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 topics that may be on the exam from the text through chapter 12:

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, formal and actual parameters.
4. Statements: assignment, for, while, if-elif-else, return.
5. Global functions max, min, len, type, input, range, print – with sep and end keyword arguments.
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 floats with precision
9. Lists: indexing, slices; methods append, sort, pop, insert; can *skip* del and list comprehensions
10. Files: opening, file mode (read or write), methods: read, write, close; *distinguishing* file name, file object, file contents; iterating through file lines.
11. Dictionaries: dictionary literal syntax, assigning to, reading from, get method, iterating through keys in a for-loop
12. Functions in module random: random.random, random.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, completely tracking all 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 naiuled 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.

Review problems start on the next page.

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

1. Suppose the file 'prob1.txt' contains the two lines
Hello
Mom

What is printed by
```
fin = open('prob1.txt', 'r')
s = fin.read()
print(s.upper())
```

2. What will be the contents of the file prob2.txt? Indicate any blanks or newlines clearly.
```
    fout = open('prob2.txt', 'w')
    words = ['Hi', 'there', 'Mom']
    for w in words:
        fout.write(w)
    fout.close()
```

3. What will be printed by the function calls in parts a-d?
```
def comp(x):
    if x < 3:        #1
        print("A")   #2
    elif x > 10:     #3
        print("B")   #4
    else:
        print("C")   #5
```

a. `comp(5)` b. `comp(12)` c. `comp(-2)` d. `comp(10)`

4. What will be printed by the function calls in parts a-d?
```
def comp2(x, y):
    if x == y:                #1
        print("A", end='')    #2  empty end string each time
    elif x  < 5 and y > 2:    #3
        print("B", end='')    #4
    if x > 2 or y > 4:        #5
        print("C", end='')    #6
```

a. `comp2(5, 3)` b. `comp2(5, 5)` c. `comp2(1, 5)` d. `comp2(1, 1)`

5. What is printed? Here **end** is one space.
```
x = 1                    #1
while  x < 5:            #2
    print(x, end=' ')    #3
    x = x + 2            #4
```

6. What is printed? Carefully follow the execution sequence! Here **end** is one space.
```
for x in [30, 40]:        #1
  for y in [1, 2, 3]:     #2
    print(x+y, end=' ')   #3
  print()                 #4
```

7. What is printed? Here **end** is one space.
```
for n in [1, 3]:             #1
  for s in ['a', 'b']:       #2
    print(s*n, end=' ')      #3
```

8. What is printed by the Python code?
```
nums = list()             #1
for i in range(4):        #2
   nums.append(2*i)       #3
print(nums)               #4
```

9. What is printed?
```
x = 0                     #1
while x < 10:             #2
    x = 2*x + 1           #3
    print(x, end=' ')     #4
```

10. What is printed? .

```
s = 'Y'              #1
while len(s) < 3:  #2
    s = 2*s          #3
    print(s)         #4
```

11. What is printed?

```
print(list(range(2, 5)))
print(list(range(2, 14, 4)))
```

12. What is printed?

```
words = ['A', 'short', 'list'] #1
print(len(words))              #2
for s in words:                #3
    print(len(s))              #4
```

13. What is printed? .

```
x =  37                #1
while x > 7:           #2
    x = x - 2          #3
    print(x, end=' ') #4
    if x > 12:         #5
        x = x - 10     #6
```

14. 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
```

15. 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
```

16. What is printed?

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

17. What is printed?

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

18. Write code that inputs a number from the user and prints "High" if it is over 100, "Low" if it is less than 50, and "In between" otherwise.

19. Complete the function definition.

```
def double(numlist):
    '''One number to a line, print twice each number in the numlist.
    For example double([3, 7, 4]) prints
      6
      14
      8                     '''
```

20. Modify the previous problem so it prints out a sentence stating the multiplication fact for each number. Use a format string. For instance the example above would print

```
   Twice 3 is 6.
   Twice 7 is 14.
   Twice 4 is 8.
```

Note: A number of later coding problems ask for just a sequence of items to be printed on *one* line. By default, items should be separated by a space. An invisible blank after the final item is *optional*.

21. Complete the Python function below.

```
def printWords(wordlist):
    '''Print on one line the words in wordlist.
    For example, if words is ['he', 'is', 'his', 'hero'],
    printWords(words) prints:  he is his hero              '''
```

22. Suppose `num`, `lowVal`, and `highVal` are variables with existing numeric values, and `lowVal` <= `highVal`. Write an expression that is `True` if `num` is in the interval from `lowVal` to `highVal`, allowing the endpoint values `lowVal` and `highVal`. For instance, if `lowVal` is 2 and `highVal` is 5, your expression should be `True` if num is 2, 3, 4.4 or 5, but `False` if num is -1, 1.9, 5.1, 7 or 100000.

23. Complete the function definition.

```
def numbersBetween(numList, lowVal, highVal):
    '''Print on one line the numbers in numList that lie in the
          interval from lowVal to highVal, allowing lowVal and highVal
    For example,
      numbersBetween([2, 5, 1], 3, 5) prints:  5
      numbersBetween([2, 5, 1, 7, 4], 2, 6) prints:  2 5 4       '''
```

24. Modify the previous problem to print nothing, but put the selected numbers in a list, and return the list.

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

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

26. 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!   '''
```

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

```
SAMPLEsample
EXAMexam
```

28. 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
```

29. 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
```

30. 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                                                  '''
```

31. 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.     '''
```

32. Complete this function.

```
def firstWords(filename):
    '''A file with name passed in filename contains a        hi there
    multi-word phrase on each line, with words separated     go for it
    by blanks. Open the file; create a list of the first     yes to that
    word in each line; close the file and return the list.   I agree
    For example, with the file containing the lines to the
    right, ['hi', 'go', 'yes', 'I'] would be returned    '''
```

33. Write code to print 500 random integers, one per line, in the range 0 to 10000, inclusive. Assume the random module has already been imported.

34. What is printed:

```
x = 17
y = 5
print(x/y, x//y, x%y, 2**3, x+y)
print(str(x) + str(y))
x, y = y, x - y
print(x, y)
print('{:.1} {:.3}'.format(9.7531, 9.7531))
s = 'abcde'
print(3 in [1,3, 7], 'bc' not in s, s[1], s[2:4])
print(s.find['c'], s.find('g'), s.split('c'))
print('/'.join(list(s)))
v = [4, 9, 2, 6]
c = v[:]
w = v.pop()
print(w, v)
c.sort()
print(c)
d = {'a':2, 'x':7, 'q':3}
d['a'] = 5
print(d['x'], d['a'])
```

Answers start on the next page

Exam 1 Review Problem Answers

1. HELLO
   MOM
2. HithereMom (no spaces or new lines)

3a . C first two tests 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!

line comment	line comment	line comment	line comment
1 5 <3 false	1 12 <3 false	1 -2 <3 true	1 10 <3 false
3 5 > 10 false	3 12 > 10 true	2 print A	3 10 > 10 false
5 print C	4 print B		5 print C

4a. C b. AC c. BC d. A
Note the last if statement is completely separate from the part above, so the last test is always done. The middle test is only true if both comparisons are true. The last test is true if either comparison is true.

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

line	comment
	part a
1	5 == 3 false
3	5 <5 and 3>2: false and true: false
5	5>2 or 3>4: true or false: true
6	print C
	part b
1	5 == 5 true
2	print A
5	5>2 or 5>4: true or true: true
6	print C
	part c
1	1 == 5 false
3	1 <5 and 5>2: true and true: true
4	print B
5	1>2 or 5>4: false or true: true
6	print C
	part d
1	1 == 1 true
2	print A
5	1>2 or 1>4: false or false: false

5. **1 3**
x is printed before being increased, so the first value is printed. The last value of x is 5, but x becomes that after the last time it is printed.

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

line	x	comment
1	1	
2		1<5 true: loop
3		print 1
4	3	
2		3<5 true: loop
3		print 3
4	5	
2		5<5 false: skip loop

6. 31 32 33
41 42 43

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

line	x	y	comment
1	30		first in outer list
2		1	first in inner list
3			30+1 = 31; print 31 (stay on line)
2		2	next in inner list
3			30+2 = 32; print 32 (stay on line)
2		3	last in inner list
3			30+3 = 33; print 33 (stay on line)
2		-	no more in list - done with inner loop
4			print (advance to new line)
1	40		next in outer list
2		1	start again with first inner list element
3			40+1 = 41; print 41 (stay on line)
2		2	next in inner list
3			40+2 = 42; print 42 (stay on line)
2		3	last in inner list
3			40+3 = 43; print 43 (stay on line)
2		-	no more in list - done with inner loop
4			print (advance to new line)
1	-		done with outer loop

7. a b aaa bbb

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

line	n	s	comment
1	1		first in outer list
2		a	first in inner list
3			print a (stay on line)
2		c	second in inner list
3			print c (stay on line)
2............		-..	no more in list - end inner loop
1	3		second in outer list
2		a	start again - first in inner list
3			print aaa (stay on line)
2		c	second in inner list
3			print ccc (stay on line)
2............		-..	no more in list - end inner loop
1			no more in list -- done with outer loop

8. **[0, 2, 4, 6]**

Remember square brackets and commas. Note range(4) is the sequence 0, 1, 2, 3.

line	nums	i	comment
1	[]		
2	[]	0	first in list
3	[0]	0	append 2*0 = 0
2	[0]	1	next in list
3	[0,2]	1	append 2*1 = 2
2	[0,2]	2	next in list
3	[0,2,4]	2	append 2*2 = 4
2	[0,2,4]	3	last in list
3	[0,2,4,6]	3	append 2*3 = 6
2	[0,2,4,6]	3	done with list
4	[0,2,4,6]	3	print [0, 2, 4, 6]

9. **1 3 7 15**

line	x	comment
1	0	
2	0	0 < 10 is True; loop'
3	1	2*0+1 = 1
4	1	print 1 (stay on same line)
2	1	1 < 10 is True; loop'
3	3	2*1+1 = 3
4	3	print 3 (stay on same line)
2	3	3 < 10 is True; loop'
3	7	2*3+1 = 7
4	7	print 7 (stay on same line)
2	7	7 < 10 is True; loop'
3	15	2*7+1 = 15
4	15	print 15 (stay on same line)
2	15	15 < 10 is False; skip loop

10. **YY**
YYYY

line	s	comment
1	Y	
2	Y	1 < 3 is True; loop'
3	YY	2*'Y' is 'YY'
4	YY	print YY
2	YY	2 < 3 is True; loop'
3	YYYY	2*'YY' is 'YYYY'
4	YYYY	print YYYY
2	YYYY	4 < 3 is False; skip loop

11. **[2, 3, 4]** # start with 2, end before 5
[2, 6, 10] # start with 2, end before 14, jumps of 4
They are list objects, hence they include the square brackets and commas when printed.

12. **3** First prints the length of the list (3 elements, each a string)
1 Next loop through each word, and print the length of each word:
5 Length for a string is measured by the number of characters.
4

13.
```
35 23 11 9 7
```

line	x	comment
1	37	
2		37 > 7 is true; loop
3	35	37-2=35
4		print 35 (all on same line)
5		35 > 12 is true
6	25	35-10=25
2		25 > 7 is true; loop
3	23	25-2=23
4		print 23 (all on same line)
5		23 > 12 is true
6	13	23-10=13
2		13 > 7 true; loop
3	11	13-2=11
4		print 11 (all on same line)
5		11 > 12 is false
2		11 > 7 true; loop
3	9	11-2=9
4		print 9 (all on same line)
5		9 > 12 is false
2		9 > 7 true; loop
3	7	9-2=7
4		print 7 (all on same line)
5		7 > 12 is false
2		7 > 7 false; done

14.
```
10
xxxx
30
```

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

Line	comment
1-4	remember function definitions
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

15.
```
4 : 7 : 14 :
```

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

Line			comment
1-8			remember function definitions
9			go to foobar, passing [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)

16.
```
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

17. `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 ababab
1	c		
2		1	
3			print c
2		3	
3			print ccc

18.
```
# Creating a float safer:  Not clear if the number must be an int.
x = float(input("Enter a number: "))
    if x > 100:
      print("High")
    elif x < 50:
      print("Low")
    else:
      print("In between")
```

19.
```
for num in numlist:
    print(2*num)
```

20.
```
for num in numlist:
    print("Twice {} is {}.".format(num,2*num)) #print final period; no space before
```

21.
```
for word in wordlist:      # print(' '.join(wordlist) # single line is
    print(word, end= ' ') #   alternate - looks same; could add a final ' '
```

22.
```
lowVal <= num <= highVal
# lowVal <= num and num <= highVal  #alternate
```

23.
```
for num in numList:
    if lowVal <= num <= highVal:
        print(num, end = ' ')
```

24.
```
chosen = []
for num in numList:
    if lowVal <= num <= highVal:
        chosen.append(num)
return chosen
```

```
25. if x > y:
       return x
   return 0  # else clause not needed but OK after return

26.  def upDown(s):
        print(s.upper()+s.lower())

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

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

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

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

31. #direct translation
    #       A or B  but  not  both
    return (A or B) and not(A and B)
#or note both true cases:
    return A and not B or B and not A

# actually Python has an operator for this:  A ^ B

32.
def firstWords(filename):
    f = open(filename):
    words = []
    for line in f:
        words.append(line.split()[0]) # could use string find and slice, too
    f.close()
    return words

33.
for i in range(500):
   print(random.randrange(10001)) # one past the last allowed

34.

3.4 3 2 8 22
175
5 12
9.8 9.753
True False b cd
2 -1 ['ab', 'de']
a/b/c/d/e
6 [4, 9, 2]
[2, 4, 6, 9]
7 5
```