Quantifying Program Comprehension

Michael Hansen, Rob Goldstone, Andrew Lumsdaine

Percepts and Concepts Lab, Spring 2013

Outline

The eyeCode Experiment

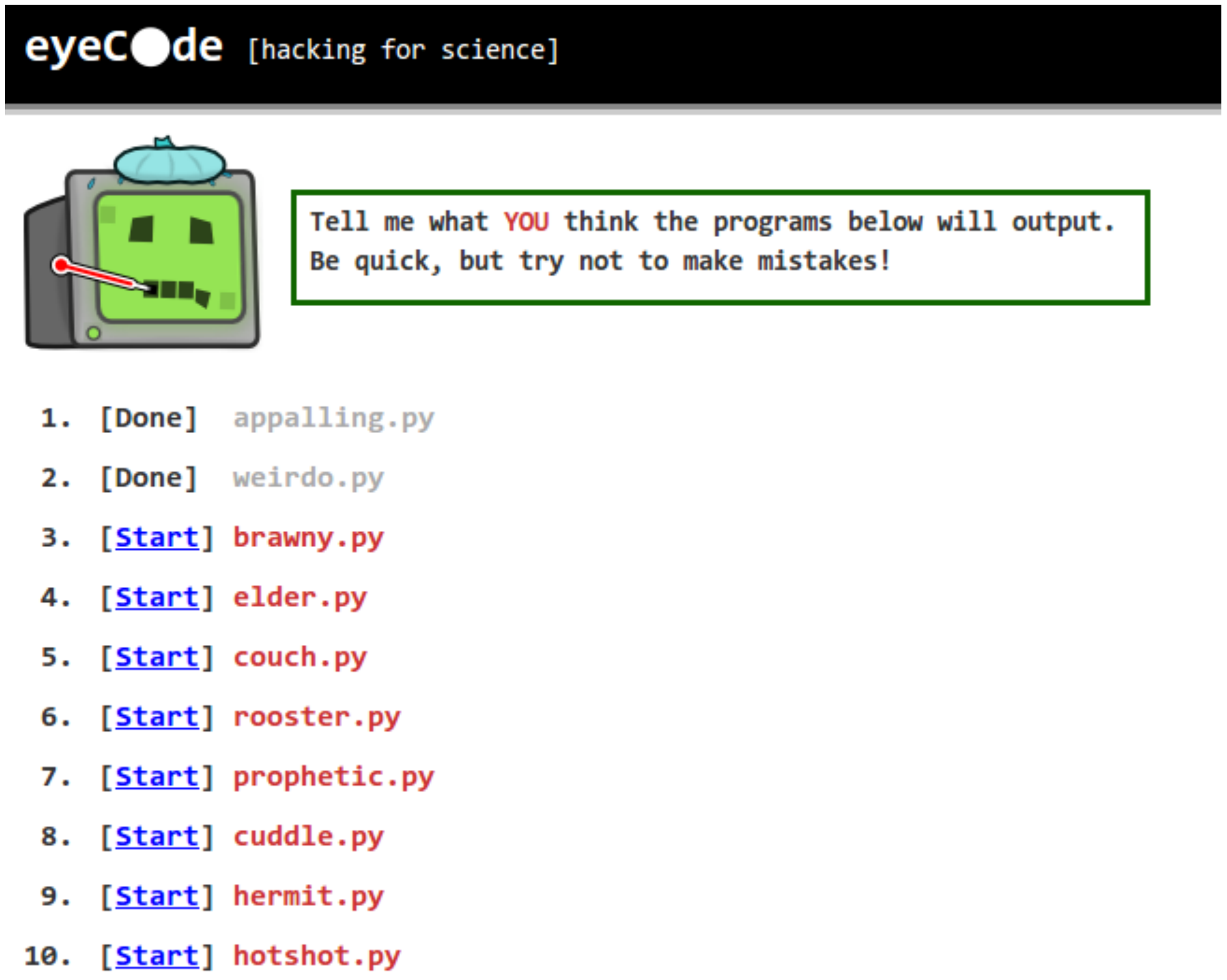
Task

- Predict printed output of 10 short Python programs
- 2-3 versions of 10 programs, randomly assigned
- Pre/post surveys

Goals

- Small code changes = large effects?
- Complexity is more than metrics
- Eye-tracking data for modeling program comprehension

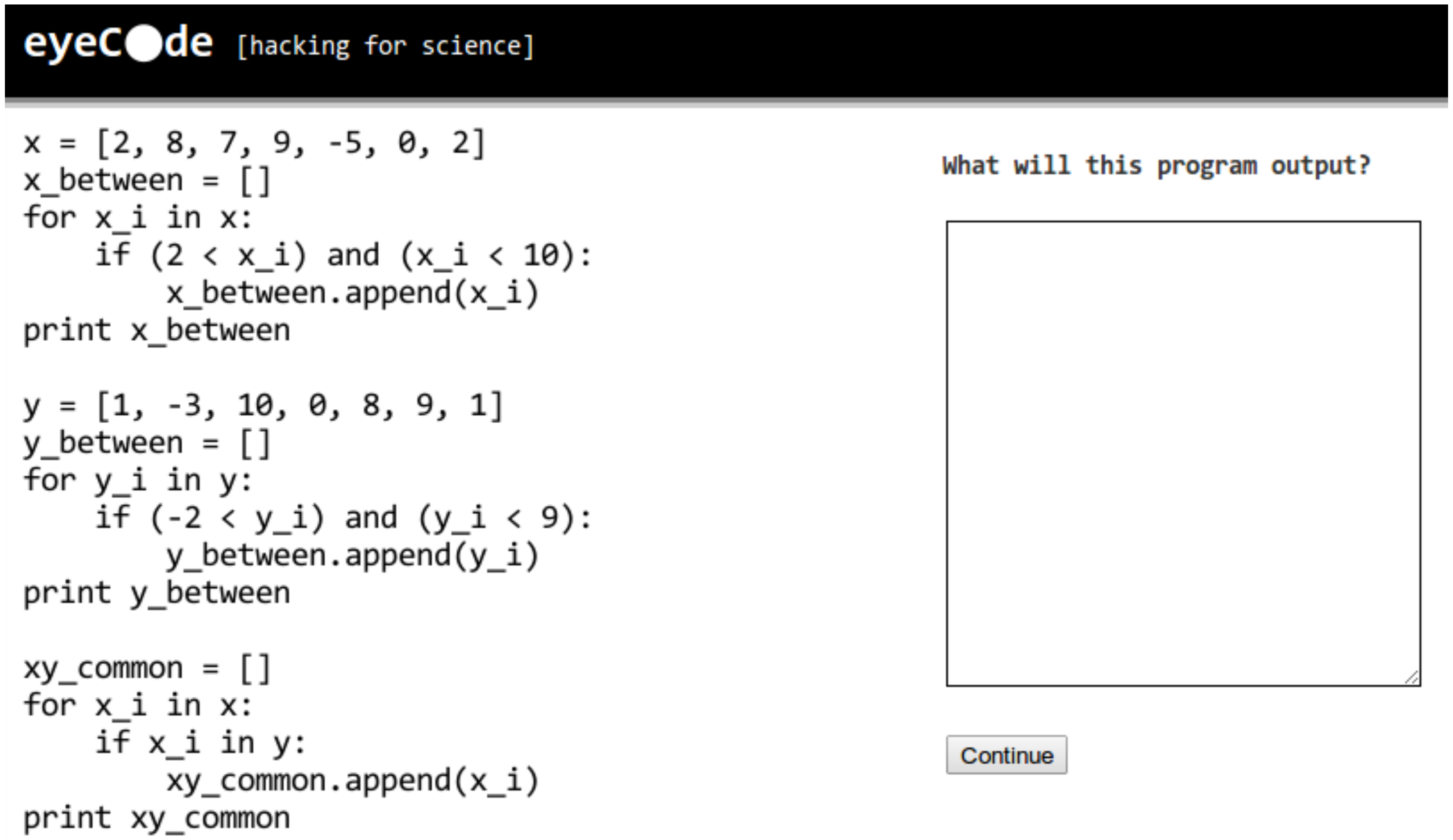
Home Screen



Tell me what YOU think the programs below will output.
Be quick, but try not to make mistakes!

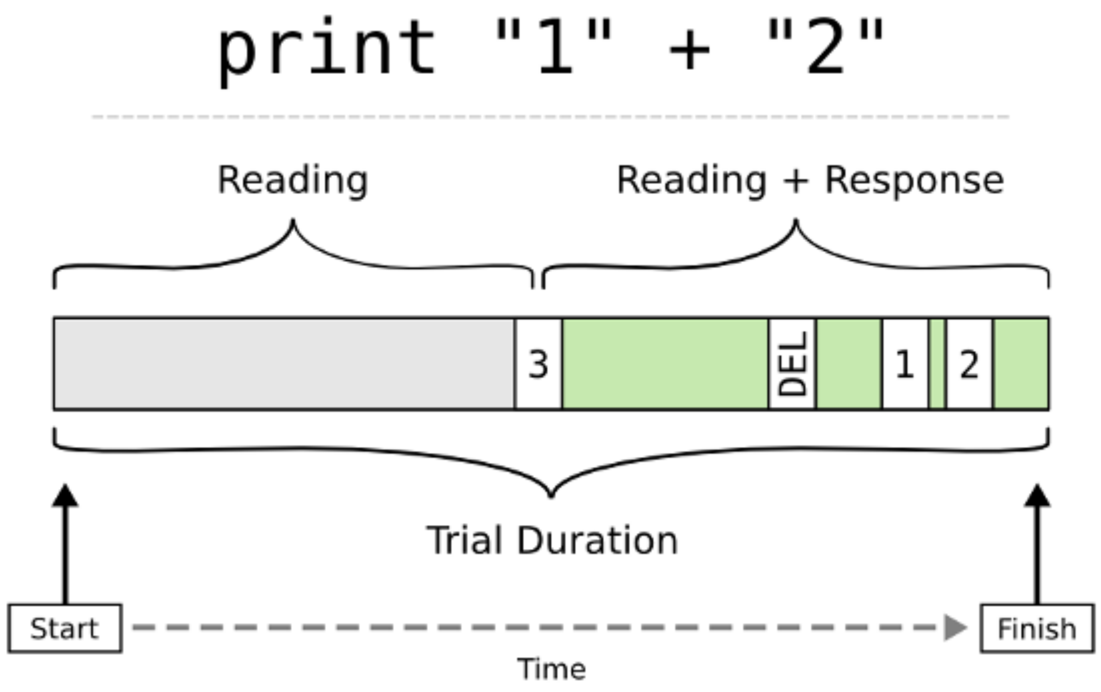
1. [Done] appalling.py
2. [Done] weirdo.py
3. [Start] brawny.py
4. [Start] elder.py
5. [Start] couch.py
6. [Start] rooster.py
7. [Start] prophetic.py
8. [Start] cuddle.py
9. [Start] hermit.py
10. [Start] hotshot.py

Trial Screen



```
x = [2, 8, 7, 9, -5, 0, 2]
x_between = []
for x_i in x:
    if (2 < x_i) and (x_i < 10):
        x_between.append(x_i)
print x_between

y = [1, -3, 10, 0, 8, 9, 1]
y_between = []
for y_i in y:
    if (-2 < y_i) and (y_i < 9):
        y_between.append(y_i)
print y_between

xy_common = []
for x_i in x:
    if x_i in y:
        xy_common.append(x_i)
print xy_common
```

Anatomy of a Trial



- Response proportion ≈ 0.5
- Keystroke coefficient = 4/2 = 2
  - Keystroke count = 4
  - True output characters = 2
- Grade = 10 (perfect)

Tobii TX300 Eye-Tracker

- Free-standing (no head mount, chin rest)
- 300 Hz (fixations ≥ 100 ms)

Programs (1/2)

10 categories, 2-3 versions each (25 total)

- `between` - filter two lists, intersection
  - `functions` - between/common in functions (**24 lines**)
  - `inline` - no functions (19 lines)
- `counting` - simple `for` loop with bug
  - `nospace` - no blank lines in loop body (3 lines)
  - `twospaces` - 2 blank lines in loop body (5 lines)
- `funcall` - simple function call with different values
  - `nospace` - calls on 1 line, no spaces (4 lines)
  - `space` - calls on 1 line, spaced out (4 lines)
  - `vars` - calls on 3 lines, different vars (7 lines)
- `overload` - overloaded + operator (number strings)
  - `multmixed` - numeric *, string + (11 lines)
  - `plusmixed` - numeric +, string + (11 lines)
  - `strings` - string + (11 lines)
- `partition` - partition list of numbers
  - `balanced` - odd number of items (5 lines)
  - `unbalanced` - even number of items (5 lines)
  - `unbalanced_pivot` - even number of items, pivot var (6 lines)

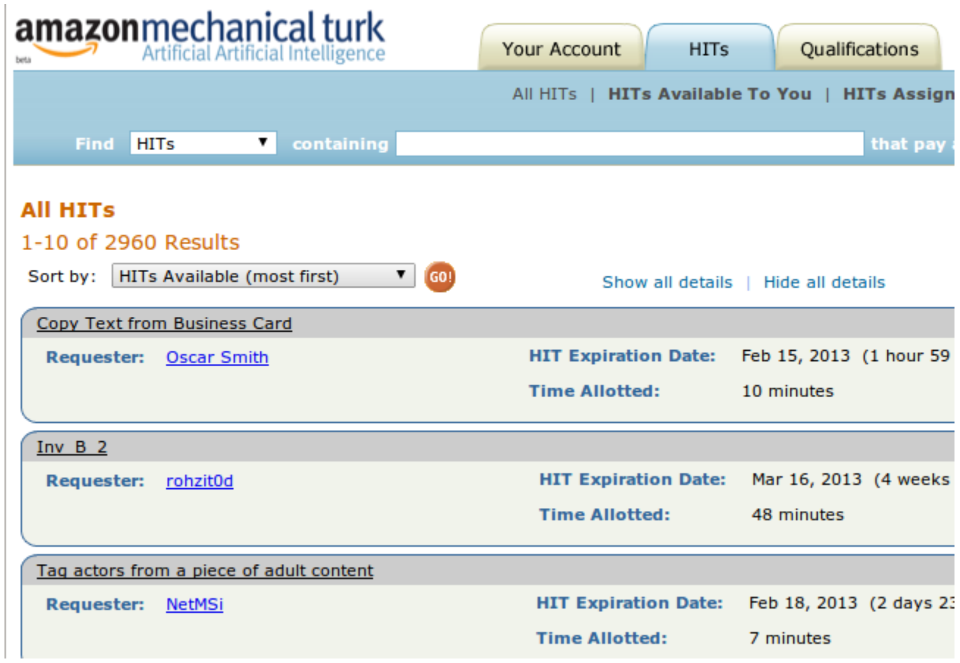
Programs (2/2)

10 categories, 2-3 versions each (25 total)

- `initvar` - summation and factorial
  - `bothbad` - bug in both (9 lines)
  - `good` - no bugs (9 lines)
  - `onebad` - bug in summation (9 lines)
- `order` - 3 simple functions called
  - `inorder` - call order = definition order (14 lines)
  - `shuffled` - call order $\neq$ definition order (14 lines)
- `rectangle` - compute area of 2 rectangles
  - `basic` - x,y,w,h in separate vars, area() in function (18 lines)
  - `class` - x,y,w,h,area() in class (21 lines)
  - `tuples` - x,y,w,h in tuples, area() in function (14 lines)
- `scope` - function calls with no effect
  - `diffname` - local/global var have same name (12 lines)
  - `samename` - local/global var have different name (12 lines)
- `whitespace` - simple linear equations
  - `linedup` - code is aligned on operators (14 lines)
  - `zigzag` - code is not aligned (14 lines)

Participants and Response Data

- 162 total participants
  - 29 Bloomington ($10)
  - 130 Mechanical Turk ($0.75)
  - 3 E-mail
- 1602 trials
  - 18 trials discarded


amazonmechanical turk
Artificial Artificial Intelligence
Your Account
HITs
Qualifications
All HITs | HITs Available To You | HITs Assign
Find HITs containing that pay
Sort by: HITs Available (most first)
GO!
Show all details | Hide all details

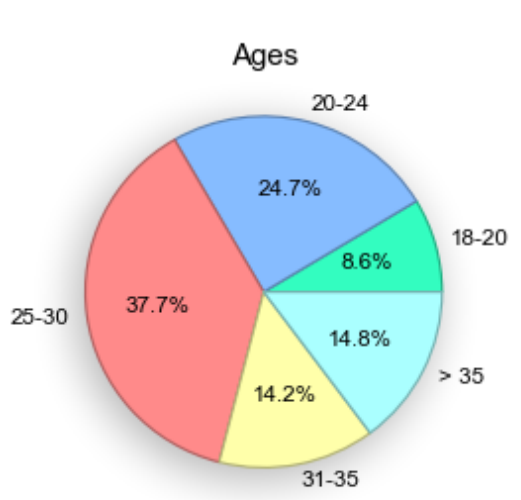
All HITs
1-10 of 2960 Results

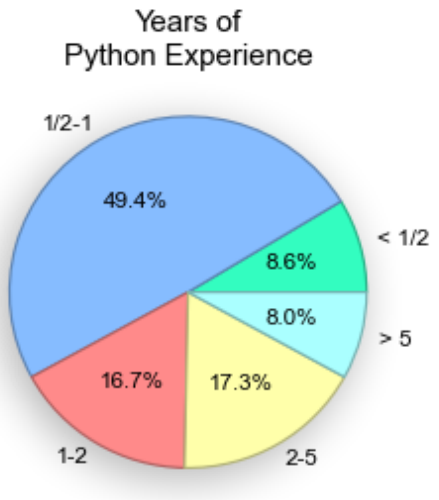
Copy Text from Business Card
Requester: Oscar Smith
HIT Expiration Date: Feb 15, 2013 (1 hour 59
Time Allotted: 10 minutes

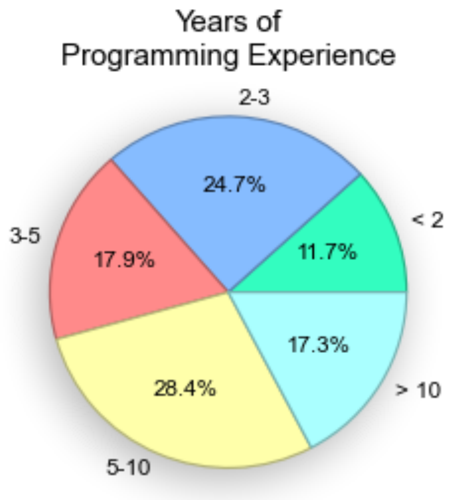
Inv_B_2
Requester: rohzit0d
HIT Expiration Date: Mar 16, 2013 (4 weeks
Time Allotted: 48 minutes

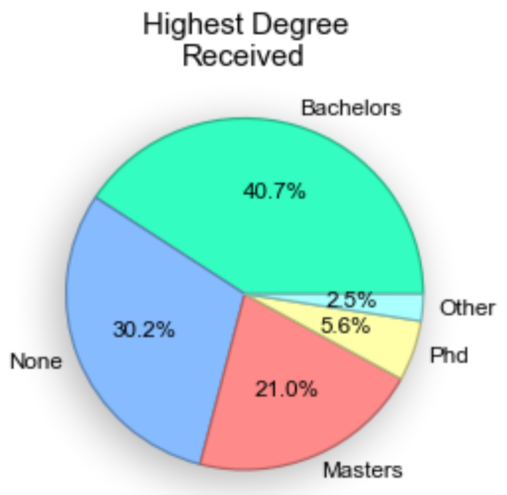
Tag actors from a piece of adult content
Requester: NetMSi
HIT Expiration Date: Feb 18, 2013 (2 days 23
Time Allotted: 7 minutes

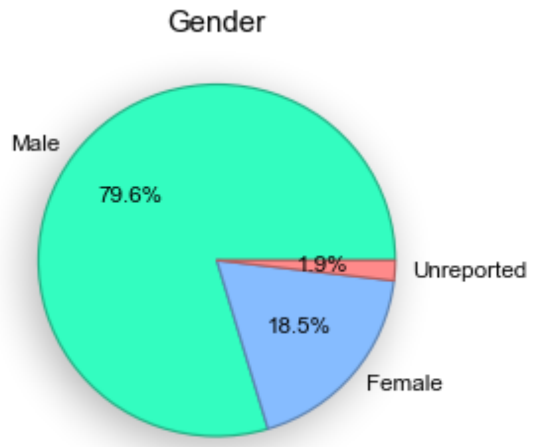
Demographics

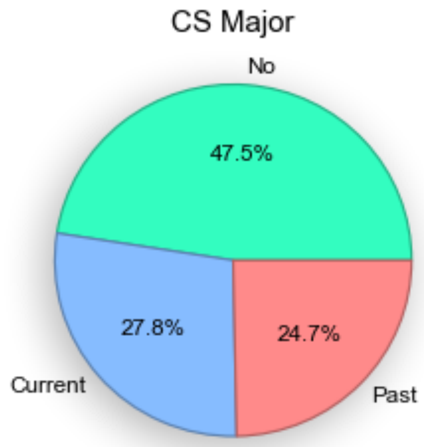












Grades

- 0 to 10 (perfect)
- ≥ 7 correct modulo formatting

```
print "1" + "2"
print 4 * 3
```

True Output

```
12
12
```

Correct (7)

```
"12",12
```

Common Error (4)

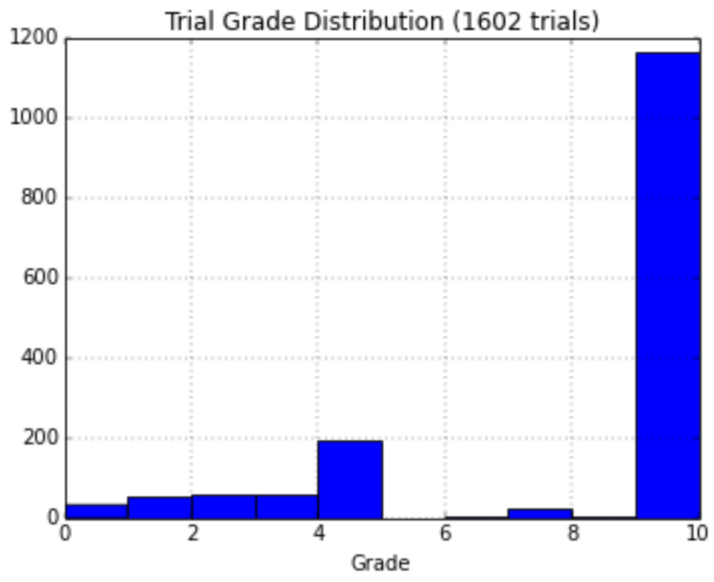
```
3
12
```

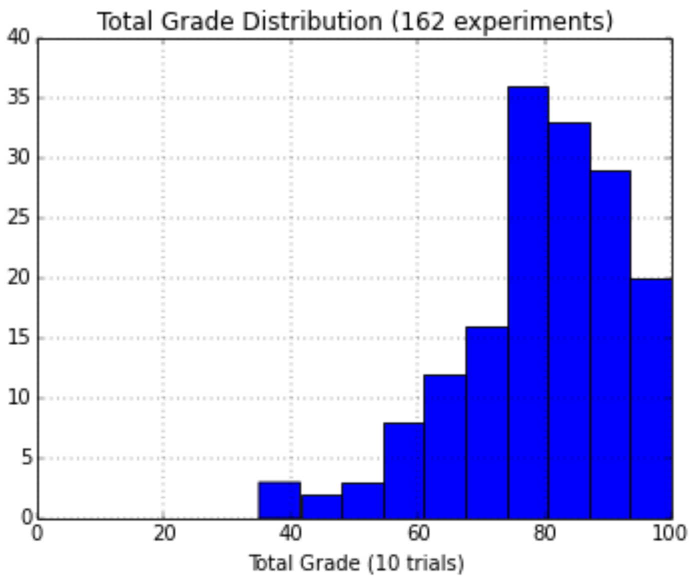
Incorrect (0)

```
barney
```

Grades

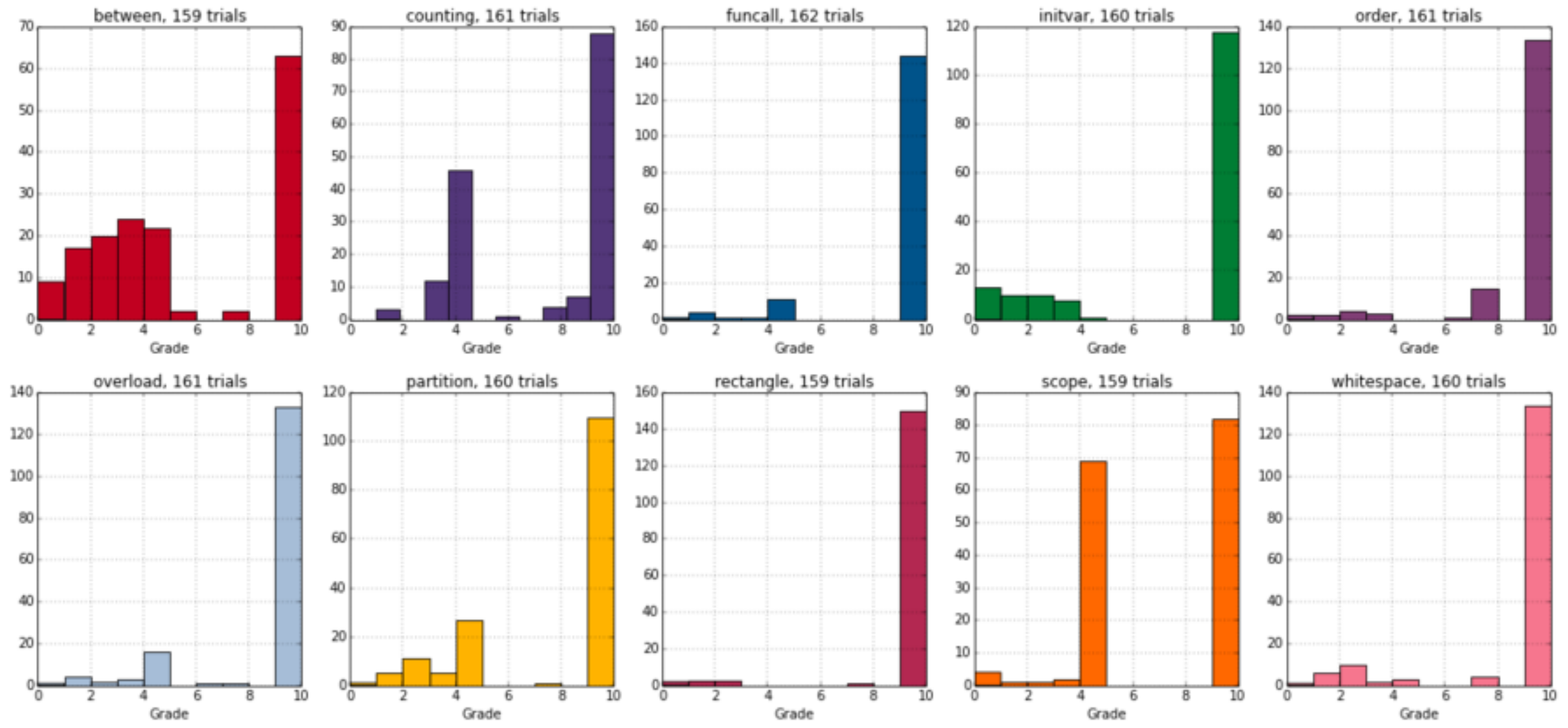
- 0 to 10 (perfect)
- ≥ 7 correct modulo formatting





- Median trial grade = 10
- Median experiment grade = 81

Grade Distributions by Program



scope - samename

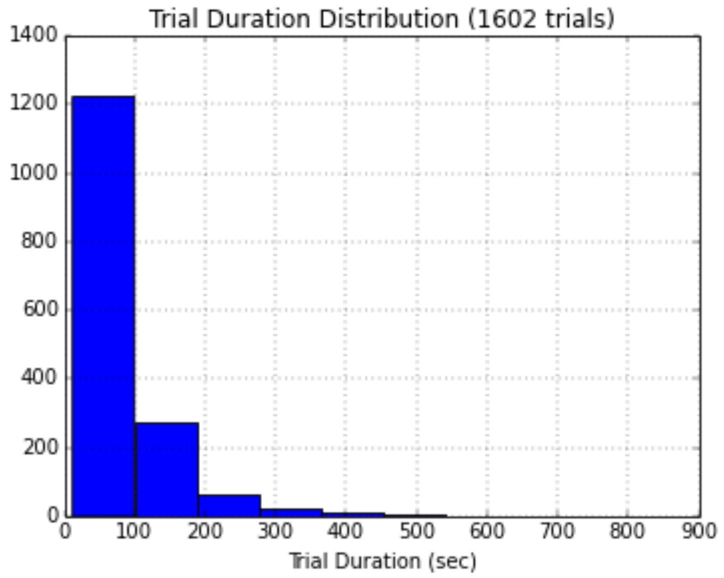
```
def add_1(added):
    added = added + 1

def twice(added):
    added = added * 2

added = 4
add_1(added)
twice(added)
add_1(added)
twice(added)
print added
```

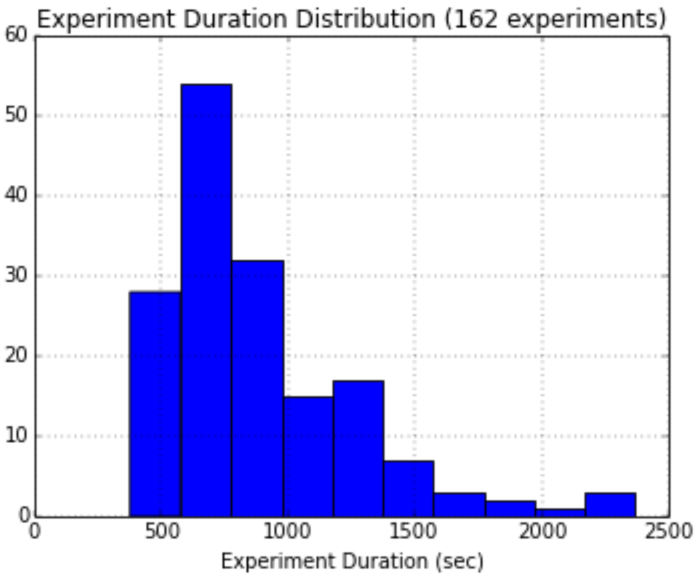
scope - diffname

```
def add_1(num):
    num = num + 1

def twice(num):
    num = num * 2

added = 4
add_1(added)
twice(added)
add_1(added)
twice(added)
print added
```

Trial Duration

- 45 minutes for entire experiment
- No time limit on individual trials

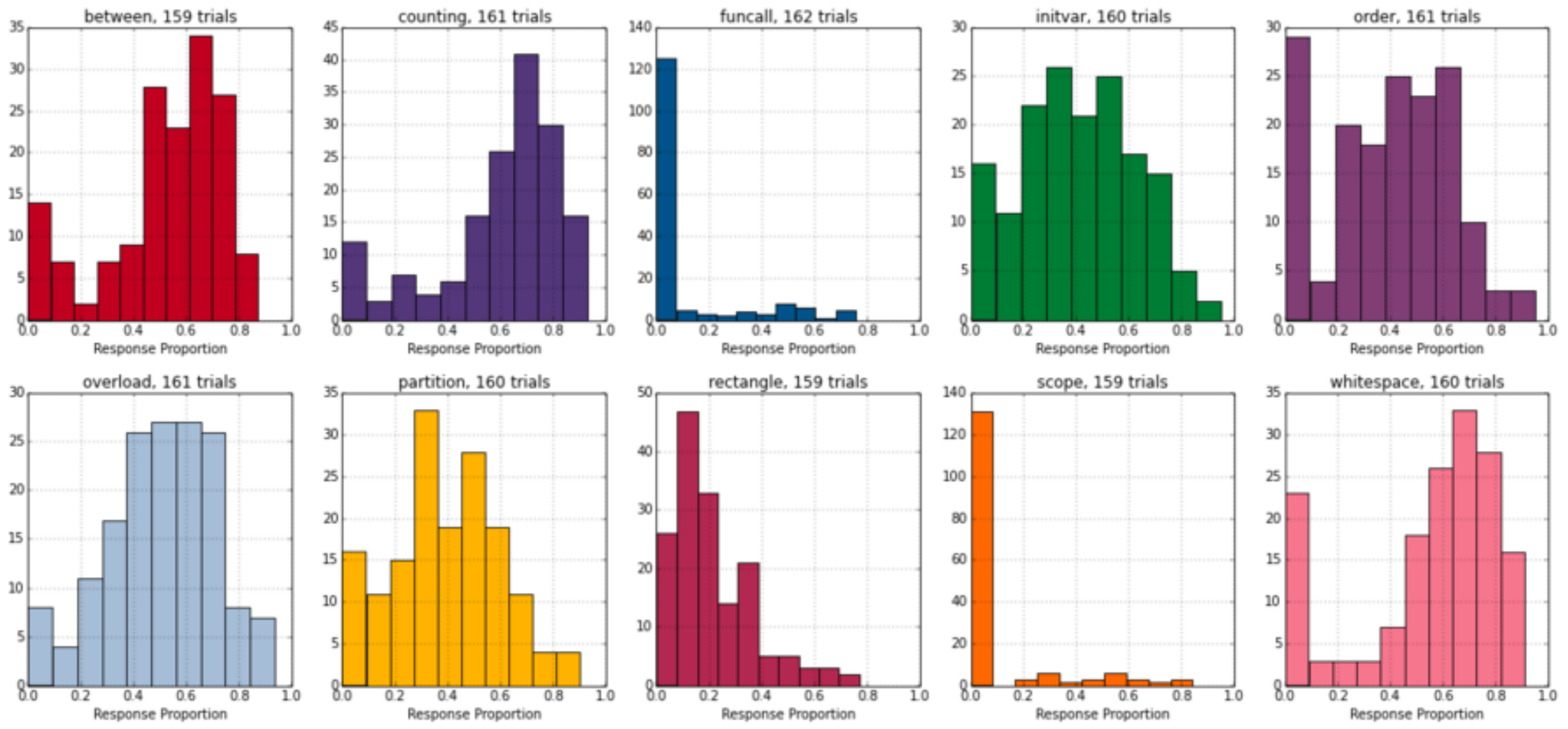




- Median trial duration: 55 sec
- Median experiment duration: 773 sec (12.9 min)

Response Proportions by Program

- Time spent responding / trial time



between - functions

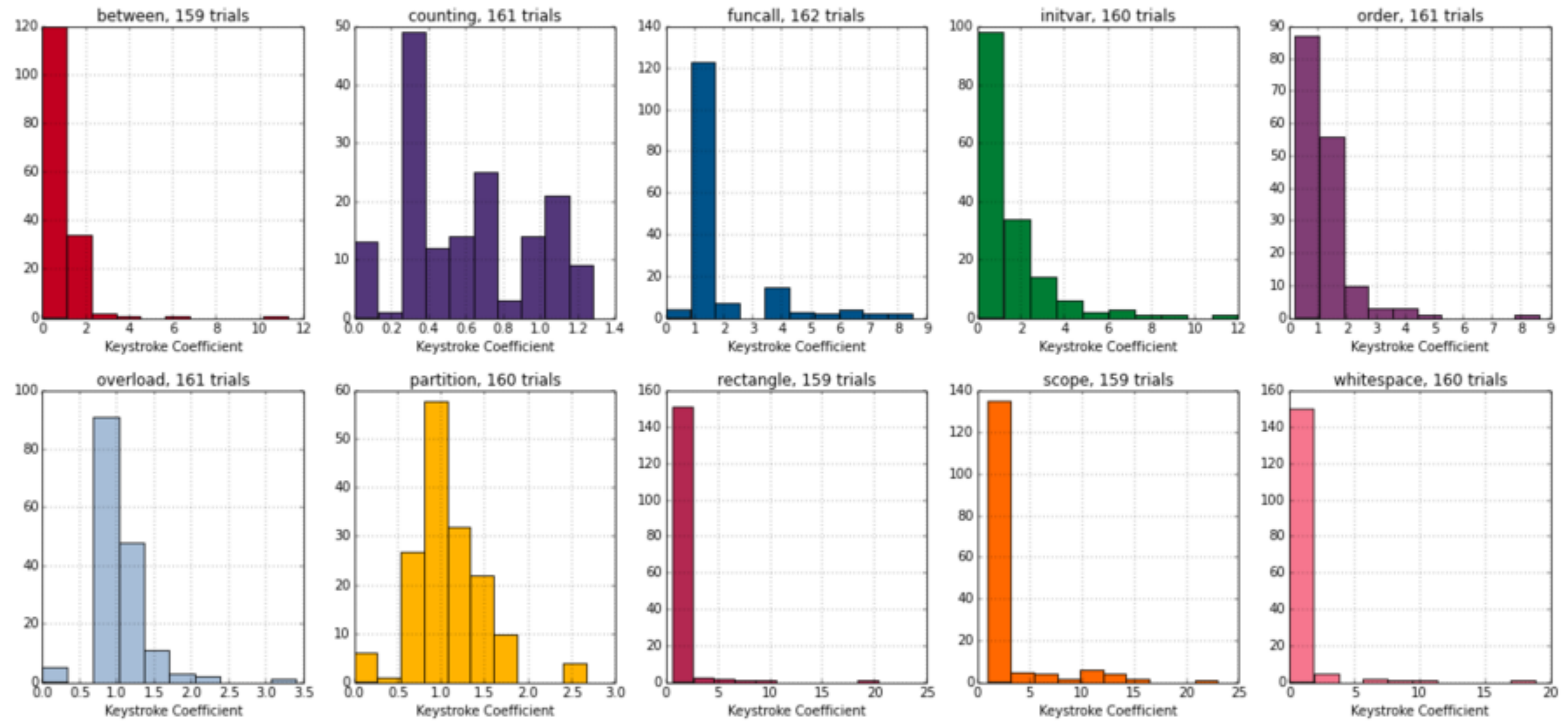
```
def between(numbers, low, high):
    winners = []
    for num in numbers:
        if (low < num) and (num < high):
            winners.append(num)
    return winners

def common(list1, list2):
    winners = []
    for item1 in list1:
        if item1 in list2:
            winners.append(item1)
    return winners

x = [2, 8, 7, 9, -5, 0, 2]
x_btwn = between(x, 2, 10)
print x_btwn

y = [1, -3, 10, 0, 8, 9, 1]
y_btwn = between(y, -2, 9)
print y_btwn

xy_common = common(x, y)
print xy_common
```

between - inline

```
x = [2, 8, 7, 9, -5, 0, 2]
x_between = []
for x_i in x:
    if (2 < x_i) and (x_i < 10):
        x_between.append(x_i)
print x_between

y = [1, -3, 10, 0, 8, 9, 1]
y_between = []
for y_i in y:
    if (-2 < y_i) and (y_i < 9):
        y_between.append(y_i)
print y_between

xy_common = []
for x_i in x:
    if x_i in y:
        xy_common.append(x_i)
print xy_common
```

Keystroke Coefficient

- Number of keystrokes / characters in true output
- > 1 is less efficient



counting - nospace

```
for i in [1, 2, 3, 4]:
    print "The count is", i
    print "Done counting"
```

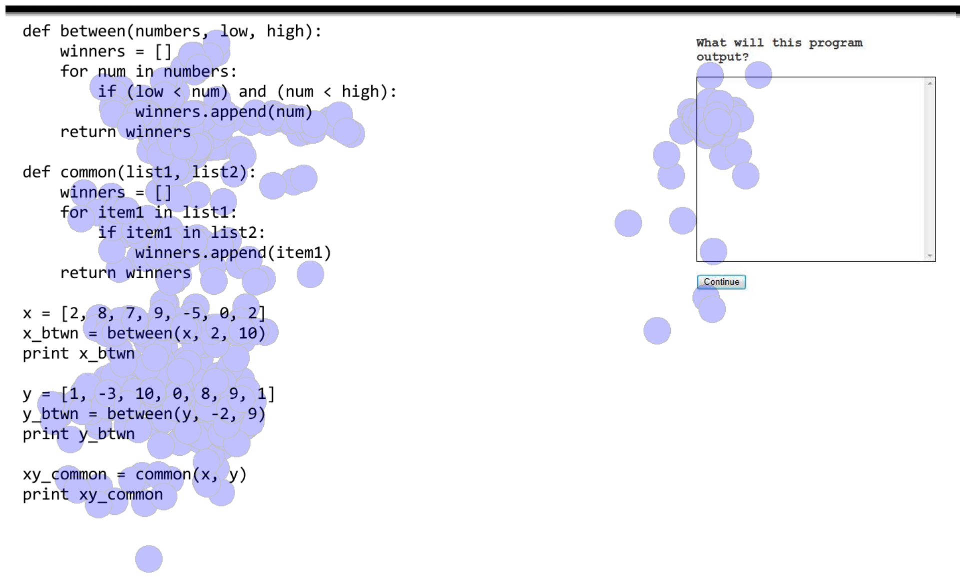
counting - twospaces

```
for i in [1, 2, 3, 4]:
    print "The count is", i


    print "Done counting"
```

Eye-Tracking Analysis

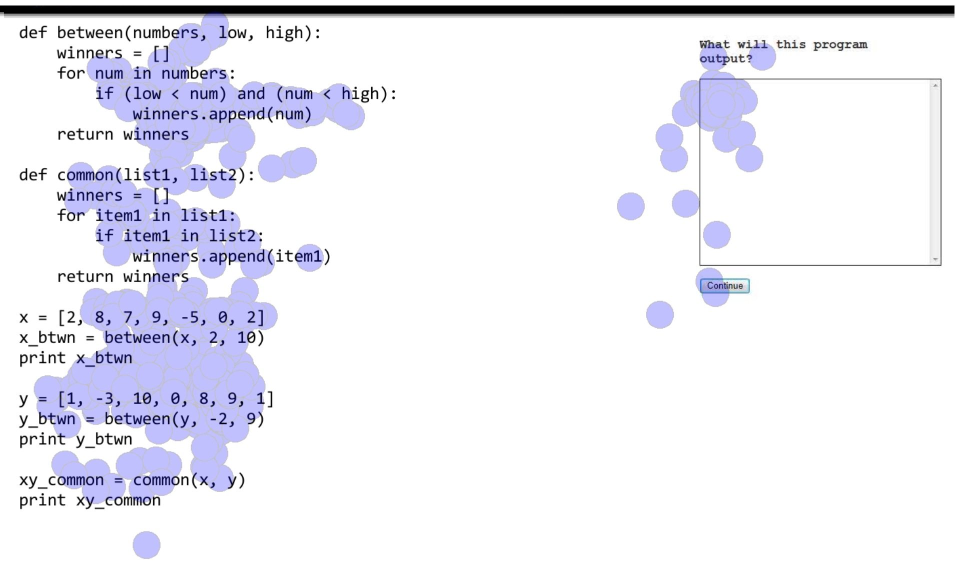
- 29 participants, 290 trials
- About $5\,\frac{1}{2}$ hours of video
- Fixations + saccades, corrected manually by experiment

Uncorrected



```
def between(numbers, low, high):
    winners = []
    for num in numbers:
        if (low < num) and (num < high):
            winners.append(num)
    return winners

def common(list1, list2):
    winners = []
    for item1 in list1:
        if item1 in list2:
            winners.append(item1)
    return winners

x = [2, 8, 7, 9, -5, 0, 2]
x_btwn = between(x, 2, 10)
print x_btwn

y = [1, -3, 10, 0, 8, 9, 1]
y_btwn = between(y, -2, 9)
print y_btwn

xy_common = common(x, y)
print xy_common
```

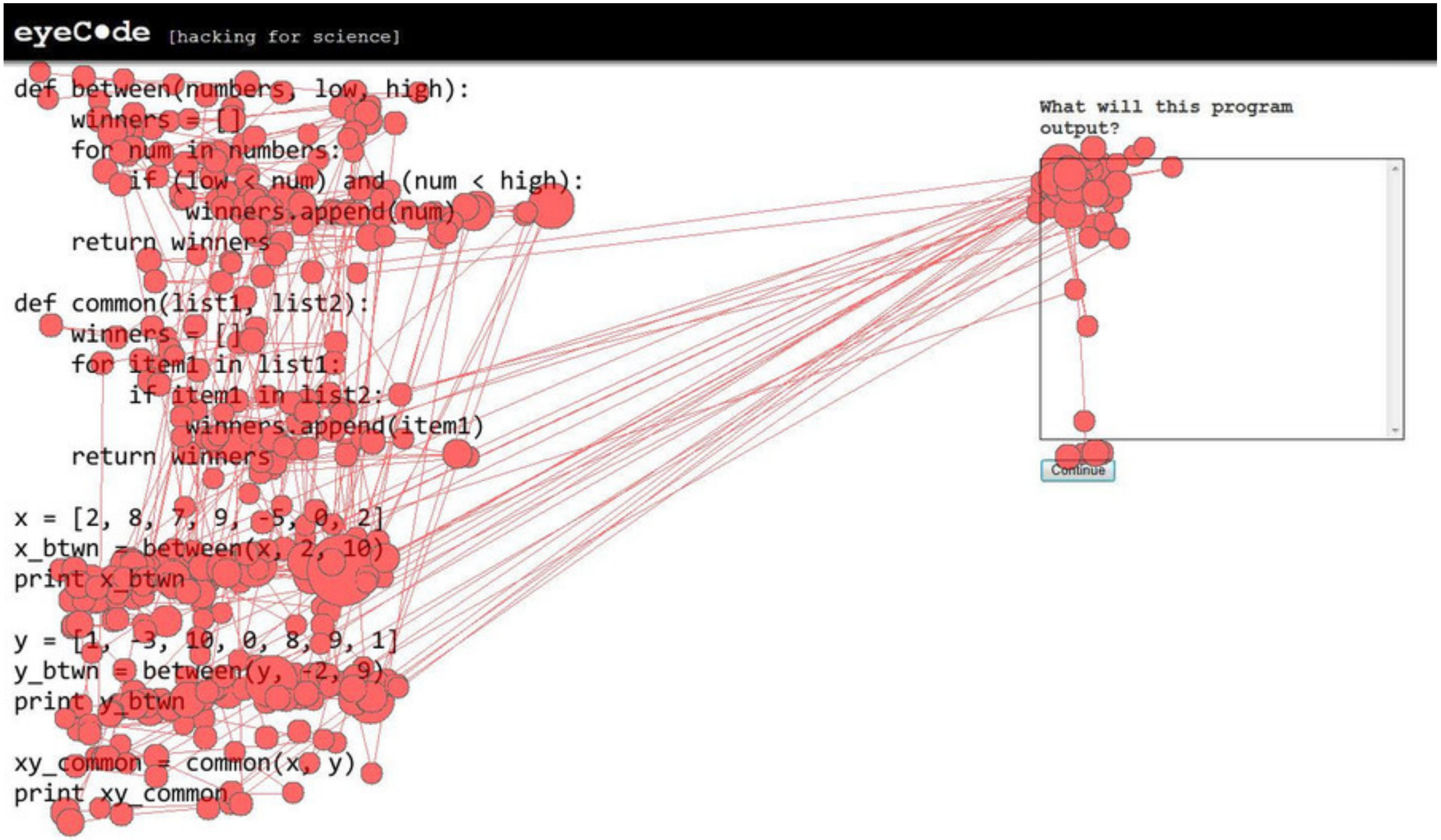
Corrected



```
def between(numbers, low, high):
    winners = []
    for num in numbers:
        if (low < num) and (num < high):
            winners.append(num)
    return winners

def common(list1, list2):
    winners = []
    for item1 in list1:
        if item1 in list2:
            winners.append(item1)
    return winners

x = [2, 8, 7, 9, -5, 0, 2]
x_btwn = between(x, 2, 10)
print x_btwn

y = [1, -3, 10, 0, 8, 9, 1]
y_btwn = between(y, -2, 9)
print y_btwn

xy_common = common(x, y)
print xy_common
```

Fixations and Areas of Interest

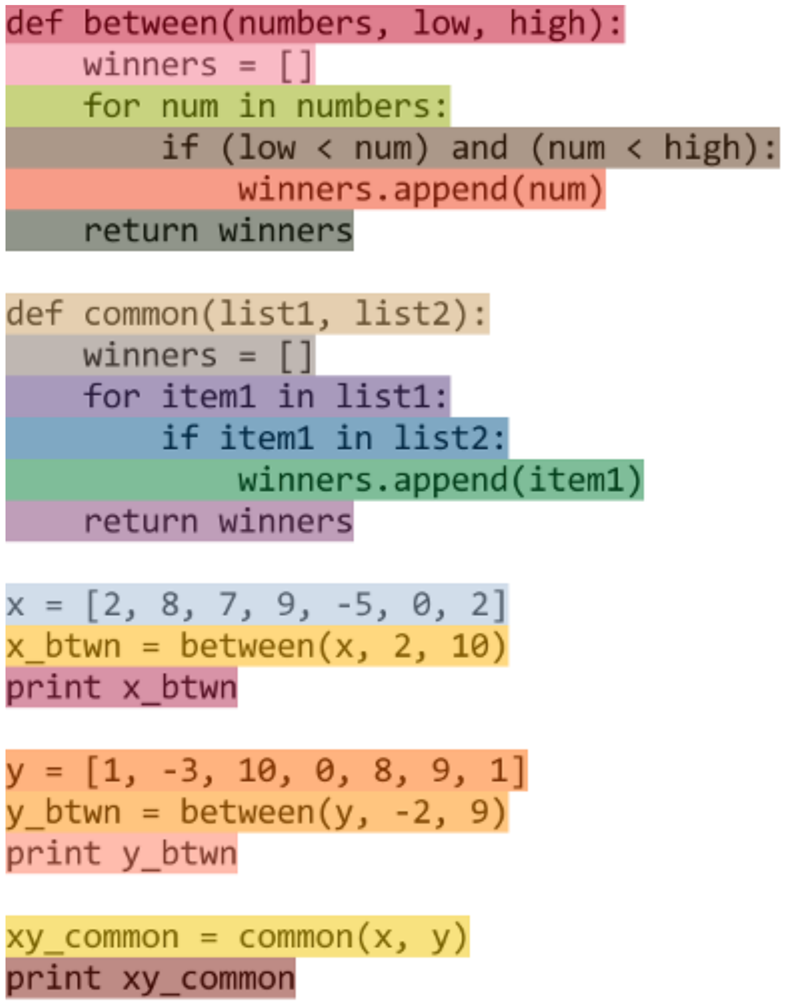
- Need to quantize fixation positions



```
def between(numbers, low, high):
    winners = []
    for num in numbers:
        if (low < num) and (num < high):
            winners.append(num)
    return winners

def common(list1, list2):
    winners = []
    for item1 in list1:
        if item1 in list2:
            winners.append(item1)
    return winners

x = [2, 8, 7, 9, -5, 0, 2]
x_btwn = between(x, 2, 10)
print x_btwn

y = [1, -3, 10, 0, 8, 9, 1]
y_btwn = between(y, -2, 9)
print y_btwn

xy_common = common(x, y)
print xy_common
```

Line-based AOIs

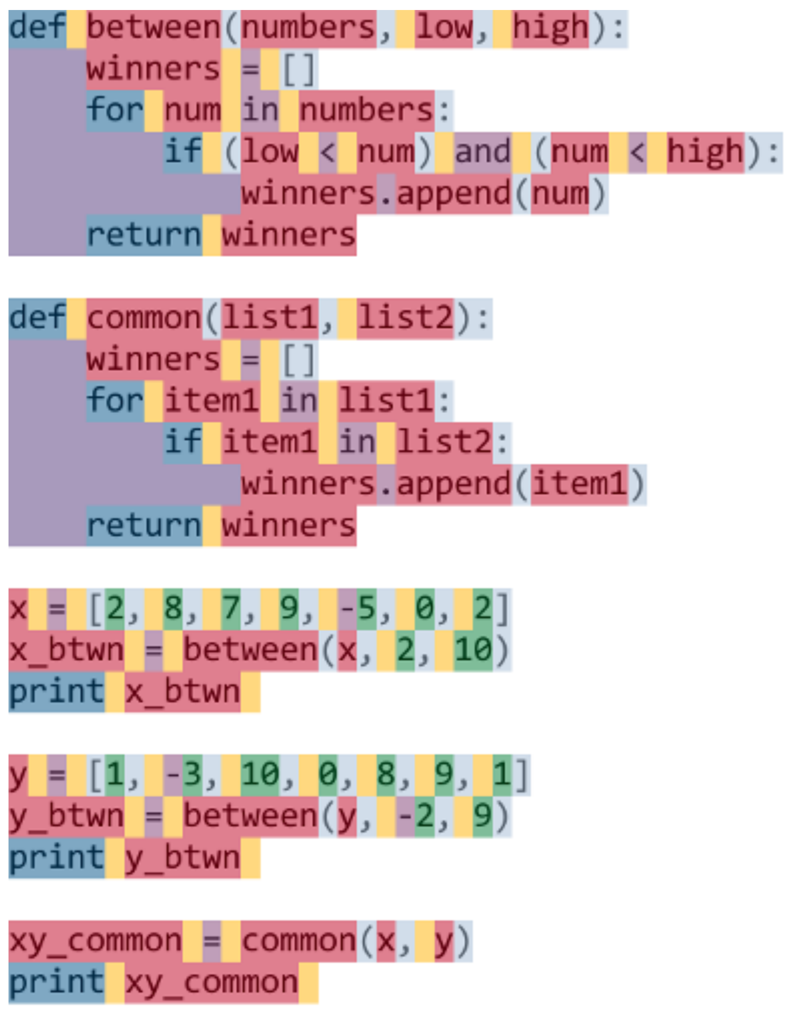
- Indentation is part of line AOI



```
def between(numbers, low, high):
    winners = []
    for num in numbers:
        if (low < num) and (num < high):
            winners.append(num)
    return winners

def common(list1, list2):
    winners = []
    for item1 in list1:
        if item1 in list2:
            winners.append(item1)
    return winners

x = [2, 8, 7, 9, -5, 0, 2]
x_btwn = between(x, 2, 10)
print x_btwn

y = [1, -3, 10, 0, 8, 9, 1]
y_btwn = between(y, -2, 9)
print y_btwn

xy_common = common(x, y)
print xy_common
```

Syntax-based AOIs

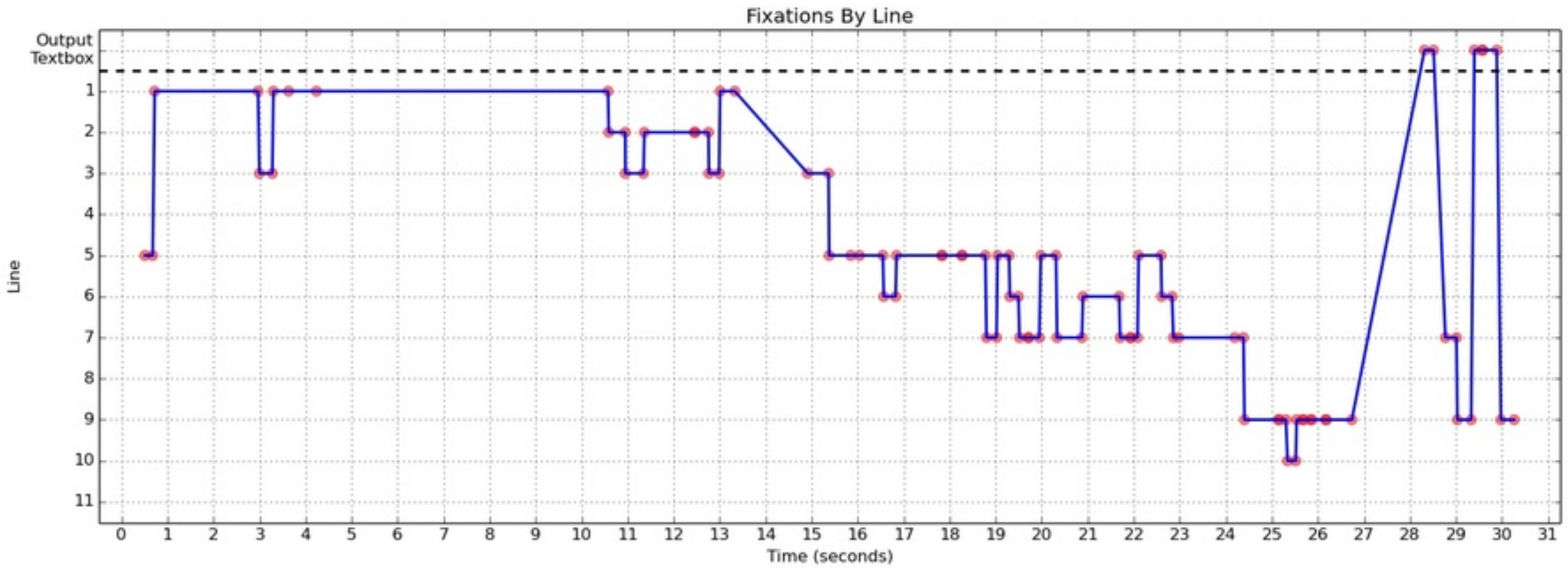
- Current data is too noisy to use syntax AOIs



```
def between(numbers, low, high):
    winners = []
    for num in numbers:
        if (low < num) and (num < high):
            winners.append(num)
    return winners

def common(list1, list2):
    winners = []
    for item1 in list1:
        if item1 in list2:
            winners.append(item1)
    return winners

x = [2, 8, 7, 9, -5, 0, 2]
x_btwn = between(x, 2, 10)
print x_btwn

y = [1, -3, 10, 0, 8, 9, 1]
y_btwn = between(y, -2, 9)
print y_btwn

xy_common = common(x, y)
print xy_common
```

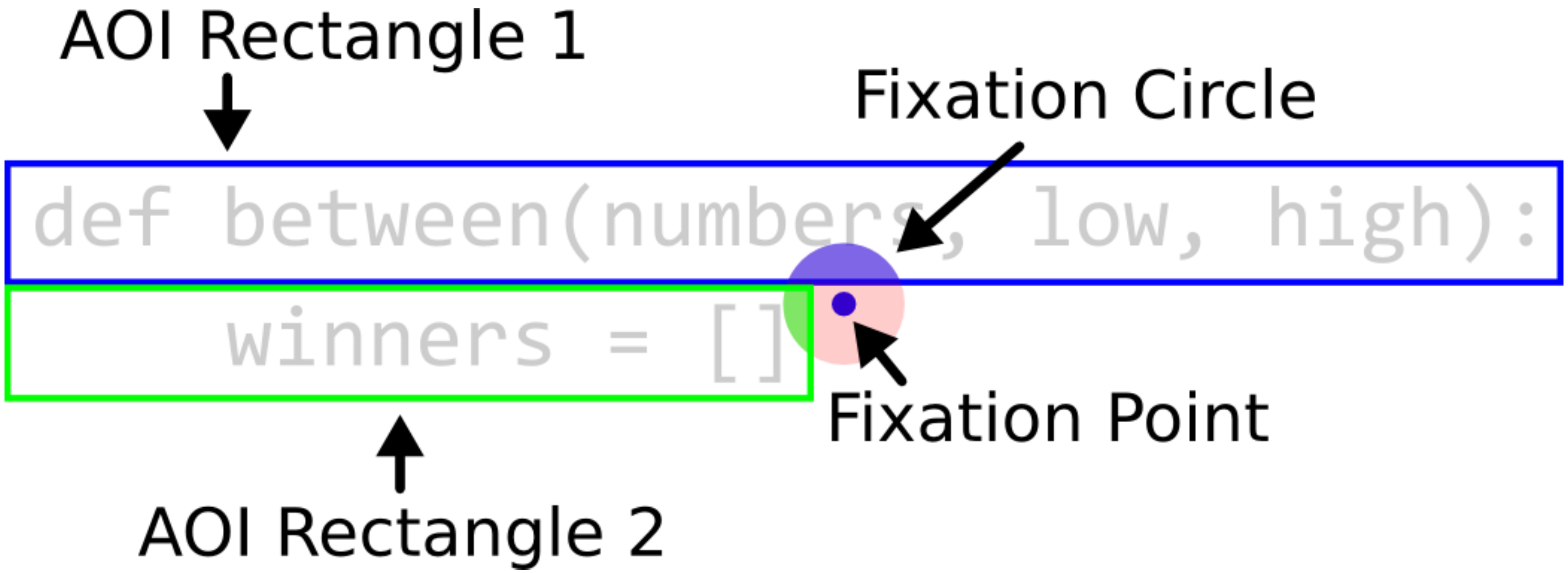
Fixations and Areas of Interest

- By line and output box



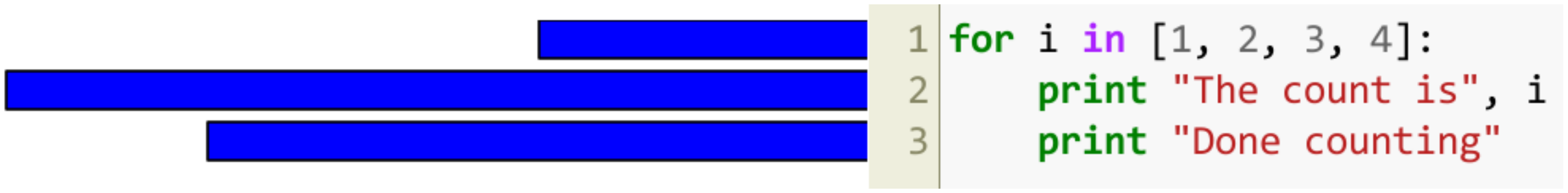
Hit Testing

- AOI with largest area overlap wins



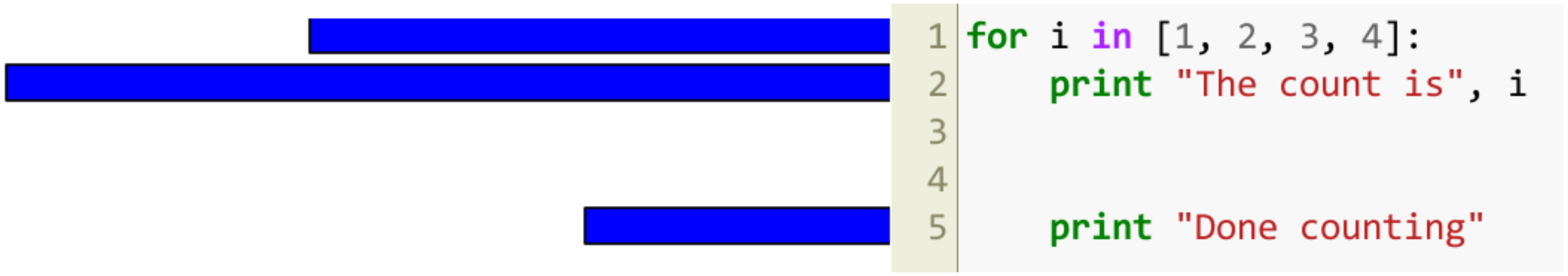
Fixation Times by Line

- Proportions of total fixation times (all participants)



```
for i in [1, 2, 3, 4]:
    print "The count is", i
    print "Done counting"
```

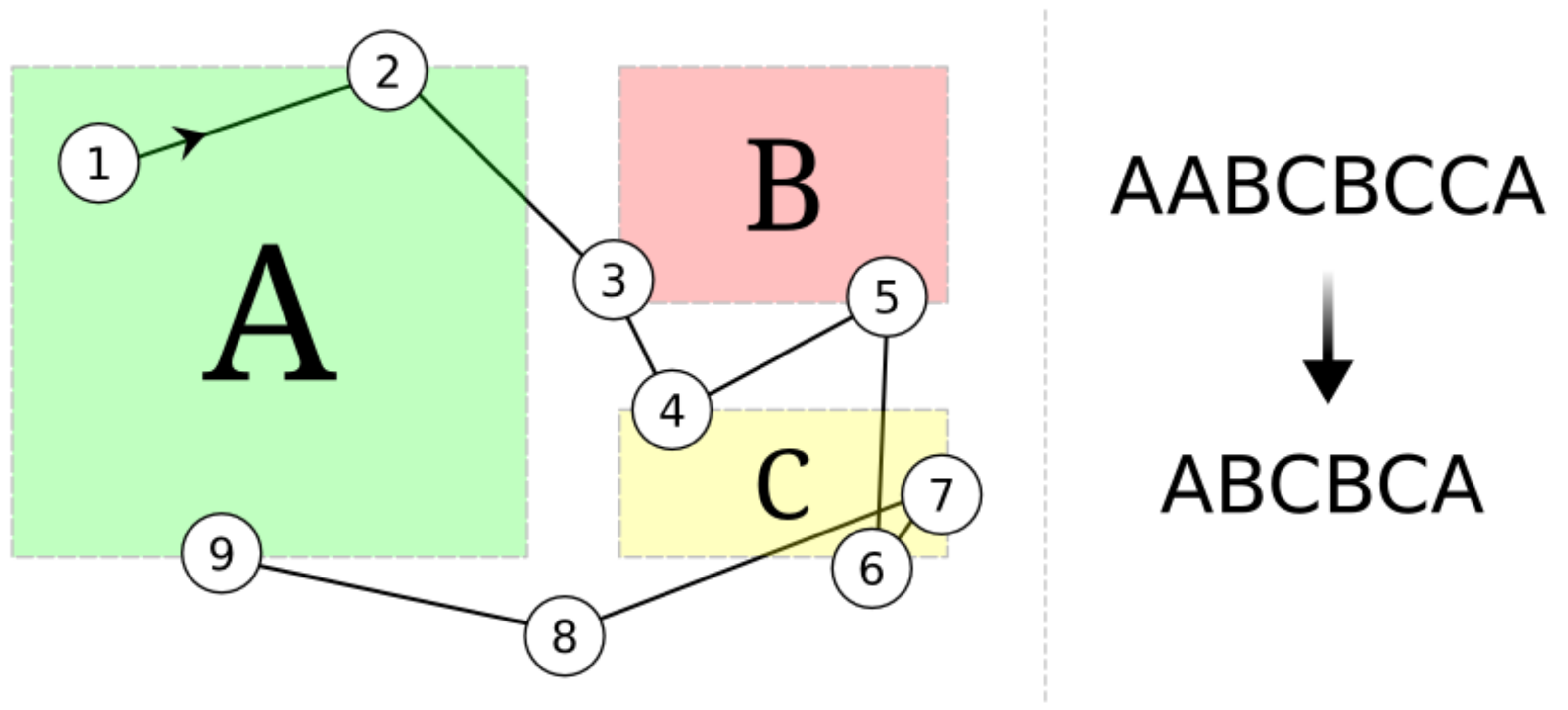
- Median grade = 10



```
for i in [1, 2, 3, 4]:
    print "The count is", i


    print "Done counting"
```

- Median grade = 4

Scanpath Comparisons

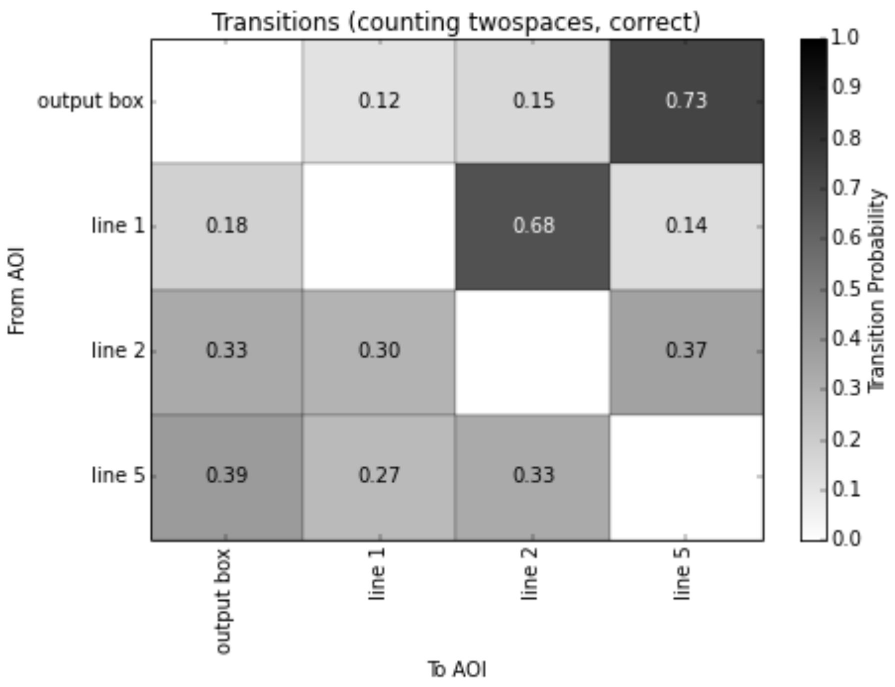


- Levenshtein distance (string edit distance)
- Needleman-Wunsch (DNA sequence matching)

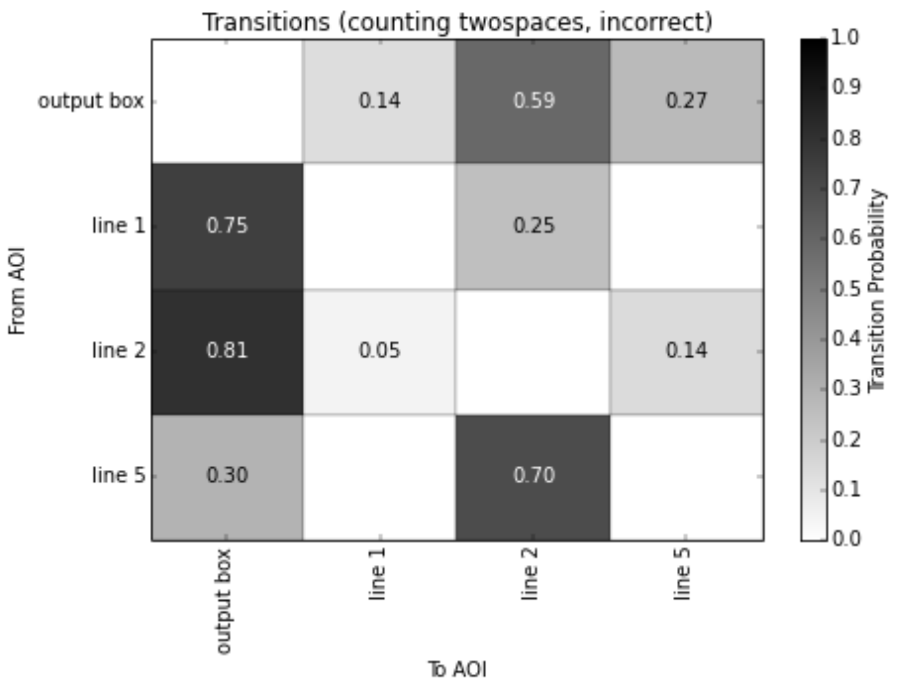
AOI Transitions

```
1    for i in [1, 2, 3, 4]:
2        print "The count is", i
3
4
5        print "Done counting"
```

Correct Trials



Incorrect Trials



Future Work

- Collect more data
    - New programs
    - Chin rest for eye-tracker
- Codify eye movements $\longrightarrow$ participant strategies
    - Differences between experts and novices
    - Implications for programming education
- Model comprehension process
    - Qualitative theories to computational model
    - Active vision model with procedural/declarative/spatial memory

Thank you!