Comp 170-400 Exam 1 Overview.

Resources During the Exam

The exam will be closed book, no calculators or computers, except as a word processor. In particular no Python interpreter running in a browser or separately. 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 through fopp chapter 14 (so not user defined classes):

1. Python types int, float, str, bool, list, dict, tuple. **Not** Turtle.
2. Converting between types.
3. Function definition and calling a function; "fruitful" and not fruitful.
4. Statements: assignment, for, while, if-elif-else, return.
5. Global functions max, min, len, type, print, input, range.
6. Operators: +. *, /, //, %, **, in, not in.
7. Boolean operations: and, or, not
8. Strings: indexing, as a sequence usable in a for-loop, slicing, adding; methods upper, lower, find, split, join, strip, count, format; syntax for formatting with field width and floats with precision
9. Lists: indexing; methods append, sort, pop, insert
10. Files: opening, file mode (read or write), methods read, readlines, write, close; *distinguishing* file name, file object, file contents.
11. Dictionaries: dictionary literal syntax, assigning to, reading from, get method, iterating through keys in a for-loop
12. 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, tracking state and individual statements, and copy just the exact characters printed, in the proper format, to the location of a final answer.
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 body. Hence understanding formal parameters is essential! Be sure to distinguish what is asked for: return vs. print!

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

1. Study first until you think you have everything nailed down, 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.

New sample problems start on the next page.

Review Problems for Exam 1 (Solutions follow the problems.)

1. What is printed? Be careful to follow the order of execution, not the order of the text!

```
def foo(x):              #1
    return x + 3         #2

def bar(a, n):           #3
    print(a*n)           #4

print(foo(7))            #5
bar('x', 4)              #6
bar(foo(2), 6)           #7
```

2. What is printed? Remember, a definition does not do anything by itself.

```
def foobar(nums):              #1
    new = list()               #2
    for num in nums:           #3
        new.append(num+2)      #4
    return new                 #5

def p(seq):                    #6
    for e in seq:              #7
      print(e, ':', end=' ')   #8

vals = foobar([2, 5, 12])      #9
p(vals)                       #10
```

3. What is printed?

```
x = 15                 #1
y = 7                  #2
while  x > 1:          #3
    print(x, y)        #4
    y -= 1             #5
    x -= y             #6
```

4. What is printed?

```
for s in ['ab', 'c']:    #1
  for n in [1, 3]:       #2
    print(s*n, end=' ')  #3
```

5. Complete the definition of the function `prob`.

```
def prob(x, y):
    '''Return x if x > y, and 0 otherwise.'''
```

6. Complete the definition of the function upDown.

```
def upDown(s):
    '''Prints out s in upper and lower case.  Examples:
    upDown('Hello') prints:  HELLOhello
    upDown('3 cheers!') prints:  3 CHEERS!3 cheers!
    '''
```

7. Use your upDown function from the previous problem to print the following (write just one possible solution):

```
SAMPLEsample
EXAMexam
```

8. Use a for-loop and your upDown function from above to print the following. Any case mixture is okay in your list:

```
SAMPLEsample
EXAMexam
HIhi
LOlo
```

9. Write interactive code that prompts the user for a word, and then calls upDown with that word, stopping (and printing nothing more) after QUIT is entered. The session could be the following (with user typing shown in ***boldface italics***):

```
Enter a word: Sample
SAMPLEsample
Enter a word: exam
EXAMexam
Enter a word: QUIT
```

10. Complete the definition of the function numbersBelow.

```
def doublesBelow(n, tooBig):
    '''Keep printing and doubling n, as long as the result is less
    than tooBig.  For example, doublesBelow(5, 25) would print
    5 10 20                                                    '''
```

11. Complete the definition below without using any if statements. Use Boolean operations only.

```
def xor(A, B):
    '''return the exclusive OR of boolean values A and B: true when
    A or B is true, but not both.  Examples:
        xor(False, True) returns True
        xor(False, False) returns False
        xor(True, True) returns False.    '''
```

Answers on the next page

Exam 1 Review Problem Answers

1. 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!

Line	comment
5	go to foo with parameter 7
1	x is 7
2	return 7+3 = 10
5	print 10
6	go to bar with parameters 'x' and 4
3	a is 'x' and n is 4
4	'x'*4 is xxxx - print it
6	done with line 6
7	inside first - go to foo with parameter 2
1	x is 2
2	return 2+3 = 5
1	go to bar with parameters 5 and 6
3	a is 5 and n is 6
4	5*6 is 30 - print it
7	done with line 7

2. `4 : 7 : 14 :` Execution starts at line 9 -- after the definitions!

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

Line	num	new	comment
9			go to foobar with parameter [2, 5, 12]
	num	new	local variable headings
1			nums is [2, 5, 12]
2		[]	
3	2		first in list
4		[4]	2+2=4, append to new
3	5		next in list
4		[4,7]	5+2=7, append to new
3	12		next (and last) in list
4		[4,7,14]	12+2=14, append to new
5			return [4, 7, 14]
9			make vals be [4, 7, 14]
10			send [4, 7, 14] to p
	e		local variable heading
6			seq is [4, 7, 14]
7	4		first in list
8			print `4 :` (stay on same line)
7	7		next in list
8			print `7 :` (stay on same line)
7	14		next (and last) in list
8			print `14 :` (stay on same line)

3. 15 7
9 6
4 5

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

line	x	y	comment
1	15		
2		7	
3			15>1 true: loop
4			print 15 7
5	6		7-1 = 6
6	9		15 - 6 = 9
3			9>1 true: loop
4			print 9 6
5	5		6-1 = 5
6	4		9-5 = 4
3			4>1 true: loop
4			print 4 5
5	4		5-1 = 4
6	0		4-4 = 0
3			0>1 false: skip loop

4. `ab ababab c 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!

line	s	n	comment
1	ab		
2		1	
3			print ab
2		3	
3			print abababab
1	c		
2		1	
3			print c
2		3	
3			print ccc

5. def prob(x, y):

5. def prob(x, y):
```
    if x > y:
        return x
    return 0
```

6. def upDown(s):
```
    print(s.upper()+s.lower())
```

16. upDown('sample') # any mixture of cases OK
```
upDown('exam')    # any mixture of cases OK
```

17. # any mixture of cases OK
```
for word in ['sample', 'exam', 'hi', 'lo']:
    upDown(word)
```

18. word = input('Enter a word: ')
```
while word != 'QUIT':
    upDown(word)
    word = input('Enter a word: ')
```

19. while n < tooBig:
```
    print(n, end=' ')  # all on same line
    n = 2*n
```

21.
```
    #direct translation
    #          A or B  but  not  both
    return (A or B) and not(A and B)
#or
    return A and not B or B and not A #both true cases
```

actually Python has an operator for this: A ^ B