COMP10001 Foundations of Computing
Semester 1, 2021

Tutorial Questions: Week 8

— VERSION: 1477, DATE: APRIL 26, 2021 —

Discussion

1. When is it useful to "return early"? How can it make our code safer and more efficient?
2. What are helper functions? How can they make our code more readable and reusable?

   Now try Exercises 1
3. Revise debugging. What are some debugging strategies which we can use when we find an error?
4. 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 Exercise 2

Exercises

1. 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'] * 1000000
print(noletter_1(wordlist))
print(noletter_2(wordlist))
```

2. For each of the following problems:

   - Read the problem specification.
   - Write down three test cases that could be useful for function verification or finding bugs.
   - Debug the associated code snippet to solve the problem [Technology allowed].

   (a) Write a function that takes a list of integers and removes the negative integers from the list.

```
def remove_negative(nums):
    for num in nums:
        if num < 0:
            nums.remove(num)
```

(b) Write a function that takes a string of words and returns a string where each word contains hyphens between each character, e.g. `"this␣string␣here"` becomes `"t-h-i-s␣s-t-r-i-n-g␣h-e-r-e"`.

```
def hyphenate_words(string):
    list = []
    for word in string.split():
        word = list(word
        hyphenated = word.join('-')
        list = list.append(hyphenated)
    return words.join('␣')
```

(c) Write a function to find the minimum of a mathematical function using a scanning search. The search should be conducted from -1 to 1 inclusive, and try all function values inbetween with a stepsize of 0.1. This function should return the minimum function value for any arbitrary function (that is finite between -1 and 1).

```
start = -1
end = 1
stepsize = 0.1

def f(x):
    return -x - 2

def g(x):
    return x**2 - 1

def find_min(function):
    minimum = 0
    while start < end:
        if f(start) < minimum:
            minimum = f(start)
            start += stepsize
    return minimum

print(find_min(f), find_min(g))
```

These debugging problems and more can be found on Grok between Worksheet 18 and the Practice Project. Many of our talented teaching staff have recorded themselves thinking aloud as they solve these very problems, demonstrating different individual styles of debugging. These recordings can be found along with your regular lectures in the LMS Lecture Capture.

Problems

1. Write a program which asks the user to enter a series of words and then checks whether any of those words are palindromes (spelled the same way backwards as forwards, like “hannah”). You should print True if there are any palindromes and False if there are none. Use lazy evaluation to save some time!

2. Write a function which takes a lowercase string as input and prints the frequency of each vowel in the string using a dictionary.

3. Write a function which takes two lists of integers and returns the average of the numbers which they both have in common.

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