CompSci 101: Test 1

Peter Lorensen
February 18, 2013

Name: _________________________________________________________________

NetID/Login: __________________________
Community Standard Acknowledgment (signature): ____________________________

<table>
<thead>
<tr>
<th>Problem</th>
<th>Value</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Problem 1</td>
<td>30 pts</td>
<td></td>
</tr>
<tr>
<td>Problem 2</td>
<td>20 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 3</td>
<td>20 pts.</td>
<td></td>
</tr>
<tr>
<td>Problem 4</td>
<td>20 pts.</td>
<td></td>
</tr>
<tr>
<td>TOTAL:</td>
<td>90 pts.</td>
<td></td>
</tr>
</tbody>
</table>

In writing code you do not need to worry about specifying the proper import statements. Don’t worry about getting function or method names exactly right. Focus on getting the arguments correct.

Assume that all libraries and packages we’ve discussed are imported in any code you write.
**PROBLEM 1 : (SMITH [CORONA—WESSON] (30 POINTS))**

**Part A (22 points)**

Each of the variables below has a *type* and a *value*. The type is one of: list, boolean, int, string, float. For example, consider the assignment to variable `x` below:

\[
\text{age} = \text{sum}([4,7,12])
\]

The type and value are shown in the first row of the table below. Fill in the other type and value entries based on the variable/expression in the first column.

<table>
<thead>
<tr>
<th>Variable/Expression</th>
<th>Type</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>age = sum([4,7,12])</code></td>
<td>int or integer</td>
<td>23</td>
</tr>
<tr>
<td><code>money = 25/3</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>a = 21 \% 6</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>e = sum(range(6))</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>rating = len(&quot;cockie&quot;) &lt; len(&quot;why&quot;)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>names = &quot;pete,jo,owen,hans&quot;.split(&quot;,&quot;)</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>pick = &quot;Center of pie&quot;[3]</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>g = &quot;Office&quot; + &quot;Hours&quot;</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>winner = &quot;repeteralism&quot;[2:7]</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>drink = 1.44*10</code></td>
<td></td>
<td></td>
</tr>
<tr>
<td><code>best = ['A', '+A', 'B', 'A', 'C'][:2]</code></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Part B (4 points)
Write a function to calculate the volume of a sphere given this formula: \((4/3) \times \pi \times r^3\)
Use math.pi or 3.14 in place of pi.

```
def sphereVolume( radius ):
    ""
    returns the volume of a sphere as a float with the given parameter radius
    ""
```

Part C (4 points)
Complete this function so that it prints what it is supposed to as shown.
Hint: string.lower() converts the string to all lower case. string.upper() converts the string to upper case.

<table>
<thead>
<tr>
<th>Call</th>
<th>Print value</th>
</tr>
</thead>
<tbody>
<tr>
<td>reverseCase(“dUKE dEVILS”)</td>
<td>Duke Devils</td>
</tr>
<tr>
<td>reverseCase(“Blue#2”)</td>
<td>bLUE#2</td>
</tr>
<tr>
<td>reverseCase(“GrEeN&gt; wOrLd”)</td>
<td>gReEn&gt; WoRlD</td>
</tr>
</tbody>
</table>

```
def reverseCase( word ):
    toBePrinted = ""
    for eachLetter in word:
```
**PROBLEM 2 (20 POINTS)**

**Part A (3)**

*A deficient number* is a number in which the sum of its factors is less than the given number. For example, the number 8 has factors (divisors) of 1, 2, 4, and 8. If you disregard 8 as a factor, then the sum $1 + 2 + 4 = 7$ and 7 is less than 8. Therefore, 8 is deficient.

Other deficient numbers are:

9 because $1 + 3 = 4 < 9$

Fill out the rest of this table on deficient numbers:

<table>
<thead>
<tr>
<th>Call</th>
<th>Return value</th>
<th>Sum of divisors</th>
</tr>
</thead>
<tbody>
<tr>
<td>is_deficient(4)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>is_deficient(5)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>is_deficient(6)</td>
<td>False</td>
<td>6</td>
</tr>
<tr>
<td>is_deficient(7)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>is_deficient(8)</td>
<td>True</td>
<td>7</td>
</tr>
<tr>
<td>is_deficient(9)</td>
<td>True</td>
<td>4</td>
</tr>
<tr>
<td>is_deficient(10)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>is_deficient(24)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>is_deficient(30)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Part B (10)

Write a function is_deficient(num) that returns True if its parameter is a deficient number and False otherwise.

def is_deficient( num ):
    ""
    return True if int parameter num is a deficient number and returns False otherwise.
    """
Part C (7)

Write a function deficient_count(start, end) that returns the number of deficient numbers between (and including) parameter start and end. You should call is_deficient and assume that it works correctly.

<table>
<thead>
<tr>
<th>Call</th>
<th>Return value</th>
</tr>
</thead>
<tbody>
<tr>
<td>deficient_count(1, 14)</td>
<td>12</td>
</tr>
<tr>
<td>deficient_count(2, 30)</td>
<td>22</td>
</tr>
<tr>
<td>deficient_count(80, 90)</td>
<td>7</td>
</tr>
<tr>
<td>deficient_count(20, 30)</td>
<td>7</td>
</tr>
</tbody>
</table>

```python
def deficient_count(start, end):
    ""
    Return the number of deficient numbers in the interval between start and end parameter (both inclusive).
    ""
```
PROBLEM 3 (20 POINTS)

Part A (4)

Write a function `cutStart( word, remove )` that cuts a number of characters the start of the `word` as specified in the parameter `remove`. It must return a string. Example:

```python
print cutStart( "Peter", 1 )
eter

def cutStart( word, remove):
    ""
    Return a string missing remove number of character. The character must be cut of the front of the word.
    ""
```

```python
print cutStart( "Peter", 1 )
eter
```
Part B (4)

Write a function `cutEnd( word, remove )` that cuts a number of characters the end of the word as specified in the parameter `remove`. Example:

```python
print cutEnd( "Peter", 2 )
Pet
```

def cutEnd( word, remove):
    '''
    Return a string missing remove number of character. The character must be cut of the end of the word.
    '''
Part C (10)

The infamous pirate GrayBeard has had a good season of raid’n pillage on the seven seas and his treasure chest is full. He needs to remove some pirates to increase each person’s share of the treasure. You must help him.

Here is GrayBeard’s list of his crew divided by crew section. Since the life of a pirate is short he doesn’t bother with writing the entire name, just the first letter. The newest arrivals in each section are last.

- Cannon crew: “ABDJES”
- Ship rushers: “KMMJDIWDGADS”
- Cutlass defenders: “LAK”
- Deep divers: “WZ”

The function below must return a list of crew sections were each sections has crewmembers removed according to arguments `blackNum` and `itsNewest` passed to the function. Example:

<table>
<thead>
<tr>
<th>Call</th>
<th>Return value</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>greedGetsMe([&quot;ASDK&quot;, &quot;WDG&quot;, &quot;RTE&quot;], 1, True)</code></td>
<td>[&quot;ASD&quot;, &quot;WD&quot;, &quot;RT&quot;]</td>
</tr>
<tr>
<td><code>greedGetsMe([&quot;DIUT&quot;, &quot;JDHU&quot;, &quot;KJUA&quot;, &quot;WER&quot;], 3, False)</code></td>
<td>[&quot;T&quot;, &quot;U&quot;, &quot;A&quot;, &quot;&quot;]</td>
</tr>
</tbody>
</table>

```
def greedGetsMe( crewList, blackNum, itsNewest ):
    "CrewList is a list of crew sections (not just the sections listed above, but any amount of sections). Ex. ["AHLC","LKPH","HJKLL"]"
    
    blackNum: (Int) number of crew men that must walk the plank (letters to remove).
    You may use the previous function from Part A and Part B and assume that they work correctly.
    
    itsNewest: if True then it is the newest crew in a section that must be removed first. Newest crew is written last in each section.
    if False, then it is the oldest crew in a section that must be removed first. Oldest crew is written first in each section
    
    (Please use next page for you code)
```
Part D (2)

What output does your function return with the following input:

    def greedGetsMe( ["MOREDV", "GOLDLP", "FORVK", "MEKS"], 2, True )

and this:

    def greedGetsMe( ["KASTVMU", "TKSLATI", "LTOLZNY", "LOWBS"], 5, False )
PROBLEM 4 (20 POINTS)

Part A (6)
Write a function that checks to see if an e-mail address is correct.

    def email_isCorrect( suspect ):
        "Returns True if the parameter suspect follows these rules:
        1. contains one and only one @
        2. contains no spaces
        Return False otherwise.
        "

Part B (7)

Write a function that check to see if a suggested password is strong enough. You may use the function email_isCorrect(xx) written in Part A and assume that I works correctly.

The password is OK if it conforms to these rules:

1. Must be longer that 5 characters. If not return 1
2. Must contain at least 1 capital letter. If not return 2
5. Must contain at least one number. If not return 5

def pass_isOK( passAttempt ):
    """Return 0 if passAttempt conforms to the above written rules. If not the function returns the number as shown above. If more than one rule is violated the function must add the numbers together and return the sum. Thus if both rule 1 and 5 is violated the function must return 6."""
Part C (7 points)

Write a function that accepts an e-mail and password. The function must return the following:

1. If both e-mail and password is OK return “OK”
2. If the e-mail is incorrect return:
   “Email error”
   Use the email_isCorrect( suspect ) function above and assume it works properly.
3. Only if the e-mail is correct, then use the pass_isOK( passAttempt ) function above (assume that it works properly). If the password is wrong, then return this message:
   “E-mail OK, password is rejected because XXXXX” where the XXXXX is a short explanation of what is wrong with the password. (i.e. is it missing capital letters, too short or missing digits or both!)

   def accountCreation( mail, pass ): 
(Extra page for your code)