COMP284 Scripting Languages
Lecture 1: Overview of COMP284
Handouts

Ullrich Hustadt

Department of Computer Science
School of Electrical Engineering, Electronics, and Computer Science
University of Liverpool

Contents

How many programming languages should you learn?

1. Academic / Educational viewpoint:
   Learn programming language concepts and use programming languages to gain practical experience with them
   - imperative / object-oriented — C, Java
   - functional — Maude, OCaml, Haskell
   - logic/constraint — Prolog, DLV
   - concurrent

   then all (other) programming languages can be learned easily
2. An employer's viewpoint:
   Learn exactly those programming languages that the specific employer needs
3. Compromise: Spend most time on 1 but leave some time for 2 to allow more than one language from a class/paradigm to be learned
4. Problem: Which additional language do you cover?
   $\rightsquigarrow$ Look what is used/demanded by employers

Programming languages: Job ads

Software Developer
(Digital Repository)
University of Liverpool - University Library
£31,020 - £35,939 pa

To work as part of a small team based in the University Library, working closely with the University's Computing Services Department on the institutional digital repository, recommending and developing technical solutions, tools and functionality to integrate the repository with other internal systems and to enable research outputs to be shared externally. You will be an experienced Software Developer with knowledge of LAMP technologies such as XML, XSLT, Perl and Javascript. You will hold a degree in Computer Science or a related discipline and/or have proven industrial experience of software development. The post is full time, 35 hours per week.

Job Ref: A-576989

Programming languages: Job ads

Senior Software Development Manager
IMDb Video and Recommendations (Seattle, WA)

IMDb (a wholly-owned subsidiary of Amazon) is recruiting for a Senior Software Development Manager to lead our "What to Watch" team. You'll be charged with transforming IMDb from a reference site to a place where hundreds of millions of people find and discover what to watch across a variety of video providers, and seamlessly connect them with watching the movies and TV shows best suited for them, wherever and whenever they may be.

Basic qualifications:

- Bachelor's degree in Computer Science, Computer Engineering or related technical discipline
- 10+ years of experience as a software developer
- 5+ years experience managing people
- Software development experience in OOP, Java, Perl, HTML, CSS, JavaScript, Linux/UNIX, AJAX, MySQL

Programming languages: Job ads

Full-time Remote Worker
AOL Tech (Engadget, TUAW, Joystiq, Massively)

AOL Tech is looking for a great front-end developer who can help us take Engadget and our other blogs to new levels.

The ideal candidate is highly proficient in JavaScript/jQuery, comfortable with PHP / mySQL and experienced in web design, optimization and related technologies for desktop and mobile. A solid understanding of mobile-first design is a must.

Requirements:

- High proficiency in JavaScript/jQuery
- Familiar with spriting, lazy loading, and other general performance-optimized techniques
- Mac access for compatibility with current tools
- HTML5/CSS3
- Git, SSH

Websites and Programming Languages

Website	Client-Side	Server-Side	Database
Google	JavaScript	C, C++, Go, Java, Python, PHP	BigTable, MariaDB
Facebook	JavaScript	Hack, PHP, Python, C++, Java, ...	MariaDB, MySQL, HBase Cassandra
YouTube	Flash, JavaScript	C, C++, Python, Java, Go	BigTable, MariaDB
Yahoo	JavaScript	PHP	MySQL, PostgreSQL
Amazon	JavaScript	Java, C++, Perl	Oracle Database
Wikipedia	JavaScript	PHP, Hack	MySQL, MariaDB
Twitter	JavaScript	C++, Java, Scala	MySQL
Bing	JavaScript	ASP.NET	MS SQL Server

Wikipedia Contributors: Programming languages used in most popular websites. Wikipedia, The Free Encyclopedia, 20 October 2017, at 11:28. `http://en.wikipedia.org/wiki/Programming_languages_used_in_most_popular_websites` [accessed 23 October 2017]

Scripting languages

Script

A user-readable and user-modifiable program that performs simple operations and controls the operation of other programs

Scripting language

A programming language for writing scripts

Classical example: Shell scripts

```
#!/bin/sh
for file in *; do
    wc -l "$file"
done
```

Print the number of lines and name for each file in the current directory

Scripting languages: Properties

- Program code is present at run time and starting point of execution
  - compilation by programmer/user is not needed
  - compilation to bytecode or other low-level representations may be performed 'behind the scenes' as an optimisation
- Presence of a suitable runtime environment is required for the execution of scripts
  - includes an interpreter, or just-in-time compiler, or bytecode compiler plus virtual machine
  - typically also includes a large collection of libraries
- Execution of script is typically slower than the execution of code that has been fully pre-compiled to machine code


```
#!/bin/sh
for file in *; do
    wc -l "$file"
done
```

Scripting languages: Properties

- Rich and easy to use interface to the underlying operating system, in order to run other programs and communicate with them
  - rich input/output capabilities, including pipes, network sockets, file I/O and filesystem operations
- Easy integration within larger systems
  - often used to glue other systems together
  - can be embedded into other applications


```
#!/bin/sh
for file in *; do
    wc -l "$file"
done
```

Scripting languages: Properties

- Variables, functions, and methods
  typically do not require type declarations
  (automatic conversion between types, e.g. strings and numbers)
- Some built-in data structures
  (more than in C, fewer than in Java)
- Ability to generate, load, and interpret source code at run time
  through an eval function

JavaScript

```
var x   = 2;
var y   = 6;
var str = "if (x > 0) { z = y / x } else { z = -1 }";
console.log('z is ', eval(str));  // Output: z is 3
x       = 0;
console.log('z is ', eval(str));  // Output: z is -1
```

Scripting languages: Properties

- The evolution of a scripting language typically starts with a limited set of language constructs for a specific purpose

  Example: PHP started as set of simple 'functions' for tracking visits to a web page

- The language then accumulates more and more language constructs as it is used for a wider range of purposes
- These additional language constructs may or may not fit well together with the original core and/or may duplicate existing language constructs
- During this evolution of the language, backward compatibility may or may not be preserved

$\rightsquigarrow$ Language design of scripting languages is often sub-optimal

Aims

1. To provide students with an understanding of the nature and role of scripting languages
2. To introduce students to some popular scripting languages and their applications
3. To enable students to write simple scripts using these languages for a variety of applications

Learning Outcomes

At the end of the module students should be able to

1. compare and contrast languages such as JavaScript, Perl and PHP with other programming languages
2. document and comment applications written using a scripting language
3. rapidly develop simple applications, both computer and web-based, using an appropriate scripting language

Delivery of the module (1)

1. Lectures
   - Structure:
     16 to 18 lectures
   - Schedule:
     1 or 2 lectures per week spread over 9 weeks

     See your personal timetable and e-mail announcements for details
   - Lecture notes and screencasts are available at
     `cgi.csc.liv.ac.uk.uk/~ullrich/COMP284/notes`
   - Revise the lectures before the corresponding practical
   - Additional self study using the recommended textbooks and the on-line material is essential

Delivery of the module (1)

❷ Practicals

- Structure:
  - 7 practicals with worksheets (3 Perl, 2 PHP, 2 JavaScript)
    ⇝ gain understanding via practice
    ⇝ get answers to questions about the lecture material
  - Up to 3 additional practicals for questions about the assignments
- Schedule:
  1 practical per week for about 10 weeks

  Practicals start in week 2
- Practicals assume familiarity with Linux and departmental Linux systems
  ⇝ To recap, use the worksheets available at
  `cgi.csc.liv.ac.uk.uk/~ullrich/COMP284/notes`
- Practicals assume familiarity with the related lecture material

How to learn a new programming language

- Once you know how to program in one programming language, additional programming languages are best learned by a process of enquiry and practice guided by existing experience
- Typically, the questions that guide you are
  - What kind of . . . are there?
    Example: What kind of control structures are there?
  - What is the syntax for . . . ?
    Example: What is the syntax for conditional statements?
  - What happens if . . . ?
    Example: What happens if I is divided by 0?
  - How do I . . . ?
    Example: How do I catch an exception?
- Talk to other people who are currently trying to learn the same language or have already learned it
  ⇝ Ask what has surprised them most

How to learn a new programming language

- Once you know how to program in one programming language, additional programming languages are best learned by a process of enquiry and practice
- The best kind of learning is learning by doing
  - ⇝ The questions posed on the previous slide are often best explored by experimenting with small sample programs ('toy' programs)
- Work on substantive programs
  - ⇝ You need to convince employers that you have worked on programs more substantive than 'toy' programs
  - ⇝ The assignments are 'pretend' substantive programs but in reality are too small
- Employers value experience, in particular, the experience that you get from overcoming challenges
  - ⇝ Assignments that are not challenging are of limited value

Delivery of the module (3)

3. Office hours

   Monday, 16:00 Ashton Room 1.03
   but always arrange a meeting by e-mail first
   (U.Hustadt@liverpool.ac.uk)

4. Announcements will be send by e-mail
   - You should check you university e-mail account at least every other day
   - Always use your university e-mail account
     if you want to contact me or any other member of staff

Recommended texts

- Core reading
  - R. Nixon:
    Learning PHP, MySQL & JavaScript.
    O'Reilly, 2009.
    Harold Cohen Library: 518.561.N73 or e-book
    Learning PHP . . ., 4th edition.
    O'Reilly, 2014.
  - R. L. Schwartz, brian d foy, T. Phoenix:
    Learning Perl.
    O'Reilly, 2011.
    Harold Cohen Library: 518.579.86.S39 or e-book
    Learning Perl, 7th edition.
    O'Reilly, 2016.
- Further reading
  - M. David:
    HTML5: designing rich Internet applications.
    Focal Press, 2010.
    Harold Cohen Library: 518.532.D24 or e-book
  - N. C. Zakas:
    Professional JavaScript for Web Developers.
    Wiley, 2009.
    Harold Cohen Library: 518.59.Z21 or e-book

Assessment

- This is a coursework-based module (no exam)
- Three assessment tasks need to be completed throughout the semester:
  - Perl Deadline: Friday, 2 March, 17:00
  - PHP Deadline: Monday, 9 April, 12:00
  - JavaScript Deadline: Friday, 27 April, 17:00
- Effort required: about 10 hours each
- Available at: `http://cgi.csc.liv.ac.uk/~ullrich/COMP284/`

Attendance and Performance

	Students	Average Lecture Attendance	Average Practical Attendance	Average Module Mark
2011-12	33	76.0%	70.0%	63.1
2012-13	58	82.0%	69.0%	64.5
2013-14	107	80.0%	60.0%	59.1
2014-15	119	71.8%	65.2%	54.5
2015-16	76	67.4%	46.8%	57.9
2016-17	114	43.8%	38.3%	53.0

- From 2014-15, screencasts of the lectures were available to students
- From 2015-16, the requirement to write a report on each program was dropped
- Hypothesis 1:
  Lecture Attendance $> 75\%$ and Practical Attendance $> 65\% \Leftrightarrow$ Module Mark > 62
- Hypothesis 2:
  Screencasts Available $\Leftrightarrow$ Module Mark < 59

Academic Integrity

- Plagiarism occurs when a student misrepresents, as his/her own work, the work, written or otherwise, of any other person (including another student) or of any institution
- Collusion occurs where there is unauthorised co-operation between a student and another person in the preparation and production of work which is presented as the student's own
- Fabrication of data occurs when a student enhances, exaggerates, or fabricates data in order to conceal a lack of legitimate data

If you are found to have plagiarised work, colluded with others, or fabricated data, then you may fail COMP284

Serious 'offenders' may be excluded from the University

Do not try to take a 'shortcut'
You must do the work yourself!

Academic Integrity: Lab rules

- Do not ask another student to see any part of their code for a COMP284 assignment
  ⇝ contravention of this leads to collusion
- Do not show or make available any part of your code relating for a COMP284 assignment to any other student
  ⇝ contravention of this leads to collusion
- Do not share (links to) on-line material that might help with a COMP284 assignment
  ⇝ contravention of this leads to collusion
- Lock your Lab PC when you leave it alone
- Where you use any material/code found on-line for a COMP284 assignment, you must add comments to your code indicating its origin by a proper academic reference
  ⇝ contravention of this is plagiarism
  ⇝ acknowledged code re-use may still result in a lower mark