans
- reading and writing files

What we've covered

Functions

- two types of functions
- how to call functions
- how functions affect mutable vs. immutable objects
- defining custom functions
- importing modules

Logic

- common solutions to logic problems
- applying objects, loops, and if statements to solve problems

What should you learn next?

You can learn from upcoming workshops, or you can work through my Jupyter notebooks on your own.

`BONUS LEVEL` workshop tomorrow

www.github.com/agithasnoname/pythonBootcamp_bonusLevel

- tuples, sets, ranges
- fstrings (easier way to build strings than using +)
- how to run scripts
- practice project

What should you learn next?

You can learn from upcoming workshops, or you can work through my Jupyter notebooks on your own.

`Next Steps in Python Lunch Lessons`

- July 21: Saving Python Objects with json and pickle
- July 28: List Comprehensions (Part I)
- August 4: List Comprehensions (Part II)
- August 11: Working with Dates and Times
- August 18: *args and **kwargs

What should you learn next?

You can learn from upcoming workshops, or you can work through my Jupyter notebooks on your own.

`Introduction to Pandas`

- July 26: Learn how to work with data tables in Python (.csv and excel files)

`Python for Automation` (no Jupyter notebooks)

- July 19: How to use scripts to automate your workflows

Demo for how to open notebooks straight from GitHub on Google Colab

How to practice Python

The best way to practice Python is to use it in your own research, or for your own job.

If you don't have a research project ready to work on, try to assign yourself a task, preferably with a deadline. If you do any grading with students, try to calculate summary statistics on the grades you assign. If you have a data cleaning task that you would normally do in Excel, try to do it in Python. If you work in a lab, ask the post doc or PI if they have a small coding task you could try in Python.

Teaching or helping others is also a great way to improve your skills - if you know someone who is just starting to learn Python, make yourself available to help answer questions, and really try to look up and find the answers.

How to get help

Research Computing Services at Northwestern provides free programming and data consultations, including help debugging code.
Link to RCS consultation request form