CIS 130 Exam #1 Review Suggestions

* last modified: 2-7-07, 2:23 pm

* 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.

* The test covers the HtDP reading packet through the end of Section 3.

* 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 you could be asked to write Python expressions and functions;

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

  * note that I could ask you what given Python code does or means; I could give you one or more expressions, a function, etc., and ask you what it does or what would then be shown in the python interpreter 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;

* what is syntax? what is semantics?

* what Python data types have we discussed so far?
  * how can you write values (literals) for each of these data types?

  * look at a Python value/literal or Python expression and give its type; write a Python value or Python expression of a particular type;

* how do you write arithmetic expressions in Python?
  * what are some of the arithmetic operations/functions provided by Python (and its **math** module)?

    * Given an arithmetic operation in "algebraic" form, write it as an equivalent Python expression;

* given a Python expression --- what is its type? (be able to determine the type of a given Python expression)

* what is a parameter? how can you declare a parameter in Python?

* when an identifier/variable represents an unchanging value within a program, we call it a "named constant";
    * why are these good to use within programs?

    * How do you declare a named constant in Python?

    * (Note that we have a style standard that such named constants should be written in all-uppercase.)

    * (Note that we also have a class style standard that named constants should be used in place of literals within code.)

* what is a comment? How can you write a comment in Python?

* functions in Python
    * what is a function header? what is a function body?

    * what is a **return** statement? What does it do?

    * be able to read, write functions, using the program design recipe;

    * given a function and a call of that function, tell what that function will return;

    * given a function, be able to write a call of that function;

    * be able to write expressions including appropriate function calls;

* how should parameters and named constants be named? how should functions be named?
    * remember that any name that a programmer chooses is called an **identifier** --- what is the basic syntax for a Python identifier?

    * Given a would-be identifier --- tell if it is a "legal" Python identifier (if it would be an accepted by, say, the python interpreter as a syntactically-correct identifier);

    * Given a scenario, judge if an identifier would be a descriptive, meaningful choice for a particular purpose within that scenario;

* what do we mean by an **auxiliary** function?

* what is an argument? what is a parameter?
    * given a fragment of Python code, could you name the arguments in that fragment? could you name the parameters in that fragment?

* what is a syntax error? what is a semantic/logical error? what is a run-time error?
    * given a fragment of Python code, characterize an error within that code as a syntax error, a semantic/logical error, or a run-time error;

    * be able to give an example of each of these categories of errors, if asked;

* you should be comfortable with the design recipe (since you should have been using it for many of the numerous functions that you have written at this point).
    * there could be questions about the design recipe itself; you should also be comfortable with the expected order of the steps in this process.

    * there could also be questions where you have to produce certain steps within the design recipe.

    * if the scenario for a problem is given, be able to:
        * perform data analysis and design a data definition, if appropriate, for the kinds of data involved in that problem;

        * give the **contract** for each desired function involved in solving that problem;

        * give the **header** for each desired function involved;

        * write an appropriate **purpose statement** for each desired function involved, being sure to make appropriate use of the parameter names within it;

        * come up with **examples** for each desired function involved,

        * come up with an appropriate **body**,

        * come up with appropriate **tests;**

* class style standard: anything within a function body MUST be indented noticeably (by at least 3 spaces)
    * (we will soon see that this is true for *any* "body", whether it is a function body, an if-condition body, an else-condition body, etc.;)

* (note that there could be questions about the class style standards, and your grade for test questions may be affected by whether you follow those class style standards in your answers.)

* (note that there could be questions where you are given code and/or design recipe elements, and asked if they are "legal", if they meet class coding standards, if they are syntactically or logically correct; you might have to correct code or design recipe elements that are "incorrect" or "illegal" in some sense.)