

Computer Science 1 — CSCI 1100

Test 1 Overview and Practice Questions

Fall Semester, 2019

Important Logistical Instructions:

- Test 1 will be held **Thursday, September 26, 2018**.
- Most students will take the exam from 6:00 - 7:30 pm. Students who provided us with an accommodation letter indicating the need for extra time or a quiet location will take the exam starting at 4:30 pm.
- Room assignments will be posted on Submitty by Tuesday night, September 24. If you have an accommodation and hear from Shianne Hulbert, please ignore any information in Submitty and follow her directions instead. If you do not, we cannot give you accommodations.
- Students **MUST**:
 - **Go to their assigned rooms.**
 - **Bring their IDs to the exam.**
 - **Sit in the correct section.**

Failing to do one of these may result in a **20 point** penalty on the exam score. Failure to do all can cost up to **60 points**.

Overview

- Coverage is Lectures 1-7, Labs 0-3, HW 1. Material from tuples and modules are part of this test. Images will not be covered.
- No calculators, no textbook, no classnotes, no electronics of any kind! BUT, you may bring a one-page, double-sided, 8.5" x 11" "crib sheet" sheet with you. You may prepare this as you wish, and you may work in groups to prepare a common crib sheet. Each of you must have your own copy during the test, you must sign your name to it, and you must turn it in at the end of the test. **They will not be returned. If you want it for the next test, please make a copy.**
- We will assume you know (perhaps via your crib sheet) the following mathematical functions, some of which are in the `math` module: `abs`, `ceil`, `float`, `int`, `max`, `min`, `round`, `sqrt`, `trunc`. This is a partial list, If we covered it in class, or you used it in your lab, lecture exercises or homework, you should know it.
- We will assume you know the following functions associated with the data of type `str`: `len`, `+`, `*` (by an integer), `count`, `title`, `find`, `capitalize`, `replace`, `str`, `strip`, `lstrip`, `rstrip`, `upper` and `lower`. This is a partial list, If we covered it in class, or you used it in your lab, lecture exercises or homework, you should know it.
- We will assume you know how to convert between the different data types discussed in class using `int`, `float`, and `str`
- We will assume you know boolean relations, and the operators `and`, `or` and `not` and can write or evaluate expressions made from them.
- Python syntax will be important on this test. On tests later in the semester it will be less important, but not entirely ignored.
- Below are **many** sample questions, far more than will be on the test. Solutions to most of the problems will be posted on-line on Monday, September 23th. These posted solutions will **not** include problems involving output from Python — test these out yourself!
- Please note that any question that requires you to write Python code can be solved in at most 10-12 lines, and may be much shorter. We will not penalize longer programs, but learn to write short programs, it will save you time during the exam.

- The test questions will be closely related to these practice problems **and** to problems from the homework, the labs, the lecture itself, and the lecture exercises. They may be harder or simpler than the ones here. Make sure you practice all the material we covered so far this semester.
- We have also posted the Spring 2017 first exam, updated to Python 3. Pay particular attention to the syntax, one-liners and what are the output sections. It is likely we will use similar sections in this exam.
- Remember, the exam is timed. You should be able to solve each programming problem in about 5-8 minutes. Time yourself when solving them, find out where you are spending too much time and go over the related material.
- *How to study?*
 - First, make sure you know the syntax of all the different programming constructs we have learned. You cannot construct solutions if you do not know the building blocks. Practice them like you would a foreign language: memorize them. A good approach is to keep typing them into the Python shell and try new variations. This will also help when you are trying to track down syntax errors in your Lecture Exercises, Labs and Homeworks.
 - Work through the sample questions, writing out solutions! Read the solutions we provided only as a last resort. Remember: reading a solution is **much easier** than actually writing one. You are graded on the writing part.
 - You are encouraged to work on the practice problems with other students, but ask yourself if you understand it enough to solve it on your own. Try to replicate a solution you worked on with someone else a little while later, without looking at anyone's solutions.
 - Review and re-do lecture examples, lecture exercises, labs and homework problems.
 - Identify the problems that cause you difficulty and review lecture notes and background reading on these topic areas. Go to office hours and ask to review any concepts you did not understand.
 - Use the Spyder IDE / Python interpreter extensively to help you understand what is happening. Notice differences between the interpreter and the programs (you must print to output in the programs). But, practice writing solutions using pencil and paper!
- **Advanced warnings:**
 - Students who arrive more than 15 minutes after the start of the exam will not be allowed to take the exam.
 - We will not answer questions from students in the middle of the test unless there is a mistake in a question.
 - Once the test starts, students may not leave (**even for bathroom breaks!**) until they turn in their tests, and once they leave they may not return.

Questions

1. What is the **exact** output of the following Python code? Show the output to the right of the code. Also, what are the arguments, the local variables, and the parameters in the code?

```
x=3
```

```
def do_something(x, y):
    z=x+y
    print(z)
    z += z
    print(z)
    z += z * z
    print(z)
```

```
do_something(1, 1)
y=1
do_something(y,x)
```

Output:

```
2
3
12
4
5
50
```

Do all compressed
Functions take
precedence?

The "+=" takes
precedence over the "*" "

2. Assuming we type the following directly into the Python interpreter, please fill in the **exact** output, making your answer clear:

```
>>> x = 5
>>> y = 12
>>> z = y-x
>>> s = "Grail"
>>> x+y > z * len(s)
```

```
>>> x<y and not z > len(s)
```

```
>>> u = "grail"
>>> u == s
```

```
>>> u == s or len(u) == len(s)
```

3. Write a short segment of Python code that asks the user for a positive integer, reads the value, and generates an output error message if the user has input a negative number or 0.
4. Write a function called `spam` that takes as arguments 4 integers and returns a string listing all the numbers and their average as a float. For example:

```
>>> s = spam(3, 10, 4, 2)
>>> s
'The average of 3, 10, 4, and 2 is: 4.75'
```

What is the output if you execute

```
>>> print(s)
```

instead?

5. What is the output of the following program?

```
def spam(a1,b1,a2,b2):
    if (a1 == a2) and (b1 > b2):
        return 1
    else:
        return 0

def egg(a1,b1,a2,b2):
    if (a1 > a2) and (b1 == b2):
        return 0
    else:
        return 1

a1 = 3
b1 = 4
a2 = 6
b2 = 4

print(spam(a2, b2, a1, b1))
```

```

print(egg(a1, b1, a2, b2))

c = spam(a1, b2, a2, b1)

print(c)

c += egg(a1, b2, a2, b1)

print(c)

```

6. Write a Python function called `right_justify` that takes two strings and an integer, `n`. It should print the two strings on two consecutive lines. In the output, the strings should be right-justified and occupy at least `n` spaces. If either of the strings is longer than `n` then `n` should be increased to the maximum length of the two strings. As examples, the call

```
right_justify( 'Bike', 'Baseball', 15 )
```

should output

```

        Bike
    Baseball

```

while the call

```
right_justify( 'Bike', 'Baseball', 5 )
```

should output

```

        Bike
    Baseball

```

7. Write a section of Python code that reads a string into variable `mystr` and then reads an integer into variable `num`. The code should then print out the value of `mystr` repeated `num` times in a line, and then repeats the line `num` times. Note: you may **not** use `for` loops (or while loops for that matter) on this problem.

As an example, given `mystr='Python'` and `num=4`, your code should output:

```

PythonPythonPythonPython
PythonPythonPythonPython
PythonPythonPythonPython
PythonPythonPythonPython

```

8. Chris, a carpenter, takes `w1` weeks and `d1` days for job 1, `w2` weeks and `d2` days for job 2, and `w3` weeks and `d3` days for job 3. Assuming `w1`, `d1`, `w2`, `d2`, `w3`, `d3` are all variables that have been assigned integer values, write a segment of Python code that calculates and outputs Chris's average number of weeks and days per job? Output integer values, and do not worry about rounding numbers. For example, given

```

w1 = 1
d1 = 6
w2 = 2
d2 = 4
w3 = 2
d3 = 6

```

The output should be

average is 2 weeks and 3 days

9. The following Python code is supposed to calculate and return the surface area of a cylinder by adding the area of the top and bottom circles to the area of the face of the cylinder (lateral area). There is at least one syntax error and at least one semantic error in this code. Identify each error in the code by rewriting the line and say whether it is a syntax error or a semantic error.

```
from math import pi
def surface_area(radius, height)
    # Calculate base area for each end of the cylinder
    base_area = (pi * radius)**2

    # Calculate lateral area
    lateral_area = 2*height*radius*pi

    # Calculate surface area
    surface_area = lateral_area + 2base_area
    return surface_area
```

10. This question has two parts. First write a function called `longest_first` which has two strings as parameters. The function should print two lines of output. The first line should be the longest word, with '*' before it and '*' after it. The second line should be the shortest word, also framed, with as many ! as needed to fill the space. For example, the call

```
longest_first( 'car', 'butterball' )
```

should print

```
* butterball *
* car!!!!!! *
```

You may assume the words are not the same length. The second part of the question is to write code that asks the user for two strings, reads the strings in, and calls `longest_first` to generate the output.

11. Assuming we type the following directly into the Python interpreter, please fill in the **exact** output of the last command, making it clear what is your answer (as opposed to scratch work). There are no syntax errors here.

Part a <pre>>>> c = 6 >>> s = 'ab' + str(c) >>> s *= 4 >>> print(s)</pre>	Part b <pre>>>> x = 15 >>> y = x // 6 >>> z = x % 6 >>> y+z</pre>
Part c <pre>>>> y = 5 >>> z = 4 >>> z += 2 ** (z-1) - y * 3 >>> z</pre>	Part d <pre>>>> p = '''a b\t c''' >>> t = 19 // len(p) >>> t</pre>
Part e <pre>>>> x = 4 >>> y = 2 % x >>> if x*y > 6: >>> z = 8 >>> else: >>> z = 7+y >>> z</pre>	Part f <pre>>>> s = "3 4" *2 + '34'*3 >>> print(s.count('43'))</pre>

12. The formula for volume of a cone is given by: $(1/3)\pi r^2 h$.

Write a function called `cone_volume` that takes as input two numbers, the radius (`r`) and height (`h`) of a cone, and returns the volume of the cone (printing nothing). The function must not use any global variables.

Now write a program that reads from the user the radius and height of a cone using `input`, and uses the function you wrote to compute and then print the volume of a cone. Assume the user enters valid floating point numbers.

Here is an example run of your program (watch out for formatting of the output float):

```
Radius ==> 2.5
Height ==> 5
Volume: 32.72
```

13. For each of the following Python code segments, determine if it uses correct syntax. If there is a syntax error, circle and explain the **first** syntax error.

```
x = 80
y = 'oak'
z = 'tree'
print(x + ' ' + y, z + 's')
```

```
name = "Alice"
print('Your name has', len(name), + "letters.")
```

```
full name = 'total supermoon eclipse'
full name.capitalize()
print(full name)
```

```
cnt = input('Enter a number => ')
print('Hello!' * (cnt+cnt))
```

```
x = 5*y-6
y = 3
print(x,y)
```

14. Write a program that reads an input containing a float, then prints the square root of the input number and the square root of the square root. Remember, the square root function comes from the `math` module.

Here is an example output of the program:

```
Enter a number => 16
Sqrt(16.0) = 4.0
Sqrt(4.0) = 2.0
```

15. Write a function called `emphasize(s1,s2,s3)` that takes three strings as input. The function prints the first two strings each on a separate line without alteration, then prints five lines with a single dot ('.'), and then the final string in upper case on the last line, preceded by 10 dots.

Here is an example run of your function:

```
>>> emphasize('This','is','Sparta!')
This
is
.
.
.
.
.
.....SPARTA!
```

16. Write a piece of code that reads from the user first a sentence and then a word to be censored. Your code must replace all occurrences of the censor word with a number of stars equal to the length of the word, and then print **Censored sentence:** and then the resulting sentence (note that the replacement is case sensitive).

Here is an example run of your program:

```
Sentence => enough is enough. I have had it with these monkey fighting snakes on this monday-to-friday plane.
Word => enough
Censored sentence:
***** is *****. I have had it with these monkey fighting snakes on this monday-to-friday plane.
```

17. Write a program that reads a three digit number and determines one number composed of the first and third digits of this number and another one composed of the third and first digits. It then prints both numbers and the result of multiplying them. Use the integer divide and modulo operators to get the digits.

Here is an example run of your program:

```
Number ==> 123
13 * 31 = 403
```

18. Suppose a health tracking system like Fitbit stores your activity in a string like '++-*****+--**' where a plus sign (+) means you went up 10 stairs (consumes 3 calories), a minus sign (-) means you went down 10 stairs (consumes 2 calories) and a star (*) means you walked 100 steps (consumes 5 calories).

Write a function called `find_cal(activity)` that takes as input a string like this one and returns the total calories you have consumed. Do not use loops.

Now write a program that reads from the user an activity string and uses the function you wrote to compute and then print the total calories consumed.

Here is an example run of your program:

```
Enter activity => ++-*****+--**
You consumed 43 calories
```

19. Assume each of the following is an individual program. Write the **exact** output of these programs in the area provided. Show your work at each step in the scratch area provided for partial credit. Note: there are no syntax errors in the code provided in this question.

<p>Part a</p> <pre>x = 5 y = x + 3 x = x + 1 z = y % x w = float(y//x) print('x: {:d}, y: {:d}'.format(x, y)) print('z: {:d}, w: {:.2f}'.format(z, w))</pre> <p>Output:</p>	Scratch area:
<p>Part b</p> <pre>def func(x): print('-'*x, x**2) x=2 func(3) print(x) print(func(4))</pre> <p>Output:</p>	Scratch area:

<p>Part c</p> <pre>def mystring(s): return s + '\n' def yourstring(s): return s * len(s) print(yourstring(mystring('cat'))) print(mystring(yourstring('fish')))</pre> <p>Output:</p>	<p>Scratch area:</p>
<p>Part d</p> <pre>s = "-" e = "\n3\n2\n1\nLiftoff!" print(10, 9, 8, 7, 6, 5, 4, sep=s, end=e)</pre> <p>Output:</p>	<p>Scratch area:</p>

20. Given the following versions of the import command, write lines of python to print out the value of `pi`, `50.5`, `e`, `log2(10)` using math functions. Not all values will be available with all import statements.

(a) `from math import sqrt, pi`

(b) `from math import sqrt, pi, e, log`

(c) `from math import *`

(d) `import math as m`

(e) `import math`

21. What is the output of the follow commands given that `s = (1, 3, 5, 7)` and `t = 2, 4`?

(a) `>>> s + t`

(b) `>>> s[0] + t[1]`

(c) `>>> x, y = t`
`>>> t = y, x`
`>>> t`

(d) `>>> t[1] = 4`

22. Assume you are writing a file `oracle.py` with one function in it called `predict(pair)`, where `pair` is a tuple (`question`, `shakes`). `question` is the question and `shakes` is an integer number of shakes.

Your function should return 1 of 3 strings 'Yes' is returned if the length of the question and the number of shakes are both even. 'No' is returned if they both are odd, 'Outlook Hazy' if one is odd and the other even.

23. Now write a main program (in a separate file) to import your oracle, input a number of shakes from the user, and then print out the answer to the question 'Is this test hard?'. Here is a sample run:

```
Shakes: 1
Outlook Hazy
```