Computer Science Principles

Unit 3: Algorithms and Programming

LECTURE 4: APP LAB (U3)

DR. ERIC CHOU

IEEE SENIOR MEMBER

Objectives

- This Big Idea covers the vast majority of the code you'll see on the AP test in the spring. It describes basic components of most programming languages such as variables, lists, and procedures.
- Unlike AP Comp Sci A, which only teaches Java, there's no programming language specification for AP CSP. Your teacher could use a block-based language like Scratch or a text-based language like Python. In order to accommodate for these differences, The AP CSP test uses a basic **Pseudocode**, or a simplified programming language.
- The College Board's Pseudocode shares many similarities with the coding language Python, which is used to help write examples across this guide.
- All photos of Pseudocode come from the Exam Reference Sheet on page 214 of the CED, found here.

Unit Overview

Exam Weighing:

- 30-35% of the AP Exam
- Practically, this translates to a good portion of the questions on the test. This unit also makes up the bulk of your final Create project. It's a big part of this course.

Introduction to App Lab

LESSON 1 [CODE.ORG]

Introduction to Design Mode

LESSON 2 [CODE.ORG]

Project - Designing an App Part 1/2

LESSON 3/4 [CODE.ORG]

The Need for Programming Languages

LECTURE 1/LESSON 5 [CODE.ORG]

Objectives

- The main purpose of this lesson is to demonstrate why programming languages are necessary, and how they come into being.

Vocabulary

- **Algorithm** - A precise sequence of instructions for processes that can be executed by a computer
- **Programming Language** - derived from the human need to concisely give instructions to a machine
- **Ambiguity** - refer either to something (such as a word) which has multiple meanings, or to a more general state of uncertainty.

Programming

- One of the most important things you can do in programming starts well before you write any code.

It's about how you think.

- "Everybody should learn how to program a computer . . ., because it teaches you to think." -Steve Jobs

Why do we have programming languages? What is a language for?

1. way of thinking -- way of expressing algorithms
2. languages from the user's point of view
3. abstraction of virtual machine -- way of specifying what you want
4. the hardware to do without getting down into the bits
5. languages from the implementor's point of view

The Art of Language Design

Evolution	Ease of Implementation
Special Purposes	Standardization
Personal Preference	Open Source
Expressive Power	Excellent Compiler
Ease of Use for Novice	Economics/Patronage/Inertia

Modern Real-World Languages

XML HTML CSS

Markup/Data Languages

Perl Ruby A Programmer's Best Friend

php python

Database Languages

Network (Web-server) Languages

OBJ C

Mobile (App) Languages

R MATLAB

Number and Data Processing Languages

Desktop (.exe) Languages

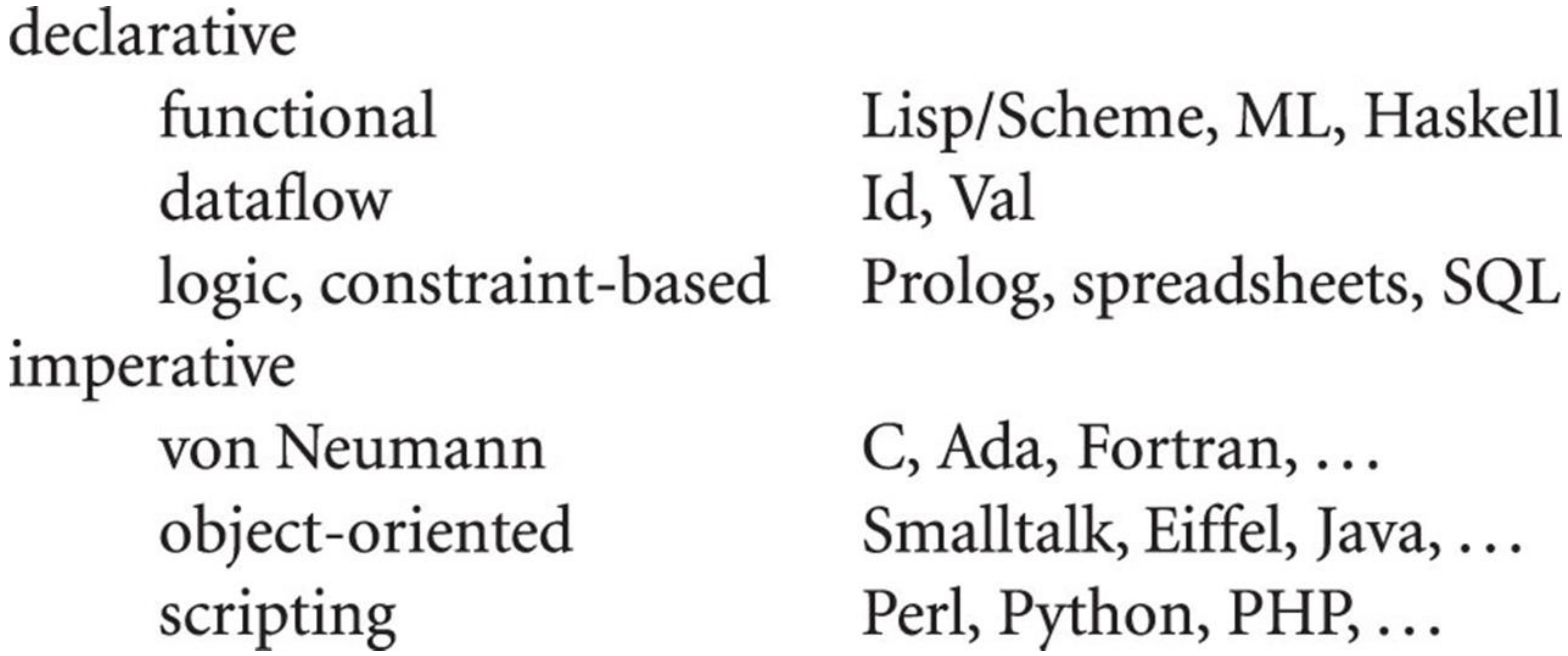
VHDL Verilog SYSTEM

Hardware Description Languages

Edif HSPICE®

Electronics Languages

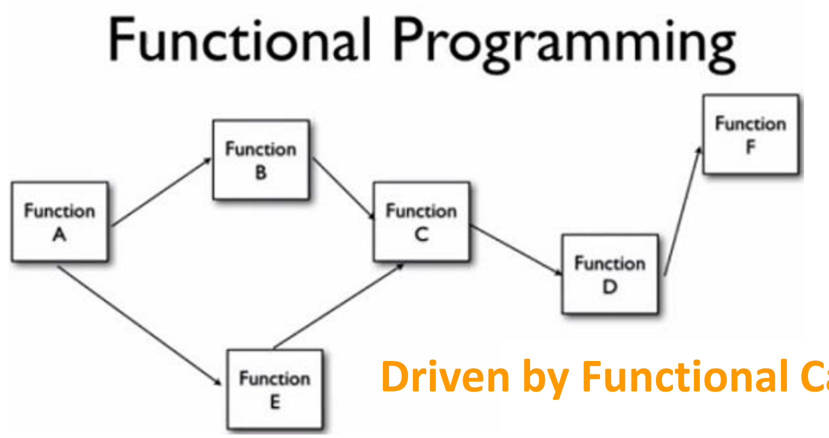
Computer Scientist Group Language as …



declarative	
functional	Lisp/Scheme, ML, Haskell
dataflow	Id, Val
logic, constraint-based	Prolog, spreadsheets, SQL
imperative	
von Neumann	C, Ada, Fortran, …
object-oriented	Smalltalk, Eiffel, Java, …
scripting	Perl, Python, PHP, …

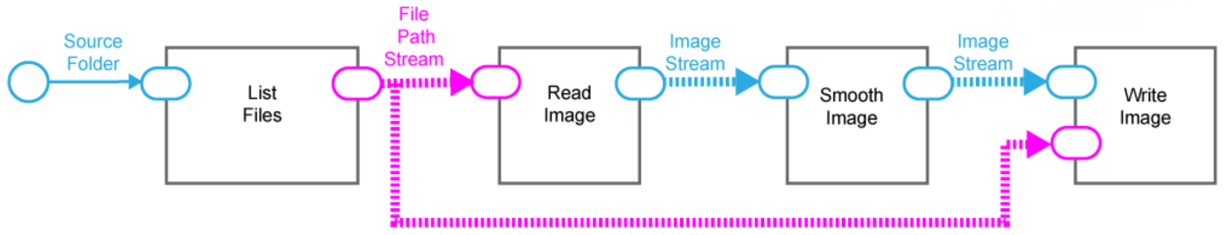
Programming Paradigm

Declarative Languages



Driven by Functional Call

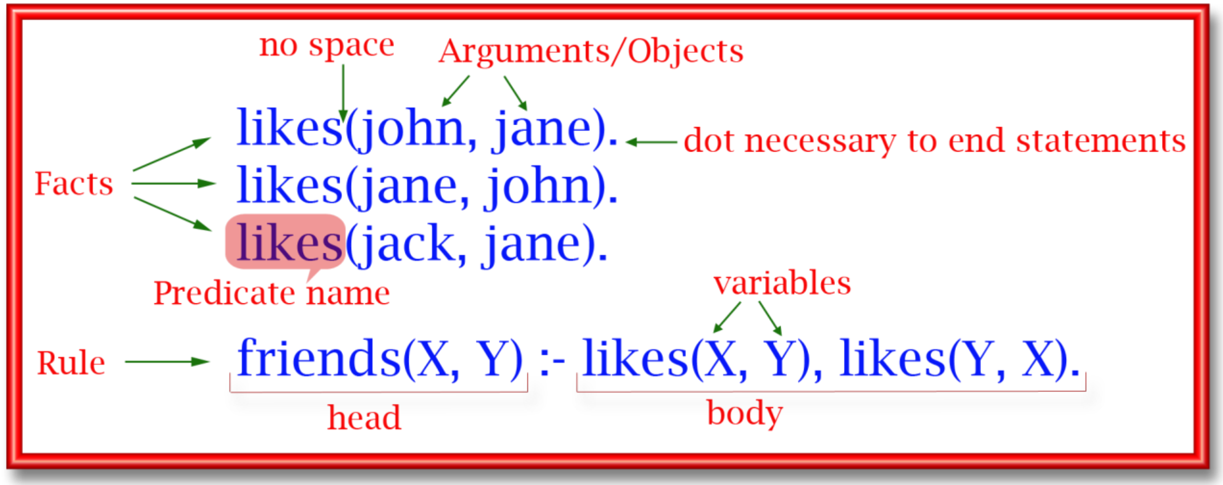
Data Flow Programming



Driven by Data Flow

Logic Programming

Program Window



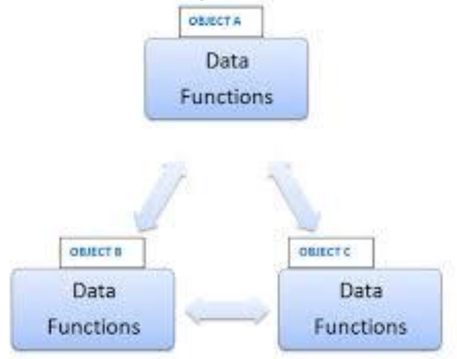
```
likes(john, jane).
likes(jane, john).
likes(jack, jane).
friends(X, Y) :- likes(X, Y), likes(Y, X).
```

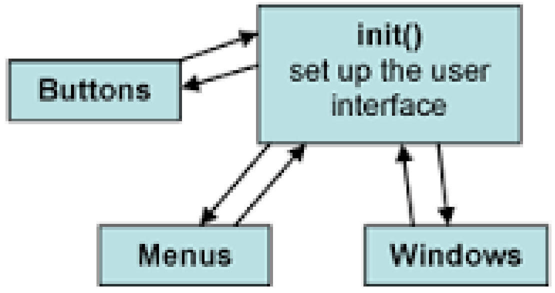
Driven by Logic Reasoning

Programming Paradigm

Imperative Languages

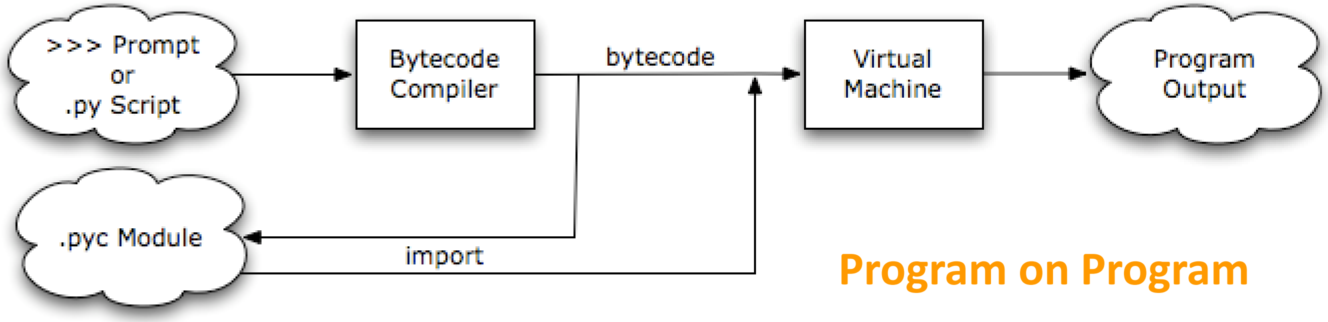
Object-Oriented Programming





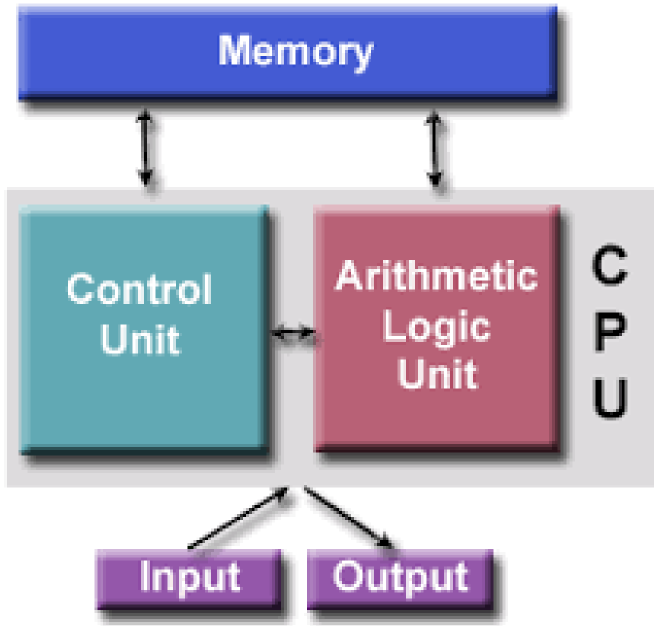
Event-Driven Programming

Scripting Programming



Program on Program

Von Neumann Programming
(Accumulator Model)



You should Learn to Program

VIDEO

Video

- If you did not watch this video in Lesson 1, watch it here:
- **Video:**

You Should Learn to Program: Christian Genco at TEDxSMU - Video

Get Started for Programming

- We're going to launch into an activity right now that reveals an important principle of programming.
- Try your best, and we'll discuss at the end.

Activity Guide – LEGO Instructions

LEGO Instructions

- Everyone needs a LEGO package of 6 LEGOS and the Activity Guide. You also need a plain piece of paper.
- Find a place on the floor to work – don't let anyone see your "artwork".
- Create a simple LEGO arrangement (and record it)

Rules to follow:

- All pieces must be connected in a single arrangement. YOU MUST USE ALL THE PIECES!
- Color matters: the final arrangement must be absolutely identical to the original.
- You should record their arrangement by taking a picture or creating a simple drawing.

Activity – Writing Instructions

- Write instructions for building your arrangement on the plain paper.
- On a separate sheet of paper, each person should write out a set of instructions, that another person could use to create the same arrangement.

Rules:

- You may only use words when creating these instructions. In particular, you may not include your image or draw images of your own.
- Try to make your instructions as clear as possible. Your classmates are about to use them!

LEGO Creation

- Take your LEGO creation apart, put the pieces back in the bag.
- Trade instructions and LEGO bags with another person and attempt to follow them with no help from the "artist".
- Hide the original image or drawing of the arrangement somewhere it cannot be seen.
  Move around the room to other people's instructions and try to follow them to build the desired arrangements.
- Have the original artist check whether the solutions are correct. Show the picture of the original artwork and see how close your classmate was to your idea!
- You should try at least 5 classmates' instructions.

Building Block -> LEGO Project

- Were you always able to create the intended arrangement? Were your instructions as clear as you thought?
- Why do you think we are running into these miscommunications? Is it really the “fault” of your classmates or is something else going on?
- If we were going to change human language to improve our odds of succeeding at this activity, what types of changes would we need to make?

Language Struct -> Programming Project

- So long as there are multiple ways to interpret language, we cannot have perfect precision.
- If we rigorously define the meaning of each command we use, then we can avoid misinterpretation and confidently express algorithms.
- This is different from the way we normally think and talk, and it might even take a while to get comfortable with communicating in this way.

Imagination -> Reality

- **Ambiguity** in human language led to issues or at least difficulty in creating the arrangements.
- We need to create a well-defined set of commands that all parties can agree upon for expressing the steps of a task.
- We need a **programming language**.

Abstraction -> Programming Language

- Today we saw how human language may not always be precise enough to express algorithms, even for something as simple as building a small LEGO arrangement.
- The improvements you have suggested actually create a new kind of language for expressing algorithms, which we as computer scientists call a **programming language**.
- In the coming unit we are going to learn a lot more about how we can use programming languages to express our ideas as **algorithms**, **build new things**, and **solve problems**.

Programming

Building Block -> LEGO Project

Imagination -> Reality

Abstraction -> Implementation

Language Struct -> Programs

Code Studio

- Finish up in Code Studio with Assessments and Reflections

Intro to Programming

LESSON 6 [CODE.ORG]

Debugging

LESSON 7 [CODE.ORG]

Project - Designing an App Part 3/4/5

LESSON 8/9/10 [CODE.ORG]