Section 1

1. Preliminaries

1.1 Administration
1.2 Motivation
1.3 Hello, World!
1. Preliminaries

1.1. Administration

1.2 Motivation

1.3 Hello, World!
Instructor

- Prof. David H. Lorenz

- Office hours:
  Wednesday 10:30–11:30; Taub 743
  [http://openu.YouCanBook.me](http://openu.YouCanBook.me)

- Mail: [mailto:dhlorenz@CS.Technion.AC.IL](mailto:dhlorenz@CS.Technion.AC.IL)
Chief Teaching assistant

- Mr. Matan Peled:
  - Office hours: Sun 16:30–17:30; Taub 313
  - Tutorials hours: Sun 14:30–16:30; Taub 3
  - Mail: mailto:mip@CS.Technion.AC.IL
Teaching assistants I

- Mr. Tomer Gelber:
  - Office hours: Thu 11:00–12:00; Taub 321
  - Tutorials hours: Mon 8:30–10:30; Taub 8
  - Mail: tomerghelber@CS.Technion.AC.IL

- Mr. Tal Shinkar:
  - Office hours: Tue 12:30–13:30; Taub 320
  - Tutorials hours: Tue 10:30–12:30; Ulman 604
  - Mail: tal.s@CS.Technion.AC.IL
Teaching assistants II

- Mr. Lior Ben Yamin:
  - Office hours: Wed 16:30–17:30; Taub 215
  - Tutorials hours: Wed 14:30–16:30; Ulman 311
  - Mail: mailto:lior.b@cs.technion.ac.il
Graders

- Ms. Ofir Alexi
- Mr. Guy Suday
Syllabus

Teaser: unfamiliar (probably) terms

\(\lambda\)-functions, piddles, currying, gradual typing, Unit type, None type, generators, dynamic scoping, continuations, punning, closures, structograms, ...

- **Languages and paradigms**
  - Imperative: C, C++, Pascal, Java, AWK, Go, ...
  - Functional: ML, Haskell,
  - Declarative: Prolog

- **Schedule (bird’s eye)**

<table>
<thead>
<tr>
<th>Topic</th>
<th>Weeks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>1–2</td>
</tr>
<tr>
<td>Types</td>
<td>3–5</td>
</tr>
<tr>
<td>Storage</td>
<td>6–8</td>
</tr>
<tr>
<td>Commands</td>
<td>9–10</td>
</tr>
<tr>
<td>Advanced constructs</td>
<td>11–13</td>
</tr>
</tbody>
</table>
At the end of the course you’ll know

- What distinguishes a certain PL from other PLs.
- A variety of mechanisms in familiar and less familiar PLs
- Programming in the functional language ML
- Some basic concepts from Pascal, Prolog and other PLs
- Main skills:
  - Quickly learn a new PL
  - Evaluate PLs
  - Use any PL more cleverly
Course material

Official web site

http://webcourse.cs.technion.ac.il/234319

where you will find:

- Printouts of slides in a variety of formats
- Lecture notes by Prof. J. Gil
- Some notes taken by students
- Past exams

And the usual: grades, current assignment information & FAQs, periodical announcements,...
Resources managed by students

Indexed Q&A site

https://safot.cs.technion.ac.il/
Text books

Main text book


But also lecture notes and slides by Prof. J. Gil.
For further, in depth, reading:


“Programming Languages: Concepts and Constructs” (2nd Ed), Ravi Sethi. Addison-Wesley, 1996.
Bibliography for tutorials

Programming languages taught: Pascal, ML, Prolog

- “Prolog Programming for Artificial Intelligence”, by Ivan Bratko. Addison-Wesley.
Regulations

**Highlights**

- **Midterm exam** none
- **Prerequisites** enforced
- **Co-requisites** enforced
- **Homework grades** crucial
- **Old homework grades** cannot be transferred

**Appeals**

- considered favorably
- treated seriously
- must be in writing
- must be signed by student
- no “grades negotiation”

*discussing appeals or grades with staff voids all appeal rights*
Grade components

Assignments (grade denoted by $A$, $0 \leq A \leq 100$)
- Every 2–3 weeks
- Mandatory
- Typically includes both programming and mini-research problems
- Teams of two students each (strict!)
- Matching services provided by teaching assistants

Exam (grade denoted by $E$, $0 \leq E \leq 100$)
- Typically includes:
  - at least one homework assignment
  - at least one past exam question
  - 8–12 questions
Final grade

Denoted by $F$, $0 \leq F \leq 100$: 

$$F = \begin{cases} 
\text{round}(E) & \text{if } E - 1.5 \leq A \leq E \\
\text{round}(E) & \text{if } E \leq 50 \\
\text{round}(E + \sqrt{A - E}) & \text{if } 50 < E \leq A \\
\text{round}(E - \sqrt{E - A}) & \text{otherwise}
\end{cases}$$  

(1.1)
1. Preliminaries

1.2. Motivation
Why take this course?

- **Fun**
  - ML is neat
  - **Prolog** is elegant
  - There is more to it than *C*, *C++*, *Java*
  - Enhance thinking flexibility

- **Professional skills**
  - Over 2,000 different languages out there
  - Common concepts and shared paradigms
  - Framework for comparative study of PLs

- **Useful for other courses**:
  - OOP
  - Compilation
  - Software Engineering
  - Semantics of PLs
  - Memory Management
Discovering you speak prose all your life

More generally, learning, something new about old things.

So, yes, the course will be telling you new stuff about old stuff...

And, we will practice some new modes of thought
New modes of thought

New

- Programming languages mechanisms much beyond the `if` and `while`
- Programming techniques
- Paradigms of thought
- Directions for your minds

And also,

- Get ready to *Object Oriented Programming* and other advanced courses. (the instructor of the *Object Oriented Programming* course paid me to mention this)
- Hone web-search skills. (Google paid me to include this in the topic of our course)
But also many practical benefits

**Main objective**

learn, understand, and evaluate any new programming language
Or, at least, understand the terminology/jargon of the trade:

What kind of a beast is JavaScript?

- Imperative,
- With prototypes (object-based, but not object-oriented),
- Functions are first-class entities,
- Has lambda functions,
- With closures,
- Is weakly typed,
- Has dynamic typing,
- Has static scoping,
- ... and a must-know for any modern website developer!

By the end of these course, many of these terms will be covered in depth.
Reasons to drop this class

- No mathematical interest. This is not yet another technical course:
  - Many “soft” definitions
  - Much reliance on common sense
  - No theorems, proofs, lemmas, or integration by parts
  - No easy grades for mathematical genuises

- No “computational” interest. The expressive power of all programming mechanisms and computational devices is basically the same

The Church-Turing hypothesis

- The DOS batch language and JAVA are equivalent
- The Commodore 64 and the latest 8-core CPUs are equivalent

- No “algorithmic” interest. You don’t discover new fascinating algorithms using better programming languages.
Possible approaches to teach PL

- Define and compare paradigms of PLs
- Present formal approaches to syntax and semantics
- Present ways of implementing and analyzing programs in various PLs
- Show the concepts that must be dealt with by any PL, and the possible variety in treatment
Why PL is a difficult course

- To teach you PL theory, we need to draw examples from different PLs.
- Right now, most of you know \( \approx 2.5 \) languages (C, C++, Unix shell scripts).
- Examples in these slides come from (alphabetically): Ada, Algol, AWK, C, C++, C#, Eiffel, Fortran, Haskell, Java, ML, Lazy-ML, Lisp, Pascal, Prolog, Python, SQL, and probably a few more I forgot.
- Can you please learn all these for next week?
- Recitations are here to help.
Who needs PLs?

- Computers’ native tongue is machine language
- Programmers need higher level languages, because:
  - They can’t write machine language correctly
  - They can’t read machine language fluently
  - They can’t express their ideas in machine language efficiently
  - Life is too short to program in machine language.
- A formal language is not only a man-machine interface, but also a person-to-person language!

**Conclusion:** PLs are a compromise between the needs of humans and the needs of machines
What is a PL?

- A linguistic tool with formal syntax and semantics
- A consciously designed artifact that can be implemented on computers
- “A conceptual universe for thinking about programming” (Alan Perlis, 1969)
Language processors

- A system for processing a language:
  - Compiler
  - Interpreter
  - Syntax directed editor
  - Program checker
  - Program verifier

- Studied in other courses:
  - Compilation
  - Program verification
  - Software engineering

To know the semantics of a language (the *function* a program encodes) one can ignore its implementation.
## Relations to other fields in computer science

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Databases and Information Retrieval</td>
<td>Query languages - languages for manipulating databases.</td>
</tr>
<tr>
<td>Human-Computer Interaction</td>
<td>PLs are designed to be written and read by humans</td>
</tr>
<tr>
<td>Operating Systems</td>
<td>Input-Output support. Storage management. Shells are in fact PLs.</td>
</tr>
<tr>
<td>Computer Architecture</td>
<td>PL design is influenced by architecture and vice versa.</td>
</tr>
<tr>
<td></td>
<td>Instructions sets are PLs. Hardware design languages.</td>
</tr>
</tbody>
</table>
Closely related topics

Automata and Formal Languages, Computability  Provide the foundation for much of the underlying theory.

Compilation  The technology of processing PLs.

Software engineering  The process of building software systems.
1. Preliminaries

1.3. Hello, World!
How many ways to say “Hello, World”? 

- The following slides present examples of some of the most popular computer program “Hello, World” in various programming languages.
- See how many you can recognize?
- More examples:
  - http://www.latech.edu/~acm/HelloWorld.shtml
Assembly 8086 (for MS DOS)

```assembly
.model small
.stack 100h
.data
.helloMessage db 'Hello, World',0dh,0ah,'
.code
main proc
  mov ax,@data
  mov ds,ax
  mov ah,9
  mov dx,offset helloMessage
  int 21h
  mov ax,4C00h
  int 21h
main endp
end main
```
Hello, world.

```fortran
PROGRAM HELLO
WRITE(*,10)
10 FORMAT('Hello, world')
END
```
/* HELLO WORD PROGRAM 
TO OUTPUT HELLO WORLD */

HELLO: PROCEDURE OPTIONS (MAIN);

   PUT SKIP DATA( 'HELLO, WORLD' );

END HELLO;
Ada

with i_o;
use i_o;
procedure hello is
begin
  put ("Hello, World");
end hello;
Prolog

hello :-
   printstring("Hello, World").
printstring([]).
printstring([H|T]) :-
   put(H),
   printstring(T).
Snobol 4

```
OUTPUT = 'Hello, World'
END
```
The Chef PL

Lots of food for one person

Hello World Souffle.
Ingredients. 72 g haricot beans 101 eggs 108 g lard 111 cups oil 32 zucchinis 119 ml water 114 g red salmon 100 g dijon mustard 33 potatoes
Serves 1.
LISP

( DEFUN HELLO-WORLD ()
  (PRINT (LIST 'HELLO 'WORLD)) )
1. Preliminaries

1.3. Hello, World!

Transcript show: 'Hello, World'; cr
**PostScript**

```
%!PS
1.00000 0.99083 scale
/Courier findfont 12 scalefont setfont
0 0 translate
/row 769 def
85 {/col 18 def 6 {col row moveto (Hello, World) show /col col 90 add def}
repeat /row row 9 sub def} repeat
showpage save restore
```