Comp 150 Exam 1 Overview.

Resources During the Exam

The exam will be closed book, no calculators or computers to help solve problems. You may bring **notes** on two sides of 8.5x11 inch paper (either both sides of one sheet, or two sheets written on single sides). Write this as you study! I test you on concepts, not memorized rote facts. *Bring* the facts if you like! *Learn* the concepts.

Main topics that may be on exam 1
Python Tutorial Chapter 1: See the summary at the end of the chapter.

How the Python topics get used:
1. In the tutorial you were asked to "predict and try". On an exam, you just get to predict final results or give a more complete explanation, playing computer. Follow fairly arbitrary code using the elements above, and show the results
2. Write a line of code translating an idea into Python, or put a few steps together.

Read the following before looking at either the problems or the solutions!

1. Study first, gathering your written notes. Look at the chapter summary and start by filling in any holes. 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, and will not get an idea how well you are prepared in general.
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.

The review problems are several times as long as an exam.

Sample problems start on the next page.

Review Problems for Chapter 1 using Python 3.2+ (Solutions follow the problems.)

1. What is printed by the Python code?

```
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?

```
print(14//4, 14%4, 14.0/4)
```

3. What is printed by the Python code?

```
print(2*'No' + 3*'!')
print(2 * ('No' + 3*'!'))
```

4. What is printed by the Python code?
   Be careful: Note the backslashes:

```
print('how\nis it\nnow')
```

5. What is printed by the Python code?

```
for z in [2, 4, 7, 9]:
    print(z - 1)
```

6. What is printed by the Python code?

```
print('2' + '3')
```

7. What is printed by the Python code?

```
def f1():
 print('Hi')
def f2():
 print('Lo')

f2()
f1()
f1()
```

8. What is printed by the Python code?

```
def func():
    print('Yes')

print('No')
func()
```

9. What is printed by the Python code?

```
def func(x):
    print(2*x)

func(5)
func(4)
```

10. What is printed by the Python code?

```
def func(x):
    return x - 1

print(func(3) * func(5))
```

11. What is printed by the Python code?

```
n = 3                    #1
for x in [2, 5, 8]:      #2
    n = n + x            #3
print(n)                 #4
```

12. What is printed by the Python code?

```
print(list(range(3)))
```

13. What is printed by the Python code?

```
for i in range(3):
    print('Hello again!')
```

14. What is printed by the Python code?

```
for i in range(4):
    print(i)
```

15. What is printed by the Python code?

```
def s(x):                #1
    return x*x           #2

for n in [1, 2, 10]:     #3
   print(s(n))           #4
```

16. What is printed by the Python code?

```
def s(x):                #1
    return x*x           #2

tot = 0                  #3
for n in [1, 3, 5]:      #4
   tot = tot + s(n)      #5
print(tot)               #6
```

17. What is printed by the Python code?

```
x = 2.5679
y = 9.0
print('Answers {:.3f} and {:.3f}'.format(x, y))
```

18. What is printed by the Python code?

```
d = dict()
d['left'] = '<<'
d['right'] = '>>'
print('{left} and {right} or {right} and {left}'.format(**d))
```

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

20. Given a string s, write a short expression for a string that includes s repeated five times.

21. 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.

22. Suppose you are given a list of words, *wordList*. Write Python code that will write one line for each word, repeating that word twice. For example if *wordList* is `['Jose', 'Sue', 'Ivan']`, then your code would print

```
Jose Jose
Sue Sue
Ivan Ivan
```

23. Write code to create a Python dictionary (the dict type). Add two entries to the dictionary: Associate the key 'name' with the value 'Juan', and associate the key 'phone' with '508-1234'

24. Complete the code for the following function so it matches its documentation:

```
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. Assuming a function `process` is already defined, write two lines of code, using a **for**-loop, that is equivalent to the following:

```
process('Joe')
process('Sue')
process('Ann')
process('Yan')
```

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 details are not required in an exam, but they might help you get partial credit if you make a mistake somewhere in the middle! Details:

```
line  x  y  z  comment
1     5  -  -
2        8     8=5+3
3     4        4=5-1
4           10
5    14        14=10+4
6               substitutes into format
                and prints result above
```

```
2.   3 2 3.5
```

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

```
3.  NoNo!!!
    No!!!No!!!
```

```
4. how
   is it
   now
```

In a string literal \n means newline.

```
5.  1
    3
    6
    8
```

Print one less than each number in the list.

```
6.  23
```

```
7.  Lo
    Hi
    Hi
```

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

```
8.  No
    Yes
```

First the function is remembered. It is only called after 'No' is printed.

```
9.  10
    8
```

First the function is remembered. Afterward it is called with x = 5, returning 10=2*5. Finally it is called with x=4, returning 8 = 2*4.

```
10.  8
```

(3-1)*(5-1) = 2*4 = 8. The function is called twice and the results are combined.

```
11.  18
```

details:
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       5 second value in list
3    10   10=5+5
2       8 last value in list
3    18   18=10+8
2         done with list and loop
4         prints 18
```

```
12.  [0, 1, 2]
```

start with 0, ends before 3

```
13.  Hello again!
     Hello again!
     Hello again!
```

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

```
14.  0
     1
     2
     3
```

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

```
15.  1
     4
     100
```

Evaluates s, the squaring function, for each element in the list, and prints the results.

```
16.  35   # 0 + 1*1 + 3*3 + 5*5
```

details:

```
line  tot n x comment
1-2           definition
3       0
4         1   first value in list
5             evaluate s(1), returns 1 = 1*1
        1     so tot = tot+1 = 0+1 = 1
4         3   next value in list
5             evaluate s(3), returns 9 = 3*3
   10     so tot = tot+9 = 1+9 = 10
4         5   last value in list
5             evaluate s(5), returns 25 = 5*5
   35     so tot = tot+25 = 10+25 = 35
4             done with list and loop
6             prints 35
```

```
17.  Answers 2.568 and 9.000
```

Substitutions into format string with floating point formats. Both show 3 decimal places because of the 3's in the floating point formats. Results are also *rounded* automatically: 2.568, not 2.567.

```
18.  << and >> or >> and <<
```

Formats with {key} substitute strings from the dictionary

19.

```
  x = int(input('Enter a number:  '))         # or some such prompt
  y = int(input("Enter another number:  "))   # or some such prompt
  print('The product is ', x*y)               # or some such label
```

20. `s*5    # or:   s+s+s+s+s`

21. `print(x + int(ys))`

22.

```
for word in wordlist:       # variable word is arbitrary
      print(word, word)     #   but must match here!
```

23.

```
d = dict()  # name d is arbitrary, but match it in the next lines
d['name'] = 'Juan'
d['phone'] = '508-1234'
```

24.

```
def doubleList(numberList):
    ''' skip repeating docs... '''
    for n in numberList:
        print(2*n)
```

25.

```
for name in ['Joe', 'Sue', 'Ann', 'Yan']:
    process(name)
```