Programming Lab II
Handout 2

Antonio de la Vega de León, Bijun Zhang, Thomas Blaschke, Dr. Martin Vogt
martin.vogt@bit.uni-bonn.de

April 26, 2016

Assignments for classes on April 26, April 28

You can either write your solutions in notebooks or as normal Python scripts (`*.py`). Exercises marked with a (*) are a bit more difficult.

Exercises

Ex.0 *Reading*

(*Note:* It is expected that reading of the material is done outside class hours and the time during class is used for doing practical assignments and asking questions to the tutors.)

The material in this handout is covered in chapters 5 to 9 of the book *Think Python*.

Ex.1 *Conditionals. (Ex. 5.3 of Think Python)*

If you are given three sticks, you may or may not be able to arrange them in a triangle. For example, if one of the sticks is 12 inches long and the other two are one inch long, it is clear that you will not be able to get the short sticks to meet in the middle. For any three lengths, there is a simple test to see if it is possible to form a triangle:

> If any of the three lengths is greater than the sum of the other two, then you cannot form a triangle. Otherwise, you can. (If the sum of two lengths equals the third, they form what is called a "degenerate" triangle.)

(a) Write a function named `is_triangle` that takes three integers as arguments, and that prints either "Yes" or "No," depending on whether you can or cannot form a triangle from sticks with the given lengths.

(b) Write a function that prompts the user to input three stick lengths, converts them to integers, and uses `is_triangle` to check whether sticks with the given lengths can form a triangle.

Ex.2 *Ancient algorithms (Ex. 6.5 of Think Python)*

The greatest common divisor (GCD) of a and b is the largest number that divides both of them with no remainder.

One way to find the GCD of two numbers is Euclid's algorithm, which is based on the observation that if r is the remainder when a is divided by b, then $gcd(a, b) = gcd(b, r)$. As a base case, we can use $gcd(a, 0) = a$.

Write a function called `gcd` that takes parameters `a` and `b` and returns their greatest common divisor. If you need help, see `http://en.wikipedia.org/wiki/Euclidean_algorithm`. (`%` is the remainder operator, i.e. `a%b` gives the remainder of `a` divided by `b`).

Ex.3 *Srinivasa Ramanujan calculates π (Ex. 7.3 of Think Python)*
The mathematician Srinivasa Ramanujan found an infinite series that can be used to generate a numerical approximation of π:

$$\frac{1}{\pi} = \frac{2\sqrt{2}}{9801} \sum_{k=0}^{\infty} \frac{(4k)!(1103 + 26390k)}{(k!)^4 396^{4k}}$$

Write a function called `estimate_pi` that uses this formula to compute and return an estimate of π. It should use a `while` loop to compute terms of the summation until the last term is smaller than `1e-15` (which is Python notation for 10^{-15}). You can check the result by comparing it to `math.pi`.

Ex.4 *Happy numbers*
Happy numbers are defined by the following process: Start with a positive number. Replace the number with the sum of the squares of its digits and repeat until you reach the number 1 or the process enters a loop not involving the number 1. A number that reaches 1 is called a happy number all other numbers are unhappy.

(a) Write a function `isHappy(n)` that checks whether a number is happy or unhappy. It should return `true` if the the number is happy and `false` otherwise.

Hint: If a number is unhappy it will enter a loop involving the number 4. Thus you can repeat the process until the number reaches either 1 (happy) or 4 (unhappy).

Hint: You will somehow need to extract the individual digits of a number. Section 8.7 (pg. 75) of Think Python explains how to iterate over the individual characters of a string. A number can be represented as a string using the `str` function and vice versa a string consisting of digits can be converted back to an integer using the `int` function.

(b) Find all happy numbers from 1 to 100.

(c) (*Optional*) Solve the problem in different ways using a) `while`-loops and b) recursion.

(d) (*Optional*) Modify the problem by taking the sum of cubes instead of the sum of squares. Find all the loops that can occur and all numbers that are equal to the sum of the cubes of their digits. A number is called cube-happy if its iteration ends in a number that is identical to the sum of the cubes of its digits. Find all cube-happy numbers from 1 to 1000.

Ex.5 *Words and predicate logic (Ex. 9.3 to 9.5 of Think Python)*
The folder `\\bitsmb\groups\workshops\proglab2\` contains the file `words.txt` from chapter 9. As a preliminary exercise do exercise 9.1. of Think Python, i.e. read the word list line by line (one word per line) and print all words with more than 20 characters.

(a) Write a function named `avoids` that takes a word and a string of forbidden letters, and that returns `True` if the word doesn't use any of the forbidden letters.

Modify your program to prompt the user to enter a string of forbidden letters and then print the number of words in the file `words.txt` that don't contain any of them.

(b) Write a function named `uses_only` that takes a word and a string of letters, and that returns `True` if the word contains only letters in the list. Can you make a sentence using only the letters `acefhlo`? Other than "Hoe alfalfa?". To this end, write a program that prints all the words from `words.txt` that only made up exclusively of the letters `acefhlo`.

(c) Write a function named `uses_all` that takes a word and a string of required letters, and that returns `True` if the word uses all the required letters at least once. How many words are there in `words.txt` that use all the vowels `aeiou`? How about `aeiouy`?

(d) (*Optional*) Can you write `uses_all` in such a way that it calls `uses_only`?

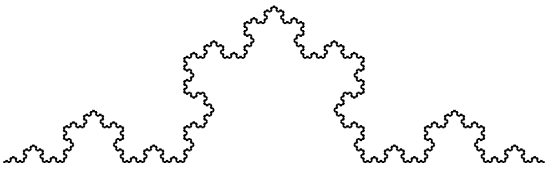

Figure 1: A Koch curve.

Ex.6 *Ordered Words (Ex. 9.6 of Think Python)*
Write a function called `is_abecedarian` that returns `True` if the letters in a word appear in alphabetical order (double letters are ok). How many abecedarian words are there in `words.txt` consisting of 4 or more letters?

Ex.7 *(*) Recursion (or pretty things computer scientists get exited about) (Ex. 5.5 & 5.6 of Think Python)*
The following exercise use the swampy package (see Chapter 4 of Think Python):

(a) Read the following function and see if you can figure out what it does. Then run it (see the examples in Chapter 4 of Think Python).

```
def draw(t, length, n):
    if n == 0:
        return
    angle = 50
    t.fd(length*n)
    t.lt(angle)
    draw(t, length, n-1)
    t.rt(2*angle)
    draw(t, length, n-1)
    t.lt(angle)
    t.bk(length*n)
```

(b) The Koch curve is a fractal that looks something like Figure 1. To draw a Koch curve with length x, all you have to do is

i. Draw a Koch curve with length $x/3$.
ii. Turn left 60 degrees.
iii. Draw a Koch curve with length $x/3$.
iv. Turn right 120 degrees.
v. Draw a Koch curve with length $x/3$.
vi. Turn left 60 degrees.
vii. Draw a Koch curve with length $x/3$.

The exception is if x is less than 5: in that case, you can just draw a straight line with length x.

i. Write a function called `koch` that takes a turtle and a length as parameters, and that uses the turtle to draw a Koch curve with the given length.
ii. Write a function called `snowflake` that draws three Koch curves to make the outline of a snowflake.

(c) (*Optional*) In a similar vein try to write a function that draws a dragon-curve. For a definition see: `http://en.wikipedia.org/wiki/Dragon_curve`.

Ex.8 *(Optional) What's the word? (Ex. 9.7 of Think Python)*
Write a program to find a word matching the following description in `words.txt`.

> Give me a word with three consecutive double letters. I'll give you a couple of words that almost qualify, but don't. For example, the word committee, c-o-m-m-i-t-t-e-e. It would be great except for the 'i' that sneaks in there. Or Mississippi: M-i-s-s-i-s-s-i-p-p-i. If you could take out those i's it would work. But there is a word that has three consecutive pairs of letters and to the best of my knowledge this may be the only word. Of course there are probably 500 more but I can only think of one. What is the word?