RANDOM PYTHON TOOLS

Eric Prebys
Phy 40
Fall 2025

Tools We Won't Discuss Elsewhere…

- sympy
  - Python symbolic manipulation package.
- pickle
  - Useful for storing objects and data
- OS tools
  - OS information
  - File access
- GUI controls
  - Tkinter (python)
  - ipywidgets (native jupyter)
- Asynchronous processes
- TCP/IP Communication

Symbolic Python (sympy)

- Symbolic python (sympy) is a powerful Computer Algebra System (CAS) tool; that is, it's a tool for handling *symbolic* mathematical operations.
- Good substitute for things like Mathematica, Maple, and the CAS part of Matlab.
  - All of which have their own learning curve and cost money.
  - Sympy is extremely intuitive and easy to use.
- Capabilities include
  - Integration.
  - Differentiation.
  - Analytical and numerical root finding.
  - Plotting
  - Matrix and vector operations
  - LaTeX rendering

(go to demo)

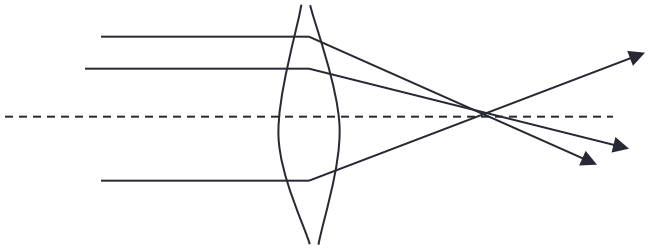
Classical Optics Example

- If we use the ”parallax” approximation for small angles

$$\theta \ll 1 \rightarrow \sin\theta \approx \tan\theta \approx \theta = x'$$

- Then we can represent lenses and “drifts” by “transfer matrices”..

Lens:



Transverse position

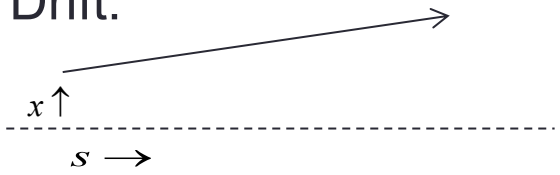
$$x = x(0)$$

$$x' = x'(0) - \frac{1}{f}x(0)$$

$$\Rightarrow \begin{pmatrix} x \\ x' \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -\frac{1}{f} & 1 \end{pmatrix} \begin{pmatrix} x(0) \\ x'(0) \end{pmatrix}$$

Slope wrt axis

Drift:



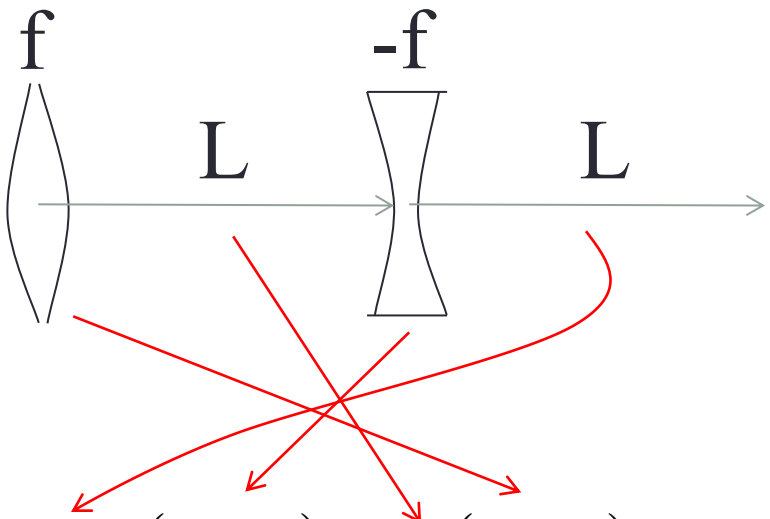
$$x(s) = x(0) + sx'(0)$$

$$x'(s) = x'(0)$$

$$\Rightarrow \begin{pmatrix} x(s) \\ x'(s) \end{pmatrix} = \begin{pmatrix} 1 & s \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x(0) \\ x'(0) \end{pmatrix}$$

Classical Optics Example (cont'd)

- Representing a system of lenses and drifts



$$\Rightarrow \mathbf{M} = \begin{pmatrix} 1 & L \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ +\frac{1}{f} & 1 \end{pmatrix} \begin{pmatrix} 1 & L \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\frac{1}{f} & 1 \end{pmatrix} = \; ?$$

Remember: motion is usually drawn from left to right, but matrices act from right to left!

(go to demo)

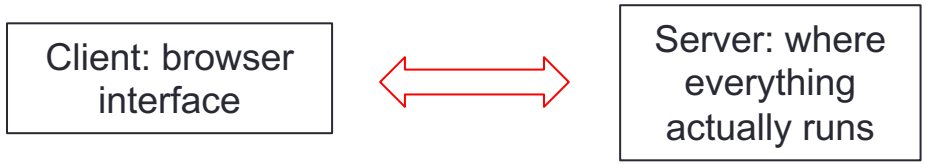
Pickle

- Pickle is a powerful tool for storage and recovery of arbitrarily complex structures in Python
  - This include objects, including both their data and methods
- Similar to serializability in Java.
- No truly equivalent tool in C++

(go to demo)

Accessing System Information

- Recall that jupyter works on a client/server model



- For everything you do in this class, these are both running on the same machine, but this isn't always the case
  - Often jupyter servers will be set up on remote systems to access software ecosystems, computing capabilities, etc
    - Example: Google Colab
  - The server may even be running on hardware
    - Example: Pynq board
- Remember that when you're accessing the system, you're accessing the *server system*.

(go to demo)

Graphical User Interface (GUI)

- The GUI is the biggest difference between standalone python and a jupyter notebook.
- The two main python GUI packages are
  - qt5
  - tkinter (this one seems to be winning)
- These packages open windows from the server rather than the client, so they can only be used if the server and the client are running on the *same computer*
  - This is how we're doing it in this class but that may not always be the case.
- The native GUI tools in Jupyter use "ipywidgets"
  - These run inside the jupyter window like all other graphics
  - They're not nearly as good as qt5 or tkinter
  - They CANNOT run in standalone python

(go to demo)

Asynchronous Operation

- As you may have noticed, you can run cells in a jupyter notebook in a different order, but they still execture sequectially
  - A new cell will not execute until the previous one has completed.
- Sometimes, you might want code to execute asynchronously; that is, run two or more processes at the same time.
  - Eg, you might have one process waiting for data and another one processing the data.
- The package for doing this is “asyncio”

(go to demo)

TCP/IP Communication

- It's easy to communicate between Python processes using TCP/IP communication.
  - Again, this is particularly useful when communicating with hardware.
- The TCP/IP library uses "sockets" for communication.
- The destination of a TCP/IP packet is specified by a 4 byte "IP Address" and a 16 bit "port"
  - Thing of it as a phone number and an extension
  - IP address "127.0.0.1" is a loopback address to the local system
  - Ports are allocated
    - 0-1023: Reserved for essential things like HTTP, FTP, etc
    - 1024-49151: Registered for specific applications and services
    - 49152-65535: Private/temporary ports that can be used for general communication.

(Go to demo)