Python is Procedural and Functional, Strong and Interpreted.

Objects are: Data Definition, &

Structure Definition.

Modules are objects.

An instance contains value(s).

Ex: Object: Name Record.

Instance : “Barnard” “Joshua”

Differentiate Base: 0b\_ 🡨 binary (2)

0o\_ 🡨 octal (8)

0h\_ 🡨 hex (16)

Data Abstraction – The ability to manipulate data w/out knowledge of the data’s internal structure/workings.

Constructor – Automatically invoked when an object comes into existence. The same name as the class, except w/out a return type (not even void).

Destructor – Executes when an object goes out of existence. They put tilde’s (~) infront of objects; ie: ~ClassName.

Cast – Temporarily converts one data type to another; ie: (float)one/(float)two == float(one)/float(two)

\ (Escape) – Causes the following character to be interpreted as an effect.

**Atomic Data Types:**

int: Integer #. (Add .5 to round up.)

float: Floating Point #

str: String

bool: Boolean

**Built-In Functions (Atomic):**

**Operations (Atomic):**

**Molecular Data Types:**

Lists: Mutable; Ordered; Item values can be changed.

Tuples: Ordered, not mutable.

Sets: Unordered; UnIndexed.

Dictionaries: Unordered; Indexed.

**Built-In Functions (Molecular):**

**Operations (Molecular):**

del Dictionary[‘Index’]

Dictionary[‘Index’] = “newValue”

set() 🡨 creates a null set

**Commands:**

del

Type (\_) 🡨 Returns the inputted object’s type.

Len(\_) 🡨 Built-in function. Counts and returns the number of characters in a string, items in list, etc.

Sorted() 🡨 Displays the sorted list

**Methods:**

**Strings:**

\*.split() 🡨 Separates strings into lists using space delineation.

**Lists:**

\*.sort() 🡨 Sorts and resaves the list.

\*.count(\_) 🡨 Counts # of \_ w/in a list.

\*.extend(\_) 🡨 Adds the \_ list to the \* pre-existing list.

\*.insert(#,”\_”)🡨 Add \_ to # spot in list.

\*.append(“\_”) 🡨 Adds \_ to list.

\*.remove(\_) 🡨 Removes \_ from list.

**Dictionaries:**

\*.update(Index : “newValue”) 🡨 Adds value to the \* dictionary, or replaces a value in the \* dictionary.

\*.del() 🡨

\*.items() 🡨

\*.keys() 🡨

\*.values() 🡨

**Sets:**

\*.add(\_) 🡨 adds \_ item(s)

\*.remove(\_) 🡨 remove \_ item(s)

**Operators:**

+ \_ Addition

- \_ Subtraction

\* \_ Multiplication

/ \_ Floating pt division

// \_ Integer division

% \_ Mod

\*\* \_ Exponentiation

**Logical Operators:**

& (AND) - Intersection

| - Union

^ (OR) - XOR (in one but not both)

- - Difference

(NOT()) - Not

**Looping:**

break – exits the loop

continue – go straight to the next iteration of the loop.

**Comprehensions:**

Ex: [number for number in range (#, #)]

**Functions:**

Functions can be passed to functions.

Functions can be returned by functions.

Generators – Very efficient in resources, generalization of “list comprehensions” for and in keywords.

Decorators – “Wraps” or “decorates” a pre-existing function. Process: 1) Create Inner Function; (2) Assign function to a variable; (3) Define functions inside function; (4) Pass functions as arguments; (\5) Con return functions.

**Sample Code:**

**For Loops:**

**While Loops:**

**Iterations, range():**