CS40032: Principles of Programming Languages
Module 01: Course Information

Partha Pratim Das

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

ppd@cse.iitkgp.ac.in

Jan 04, 2021

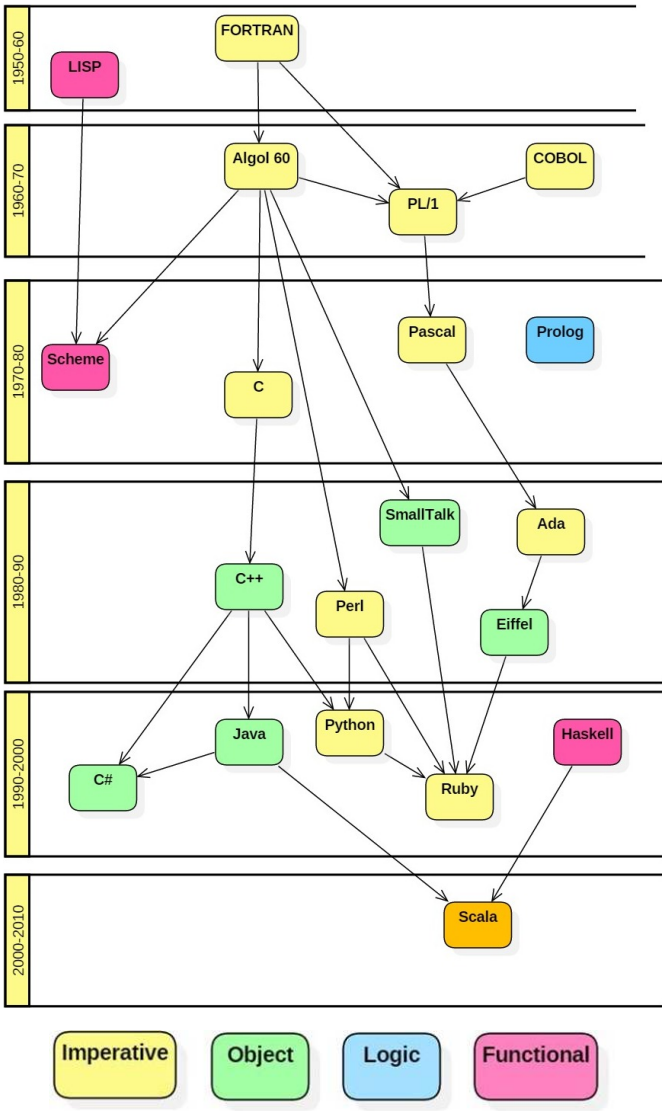
Table of Contents

Expectations

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: http://www.cs.sjsu.edu/ pearce/modules/lectures/languages2/concepts/evolution.htm

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 2021	Jan 2020	Change	Programming Language	Ratings	Change
1	2	↑	C	17.38%	+1.61%
2	1	↓	Java	11.96%	-4.93%
3	3		Python	11.72%	+2.01%
4	4		C++	7.56%	+1.99%
5	5		C#	3.95%	-1.40%
6	6		Visual Basic	3.84%	-1.44%
7	7		JavaScript	2.20%	-0.25%
8	8		PHP	1.99%	-0.41%
9	18	⇈	R	1.90%	+1.10%
10	23	⇈	Groovy	1.84%	+1.23%
11	15	⇈	Assembly language	1.64%	+0.76%
12	10	↓	SQL	1.61%	+0.10%
13	9	⇊	Swift	1.43%	-0.36%
14	14		Go	1.41%	+0.51%
15	11	⇊	Ruby	1.30%	+0.24%
16	**20**	⇈	**MATLAB**	**1.15%**	**+0.41%**
17	19	↑	Perl	1.02%	+0.27%
18	13	⇊	Objective-C	1.00%	+0.07%
19	12	⇊	Delphi/Object Pascal	0.79%	-0.20%
20	16	⇊	Classic Visual Basic	0.79%	-0.04%

Why study *Principles of Programming Languages*?

1. **Learning Widely-Applicable Design and Implementation Techniques**
   - Domain Abstractions ⇒ Programming Language Models / Features
   - Model of Programming Language ⇒ Design and Implementation of Abstraction

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

Why study *Principles of Programming Languages*?

1. **Learning Widely-Applicable Design and Implementation Techniques**
   - Domain Abstractions ⇒ Programming Language Models / Features
   - Model of Programming Language ⇒ Design and Implementation of Abstraction
2. **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.

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

Why study *Principles of Programming Languages*?

1. **Learning Widely-Applicable Design and Implementation Techniques**
   - Domain Abstractions ⇒ Programming Language Models / Features
   - Model of Programming Language ⇒ Design and Implementation of Abstraction
2. **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.
3. **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*?

1. **Learning Widely-Applicable Design and Implementation Techniques**
   - Domain Abstractions ⇒ Programming Language Models / Features
   - Model of Programming Language ⇒ Design and Implementation of Abstraction
2. **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.
3. **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
4. **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 ⇒ C++ (very high performance with complex behavior)
     - Embedded Programming ⇒ C (very high performance with frugal dev tools)
     - Application Programming ⇒ Java (medium performance with quick & robust app)
     - Web Programming ⇒ Python (low performance with portability)

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

Understanding Computation

1. **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
2. **Paradigms**:
   – Imperative / Procedural, Object-Oriented, Functional, Logic, Generic / Meta-Programming, Declarative, Concurrent / Parallel
   - Unknown, Heard of, Vaguely understand, Wholly understand, Is master of
3. **Computation Model**:
   – Turing Machine, Lambda Calculus, Predicate Calculus, Relational Calculus, Communicating Sequential Processes (CSP)
   - Unknown, Heard of, Vaguely understand, Wholly understand, Is master of
4. **Application Domains**:
   – System Applications, Business Applications, Web Applications, Embedded Applications, Engineering Applications, Graphics Applications
   - Unfamiliar, Remotely familiar, Deeply familiar, Have developed meaningful applications
5. **Language – Library Trade-off**: (C++, pthread) & Java; (C++, list) & Python; (C, setjmp) & C++; (C++, SystemC) & e; (C, string) & Python;
6. ...

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 Modules

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

Module 02: λ Calculus – Syntax

1. Relations
2. Functions
   1. Compositions
   2. Currying
3. λ Calculus
   1. Concept of λ
4. λ Syntax
   1. λ Expressions
      1. Notation
   2. Example
      1. Simple
      2. Composition
      3. Boolean
      4. Numerals
      5. Recursion
      6. Curried Functions
      7. Higher Order Functions

Module 03: λ Calculus – Semantics

1. Free and Bound Variables
2. Substitution
3. Reduction
   1. α-Reduction
   2. β-Reduction
   3. η-Reduction
   4. δ-Reduction
4. Order of Evaluation
   1. Normal and Applicative Order

Module 04: Typed λ Calculus

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

Module 05: λ in C++

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

Module 06: Type Systems

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

Module 07: Denotational Semantics

1. Styles
2. Syntax
3. Domains
    1. Domains
        1. Product
        2. Sum
    2. Rat
4. Algebra
    1. Nat, Tr
    2. String
    3. Unit
    4. Product Dom
    5. Sum Dom
    6. Lists
    7. Function
    8. Arrays
    9. Lifted Domain
    10. Recursive Function
5. Denotational Definitions
    1. Binary
    2. Calculator

Module 08: Imperative Languages

1. Imperative Languages
   1. Lifted Domains
2. Language + Assignment
3. Programs are Functions
4. Interactive File Editor
5. Dynamically Typed Language (with IO)
6. Recursive Definitions
7. Language with
   1. Contexts
   2. Block Structured Language
   3. Applicative Language
8. Summary

Module 09: λ Calculus – Languages

1. Overview of Functional Programming
2. Haskell
3. Scheme
4. Lisp

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.

About the Course: Interactions

- Timings: MON(10:00-11:00), WED(08:00-10:00)
- Classes and interactions will be held on Microsoft Teams: POPL CS40032
- 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

- Offline Assignments
  - Marks: 10~20
  - # of Assignments: 6
  - Total Marks: 70
  - Total of the 6 assignments will be scaled to 70
  - To be hand-written, scanned and uploaded - write clearly preferably using a little bigger font styles
- Online Quiz
  - Marks: 15
  - Time: 1 hour
  - # of Test: 3
  - Best 2 of 3
  - Total Marks: 30
- Relative Grading
  - Marks of assignments and quizzes 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: STUPOPL
- All assignments / presentations / material will be uploaded to Moodle
- The submissions will be accepted *only* through Moodle up to the specified deadline. No submission through mail will be entertained
- 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
- Dates for Class Tests will be declared by next week
- 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: 2021_H1 PoPL Lectures

Tentative Schedule for Quiz

Quiz	Date	Time
Quiz 1	27-Jan-21	8:15–9:45
Quiz 2	24-Feb-21	8:15–9:45
Quiz 3	31-Mar-21	8:15–9:45

TA and Teachers

Sr. No.	Name	Mobile / WhatsApp	Gmail Id	Institute Id
1	Srijoni Majumdar	9674474267	majumdar.srijoni@gmail.com	srijoni.majumdar@iitkgp.ac.in
2	Soumen Paul	7980054589	soumenpaul165@gmail.com	soumenpaul2019@iitkgp.ac.in
3	Megha Bansal	9926954320	bansalmegha98@gmail.com	bansalmegha98@iitkgp.ac.in
4	Priti Shekhawat	9426991072	shekhawatpriti8@gmail.com	pritishekhawat@iitkgp.ac.in
5	Arnab Sinha	7602713137	arnab.sinha77@gmail.com	arnab.sinha77@iitkgp.ac.in
6	Anirban Saha	8961670359	anirban.jis2013@gmail.com	anirbansaha@iitkgp.ac.in
7	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