CSCI 1300 Spring 2013 - Test 1

Your name: _____

Your 9-digit student id: _____

Your TA's name: _____

1. (2 points) You just started a new command line session. Your current working directory is /home/user/. Your python program is in /home/user/code/my_groovy_program.py. Write the sequence of command line statements that are necessary to change directories and run your program using the Python interpreter.

cd code
python my_groovy_program.py

another way of doing it:

cd code
./my_groovy_program.py

2. (2 points) Your file 'test-1-code.py' looks like this:

```
1 if my_num == 4
2     print "Yay"
3 else
4     print "Not four."
```

When you run the program is gives the following problem report:

```
  File "test-1-code.py", line 1
    if my_num == 4
                 ^
SyntaxError: invalid syntax
```

What is the bug? Write code that fixes the bug, or (if possible) modify the code above to prove that you know what's wrong. Note that there might be two bugs of the same kind.

It is missing the colons at the end of lines 1 and 3.

3. This is a multi-part question based on the following code, which is syntactically correct and does print something out—it just doesn't print out what it is supposed to, according to the comment that documents the desired behavior:

```
 8 # find the sum of all the given
 9 # numbers divisible that are by six
10 numbers = range(25, 40)
11 result = 0
12 for num in numbers:
13     if num % 6 == 1:
14         result = num
15 print "Result: " + str(result)
```

(a) (2 points) On line 10, we create a list. What are the contents of this list? You don't have to enumerate them all, just describe what it is.

Numbers 25 to 39 inclusive.
We did give credit for 25 to 40, but that isn't right.

(b) (1 point) Without changing the code, what does it print out on line 15?

It prints 37. This is one plus the largest number in the range that is divisible by six.

(c) (2 points) There are two bugs in the above code. One is on line 13, and the other is on line 14. In the space below, write what those lines should both be.

```
13   if num % 6 == 0:
14       result = result + num
```

(d) (1 point) Now that you've fixed the code, what is the correct value that it prints on line 15?

It prints 66.

The two numbers in the range [25, 40) that are divisible by six are 30 and 36. Their sum is 66.

4. This question is all about the following syntactically correct code:

```
19 def is_purple(num):
20     if num > 100:
21         return True
22     if num < 40:
23         return False
24     for x in range(2, num):
25         if num % x == 0:
26             return False
27     return True
```

A second way to do this is to use the print function with several params: print "Is 53 purple?", is_purple(53)

(a) (1 point)

Let's say I have the following function call.

```
print "Is 53 purple?" + is_purple(53)
```

... but when we run that line of code, Python complains:

```
TypeError: cannot concatenate 'str' and 'bool' objects
```

What does the above error mean? How would we fix it?

The string and boolean objects can not be combined using the plus (+) operator. We have to cast the bool to a string: print "Is 53 purple?" + str(is_purple(53)

(b) (1 point) Is 33 purple? (yes/no)

No because it is less than 40.

(c) (1 point) Is 400 purple? (yes/no)

Yes because it is larger than 100.

(d) (1 points) Describe (1 or 2 sentences) which numbers are purple.

Numbers larger than 100 are purple.
Prime numbers larger than 40 are purple.

Instead of talking about primes you could also say that numbers above 40 that are divisible by anything below them are purple.

5. Consider the following syntactically correct function definition:

```
19 def is_purple(num):
20     if num > 100:
21         return True
22     if num < 40:
23         return False
```

Describe in a sentence what role the following special words play:

(a) (1 point) **def**

starts the function definition

(b) (1 point) **is_purple**

gives the function the name 'is_purple'

(c) (1 points) **num**

names the first and only input parameter to the function

(d) (1 point) **return**

stops function execution & returns a value

6. (4 points) Write Python code that prints the square of 0, 1, 2, 3, and 4, in that order *using a for-loop*. There are a few ways of doing this. The only requirement is you use a for-loop.

```
for x in [0, 1, 2, 3, 4]:
    print x * x
```

```
for x in range(5):
    print x * x
```

7. (4 points) Write a function called 'print_third' that takes a list as input and returns the third item in that list. If the list doesn't have at least three elements, it should return None.

Note: you can use the following code to get the length of a list called my_list:

```
size = len(my_list)
```

'size' is now an integer that could be 0, 1, 2, or anything larger, depending on how many things are contained in my_list.

Any of these work. There may be other solutions.

Remember to make your function syntactically correct!

```
def print_third(foo):
    if len(foo) < 3:
        return None
    return foo[2]
```

```
def print_third(foo):
    if len(foo) < 3:
        return None
    return foo.pop(2)
```

```
def print_third(foo):
    if len(foo) < 3:
        return None
    ret = None
    count = 0
    for val in foo:
        ret = val
        if count == 2:
            return ret
        count = count + 1
```

8. (4 points) Write a function called 'years_left' that takes a single number as input that represents a person's age. The function should return an integer that is the number of years that person has until retirement. Assume that the retirement age is 65.

```
def years_left(age):
    return 65 - age
```

```
def years_left(age):
    if age >= 65:
        return 0
    return 65 - age
```