Python

Luis Calderon

October 2019

1 Python Definitions and Notes

Let this be an introduction to Python. More for the sake of terminology, conventions and enhanced readability for your code. Still working on the list and will add throughout the semester. All errors are my own.

- **Spacing**: make 4 spaces your default identation. Thus, if writing a function or a for loop, 4 spaces is enough when writing the subsequent line.

```
for column in list_of_columns:
    print(column)
```

- **Object**: Everything in python is an object. Types of objects can be dataframes, series, lists, strings etc.
- **Commenting**: Use the # to comment out lines. Leave a space in between the # and the start of the comment.
- **Functions**: Functions are called using parenthesis, where zero or more arguments go in the parenthesis.
- **Methods**: Methods act on an object and performs a function.

```
# This is a function
f(x,y,z)
#This is a method
dataframe.groupby('Turtles')
```

- **Variable Names**: These are also known as *references or bound variables*. They *refer* to the object. The takeaway is that when you give an object 2 references, let's say **a** and **b**, then any functions acted on **a** is also realized for **b**.
- **Imports**: Known as modules, these are files with variable definitions and functions. The reason they are called imports is because they can be used *from another file*.

- **Arrays**: An array (think of a matrix) can have any dimensionality, as long as the entrants are of a single type. This is different from a dataframe, which only has 2 dimensions and multiple types (i.e. float, int, string etc).
- **Mutable**: Mutable objects are frankly objects or values that can be modified. These are most objects in Python.
- **Immutable**: Immutable objects are objects that cannot be modified, like tuples.
- **Strings**: Can be indicated with either ' or ", and for multi-line strings, use triple quotes "' or """

```
def _function_ (object):
    ''' This function will do something
    to an object '''

    object = object.append(a)
return object
```

- **list function**: The **list** function takes a string or a tuple or any other iterable object and returns the sequence of characters with their own entrant. List, at least for me, are So:

```
    s= 'python'
    list(s)
Output: ['p','y','t','h','o','n']
```

- **Control Flow**: These are in respect to statements of the following: *if, elif, else* .

```
def _function_ (x,y,z):

    if x < y < z:
        print('x is so small')
    elif x==y==z:
        print('equality')
    else:
        print('Things to consider.')
```

- **Interesting keywords**: *continue, break, while & pass.* These are mostly used when writing functions, and mean exactly what they are. Continue means to tell the function to keep evaluating given a condition. Pass is similar, but it's more of a placeholder than a final return. Thus, if you're creating a function and are still having trouble, then just run script *pass* and it'll just do nothing (pending exactly what you are or are not telling to pass). Break tells the function to stop if a break condition is met i.e. adding numbers of a list and telling it to break once it reaches a certain

sum. While tells the function to keep evaluating something while a certain condition holds and stop when it no longer holds; common for fixed point problems in mathematics and economics.