COMP10001 Foundations of Computing
Semester 1, 2019

Tutorial Questions: Week 7

— VERSION: 1533, DATE: APRIL 16, 2019 —

Discussion

1. What do we mean by "mutability"? Which data types are mutable out of what we've seen?
2. What is the difference between `sorted()` and `.sort()` when applied to a list? What does it mean to edit an object "in-place"?
3. What is a "namespace"? What do we mean by "local" and "global" namespace?

   Now try Exercises 1 & 2
4. Why is it important to write comments for the code we write? Wouldn't it save time, storage space and processing time to write code without comments?
5. How should we choose variable names? How do good variable names add to the readability of code?
6. What are "magic numbers"? How do we write code without them and how should we store global constants?
7. What is a "docstring"? What is its purpose?

   Now try Exercises 3 & 4

Exercises

1. What is the output of this code? Why?

```
def mutate(x, y):
    x = x + "The End"
    y.append("The End")
    print(x)
    print(y)

my_str = "It was a dark and stormy night."
my_list = my_str.split()
my_list_2 = my_list
mutate(my_str, my_list_2)
print(my_str)
print(my_list_2)
```

2. What is the output of the following code? Classify the variables by which namespace they belong in.

```
def foo(x, y):
    a = 42
    x, y = y, x
    print(a, b, x, y)

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

3. Consider the following programs. What are the problematic aspects of their variable names and use of magic numbers? What improvements would you make to improve readability?

   (a)
   ```
   a = float(input("Enter␣days:␣"))
   b = a * 24
   c = b * 60
   d = c * 60
   print("There␣are", b, "hours,", c, "minutes", d, "seconds␣in", a, "days")
   ```

   (b)
   ```
   word = input("Enter␣text:␣")
   words = 0
   vowels = 0
   word_2 = word.split()
   for word_3 in word_2:
       words += 1
       for word_4 in word_3:
           word_5 = word_4.lower()
           if word_5 in "aeiou":
               vowels += 1
   if vowels/words > 0.4:
       print("Above␣threshold")
   ```

4. Fill in the blanks with comments and a docstring for the following function, which finds the average frequency of letters over frequency `n` in `text`. There's no definite right or wrong answer here, try and develop your style.

   ```
   def average_freq_over(text, n):
       """ ... """
       freqs = {}

       # ...
       for letter in text:
           if letter in freqs:
               freqs[letter] += 1
           else:
               freqs[letter] = 1

       # ...
       over = []
       total = 0
       for key in freqs:
           if freqs[key] > n:
               over.append(key)
               total += freqs[key]

       # ...
       average = total / len(over)
       return average
   ```

Problems

1. Write a function which takes a string containing an FM radio frequency and returns whether it is a valid frequency. A valid frequency is within the range 88.0-108.0 inclusive with 0.1 increments, meaning it must have only one decimal place.

2. Write a function which takes a string and returns a 2-tuple containing the most common letter in the string and its frequency. In the case of a tie, it should return the letter which occurs first in the text.