Intro to Python

PHYS 355

Madeline Galbraith

February 11, 2020

Why to use Python?

- Not compiled
- Easy to install and download additional packages
- Syntax is very simple compared to others
- Python has very robust capabilities in it's packages

Goal at end of today's lecture

Use python to solve HW#2 problem 1d.
Consider a two-state folder in a stop-flow experiment. Changing urea concentration [Urea], the rate of change of the unfolded protein fraction $\frac{dU}{dt} = -f_f U + k_u N$, wheras the rate of change of the folded protein fraction $\frac{dN}{dt} = k_f U - k_u N$. The experiment measures the overal signal associated to the state of the system as a function of time. Show that the decay towards equilibrium is governed by the rate $k_f + k_u$, whereas the equilibrium state is given by $\frac{k_f}{k_u}$. (Hint use the Euler method)

Opening Jupyter notebook

Note we are using this because of it's format where you can run parts of code and put good notes in the same spot

1. If you have jupyter installed
   - Open a command window
   - Go to the folder or directory you want to use
   - type "jupyter notebook" it should open up a web browser.

   - You may also be able to go directly to your packages location and open it that way
2. An online server **cocalc.com**
   - You can create an account now or later if you want
   - Navigate to that page and click "Create New project"

Variables

- Assignment

```
x=2       # This will set x to 2
x=2*y     # This won't work because we haven't defined y y
a = 1     # sets a to 1
print(a)  # need parentheses because this is a function
```

- Checking inequalities

```
x > 3   # True if x>3, false otherwise
x >= 2  # This checks if x is greater than or equal to 2
a== 3   # Checks if a is equal to 3
```

Modules

- There is a package/module for everything
- NumPy - does linear algebra efficiently and is useful in scientific computing
- SciPy - useful for integration, interpolation, optimization, statistics, and solving ODEs
- Matplotlib - Python's plotting library
- Pandas - a data analysis library. Useful for parsing many different file formats quickly and efficiently

NumPy basics

```
import numpy as np
np.sin(1.4)        # np is a shortcut for the full name numpy

import numpy
numpy.sqrt(9)      # but you can also use the full name
np.pi              # not a function so no parentheses

from numpy import tan
tan(3.14/4)        # can directly import functions
```

Lists vs arrays

- Creating a list

```
a = [0.1,40,3] # this is a list
```

- Getting elements from the list (aka indexing the list)

```
print(a[0])   # Indices start at 0 in computer science!
print(a[1]) # This will print 40
```

- Creating a matrix

```
b = [[1,2,3],[4,5,6]]  # A list of lists (aka a matrix)
```

- Indexing a matrix is matrix[row][column]

```
# print the row 1, column 2
print(b[1][2])  # Should print 6
                #remember we start counting from 0

print(b[0]) #prints the first row
```

Lists vs arrays, cont

- Creating a numpy array from a list

```
b_array = np.array(b)    # converts the list to a numpy ar
```

- NumPy arrays have similar ways of indexing

```
print(b[0])   # prints row 0
print(b_array[0]) #prints row 0
```

- Numpy has a shortcut for indexing
  - Note that Loops and indexes actually end one before the last number so if you want the first 10 numbers in a list of 100 you have to do *list[0:10+1]* not *list[0:10]*

```
# print the first 2 elements in both rows
print(b[:][0:2])
print(b_array[:,0:2])
```

Variable names vs objects

- In computer science an object is an abstraction that refers to the underlying structure (An object can be a list, array, plot, etc)
- We sometimes must be careful that a single list is not given two names (this is more important when you start using functions)

```
a=b   # this is a variable assignment
a==b  # Check if they have same elements
a is b # Check if they are the same object
# proof that a and b are the same object
print(b)
a[0][0] = 5
print(b)
# Make a copy that won't change the original
c = b.copy()
c[0][0] = 100
print(b)
print(c)
```

Graphs and vectorization

- NumPy is useful because you can act on all elements in the array at once instead of having to loop over it
- Matplotlib is an extremely powerful plot generator with similar syntax to Matlab

```
import matplotlib.pyplot as plt

# plot as simple scatter plot first
x =[1,2]; y=[1,2]
plt.plot(x,y)

# now plot a sine wave
x_vals = np.arange(0,10,0.1)
y_vals = np.sin(x_vals)
plt.plot(x_vals,y_vals)
```

Putting it all together: Numerical simulation of an ODE

Now we are going to use the Euler method to numerically solve the homework problem from last week.