

Programming

MICHAEL BERNSTEIN

CS 376

**Reminder:
project fair II
Monday after
Thanksgiving**

A Small Matter of Programming

- Software engineering is a highly complex task, a microcosm of many challenges in HCI
- Making software engineering more accessible could empower millions to customize applications and write programs

Research agenda

- Understand the challenges in programming
- Design more effective software engineering interfaces
- Aid novices in learning to program or writing programs
- Abstract best practices into toolkits

Understanding programmers

Information Needs in Programming

[Ko, DeLine and Venolia, ICSE '07]

- Observed 17 developers in 90-minute sessions and transcribed all activities
- Thematic coding of information needs
 - Writing code e.g., how do I use this method?
 - Submitting a change e.g., which files are included?
 - Triaging bugs e.g., is the problem worth fixing?
 - Reproducing failure e.g., what are failure conditions?
 - Understanding execution e.g., what caused this behavior?
 - Design e.g., why is the code implemented this way?
 - Awareness e.g., what are my collaborators working on?
- Most common need: collaborator awareness

Obstacles to learning APIs

[Robillard and DeLine, Empir. Software Engineering 2011]

- Survey and in-person interviews, combined reaching 440 professional software engineers
- Biggest challenge: inadequate documentation
- API intent: how it was intended to be used
 - “Nowhere in there does it say, and we intended to be used for a few graphics of small size because the memory footprint is going to be this.”
- Code examples: snippets, tutorials, working apps
- Penetrability: how much detail and implementation to expose?

Web foraging and programming

[Brandt et al., CHI '09]

- Laboratory study: ask programmers to implement a chat room in PHP
- This paper articulated how programmers make heavy use of the web
 - JIT learning of new skills
 - Clarifying existing skills
 - Reminding themselves of details
- Average participant spent 19% of their programming time on the web

Software engineering interfaces

Goals of software engineering interface research

- Design a better toolbench, produce a better programmer
- This research typically assumes that the programming language is static, but the interface of the IDE can be molded

Example-centric programming

[Brandt et al., CHI '10]

- Close the loop between the development environment and web search
- Autocomplete code via web examples

The screenshot shows a browser window with a search bar containing 'load image' (A). Below the search bar is a search result titled 'Loading an Image in Flex 3' (B) with a star rating (G). The result text (C) describes using a script to load an image into an Image Control after a Button is pressed, with a link to <http://livedocs.adobe.com/flex/3/langref/mx/controls/Image.html>. Below the text is a code snippet (D) for an XML-based application:

```
<?xml version="1.0"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">

  <mx:Image x="50" y="60" id="img" />
  <mx:Button click="loadImage(e)" />
  <mx:Script>
    <![CDATA[
      private function loadImage(e:MouseEvent):void {
        img.source = "image.jpg";
      }
    ]]>
  </mx:Script>
</mx:Application>
```

The code snippet includes annotations: 'loadImage(e)' is highlighted in blue, and the function definition 'private function loadImage(e:MouseEvent):void { img.source = "image.jpg"; }' is highlighted in green, with a circled 'F' next to it.



*bpVideo.mxml

Source Design

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute" creationComplete="loadData()">
3
4 <mx:Script>
5   <![CDATA[
6
7     public function loadData():void {
8       URLLoader
9     }
10  ]]>
11 </mx:Script>
12 </mx:Application>
13
```

URLLoader

URLLoader

```
import flash.xml.*;
import flash.events.IOErrorEvent;

public class URLLoader_loadExample extends Sprite {
    private var xmlTextField:TextField = new TextField();
    private var externalXML:XML;
    private var loader:URLLoader;

    public function URLLoader_loadExample() {
        var request:URLRequest = new URLRequest("xmlfile.xml");

        loader = new URLLoader();

        try {
            loader.load(request);
        }
        catch (error:SecurityError)
        {
            trace("A SecurityError has occurred.");
        }

        loader.addEventListener(IOErrorEvent.IO_ERROR, errorHandler);
        loader.addEventListener(Event.COMPLETE, loaderCompleteHandler);

        xmlTextField.x = 10;
        xmlTextField.y = 10;
        xmlTextField.background = true;
        xmlTextField.autoSize = TextFieldAutoSize.LEFT;
    }
}
```

urloader actionsript
urloaderdataformat
urloader events
urloader flash

Problems Console

```
<terminated> bpVideo [Flex Appli
[SWF] Users:mirad:work:pro
undefined
```

Asking 'why' questions of code

[Ko and Myers CHI '04, ICSE '09]

- Debugging problems often reduce to “why” questions
- Analyze program traces to answer them

The screenshot shows the 'Whyline for Java - Paint' application. The main window is titled 'PaintWindow #1,785' and contains a canvas with a green and black drawing. On the left, there is a tool palette with 'Pencil', 'Eraser', and 'Line' options, and color sliders for 'Red', 'Green', and 'Blue'. Below the canvas are buttons for 'Clear the canvas' and 'Undo my last stroke'. A status bar at the bottom indicates 'after this window repainted...'. On the right side, there is a list of 'why' questions, with 'why didn't update() execute?' highlighted in yellow. The list includes categories like 'properties of this filled rectangle', 'objects rendering this', 'windows', 'JComponent "currentColorComponent"', 'JPanel "colorPanel"', 'JPanel "controlPanel"', 'JPanel "c"', 'PaintWindow', 'booleans', 'floats', 'ints', 'Colors', 'Components', 'Dimension2Ds', 'Fonts', 'Listeners', 'Maps', 'Supports', and 'other fields'. At the bottom, there is a toolbar with icons for zooming and a 'showing all I/O events' indicator.

Missing user-facing feedback

[Ko and Zhang, CHI '11]

- Usability heuristic: all user inputs should produce some form of feedback
- Statically analyze code to identify user inputs that produce no feedback

Feedlack!

project **Calculator**

Feedlack found **54** places in your code that appear to be missing feedback:

nd() at overlib.js 927 may not produce feedback

script() at Calculator.html 90 may not produce feedback

func(f) at newcalc.js 919 may not produce feedback

digit(n) at newcalc.js 820 may not produce feedback

script() at Calculator.html

```
602         'return overlib('Sets
603         onmouseout='nd()';'
604         onmousedown=
605         'if(base==10){topbar.
606         style='cursor: defau
607         type='radio'
```

nd() at overlib.js 927

When the user performs a

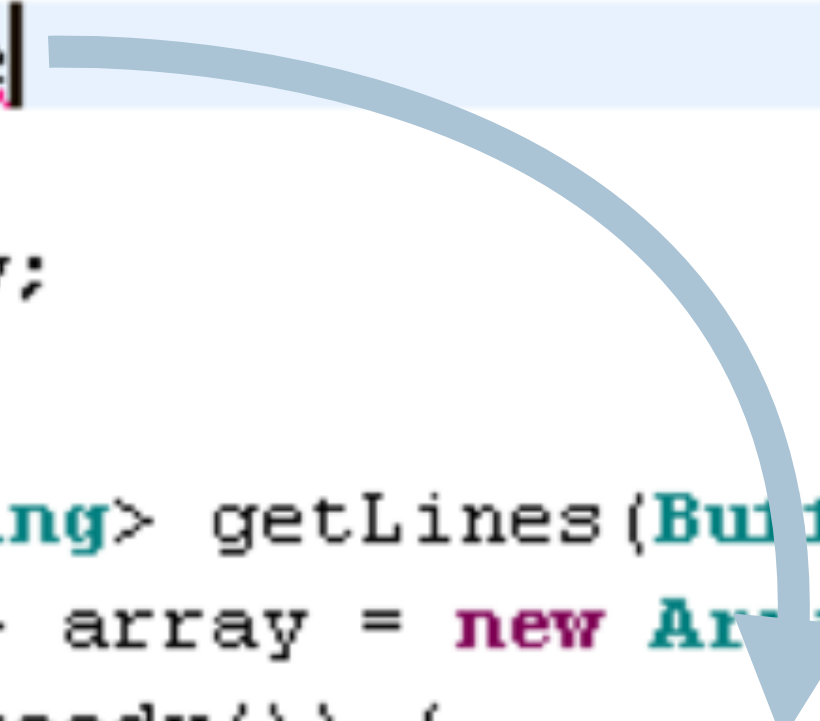
- mouseout (Calculator.html 603),
- mouseout (Calculator.html 947),
- mouseout (Calculator.html 1025),
- mouseout (Calculator.html 598)

Keyword programming

[Little and Miller, UIST '06, ASE '09]

- Macro programming is difficult to learn
- Allow keyword search over an API:
e.g., “click search button” or “left margin 2 inches”

```
public List<String> getLines(BufferedReader src) throws Exception {  
    List<String> array = new ArrayList<String>();  
    while (src.ready()) {  
        add line  
    }  
    return array;  
}  
  
public List<String> getLines(BufferedReader src) throws Exception {  
    List<String> array = new ArrayList<String>();  
    while (src.ready()) {  
        array.add(src.readLine());  
    }  
    return array;  
}
```



Visual layout of code snippets

[Bragdon et al., CHI '10]

- Most engineering time is spent navigating across multiple related code snippets
- So, design for many small windows into files


```
ShapeDraw ▶ MainPanel ▶
public MainPanel()
{
    this.layoutAsCardinalDirections();

    this.createPropertyButtons();

    Button featureButton = SpecialFeatureButton.
        getInstance(this);

    Button randomShapes = this.
        createRandomShapeButton();

    String[] messages = this.
        generateStatisticsMessages();
    this.handleStatisticsGUI(messages);

    MenuBar menuBar = this.createMenuBar();

    ShapeButton[] shapeButtons = this.
        createShapeButtons();
    Panel shapePanel = this.makeShapeButtonPanel(
        shapeButtons);

    Panel moreFunctionsPanel = new Panel();
    moreFunctionsPanel.layoutAsGrid();
    Label moreFunctionsLabel = new Label(
        "More Functions");
    moreFunctionsLabel.center();
    moreFunctionsPanel.add(moreFunctionsLabel);
    moreFunctionsPanel.add(randomShapes);
    moreFunctionsPanel.add(_deleteShape);
    moreFunctionsPanel.add(_statsButton);
    moreFunctionsPanel.add(featureButton);

    _shapeInfoPanel = new ShapeInfoPanel();
}
```

```
ShapeDraw ▶ MainPanel ▶
public void createPropertyButtons()
{
    _deleteShape = DeleteButton.getInstance(
        this);
    ((ShapeButton) _deleteShape).storeName(
        "Delete Active Shape");
    _deleteShape.setFocusable(false);

    init();
}
```

```
ShapeDraw ▶ MainPanel ▶
public Button createRandomShapeButton()
{
    Button button = Dropdown.getInstance(
        this);
    button.setFocusable(false);

    return button;
}
```

```
ShapeDraw ▶ MainPanel ▶
private void createMenu1(Menu m)
{
    MenuItem textInput = TextMenuButton.
        getInstance();
    m.add(textInput);
}
```

```
ShapeDraw ▶ TextMenuButton ▶
public static MenuItem getInstance()
{
    TextMenuButton item = new TextMenuButton();
    item.setText("Text Input");
    return item;
}
```

MainPanel.createMenuBar Undo

Debugging with runtime info

[Lieber, Brandt, and Miller, CHI 2014]

The image shows a side-by-side view of a web browser and a code editor. The browser window on the right displays a form titled "Theseus Demo" with two input fields: "Name: Tom" and "Location: Boston", and a "Save" button. The code editor on the left shows the source code for "demo/public/index.html". The code includes a click handler for a button that calls a "save" function, which in turn calls a "getData" function. The "save" function uses jQuery's \$.ajax to perform a POST request. Call counts are shown in the left margin of the code editor: the click handler has 1 call, while the save and getData functions have 0 calls. The status bar at the bottom indicates "Line 40, Column 26 — 49 Lines".

```
21 <script>
22   1 call $(function () {
23     0 calls   $("button").on("click", function () {
24       save(getData());
25     });
26   });
27
28   0 calls function getData() {
29     return { name: $("#name").val(), location: $("#location").val() };
30   }
31
32   0 calls function save(obj) {
33     $("#status").text("Saving...").show();
34     $.ajax({
35       type: "POST",
36       url: "/",
37       data: obj,
38     }).done(function() {
39       $("#status").text("Saved!");
40     }).fail(function () {
41       $("#status").text("Error!");
42     }).always(function () {
43       setTimeout(function () {
44         $("#status").hide();
45       }, 3000);
46     });
47   }
48 </script>
49
```

Languages that learn from crowds

[Fast and Bernstein, UIST '16]

- If your functions sent back information to a central community server, could they...
 - Recover from crashes?
 - Auto-optimize?
 - Test themselves?

Count the vowels in a string

str to int

60 users

```
import re

@meta(parent="5700375c2f6a2f000330436a")
def count_vowels(s):
    return len(re.findall('[aeiou]', s, flags=re.I))
```

Warning: Meta has found a possible alternative that is **1.3** times faster

Example inputs:

```
count_vowels("UIST") #=> 2
```

```
count_vowels("CHI") #=> 1
```

Known errors:

```
count_vowels(['CHI', 'UIST']) #=> expected string or bytes-li
```

You can load this snippet with:

```
count_vowels = meta.load("http://www.meta-lang.org/snippets/5
```

Learning programming

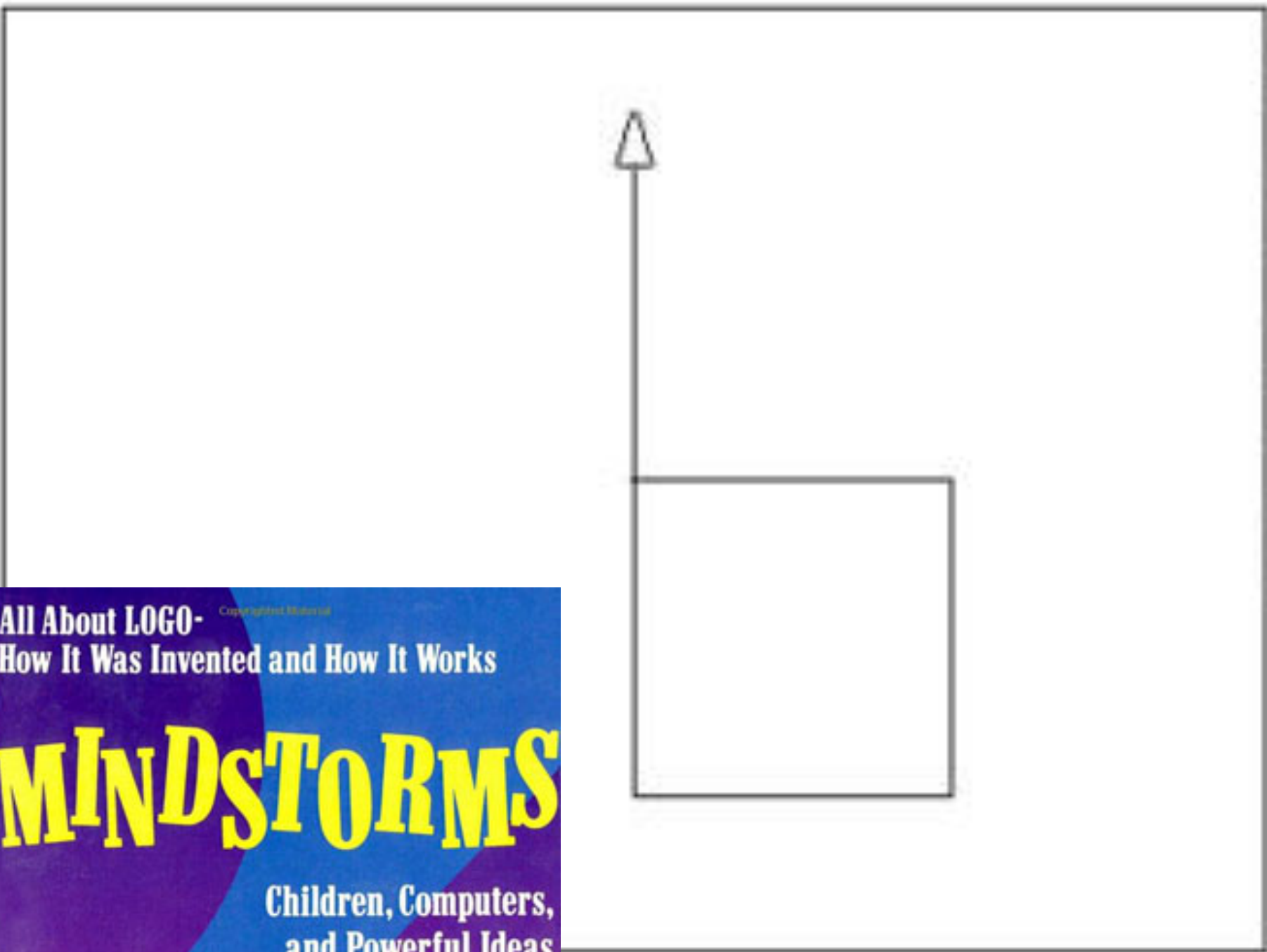
Goals of programming education

- Make programming accessible to new populations: children, scripters, interested amateurs
- Tools and innovations depend on the population

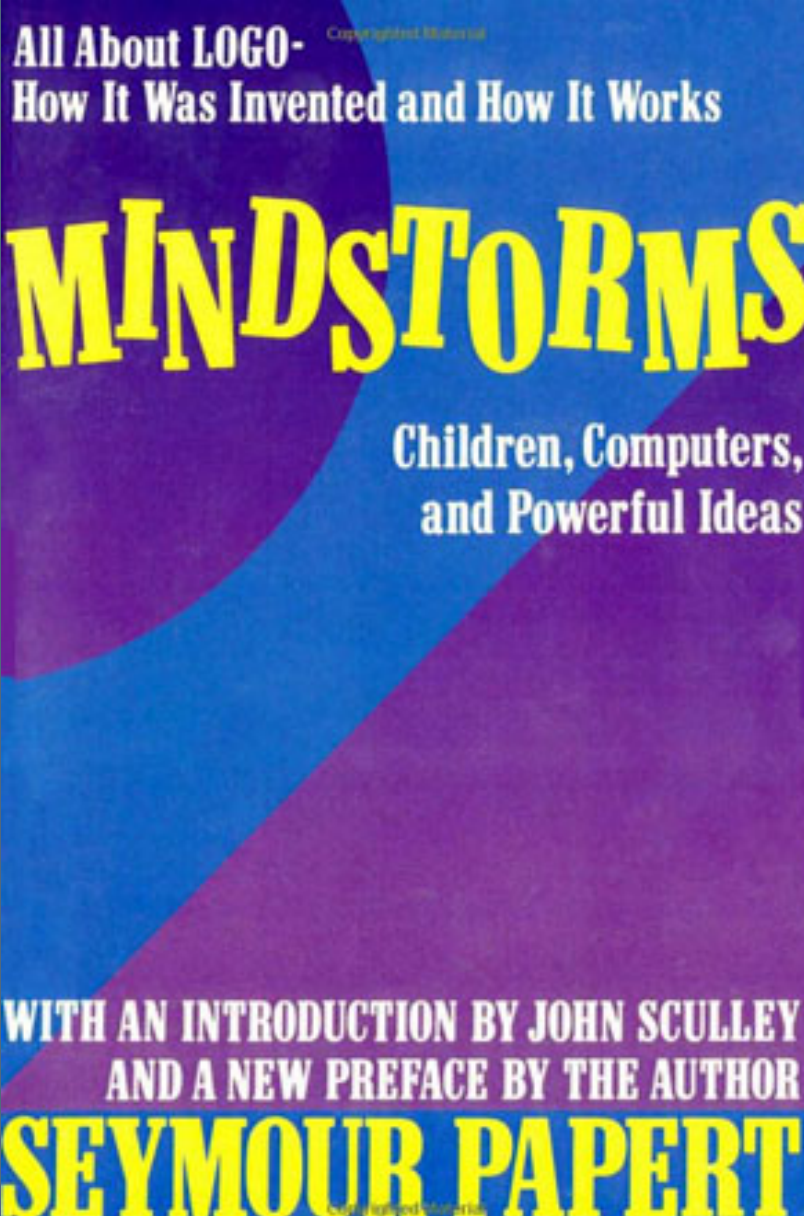
Logo: programming for children

[Papert '93]

- Constructionist learning: learning happens most effectively when people are making tangible objects
- Lego Mindstorms followed this mold and was named after it



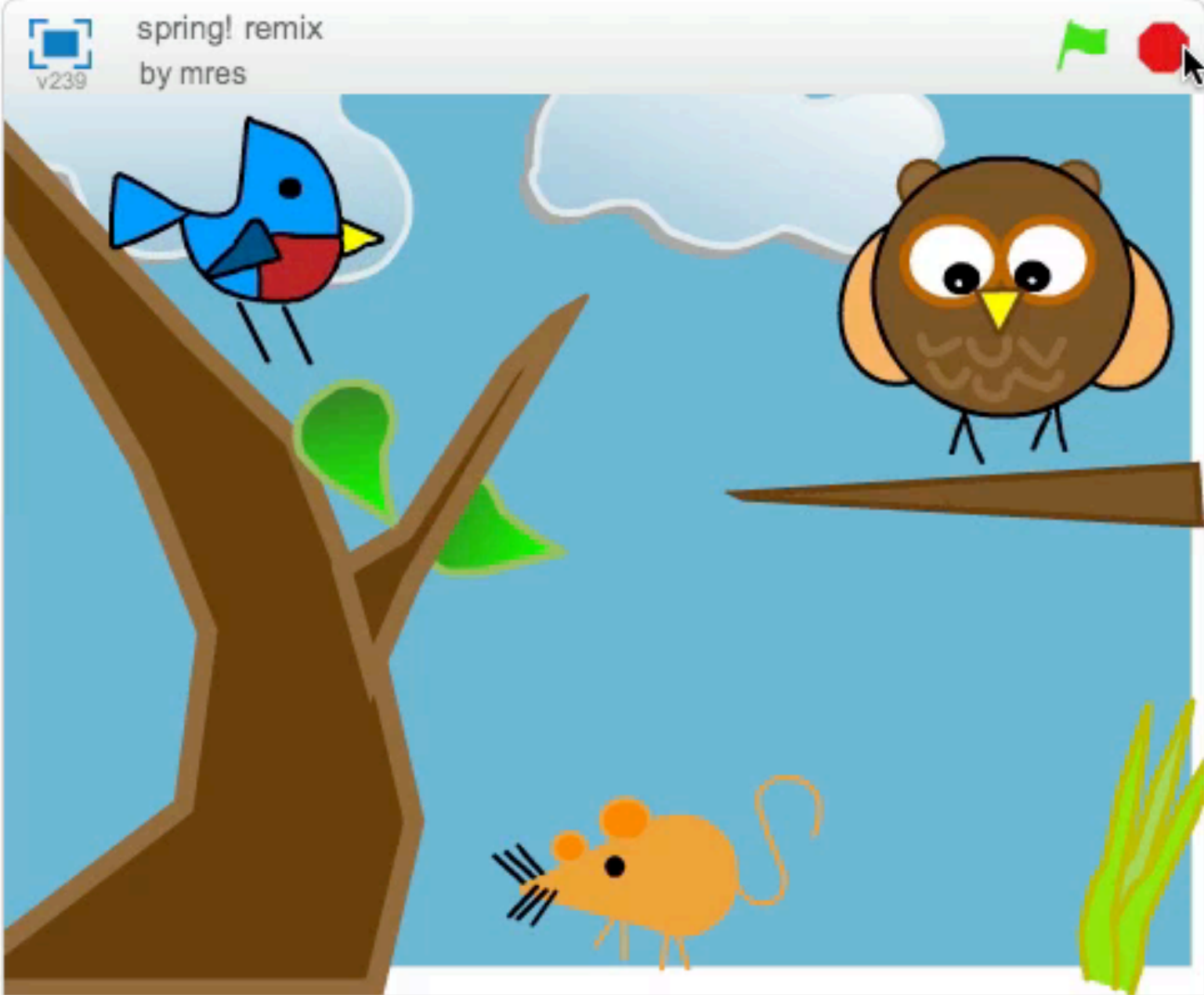
```
Welcome to Berkeley Logo
? right 90
? forward 100
? right 90
? forward 100
? right 90
? forward 100
? right 90
? forward 100
? forward 100
? □
```



Scratch: kids remix and create

[Resnick et al., CACM '09]

- Social: upload and remix others' programs
- All programming has been done online. This data has led to many papers on understanding notions of authorship and creative remixing.



x: 231 y: 180

Sprites New sprite: [Icons]

Stage 1 backdrop

New backdrop: [Icons]

Sprite5 Sprite Sprite6 Sprite1 Sprite3

Sprite2 Sprite4

Scripts Costumes Sounds

- Motion
- Looks
- Sound
- Pen
- Data
- Events
- Control
- Sensing
- Operators
- More Blocks

Make a Block

jump 1

```

when green flag clicked
  go to x: 160 y: 75
  forever loop
    jump 20
  jump 30

define jump height
  change y by height
  wait 0.5 secs
  change y by -1 - height
  wait 0.5 secs
  
```

Backpack

Online python tutor

[Guo, SIGCSE '13]

- Embeddable Python data structure visualization
- Over 200,000 users and a dozen universities using it

```
1 def listSum(numbers):  
2     if not numbers:  
3         return 0  
4     else:  
5         (f, rest) = numbers  
6         return f + listSum(rest)  
7  
8 myList = (1, (2, (3, None)))  
9 total = listSum(myList)
```

[Edit code](#)

< Back Step 11 of 18 Forward >

→ line that has just executed
→ next line to execute

Frames

Global variables

- listSum → function listSum(numbers)
- myList → tuple (1, (2, (3, None)))

listSum

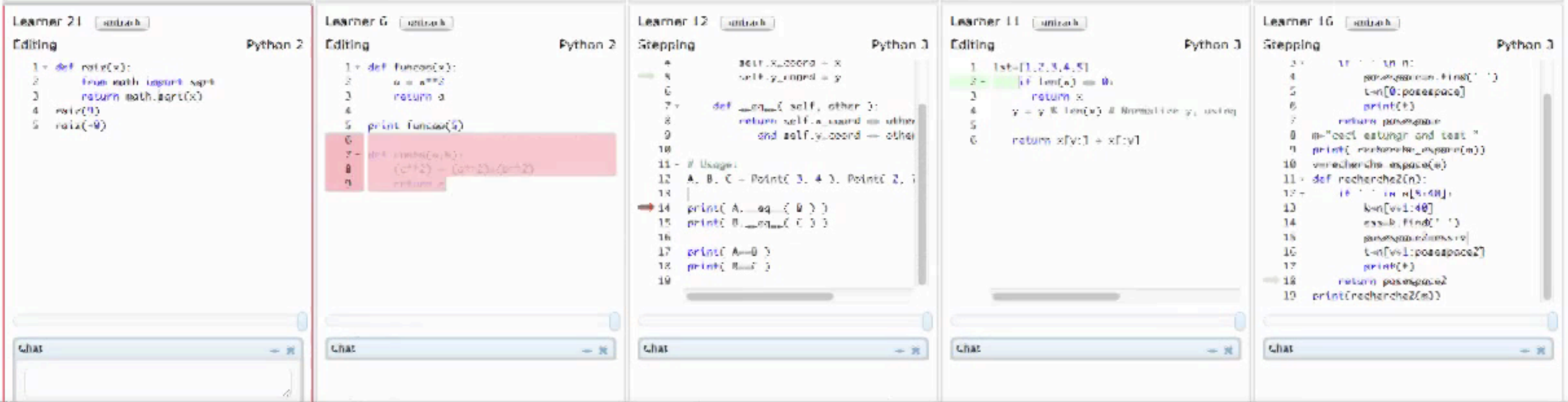
- numbers → tuple (1, (2, (3, None)))
- f → 1
- rest → tuple (2, (3, None))

listSum

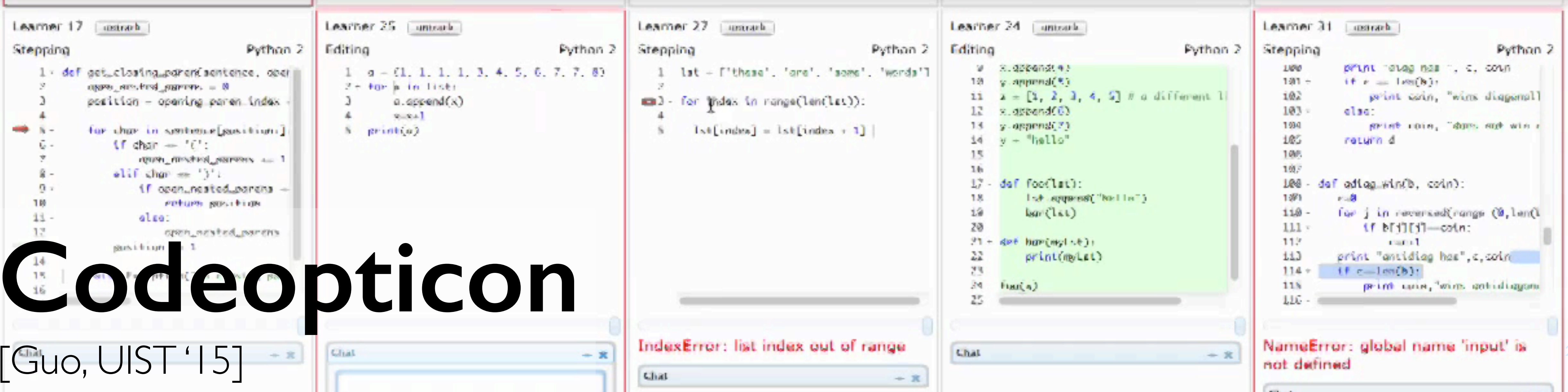
- numbers → tuple (2, (3, None))
- f → 2
- rest → tuple (3, None)

Objects

- function listSum(numbers)
- tuple (1, (2, (3, None)))
- tuple (2, (3, None))
- tuple (3, None)



Watch many learners code and debug in real time



Codeopticon

[Guo, UIST '15]

Programming by demonstration

Goals of PBD

- Teach a computer to program simply by demonstrating what should be done
- Challenges
 - There is an infinite, and hugely branching, space of programs that might be inferred
 - Inferred macros can be extremely brittle

Recall: EAGER

[Cypher, CHI '91]

- Infer a macro by watching the user's behavior

Creating a Subject List

A user has a stack of message cards (a) and wants to make a list of the subjects of the messages. The user copies the subject from the first message, goes to the "**Subject List**" card, types "1.", and pastes in the first subject (b). The user then goes to the second message, copies its subject, and adds it to the Subject List.

At this point, the Eager icon pops up (c), since Eager has detected a pattern in the user's actions. Eager highlights the right-arrow button in green (c), since it anticipates that the user will click here next. Eager continues anticipating that the user will navigate to the third message, select (d) and copy its subject, go to the Subject List, type "3." (e) and then paste in the subject (f).

The user is now confident that Eager knows what to do, and clicks on the Eager icon. It completes the task automatically (g).

File Edit Go Tools Objects



(a)

File Edit Go Tools Objects



(b)

File Edit Go Tools Objects



File Edit Go Tools Objects



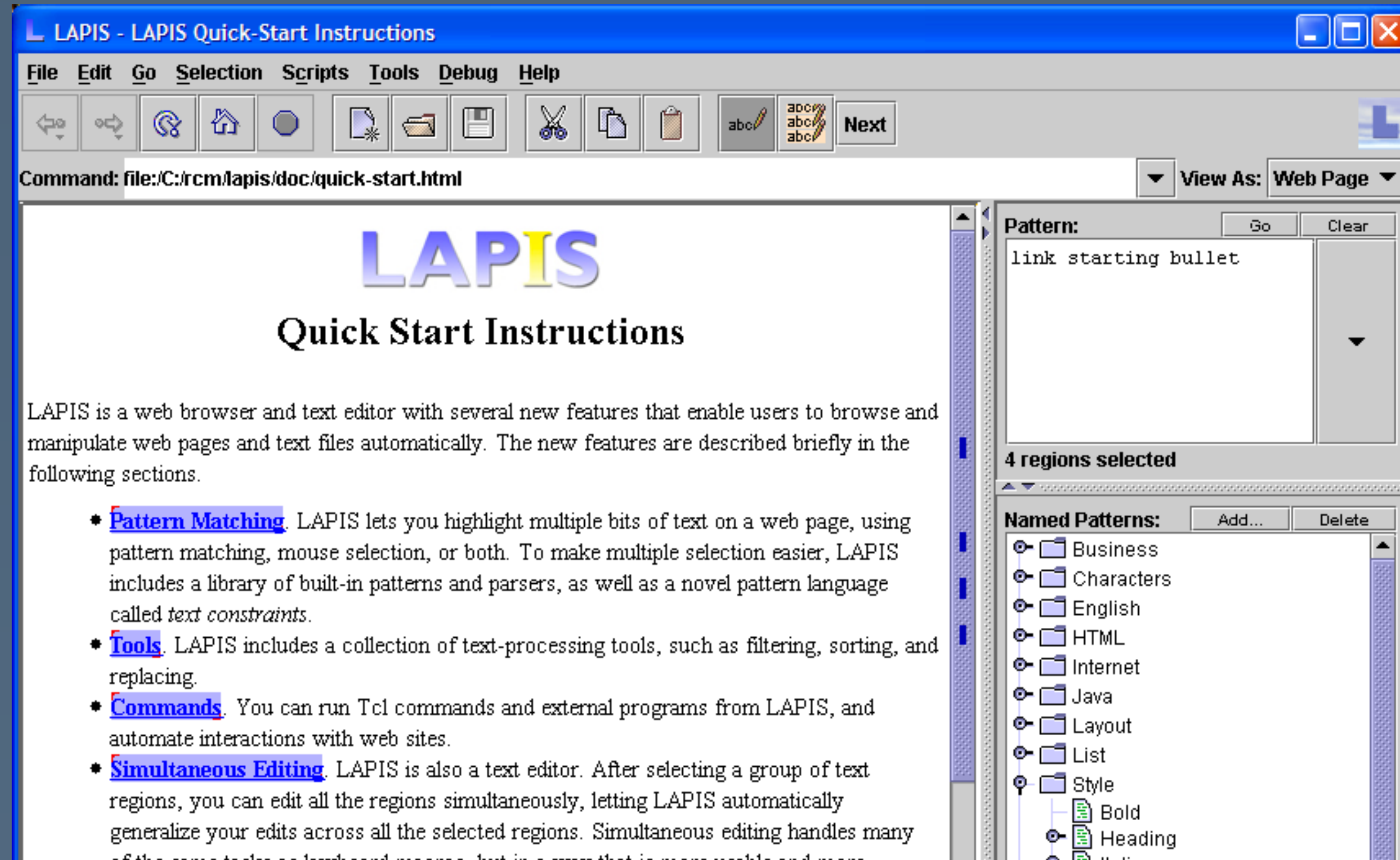
File Edit Go Tools Objects



Simultaneous structured editing

[Miller and Myers, USENIX '01]

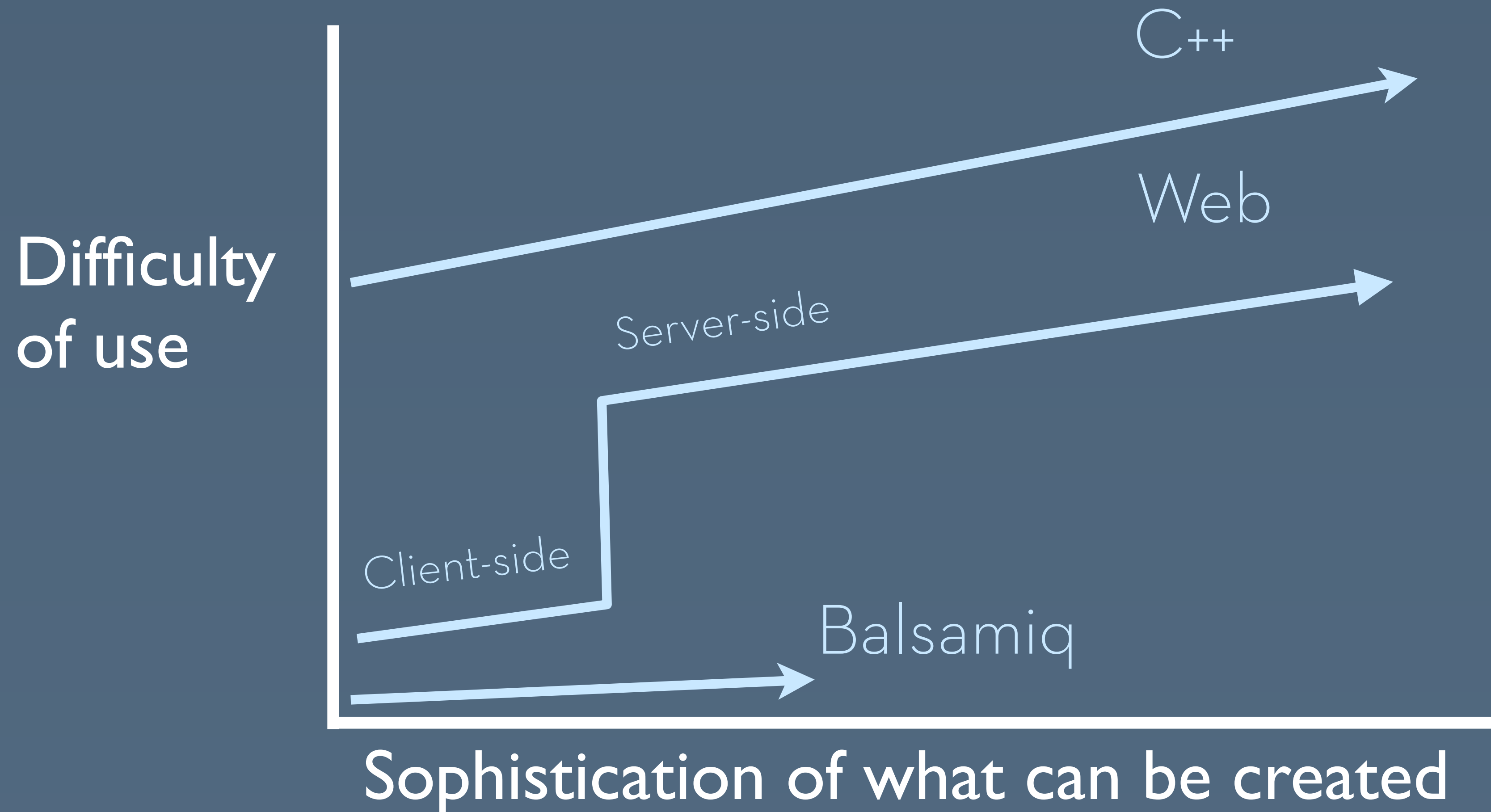
- Utilize lightweight structure in text
- Today, versions of this exist in Sublime Text



Toolkits

Threshold/Ceiling Tradeoff

[Myers, Hudson and Pausch, TOCHI 2000]



Research agenda: toolkits

- Crystallize and formalize a perspective on a difficult engineering problem
- If successful, shift the entire programming practice for the area

Sikuli: programming with screenshots

[Yeh, Chang, and Miller, UIST '09]

- Visual template search in desktop scripting



```
IDE\src\test-sikuli.py
File
[Icons: Camera, Monitor, Pin, Play]
while True:
    if find().inside().find():
        popup("bus has arrive|")
```

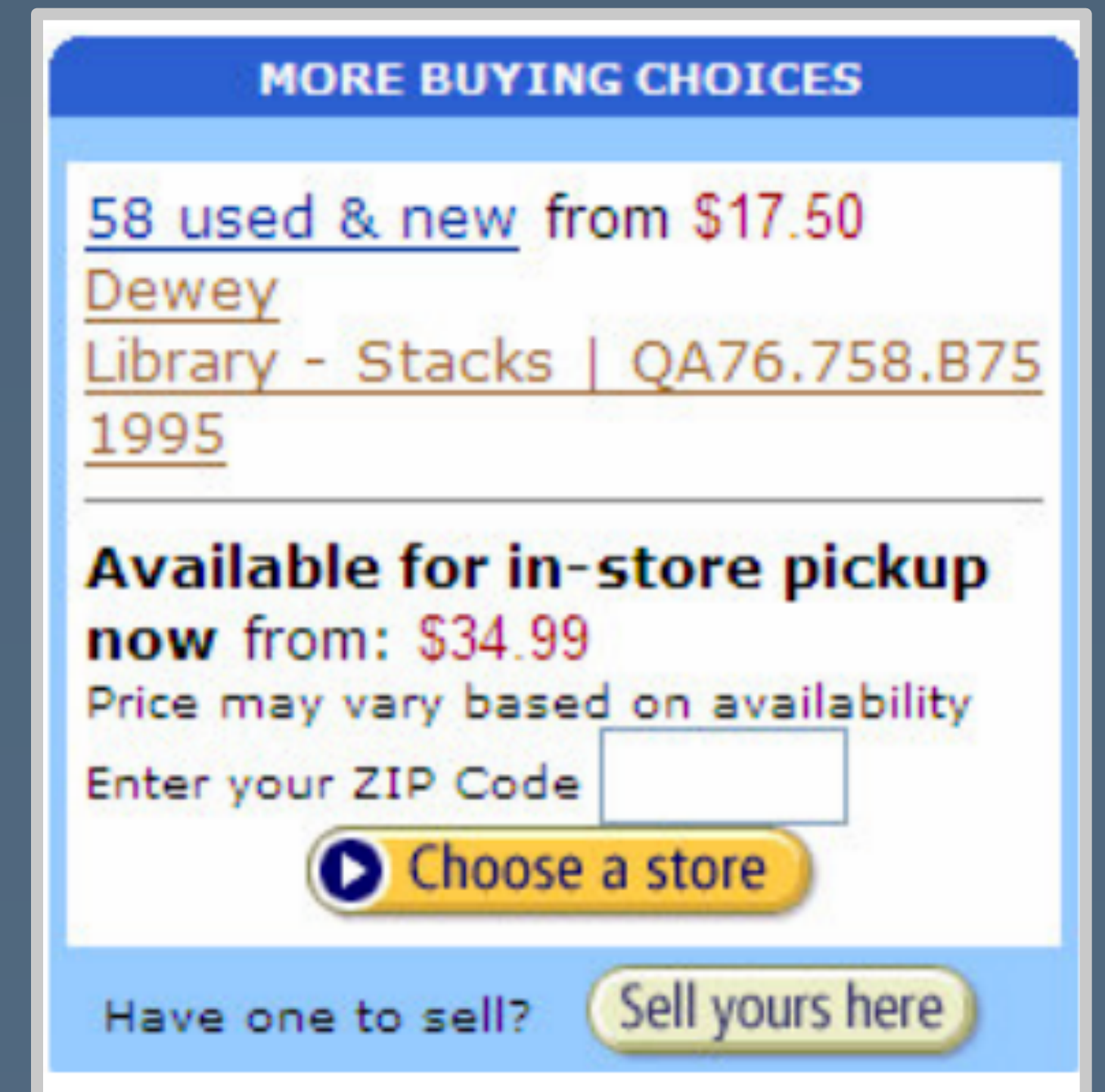
The screenshot shows a Python IDE window titled "IDE\src\test-sikuli.py". The code defines a loop that continuously checks for a specific visual pattern. The first pattern is a red rectangle on a map, and the second is a red location pin. When both are found, a popup message "bus has arrive|" is displayed.

Recall: Chickenfoot

[Bolin et al., UIST 2008]

- Lower the threshold to writing programs
- Allow users with little programming skill to author behaviors
 - e.g., Chickenfoot

```
isbn = find('number just after isbn')
with (fetch('libraries.mit.edu')) {
  pick('Keywords');
  enter(isbn)
  click('Search')
  link=find('link just after Location')
}
// back to Amazon
if (link.hasMatch) {
  insert(before('first rule after "Buying"'),
  link.html)
```



The screenshot shows a section titled "MORE BUYING CHOICES" for a book. It lists two options: "58 used & new from \$17.50" and "Available for in-store pickup now from: \$34.99". The book title is "Dewey Library - Stacks | QA76.758.B75 1995". There is a text input field for "Enter your ZIP Code" and a "Choose a store" button. At the bottom, there is a "Sell yours here" button.

Research agenda: HCI and programming

- Understand the challenges in programming
- Design more effective software engineering interfaces
- Aid novices in learning to program or writing programs
- Abstract best practices into toolkits

Discussion rooms

Rotation	Littlefield 107	Littlefield 103
a	12	34
b	24	13
c	14	23
d	34	12
e	13	24
f	23	14