**Intro to Python Cheat Sheet**

Objects: An object in python is any (and all) entities that can carry, define, and operate on information. Objects are the variables, lists, and even functions!

Packages: are modules of python code that can augment a user’s ability to interact with their data with new/different functions, method, and attributes. Packages are not a part of the main Python framework and thus must be (1) installed into a user’s environment and (2) imported into their code when needed.

* Ex. pandas, numpy, earthpy

Functions: are set blocks of code that are called on to output specific data. They are recognized by having parentheses after them. The information contained is the parentheses is referred to as the argument.

* Ex. **print**(“Print is a function”)

Methods: are functions made designed for specific objects based on object’s type. Like functions, they are followed by parentheses to add further data for a desired output. They can be recognized by following an object name with a period placed in between and followed by parentheses, however imported modules/functions use the same format.

* Ex. **Correct:** object\_list.insert(1,8); ‘object\_list’ is the object name and ‘insert’ is the method
* Ex. **Incorrect:** np.mean(array\_1); ‘np’ is the shorthand name of the numpy package, ‘mean’ is a function specific to np, and ‘array\_1’ is the object name.

Attributes: are internal characteristics of an object. They are specific to each individual object and vary by object data type and can be compared to some objects length, width, height, or mass.

* Ex. my\_list.size will output the number of elements contained in the list.
* Ex. array\_1.shape will output the size of each dimension in an array.

**Intro Python Data Structures:**

Lists: are objects that contain a list of *ordered* elements. This means that each element has its assigned place through an associated index. The elements do not need to be of the same data type.

* Ex. my\_list = [3, ‘j’, 52.1]

Lists are also *mutable* which means elements can be added to or removed from the list.

* Ex. my\_list.append[2, ‘what is an index’]
  + My\_list will now contain [3, ‘j’, ‘what is an index’, 52.1]

Unfortunately, the list’s flexibility is significantly diminished when incorporates different data types. When lists contain only integers (e.g., 1, 3, 89) or only floats (e.g., 2.0, 0.842, 1.0003), various operators (e.g., +, -, \*) can be employed to manipulate the data they contain. Furthermore, in order to manipulate a list’s elements with mathematical operations, each individual element must be called on with a *ForLoop* which will be discussed later.

Arrays: are similar to lists as a data structure but they are better for handling large amounts of data. Like lists, they are *ordered*, *mutable*, and *non-unique*. Arrays are also an example of an object that must be imported into the Python through a package such as *Numpy*. As such, they must be declared when creating them as demonstrated below:

* Ex. my\_array = **np.array**([1,2,3,4])
* Ex. my\_list = [1,2,3,4]

Depending on the package imported into the script, arrays may or may not be able to incorporate different data types.

A significant advantage of the array is the manner in which is simplifies mathematical operations.

* Ex. exp2 = my\_array\*\*2

print(exp2) => [1 4 9 16]

Indexing: