Type hierarchy: the standard types are not a simple list, but rather a hierarchy.
*Note: I will not list all the types. See Ch. 3 of The Python Language Reference or Ch. 5 of The Python Standard Library for more complete lists. Both are online.

None: this type has a single value, and only the object named 'None' has this value. Hence, you cannot name anything else 'None.' True and False, which are the values of the Boolean type (below) are also constants.

Numbers
> **Integral**
>> **Integer**: Integer numbers between at least -2147483648 and 2147483647
>> **Boolean:** Two values – True and False.
> **Real**
>> **Float:** Supports some range of decimal places, depending on memory

Sequences: A finite *ordered* set. Can be "indexed" and "sliced," (more on these later).
> **Immutable Sequences:** Cannot be changed after creation
>> **Strings:** 0 or more characters (represented by ASCII)
>> **Tuples:** Grouping of two objects: (1,2). (1,) = singleton, () = empty tuple.
> **Mutable Sequences:** May be changed by functions after creation
>> **Lists:** Grouping of N objects: [1,2,3]
>> **Arrays** (available in extension modules only): like a multidimensional list

Set Types: Unordered, finite sets of *unique*, immutable objects. Good for fast membership testing, but not much else as far as I can tell.
> **Sets:** Mutable sets, e.g. can be added to
> **Frozen Set:** Immutable

Mappings: Finite sets of unordered objects indexed by arbitrary index sets.
> **Dictionaries:** Key-value pairs, represented {key:value}. Keys index values. Values are arbitrary objects, but keys can only be immutable types.

   *The following types get fancier, and we'll cover some of them later. You may never use some of them (you'll definitely need a few), but it's important to be aware of all of them in case you want to build larger, fancier code in the future.

Callable Types: These are types you can "call" with the expression: type().
> **User-defined functions:** A function you make with a function definition statement
> **Generator functions:** A special type of function that returns an iterator

Modules: Typically just files of python code. These will have sets of functions which work within the "namespace" of the module. Load a module (named "my_module" in this case) with:

```
import my_module
```
Get the available functions (names) with:
```
dir(my_module)
```
Execute a function from the module with:
```
my_module.my_function
```
The name of the module is denoted:
```
__name__
```

Files: This refers to files that have been opened with the open() function. The objects, sys.stdin, sys.stdout, and sys.stderr are all initialized to the file object once it is opened, which means you can control these flows from within the python script.

Classes: Custom data representations. They are what is often meant by "object" in python terminology, because they allow the user to interact with very complex data structures via a few, neat functions that the programmer specifies. This prevents the user from messing with the underlying data. Classes are an advanced subject, and we probably wont cover them in this course (not least because we don't know much about them)