COMP10001 Foundations of Computing
Semester 2, 2022

Tutorial Questions: Week 7

— VERSION: 1477, DATE: SEPTEMBER 5, 2022 —

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. What is a docstring? What is its purpose?

   Now try Exercises 1
3. How should we choose variable names? How do good variable names add to the readability of code?
4. What are magic numbers? How do we write code without them by using global constants?

   Now try Exercise 2
5. What do we mean by mutability? Which data types are mutable out of those we've seen?
6. What is a namespace?
7. What do we mean by local and global namespace? What is scope?

   Now try Exercises 3 & 4
8. When is it useful to return early? How can it make our code safer and more efficient?
9. What are helper functions? How can they make our code more readable and reusable?

   Now try Exercise 5

Exercises

1. 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', 'pig', 'cat', 'pig', 'dog']`, in which case the function returns `['dog', 'pig']`. There's no definite right or wrong answer here, try to develop your own 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
```

2. 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: ")
x = 0
vowels = 0
word_2 = word.split()
for word_3 in word_2:
    x += 1
    for word_4 in word_3:
        if word_4.lower() in "aeiou":
            vowels += 1
if vowels/x > 0.4:
    print("Above threshold")
```

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

```
def mystery(x):
    x.append(5)
    x[0] += 1
    print("mid-mystery:", x)

my_list = [1, 2]
print(my_list)
mystery(my_list)
print(my_list)
mystery(my_list.copy())
print(my_list)
```

4. 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)
```

5. Compare the two functions below. Are they equivalent? Why would we prefer one over the other?

```
def noletter_1(words, letter='z'):
    for word in words:
        if letter in word:
            return False
    return True

def noletter_2(words, letter='z'):
    no_z = True
    for word in words:
        if letter in word:
            no_z = False
    return no_z

wordlist = ['zizzer'] + ['aadvark'] * 1_000_000
print(noletter_1(wordlist))
print(noletter_2(wordlist))
```

Problems

1. Write a function which takes a list of words and checks whether any of those words are palindromes (spelled the same way backwards as forwards, like "hannah"). It should return True if there are any palindromes and False if there are none. Use a timely return to save some time!
   `any_palindrome(['hannah', 'sven', 'mei'])` should return `True`.

2. Write a function which takes a list of lists of integers and returns a sorted list containing the unique numbers. `unique_2d_list([[1, 5, 3], [2], [5, 1, 2]])` should return `[1, 2, 3, 5]`.

3. 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. `most_common_char('hi there')` should return `('h', 2)`.