Comp 170 Exam 1 Overview.

Resources During the Exam
The exam will be closed book. You may use notes on three sides of 8.5x11 inch paper (single or double sided, but three sides total). Write this as you study! I mostly want to test you on concepts and process, not memorized rote facts.

Main Python topics that may be on the exam from the text fopp chapter 1-9, 12: 1-12. No files or dictionaries, or the end of Chapter 12):
1. Python types int, float, str, bool, list. **Not Turtle – so not most of Ch 5.**
2. Converting between types.
3. Function definition and calling a function; “fruitful” and not fruitful.
4. Statements: assignment, for, return, if-elif-else.
5. Global functions max, min, len, type, print, input, range (with 1-2 parameters).
7. Boolean operations: and, or, not
8. Strings: indexing, as a sequence usable in a for-loop, slicing, adding, multiplication by an int; methods: upper, lower, find, split, join, strip, count, format; syntax in format string for formatting floats with precision
9. Lists: indexing; method append
10. In module random: function randrange

How the Python topics get used:
1. To test that you understand syntax and its sequence, expect “What does this print?” for any given code with any combination of syntax, like loops, nested loops, loops+if, nested if, formatting, function calls. Though it is not required with a correct answer, it is good, particularly with complicated code, to “play computer” on the side, doing essentially what CodeLens does, tracking state through individual statements, and copy just the exact characters printed, in the proper format, to the location of a final answer. CodeLens is dynamic, only showing you the current values. If you want a record, it is good to write down the sequence of changes in the order the statements are executed. In my solutions of many “what is printed?” exercises, I have a line for each statement executed, with columns for each variable in the code snippet, and I show any new values for each variable.
2. Writing short snippets, no more than a few lines, largely with a function heading and description given, looking for you to fill in the function body. Hence understanding formal parameters is essential! Be sure to distinguish what is asked for: returning something vs. printing!

Read the following before looking at either the problems or the solutions:
1. Study first, writing details that are not solid for you in your notes, until you think you have everything nailed down with the help of your notes, and then look at the sample problems. The sample problems cannot give complete coverage, and if you look at them first, you are likely to study just these points first, and will not get an idea how well you are prepared in general. Look at the list at the top of the page and start by filling in any holes.
2. Do not look at the answers until you have fully studied and tried the problems and gotten help getting over rough spots in the problems if you need it! Looking at the answers before this time makes the problems be just a few more displayed examples, rather than an opportunity to actively learn by doing and check out where you are. The doing is likely to help you be able to do again on a test.
3. For checking yourself after you are done, you can test yourself by copying code into your computer, but for the exam it will just be on paper.

New sample problems start on the next page. They are more than twice the length of an actual exam.
Review Problems for Exam 1  (Solutions follow the problems.)
1. What is printed by the Python code? Keep track of the latest value of each variable!
   ```python
x = 5
y = x + 3
x = x - 1
z = 10
x = x + z
print('x: {}, y: {}, z: {}'.format(x, y, z))
```

2. What is printed by the Python code?
   ```python
   print(14//4, 14%4, 14.0/4)
   ```

3. What is printed by the Python code?
   ```python
   for z in [2, 4, 7, 9]:
       print(z - 1)
   ```

4. What is printed by the Python code?
   ```python
   print('2' + '3')
   ```

5. What is printed by the Python code? Be careful to follow the order of execution, not the order of the text!
   ```python
   def f1():
       print('Hi')
   def f2():
       print('Lo')
   f2()
f1()
f1()
   ```

6. What is printed?
   ```python
   def foo(x):
       return x + 3
   def bar(a, n):
       print(a*n)
   print(foo(7))
   bar('x', 4)
   bar(foo(2), 6)
   ```

7. What is printed by the Python code?
   ```python
   print(list(range(3)))
   ```

8. What is printed by the Python code?
   ```python
   for i in range(3):
       print('Hello again!')
   ```

9. What is printed by the Python code?
   ```python
   for i in range(4):
       print(i)```
10. What is printed?
   
   ```python
   for s in ['ab', 'c']:
       for n in [1, 3]:
           print(s*n)
   ```

11. What will be printed by the function calls in parts a-d below?
   ```python
def comp(x):
    if x < 3:
        print("A")
    elif x > 10:
        print("B")
    else:
        print("C")
   ```
   
a. comp(5) b. comp(12) c. comp(-2) d. comp(10)

12. What will be printed by the function calls in parts a-d below?
   ```python
def comp2(x, y):
    if x == y:
        print("A")
    elif x < 5 and y > 2:
        print("B")
    if x > 2 or y > 4:
        print("C")
    print("Z")
   ```
   
a. comp2(5, 3) b. comp2(5, 5) c. comp2(1, 5) d. comp2(1, 1)

13. What is printed by the Python code?
   ```python
   n = 3
   for x in [2, 5, 8]:
       n = n + x
   print(n)
   ```

14. What is printed by the Python code?
   ```python
def s(x):
    return x**x

tot = 0
for n in [1, 3, 5]:
    tot = tot + s(n)
print(tot)
```

15. What is printed by the Python code?
   ```python
   for n in [1, -2, 10, 5, 3, 7, 4]:
       if n > 4:
           print(n)
   ```

16. What is printed by the Python code?
   ```python
   x = 2.5679
   y = 9.0
   print('Answers {:.3f} and {:.3f}'.format(x, y))
   ```
17. What is printed by the Python code?
   ```python
   s = 2  
   for n in [3, 5, 7, 2]:  
       if n > 3:  
           s = s + n  
       else:  
           s = s * n 
   print(s)
   ```

18. Suppose you know $x$ is an integer and $ys$ is a string representing an integer. For instance, $x$ is 3 and $ys$ is '24'. Write code to print out the arithmetic sum of the two. In the example case, 27 would be printed.

19. Write a Python program that prompts the user for two integers, reads them in, and prints out the product, labeled.

20. Complete the definition of the function `prob`.
   ```python
   def prob(x, y):
       '''Return $x$ if $x > y$, and 0 otherwise.'''
   ```

21. Complete the definition of the function `upDown`.
   ```python
   def upDown(s):
       '''Prints out string s in upper and lower case. Examples:
       upDown('Hello') prints: HELLOhello
       upDown('3 cheers!') prints: 3 CHEERS!3 cheers!'''
   ```

22. Use your `upDown` function from the previous problem to print the following (write just one possible solution):
   ```python
   SAMPLEsample
   EXAMexam
   ```

23. Complete the for-loop and use your `upDown` function from above to print the following. You need to supply a list in the for-loop heading. Any case mixture is okay in your list:
   ```python
   SAMPLEsample
   EXAMexam
   HIhi
   LoLo
   ```

24. Complete the code for the following function so it matches its documentation:
   ```python
   def doubleList(numberList):
       '''For each of the numbers in the list numberList, print a line containing twice the original number. For example,
       doubleList([3, 1, 5]) would print
       6
       2
       10
       '''
   ```

25. Complete the code for the following function so it matches its documentation:
   ```python
   def rand2Digits(n, rand):
       '''n is a positive int, and rand is of type Random. Use rand to create a list of n random integers, each in the range 10 to 99 inclusive. Return the list.'''
   ```

Answers start on the next page
Exam 1 Review Problem Answers

1: x: 14, y: 8, z: 10

Here like in all the answers, the detailed steps are not required in an exam, but they might help you get partial credit. If you make a mistake somewhere in the middle! I show the just executed line number and the variable values that CodeLens would show. Instead of using arrows and replacing the last state, the values are in the same column as the variable name. Columns show the sequence of values for a variable. Details:

<table>
<thead>
<tr>
<th>line</th>
<th>x</th>
<th>y</th>
<th>z</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>8=5+3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>4=5-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td>14=10+4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>substitutes into format and prints result above</td>
<td></td>
</tr>
</tbody>
</table>

2: 3 2 3.5

14 divided by 4 is int 14//4=3 with a remainder of 14%4=2.
Because of the single '/' in last part, the result has a decimal point.

3: Print one less than each number in the list.

<table>
<thead>
<tr>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>8</td>
</tr>
</tbody>
</table>

4: 23 (concatenation of strings)

5: First the functions are remembered. Afterward they are called in the order given.

Lo
Hi
Hi

6: 10 Execution starts at line 5 -- after the definitions!

xxxx
30

step by step – does not show the spaces and newlines, not a complete substitute for the final answer!

<table>
<thead>
<tr>
<th>line</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>go to foo with parameter 7</td>
</tr>
<tr>
<td>1</td>
<td>x is 7</td>
</tr>
<tr>
<td>2</td>
<td>return 7+3 = 10</td>
</tr>
<tr>
<td>5</td>
<td>print 10</td>
</tr>
<tr>
<td>6</td>
<td>go to bar with parameters 'x' and 4</td>
</tr>
<tr>
<td>3</td>
<td>a is 'x' and n is 4</td>
</tr>
<tr>
<td>4</td>
<td>'x'*4 is xxxx - print it</td>
</tr>
<tr>
<td>6</td>
<td>done with line 6</td>
</tr>
<tr>
<td>7</td>
<td>to first function call - go to foo with parameter 2</td>
</tr>
<tr>
<td>1</td>
<td>x is 2</td>
</tr>
<tr>
<td>2</td>
<td>return 2+3 = 5</td>
</tr>
<tr>
<td>1</td>
<td>go to bar with parameters 5 and 6</td>
</tr>
<tr>
<td>3</td>
<td>a is 5 and n is 6</td>
</tr>
<tr>
<td>4</td>
<td>5*6 is 30 - print it</td>
</tr>
<tr>
<td>7</td>
<td>done with line 7</td>
</tr>
</tbody>
</table>
7: \[0, 1, 2\] (start with 0, ends before 3; list notation with square braces and commas)

8: Hello again! The sequence range(3) has 3 elements so the loop is repeated 3 times.
Hello again! Simple repeat loop: loop variable ignored.
Hello again!

9: 0 (range(4) contains 0, 1, 2, 3)
   1
   2
   3

10: ab Nested loops!
    ababab Inner loop completes in one iteration of outer loop,
    c and then completes again in next iteration of outer loop.
    ccc

The first time through the loop in line 1, s is 'ab', so when the body in lines 2-3 is run, 'ab' is repeated once and three times. The next time through the loop of line 1, s is 'c' and when lines 2-3 are executed, 'c' is repeated once and three times.

step by step – does not show the spaces and newlines, not a complete substitute for the final answer!

<table>
<thead>
<tr>
<th>line</th>
<th>s</th>
<th>n</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ab</td>
<td>first outer element</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>first inner element</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>print ab</td>
<td>3</td>
<td>print c</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>other inner element</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>print abab</td>
<td>3</td>
<td>print ccc</td>
</tr>
</tbody>
</table>

11a. C first two inequalities are false 5<3, 5>10: falls through to else
   b. B first true part is 12 > 10. Never get to else
   c. A stop at first test -2 < 3
   d. C both tests false as in part a.

step by step – does not show the spaces and newlines, not a complete substitute for the final answer!

<table>
<thead>
<tr>
<th>a.</th>
<th>line comment</th>
<th>b.</th>
<th>line comment</th>
<th>c.</th>
<th>line comment</th>
<th>d.</th>
<th>line comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5 &lt;3 false</td>
<td>1</td>
<td>12 &lt;3 false</td>
<td>1</td>
<td>-2 &lt;3 true</td>
<td>1</td>
<td>10 &lt;3 false</td>
</tr>
<tr>
<td>3</td>
<td>5 &gt;10 false</td>
<td>3</td>
<td>12 &gt; 10 true</td>
<td>2</td>
<td>print A</td>
<td>3</td>
<td>10 &gt; 10 false</td>
</tr>
<tr>
<td>5</td>
<td>print C</td>
<td>4</td>
<td>print B</td>
<td></td>
<td></td>
<td>5</td>
<td>print C</td>
</tr>
</tbody>
</table>

More on next page
Note the last if-statement is completely separate from the if-elif above, so the last test is always done. The final statement is outside any if-statement, so Z is always printed.

Step by step – does not show the spaces and newlines, not a complete substitute for the final answer!

part a

```
part c
1 5 == 5 false
2 print A
3 1 <5 and 5>2: true and true: true
4 print B
5 1>2 or 5>4: false or false: false
6 print C
7 print Z
```

part b

```
part d
1 1 == 1 true
2 print A
3 1>2 or 1>4: true or true: true
4 print B
5 print Z
```

13. 18 short version: 3+2+5+8 = 18
long version:
```
line n x comment
1 3 -
2 2 first value in list
3 5 5=3+2
2 second value in list
3 10 10=5+5
2 last value in list
3 18 18=10+8
4 prints 18
```

14: 1 Evaluates s, the squaring function, for each element in the list, and prints the partial sums.
```
10
35
```

15: 10 (numbers in list that are bigger than 4)
```
5
7
```

16: Answers 2.568 and 9.000 (Round to three places; always show 3 places)
17: 36

Steps:
Line s n comments
1  2   -
2  3 first element
3    3>3: false
5  6  2*3=6
2  5 next element
3    5>3: true
4 11  6+5 = 11
2  7 next element
3    7>3: true
4 18  11+7=18
2  2 last element
3    2>3: false
5 36  18*2=36
6   print 36 (after loop!)

18: print(x + int(ys))

19:  
x = int(input('Enter a integer: '))
y = int(input('Enter another integer: '))
print('Product:', x*y)

20: Body could be:
    if x > y:
        return x
    return 0

21: Body could be:
    print(s.upper()+s.lower())

22: # any mixture of cases OK in the two strings
    upDown('sample')
    upDown('exam')

23: # any mixture of cases OK; any consistent loop variable OK
    for word in ['sample', 'exam', 'hi', 'lo']:
        upDown(word)

24: Body could be:
    for num in numberList:
        print(2*num)

25: Body could be:
    rList = []
    for _ in range(n):
        rList.append(rand.randrange(10, 100))# note 100 so 99 included
    return rList