Lecture 2

Last class we introduced a bunch of the Python basics

This class we’re going to talk about some more advanced concepts and techniques that will be very useful to you in your future programming endeavors, whether you program in Python or not

First of these is functions

You’ve probably worked with functions in mathematics

Functions in programming languages are kind of similar

Functions

* I like to think of it as a machine, it takes some inputs and produces some output
* Has anyone ever used a Python function?
* Very sneakily, you all have in fact used Python functions, we used
* Whenever you see the 2 parentheses in Python, one of 2 things is happening
  + Either you’re invoking a function
  + Or you’re creating an object – and really you can think of creating an object as just being a function that creates an object, so really it’s all the same
  + 2 parentheses means you’re taking some inputs, creating some output
* We’ve used a number of functions already in Python without even knowing it
* Can anyone name some of the functions we’ve used?
  + print function – takes some value as an argument, and then it writes to the console
  + int function – takes some value, maybe a string, and tries to spit out an integer
* functions are something we’re going to be using all the time in Python, often other people’s functions
* really critical to building complex programs
* they allow us to take a big complex task, break it into small easy pieces
* one of most important concepts in all of programming
* this is how we define a function in Python

def my\_function(input1, input2):

return input1 + input2

* what happens if I run this?
  + Nothing
* I have only created the machine, I have not invoked it
* or as we say amongst programmers, I have not called the function
* to call the function, I just reference the function name, and I pass the 2 arguments that it requires
  + Example:

In : my\_function(3, 5)

Out: 8

* Some trivia
  + Technically, I don’t have to take any inputs in my function
  + Technically, I don’t have to return any values
* Any questions about how this works?

Quiz:

* Create a function that accepts a list as an argument and when called, returns the product of all the values in the list
* Use your function to calculate the product of the following list:

[232, 543, 963, 12844]

Objects

* Does anyone know what an object is, in programming world?
* Has anyone ever worked with one?
* Actually, all of you have
* In fact, almost everything in Python is an object
* Objects are another important technique for organizing code
* An object is something that has some value or values associated with it, and it has some functions associated with it as well
* An example that we’ve seen already is a string
* In Python, if you’re using the Ipython console, you can see the values (also called attributes) and the functions (aka methods) associated with an object by typing period and hitting tab
* Objects are created by classes, which you can think of as big cookie cutters
  + The class defines the methods and possibly the values associated with the object
  + And then you invoke the class, to create an object defined by the class
* We’re not going to be creating our own object classes, I mention this because a big part of programming is just knowing the language, and you will hear people talk about objects and methods and attributes, and now you know what they mean

Modules

* The last sort of abstract programming concept we’re going to talk about are modules
* Modules are just Python programs sitting out there
* A very common pattern is that people will create a bunch of functions and classes to do useful things and they’ll save them in a Python program, and then they’ll import those functions into a different program later when they need to use them
* To do this:
  + First save our little python program to somewhere we can find
  + Now we need to tell Python where to look, we can do
    - Tools > PYTHONPATH Manager > Add path
  + Now, if we ever want to use this function again, we just do:

from my\_little\_module import my\_function

we can use the function

* Modules are, guess what, objects too
* You can use the period tab trick to see what’s in them

Text Processing in Python:

* Enough with our abstract programming stuff, let’s get back to our task at hand, which is autocoding
* One of the many things that makes Python so great is that it has lots of great tools for string manipulation
* We saw some last time, we’re going to look at some more now
* handy string functions and operations:
  + string.lower()
    - “Foot was broken”.lower() = ‘foot was broken’
  + string.upper()
    - “Foot was broken”.upper() = ‘FOOT WAS BROKEN’
  + string.split(‘ ‘)
    - Example:

In : ‘What a day!’.split(‘ ‘)

Out: [‘What’, ‘a’, ‘day!’]

* + string in <iterable>
    - Example:

In: “foot” in [“employee”, “injured”, “foot”, “today”]

Out: True

* + - Example:

In: “foot” in [“employee”, “hurt”, “arm”, “playing”, “football”]

Out: False

* + string is in fact an iterable itself
    - can grab the pieces of a string just like we did with lists
    - can even loop through the string if you want to
* regular expressions:
  + import re
  + pattern – a string that defines a sequence of characters to match
  + re.findall(pattern, string)
    - returns a list of the parts of the string that match the pattern
  + Various patterns
    - pattern that matches any alphanumeric character and underscore:
      * \w
      * equivalent to: [a-zA-Z0-9\_]
      * Example:

In: re.findall(‘\w’, ‘a1 b2’)

Out: [‘a’, ‘1’, ‘b’, ‘2’]

* + - modifies the preceeding pattern to match one or more repetitions of itself
      * +
      * Example:

In: re.findall(‘\w+’, ‘a1 b2’)

Out: [‘a1’, ‘b2’]

* + - matches any character except a new line
      * .
      * Example:

In: re.findall(‘.’, ‘a1 b2’)

Out: ['a', '1', ' ', 'b', '2']

* + - matches individual words
      * \w+
      * Example:

In: re.findall(‘\w+’, ‘He fell down.’)

Out: [‘He’, ‘fell’, ‘down’]

* + - matches numbers
      * \d+
      * Example:

In: re.findall(‘\d+’, ‘He injured 2 fingers while lifting 10 lbs.’)

Out: [‘2’, ‘10’]

Quiz 2

So far we’ve focused on very small chunks of data, things you type in yourself

What we’re going to do now is talk about how to work with larger datasets

In particular, how to work with whole tables of data in Python

Python actually offers a few different ways to do this

I’m going to introduce one of the most basic ways of doing this

First we have to introduce another basic Python data structure, the dictionary

Dictionaries

* Used to store structured data – think of a dictionary as a row of data in a table
* Consists of key, value pairs
* Creating a dictionary
  + Empty:
    - state\_name = {}
  + Non empty:
    - state\_name = {‘01’: ‘Alabama’, ‘02’: ‘Alaska’, ‘04’: ‘Arizona’}
* Adding values to a dictionary
  + state\_name[‘05’] = ‘Arkansas’
* Retrieving values from a dictionary
  + print(state\_name[‘05’])

Files

* Open a file for reading
  + f = open(‘my\_file.txt’, ‘rb’)
* Read a line from an open file
  + f.readline()
  + f.close()
* Iterate through all the lines in a file
  + for line in f:
    - print(line)
* Open a file for writing
  + f = open(‘my\_file.txt’, ‘wb’)
  + f.write(‘I just wrote a line to my file’)
  + f.close()
* Trick with files
  + raw strings and special characters

List of Dictionaries

* using csv module
  + csv.DictReader(csvfile, fieldnames=None, …)
    - f = open(‘msha.txt’, ‘rb’)
    - reader = csv.DictReader(f)
    - rows = [row for row in reader]
    - f.close()
    - rows[0]
  + csv.DictWriter(csvfile, fieldnames, extrasaction)
    - f = open(‘output.csv’, ‘wb’)
    - writer = csv.DictWriter(f, fieldnames=[], extrasaction=’ignore’)
    - writer.writeheader()
    - writer.writerow(row)
    - f.close()
    - also, writer.writerows(rows)
  + example, read in file, True if fips\_state\_code= 24 (MD), 42 (PA), 51 (VA), or 54 (WV) False otherwise

Quiz 3

Pandas (just a little bit):

* Dataframe – a representation of tabular data
* Creating a dataframe (from csv, xls, or list of dictionaries)
  + from csv
    - df = pd.read\_csv (‘msha.txt’)
  + from excel
    - df = pd.read\_excel(‘msha.xlsx’)
  + to csv
    - df.to\_csv(‘msha.txt’)
  + to excel
    - df = to\_excel(‘msha.xlsx’, encoding=’latin-1’)