ISTA 350 Final Study Guide

Know your big O material!

Know how to draw a linked parse tree per hw2. There are no `dict` parse trees on the test.

Know how to do the kinds of problems that were on the two's complement worksheet.

Know your other linked data structures that we have explored, including how to write recursive methods for BST's.

Regular expressions will not be on the final.

Memory diagrams will not be on the final.

Know your `Binary` class from hw4 (there is more on this below).

Know how to traverse a list of lists using nested loops. Understand the difference between indices and elements and when and how to use them both in conditions.

Know how to use default arguments.

Know the syntax for defining a class and creating instance variables:

```
class ClassName:

    def __init__(self, arg1, arg2):
        self.inst_var1 = arg1
        self.inst_var2 = arg2
```

Know how to access an instance variable from a method inside a class definition:

```
    def __add__(self, other_instance):
        """ This is an instance method. """
        iv1 = self.inst_var1 + other_instance.inst_var1
        iv2 = self.inst_var2 + other_instance.inst_var2
        return ClassName(iv1, iv2)
```

Know how to access an instance variable from outside the class definition:

```
def my_add(instance1, instance2):
    """ This is a function. """
    iv1 = instance1.inst_var1 + instance2.inst_var1
    iv2 = instance1.inst_var2 + instance2.inst_var2
    return ClassName(iv1, iv2)
```

Know how to call methods and functions:

```
def main():
    instance1 = ClassName([0, 1, 2], 'dog')
    instance2 = ClassName(list_var, string_var)

    sum1 = instance1 + instance2    # this calls __add__() method
    sum1 = instance1.__add__(instance2)  # same as the previous
                                         # line, but never do this
    sum2 = my_add(instance1, instance2)

    if sum1 == sum2:
        print('good!')
    else:
        print('define __eq__, please')

    # access an instance variable:
    for item in instance1.inst_var1:
        print(item)
```

Know what magic methods are and how to define and use them. Understand operator overloading (see the examples above).

Know how to use the *@classmethod* and *@staticmethod* decorators.

Understand hw4. Understand that `Binary` objects are integers and that once you tell Python how operators (+, -, <, etc.) should work on them, you can use these on them. Understand the difference between an instance and an instance variable. Know that operators defined to work on instances cannot be used on instance variables (at least not with the desired result).

Know the *@functools.total_ordering* decorator and what it is used for.

The list of parameter for an instance method will always have length greater than the argument list for any call to that method. Why?

Understand this code:

```
one = Binary("01")
neg_one = Binary("1")
zero = one + neg_one
```

What method is called in the last line of the example? What is the argument passed to it?

If you are defining the `add` magic method for a class, can you use the plus sign (+) in it? When and when not?

Understand what happens when you pass a mutable object to a function and change that object in-place inside the function.

Know the difference between sample and population statistics, discrete and continuous data, snapshots and time series, frequencies and probabilities, and probability mass functions and probability density functions.

Know what a normal distribution is and how you can use the 68-95-99.7 rule to make predictions about normally distributed data.

Know what an HTTP request is and the difference between GET and POST. Know how to use requests to make a GET request. Know how to use BeautifulSoup to parse the response HTML and get the data out of an HTML table.

Understand what an HTML element is, what tags are, and how to turn HTML into a tree.

Know what the transpose of a matrix is.

Know `try-except-finally`, `with`, `enumerate`.

Know what JSON is and how to save Python objects to JSON files and load them from said files.

Know pandas.

Know stacks, queues, and graphs.