COMP10001 Foundations of Computing
Semester 1, 2021

Tutorial Questions: Week 7

— VERSION: 1475, DATE: APRIL 19, 2021 —

Discussion

1. 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?
2. How should we choose variable names? How do good variable names add to the readability of code?
3. What are "magic numbers"? How do we write code without them by using global constants?
4. What is a "docstring"? What is its purpose?

   Now try Exercises 1 & 2
5. What is a "bug"? What are some debugging strategies which we can use when we find an error?
6. What are the three types of errors we've learned about and what do they mean?
7. How can we use testing to find bugs or confirm our code runs properly? What are some strategies we can adopt to write comprehensive test cases for our code?

   Now try Exercises 3 & 4

Exercises

1. 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")
```

2. Fill in the blanks with comments and a docstring for the following function, which finds the most popular animals by counting ballots. An example for `ballots` is `['dog', 'python', 'cat', 'python', 'dog']`, in which case the function returns `['dog', 'python']`. There's no definite right or wrong answer here, try and develop your style.

```
def favourite_animal(ballots):
    """ ... """
    tally = {}

    # ...
    for animal in ballots:
        if animal in tally:
            tally[animal] += 1
        else:
            tally[animal] = 1

    # ...
    most_votes = max(tally.values())
    favourites = []
    for animal, votes in tally.items():
        if votes == most_votes:
            favourites.append(animal)

    return favourites
```

3. Find the errors in the following programs, classifying them as (a) syntax, (b) runtime or (c) logic errors. Fix them with a correct line of code.

(a)
```
def disemvowel(text):
    """ Returns string `text` with all vowels removed """
    vowels = ('a', 'e', 'i', 'o', 'u')
    answer = text[0]
    for char in text:
        if char is not in vowels:
            answer = char + answer
    print(answer)
```

(b)
```
def big-ratio(nums, n):
    """ Calculates and returns the ratio of numbers
    in list `nums` which are larger than `n` """
    n = 0
    greater_n = 0
    for number in nums:
        if number > n:
            greater_n += 1
            total += 1
    return greater_n / total

nums = [4, 5, 6]
    low = 4
print(f"{100*big_ratio(nums, low)}% of numbers are greater than {low}")
```

4. Imagine we are given some code which we must test to ensure it works as intended. The code takes a student's ID and a year as input, fetches their records and calculates their average mark for that academic year. Describe three test cases you could construct to test different aspects of the code's functionality.

Problems

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

2. 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.