COMP10001 Foundations of Computing
Semester 2, 2016

Tutorial Questions: Week 8

1. Convert the following into a list comprehension:

```
consonants = []
for char in "aardvark":
    if char not in "aeiou":
        consonants.append(char)
```

A:

```
consonants = [char for char in "aardvark" if char not in "aeiou"]
```

2. Evaluate the following list comprehensions:

   (a) `[float(i) for i in range(0,4)]`

   A: `[0.0, 1.0, 2.0, 3.0]`

   (b) `"".join([letter.upper() for letter in "hello"])`

   A: `'HELLO'`

   (c) `[i**2 for i in range(5) if i % 2 == 0]`

   A: `[0, 4, 16]`

3. Evaluate the following expressions, and provide the output in each case

   (a) `"{}{}{}".format("swedish"[-2:], "beet"[1:3], "shampoo"[:2])`

   A:

   ```
   'sheesh'
   ```

   (b) `list({1: "practice", 2: "practice"}.values())`

   A:

   ```
   ['practice', 'practice']
   ```

   (c) `"`Twas brillig, and the slithy toves".split()[1][:5]`

   A:

   ```
   'brill'
   ```

   (d) `bool("another one" and "a bunch of em" and not "lunch")`

   A:

   ```
   False
   ```

4. The following code is meant to print out the list of only the even numbers contained in `nums` (i.e. it should print `[6, 2]` for the value of `nums` provided below). Provide code to insert into each of the numbered boxes to complete the code. Note that your code should run at the indentation level indicated for each box.

```
def purify(numlist):
    [ 1 ]
    for num in numlist:
        if [ 2 ]
            [ 3 ]
        [ 4 ]

nums = [1,3,6,2,7]
print([ 5 ])
```

A:

```
def purify(numlist):
    retlist = []
    for num in numlist:
        if not num % 2:
            retlist.append(num)
    return retlist

nums = [1,3,6,2,7]
print(purify(nums))
```

5. The following code is meant to print out the number of *distinct* words in a given file (`jabberwocky.txt` in the case of the provided code). Provide code to insert into each of the numbered boxes to complete the code. Note that your code should run at the indentation level indicated for each box.

```
        1

def     2
    return word.lower().rstrip(",.:;!?-'\"")


def count_words(doc):
    words = defaultdict(int)
        3
        word = strip_punct(word)
        if word:
            4
        5

print(count_words("jabberwocky.txt"))
```

A:

```
from collections import defaultdict

def strip_punct(word):
    return word.lower().rstrip(",.:;!?-'\"")


def count_words(doc):
    words = defaultdict(int)
    for word in open(doc).read().split():
        word = strip_punct(word)
        if word:
            words[word] += 1
    return len(words)

print(count_words("http://tinyurl.com/jabberwocky-txt"))
```

6. The following function is intended to take a single string argument, and return a Boolean value evaluating whether the argument passed to the function is an "isogram" or not — an isogram is a word where each letter occurs the same number of times (e.g. `"unforgivable"` is an isogram, as each letter occurs exactly once in the word). The lines of the function are out of order, however, and all indentation has been lost. Put the line numbers in the correct order and introduce appropriate indentation (indent the line numbers to show how the corresponding lines would be indented in your code).

```
1 for ch in word:
2 if count != counts[0]:
3 stock[ch] += 1
4 def isogram(word):
5 counts = stock.values()
6 return False
```

```
return True
stock = defaultdict(int)
from collections import defaultdict
for count in counts[1:]:
```

A:
```
9
4
    8
    1
        3
    5
    10
        2
            6
    7
```

That is, the code is:

```
from collections import defaultdict
def isogram(word):
    stock = defaultdict(int)
    for ch in word:
        stock[ch] += 1
    counts = stock.values()
    for count in counts[1:]:
        if count != counts[0]:
            return False
    return True
```

7. The following function is intended to take a single string argument (a name, in the form of a string), and return a Boolean value evaluating whether the name passed to the function is a "cool dude name" or not — a cool dude name is one which contains at least 4 vowels in total AND at least 2 vowels in one of the component names (e.g. `"Timothy Baldwin"` is a cool dude name, as it contains 4 vowels in total, and `"Timothy"` and `"Baldwin"` each contain 2 vowels). The lines of the function are out of order, however, and all indentation has been lost. Put the line numbers in the correct order and introduce appropriate indentation (indent the line numbers to show how the corresponding lines would be indented in your code).

```
1  if vowelcount(name) >= 2:
2  return False
3  return False
4  for name in fullname.split(" "):
5  for letter in string:
6  vowels += 1
7  if letter in 'aeiou':
8  def vowelcount(string):
9  return True
10 def cool_dude(fullname):
11 vowels = 0
12 return vowels
13 if vowelcount(fullname) < 4:
```

A: `10`

```
        8
        11
        5
            7
                6
            12
        13
            2
        4
            1
                9
            3
        7
```

Note that lines 2 and 3 are interchangeable. The code is:

```
def cool_dude(fullname):
    def vowelcount(string):
        vowels = 0
        for letter in string:
            if letter in 'aeiou':
                vowels += 1
        return vowels
    if vowelcount(fullname) < 4:
        return False
    for name in fullname.split(" "):
        if vowelcount(name) >= 2:
            return True
    return False
```

8. Rewrite the following code, replacing the `for` loop with a `while` loop, but otherwise preserving the original functionality of the code:

```
MIN_WORD_LEN = 6
long_words = 0
for word in text.split():
    if len(word) >= MIN_WORD_LEN:
        long_words += 1
```

A:

```
MIN_WORD_LEN = 6
long_words = 0
words = text.split():
while words:
    word = words.pop()
    if len(word) >= MIN_WORD_LEN:
        long_words += 1
```

9. Rewrite the following code, replacing the `while` loop with a `for` loop, but otherwise preserving the original functionality of the code:

```
def prime(num):
    from math import sqrt
    factor = 2
    max_factor = int(sqrt(num))
    while factor < max_factor:
        if not(num % factor):
            return False
        factor += 1
    return True
```

A:

```
def prime(num):
    from math import sqrt
    for factor in range(2, int(sqrt(num)) + 1):
        found_factor = not(num % factor)
        if not(num % factor):
            return False
    return True
```