



# The Beauty and Joy of Computing

## Lecture #5 Programming Paradigms



UC Berkeley  
EECS

Guest Lecturer  
TA Peter Sujan

### CITRIS INVENTION LAB!

It provides prototyping resources such as 3D printing, laser cutting, soldering stations, hand and power tools. We are opening the lab to a limited number of students, staff & faculty to work on outside projects.

### 3D PRINTED GUNS!

Is gun control a fruitless effort?



[invent.citris-uc.org](http://invent.citris-uc.org)

[motherboard.vice.com/read/click-print-gun-the-inside-story-of-the-3d-printed-gun-movement-video](http://motherboard.vice.com/read/click-print-gun-the-inside-story-of-the-3d-printed-gun-movement-video)

# Programming Paradigms Lecture

- What are they?
  - Most are Hybrids!
- The Four Primary ones
  - Functional
  - Imperative
  - Object-Oriented
    - OOP Example: Sकेcthpad
  - Declarative
- Turing Completeness
- Summary



# What are Programming Paradigms?

- “The concepts and abstractions used to represent the elements of a program (e.g., objects, functions, variables, constraints, etc.) and the steps that compose a computation (assignment, evaluation, continuations, data flows, etc.).”
- Or, a way to **classify the style** of programming.



# Of 4 paradigms, how many can Snap! be?



- a) 1 (functional)
- b) 1 (not functional)
- c) 2
- d) 3
- e) 4



# Most Languages Are Hybrids!

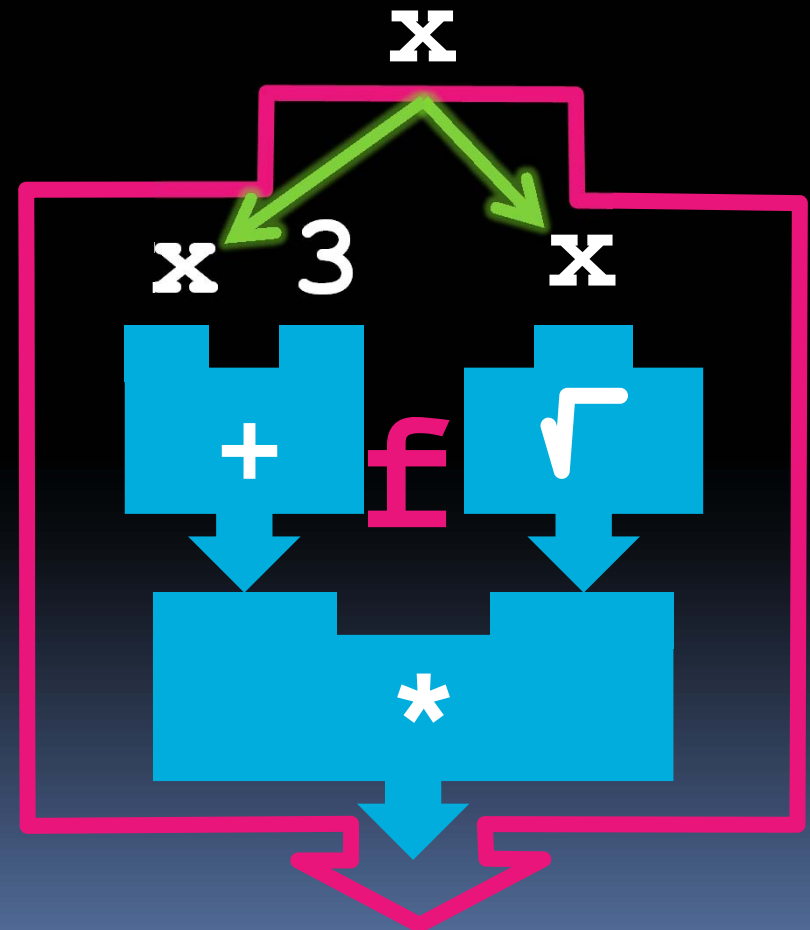
- This makes it hard to teach to students, because most languages have facets of several paradigms!
  - Called “Multi-paradigm” languages
  - Scratch too!
- It’s like giving someone a juice drink (with many fruit in it) and asking to taste just one fruit!



# Functional Programming (review)

- Computation is the evaluation of **functions**
  - Plugging pipes together
  - Each pipe, or function, has exactly 1 output
  - Functions can be input!
- Features
  - No state
    - E.g., variable assignments
  - No mutation
    - E.g., changing variable values
  - No side effects
- Examples (tho not pure)
  - Scheme, Scratch BYOB

$$f(x) = (x+3) * \sqrt{x}$$



# Imperative Programming

- “Sequential” Programming
- Computation a series of steps
  - Assignment allowed
    - Setting variables
  - Mutation allowed
    - Changing variables
- Like following a recipe. E.g.,
- Procedure  $f(x)$ 
  - $ans = x$
  - $ans = \sqrt{ans}$
  - $ans = (x+3) * ans$
  - $return\ ans$
- Examples: (tho not pure)
  - Pascal, C

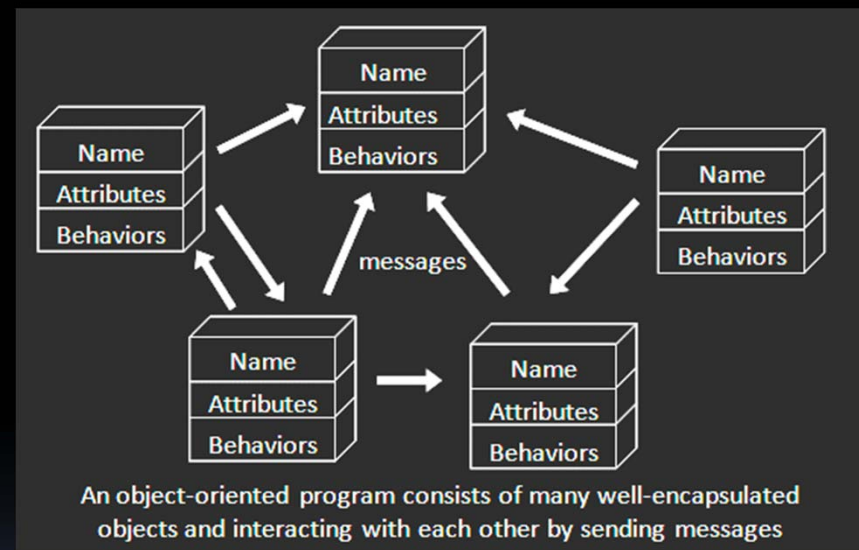
$$f(x) = (x+3) * \sqrt{x}$$





# Object-Oriented Programming (OOP)

- Objects as data structures
  - With methods you ask of them
    - These are the behaviors
  - With local state, to remember
    - These are the attributes
- Classes & Instances
  - Instance an example of class
  - E.g., Fluffy is instance of Dog
- Inheritance saves code
  - Hierarchical classes
  - E.g., pianist special case of musician, a special case of performer



[www3.ntu.edu.sg/home/ehchua/programming/java/images/OOP-Objects.gif](http://www3.ntu.edu.sg/home/ehchua/programming/java/images/OOP-Objects.gif)

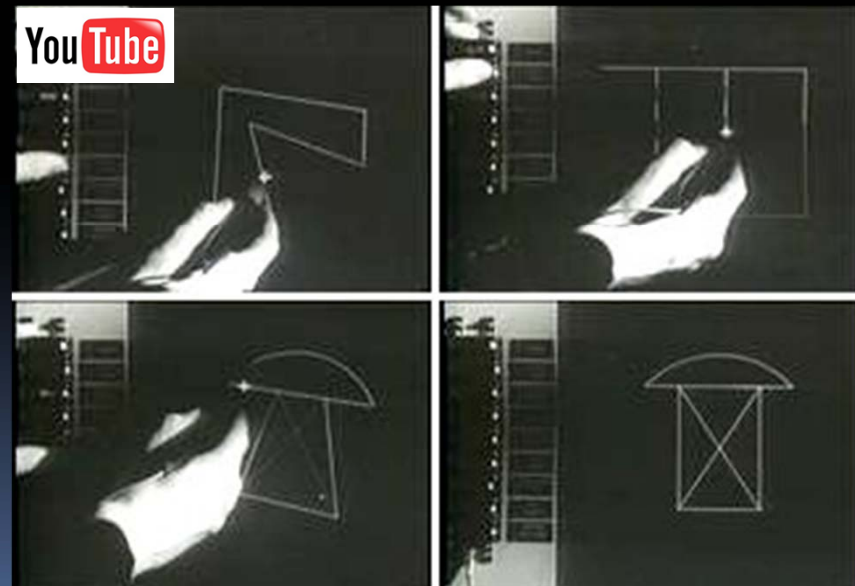


# OOP Example : SketchPad

- Dr. Ivan Sutherland
  - “Father of Computer Graphics”
  - 1988 Turing Award (“Nobel prize” for CS)
  - Wrote Sketchpad for his foundational 1963 thesis
- The most impressive software ever written
- First...
  - Object-oriented system
  - Graphical user interface
  - non-procedural language



Spent the past few years doing research @ Berkeley in EECS dept!



# OOP in BYOB

```
new counter
script variables count
set count to 0
report the script
  change count by 1
  report count
```

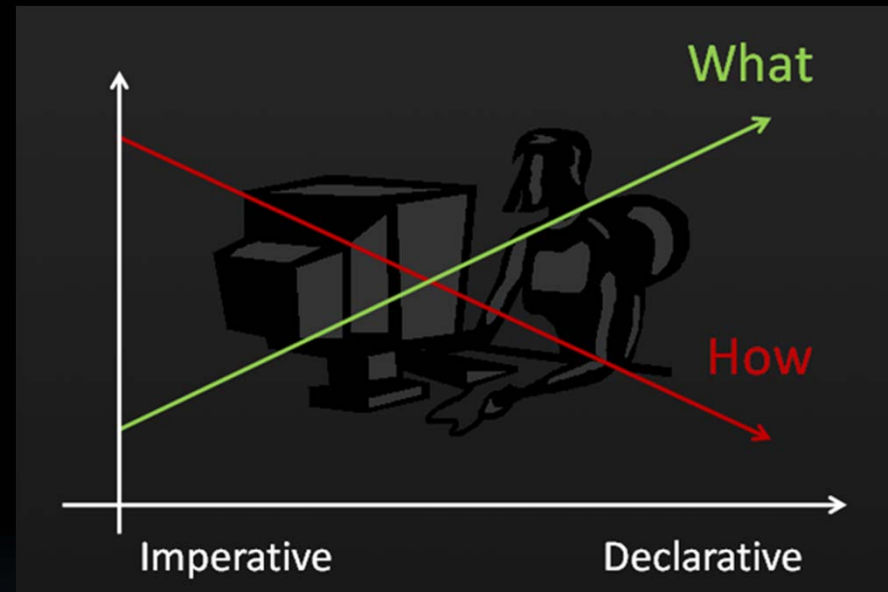
```
set counter1 to new counter
set counter2 to new counter
say call counter1 for 2 secs
say call counter1 for 2 secs
say call counter1 for 2 secs
think call counter2 for 2 secs
think call counter2 for 2 secs
say call counter1 for 2 secs
```



```
broadcast Dance
```

# Declarative Programming

- Express what computation desired without specifying how it carries it out
  - Often a series of assertions and queries
  - Feels like magic!
- Sub-categories
  - Logic
  - Constraint
    - We saw in Sketchpad!
- Example: Prolog



Anders Hejlsberg  
“The Future of C#” @ PDC2008  
[channel9.msdn.com/pdc2008/TL16/](http://channel9.msdn.com/pdc2008/TL16/)

# Declarative Programming Example

- Five schoolgirls sat for an examination. Their parents – so they thought – showed an undue degree of interest in the result. They therefore agreed that, in writing home about the examination, **each girl should make one true statement and one untrue one**. The following are the relevant passages from their letters:
  - Betty
    - Kitty was 2<sup>nd</sup>
    - I was 3<sup>rd</sup>
  - Ethel
    - I was on top
    - Joan was 2<sup>nd</sup>
  - Joan
    - I was 3<sup>rd</sup>
    - Ethel was last
  - Kitty
    - I came out 2<sup>nd</sup>
    - Mary was only 4<sup>th</sup>
  - Mary
    - I was 4<sup>th</sup>
    - Betty was 1<sup>st</sup>



Of 4 paradigms, what's the most powerful?

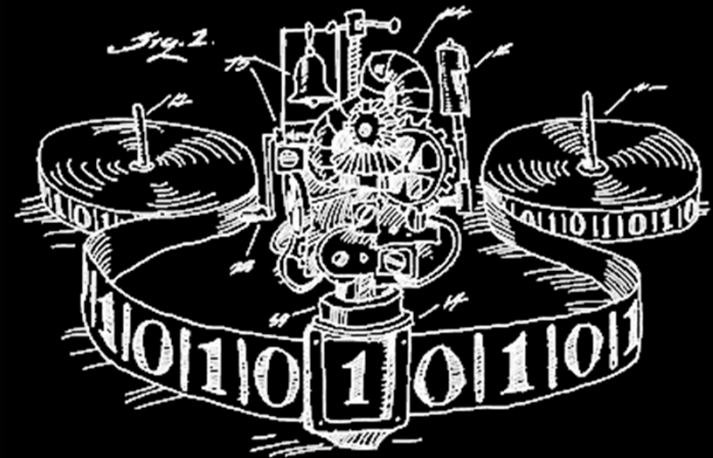
- a) Functional
- b) Imperative
- c) OOP
- d) Declarative
- e) All equally powerful



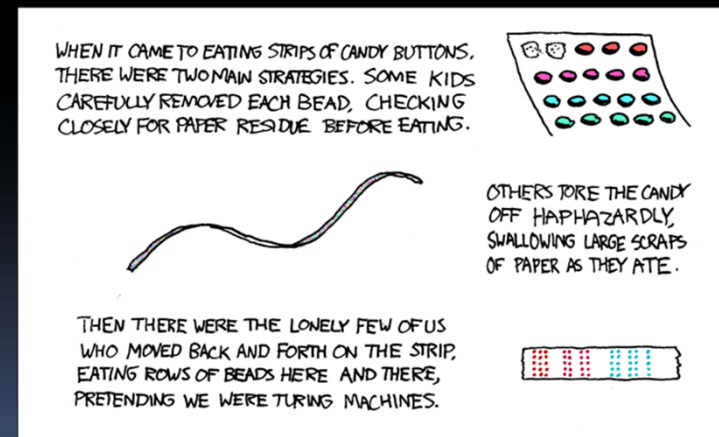


# Turing Completeness

- A Turing Machine has an infinite tape of 1s and 0s and instructions that say whether to move the tape left, right, read, or write it
  - Can simulate any computer algorithm!
- A Universal Turing Machine is one that can simulate a Turing machine on any input
- A language is considered Turing Complete if it can simulate a Universal Turing Machine
  - A way to decide that one programming language or paradigm is just as powerful as another



Turing Machine by Tom Dunne



Xkcd comic "Candy Button Paper"

# Ways to Remember the Paradigms

- Functional
  - Evaluate an expression and use the resulting value for something
- Imperative
  - First *do this* and next *do that*
- Object-oriented
  - Send messages between objects to simulate the temporal evolution of a set of real world phenomena
- Declarative
  - Answer a question via search for a solution

[www.cs.aau.dk/~normark/prog3-03/html/notes/paradigms\\_themes-paradigm-overview-section.html](http://www.cs.aau.dk/~normark/prog3-03/html/notes/paradigms_themes-paradigm-overview-section.html)



# Summary

- Each paradigm has its unique benefits
  - If a language is Turing complete, it is equally powerful
  - Paradigms vary in efficiency, scalability, overhead, fun, “how” vs “what” to specify, etc.
- Modern languages usually take the best from all
  - E.g., Snap!
    - Can be functional
    - Can be imperative
    - Can be object-oriented
    - Can be declarative

