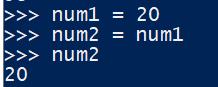
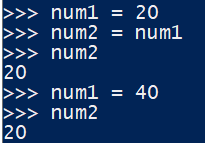
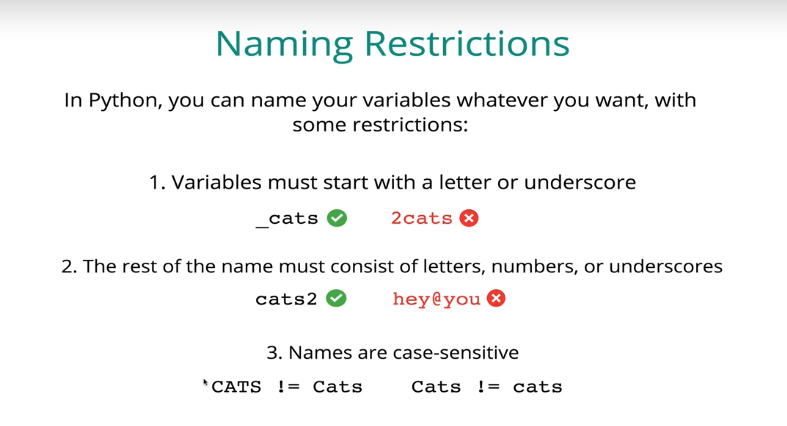
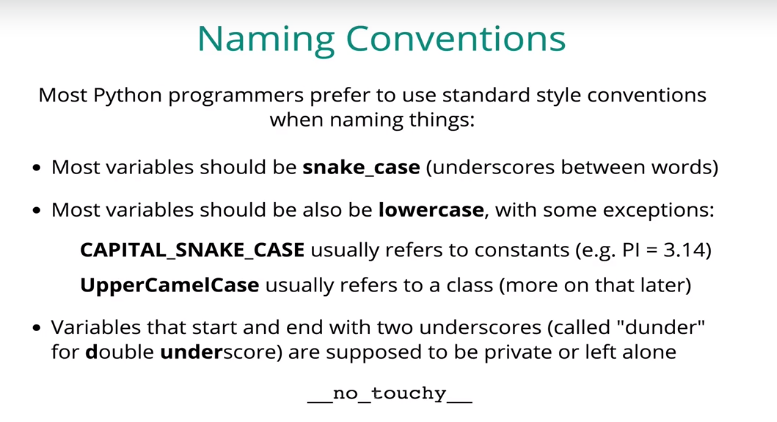
Section 7: Variables and Data Types

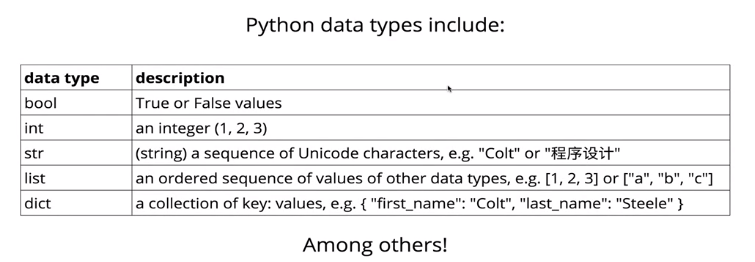
* A variable is a like a container; it stores some data that you can pull out later
  + They can hold all sorts of data, including numbers, Booleans, and strings
  + It is a named symbol that hold a value
  + The values can be recalled later
* Variable assignment is done with an equal sign
  + Variable name on the left, an equal sign, and a value on the right
* Variables are useful because most data is NOT static or not viewable. We want a way to work with our data or the values of that data without knowing what it is
  + Consider having to work with passwords that are input by the user
* Variables must be assigned *before* they can be used
* You can assign multiple variables at the same time! Not done very often, but the option exists.
  + 
* Variables can also be assigned to other variables
  + 
  + But if the first variable is reassigned to a new value, the second variable retains the original assigned value. They DO NOT refer to the same place in memory
  + 
* Variable Naming Restrictions



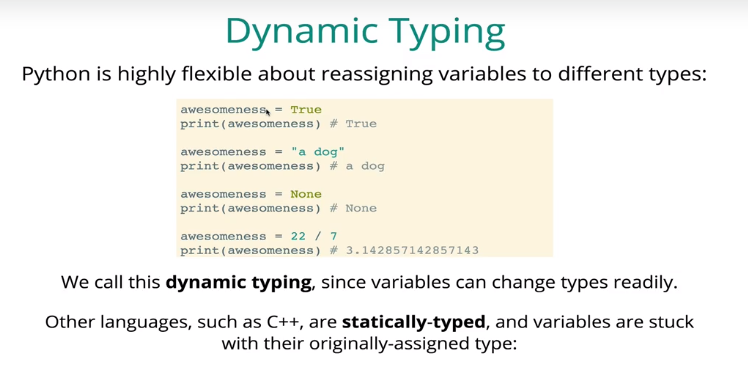
* + You can generally name variables whatever you want with some restrictions
  + Variable names must start with a letter or an underscore
  + The remainder of the name must consist of letters, numbers, or underscores
  + Names are case-sensitive
* Naming Conventions



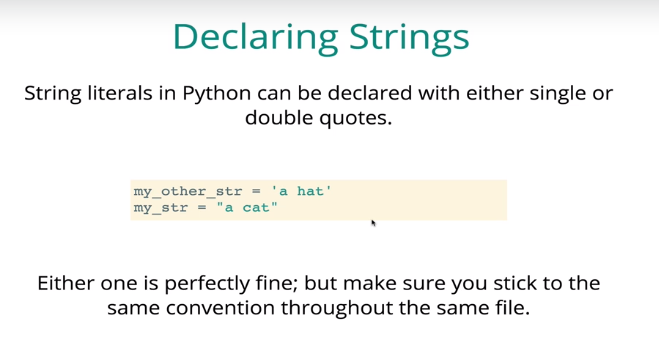
* + You don’t *have* to follow naming conventions, but you probably should
  + The python convention is to use snake\_case
  + ALL\_CAPS is usually used for variables that are intended to be constant, not to be changed
* Data Types – so many things aside from numbers
  + Images, descriptions, numbers, etc



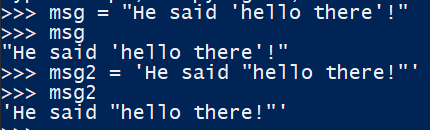
* + **Boolean** logic is incredibly useful for all sorts of stuff
    - True and False are always capitalized to be recognized as Boolean
  + Strings are just “characters”, which can include any Unicode character. Letters, numbers, symbols, anything goes!
    - They are inside of quotes
  + A **list** is a data structure, which can store other types of data within them!
    - They are *ordered*. More on that later
  + **Dictionaries** are not ordered; they are a collection of values that are accessed by their keys
* Dynamic typing
  + Python allow you go do things like change a variable value from 99 to (for example) a string. Not all languages necessarily support this



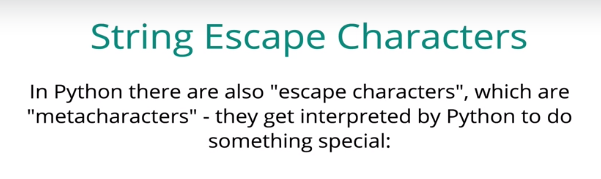
* What is None?
  + It is the only instance of the type “NoneType”
  + None is a special term that has a value of nothing, representing the idea of nothing
  + It is a nice alternative to not setting a value at all if the variable itself is useful later
    - For example, maybe you have no children now, but you want the variable container for children to exist in case you have children later
* Dealing with double and single quotes in strings
  + When declaring strings, it doesn’t really matter which you decide to use, but it needs to be consistent within the same file

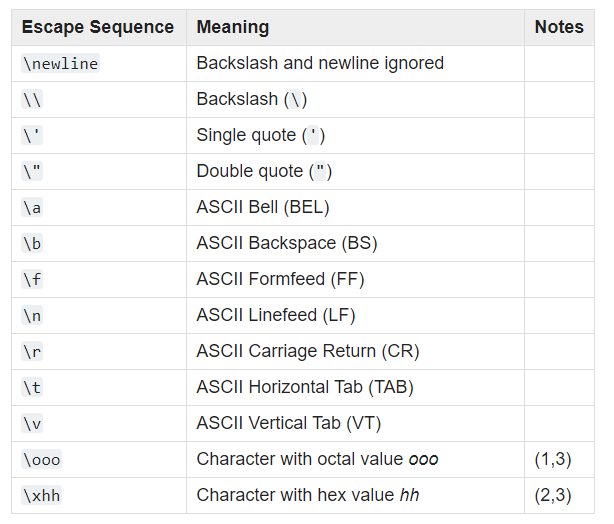


* + You can also use quotes within other quotes. You can use double quotes inside of single quotes, or single quotes inside of double quotes

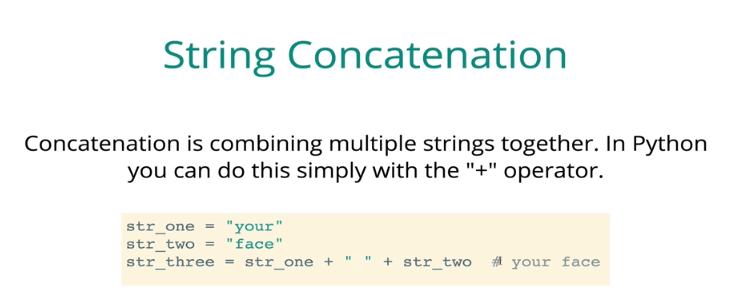


* Escape Characters!

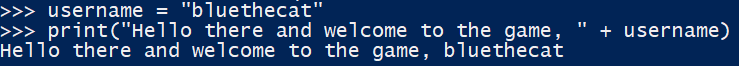




* + Some of these are kind of quirky. For example if you want to actually print a backslash, you have to put a double backslash so that the first backslash escapes the second one. Otherwise a single backslash would be interpreted as the beginning of an escape
  + The backspace one is also weird. It just deletes whatever character comes before it
  + Another useful use is escaping quotes, so that the quotes you type are not interpreted as string identifiers but rather as quotes within the string. You can use this trick to use double quotes as the string identifier and as quotes in the string:
    - 
* String concatenation: adding two strings together
  + Concatenating strings together using the “+” operator literally just shoves them together



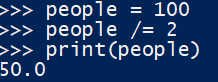
* + Variables containing strings can easily be used in printing strings



* + You can only string concatenate actual strings. You cannot concatenate a string with something else, like a number. We’ll learn how to force something into a string later
  + The += operator is one way to quickly update a variable. Basically it says take the current value of the variable, add the value after the operator and set the variable equal to that new value.
    - This works for both strings and numbers

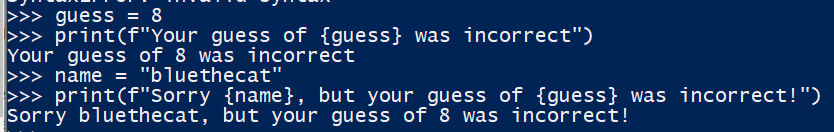


* + You can also do this with “-=”, “\*=”, “/=”

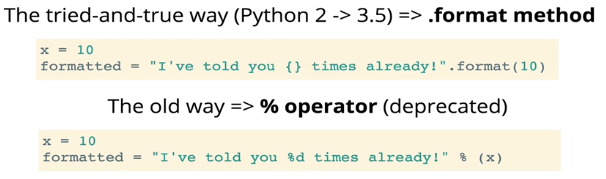


* String Formatting
  + <https://realpython.com/python-string-formatting/>
  + What if we want to do something like add an integer into a string?
  + **f strings** are the new way to do this in Python 3.6+, which allows you to use curly braces to escape from the string and put a variable in there.
    - This is the method we will use in this class
    - You can use as many variables as you want in an f strings

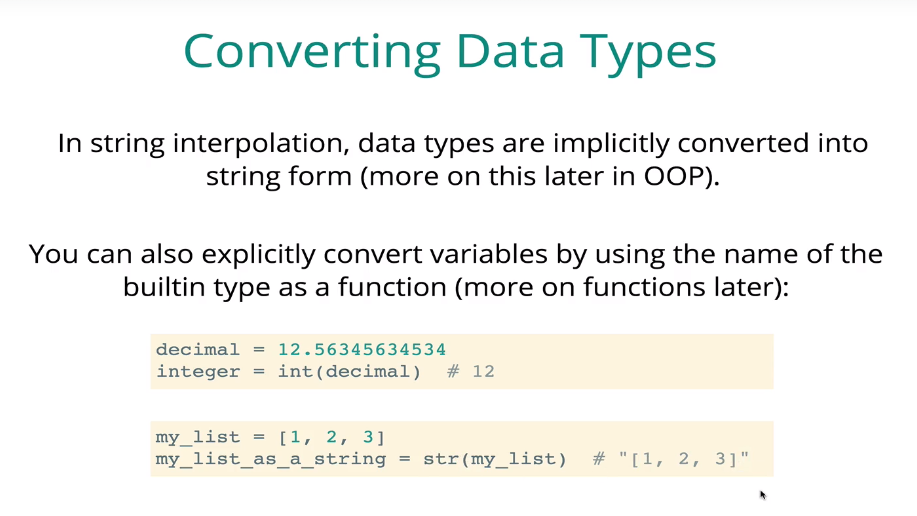




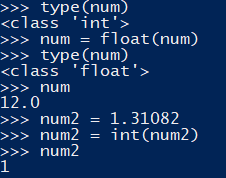
* + There are also older, clunkier ways of doing this:
* Number rounding!
  + You can use the round() function to limit the number of decimals
  + It takes two arguments: the value to be rounded, and the number of decimals to round to.



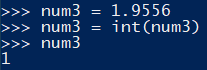
* + Unfortunately we need to use the .format() method in Udemy, which only supports up to Python 3.5. F string support is in the works!
  + Old string formatting specifiers:
    - %s - String (or any object with a string representation, like numbers)
    - %d - Integers
    - %f - Floating point numbers
    - %.<number of digits>f - Floating point numbers with a fixed amount of digits to the right of the dot.
    - %x/%X - Integers in hex representation (lowercase/uppercase)
* String indexing
  + You can access specific characters within a string by referencing its index in brackets []
  + Indexing in Python starts at 0
    - For “lol”, the [0] index is “l”, the [1] index is “o”, and the [2] index is the second “l”
  + This can be useful when we need user input, for example prompting them for a “yes” or “no” answer. We can
  + We can also access string positions with negative numbers:
    - For “yessir”, the [-1] position gives you “r”, [-2] position gives you “I”, and so forth.
* Converting Data Types
  + You can do things like converting a float into an integer and vice versa, or turn a string “8” into an integer 8







* + Converting a float into an integer does NOT round the number, it simply floors it



* + Don’t make variables with the names “print”, “str”, “int”, “float”, etc because you will end up overwriting the built-in functions that Python already has
  + Remember that when using f strings, anything you put into the string is automatically converted into a string behind the scenes