Dev I and Dev II - kantelenplan

Dr. Giuseppe Maggiore, Tony Busker

July 6, 2015

1 Introduction

In this document we give an overview of the activities of the students during the various lectures. The lectures cover a mixture of theory and practice with an interactive format:

- some theory will be explained in a burst of roughly half an hour;
- one or more exercises (in small groups) will be given;
- the solution to the exercises will then be presented by one of the groups
- the lecturer and the rest of the class will assist the student called.

2 Lectures

In this section we describe each lecture topics and activities.

2.1 Lecture 1 - logical model of computation

The first lecture covers basic concepts of computation from a logical standpoint.

Topics

- Following a path (example: *take three steps forward, turn left, ...*);
- Following a path with state (example: *read N from the whiteboard, take N steps forward, ...*);
- Following a path with wrongly typed state (example: *take Monday steps forward, ...*);
- Following a path with state and conditionals (example: *take N steps forward if the lecturer is smiling, ...*).

Activities

- Let students follow instructions;
- Introduce elements of state and let students follow instructions with state (*take N/4 steps forward; N is your age*);
- Introduce elements of writable state and let students follow instructions with writable state (*take N/4 steps forward; N is written under the yellow sticker*);

- Introduce elements of decision-making and let students follow instructions with state and decision making (*if the sun is shining, then take N/4 steps forward; otherwise, go sit down*);
- Introduce elements of iteration and let students follow instructions with state, decision making, and iteration (*divide the students in teams, and let run some script battling for the toy farm*).

2.2 Lecture 2 - concrete model of computation

The second lecture covers basic concepts of computation from a logical standpoint.

Topics

- CPU and memory;
- a basic overview of the various things that an imperative language can do, independent of syntax;
- introduction to semantics and post-conditions.

Activities

- Formalize the concept of instructions seen in the previous lecture by rewriting the scripts;
- Formalize the concept of state and mutation seen in the previous lecture by rewriting the scripts;
- Formalize the concept of decision-making seen in the previous lecture by rewriting the scripts;
- Formalize the concept of iteration seen in the previous lecture by rewriting the scripts.

2.3 Lecture 3 - Hello Python! and variables

The third lecture covers variables and an introduction to Python.

Topics

- brief history of programming languages;
- brief introduction to Python: what it does, what it does not, why we have chosen it;
- variables in python for integers;
- the effect of variable assignment on memory;

2.4 Lecture 4 (practicum) - Python basics

Set up a Python environment. Write a simple Python script that declares and assigns variables with various int, string, and float expressions.

2.5 Lecture 5 - datatypes, and expressions

The fourth lecture covers primitive datatypes and their associated expressions in the Python programming language.

Topics

- what are data-types and *why we do need them*?
- different Python data-types;
- arithmetic expressions;
- integer and floating point operators;
- boolean expressions;
- conditional expressions;
- very long expressions with conditionals vs temporary variables: the art of naming to encode knowledge.

Activities Call upon students to solve small riddles related to sample Python scripts on:

- Integers, strings, floats, bools;
- Integer, string, float, and bool varialbes;
- Semantics and post-conditions on variable-assignments.
- Integers, strings, floats, bool expressions;
- Conditional expressions;
- Semantics and post-conditions on expressions and conditional expressions.

2.6 Lecture 6 (practicum) - Python basics and grading

Grading and feedback on previous practicum.

2.7 Lecture 7 - conditional control-flow statements

The fifth week covers conditional control-flow statements in the Python programming language.

Topics

- making choices;
- `if-then` statements;
- `if-then-else` statements;
- the importance of an `else` statement;
- (slightly informal) semantics;
- exponential explosion of potential control-paths;
- expressive power of `if-then-else`.

Activities Call upon students to solve small riddles related to sample Python scripts on:

- `if-then` and `if-then-else` statements;
- how many possible final states of a program;
- semantics and post-conditions on conditional statements.

2.8 Lecture 8 (practicum) - conditionals

Write a simple *turtle* script that reads some input values and makes the turtle move according to the input values by using `if` statements.

2.9 Lecture 9 (practicum) - conditionals grading

Grading and feedback on previous practicum.

2.10 Lecture 10 - looping control-flow statements

The sixth (and last) week covers looping control-flow statements in the Python programming language.

Topics

- repeated behaviors;
- `while` statements;
- (slightly informal) semantics;
- (more than) exponential explosion of potential control-paths;
- expressive power of `while`;
- `for` statements;
- (slightly informal) semantics;
- `for` as a *limited* form of `while`.

Activities Call upon students to solve small riddles related to sample Python scripts on:

- `while` and `for` loops;
- how many possible final states of a program;
- semantics and post-conditions on loops.

2.11 Lecture 11 (practicum) - loops

Write a simple *turtle* script that makes the turtle move in a loop, square, or even spiral by using `for` or `while` statements.

2.12 Lecture 12 (practicum) - loops grading

Grading and feedback on previous practicum.