Special Topics in Standard Python

Ben Rose

brose3@nd.edu

June 8, 2018

Course Goal

Learn by doing

1. Be able to use Python to **do something** with scientific data
2. Teach yourself to learn more
3. Understand the basic Python vocabulary
4. Gain confidence in your programing skills

Todays goals

1. Be able to use `git`'s `pull`, `add`, `commit`, and `push` commands (1)
2. Be able to import and manipulate text or data from a file (1)
3. Know what and `f-string` is and how to use it (3)
4. Be able to describe how and when to use `range`, `enumerate`, and `zip` (3)
5. Be able to use at least one special container form the `collections` class (2, 4)
6. Know how and when to use `assert`, `try`, and `except` (3,4)

Last Time

- `northwind.txt`
  - separate by work?
  - Able to count the occurrence of each word?
- `sunspots.txt`
  - What was hard?

File I/O

- For *prose* -- `f.read()`
- For *tabular data* -- `csv.read()`
    - This is the "best" because you will mostly be working with tabular data

Git

Setup: Add a new remote via URL `https://github.com/ND-Computational-Physics/2018-REU-CMP-assignments`

- `pull` from Classroom
- change `Day-04/change_this_file.txt`
- `add` and `commit` changes
- `push` to your own remote repository

Better printing

```
s1 = 'num: {}'.format(value)
s2 = 'num: {:.4f}'.format(value)
```

Better printing (Python 3.6)

```
s1 = f'num: {value}'
s2 = f'num: {value:.4f}'
```

More Lists and Looping

Lists for math

- If a list contains *only* numbers, we can treat it as a vector:

```
v = [0., 0., -9.81]
```

- If a nested list contains *only* numbers, we can treat it as a matrix:

```
m = [[0, 1], [1, 1]]
```

- This is inefficient, but we don't know a better way yet...

Lists expanded

Create a string from a list: `''.join(l)`

- everything in `l` **must be a string**

List from a sequence

`.append()` is slow, so use this trick:

- `l = [i**2 for i in range(10)]`
- This is extremely useful for a lot of different situations, like file processing:

```
with open(filename, 'r') as f:
    data = [line.split() for line in f]
```

Special for-looping functions

We already know about `range`

`enumerate` gets the index and value

`zip` "walks" through two lists at the same time

Special collections

Aside from lists, tuples, sets, and dictionaries, there are a few special-case
collections we can use

```
import collections

counter = collections.Counter()
colors = ['red', 'blue', 'red',
          'green', 'blue', 'blue']
for word in colors:
    counter[word] += 1

Isotope = collections.namedtuple(
    'Isotope', ['symbol', 'A', 'Z'])
he4 = Isotope('He4', 4, 2)
```

Testing

`assert <condition>`

- stops program execution if condition is false
- useful for making sure we read in the right amount of data, for instance

Protecting yourself

We place operations within `try` -blocks to safeguard ourselves from unwanted
operations (bad division, importing the wrong thing, requiring user input, etc.)

```
try:
    x = 5 / 0
except Exception as e:
    print('divided by zero')
    x = 0.
```

- `Exception` the most generic
  - Will often use `TypeError` or `ZeroDivisionError`
- `e` now contains the Exception message, but not the Exception name.

Practice Time!

- Practice with the handout
    - **Question 4 needs you to teach yourself some new information**

Finish and clean-up

- Import `nothwind.txt` then separate by word
    - Try to count how many times each word appears now using the new tools you have.
- Calculate the average sunspot form `sunspots.txt`
    - Can you count the days who's number of sunspots fell with an arbitrary range?

Read Newman Ch 3 - Intro to Matplotlib & I will update the External Resources section with a few Numpy articles. **Office Hours:** Tuesdays at 3pm, NSH 186