CIS 480 Exam #1 Review Suggestions

* last modified: 9-22-05, 11:53 am

* remember: YOU ARE RESPONSIBLE for course reading, lectures/labs, and especially anything that's been on a homework, in-lecture exercise, or lab exercise; BUT, here's a quick overview of especially important material.

* you are permitted to bring into the exam a single piece of paper (8.5" by 11") on which you have handwritten whatever you wish. This paper must include your name, it must be handwritten by you, and it will **not** be returned.

  Other than this piece of paper, the exam is closed-note, closed-book, and closed-computer.

* this will be a pencil-and-paper exam, but you will be reading and writing Python code, statements, and expressions in this format.

  Note that a packet of Python code may be included along with the exam, both for reference and for use directly in some exam questions --- the ability to make use of existing code as a reference is a vital skill in Python (as in most programming languages).

  * note that you could be asked to write Python expressions, statements, functions, or up to and including entire Python modules;

    (note, too, that answers may lose points if they show a lack of precision in terminology; for example, if I ask for a literal or an expression and you give an entire statement, instead)

  * note that I could ask you questions *about* Python, or about various aspects of Python;

  * note that I could ask you what given Python code does or means; I could give you one or more statements, a function, etc., and ask you what it does or what it would output in a given situation (or how you could write a call to use it, etc.)

  * you could be asked to modify a piece of code or function or module, or to correct a segment of code or a function or a module, as well;

* **Python basics**

  * There could be questions *about* Python in general; who created it, its general design philosophy, what it was named after, etc.

  * There could be questions about how we have run Python thus far --- how to start up and run the python interpreter, how you can run a Python module from a UNIX/Linux command line, how you can import a module into the Python interpreter or into another module, how you then use functions so imported;

  * There was no reading assignment for the "whirlwind tour" of Python (week 2 lecture); you are responsible for the features discussed in that lecture.

  * You should be comfortable with how Python differs from C++/Java (for example, no variable declarations, blocks indicated by indentation, strong but dynamic typing, etc.)

  * Some of the basic features of Python from that "whirlwind tour" included how to write arithmetic and boolean expressions, how to set up variables, how to write and call functions, how to write **if** statements, how to write basic **while** loops, simple output, and more;

  * At this point, you should be comfortable with and familiar with the basic Python types, including **NoneType**, whose one literal is **None**;

  * How can you determine the type of any Python expression?

* **Python strings**

  * You are responsible for the Chapter 5 reading on strings; you do not need to worry about the Chapter 27 part. I will also not be asking any questions about the older **string** module; we'll stick to **str** objects and their operators and methods.

  * Note that we covered basic **for**-loops here;

  * We've had lectures, lab exercises, and homeworks on strings; I'll expect you to be quite comfortable with them.

  * Some important string features include: their basic operators (indexing, concatenation, repetition, slicing, etc.); the string methods we have discussed so far; how they are **immutable**, how you can do string formatting, etc.

  * what are some of the different ways to write string literals? Why are some more useful in some situations than others?

  * We say that a string is an **immutable sequence**; what are the implications of this?

  * what kinds of things are strings good for?

* **Python lists**
    * You are responsible for the Chapter 6 reading on lists.
    * How can you use a **for**-loop to traverse a list? How does this differ from using a **for**-loop to traverse a string?
    * We've had lectures, lab exercises, and homeworks on lists; I'll expect you to be quite comfortable with them.
    * Some important list features include: their basic operations (indexing, concatenation, repetition, slicing, etc.); the list methods we have discussed so far, how they are **mutable**, how ANYTHING can go into a list, how they can grow and shrink, etc.
    * We've also discussed some string methods that involve lists (join, split)
    * In this discussion, we got to discuss further functions/methods that do not return anything (for example, list methods append, sort, and reverse); how are such functions written? How does this affect how you use these functions/methods?
    * Some important list features include: their basic operators (indexing, concatenation, repetition, slicing, etc.); the list methods we have discussed so far; how they are **mutable**, etc.
    * We say that a list is an **mutable sequence**; what are the implications of this?
    * how do you access a list of lists?
    * what kinds of things are lists good for?
* **Python dictionaries**
    * You are responsible for the Chapter 6 reading on dictionaries.
    * We've had a lecture on dictionaries; I won't expect quite as much adeptness with them (yet!) as you have with lists and strings, but you should be comfortable with the basics about them, with the topics read and discussed about them, and with how they compare/constrast with lists (and strings, even)
    * How can you use a **for**-loop to traverse a dictionary? How does this differ from using a **for**-loop to traverse a string or a list?
    * Some important dictionary features include: their basic operations (grabbing the

value given a key, how to add a key-value pair, how to change a key's value, etc.); the dictionary methods we have discussed so far, how they are **mutable** (but keys aren't!), how ANYTHING can go into a dictionary (except for those key limitations), how dictionaries can grow and shrink etc.

* We say that dictionaries are **mutable** but not **sequences**; what are the implications of these?

* how do you access the contents of a list within a dictionary? of a dictionary within a list? etc.!

* what kinds of things are dictionaries good for?

* **Python tuples**

  * You are responsible for the Chapter 7 reading on tuples (but only through tuples, for this exam).

  * We've discussed the basics of tuples; I won't expect quite as much adeptness with them (yet!) as you have with lists and strings, but they are relatively simple and share some important characteristics with lists, so you should be comfortable with the basics about them, with the topics read and discussed about them, and with how they compare/constrast with lists (and strings, even)

  * how DO tuples differ from lists? You should be able to come up with at least one scenario where a tuple can be used, but not a list;

  * How can you use a **for**-loop to traverse a tuple?

  * Some important tuple features include: their basic operations (indexing, concatenation, repetition, slicing, etc.); how there are not tuple methods (!), how they are **immutable**, etc.

  * We say that a tuple is an **immutable sequence**; what are the implications of this?

  * what kinds of things are tuples good for?