

[220 / 319] Programming

Meena Syamkumar

Andy Kuemmel

Learning Objectives

Skills:

- Run Python
- Run Jupyter

Reading: Chapter 1 of Think Python

Learn common Python operators:

- Mathematical (e.g., “+” and “-“)
- Comparison (e.g., “==” and “>”)
- Logical (e.g., “and” and “not”)

Learn about different data types:

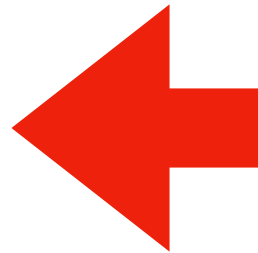
- int, float, str, bool

Learn about boolean logic

Today's Outline

Software

- Interpreters
- Editors
- Notebooks



Demos

Operator Precedence

Demos

Boolean Logic

Demos

What you need to write/run code

An interpreter

- Python 3 (not 2!)
- Some extra packages (installed with pip)

An editor

- Which one doesn't matter much
- idle comes with Python

Jupyter Notebooks

- installed with pip

Interpreter

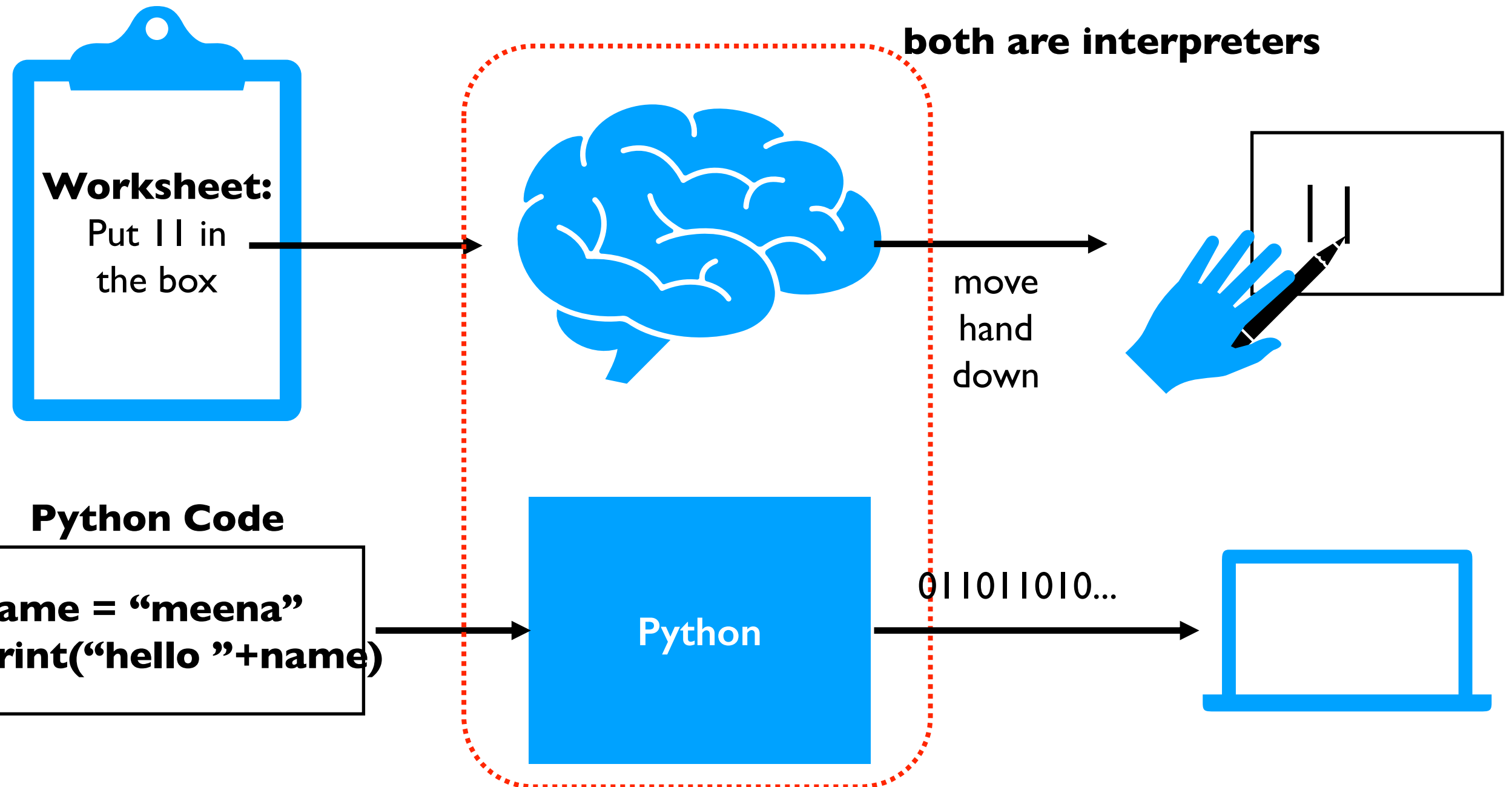
A program that runs a program

- Translates something the human likes (nice Python code) to something the machine likes (ONEs and ZEROs)

Interpreter

A program that runs a program

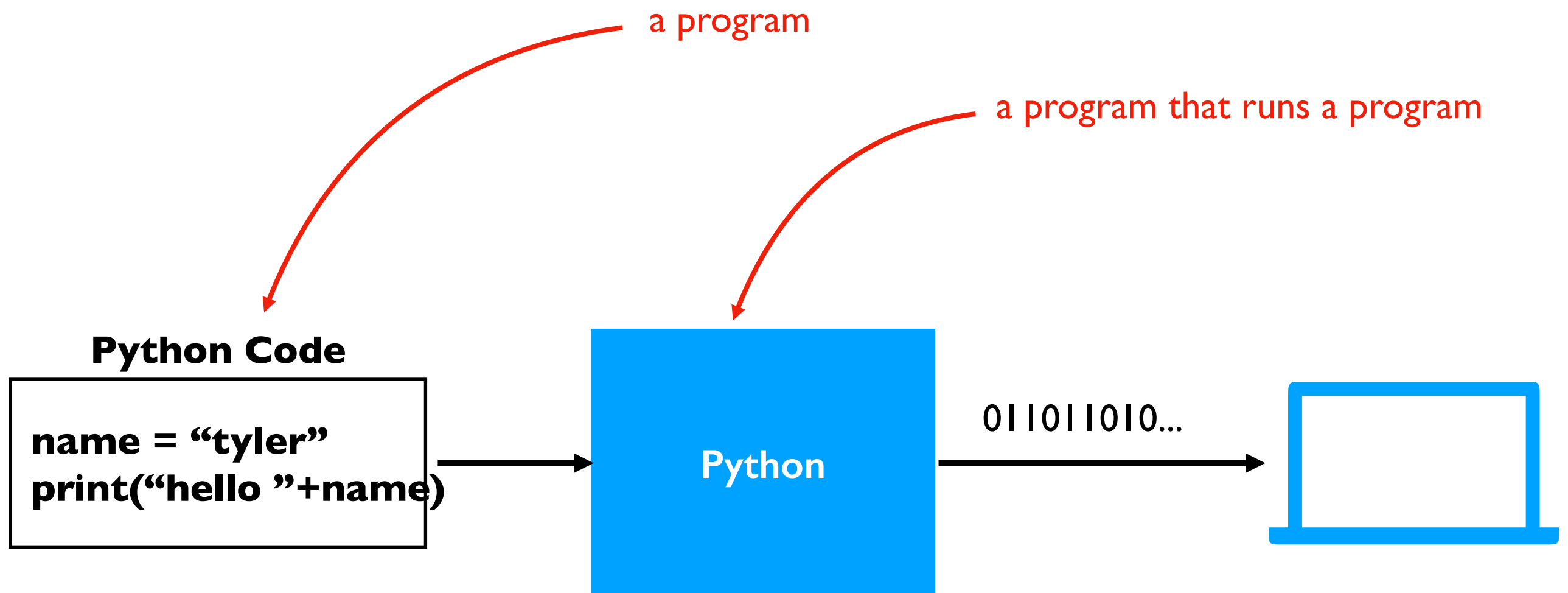
- Translates something the human likes (nice Python code) to something the machine likes (ONEs and ZEROs)



Interpreter

A program that runs a program

- Translates something the human likes (nice Python code) to something the machine likes (ONEs and ZEROs)

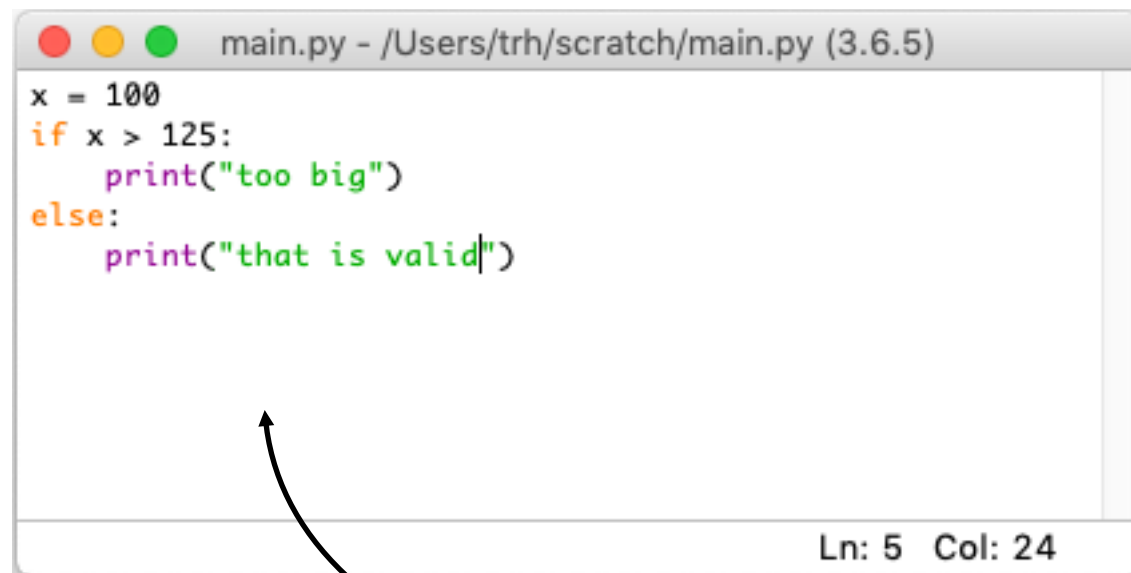


Editor

Program for typing code

- Different editors can open the same **.py** files (Python programs)
(like different browsers can show the same page)

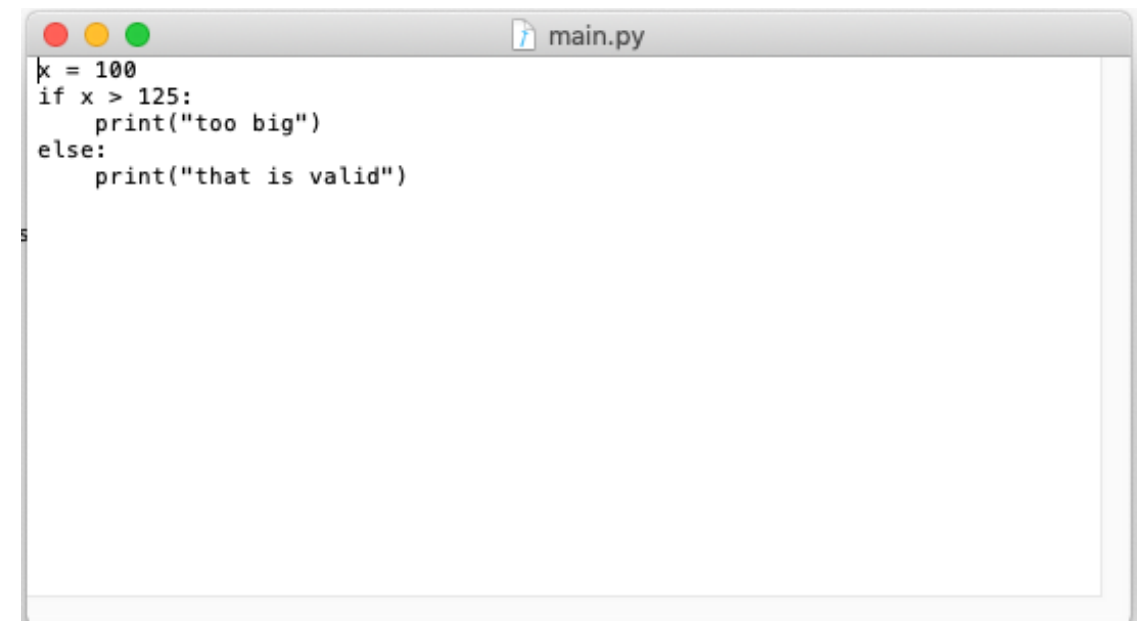
Idle



The screenshot shows the Idle Python IDE window. The title bar reads "main.py - /Users/trh/scratch/main.py (3.6.5)". The code is color-coded: keywords like "if", "else", and "print" are in orange, strings are in green, and the variable "x" is in black. The cursor is at the end of the last line of code. The status bar at the bottom right shows "Ln: 5 Col: 24".

```
x = 100
if x > 125:
    print("too big")
else:
    print("that is valid")
```

TextEdit



The screenshot shows the TextEdit window. The title bar reads "main.py". The code is displayed in a monospaced font without any color highlighting. The cursor is at the end of the last line of code.

```
x = 100
if x > 125:
    print("too big")
else:
    print("that is valid")
```

some editors might colorize code

Jupyter Notebooks

notebooks breakup code into
"cells" containing Python code

...

```
In [35]: #q22
df = pd.read_sql("""
SELECT continent, count() as num_countries
from countries_table
group by continent
ORDER BY num_countries, continent
""", conn).set_index("continent")

ax = df.sort_index().plot.bar()
ax.set_ylabel("number of countries")
ax.set_xlabel("")
```

Tool for mixing analysis code with other things
(e.g., documentation, images, tables, etc.)

Jupyter Notebooks

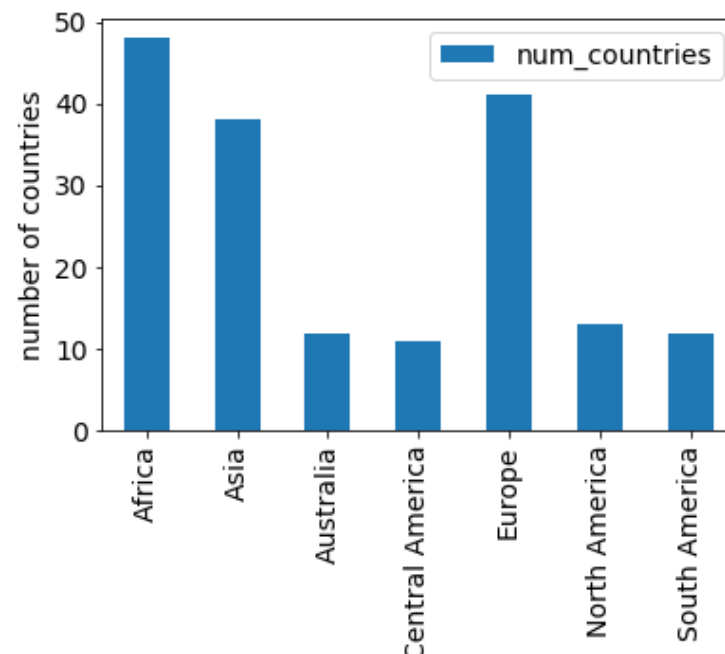
notebooks breakup code into
"cells" containing Python code

...

```
In [35]: #q22
df = pd.read_sql("""
SELECT continent, count() as num_countries
from countries_table
group by continent
ORDER BY num_countries, continent
""", conn).set_index("continent")

ax = df.sort_index().plot.bar()
ax.set_ylabel("number of countries")
ax.set_xlabel("")
```

Out[35]: Text(0.5, 0, '')



visuals produced by the
code are interleaved

.ipynb (Interactive Python Notebook) files are not easy to open in a regular text editor

3 ways we'll run Python

1. interactive mode

```
ty-mac:~$ python
Python 3.7.2 (v3.7.2:9a3ffc0492, Dec 24 2018, 02:44:43)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 1 + 1
2
```

triple arrows mean Python code runs as you type it

2. script mode

the interpreter program is named "python"; run it

```
ty-mac:~$ python my_program.py
```

*the name of the file containing your code (called a "script")
is passed as an argument to the python program*

3. notebook "mode"

```
ty-mac:~$ jupyter notebook
```

open Jupyter in a web browser

we'll do most work in notebooks this semester

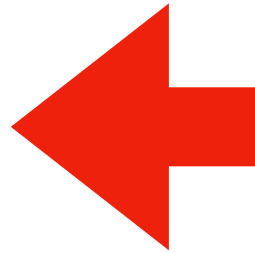
Today's Outline

Software

- Interpreters
- Editors
- Notebooks

Demos

Operator Precedence



Demos

Boolean Logic

Demos

Order of Simplification

Python works by simplifying, applying one operator at a time

$$3 * 3 + 2 * 2 + 16 ** (1/2)$$

$$3 * 3 + 2 * 2 + 16 ** (0.5)$$

$$3 * 3 + 2 * 2 + 4$$

$$9 + 2 * 2 + 4$$

$$9 + 4 + 4$$

$$13 + 4$$

17

Rules

- First work within parentheses
- Do higher precedence first
- Break ties left to right

Operator Precedence

Mathematical	What is it?	Python Operator	simplify first
	exponents	**	
	signs	+x, -x	
	multiply/divide	*, /, //, %	
	add/subtract	+, -	simplify last*
Logic	comparison	==, !=, <, <=, >, >=	
	boolean stuff	not	
	...	and	
Logic	...	or	

these are the ones you should be learning at this point in the semester (there are a few more not covered now)

* one exception is an optimization known as "short circuiting"

Today's Outline

Software

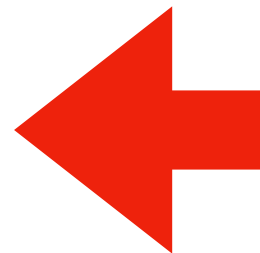
- Interpreters
- Editors
- Notebooks

Demos

Operator Precedence

Demos

Boolean Logic

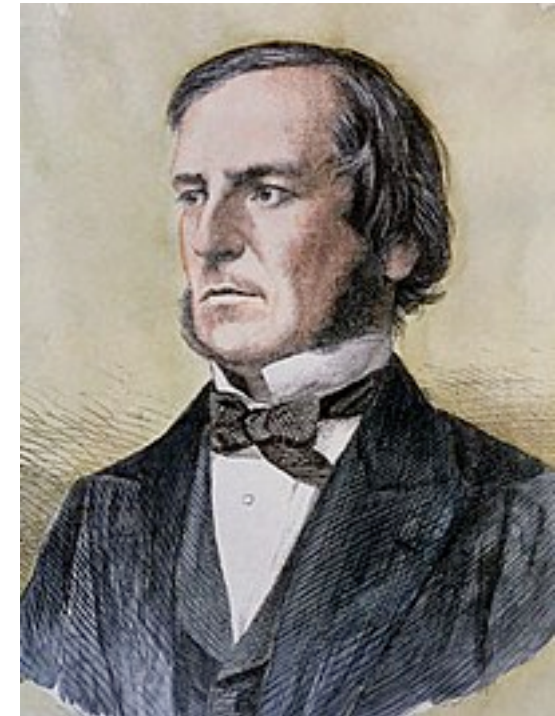


Demos

Boolean Logic

The logic of truth:

- Named after George Boole
- Two values: True and False
- Three operators: **and**, **or**, and **not**



AND

	False	True
False	False	False
True	False	True

OR

	False	True
False	False	True
True	True	True

NOT

False	True
True	False

FALSE!

It's a Saturday **AND**
we're in CS 220 live session

AND

	False	True
False	False	False
True	False	True

OR

	False	True
False	False	True
True	True	True

NOT

	False	True
False	True	False

TRUE!

Project I is due on Wednesday

OR I'll eat my hat



AND

	False	True
False	False	False
True	False	True

OR

	False	True
False	False	True
True	True	True

NOT

False	True
True	False

Control Flow: Remember that conditionals and loops *sometimes* do something.
We'll use bool logic a LOT to control when we do/don't.

AND

	False	True
False	False	False
True	False	True

OR

	False	True
False	False	True
True	True	True

NOT

False	True
True	False

Today's Outline

Software

- Interpreters
- Editors
- Notebooks

Demos

Operator Precedence

Demos

Boolean Logic

Demos

