EXAM 2 –Tevis Gehr

PART ONE

>>> df['event'].value\_counts()

1.

SELECT Event, COUNT(\*)

FROM [df]

GROUP BY Event

2.

def value\_counter1(lst1,lst2,lst3):

import pandas as pd

s=pd.Series(lst1+lst2+lst3)

return dict(s.value\_counts())

3.

def value\_counter2(lst1,lst2,lst3):

import pandas as pd

s=pd.Series(list(set(lst1))+list(set(lst2))+list(set(lst3)))

return dict(s.value\_counts())

EC1:

def value\_counter\_any\_number\_args(\*arg):

import pandas as pd

big\_lst =[]

for lst in arg: big\_lst.extend(list(set(lst)))

s=pd.Series(big\_lst)

return dict(s.value\_counts())

#It should be noted that these functions do differentiate by letter case.

#If we want "Hi" to be treated as equal to "hi" the we will need to use:

#x.lower() for x in lst

#for each list before the counting takes place.

PART TWO

1. In Python, a dictionary is way of assigning one and only one value to any given item in the index column (individually called “keys”). In mathematical terms this would be referred to as a discrete, unordered, surjective mapping. Both keys and values can be of a variety of data types, and a value need not be of the same data type as its key. While each key is unique, a value can be repeated across any number of keys. A dictionary is ideal whenever there is a need to look up a value that is always associated with some other value. While lists can be used for lookups, it is not as simple and not as computationally efficient for large data sets.
2. In many cases a Python list and a NumPy array can be used interchangeably, with some differences in syntax. To start, lists are built into Python, whereas NumPy requires the installation of additional software. On a functional level, there are many more complex operations that can be performed on a NumPy array. The programmer has access to all of the NumPy libraries. Also, because the array is stored in memory as a continuous block of literal values, computation is faster. The main advantages with lists are simplicity and flexibility. Each element of a list is really just a pointer that can point to a values, a variable, or even another list. The user can add additional elements. In general I would probably default to using a list and only switch over to a NumPy array or Pandas dataframe if it would make my code simpler or more efficient.
3. Git is a version control system. Its primary purpose is to allow complete accounting for changes that occur over the course of a development project, especially when done by multiple people.

GitHub is a website that provides data storage and version control via Git, as described above. Its purpose is to be a central location where software developers can share version controlled code and collaborate on projects.

1. ls – Lists the files and folders in a the current directory. I use this to either determine where to move to or exactly what the name of the file is that I am trying to run or reference without having to open the windows explorer.

cd – Changes directory. Used to move either up or down in the folder structure from the current location.

mkdir – Creates a new folder. Useful if you are navigating a system from the terminal and don’t have easy access to a file browser, or if you are just accustomed to working in Linux and can work faster without a mouse.

clear – Clears the screen. Useful when running Python scripts from the terminal because it will often get filled up with a lot of text which can be distracting.

mv – Moves a file. Useful for the same reasons as in mkdir above.

1. Inheritance – In Python and other OOP languages such as Java, all objects are of a specific type that exists in a hierarchy of object types. These types are called classes. Whenever using an object of a certain class, the programmer can also use any of the attributes and methods that apply to any of its parent classes. Let’s say you have a class called MazdaMiata() which represents this particular type of car. This class may have a parent class called Car(). Perhaps one of the attributes of the Car() class is number\_of\_wheel which is set equal to 4. Then this attribute also applies to the MazdaMiata() class, such that MazdaMiata.number\_of\_wheels will return the number 4.

Encapsulation – Rather than trying to create a linear script or a set of functions that attempt to pass values between themselves, it is often helpful to instantiate a class object which will continue to exist in memory until terminated and will hold values that are assigned to its attributes. This is the primary advantage to encapsulation, along with the more philosophical desire to organize code into intuitive sections for easy of understanding and modifications.

Polymorphism – From what I have read this principle is the ability of and OOP language to handle different objects differently depending on what type or class they are from. I don’t really have any experience with this concept that I can think of.

1. When confronted with a new data structure I would generally take the following steps, although not necessarily in sequence:
   1. Perform Google search and read up on available documentation to learn how the data structure works.
   2. Look up a list of the most common methods that are used with that data structure.
   3. Write a very basic script that allows me to make one small change at a time in trying out various syntax, methods, and algorithms (basically using the scientific method) in order to test hypotheses and gain an intuition of how it can be used.
   4. Incorporate the data structure into the solution of whatever problem I had that prompted me to look it up in the first place.

Like I said, I don’t necessarily do these one at a time. I may play around with the code a bit before doing a Google search just to see if I can figure it out on my own, but reading up is normally one of the first steps. After that it’s an iterative process of trying out different things until I understand it well enough to make it work and satisfy any additional curiosity.