REMINDERS

- Exam 1 will be held **Monday, March 2, 2015**. Most of you will take the exam from 6:00-7:50PM in DCC 308.
- Students who have provided Prof. Goldschmidt with an accommodation letter requiring extra time will take the exam starting at 5:00PM in DCC 239.
- You **MUST BRING YOUR RPI ID** to the exam. Missing IDs will result in a 20-point penalty.

SOLUTIONS

Below are solutions to the practice problems. Please be aware that there may be more than one way to solve a problem, so your answer may be correct despite being different from ours.

Overview

- No calculators, books, electronics of any kind, etc.! You may bring a one-page, double-sided, 8.5” x 11” “crib sheet” sheet with you. You may prepare this as you wish. Feel free to work in groups to prepare a common crib sheet. Of course, each of you must have your own copy during the exam.
- 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, sqrt, trunc
- We will assume you know the following functions associated with the data of type `str`: len, +, * (by an integer), capitalize, replace, str, upper, and lower.
- Python syntax will be important on this exam. On exams 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 exam. Solutions to most of the problems will be posted on the course website on Saturday, February 28. These posted solutions will **not** include problems involving output from Python — try those out yourself!
- Please note that your solution to any question that requires you to write Python code will be at most a few (5-7) lines long, and may be much shorter. Focus on writing short programs, as it will save you time during the exam.
- The exam questions will be closely related to the practice problems below and problems from the homeworks, labs, and lecture exercises.
- Remember that 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 practice 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. The best approach is to type them into the Python shell and try new variations of code. This will also help when you are trying to track down syntax errors.

– Work through the sample questions, writing out solutions by hand (since you won’t have laptops at all for the exam!). Read solutions only as a last resort. Remember that reading a solution is much easier than actually writing one, and you are graded on the writing part.

– You are encouraged to work with other students as you study, but ask yourself if you understand the questions and material enough to solve problems on your own. Try to replicate a solution you worked on with someone a little while later without looking at any solutions.

– Review and re-do lecture exercises, lab, 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 a concept you did not understand.

– Use the Wing IDE / Python interpreter extensively to help you understand what is happening, but practice writing solutions out without using your laptop.

• Advance warnings:

– We will not answer questions from students in the middle of the exam unless there is a mistake in the question itself. In other words, do not ask what a certain line of code means or if your answer is correct.

Questions

1. Identify all Python syntax errors in the following code. In identifying each error, unlike what the Python interpreter does, assume the errors on the previous lines have been fixed.

```python
y = 5
x = y + 5*6 - # ill-formed expression
z = x+y
if z > x
    print 'z is smaller'
else:
    print 'z is larger'

print 'Sqrt of y is', sqrt(y)

print 'Abs of x is ', abs(x1)

s = str(z)
s = s + 6
s = s * 6
print "String s is now' s"
```

Solution: There are six syntax errors, as highlighted below.

```python
y = 5
x = y + 5*6 - # ill-formed expression
z = x+y
if z > x # missing : at end of line
    print 'z is smaller'
else:
    print 'z is larger'
```
2. What is the exact output of the following Python code? Show the output to the right of the code. Also, what are the local variables in the given code?

```python
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)
```

Solution to the second part: z is the only local variable.

3. Assuming we type the following directly into the Python interpreter, please fill in the exact output:

```python
>>> x = 5
>>> y = 27
>>> x / y

>>> y - x*3

>>> z = x + y - 7 * 3 % 4
>>> z

>>> x = 14 - 3 ** 3 + 2 % 6
>>> print x

>>> y += y * len(str(y))
```
>>> y

>>> 19.5 / 5 / 3

>>> 1 + 2 * 3 - 4 / 5 + 6 * 7 % 8

>>> import math
>>> x = 'the answer is ' + ( str( math.sqrt( 81 ) ) * 3 )
>>> print x

>>> s = '3' + '4' '5'
>>> s

>>> print s

>>> int(s) / 10

>>> float(s) / 10

>>> 'aBdcSf'.lower().capitalize() + 'BdkgE'.capitalize().lower()

Solution: Please test them for yourself.

4. Assuming we type the following directly into the Python interpreter, please fill in the exact output:

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

>>> z > len(s)

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

>>> len(u) == len(s)

Solution: Please test them for yourself.
5. 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.

Solution:

```python
x = int( raw_input( 'Enter a positive integer ==> ' ) )
if x <= 0:
    print 'Error: the number is not positive'
```

6. In the United States, a car’s fuel efficiency is measured in miles driven per gallon used. In the metric system it is liters used per 100 kilometers driven. Using the values 1.609 kilometers equals 1 mile and 1 gallon equals 3.785 liters, write a Python function that converts a fuel efficiency measure in miles per gallon to one in liters per 100 kilometers and returns the result.

Solution: The solution is spread across several lines to make it easier to write and read. This is not necessary for the exam, but it is good practice

```python
def convert( mpg ):
    km_per_mile = 1.609
    liters_per_gallon = 3.785
    miles_per_liter = mpg / liters_per_gallon
    liters_per_mile = 1 / miles_per_liter
    liters_per_km = liters_per_mile / km_per_mile
    return liters_per_km * 100
```

7. Write a function called `spam` that takes as input 4 numbers and returns a string listing all the numbers and their average as a float. For example:

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

Solution: There are many ways to do this.

```python
def spam( n1, n2, n3, n4 ):
    avg = ( n1 + n2 + n3 + n4 ) / 4.0
    s = "The average of %d, %d, %d, and %d is: %.2f" %( n1, n2, n3, n4, avg )
    return s
```

8. Write a Python program that reads from the user the names of two people (on two separate lines of input) and prints out the shorter of the two names. If the two names have equal length, then print out the first name.

Solution:

```python
name1 = raw_input( 'Please enter the first name ==> ' )
name2 = raw_input( 'Please enter the second name ==> ' )
if len( name1 ) <= len( name2 ):
    print name1
else:
    print name2
```
9. What is the output of the following program?

```python
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
```

Solution: Please test them for yourself.

10. 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

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

should output

```
   Bike
   Baseball
```

while the call

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

should output

```
   Bike
   Baseball
```

Solution:
def right_justify( s1, s2, n):
    n = max( len( s1 ), len( s2 ), n )
    line1 = ' ' * ( n - len( s1 ) ) + s1
    line2 = ' ' * ( n - len( s2 ) ) + s2
    print line1
    print line2

11. Write a Python function that takes as input two strings, string1, string2. It should print the first string
3 times on each each of three lines, separated by spaces. Next it should output a blank line and then it
should print the second string 6 times, again separated by spaces. For example:

chant("Let's go red!", "Fight!")

should output

Let's go red! Let's go red! Let's go red!
Let's go red! Let's go red! Let's go red!
Let's go red! Let's go red! Let's go red!

Fight! Fight! Fight! Fight! Fight! Fight!

Try to do this with code where the names string1 and string2 only appear in your function once each!

Solution:

def chant( s1, s2 ):
    line1 = ( s1 + ' ' ) * 3
    print line1, '\n', line1, '\n', line1, '\n'
    print ( s2 + ' ' ) * 6

12. Suppose Shirley Ann Jackson is tracking the number of friends she has on Facebook. Let the variable
week1 represent the number of friends she has at the end of the first week, let week2 represent the number
of friends she has at the end of the next week, and finally let week3 represent the number of friends she
has at the end of the third week. Write a single Python expression that calculates the average change,
including both up and down changes, of Shirley’s number of friends, and assigns the value to the variable
change. For example if she has 100 friends in week 1, 150 in week 2, and 125 in week 3, then the total
change is 75 and the average change is 75/2=37.5.

Solution:

change = ( abs( week1 - week2 ) + abs( week2 - week3 ) ) / 2.0

13. 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
repeat 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

Solution: In the solution below, the last line, the one that is commented out, has the advantage of not
printing an extra line at the end.
mystr = raw_input( 'Enter a string ==> ' )
num = int( raw_input( 'How many times ==> ' ) )
line = mystr * num + '\n'
print line * num
# print line * ( num - 1 ), mystr * num

14. Write a Python function that takes two strings as input and prints them together on one 35-character line, with the first string left-justified, the second string right-justified, and as many periods between the words as needed. For example, the function calls

    print_left_right( 'apple', 'banana')
    print_left_right( 'syntax error', 'semantic error')

should output

apple...........................................banana
syntax error..............................semantic error

You may assume that the lengths of the two strings passed as arguments together are less than 35 characters.

**Solution:**

def print_left_right( s1, s2 ):
    spaces = 35 - ( len( s1 ) + len( s2 ) )
    line = s1 + '.' * spaces + s2
    print line

15. Chris, a carpenter, takes $w_1$ weeks and $d_1$ days for job 1, $w_2$ weeks and $d_2$ days for job 2, and $w_3$ weeks and $d_3$ days for job 3. Assuming $w_1, d_1, w_2, d_2, w_3, d_3$ 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

    $w_1 = 1$
    $d_1 = 6$
    $w_2 = 2$
    $d_2 = 4$
    $w_3 = 2$
    $d_3 = 6$

The output should be

average is 2 weeks and 3 days

**Solution:** The trick is that averages must be calculated based on converting to total days.

    avg_days = ( ( w1 + w2 + w3 ) * 7 + d1 + d2 + d3 ) / 3
    print 'average is %d weeks and %d days' %( avg_days / 7, avg_days % 7 )

16. Write one line of Python code to calculate and output the average of the maximum and the minimum of four numbers, stored in the variables $u$, $x$, $y$ and $z$. For example, if the variables have the values 6, 5, 21, -3, then the code should output 9.0.

**Solution:**

    print 0.5 * ( max( u, x, y, z ) + min( u, x, y, z ) )
17. Write a Python function called `compare_date` that takes as arguments two lists of two integers each. Each list contains a month and a year, in that order. The function should return -1 if the first month and year are earlier than the second month and year, 0 if they are the same, and 1 if the first month and year are later than the second. Your code should work for any legal input for month and year. Example calls and expected output are shown below:

```python
>>> compare_date( [10,1995], [8,1995] )
1
>>> compare_date( [5,2010], [5,2010] )
0
>>> compare_date( [10,1993], [8,1998] )
-1
```

Solutions:

```python
def compare_date( x, y ):
    if x[1] > y[1]:
        return 1
    elif x[1] < y[1]:
        return -1
    elif x[0] > y[0]:
        return 1
    elif x[0] < y[0]:
        return -1
    else:
        return 0
```

Second version, combining the logic:

```python
def compare_date( x, y ):
        return 1
        return -1
    else:
        return 0
```

18. 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). Remember, there are no syntax errors here.
19. Assuming we type the following directly into the Python interpreter, please fill in the exactly output, making it clear what is your answer:

```python
>>> t = ' ab c-
>>> len(t)

>>> x = '32'
>>> x * len(x)

>>> int(x) / len(x)
```
>>> a = [ 54, 'abc', "67.5", 12 ]
>>> print a[1]

>>> s = a[2]
>>> print s[-1]

>>> names = [ 'graham', 'john', 'terry', 'terry', 'eric', 'michael' ]
>>> a = str( len( names[0] ) ) + '2'
>>> b

Solution: Please test them for yourself.

20. Assume v is a list containing numbers. Write Python code to find and print the highest two values in v. If the list contains only one number, print only that number. If the list is empty, print nothing. For example, if we assigned

v = [ 7, 3, 1, 5, 10, 6 ]

then the output of your code should be something like

7 10

If we are given that

v = [ 7 ]

then the output of your code should be

7

Solution:

v.sort()
if len( v ) == 1:
    print v[0]
elif len( v ) > 1:
    print v[-2], v[-1]

21. Clearly show the output from the following Python program:

    def thingamajig( a , b , c ):
      c = c + "x"
      if a < b:
        b = b-a
        c = c * ( b ** 2 )
else:
    a = a*a
    c = c * ( ( 3 / 2 ) * a )
print "msg:", c

thingamajig( 2 , 3 , "Hola" )
thingamajig( 2 , 2 , "Bonjour" )
thingamajig( 2 , 1 , "Ciao" )

Solution:

msg: Holax
msg: BonjourxBonjourxBonjourxBonjourx
msg: CiaoxCiaoxCiaoxCiaox

22. 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.

```python
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 + 2*base_area
    return surface_area
```

Solution:

```python
def surface_area(radius, height):  # Missing : at end of line
    base_area = pi * r ** 2  # Semantic error, exponent on the wrong term
    surface_area = lateral_area + 2 * base_area  # Syntax error, missing *
```

23. 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

```python
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.

Solution:

def longest_first( s1, s2 ):
    if len( s1 ) > len( s2 ):
        print '* ' + s1 + ' '* ( len( s2 ) - len( s1 ) ) + ' *
        print '* ' + s2 + ( '!' * ( len( s2 ) - len( s1 ) ) ) + ' *
    else:
        print '* ' + s2 + ' '* ( len( s2 ) - len( s1 ) ) + ' *
        print '* ' + s1 + ( '!' * ( len( s1 ) - len( s2 ) ) ) + ' *

s1 = raw_input( 'Enter string 1: ' )
s2 = raw_input( 'Enter string 2: ' )
longest_first( s1, s2 )

24. Write a Python program that requests two strings from the user using raw_input(). The program should then print each string on a separate line such that the second string appears just below the first string, overlapping by one letter, followed by a line of asterisks that underlines both strings.

For example, given the words python, and programming, your program should output:

    python
    programming
    ****************

Solution:

    w1 = raw_input()
    w2 = raw_input()

    print w1
    print ' ' * ( len( w1 ) - 1 ) + w2
    print '*' * ( len( w1 ) + len( w2 ) - 1 )

25. Given the following two functions:

    def bunny(bpop, fpop):
        ## Returns next year's bunny population given current bunny and fox populations
        bpop_next = (10*bpop)/(1+0.1*bpop) - 0.05*bpop*fpop
        return int(max(0,bpop_next))

    def fox(bpop, fpop):
        ## Returns next year's fox population given current bunny and fox populations
        fpop_next = 0.4 * fpop + 0.02 * fpop * bpop
        return int(max(0,fpop_next))

Assume these functions are already correctly defined in your program. Suppose the current bunny population is 200, and the current fox population is 10.

Write code using the above functions to compute the projected populations of bunnies and foxes in two years from now, and output one of the three strings below based on these numbers using an if statement.

    year 2016: more bunnies
    year 2016: more foxes
    year 2016: same number of bunnies and foxes
Solution:

b = bunny( 200, 10 )
f = fox( 200, 10 )
b2 = bunny( b, f )
f2 = fox( b, f )
if b2 > f2:
    print 'year 2016: more bunnies'
elif b2 < f2:
    print 'year 2016: more foxes:'
else:
    print 'year 2016: same number of bunnies and foxes'

26. Write a Python program that reads a sentence from the user using `raw_input()` and then does the following:

- Removes all instances of the string `and` if only they appear between two other words surrounded by spaces.
- Replaces all occurrences of the string `but` with `ot`.
- Prints the remaining string.
- Prints how many characters were removed from the original string.

For example, if the user types the following string:

```
butters and kenny but not andrew
```

your program should output:

```
otters kenny ot not andrew
Removed 5 characters
```

Solution:

```python
sentence = raw_input()
slen = len( sentence )
sentence = sentence.replace( ' and ',', ' )
sentence = sentence.replace( 'but', 'ot' )
print sentence
print 'Removed', slen - len( sentence ), 'characters'
```