**Python for Everyone: Notes**

**Chapter 4: Functions**

* Store and reused steps
  + Reusable code are functions
* Python functions
  + Two kinds of functions in python
    - Built in functions that are provided as part of python – print(), input(), type()
    - Functions we define ourselves and use
  + We treat built in function names as new reserved words
* Function Definition
  + In python a function is some reusable code that takes arguments as input, does some computation, and then returns a result or results
  + We define a function using the def reserved word
  + We call/invoke the function by using the function name, parentheses, and arguments in an expression
* Max function
  + A function is some stored code that we use. A function takes some input and produces an output
* Type conversion
  + When you put an integer and floating point in an expression, the integer is implicitly converted to a float
  + You can control this with the built in functions int() and float()
* String conversion
  + You can also use int() and float() to convert between strings and integers
  + You will get an error if the string does not contain numeric characters
* Building our own functions
  + We create a new function using the def keyword followed by optional parameters n parentheses
  + We indent the body of the function
  + This defines the function but does not execute the body of the function
* Definitions and uses
  + Once we have defined a function, we can call or invoke it as many times as we like
  + This is the store and reuse pattern
* Arguments
  + An argument is a value we pass into the function as it input when we call the function
  + We use arguments so we can direct the function to do different kinds of work when we call it at different times
  + We put the arguments in parentheses after the name of the function
* Parameters
  + A parameter is a variable which we us in the function definition. It is a handle that allows the code in the function to access the arguments for a particular function invocation
* Return values
  + Often a function will take its arguments, do some computation, and return a value to be used as the value of the function called in the calling expression.
  + Return keyword
  + A fruitful function is one that produces a result or return value
  + The return statement ends the function execution and sends back the results of the function
* Multiple parameters / arguments
  + We can define more than one parameter in the function definition
  + We simply add more arguments when we call the function
  + We match the number and order of arguments and parameters
* Void (non-fruitful) functions
  + When a function does not return a value, we call it a void function
  + Functions that return values are fruitful functions
  + Void functions are not fruitful
* To function or not to function
  + Organize your code into paragraphs – capture a complete thought and name it
  + Don’t repeat yourself – make it work once then reuse
  + If something gets too long or complex, break it up into logical chunks and put those chunks into functions
  + Make a library of common stuff that you do not want to do over and over