ISTA 350 Midterm Study Guide

The test has 105 points worth of questions, so the max possible score is 105/100 (essentially, there is 5 points of extra credit built in). This study guide covers material that you should have known before starting 350 and the material that we have covered.

SQL and database access will not be on the test.

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, except BST's, which will be on the final. **But there is a linked list problem on the test.**

Know the basics of regular expressions. Know your repetition operators and the or operator.

There is a 5-pt memory diagram question on the test.

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

Know how to create a list by assigning a list literal to a variable, how to construct an empty list, and how to add things to a list. Know how to build a list in a loop. Know how to traverse lists and strings by element and by index. Know how to convert each kind of `for` loop into the other, if possible, and into a `while` loop and vice versa. Be able to recognize when you have to traverse by index and when you can traverse by element. Know how to turn a string into a list and vice versa (both are string methods). Know how to access elements of a list by index and to assign to a given position. Know how to use slicing to extract sublists (are they new objects or views?). Know the list methods including `insert`, `append`, `sort`, `reverse`, and `pop`. Know the functions that can take lists as arguments or which return lists including `sorted`, `sum`, `max`, `min`, and `len`. Understand the difference between mutating a list (altering it in place) and constructing a new one. Understand how to get information out of a function by passing a list (or any mutable object) as an argument.

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 the string methods including `count`, `find`, `lower`, `islower`, `replace`, `capitalize`, `join`, `split`, `strip`, and `isdigit`.

Know when and how to typecast.

Know how to use default arguments.

Know how to create a dictionary by assigning a dictionary literal to a variable, how to create an empty dictionary, and how to add key-value pairs to a dictionary by indexing on the key. Know how to get values from a dictionary. Know how and when to check whether or not a key is already in a dictionary. Know how to work with a dictionary that has lists as values. Know the dictionary methods and functions. Understand how views work.

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. Specifically, on this test, know how to make a class method.

Know how to open and close files. Know how to traverse a file in more than one way.

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 parameters 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?

Know how to traverse an object and update a variable.

Know the built-in tuple and set data structures. Understand how to return multiple objects in a tuple.