1 Review

Last time:

- Conditionals, loops, functions, errors
- Questions?

Today: we will discuss a miscellany of topics. This will round off the basic intro to Python. After this lecture we'll be discussing very important packages for scientific computing: numpy, scipy, matplotlib, pandas

2 List Comprehensions

- Shorthand for creating a NEW list BASED on an existing list.
- Iterate through list and perform an operation given an optional condition
- Syntax is similar to a for loop `[EXPRESSION for x in ITERABLE if CONDITION]`
- They sometimes make your life easier. Downside: code is harder to read

```
lst = [3, 1, 4, 1]
for i in range(len(lst)):
    lst[i] = lst[i]*2

# with list comprehension
double = [x*2 for x in lst]

# keep only values greater than
new_lst = [x for x in lst if x>1]

# create a list of prime numbers that are between 1 and 1000
primes = [x for x in range(1, 1001) if is_prime(x)]
```

3 2D Arrays

A list is an ordered collection of objects. A list is an object. You can have a list of lists. A list of lists is a two-dimensional list/array (list and array are often interchangeable...)

```
sigma_x = [[0, 1], [1, 0]]

sigma_x = [ [0, 1],
            [1, 0] ]
```

- What is `sigma_x[0]`?
- What is `sigma_x[0][0]`?

- Syntax: `array[row][col]`
- Can have higher dimensional arrays. But if you NEED to use a higher dimensional array to store data, you should probably define a class (object orientated programming) or find a better way to format your data.

4 Tuples

Lists: an ordered collection that allows duplicates

Tuples: an ordered collection with duplicates BUT is immutable (can not be changed after creation).

- Tuples are denotes by parenthesis `()`
- Single tuple denoted by `(element,)`. Comma indicates tuple.
- Tuples are faster than lists (no dynamic memory allocation)

```
tup = (3, 1, 4)

>>> tup[0]
3
>>> tup[0] = 5
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment

tup = ('1-tuple',)

>>> len(tup)
1
```

5 Dictionaries

Dictionary: an unordered, mutable collection of key:value pairs.

- Keys must be unique and immutable (string, tuple, number)
- Why is this data type called a dictionary?

```
dict = {
    'name' = 'ulab',
    'age' = 5
}

# view dictionary contents
print(dict)
```

```
# method returns array of keys
keys = dict.keys()

# checking if a key exists
contains = 'age' in dict

# accessing key value
age = dict['age']

# if KEY does not already exists in the dictionary
# a new key:value pair is created
dict['email'] = 'physics.ulab@gmail.com'

# if KEY already EXISTS in the dictionary
# the key value is updated
dict['email'] = 'physics@ulab.berkeley.edu'

# pop removes the key:value pair and returns the value
dict.pop('email')

# looping through dictionaries
for key in dict:
    print(key)
```

6 Lambda Functions

Setup: functions can have functions as arguments

```
def map(f, iter):
    return [f(x) for x in iter]

def triple(x):
    return e*x

lst = [3, 1, 4]
lst_new = map(triple, lst)

>>> lst_new
[9, 3, 12]
```

Lambda functions:

- Sometimes we want to define small ANONYMOUS functions
- Anonymous: function does not have a name

- Lambda functions are most often used when passing in a function as an argument
- Lambda functions can take any number of arguments, but must return a single expression (no conditionals loops).
- Syntax:

```
lambda [args] : [expression]
```

- Example:

```
lst_new = map(lambda x: 3*x, lst)

# you can assign a variable the value of a lambda function
triple = lambda x: 3*x
add = lambda x, y: x+y

>>> add(1, 2)
3
```

7 Importing

Modules: a Python file containing definitions for variables, functions, and classes (we wont' discuss object oriented programming):

- A module IS a .py file. There is nothing special about a module.
- A module contains code that you or someone else has written to serve a specific purpose.
- We IMPORT modules when we want to use their functionality
- Python has a ton of built-in modules: math, random, sys, etc...
- It is convention to place import statement at the start of your code

Our own module example:

```
# file: mathtools.py

pi = 3.14
e = 2.718

# this function is recursive...but we could also use a for loop
def fact(x):
        if x==0:
                return 1
        return x*fact(x-1)

# calculates e^x using Taylor series
def exp(x):
```

```
        val = 0
        for n in range(30):
                val += (x**n) / fact(n)
        return val
```

Now, let's import!

```
# basic import
import mathtools
>>> mathtools.pi
3.14
>>> mathtools.fact(3)
6

# import specific names
from mathtools import fact
>>> fact(3)
6

# import all names
from mathtools import *
>>> pi
3.14
>>> e
2.718

# import and rename
from mathtools import fact as f
>>> f(3)
6

# Python has a built-in math module with exp defined
import math
>>> math.exp(3.14)
23.103866858722185
>>> mathtools.exp(3.14)
23.103866858722185
```

Packages: a directory containing a collection of modules and sub-packages.

```
/package
|   __init__.py
|   module1.py
|   module2.py
|   /sub-package1
    |   __init__.py
```

```
|   module3.py
```

- A package must contain `__init__.py`
- Importing packages work the same way as importing modules
- Packages are used to share
- We will use the Numpy, Matplotlib, Pandas, and Scipy packages!

pip: a package installer for Python. pip installs packages located on the Python Package Index (PyPI) repository. On the command line: `pip install PACKAGE_NAME`

8 Next Time

- Numpy!