LECTURE 1

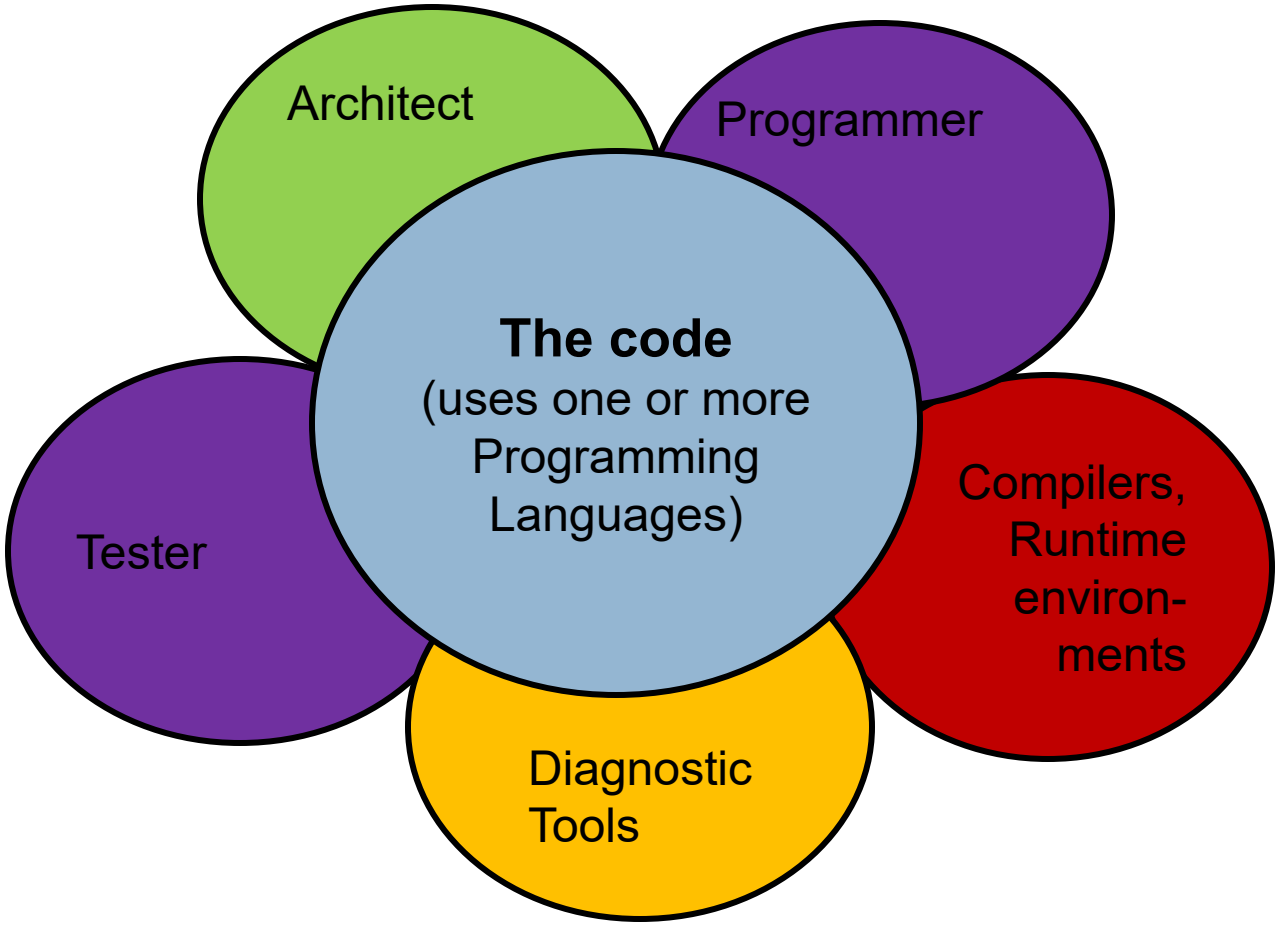
Instructors: Crista Lopes

Outline

- Course objectives
- The course

Credits:
This presentation uses material from
https://courseware.stanford.edu/pg/courses/lectures/96023

Where the rubber hits the road



Programming Languages

- Universe of design ideas
  - Some get baked into languages, others are in libraries, idioms or **programming styles**
- Language design concepts often pop out into systems design concepts
  - E.g. Map-Reduce, stateless – REST, dataflow, …

Programming Languages vs. Programming Styles

- Styles are more generic than languages
  - Languages are what happens when styles are enforced
- Examples:
  - OOP in C or Scheme
  - Actors in Java
- Important to know these foundational concepts independent of languages

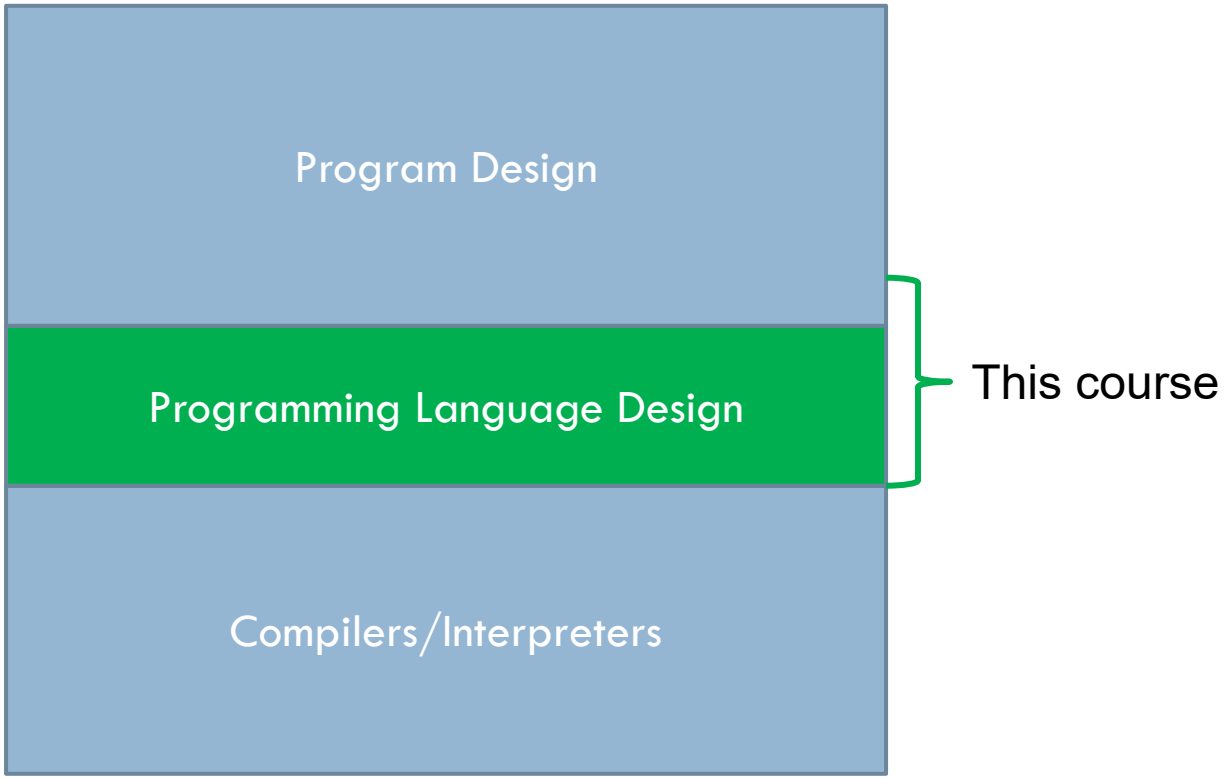
Style [Programming, Architectural]

- Constraints
  - Things you **must not** do
- Principled decisions
  - Things **you must** do

Programming Languages History

- Most concepts were invented in the 40s through 70s
  - A few in the 80s, even less in the 90s
  - Lots of detail work, combinations and optimizations
- Some concepts get baked into PLs, others are just floating around
- People keep reinventing things, when the context is right
- "Know the past to understand the present"

Where this course stands

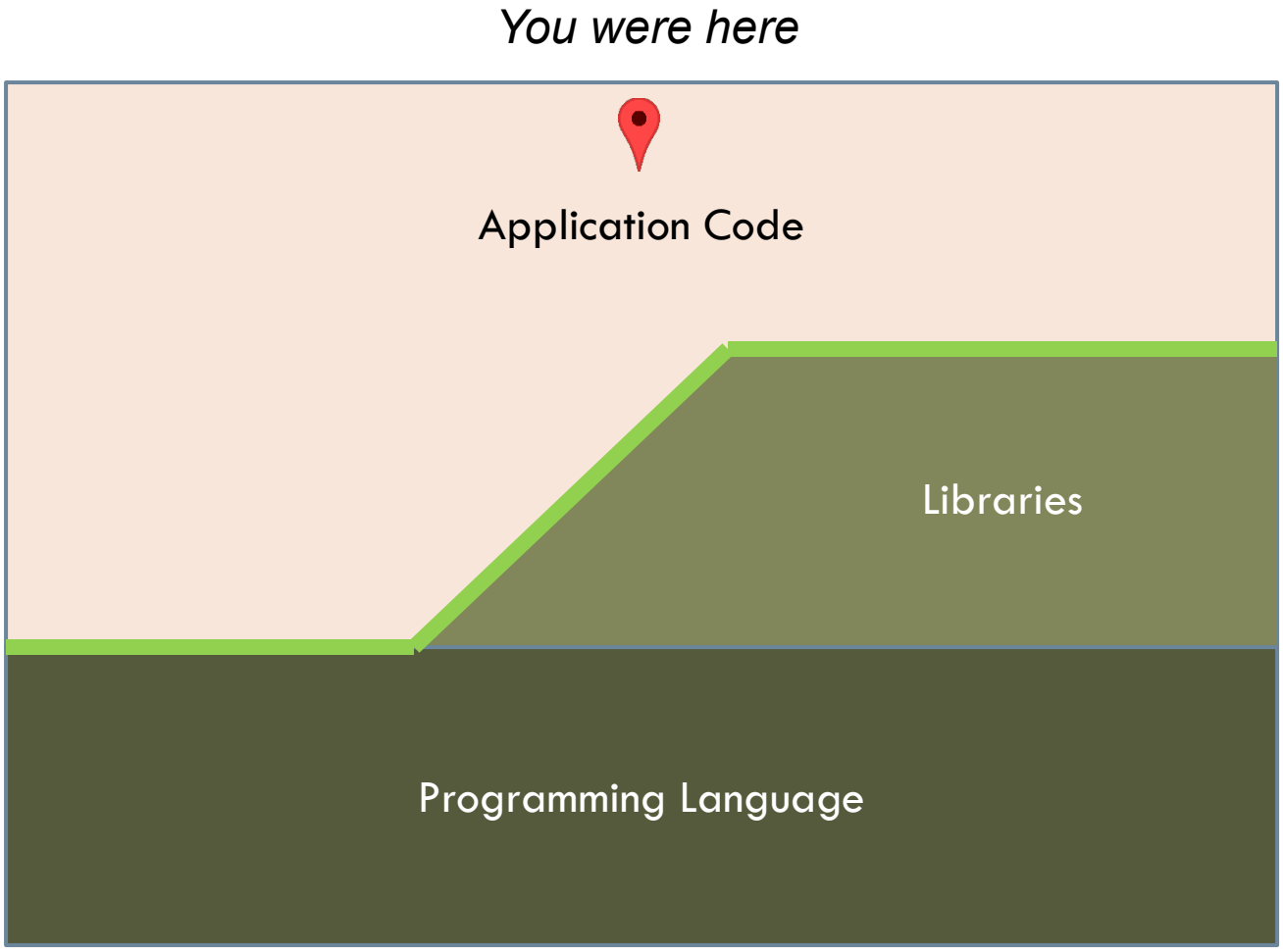


We will cover how PLs influence program design and vice-versa

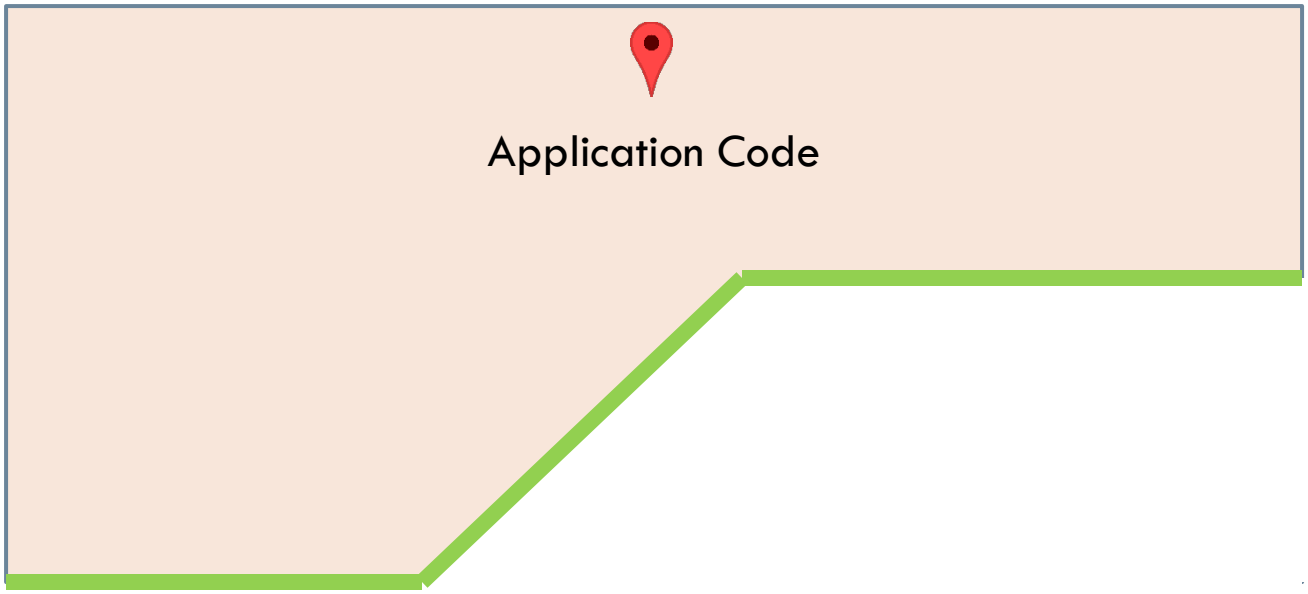
Not covered: implementation of PLs. Recommended:

- CS 241 Advanced Compiler Construction
- EECS 221 Program Analysis

Map of the programming territory



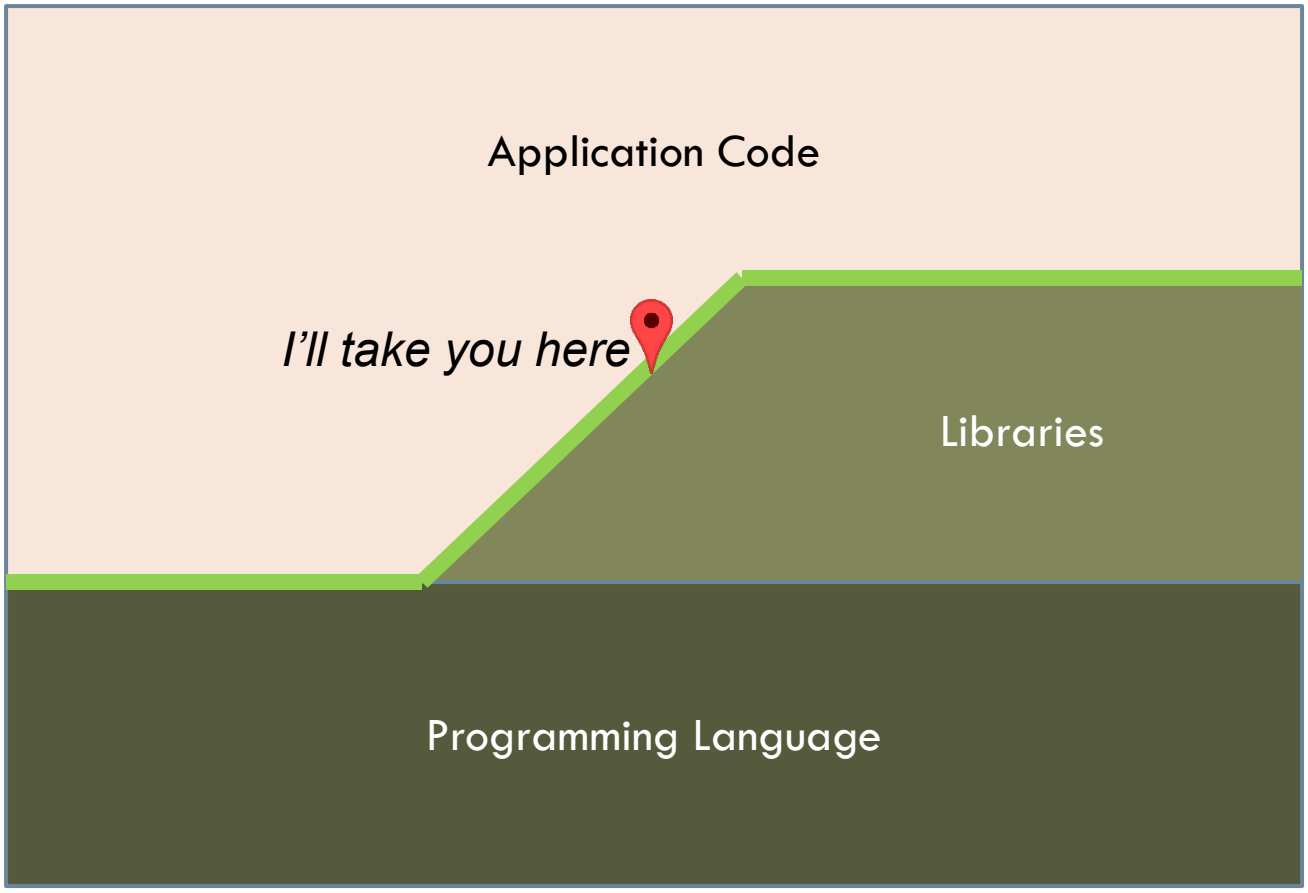
Map of the programming territory



Characteristics of app-land code:

- Passive use of familiar concepts
- Focus on functional correctness
- Blindness wrt abstraction opportunities
- Obliviousness wrt *overall* performance, extensibility, evolution,...

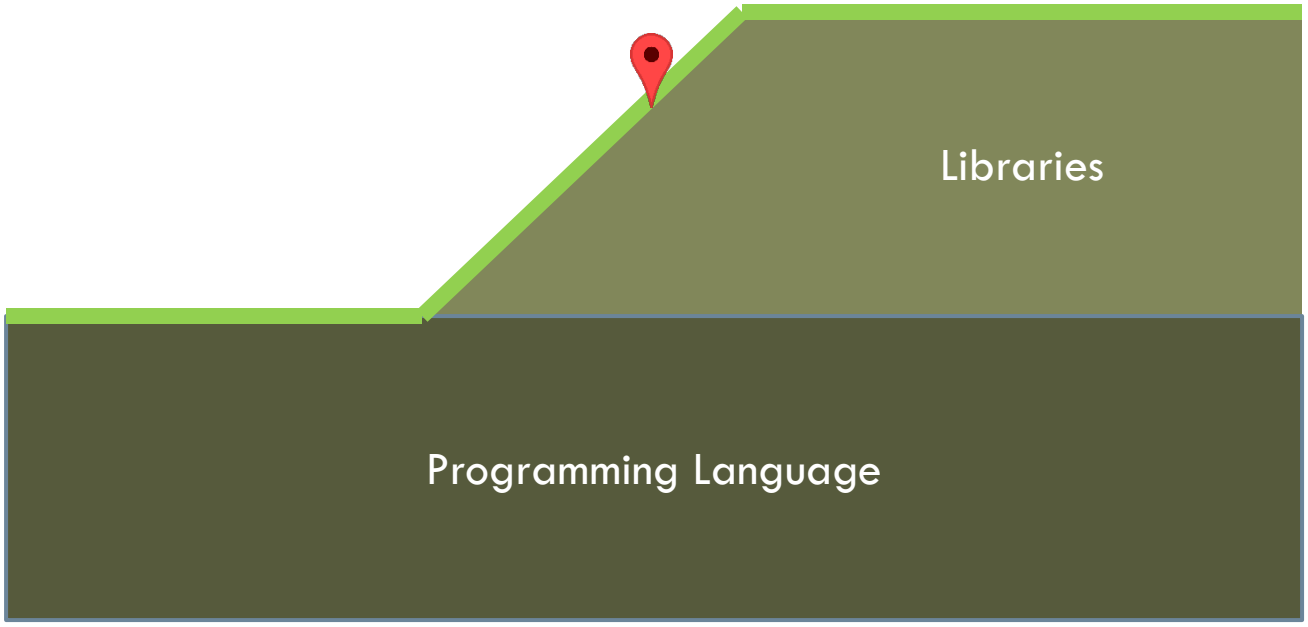
Map of the programming territory



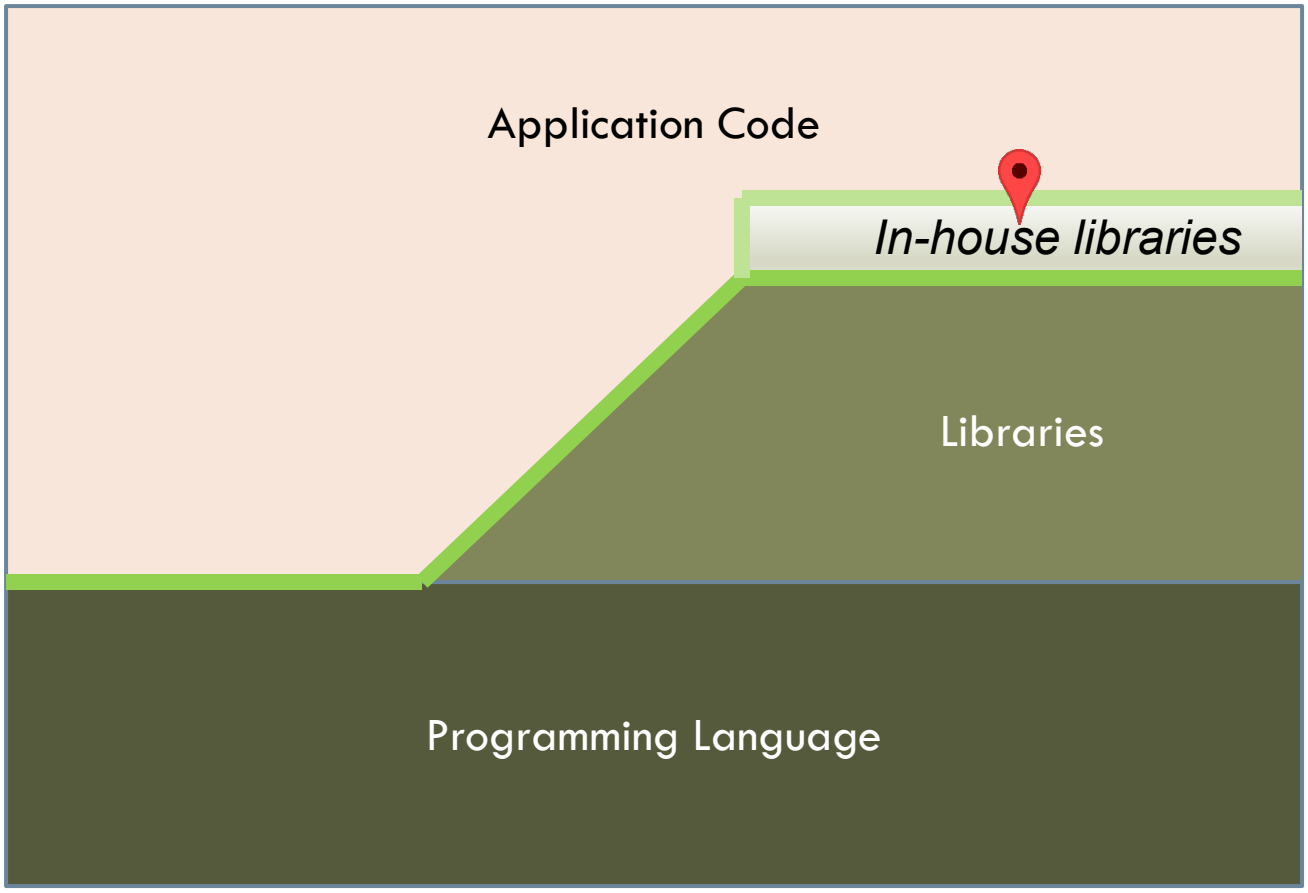
Map of the programming territory

Characteristics of lib-land code:

- Creation of concepts for others to use
- Focus on the interface (what programmers see, how they use it)
- Takes advantage of abstraction opportunities
- Cares about *overall* performance, extensibility, evolution, ...



Map of the programming territory



Objectives of the course

- Understand many advanced programming concepts
  - 100's of PLs, all *look* different ➔ they aren't that different
  - Appreciate history, diversity of ideas
  - Be prepared for new languages
  - See beyond hype & sales pitches
- Learn some important facts about existing language systems and techniques
- Learn and think critically about design tradeoffs

Learning Outcomes

- Level up your programming skills by several notches

- Code for your **peers**
  - Application programmer → library and framework programmer