



# **Python & Notebooks for AI**

# Why Python

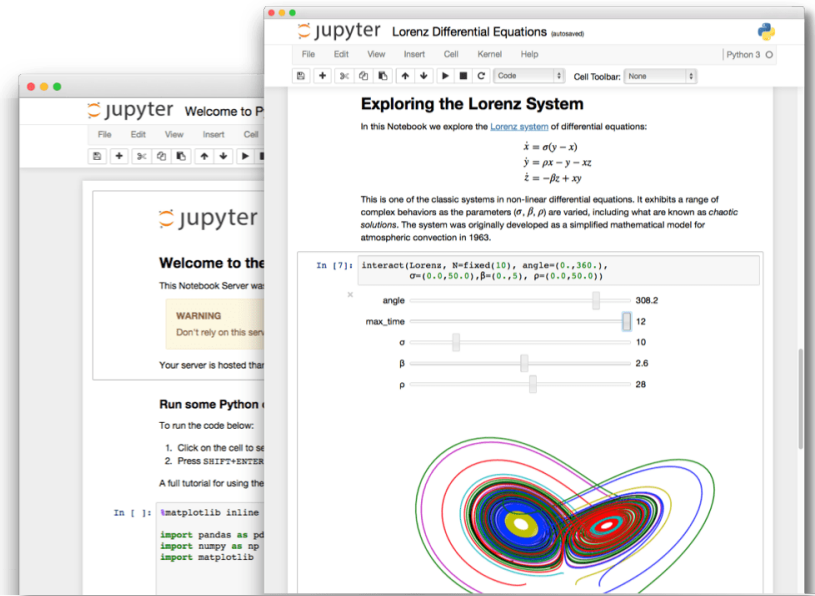
- Python has become the most popular programming language today by some metrics
- It's a great higher-level programming language
  - Easy to learn and use
  - Many interesting and powerful features
  - Large library of modules; easy to install
- Drawbacks: slow (interpreted); few built-in datatypes (no arrays!), poor multiprocessing,...
- Overcome by new modules implementing efficient data structures in C (e.g., [Numpy](#))

# Why Python for AI?

- AI is currently undergoing a [Cambrian explosion](#)
- New ideas appear every week and must be evaluated and explored rapidly to maintain an advantage or even keep up
  - Applies to companies, researchers, students
- Python is great for rapid development
- New neural network packages (e.g., TensorFlow, PyTorch) use efficient C modules for expensive computing parts, visualization, etc.

# Jupyter Notebooks

- Current Python notebook software, replacing iPython
- Open source, browser-based application to create and share *interactive documents* with
  - Live Python code (also R, Julia, Scala, Bash, ...)
  - Visualizations
  - Narrative Text
- Also has a console window and file mover



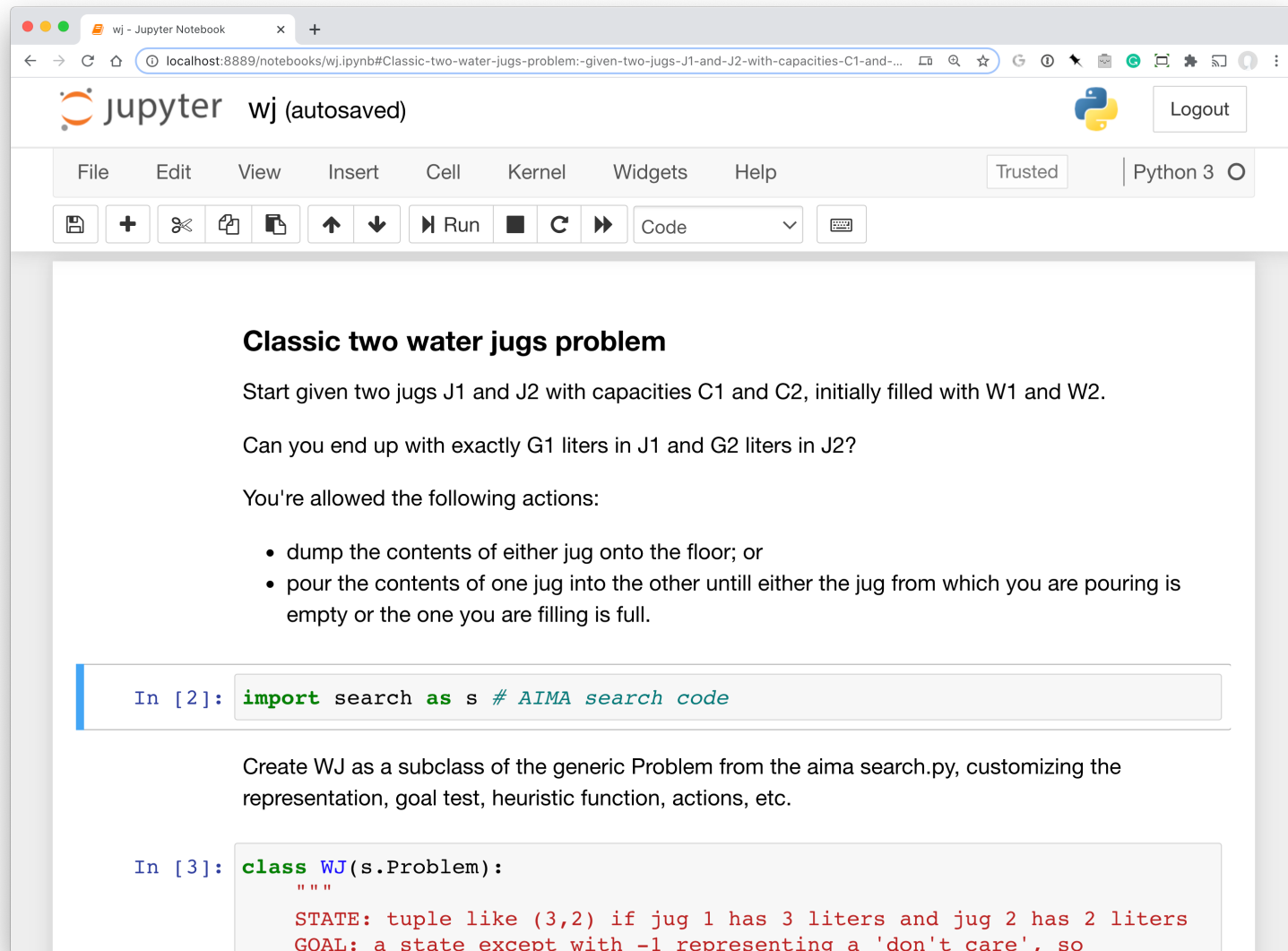


# Many ways to use...

- Install on your computer
  - Pip install jupyter; cd to working directory; execute command “jupyter notebook”; visit browser
- In your browser, access UMBC’s remote Jupyter server, <https://jupyter.rs.umbc.edu/>
  - Requires authorization and then logging in
- Access Google’s free Colab server at <https://colab.research.google.com/>
  - Powerful, harder to work with your local files
- Binder <https://jupyter.org/binder> is another free remote service

# Let's try it

Example: code to solve water jug problems with two jugs



The screenshot shows a Jupyter Notebook window titled "wj - Jupyter Notebook". The browser address bar shows the URL: `localhost:8889/notebooks/wj.ipynb#Classic-two-water-jugs-problem:-given-two-jugs-J1-and-J2-with-capacities-C1-and-...`. The notebook interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for saving, adding cells, and running code, and a status bar indicating the environment is "Trusted" and using "Python 3".

**Classic two water jugs problem**

Start given two jugs J1 and J2 with capacities C1 and C2, initially filled with W1 and W2.

Can you end up with exactly G1 liters in J1 and G2 liters in J2?

You're allowed the following actions:

- dump the contents of either jug onto the floor; or
- pour the contents of one jug into the other until either the jug from which you are pouring is empty or the one you are filling is full.

**In [2]:** `import search as s # AIMA search code`

Create WJ as a subclass of the generic Problem from the aima search.py, customizing the representation, goal test, heuristic function, actions, etc.

**In [3]:** `class WJ(s.Problem):`  
 `"""`  
 `STATE: tuple like (3,2) if jug 1 has 3 liters and jug 2 has 2 liters`  
 `GOAL: a state except with -1 representing a 'don't care', so`