Intro

* Defining variables is done by stating the variable and what it is equal to
* Variable types will set automatically by what value they’re given
  + 1.12 would default to float
  + 1 would default to int
  + True would default to boolean (**booleans must have first letter capital)**
* Return is used to return a value, which can be used in further code
* Print is used to return human text to the user only
* # is used for single line notes, “” “” for multi line.
* Math operators:
  + 5+5 returns 10
  + 5-5 returns 0
  + 5\*5 returns 25
  + 5/5 returns 1
  + 5\*\*2 returns 25
  + 5%4 returns 1 (modulo = remainder)
* variable=raw\_input(“Question?”) will cause a value set to it to prompt for an answer

Strings

* Can be created in ‘’ or “”
* variable=”Text”[1] will save “e” as the string
* len(variable) will return 4 as the length
* variable.lower() will return “text” as all lowercase
* variable.upper() will return “TEXT” as all uppercase
* str(5) would turn 5 into “5” as a string instead of an int (works for variables’ values also)
* **Dot notation can only be used on strings**
* % String notation: sentence=”sentence” example=”example” print “This is a %s %s.” % (sentence, example) will print “This is a sentence example”

Datetime

* from datetime import datetime
* datetime.now() returns the current time, default YYYY-MM-DD HH:MM:SS.S
* Modifiers for datetime.now() can be applied after () or to the variable
  + .year, .month, .day, .hour, .minute, .second

Conditionals

* == is equal to, != is not equal to, <= is less than or equal to, >= is greater than or equal to
* Boolean operators
  + and
  + or
  + not
* True has precedence with ‘or’, False has precedence with ‘and’
* In a single statement, the order of evaluation outside parentheses is not, and, or
* if format:
  + def functionname():
    - if condition:
      * action
    - elif condition:
      * action
    - else:
      * Action
* for format:
  + for variable in variable:
    - function
* The above formula will repeat the function for each time the conditional applies

Functions

* def functionname(argument):
  + print argument
* Importing functions from modules:
  + from [modulename] import [functionname]
  + from [modulename] import \*
  + import [modulename]
    - You must always then reference the modulename when calling functions from it when using this method: ex. modulename.functionname() rather than just using functionname()
  + print dir(modulename)
    - Shows all functions within a module
* Math Functions:
  + abs() - absolute value of a number
  + min() - smallest number in a range
  + max() - largest number in a range
* type() displays type of entry (integer, float, string)

Dictionaries and Lists

* lists are numbered starting with 0, and accessed by varname[#]
* varname.append(‘entry’) adds an entry to the end of the list
* Number ranges start with the first number and end with the number after the one you are including (list[0:2] will include 0 and 1, ending at 2 but not including it
  + Leaving one number blank in a range includes everything up to or after that point, serving basically as an asterisk does in SQL
* varname.insert(#,’entry’) places the entry in that position, and shifts all after it to the next spot
* varname.index(‘entry’) identifies the number of the spot that entry is in
* varname.sort() sorts a list into alphanumerical order
* Dictionaries function by key instead of by index
  + dictionaryvar = {‘key1’:1,’key2’:2}
  + Keys can be accessed the same way as indexes
    - dictionaryvar[‘key1’] would return 1
    - Adding entries is done as follows:
      * dictionaryvar[‘keyname’] = entryvar
  + To remove an entry use the commend del dictionaryvar[key]
* Removing items from a list is done with varname.remove(“entry”)
* Alternatively, varname.pop(index) will remove the item at that index, and also returns it
* del(varname[index]) also removes the item at that index, but without returning it
* Ranges
  + range(x) - gives full range of values from 0 to x
  + range(x,y) - gives full range of values from x to y
  + range(x,y,z) - gives range of values from x to y by steps of z
* Tuples
  + varname = (entry, entry2)
    - varname will always have 2 entries, never more or less
    - A tuple of one item will have a comma and nothing following