Principles of Programming Languages

Module M01: Course Information

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ac.in

January 05, 2022

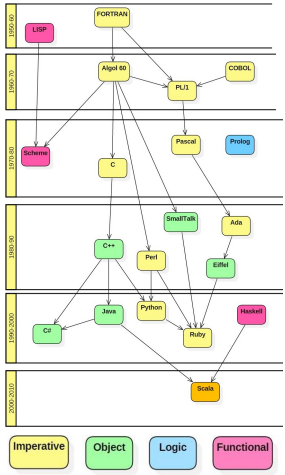
Table of Contents

Why PoPL?

Why PoPL? What do you expect from this course?

Why study *Principles of Programming Languages*?

History of Programming Languages



Paradigms: *Imperative*: Algorithms + Data, *Object*: Data, *Logic*: Facts + Rules + Queries, and *Functional*: Functions

- **FORTRAN**: IBM
- **LISP**: John McCarthy
- **Algol 60**: John Backus & Peter Naur
- **COBOL**: Grace Murray Hopper
- **PASCAL**: Niklaus Emil Wirth
- **Prolog**: Alain Colmerauer & Philippe Roussel
- **Scheme**: Guy L. Steele & Gerald Jay Sussman
- **C**: Brian W. Kernighan & Dennis M. Ritchie
- **SmallTalk**: Alan Kay, Dan Ingalls, & Adele Goldberg
- **Ada**: Jean Ichbiah & Tucker Taft
- **C++**: Bjarne Stroustrup
- **Objective-C**: Brad Cox
- **Perl**: Larry Wall
- **Java**: James Gosling
- **Python**: Guido van Rossum
- **Haskell**: Paul Hudak
- **C#**: Microsoft Corporation
- **Ruby**: Yukihiro Matsumoto
- **Scala**: Martin Odersky

Source: Programming Language Evolution

Why study *Principles of Programming Languages*?

TIOBE Index of Programming Languages

Dec 2019	Dec 2018	Change	Programming Language	Ratings	Change
1	1		Java	17.253%	+1.32%
2	2		C	16.086%	+1.80%
3	3		Python	10.308%	+1.93%
4	4		C++	6.196%	-1.37%
5	6	^	C#	4.801%	+1.35%
6	5	v	Visual Basic .NET	4.743%	-2.38%
7	7		JavaScript	2.090%	-0.97%
8	8		PHP	2.048%	-0.39%
9	9		SQL	1.843%	-0.34%
10	14	^^	Swift	1.490%	+0.27%
11	17	^^	Ruby	1.314%	+0.21%
12	11	v	Delphi/Object Pascal	1.280%	-0.12%
13	10	v	Objective-C	1.204%	-0.27%
14	12	v	Assembly language	1.067%	-0.30%
15	15		Go	0.995%	-0.19%
16	16		R	0.995%	-0.12%
17	13	vv	MATLAB	0.986%	-0.30%
18	25	^^	D	0.930%	+0.42%
19	19		Visual Basic	0.929%	-0.05%
20	18	v	Perl	0.899%	-0.11%

Why study *Principles of Programming Languages*?

TIOBE Index of Programming Languages

Jan 2022	Jan 2021	Change	Programming Language	Ratings	Change
1	3	↑	Python	13.58%	+1.86%
2	1	↓	C	12.44%	-4.94%
3	2	↓	Java	10.66%	-1.30%
4	4		C++	8.29%	+0.73%
5	5		C#	5.68%	+1.73%
6	6		Visual Basic	4.74%	+0.90%
7	7		JavaScript	2.09%	-0.11%
8	11	↑	Assembly language	1.85%	+0.21%
9	12	↑	SQL	1.80%	+0.19%
10	13	↑	Swift	1.41%	-0.02%
11	8	↓	PHP	1.40%	-0.60%
12	9	↓	R	1.25%	-0.65%
13	14	↑	Go	1.04%	-0.37%
14	19	⇈	Delphi/Object Pascal	0.99%	+0.20%
15	20	⇈	Classic Visual Basic	0.98%	+0.19%
16	16		MATLAB	0.96%	-0.19%
17	10	⇊	Groovy	0.94%	-0.90%
18	15	↓	Ruby	0.88%	-0.43%
19	30	⇈	Fortran	0.77%	+0.31%
20	17	⇊	Perl	0.71%	-0.31%

Why study *Principles of Programming Languages*?

- **Learning Widely-Applicable Design and Implementation Techniques**
  - Domain Abstractions ⇒ Programming Language Models / Features
  - Model of Programming Language ⇒ Design and Implementation of Abstraction
- **Creating New Domain Specific Languages or Virtual Machines**
  - Mathematica and MATLAB – manipulating mathematical formulas
  - Verilog and VHDL – describing computer hardware circuit designs
  - Cg (C for Graphics) – rendering algorithms that run directly on graphics hardware
  - LaTeX – typesetting, Flex and Bison – translators, e – h/w-s/w co-design etc.
- **Learning New Computational Models and Speeding Language Learning**
  - Knowledge of OOP (Java) expedites learning of C++ / C# / Python
  - Knowledge of Managed Resources (Java) expedites learning of C# / Python
  - Knowledge of Functional Programming (LISP) expedites learning MapReduce mechanism

Why Undergraduates Should Learn the Principles of Programming Languages? by ACM SIGPLAN (2011)

Why study *Principles of Programming Languages*?

- **Choosing the Right Language**
  - Most systems need several languages for different parts of the system
    - HTML for front-end rendering and Javascript for active front-end logic
    - Java for servlet (business layer) and JSP for server-end embedding
    - SQL for data manipulation
  - Nature of Application decides the suitable language
    - Systems Programming $\Rightarrow$ C++ (very high performance with complex behavior)
    - Embedded Programming $\Rightarrow$ C (very high performance with frugal dev tools)
    - Application Programming $\Rightarrow$ Java (medium performance with quick & robust app)
    - Web Programming $\Rightarrow$ Python (low performance with portability)

Why Undergraduates Should Learn the Principles of Programming Languages? by ACM SIGPLAN (2011)

Understanding Computation

- **Languages**:
  - Fortran, LISP, Algol, Cobol, APL, Simula, SNOBOL, BASIC, PL/1, B, Pascal, Forth, C, Smalltalk, Prolog, ML, Scheme, C++, Ada, Eiffel, Objective-C, Erlang, Perl, Tcl, Haskell, Python, Visual Basic, Ruby, R, Java, Javascript, PHP, D, C#, AspectJ, Visual Basic.NET, AspectC++, Scala, F#, Go
  - SQL
  - MATLAB
  - VHDL, Verilog, SystemC, e

  Unheard of, Aware, Can read programs, Can write programs, Have developed meaningful applications
- **Paradigms**:
  - Imperative / Procedural, Object-Oriented, Functional, Logic, Generic / Meta-Programming, Declarative, Concurrent / Parallel

  Unknown, Heard of, Vaguely understand, Wholly understand, Is master of

Understanding Computation

- **Computation Model**:
  - Turing Machine, Lambda Calculus, Predicate Calculus, Relational Calculus, Communicating Sequential Processes (CSP)

  Unknown, Heard of, Vaguely understand, Wholly understand, Is master of
- **Application Domains**:
  - System Applications, Business Applications, Web Applications, Embedded Applications, Engineering Applications, Graphics Applications

  Unfamiliar, Remotely familiar, Deeply familiar, Have developed meaningful applications
- **Language – Library Trade-off**:
  - (C++, pthread) & Java; (C++, list) & Python; (C, setjmp) & C++; (C++, SystemC) & e; (C, string) & Python;
- ...

Prerequisites

Prerequisites

[1] Programming
[2] Data Structure
[3] Algorithms
[4] Software Engineering
[5] Compilers
[6] Formal Languages and Automate Theory
[7] Theory of Computation (desirable)

Syllabus

Module 01: Course Information

[1] Module 01: Course Information
[2] Module 02: Syntax of λ Calculus
[3] Module 03: Semantics of λ Calculus
[4] Module 04: Typed λ Calculus
[5] Module 05: λ in Functional Programming Languages
[6] Module 06: λ in C++
[7] Module 07: Type Systems
[8] Module 08: Denotational Semantics
[9] Module 09: Imperative Languages

Refer: Syllabus of Principles of Programming Languages

Module 02: Syntax of λ Calculus

- Relations
- Functions
  - Compositions
  - Currying
- λ Calculus
  - Concept of λ
- λ Syntax
  - λ Expressions
    - Notation
  - Example
    - Simple
    - Composition
    - Boolean
    - Numerals
    - Recursion
    - Curried Functions
    - Higher Order Functions

Module 03: Semantics of λ Calculus

- Free and Bound Variables
- Substitution
- Reduction
  - α-Reduction
  - β-Reduction
  - η-Reduction
  - δ-Reduction
- Order of Evaluation
  - Normal and Applicative Order

Module 04: Typed λ Calculus

- $\Lambda^{\rightarrow}$
  - Type Expression
  - Pre-Expression & Expression
  - Type-checking Rules
    - Examples
- $\Lambda^{\rightarrow}_{rr}$
  - Types
    - Tuple Type
    - Record Type
    - Sum Type
    - Reference Type
    - Array Type
  - Type Expression
  - Pre-Expression
  - Type-checking Rules
    - Derived Rules

Module 05: λ in Functional Programming Languages

- Overview of Functional Programming
- Haskell
- Scheme
- Lisp

Module 06: λ in C++

- Functors
  - Callable Entities
  - Function Pointers
    - Replace Switch / IF
    - Statements
    - Late Binding
    - Virtual Function
    - Callback
    - Issues
  - Basic Functors
    - Elementary Example
    - Examples from STL
- λ in C++
  - λ Expression
  - Closure Object
  - Examples
    - Factorial
    - Fibonacci
    - Pipeline
  - Curry Function
- More on λ in C++

Module 07: Type Systems

- Type Systems
  - Type & Type Error
  - Type Safety
  - Type Checking
  - Type Inference
- Type Inference
  - `add x = 2 + x`
  - `apply (f, x)`
  - Inference Algorithm
    - Unification
- Examples
  - `sum`
  - `length`
  - `append`
  - Homework
- Type Deduction
  - Polymorphism
    - Ad-hoc
    - Parametric
    - Subtype
  - C++11,...

Module 08: Denotational Semantics

- Styles
- Syntax
- Domains
  - Domains
    - ▷ Product
    - ▷ Sum
  - Rat
- Algebra
  - Nat, Tr
  - String
  - Unit
  - Product Dom
  - Sum Dom
  - Lists
  - Function
  - Arrays
  - Lifted Domain
  - Recursive Function
- Denotational Definitions
  - Binary
  - Calculator

Module 09: Imperative Languages

- Imperative Languages
  - Lifted Domains
- Language + Assignment
- Programs are Functions
- Interactive File Editor
- Dynamically Typed Language (with IO)
- Recursive Definitions
- Language with
  - Contexts
  - Block Structured Language
  - Applicative Language
- Summary

Course Information

Course Material

- Slides will be uploaded to Moodle
- Books:
  - Programming Languages: Principles and Practices by Kenneth C. Louden and Kenneth A. Lambert (Cengage Learning)
  - Programming Language: Principles and Paradigms by Allen Tucker and Robert Noonan (McGraw-Hill Education)
  - Principles of Programming Languages: Design, Evaluation, and Implementation by Bruce J. MacLennan (Oxford University Press)
  - Concepts of Programming Languages by Robert W. Sebesta (Pearson)
  - Programming Language Pragmatics by Michael L. Scott (Morgan Kaufmann)
  - Compilers: Principles, Techniques, and Tools by A. V. Aho, Monica S Lam, R. Sethi, Jeffrey D. Ullman (Pearson / Addison-Wesley)
  - Books and Websites of various languages, computation models etc.

Refer: Syllabus of Principles of Programming Languages

About the Course: Interactions

- Timings: MON(10:00-11:00), WED(08:00-10:00). Slot C4
- Classes and interactions will be held on Microsoft Teams:
  - Link: Principles of Programming Languages 2022
  - Code: 2sb8kxx
- Kindly keep your microphone muted
- Kindly keep your video off
- Kindly put your comments / doubts on the chat – chats will be periodically checked and responded
- Kindly raise your hand to ask a question
- Deeper interactions / feedback will be over Forum on Moodle
- Interaction Outside Class: By appointment through mail - over audio / video chat

About the Course: Evaluations

- Assignments
  - In-Class Assignments
    - Marks: 5~10 each
    - Time: 15-30 minutes. Completion within the class.
  - Offline Assignments
    - Marks: 10~20 each
    - Time: 1-2 weeks.
  - Total Marks: 70. Total of the assignments will be scaled to 70
  - To be hand-written, scanned and uploaded - write clearly using bigger font styles
- Online Test
  - Marks: 15. Time: 1 hour. # of Test: 3. Best 2 of 3
  - Total Marks: 30
- Relative Grading
  - Marks of assignments and tests will be added to get to total out of 100
  - Grade boundary will be decided relatively based on the bell curve

The Coordinating Platforms

- Moodle will be used for the course. Register on Moodle immediately to:
  - CS40032: Principles of Programming Languages
  - Course Key: POPL22STU
- All assignments / presentations / material will be uploaded to Moodle
- Online texts will be conducted on Moodle
- The submissions will be accepted *only* through Moodle up to the specified deadline. No submission through mail will be entertained

The Coordinating Platforms

- Extensions permissible only on medical ground (B C Roy certificate) and IIT duty (like inter-IIT Sports meet on Dean's Order)
- 10% to 50% penalty (depending on assignment and amount of delay) on late submission on discretionary basis
- Zero tolerance to plagiarized submissions. Penalty applies to both parties
- Class Tests will be held online in Moodle
- All announcements will be made on Moodle. Keep checking
- ERP will also be used at times for communication. Make sure that your registered email at ERP works
- Recording of class lectures will be posted on YouTube

Schedule for Tests

Test	Date	Time
Test 1	02-Feb-22	8:15–9:45
Test 2	23-Feb-22	8:15–9:45
Test 3	13-Apr-22	8:15–9:45

TA and Teachers

Sr. No.	Name	Mobile / WhatsApp	Gmail Id	Institute Id
1	Soumen Paul	7980054589	soumenpaul165@gmail.com	soumenpaul2019@iitkgp.ac.in
2	Abhishek Kumar	7018763100	merealone2516@gmail.com	ABHISHEK16@KGPIAN.IITKGP.AC.IN
3	Kshitiz Sharma	8768117888	kshitizs2809@gmail.com	kshitizs2809@iitkgp.ac.in
4	Apoorve Singhal	8003115061	sapoorve@gmail.com	SAPOORVE@IITKGP.AC.IN
5	Rohit	9304016011	rohitsanjay64@gmail.com	rohit@iitkgp.ac.in
6	Partha Pratim Das	9830030880	partha.p.das@gmail.com	ppd@cse.iitkgp.ac.in

Prefer to contact by email. Use mobile call only for extreme urgency