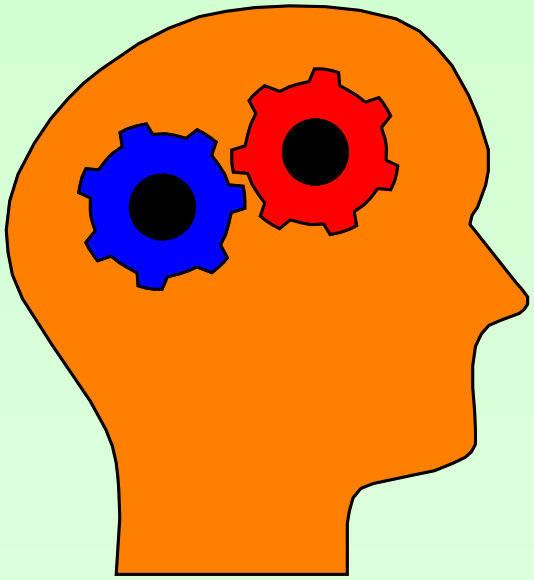
CS2104: Programming Languages Concepts

Lecture 1 : **Overview**



"Language Concepts to Support Programming and Abstraction"

Lecturer : Chin Wei Ngan

Email : chinwn@comp.nus.edu.sg

Office : COM2 4-32

Course Objectives

- cover key concepts in programming languages
- knowledge of language paradigms
- support for software *abstraction*, *reuse* and *safety*
- improving your programming skills
- make learning a new programming language easier
- highlight successful/advanced languages

Course Outline

- Lecture Topics (12 weeks)
  - Review key programming concepts (C vs Haskell/OCaml)
  - Values, Objects, Functions and Recursion (with Haskell)
  - Typeful and Higher-Order Programming (with Haskell)
  - Abstraction with Type Classes (with Haskell + Scala)
  - Monads and Parsing (with Haskell)
  - Imperative Programming (with OCaml/Scala)
  - OO Programming (with OCaml/Scala)
  - Dynamic Languages with Python
  - Logic programming with Prolog
  - CLP – constraint solving
  - Lambda Calculus

Administrative Matters

- IVLE for forum/lecture notes/exercises/submissions

- 2-hour Tutorial/Labs
  - (i) tutorial questions
  - (jj) lab exercises (4 graded assignments & 1 project)

- course CA/exams breakdown

- tutorial participation	10%
- lab/project assignments	35%
- quizzes/tests	15%
- examination	40% (approx)

Online ReadingTextbooks

- lecture slides/notes via IVLE

- Reading Materials (mostly online):

  ocaml.org
  caml.inria.fr
  www.haskell.org
  www.python.org
  www.scala-lang.org
  www.swi-prolog.org
  Free PL books :
  http://www.freebookcentre.net/Language/langCategory.html
  http://www.e-booksdirectory.com/programming.php

Optional Textbooks

- Concepts of Programming Languages
  Robert W Sebesta

- Concepts in Programming Languages
  John W Mitchell

- Concepts, Techniques and Models of Computer Programming
  Peter Van Roy and Self Haridi (NUS online library)

Lab Assignment/Homework

- Lab assignments in different successful and advanced languages (OCaml, Scala, **Haskell**, Prolog, Python).
- Reinforce concepts taught in class.
- Programming is a skill. It requires lots of practice.
- Pre-requisite to passing course
  Do Homework seriously → Pass Course

Why Study Concepts of PLs?

- Inside any successful software system is a good PL

  Emacs : *Elisp*

  Word, PPT : *VBScript*

  Quake : *QuakeC*

  Facebook : *FBML, FBJS, Hack (in HHVM)*

  Twitter : *Ruby on Rails/Scala*

  Also: *Latex, XML, SQL, PS/PDF*

Benefits of Good PL Features

- Readability
- Extensibility.
- Modifiability.
- Reusability.
- Correctness.
- Easy Debugging

What Drives the Development of PL?

- Novel ways of expressing computation
- Better execution model (e.g. dataflow)
- Tackle complex problems (with simpler solution)
- Proof of Concept
- Puristic viewpoint
- Better Reliability
- Domain-Specificity

History of Programming Languages

- Assembly (early 1950s)
- Fortran (late 1950s)
- Lisp (1958)
- Algol (1960s)
- Cobol (1960s)
- Prolog (1972)
- C (1973 – birth of Unix)
- Ada (1970s – defense)
- SQL (late 1970s)
- C++ (1985)
- ML (1980), OCaml (early 1990), Haskell (1987)
- Java (1995)
- Perl, Python, Javascript,PHP, VB (1990s)
- Scala (first released in 2003)
- C# (2000)
- Go (2009)

Lambda Calculus (1930s)

Programming Paradigms

- Imperative Programming
- Functional Programming
- Logic Programming
- Object-Oriented Programming
- Constraint Programming
- Event-Driven Programming (not covered)
- Aspect-Oriented Programming (not covered)

Advanced Language

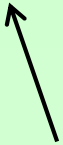
- Strongly-typed with polymorphism
- Higher-order functions
- Pure and Lazy Language.
- Algebraic data types + records
- Exceptions
- Type classes, Monads, Arrows, etc
- Advantages : concise, abstract, pure

- Why use Haskell ?

Cool, clean and Pure

Hello World in Haskell

```
putStrLn "Hello World!"
```



pure function with type `[Char] -> IO()`

Compilation:

```
ghc -o hello hello.hs
```

Execution:

```
./hello
```

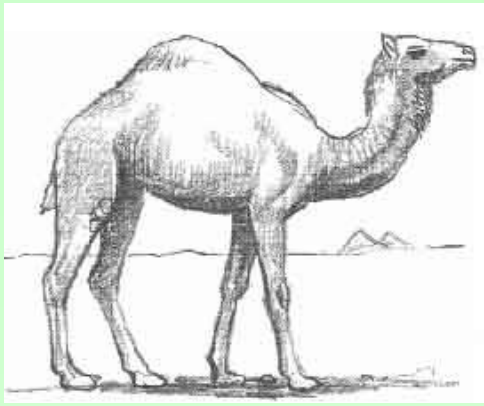
Increment Method (in Haskell)

```
inc :: Int -> Int
inc x = x+1
```

```
\ x -> x+1
```

```
(+1)
```

Versatile Language - OCaml



- Rich data structures (algebraic data types, records, polymorphism, variants, GADT).
- Typeful higher-order functional and object-oriented language.
- Support for stateful imperative programming.
- Powerful module system.
- Advantages : versatile, abstract, easy reuse
- Why OCaml?

Powerful, Versatile and Practical

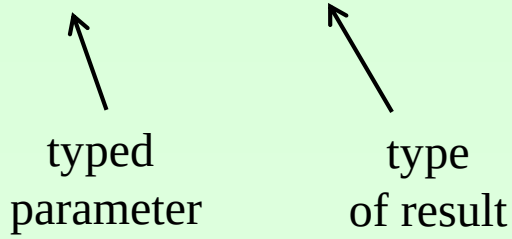
Examples (OCaml)

- Hello World

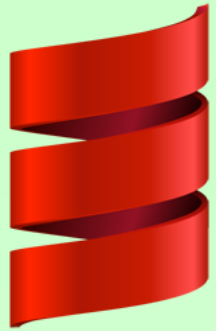
```
print_endline "Hello, World!"

print_endline :: String -> ()
```

- Increment method:

```
fun x -> x+1 end
or
let inc x = x+1
or
let inc (x:int) : int = x+1
```



Scala Programming Language



- stands for "scalable language" – building from reuseable components
- multi-paradigm language
- runs on standard Java and .NET platforms
- interoperates with all Java libraries
- Why study Scala?

concise, Java-compliant

Hello World in Scala

```
object HelloWorld extends App {
  println("Hello, World!")
}
```

Compilation:

```
scalac HelloWorld.scala
```

Execution:

```
scala HelloWorld
```

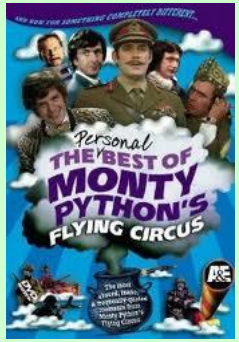
Increment Method

```
object XXX extends App {
  def inc (x:int) : int = x+1
}
```

```
(x:Int) => x+1
```

```
new Function1[Int, Int] {
    def apply(x: Int): Int = x + 1
}
```

Python Language



- powerful dynamic programming language
- clear readable syntax (indentation and off-side rule)
- Strong introspection capability
- high-level dynamic types
- excellent "battery-included" libraries
- Why study Python?

fast, nimble
good for scripting

Python Example

Hello World:

```
print `Hello, World!'
```

Increment method:

```
def inc(x):
      return x+1
```

Prolog

- one of first language based on first-order logic
- it is used to define “relations” and relies on unification for execution
- Popular in AI and database applications (via datalog)
- Why study Prolog?

relations, logic and unification

Prolog Example

Hello World:

```
main :- write('Hello, World!'),nl.
```

Increment method:

```
inc(X,Res):- Res is X+1.
```

How to Pass CS2104

Expressible in Prolog:

```
pass2104(X):-
        attend_lecture(X,2104),
        attend_tutorial(X,2104),
        do_assignment(X,2104),
        attempt_exam(X,2104).
```

Untyped Lambda Calculus

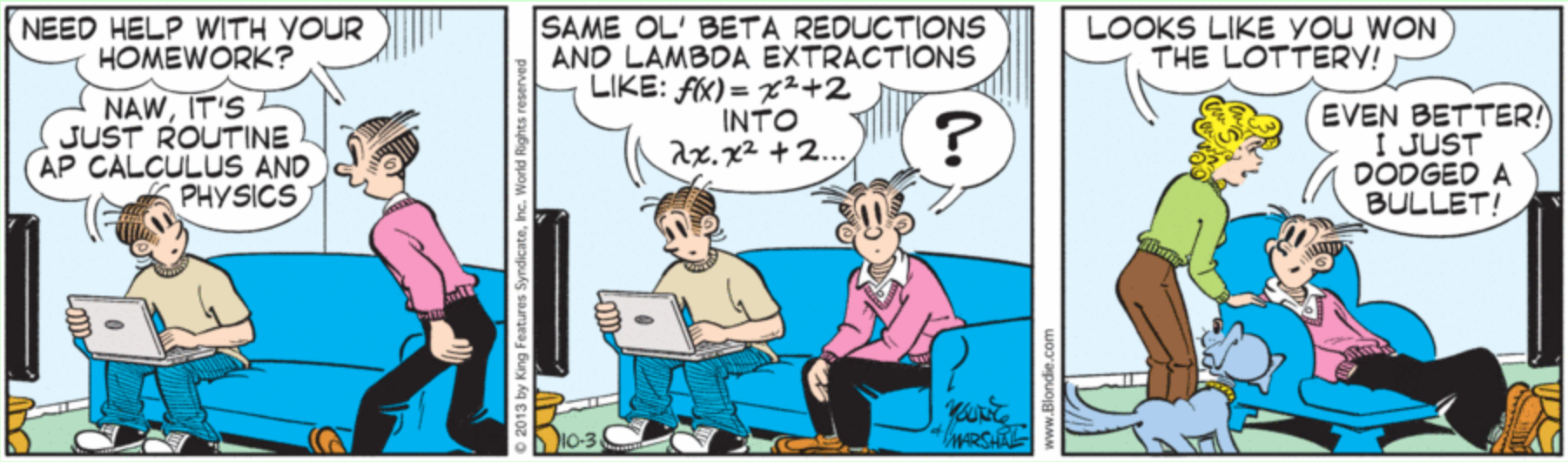
- Extremely simple programming language which captures *core* aspects of computation and yet allows programs to be treated as mathematical objects.
- Focused on *functions* and applications.
- Invented by Alonzo (1936,1941), used in programming (Lisp – 2nd oldest language) by John McCarthy (1959).
- Why is it significant?

Basis of Computability

Increment Method

With integer primitive, increment method can be written as:

$f\ x = x+1$ ➔ $(\lambda\ x\ .\ x+1)$



Syntax

In purest form (no constraints, no built-in operations), the lambda calculus has the following syntax.

t ::=		terms
	x	variable
	λ x . t	abstraction
	t t	application

This is simplest universal programming language!

How Expressible is Lambda Calculus?

- Very expressive!
  - *Boolean*
  - Integer
  - Functions
  - Recursion
  - Data structures
  - Loops!
  - It is Turing-complete