Programming Languages

Outline
- Questions re/ Programming Languages
- History of Programming Languages
- Traits of a Good Programming Language
- Programming and Operating Environment
- Four Language Paradigms

Language Family Questions
- Often work at job with 1-2 languages.
- Why is C like FORTRAN like Pascal not like LISP not like Java? What are the characteristics that they share? Differ?
- If we can describe the characteristics that make a family of languages similar, then we can come up with a modeling language to represent the characteristics.

History
- 1950s:
  - FORmula TRANslator
    • FORTRAN
  - International Algorithmic Language
    • IAL, became Algol
  - Common Business Oriented Language
    • COBOL
  - LIST Processing Language
    • Lisp
History

- **1970s:**
  - Ada
  - C
  - Pascal
  - Prolog
  - Smalltalk

- **1980s:**
  - C++

- **1990s:**
  - HTML
  - Java

Programming Language Families

**Common Form for Discussion**
- Type
- Traits
- General Form
- Example Program
- Languages in Family
Programming Language Families
Type: Procedural or Imperative

- **Traits**
  - Command-driven or statement-oriented
  - Basic concept is machine state
  - Often, imperative languages are first view of programming

- **General Form**
  
```plaintext
statement1;
statement2
```

- **Example Program**
  
```plaintext
sum = 0; count = 0;
for i = 1,n
  {sum = sum + array[i];
   count = count + 1}
average = sum/count;
```

- **Example Languages**
  - FORTRAN
  - C
  - Pascal

Programming Language Families
Type: Functional or Applicative

- **Traits**
  - look at desired result rather than available data
  - Program development proceeds by developing functions from previously developed functions

- **General Form**
  
```plaintext
funcn1(funcn2(funcn1(data))..)
```

- **Example Program**
  
```plaintext
divide(sum(data),count(data))
```

- **Example Languages**
  - LiSP

Programming Language Families
Type: Logic or Rule-Based

- **Traits**
  - Check for presence of enabling cond., when satisfied execute appropriate action
  - Execution is not necessarily sequential, but is based upon enabling conditions

- **General Form**
  
```plaintext
enabling condition1 ⇒ action1
enabling condition2 ⇒ action2
```

- **Example Program**
  
```plaintext
sum_avail and count_avail ⇒ avg = sum/count;
data_avail ⇒ sum(data), sum_avail = T,
count(data), count_avail = T;
```

- **Example Languages**
  - Prolog
### Programming Language Families

**Type:** Object Oriented

- **Traits**
  - Design complex data objects, describe limited functionality to operate on data
  - Complexity obtained by extending (inheriting) traits of simpler objects
  - Close to human perception and problem domain

- **General Form**
  - **Class Name**
  - **Attributes**
  - **Operations**

- **Example Program**
  - **Class Name:** `set_of_numbers`
  - **Attributes**
    - `size`: integer
  - **Operations**
    - `find_avg()`: real

- **Example Languages**
  - C++
  - Java
  - Smalltalk