Information Technology

Introduction to programming with *Scribble*

FIT1040 Programming Fundamentals (Week 1)
Lecturer, Campus
Faculty of Information Technology
From 1969 ...

This free booklet will tell you how you can become a Computer Programmer

...and how you can train at home for big earnings in the world's newest, most exciting profession.

If you're dissatisfied with your present job, why

don't you become a programmer? So great is
the demand for programmers, you'll have your
choice of openings, with a growing future ahead.

For the vast majority of good positions, a high
school education, a logical mind and the right
preparation are all that are required. Lahalle, a
world leader in home education, will train you as
a programmer in your spare time.

Why not plan for bigger success in the field
where employers are looking for you? Authorities
agree that for many years to come there will be
more openings in computer programming than
enough to fill them! Even beginners can earn well;
and with experience, programmers can fill major
executive positions at salaries of $12,000 to
$15,000 a year—often as much as $30,000.

Without leaving your present position, you can
start your training now—at very low cost. You
can study at home, in hours of your own choosing.
Experienced Lahalle instructors will guide you by
mail every step of the way. Industry experts say
this is the most thorough course of its kind that
they have seen. The ease and rapidity with which
you learn will surprise you.

Lahalle has just published a fascinating booklet
"How To Become a Computer Programmer." In
simple, non-technical language this booklet tells
you what computer programming is; what the
programmer does; how you can earn by mail through
the school that has enrolled more than 1,500,000
people in its many home courses. Mail the cou-
upon for your free copy or write to Lahalle, 417 S.
Dearborn Street, Chicago, Illinois 60605.

© 1969 Lahalle Business Schools.

LA SALLE EXTENSION UNIVERSITY
417 S. Dearborn Street, Dept. 00-375, Chicago, Illinois 60605

blog.modernmechanix.com
Learning objectives

By the end of this week students should:
1. Know what's expected of them in the unit: the workload and the assessment
2. Know the location and nature of the resources available to help them study the unit
3. Know staff who are teaching them
4. Understand the basics of how to launch the *Scribble* application and the major components of the *Scribble* interface and how they are used to create software programs
5. Understand that the block approach (coding using pseudo-code) can be used to create complex and interesting software applications
Today’s agenda
Today’s agenda

- Learning to write computer software
- Introduction to the teaching staff
- The structure and content of the unit
- The assessment (what you have to do to get marks)
- Learning resources
- Introduction to programming with Scribble
- Summary
- Reminders
Learning to write computer software
Programming is

- … to hard for most people, it’s only nerds who can program.
- … really easy, only stupid people can’t learn to write software.
- … done overseas now, so we don’t need to learn it anymore.
- … a great career choice as there are not enough programmers locally to meet industry demand.
- … not needed for your IT career.
- … vital for your career, all IT-related jobs need some knowledge of programming.
- … all I want to learn at Monash so I can drop-out, make an app and sell it through the iTunes store and make my fortune.
WHO IS SCARED?
Introductory programming courses at Universities

- Usually a big problem for staff and students
  - Typically have high failure rates
  - Some students bored as it’s too easy
  - Many are out of their depth and have a horrible time
  - May that pass are so shaken by the experience that they are put off programming forever
  - Monash has been no exception …

- “What language will we learn?”
  - Java, Visual Basic, C#, Python, PHP, Pascal, Scheme, … there is no right answer – it’s probably the wrong question.
  - (btw we are going to learn Scribble.)
Syntax errors

- In traditional programming languages programs are made of a set instructions coded according to a set of rules that define what statements can be used and how they can be combined.
- Some of the rules are arbitrary
- Often when errors are made in the code (syntax errors) the error messages provided to the programmer to help them work out what’s gone wrong are obscure
- For beginners (and even experienced programmers), this than make programming frustrating … infuriating … or worse.
Typical syntax error and messages

- Example of a site written in traditional industrial language PHP (with CakePHP framework).

```php
<div id="content">
  <?php
  echo $this->Html->div("pagelabel", "BEST SELLERS");
  ?>
  
  <br />
  
  <br />
  <table id="products" cellspacing="0" cellpadding="0" width="100%">
    <tr width="100%">
      <th>
        foreach ($products as $product) {
          seedu = "products/".$product['Product']['image']
          $alttext = $product['Product']['name'];
          $linkurl = "products/view/".$product['Product']['url']
          echo "<img src="".$ seedu."" alt="".$alttext."">
          echo "&lt;a href="".$linkurl.""&gt;";
          echo $this->Html->image($ seedu);
          // echo "<img src="".$ seedu."" alt="".$alttext."">
          echo "&lt;/a&gt;";
          echo "</div>";
          echo "&lt;/td&gt;"
        } 
      
      <br />
      <br />
      <tr width="100%">
        <th>
          foreach ($products as $product){
            seedu = "products/".$product['Product']['image']
            $alttext = $product['Product']['name'];
            $linkurl = "products/view/".$product['Product']['url']
            echo "<img src="".$ seedu."" alt="".$alttext."">
            echo "&lt;a href="".$linkurl.""&gt;";
```
Remove a single ; symbol

```php
$imgeloc = "products/".$product['Product']["image"];
$alttext = $product['Product']["name"];
$linkurl = "products/view/".$product['Product']["cleanurl"];

echo "<td>";
echo "<div class="product">";
```

Taking out one ; and the site vanished. No errors, nothing!
NO SYNTAX ERRORS

ONLY LOGIC AND RUN-TIME ERRORS
No syntax errors in FIT1040

- Our aim is to teach you to think like a programmer
- You will be able to design and create software applications that are well structured, modular, easy to understand and as a result easy to maintain and debug
- You can deal with the delight’s of syntax errors in your second unit on programming – where you will learn to use an industrial language to do many of the things we will cover in this unit
Introduction to the teaching staff
The structure and content of the unit
FIT1040 Programming fundamentals

- Common core unit – all students taking an IT degree take this unit
  - Simply means that the unit is important!
- Students will learn to develop descriptions of algorithms and program logic using pseudo-code which will be implemented as working software programs using in a visual procedural programming language.
- The topics covered will include the fundamental concepts:
  - data types and structures,
  - basic types of input and output,
  - program control structures, and
  - modular design along with the basics of event-driven programming and objects.
Learning outcomes

Developed the skills to:
- use diagrams to design solutions for programming problems;
- apply problem solving strategies;
- use pseudo-code to design algorithms;
- create and test simple computer programs;
- analyse and debug existing programs; and
- write a test plan.

Demonstrated the communication skills necessary to:
- produce documentation for a program; and
- explain an existing program.
Learning outcomes (cont).

An understanding of:

- the relationship between a problem description and program design;
- the use of design representations;
- the sequence of steps that a computer takes to translate source code into executable code; and
- primitive data types and basic data structures.

Developed attitudes that enable them to:

- adopt a problem-solving approach;
- recognize the importance of programming and documentation conventions;
- act in accordance with best practice, industry standards and professional ethics.
Workload

Each week there is:

- Two hours of lectures
- One hour of tutorial
- Two hours of laboratories

In addition to those 5 hours in class we expect you’ll spend another 7 hours in your own time:

- Working on assignments
- Summarizing your notes
- Reading, studying and programming using Scribble

- University study is different – you need to be self organizing and self motivating. People will help, but you have to ask for it.
Lecture schedule

1. Introduction to programming with Scribble
2. Finding errors in programs: testing and debugging
3. Using variables in programs
4. Making decisions in programs
5. Using loops
6. Using loops to make art
7. Using lists in loops
8. Searching and sorting algorithms
9. Using abstraction to represent game play
10. Making games using sprites and messages
11. Software development and programming environments
12. Software development and programming environments cont…
Assessment
Assessment

- **Weekly tests – 20%**
  - In weeks 2 to 11 (10 weeks) at the end of each tutorial and laboratory there is a short test (each worth 1%)

- **Assignment 1 – 10%**
  - Sunday midnight (local time) Week 7

- **Assignment 2 – 10%**
  - Sunday midnight Week 11; Interviews held in Week 12

- **Exam – 60%**
  - 3 hours, closed book, plenty of samples and examples to help you prepare – more on that as we approach the exam time.
Assignment 1: Programming basics

This assignment will require the development of a software application with Scribble that reacts to user interface events, taking input from a user and then performing a complex calculation. The task will require creating a programming solution to a calculation-based problem, creating software that reacts to an event, makes decisions with IF-THEN-ELSE logic, stores user input in variables, and makes calculations using variables.
Assignment 2: Advanced Programming Application

This assignment will require the development of a software application with Scribble that implements a single-user game. The task will require creating an abstraction of the game state and changing that state as the user plays the game. This will require the creation of appropriate data structures to store the game state, the player's movers, and scripts that represent the game rules. The software will make decisions with a IF-THEN-ELSE logic, process data using loop-based logic, and display appropriate status updates as the game is played, won or lost.
Marking guide for assignments

(More detail on everything on the unit web site).

1. The application must run correctly. Evidence of testing is required.
2. The application must meet the problem specification.
3. The application logic should be documented with a flowchart.
4. The application sprites and scripts should be constructed in a way that makes them easy to understand and maintain.
5. Students should be able to answer questions about their own work. (#Assignment 2)
Learning resources
Learning resources

- Moodle-based web site
  - Lecture, tutorial and laboratory materials
  - Discussion forums
- Scribble and the BlockBooks
  - Available in Apple’s iBook format and in pdf
- Recommended reading (there is no prescribed text book)
- The @fit1040 twitter account
  - Paper.li
- The staff!
The unit web site on Moodle

FIT1040 Programming Fundamentals

This unit - FIT1040 Programming Fundamentals - will provide you with an overview of the fundamentals required to create programs. You will learn to develop descriptions of algorithms and program logic using pseudo-code, which will be implemented as working software programs using a visual procedural programming language called Scribble. The topics covered will include the fundamental concepts: data types and structures, basic types of input and output, program control structures, and modular design along with the basics of event-driven programming and objects. These topics will be covered while placing an emphasis on the need to design program code that is easy to maintain, readable, tested, and well documented.

At the end of the unit you will expected to be able to apply to knowledge and skills learned in further units that cover software development using industry strength programming languages.

Quick links
Week: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12
Unit guide, Assignments, Exam information, Discussion forums, Podcast

Week 1 learning objectives:
By the end of this week students should:
- Know what's expected of them in the unit: the workload and the assessment
- Know the location and nature of the resources available to help them study the unit
- Know staff who are teaching them
- Understand the basics of how to launch the BYOB application and the major components of the BYOB interface and how they are used to create software programs
Scribble and the BlockBooks

- monash-blockbooks.appspot.com
- monofonik.github.com/scribble/
Recommended reading

Twitter

- The unit has a twitter account FIT1040
- Twitter is free, if you don’t have an account, why not create one and follow along. It’s a quick and easy way of asking questions and staying in touch.
- Twitter sources are used to generate a weekly “newspaper”.
- Also used to curate a set of video help resources at screenr.com/user/fit1040
Introduction to programming with Scribble
A QUICK LOOK AT THE BLOCK APPROACH TO CODING
override public function init():void
{
    super.updatedListener(mall, function(List:Array):void{
        if (wrapper.enabled)
        {
            checkInput()
            if ((pressedLeft) && (pressedRight))
        }
    });
}

property changefont "Arial", "Arial";
Scribble and it’s relatives

- Tools to learn programming
  - StarLOGO TNG (MIT)
  - Scratch (MIT)
  - BYOB (Berkeley)
  - Scribble (Monash)
  - Cellular and Enchanting (also Monash)

- (Commercial software development tool)
  - Google App Inventor (Google, now MIT)
  - Stencyl
Why Scribble?

- Stencyl is too complex and costs $
- Scratch doesn’t have enough functionality
  - Version 2 does come closer
- BYOB/SNAP! introduces complex data structures (lists) and code reuse through prototype objects (blocks)
- Scribble is Monash developed
  - Extends BYOB by making some “art” elements easier and better
  - Also slightly better text I/O
  - Local support
  - Default icon is significantly better
Scribble programs

- Programs are scripts
- Developed by dropping and dragging pseudocode elements from a toolbox to a script building area
  - Sprites and backgrounds can have scripts
- Scribble is always live
  - Scripts can run in parallel
- Events and messages feature strongly in Scribble applications
- No syntax errors

From the The Appel Academy
3. The script building area

1. The “stage”

4. The scripting toolbox

2. The sprite selection area
A QUICK LOOK AT SCRIBBLE
Summary

- Lectures, tutorials and laboratories – 5 hours a week
  - plus 7 hours of work in your own time
- An exam and two assignments + weekly tests
- Learning to program using *Scribble*
  - It’s free and there are great resources to help you
- You can’t make syntax errors with *Scribble*
- The “block” approach can be used to make complex software
  - Translates directly to traditional code
  - Scripts are developed by dropping and dragging pseudo-code elements from a toolbox to a script building area
- In this unit you will learn to think like a programmer, later on learning a specific language will be easier.
Reminders
Reminders

- University study is different
  - You are in control

- No shortage of learning resources
  - Start now! Download Scribble and install it on your computer

- Tutorials and laboratories start this week.

- From next week on each tutorial and laboratory will end with a short test worth 1% (total of 20%).

- Relax and enjoy – the staff will do everything they can to help you learn and enjoy developing your skills and ability as programmers

- Monash is pretty great – make the most of it!