General Information & Communication Technology I (350101) Fall 2015

Michael Kohlhase
Jacobs University Bremen
http://kwarc.info/kohlhase

September 9. 2015

Abstract

This document accompanies the python tutorial in GenICT. It contains a sequence of simple (but increasingly difficult) problems designed to practice the art of recursive programming.

The problems in this document are intended for self-study, they are supplied with solutions(on a separate document at http://kwarc.info/teaching/GenICT/py-tutorial-with-solutions.pdf).

As most students have never programmed python (or programmed at all), most students only manage to solve the first five. This is to be expected, and sufficient, since the purpose of the tutorial is to get students started at all and jointly remove the first roadblocks, so that they can continue alone (or in groups) after that.

The problems from the first three assignments should be doable after the first two lectures on python, the later problems can be tackled as the lecture progresses.

Practice Problems 1: Python Basics

Problem 1.1 (Maximum)
Define a function `mymax` that takes two numbers as arguments and returns the larger of them. Use the `if-then-else` construct available in python.

Note: It is true that Python has the `max` function built in, but writing it yourself is nevertheless a good exercise.

Problem 1.2 (Sum to 10)
Write a function `makes10` that takes two integer arguments and returns `True`, iff their sum is 10.

Problem 1.3 (Positive/Negative)
Write a function `posneg` that takes two arguments and returns `True`, iff one is negative and one is positive.

Problem 1.4 (Squares)
Write a python program that prints all the square numbers from 1 to 10.

Problem 1.5 (Printing a Square of Stars)
Write a function `printSquare` that takes an integer n as argument and prints a square with $n \times n$ stars. For instance `printSquare(6)` would yield

```
******
******
******
******
******
******
```

Problem 1.6 (Squares to file)
Write a program that prints all the square numbers from 1 to 10 to a file named `squares.txt`.

Problem 1.7 (Membership)
Write a function `member` that takes a value (i.e. a number, string, etc.) x and a list l of values and returns `True` if x is a member of l and `False` otherwise.

Note: For example `member(1,[1,2,3])` returns `True`.

Note: Note that this is exactly what the in operator does, but for the sake of the exercise you should pretend python lacks this operator. A `for/in` loop is OK though.

Problem 1.8 (Guessing Numbers)
Write a program where you guess a number. The program should draw an integer number (use `n = random.randint(1, 100)`) and then you should guess the number by inputting a number on the keyboard. The program should tell you then whether the your guessed number is smaller or bigger than the hidden number, and let you try again until you have sucessfully guessed the number.

Note: Keep in mind that you need to import the `random` module.

Problem 1.9 (99 Bottles)
"99 Bottles of Beer" is a traditional song in the United States and Canada. It is popular to sing on long trips, as it has a very repetitive format which is easy to memorize, and can take a long time to sing. The song's simple lyrics are as follows:

> 99 bottles of beer on the wall, 99 bottles of beer.
> Take one down, pass it around, 98 bottles of beer on the wall.

The same verse is repeated, each time with one fewer bottle. The song is completed when the singer or singers reach zero.

Your task here is write a python program capable of generating all the verses of the song.

Problem 1.10 (Recognizing Palindromes)
Define a function `palindrome` that recognizes palindromes (i.e. words that look the same written backwards). For example, `palindrome("radar")` should return `True`.