Python 3 Essential Training Notes

**Variable, Objects, and Values**

* Everything in Python is an object
  + Variables, functions, and even code
* Every object has an ID, Type, and Value
  + ID uniquely identifies a particular instance of an object, and cannot change for the life of an object
  + Type identifies the class of an object, and cannot change for the life of an object
  + Value is the contents of an object
    - Mutable objects change value
    - Immutable objects cannot
* All variables in python are first class objects
  + What looks like a simple variable may be something more complex, e.g. defined in a library, built-in object, often has attributes/methods

**Mutable and Immutable Objects**

* Objects in Python may be mutable or immutable
* Mutable objects may change value, immutable objects may not
  + Immutable objects may looks like they are changing value but they are not
  + Distinction is visible using id()
  + Container objects (tuples, lists, etc.) may appear to change value
* Most fundamental types in Python are immutable
  + Numbers, strings, tuples
* Mutable objects include:
  + Lists, dictionaries, others depending upon implementation

**Operator Precedence**

|  |  |  |
| --- | --- | --- |
| Associativity | Operators | Description |
| Left to right | Lambda | Lambda expression |
| Left to right | Or | boolean |
| Left to right | And | boolean |
| Left to right | Not | boolean |
| Right to left | In, not in, is, is not, <, <=, >, >=, !=, == | Comparisons |
| Left to right | | | Bitwise |
| Left to right | ^ | Bitwise |
| Left to right | & | Bitwise |
| Left to right | <<, >> | Bitwise shift |
| Left to right | +, - | Addition/subtraction |
| Left to right | \*, /, //, % | Multiplication, division, remainder |
| N/A | + x, - x, ~x | Unary arithmetic |
| Left to right | \*\* | Exponentation |
| Right to left | X[slice], x(arguments…), x.attribute | Slice, function call, attribute reference |
| Left to right | (expression…), [expression…], {key:value} | Binding, tuple, list, dictionary |

**Using Regular Expressions**

* Regular expression are a very powerful method of matching patterns in text
* Actually a small language in itself, regexes can be very simple or very complex
* Implemented in Python in the “re” module
  + Import re  
    pattern = re.compile(r’\d\d\d’)  
     if re.search(regex, line): print(line)

**Using Exceptions**

* Exceptions are the key method of handling errors in Python
* “try” something, then catch an exception with “except”
  + try:  
     fh = open(‘filename’)  
     except IOError as e:   
     print(e)  
     else: for l in fh: print(l)
* You may raise your own exceptions with “raise”

**Understanding Classes**

* Classes are how you create your own objects
* A class is the blueprint for an object
  + class Duck:  
     def quack(self):  
     print(“Quaaaaak!”)  
     def walk(self):  
     print(“Walks like a duck!”)
* An object is an instance of a class
  + donald = duck  
    donald.quack()  
    donald.walk()