Teaching with PLCC

Stoney Jackson
CCSCNE 2025

Outline

- CS 351 - Programming Languages (at Western New England)
  - Schedule
  - Topics
- How to adopt PLCC for your course
  - Resources
  - Development environment
- The Future

CS 351 - Programming Languages - Overview

- Required for CS majors
- Fall Junior Year
- 15-week semester
- Two 80m sessions per week
- 20-30 students
- Curricular Context
  - CS 101 - Intro to Computing (Breadth)
  - CS 102 - Intro to Programming (Python)
  - CS 200 - Data Structures (Python)
  - CS 210 - Software Design (Java)
  - CS 220 - Software Development (Git, GitLab, Codespaces, Agile, Scrum, TDD, Licensing)

CS 351 - Programming Languages - Schedule

- Weeks 1-2: Specifying a language in PLCC
- Weeks 3-6: Build a functional language
- Weeks 7-9: Call semantics
- Weeks 10-11: Static type checking
- Weeks 12-15: Build an object-oriented language

Weeks 1-2: Specifying a Language in PLCC

- PLCC as a toolset
- **Lexical analysis** - tokens, regex, lexemes, scanner, first-longest match rule
- **Syntactic analysis** - BNF, left-most-derivation, top-down recursive-descent parser, mapping between rules and generated Java classes, AST construction
- **Semantic analysis** - walking ASTs with small methods and dynamic dispatch

Weeks 3-6: Build a Functional Language (V0 - V6)

```
letrec
  fact = proc(x)
    if zero?(x) then 1
    else *(x, .fact(sub1(x)))
in
  .fact(2)
```

- Incrementally build anguages V0-V6
- Types
  - Integers (0 is false)
- Primitives
  - Arithmetic
  - Comparison
  - Relational
- Scoping
  - Environments
  - Binding values to symbols
  - Look up symbols to get values
  - Static vs dynamic
  - Shadowing
- Functions
  - Higher-order functions
  - Anonymous functions
  - Closures
  - Defining and calling
- Recursion

Weeks 7-9: Call Semantics

- SET - pass-by-value (adds side effects)
- REF - pass-by-reference
- NAME - pass-by-name
- NEED - pass-by-need
- CICO - pass-by-copy-in-copy-out (in homework)

- Challenges with side-effects
  - Aliasing
  - Order of operations

```
let
  x = 3
  inc = proc(y) set y = add1(y)
in
  let
    result = . inc(x)
  in
    x


% Value semantics => 3
% Reference semantics => 4
```

Weeks 10-11: Static Type Checking (TYPE0 and TYPE1)

```
let
  twice =
  proc(f: [int => int], x: int): int
    .f(.f(x))
  add5 = proc(x: int): int  +(5, x)
in
  .twice(add5, 10)

% .add5(.add5(10)) => 20
```

- Type errors
- Strong vs weak type systems
- Static vs dynamic type systems
- Type annotations
- Type inference
- Type of a function
- Type checking through type evaluation

Weeks 12-15: Build an Object-Oriented Language (OBJ)

```
define shape = class
    method area = proc() -1    % default behavior
end

define rectangle = class extends shape
    field len    % length
    field wid    % width

    method init = proc(len,wid) {
        set <self>len=len ; set <self>wid=wid ; self
    }
    method area = proc() *(len,wid)
end

define s = new shape
define r = .<new rectangle>init(4,5)
.<r>area()     % => 20
.<s>area()    % => -1
```

- First-class classes
- Objects
- Static fields and methods
- Instance fields and methods
- Constructors
- Instantiation
- Dereference operator
- Object-construction
- Inheritance
- Scoping and shadowing
- Navigating scopes with `super`, `superclass`
- Dynamic dispatch and polymorphism
- Deep vs shallow binding

Additional Possibilities with PLCC

- Logic Paradigm (ABC)
- Properties (PROP)
- Continuations (CONT)
  - Exception handling
  - Concurrency
  - ...
- Data structures
  - Lists
  - Arrays
  - Trees
  - ...
- ...

Outline

- CS 351 - Programming Languages (at Western New England)
  - Schedule
  - Topics
- Adopting PLCC for Your Course
  - Resources
  - Work environments
- Future

PLCC Resources (shown again at the end)

- https://github.com/ourPLCC/Course-Template-2024
  - Template to jump start your course.
- https://discord.gg/EVtNSxS9E2 :: **Join us!**
  - Discord server
  - Get help with PLCC and your course
  - Join us for weekly meetings
    Currently Friday's at 11a ET
- https://github.com/ourPLCC/plcc
  - Installation instructions
  - A one-long-page PLCC manual
  - Links to other resources (like the ones below)
- https://github.com/ourPLCC/course
  - Text: PDFs+source of Tim Fossum's notes/slides
  - Slides: Jim Heliotis' powerpoint slides
  - Example assignments
  - Link to Stoney's course offerings on GitLab
- https://github.com/ourplcc/languages
  - Extensive set of example languages
  - Including all the languages used in courses
- https://plcc.pithon.net/
  - Where it all began!
  - PLCC and resource maintained by Tim

Work Environments

- Option 1: GitHub Codespaces
- Option 2: Dev Container
- Option 3: Docker Container
- Option 4: Native Machine

Option 1: GitHub Codespaces

Requirements

- GitHub account
- Browser

Advantages

- No "installation" for students
- Consistent development environment
- There are many ways to deliver a course using GitHub

Disadvantages

- Students need knowledge in Git/GitHub
- Faculty needs knowledge in Git/GitHub
- Faculty performs installation
- Many ways to deliver a course using GitHub

Example Workflow

- Instructor creates a GitHub repo w/ content
- Instructor copies .devcontainer/ into repo
- Instructor marks repository as a template
- Student creates repo from template
- Student grants instructor privileges
- Student works on repo in Codespace
- Student pushes changes to repo using git
- Instructor reviews work in students repo

Could use with GitHub Education

Option 2: Dev Containers

Requirements

- VS Code w/ Dev Container extension
- Docker Desktop
- Git (optional)

Advantages

- Common development environment
- Can use native tools
- Git and GitHub are optional

Disadvantages

- More to install
- Platform variances
- Docker requires 8GB RAM for reasonable use
- Docker is usually easy to install, but occasionally not

Example Workflow

- Instructor creates a folder of content
- Instructor copies .devcontainer/ into folder
- Instructor gives folder to students
- Student opens folder in VS Code
- VS Code starts a Dev Container
- Student works on files in VS Code
- When done, return folder to instructor

Option 3: Docker Container

Requirements

- Docker Desktop

Advantages

- One dependency
- Can use native tools

Disadvantages

- Platform variances
- Docker requires 8GB RAM for reasonable use
- Docker is usually easy to install, but occasionally not

Example Workflow

- Instructor send students a folder
- Student runs Docker mounting folder
- Student uses PLCC in container
- Student may use native tools on files
- Student returns the folder to instructor

Option 4: Native Machine

Requirements

- Linux. MacOS, or WSL on Windows
- Python >= 3.9
- Java >= 11

Advantages

- Installer has full control of installation

Disadvantages

- Maximum variance of development environments across students and between instructor and students

Example Workflow

- Instructor creates a folder of content
- Instructor gives students folder
- Students work on folder using native tools
- Student returns folder to instructor

Future

- PLCC-ng: https://github.com/ourPLCC/plcc-ng
  - Targeting Python for semantics
  - Allow PLCC to be easily retargeted to different languages for semantics (e.g., Go, Rust, JavaScript, etc.)
- VS Code syntax highlighting
- Interactive textbook à la Runestone.Academy

Got ideas? Want to help?

Let us know on Discord: https://discord.gg/EVtNSxS9E2

PLCC Resources

- https://github.com/ourPLCC/Course-Template-2024
  - Template to jump start your course.
- https://discord.gg/EVtNSxS9E2 :: **Join us!**
  - Discord server
  - Get help with PLCC and your course
  - Join us for weekly meetings
    Currently Friday's at 11a ET
- https://github.com/ourPLCC/plcc
  - Installation instructions
  - A one-long-page PLCC manual
  - Links to other resources (like the ones below)
- https://github.com/ourPLCC/course
  - Text: PDFs+source of Tim Fossum's notes/slides
  - Slides: Jim Heliotis' powerpoint slides
  - Example assignments
  - Link to Stoney's course offerings on GitLab
- https://github.com/ourplcc/languages
  - Extensive set of example languages
  - Including all the languages used in courses
- https://plcc.pithon.net/
  - Where it all began!
  - PLCC and resource maintained by Tim