COMP10001 Foundations of Computing
Semester 2, 2016

Tutorial Questions: Week 6

1. For the following list comprehensions: (a) evaluate the expression; and (b) write the corresponding Python code which would generate an object of the same content, without using a list comprehension (using `for` statements, etc.):

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

   A:

   (a)
   ```
   [0.0, 1.0, 2.0, 3.0]
   ```

   ```
   lst = []
   for i in range(0,4):
       lst.append(float(i))
   ```

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

   A:

   (a)
   ```
   'HELLO'
   ```

   ```
   lst = []
   for letter in "hello"]:
       lst.append(letter.upper())
   "".join(lst)
   ```

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

   A:

   (a)
   ```
   [0, 4, 16]
   ```

   (b)
   ```
   lst = []
   for i in range(5):
       if i \% 2 == 0:
           lst.append(lst.append(i**2))
   ```

2. What is the output of the following code:

```
def foo(x, y):
    global a
    a = 42
    x,y = y,x
    b = 17
    c = 100
    print a,b,x,y

a,b,x,y = 1,2,3,4
foo(17,4)
print a,b,x,y
```

A:

```
42 17 4 17
42 2 3 4
```

3. Compare the following two functions with respect to the given function calls, and answer the following questions. (1) Are there inputs for which the output of the two functions will differ? (2) Are there instances where one version should be preferred over the other?

```
def noletter1(wordlist, letter='z'):
    for word in wordlist:
        if letter in word:
            return(False)
    return(True)

def noletter2(wordlist, letter='z'):
    noz = True
    for word in wordlist:
        if letter in word:
            noz = False
    return(noz)

wordlist = ['zizzer'] + ['aadvark'] * 10000000
print(noletter1(wordlist))
print(noletter2(wordlist))
```

A: *The two functions are identical in output over all outputs BUT the first (*`noletter1`*) uses lazy evaluation and is potentially much faster, as it returns the moment it detects a letter which violates the test, rather than continuing to test other values (which can never change the final value of the test). As such, the first should be preferred, as it is potentially much more efficient.*

4. Write a function `hypotenuse(a, b)` that returns (as a `float`) the length of the hypotenuse of a right-angled triangle with side lengths `a` and `b` (both positive numbers). Use Pythagoras' theorem $a^2+b^2=c^2$, and the implementation of square root in the `math` library (`math.sqrt`).

A:

```
from math import sqrt

def hypotenuse(a, b):
    return sqrt(a**2 + b**2)
```

5. Using a `defaultdict`, write a function `hapax(text)` which returns an orthographically sorted list of all words in `text` that occur exactly once. Note that a "word" is defined simply to be a string that is surrounded by white space.

A:

```
from collections import defaultdict
def hapax(text):
    word_dict = defaultdict(int)
    for word in text.split():
        word_dict[word] += 1
    return sorted([word for word in word_dict if word_dict[word] == 1])
```

Optional Extension Questions for Self-study

1. A palindromic number is a positive integer that reads the same both ways. The largest palindromic number made from the product of two 2-digit integers is $9009 = 91 \times 99$.

   Write a function `palnum(num)` that returns the largest palindromic number made from the product of two `num`-digit numbers.

A:

```python
def palnum(num):
    max = 0
    for i in range(10**num,10**(num+1)):
        for j in range(10**num,10**(num+1)):
            product = i * j
            if str(product) == str(product)[::-1] and product > max:
                max = product
    return max
```