CFlow: Supporting **Semantic Flow** Analysis of Students' Code in Programming Problems **at Scale**

Ashley Ge Zhang [1], Xiaohang Tang[2], Steve Oney [1], Yan Chen [2]

[1] University of Michigan
[2] Virginia Tech




Programming Exercises

Problem

For each word in **`words`**, add 'd' to the end of the word if the word ends in 'e' to make it past tense. Otherwise, add 'ed' to make it past tense. Save these past tense words to a list called **`past_tense`**.

Submissions

73

```
words = ["adopt", "bake", "beam", "confide", "gri
past_tense = []

for word in words:
    if word[-1] == "e":
        past_tense.append(word+"d")
    else:
        past_tense.append(word+"ed")
```

60

```
words = ["adopt", "bake", "beam", "confide", "gri
past_tense = []

for word in words:
    if word[-1] == 'e':
        past_tense.append(word + 'd')
    else:
        past_tense.append(word + 'ed')
```

12

```
words = ["adopt", "bake", "beam", "confide", "gri
past_tense = []
for word in words:
    if word[-1] == "e":
        word = word + "d"
    else:
        word = word + "ed"
    past_tense.append(word)
```

1

```
words = ["adopt", "bake", "beam", "confide", "gri
past_tense=[]
for i in words :
     n=len(i)
     if i[n-1] == 'e':
        past_tense+=i+"e"
     else :
        past_tense+=i+"ed"
print(past_tense)
```

Not pass unit test

11

```
words = ["adopt", "bake", "beam", "confide", "gri
past_tense = []
```

11

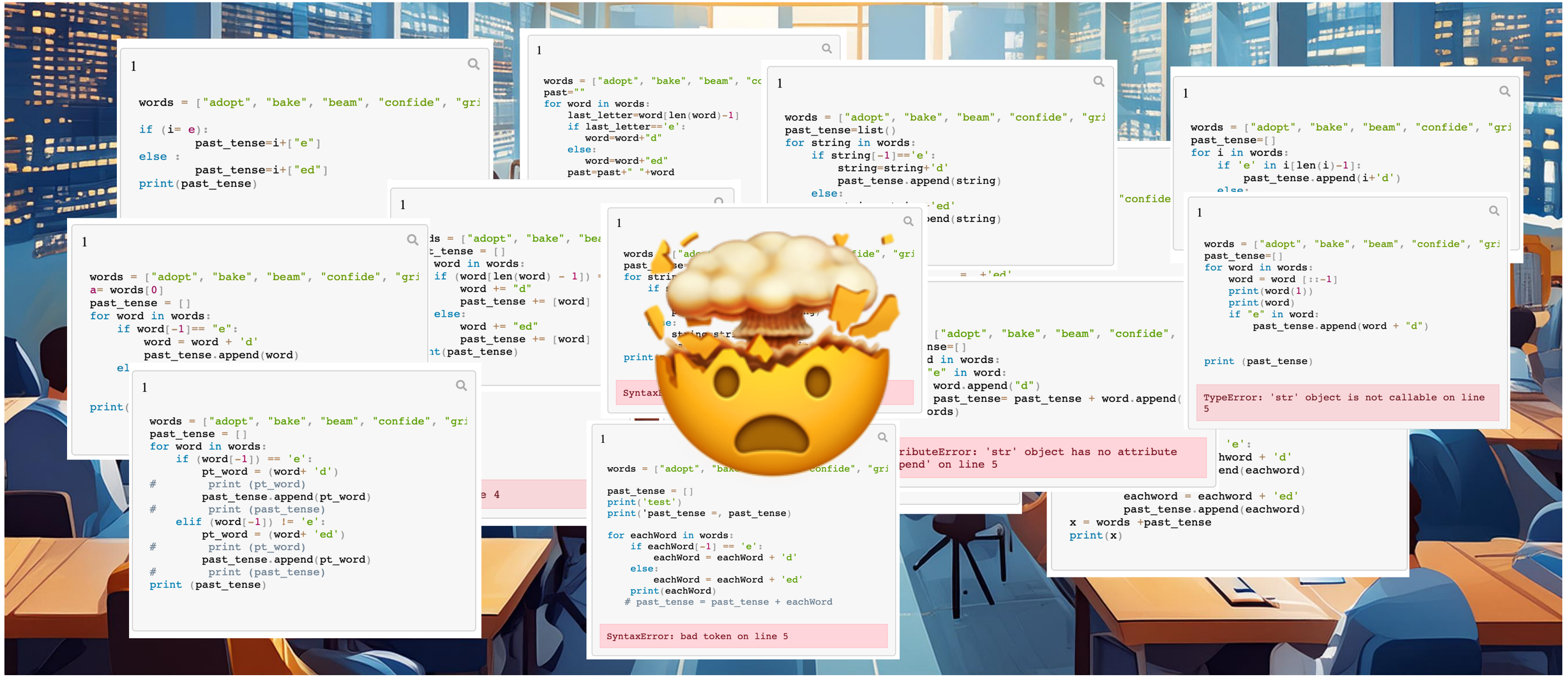
```
words = ["adopt", "bake", "beam", "confide", "gri
past_tense = []
```

Benefits

- Hands-on opportunities to practice
- Evaluate students' understandings

Introductory Programming Courses are Large

Viewing Students' Code is A Pain



Different Levels of Variation

Approach

```
words = [“egg”, “apple”, “tomato”]
e_words = []

for w in words:
    if w[-1]==“e”:
        e_words.append(w)
```

```
words = [“egg”, “apple”, “tomato”]
e_words = [w for w in words if w[-1]==“e"]
```

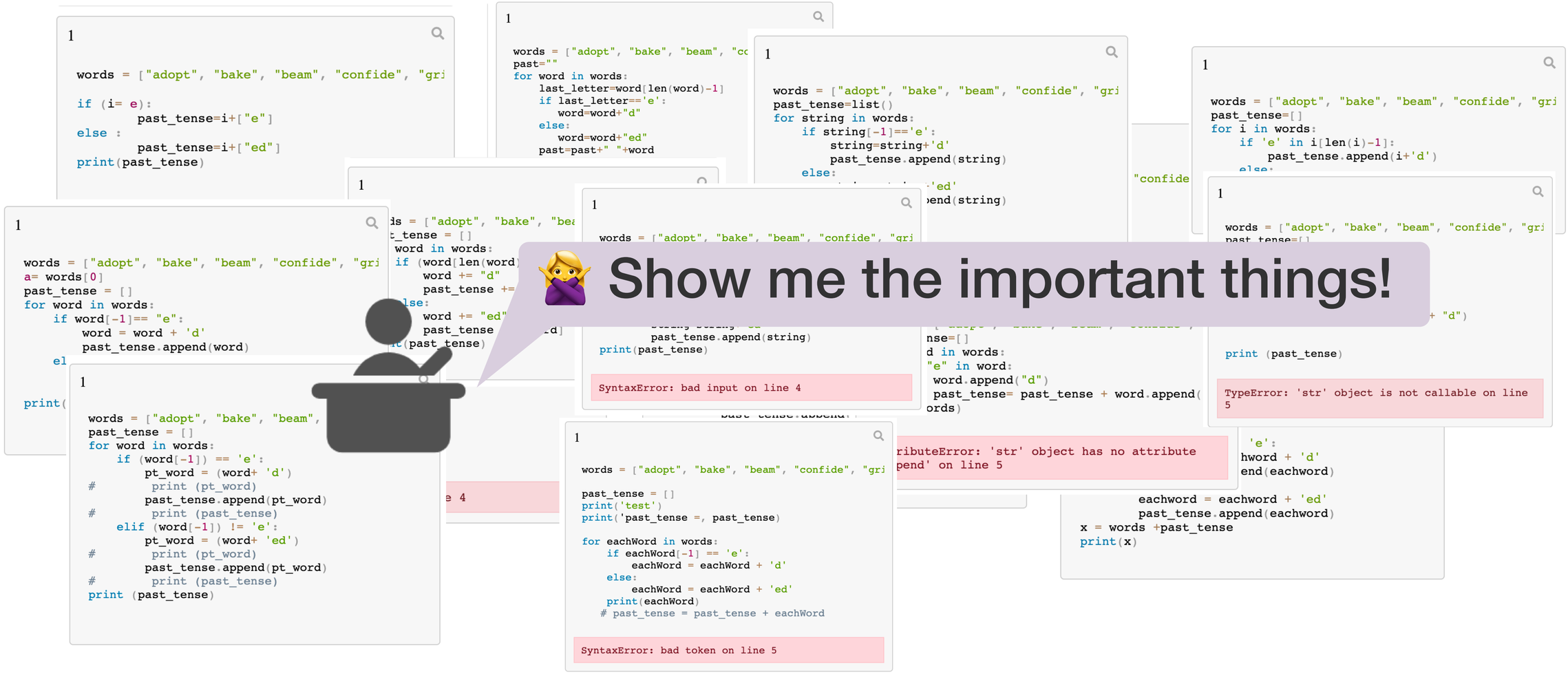
Different Levels of Variation

Code Line

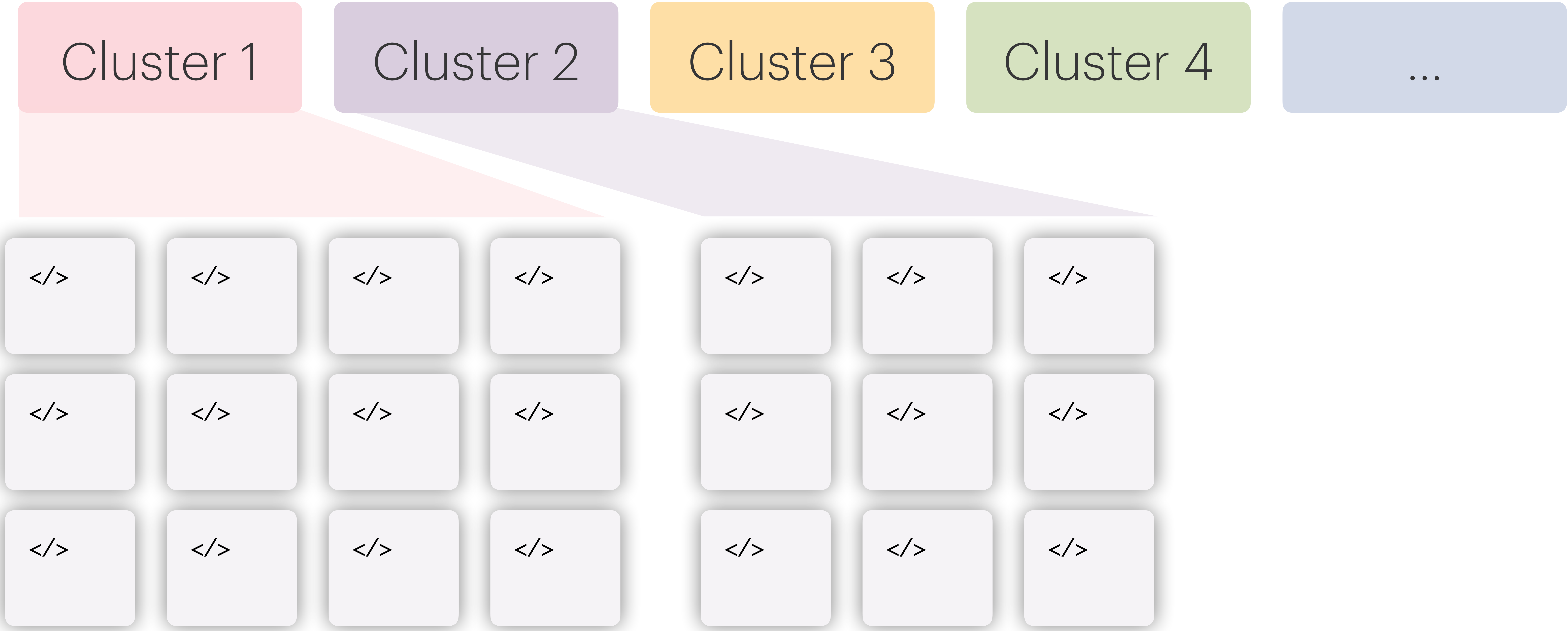
```
words = [“egg”, “apple”, “tomato”]
e_words = []

for w in words:
    if word[len(word)] == ‘e’:
        e_words.append(w)
```

```
words = [“egg”, “apple”, “tomato”]
e_words = []

for w in words:
    if word[-1] == ‘e’:
        e_words.append(w)
```

How to Simplify Students' Code?

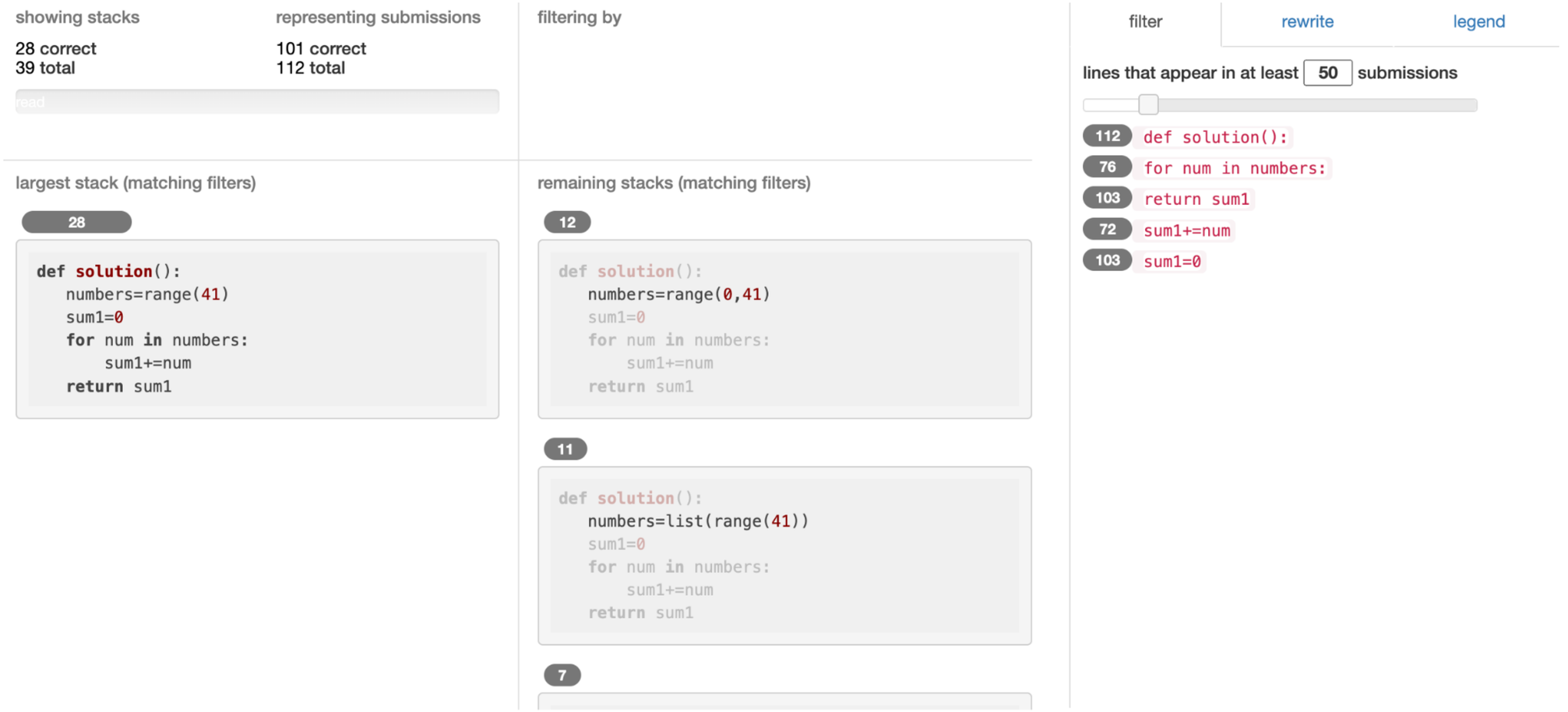


Code Clustering

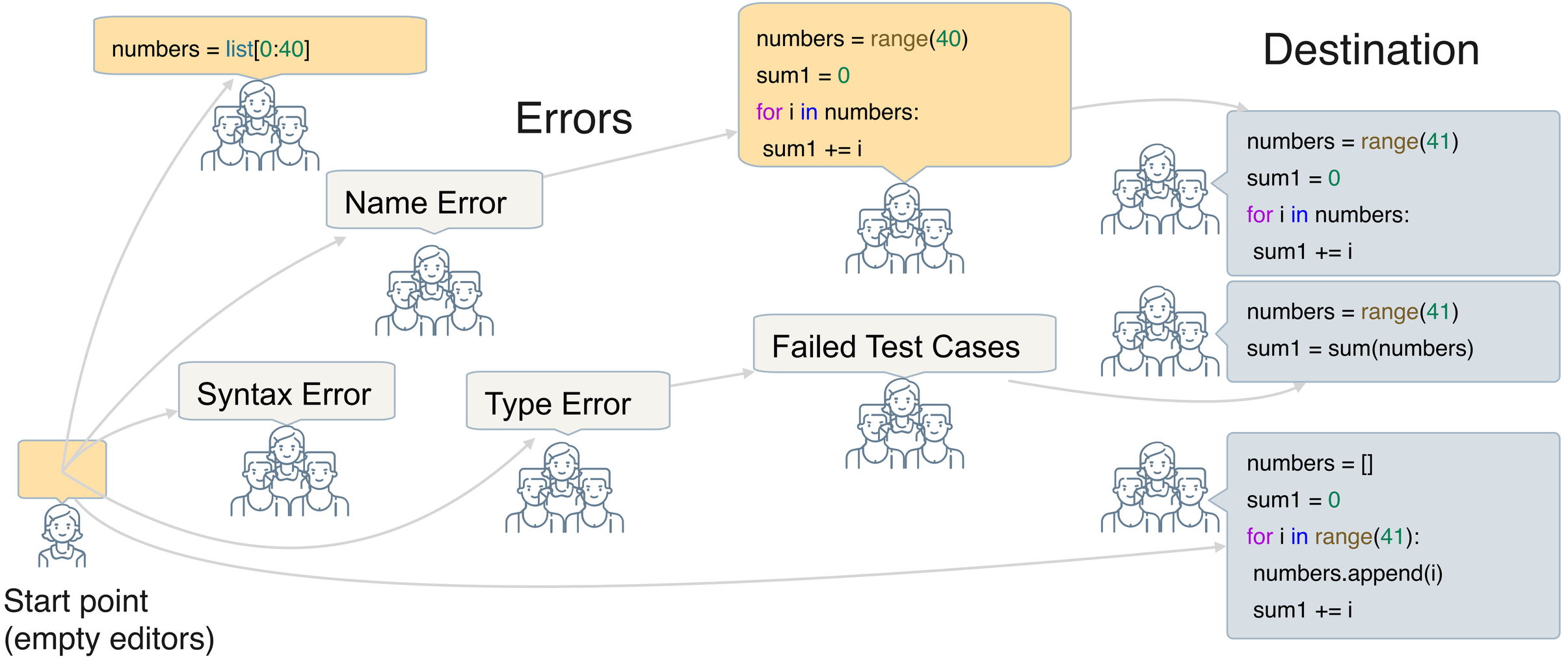


Related Work

OverCode Glassman et al., TOCHI 15

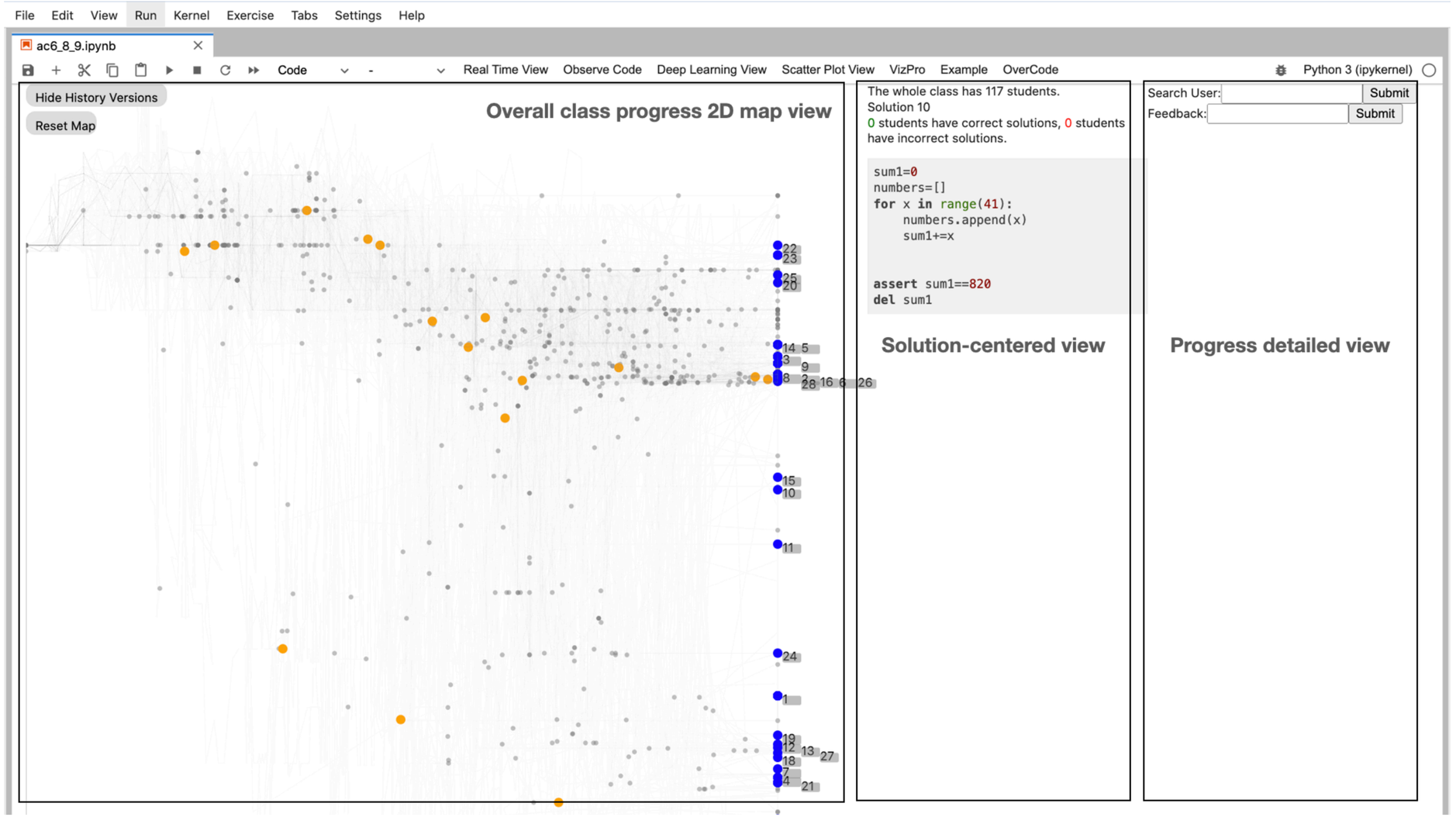


Mapping Code to A Spatial Area



VizProg

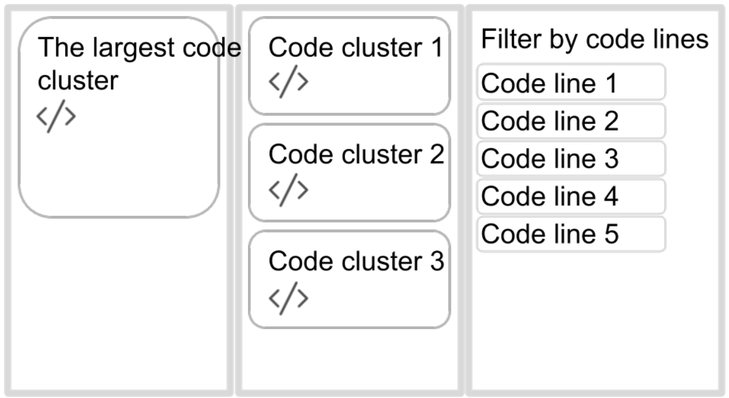
Zhang et al., CHI 23



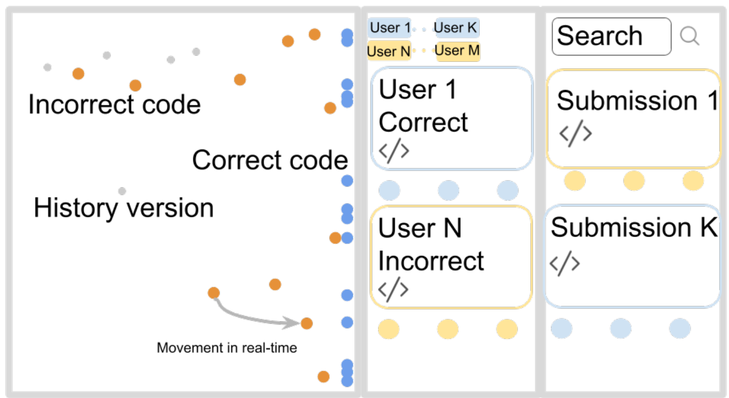
Actively Searching for Patterns

Providing a General View

OverCode



VizProg

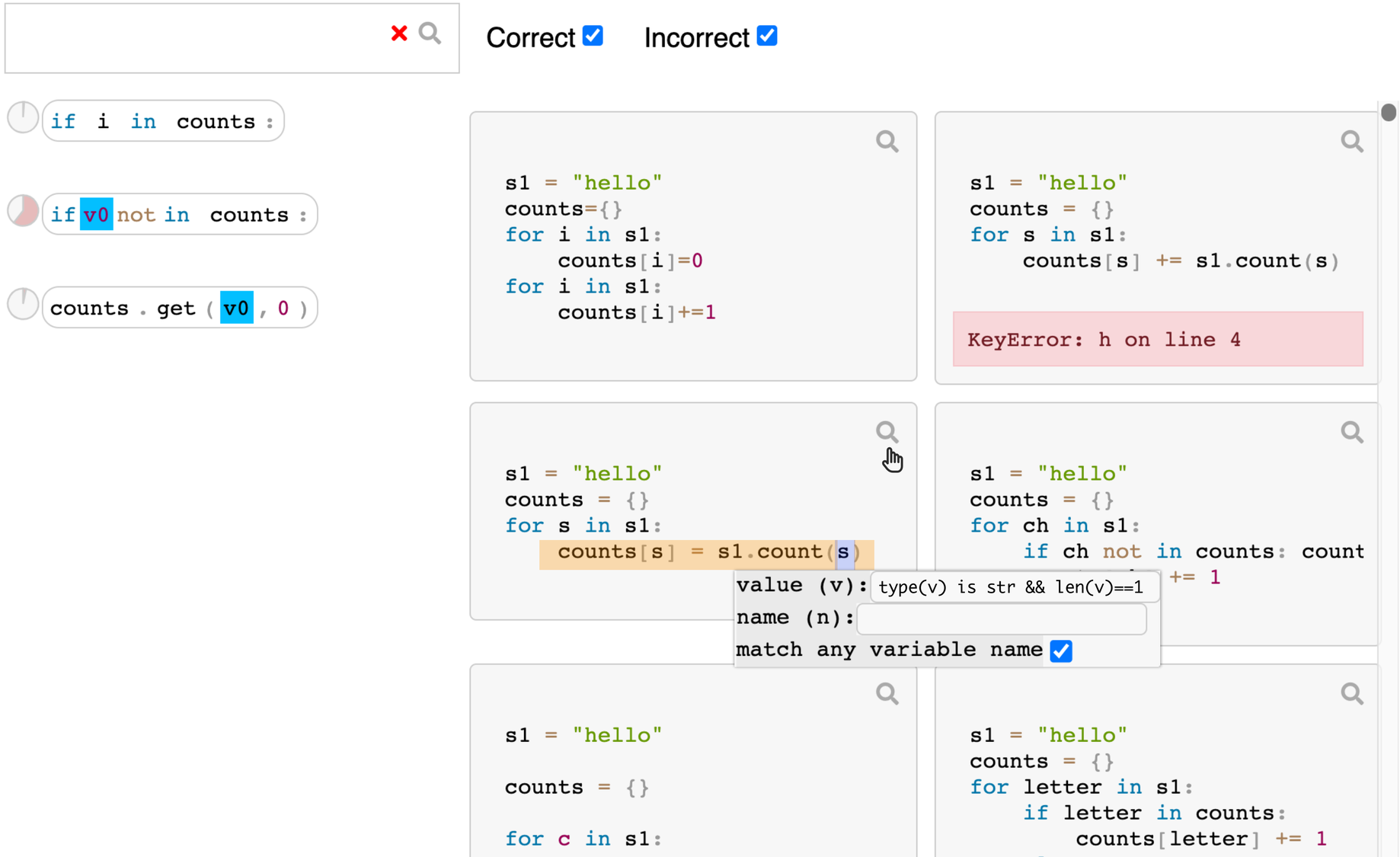


Code Search

Related Work

RunEx

Zhang et al., VL/HCC 23



Gaps

‼ Different levels of variation is not considered ‼

Gaps

‼ Different levels of variation is not considered ‼

Entire code sample

```
words = [“egg”, “apple”, “tomato”]
e_words = []

for w in words:
    if word[len(word)] == ‘e’:
        e_words.append(w)
```

```
words = [“egg”, “apple”, “tomato”]
e_words = []

for w in words:
    if word[-1] == ‘e’:
        e_words.append(w)
```

Nuanced difference

Too many clusters

Gaps

!! Different levels of variation is not considered !!

Isolated code statements

```
for i in enumerate(words):
```

```
if ‘e’ in word:
```

```
if word[len(word)] == ‘e’:
```

```
word.append(‘ed’)
```

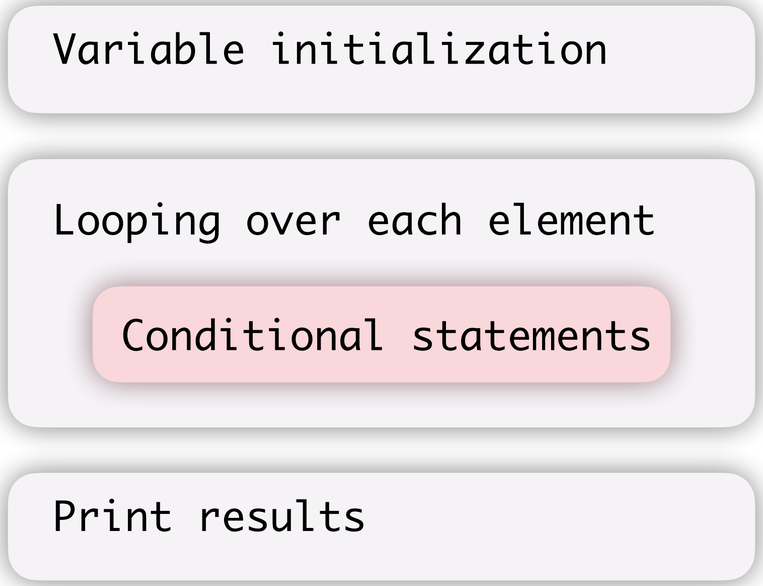
Focused too much on a single statement

↓

Lose information on its context

Semantic Flow

It’s easier to understand code with context.



This is how students approach the problem.

Semantic Flow

A single code statement

```
for a of mylist:
```

```
for a of mylist
```

```
mystr = ‘a”
```

Misconception at **syntax level**

Semantic flow

```
Variable initialization
```

```
Looping over each element
    Conditional statements
```

```
Print results
```

More complex misconceptions,
e.g. **logical problems**

Semantic flow is helpful.
But viewing thousands of them can be overwhelming!

Aggregate Semantic Flows at Scale

Thousands of code samples

A single code framework

Initialize variables

Loop over words

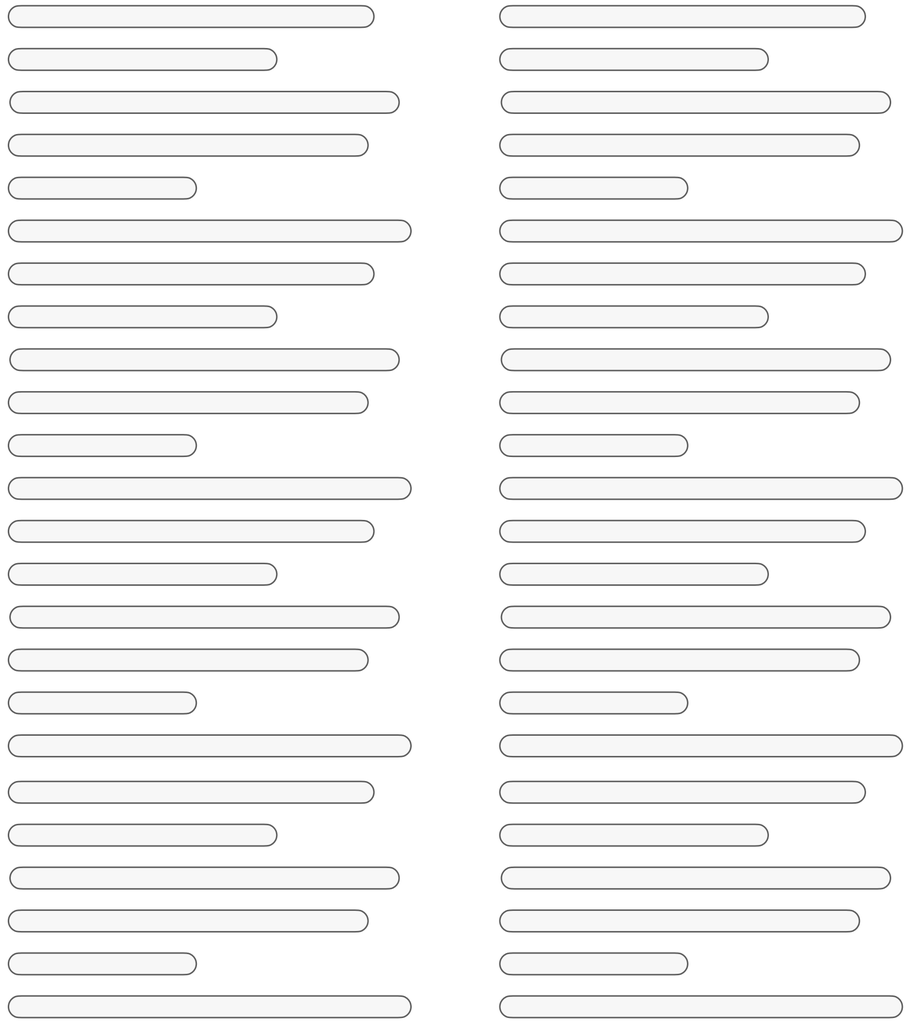
Check the end of the word

Modify the word

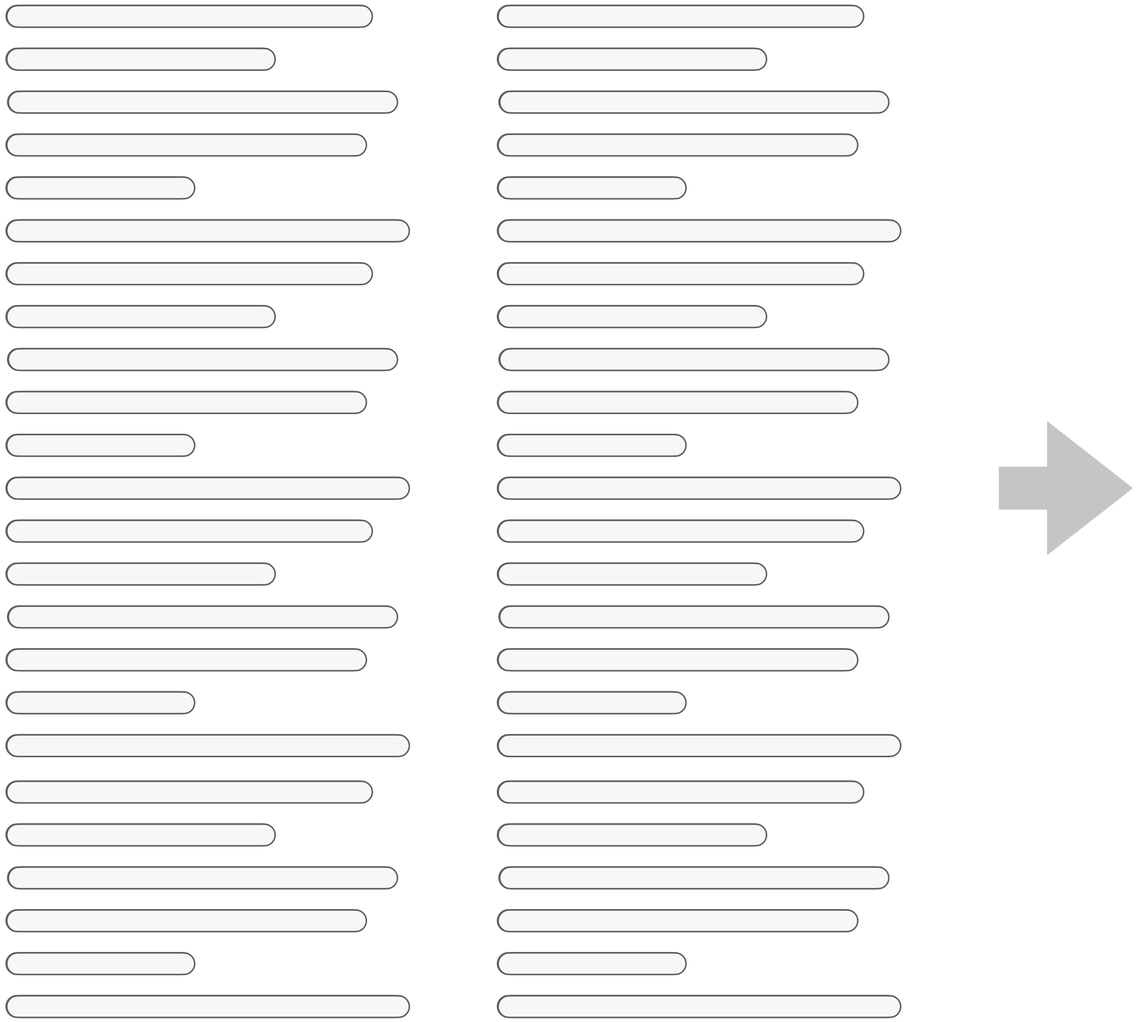
Append the new word to a new list

Other

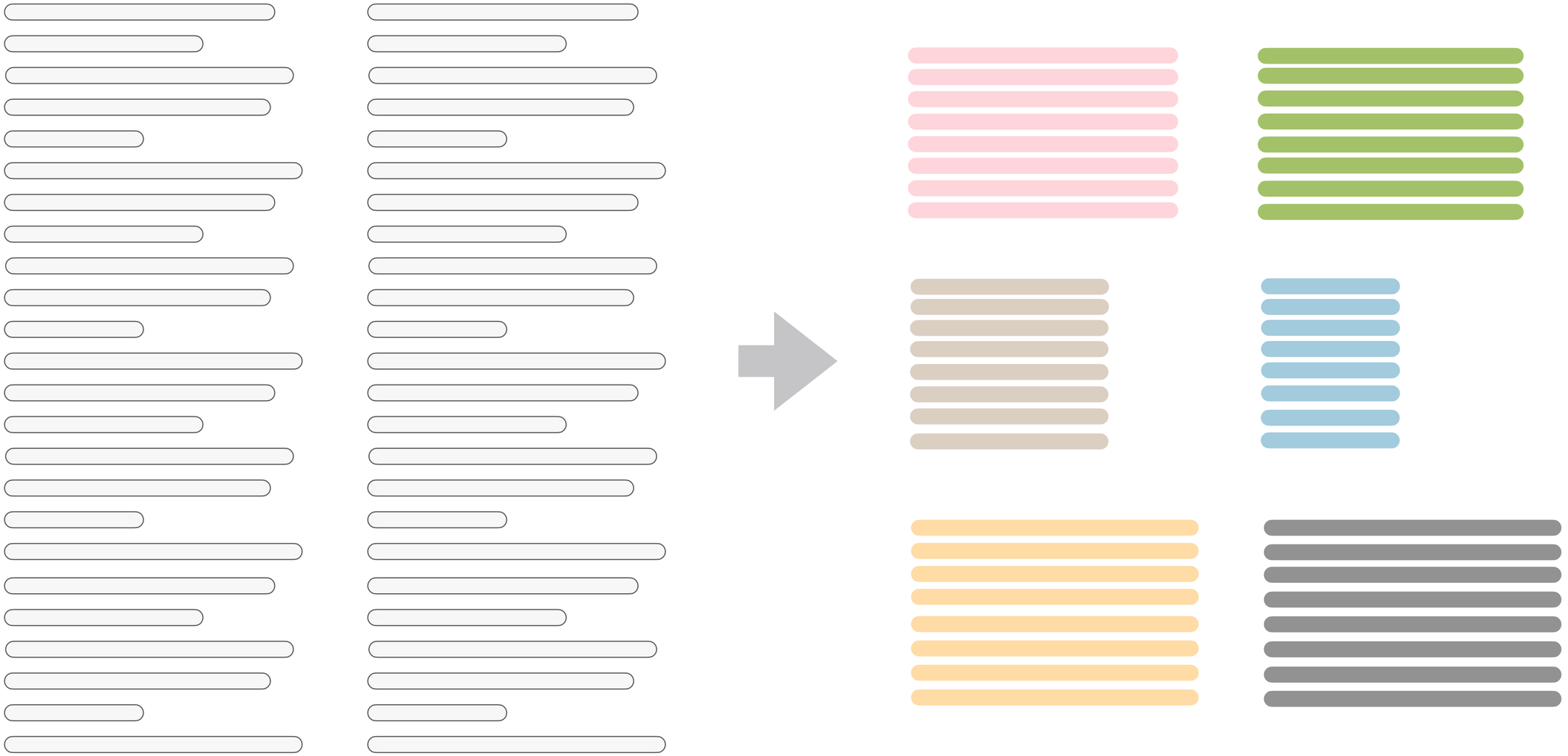
Gather Code Lines across Submissions



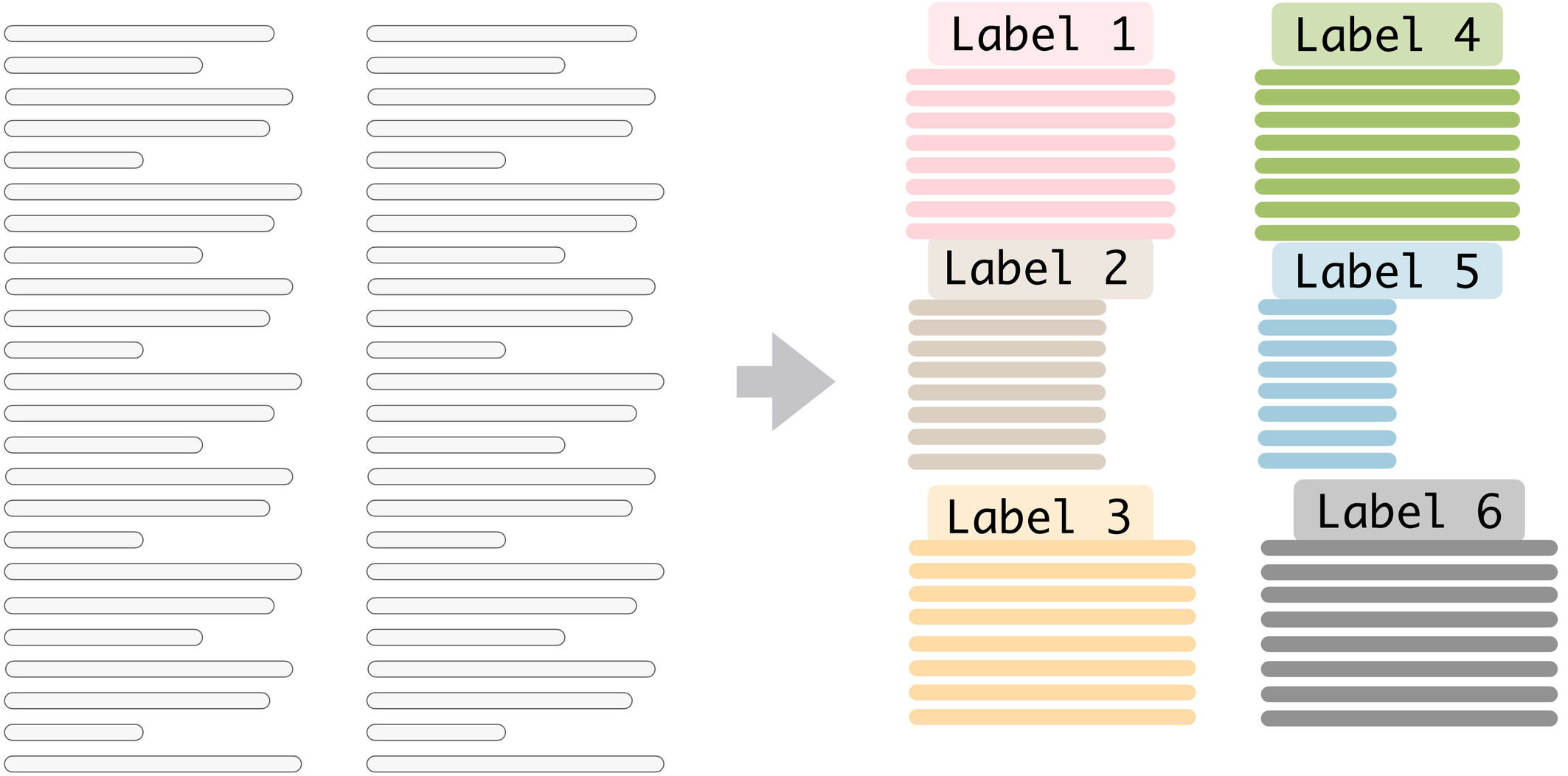
Cluster Code Lines by Semantic Meanings



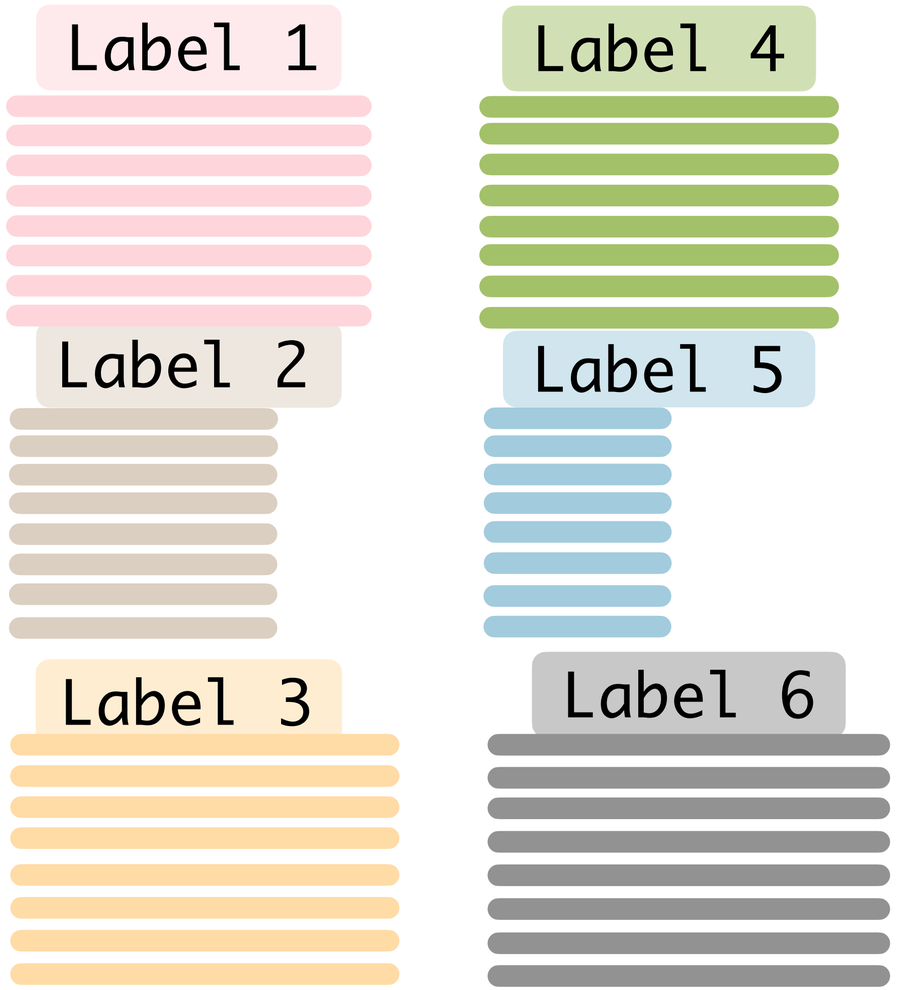
Cluster Code Lines by Semantic Meanings



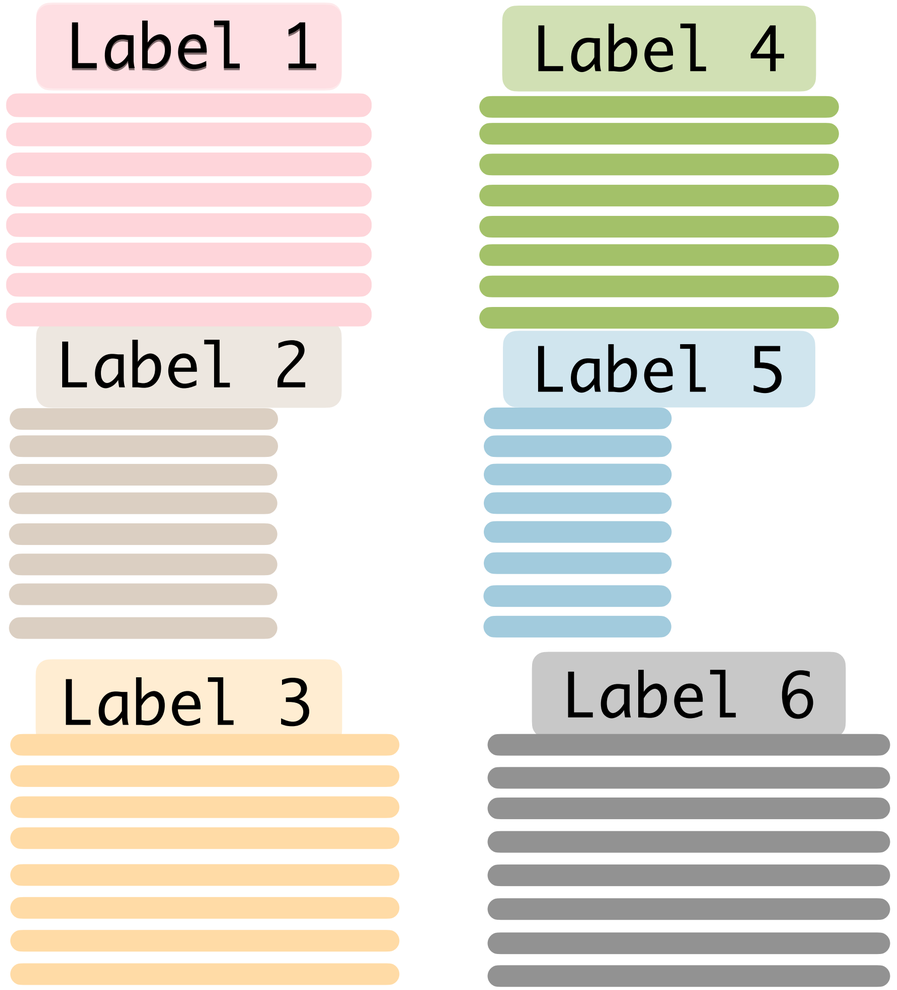
Identify Steps to Solve the Problem



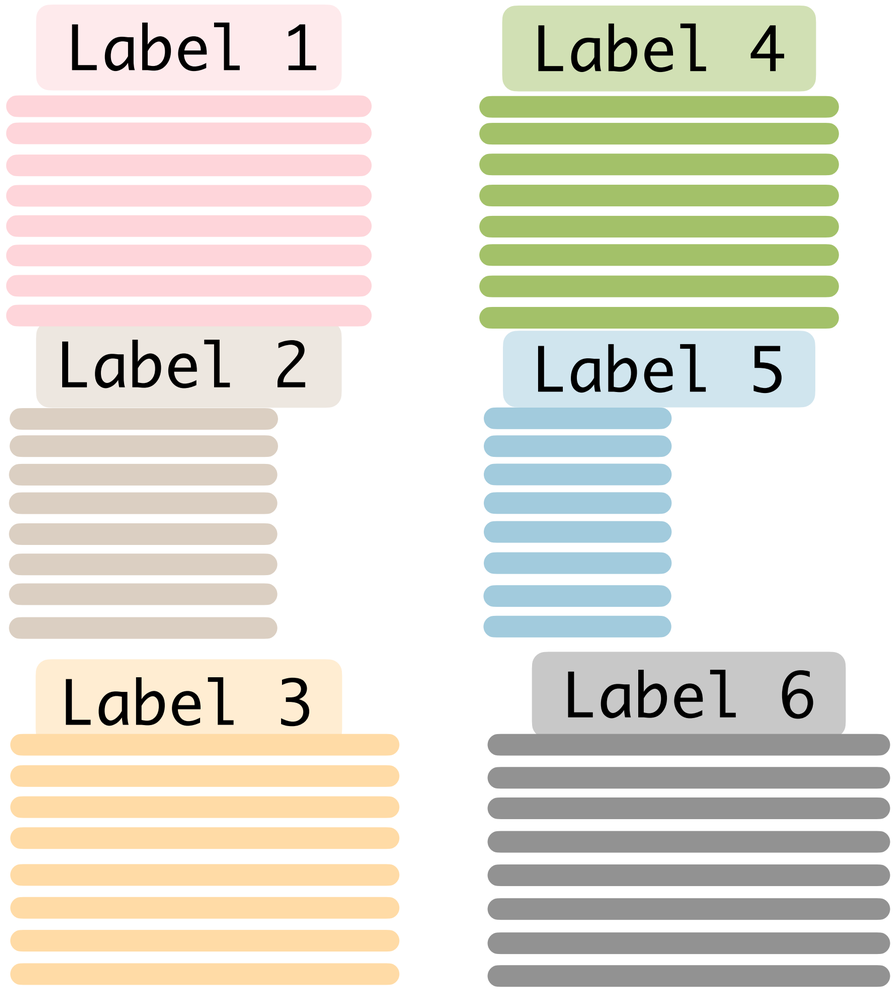
Identify Errors for Each Code Line

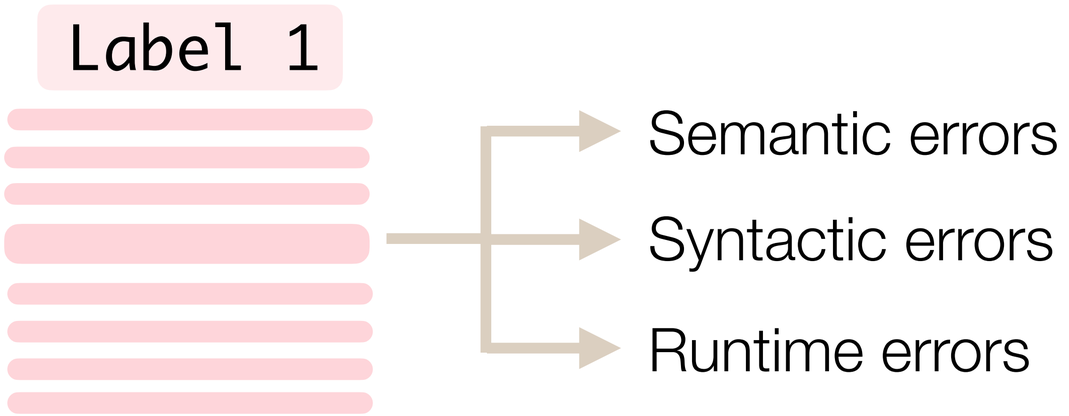


Identify Errors for Each Code Line

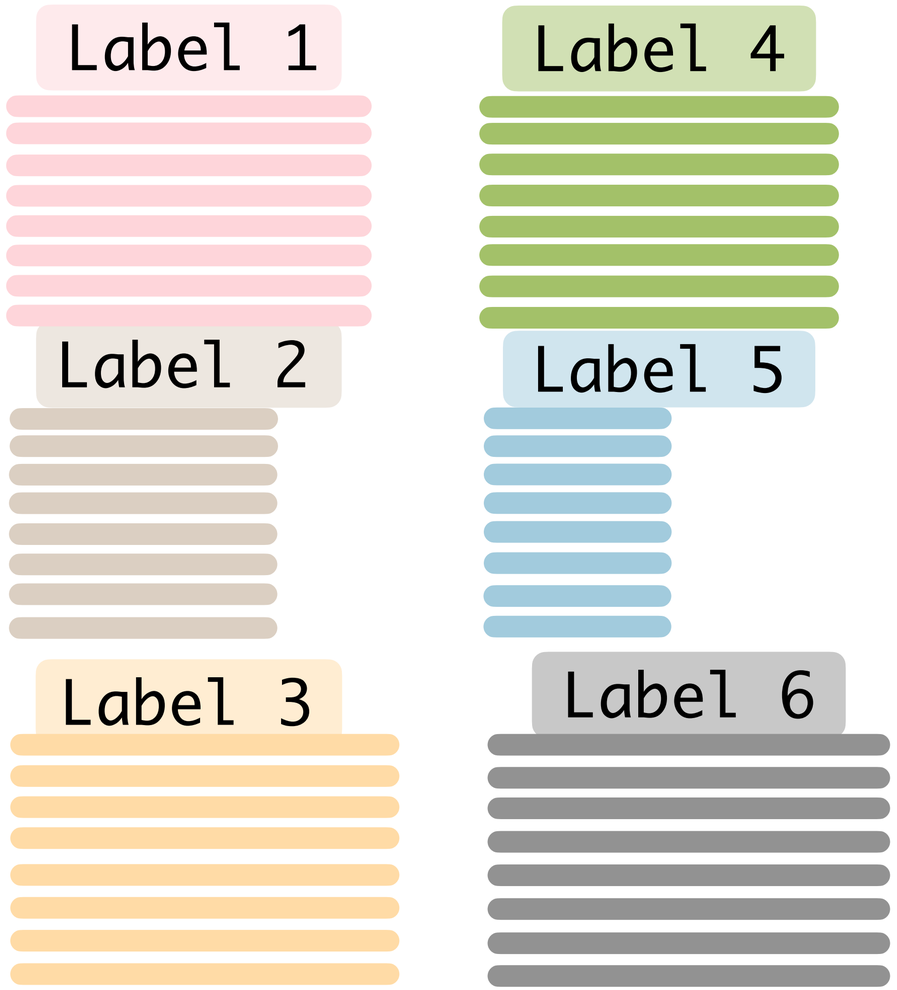


Identify Errors for Each Code Line

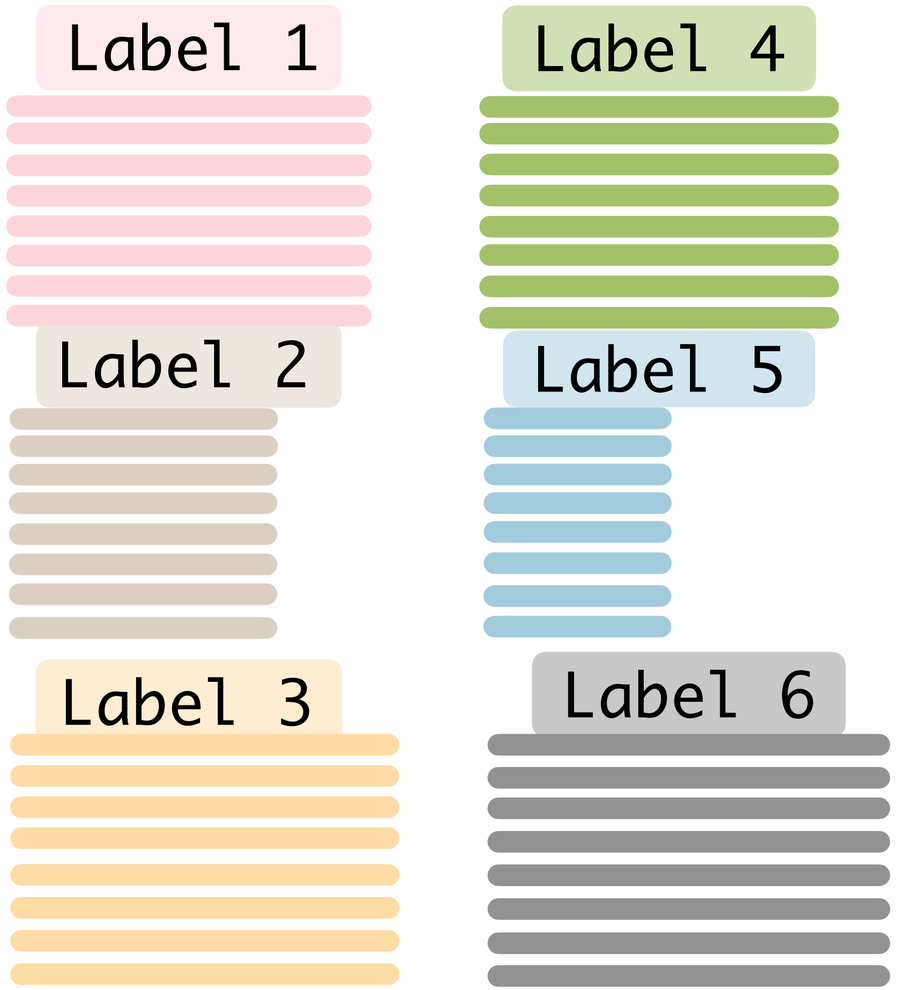




Visualize Submissions in a Semantic Flow

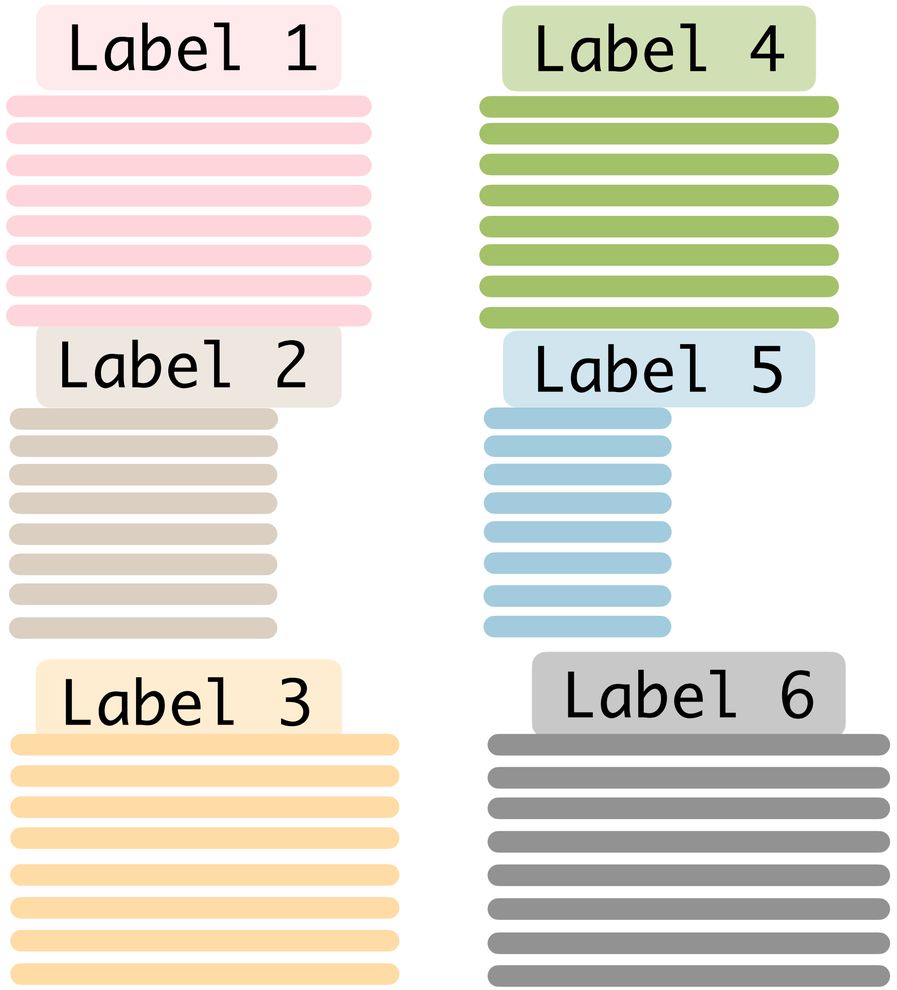


Visualize Submissions in a Semantic Flow



Label 2	Initialize variables
Label 1	Loop over words
Label 3	Check the end of the word
Label 4	Modify the word
Label 5	Append the new word to a new list
Label 6	Other

Visualize Submissions in a Semantic Flow



Label 2	Initialize variables
Label 1	Loop over words
Label 3	Check the end of the word
Label 4	Modify the word
Label 5	Append the new word to a new list
Label 6	Other

Visualize Correctness Distribution in Histograms

Label 2 Initialize variables

Label 1 Loop over words

Label 3 Check the end of the word

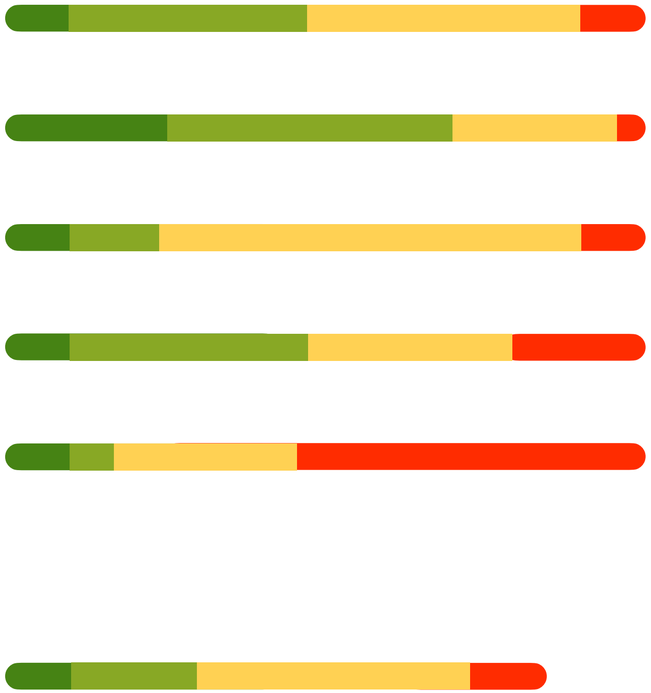
Label 4 Modify the word

Label 5 Append the new word to a new list

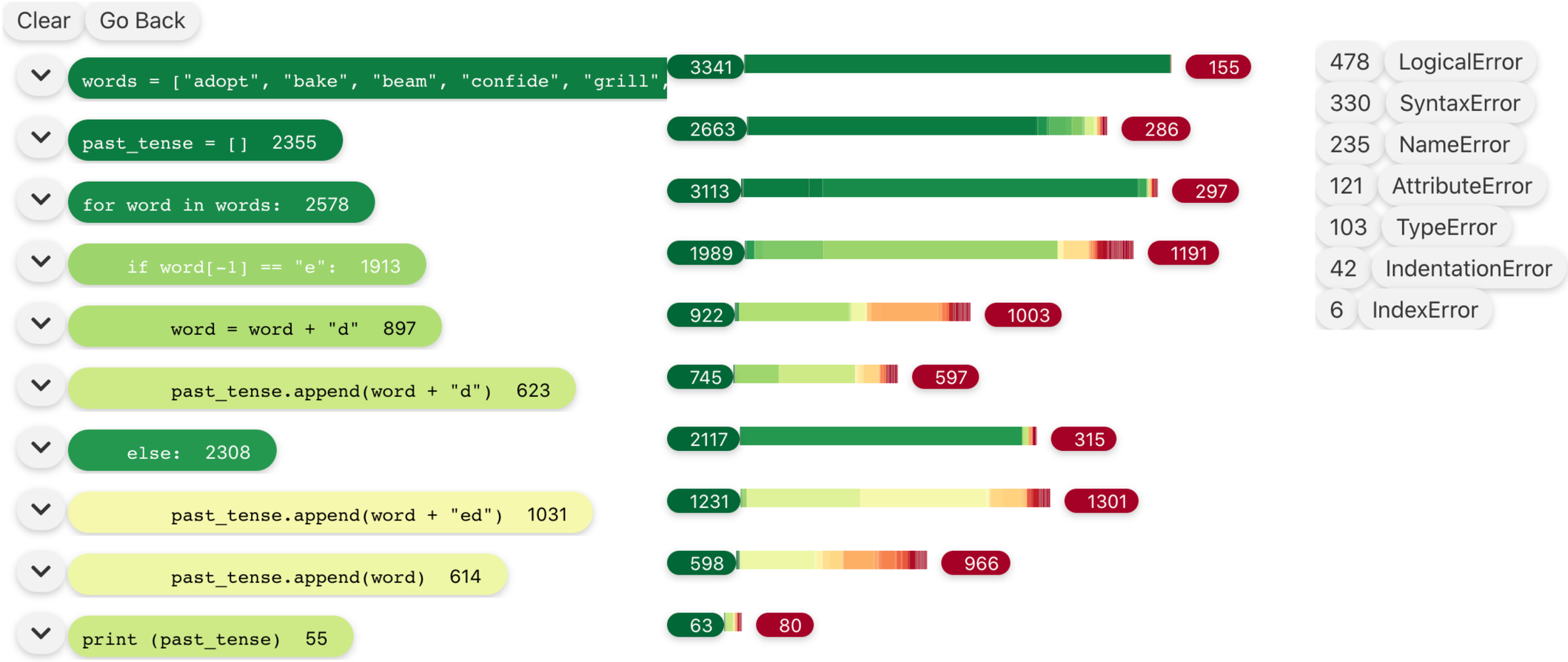
Label 6 Other

Visualize Correctness Distribution in Histograms

Label 2	Initialize variables
Label 1	Loop over words
Label 3	Check the end of the word
Label 4	Modify the word
Label 5	Append the new word to a new list
Label 6	Other



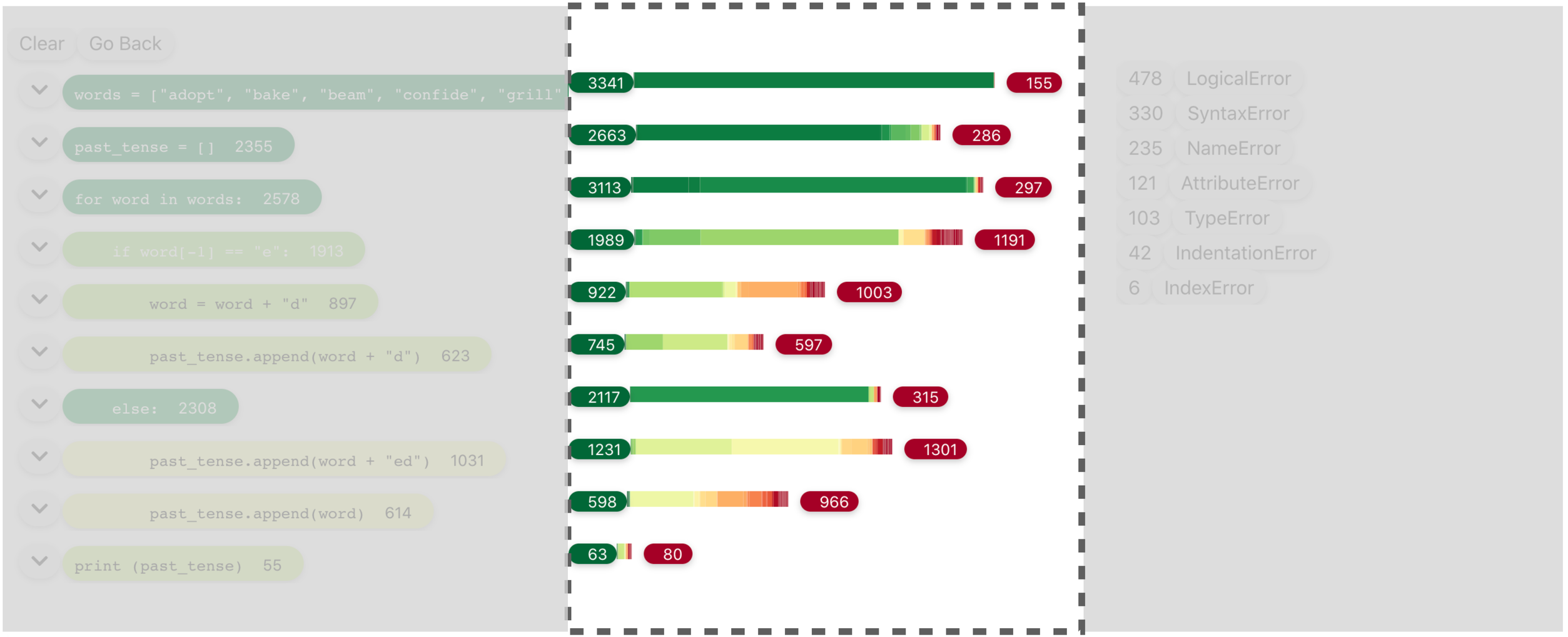
CFlow


Clear
Go Back
words = ["adopt", "bake", "beam", "confide", "grill",
past_tense = [] 2355
for word in words: 2578
if word[-1] == "e": 1913
word = word + "d" 897
past_tense.append(word + "d") 623
else: 2308
past_tense.append(word + "ed") 1031
past_tense.append(word) 614
print (past_tense) 55
3341
155
2663
286
3113
297
1989
1191
922
1003
745
597
2117
315
1231
1301
598
966
63
80
478 LogicalError
330 SyntaxError
235 NameError
121 AttributeError
103 TypeError
42 IndentationError
6 IndexError

CFlow

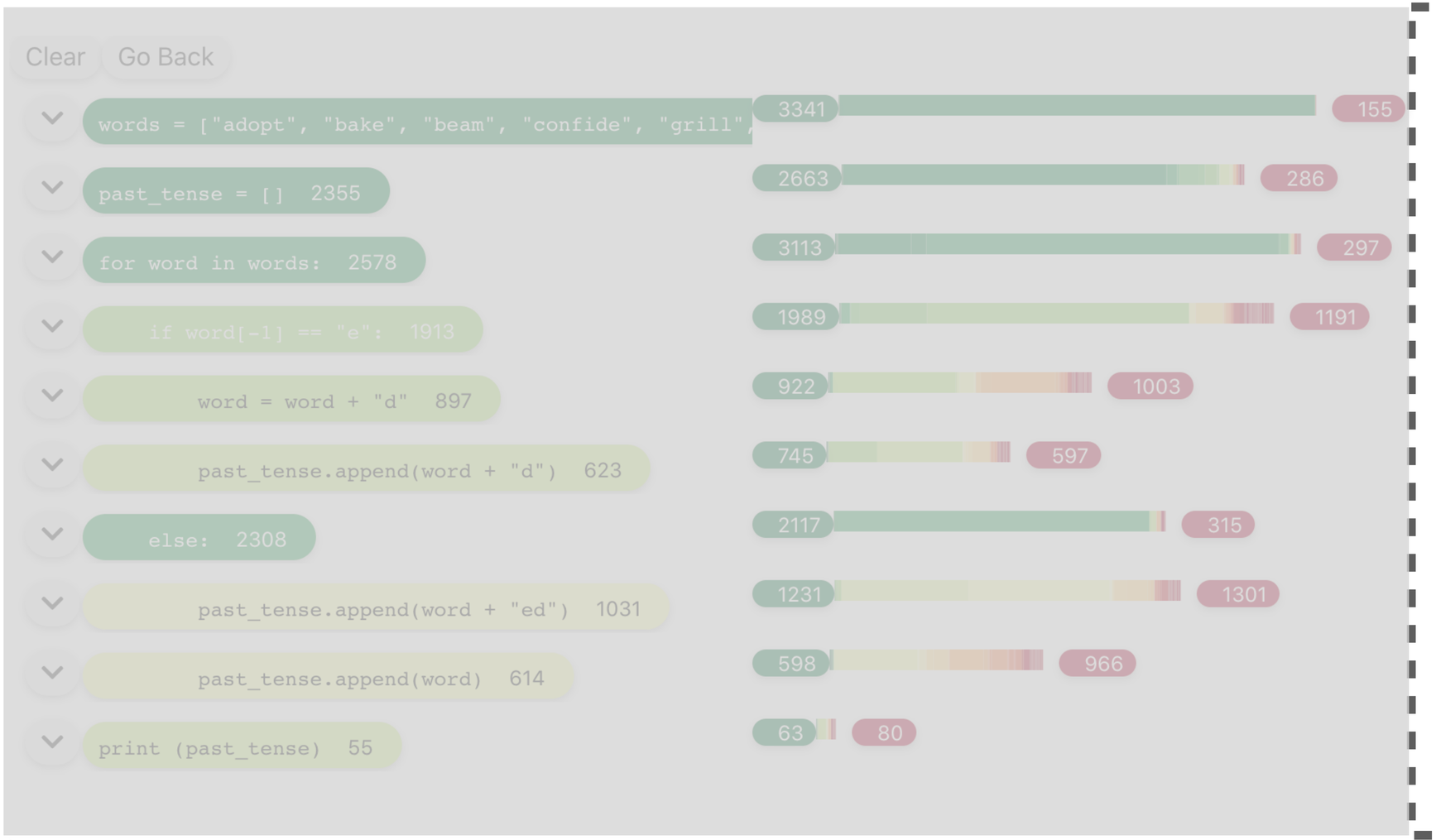
Semantic aggregation view

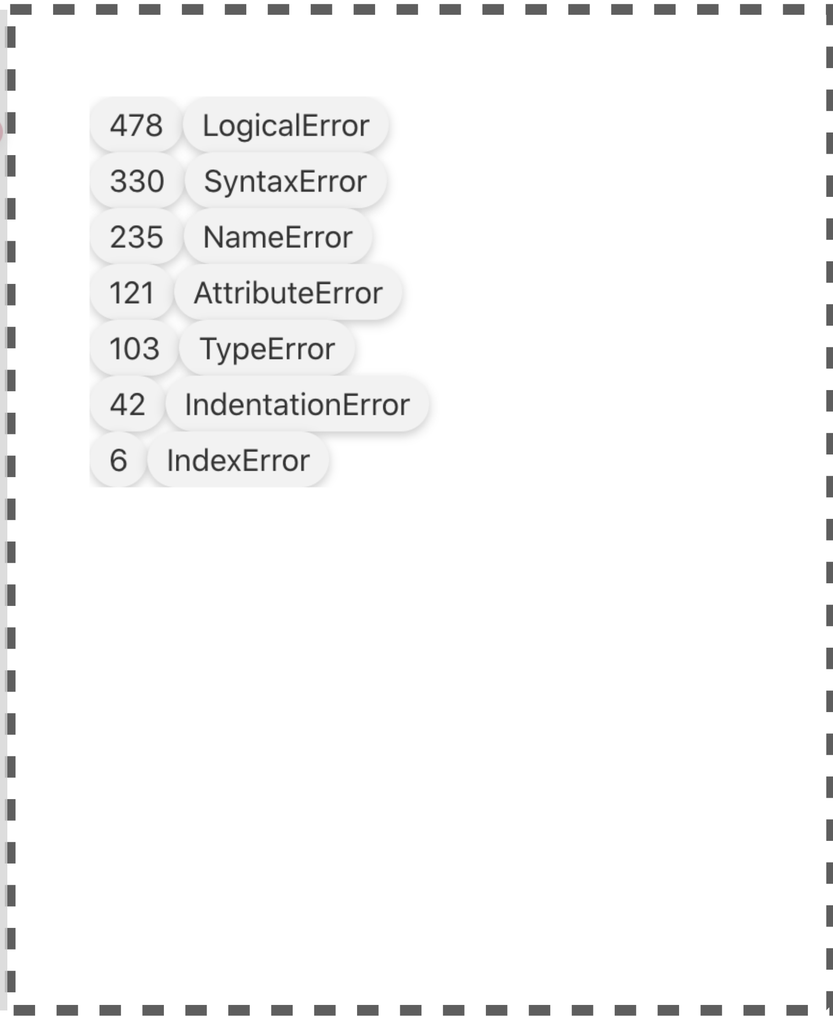
CFlow



Semantic histogram view

CFlow



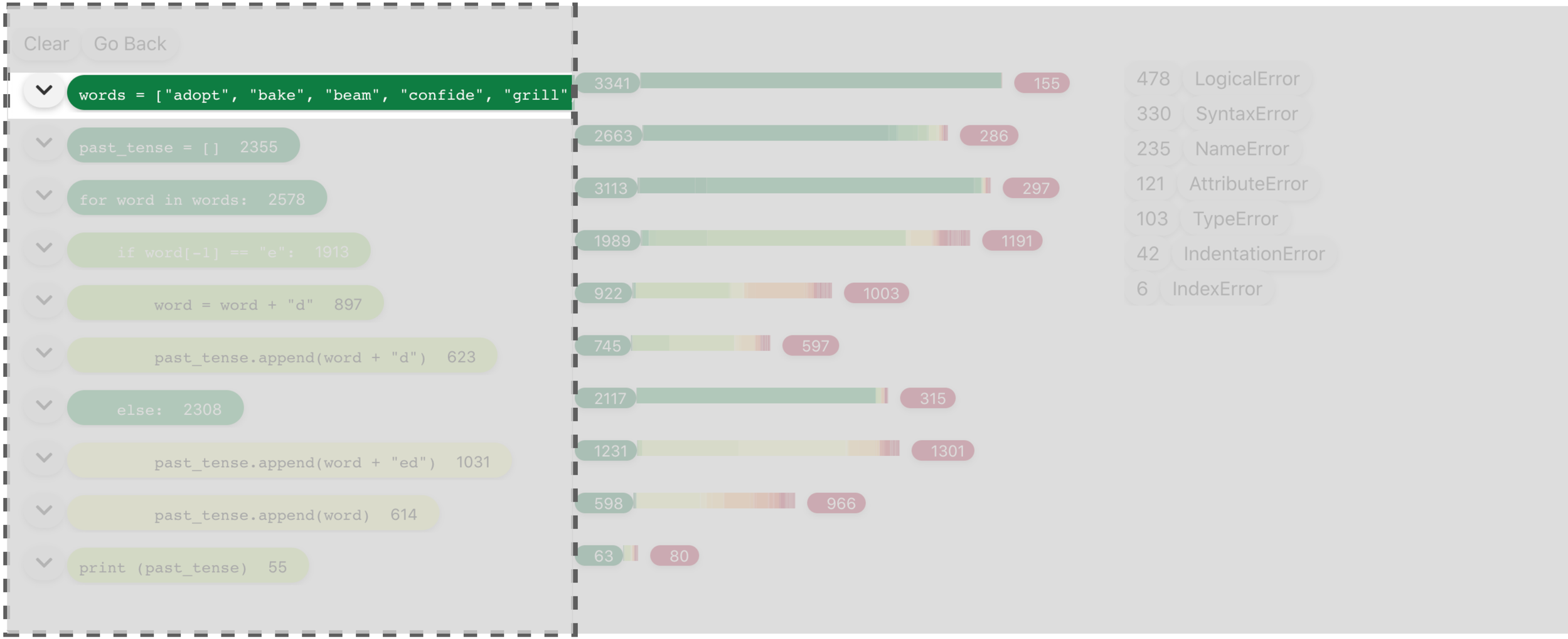


Code detailed view

CFlow

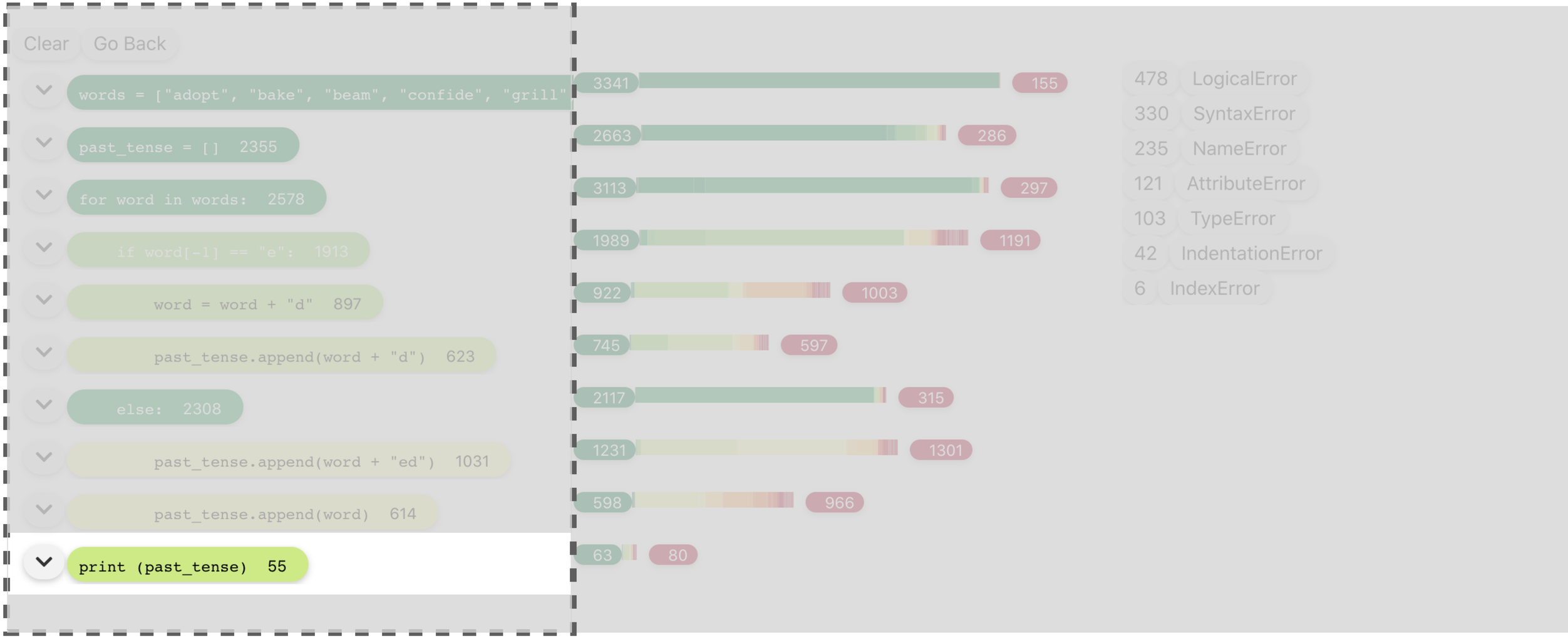
Semantic aggregation view

CFlow



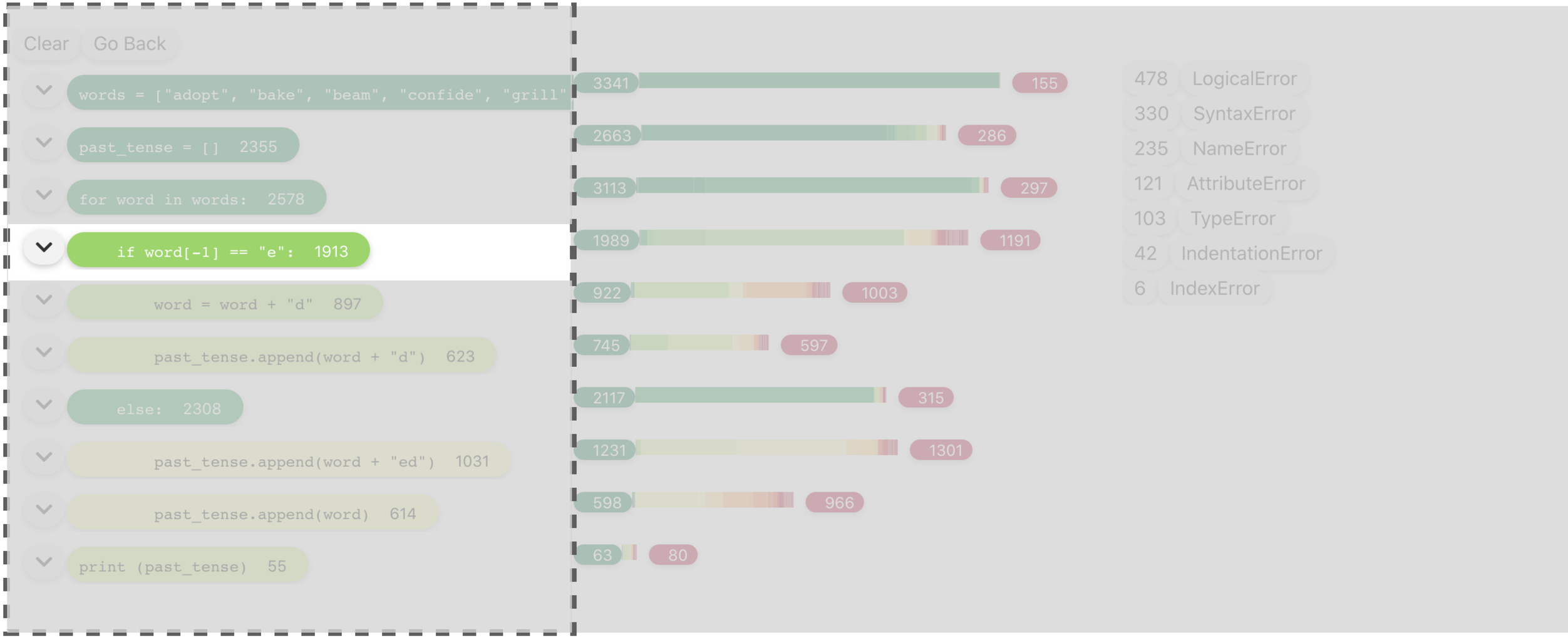
Semantic aggregation view

CFlow



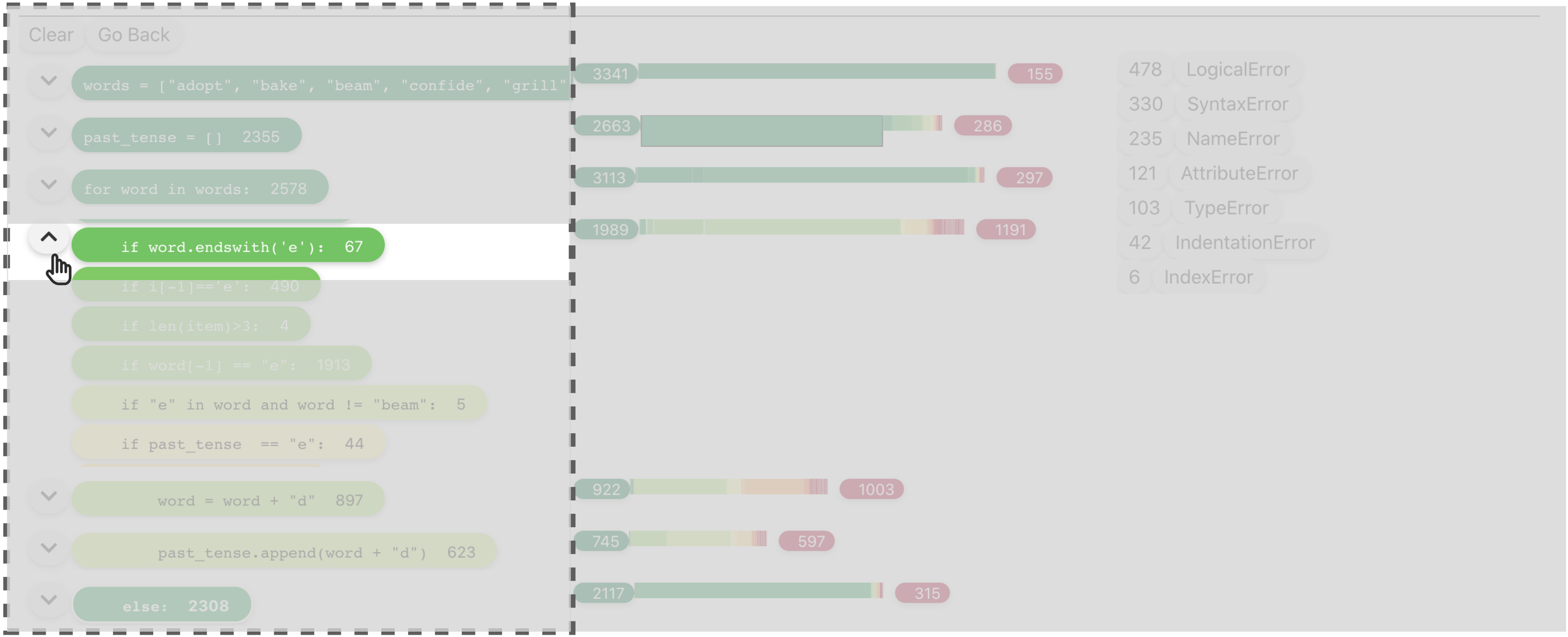
Semantic aggregation view

CFlow



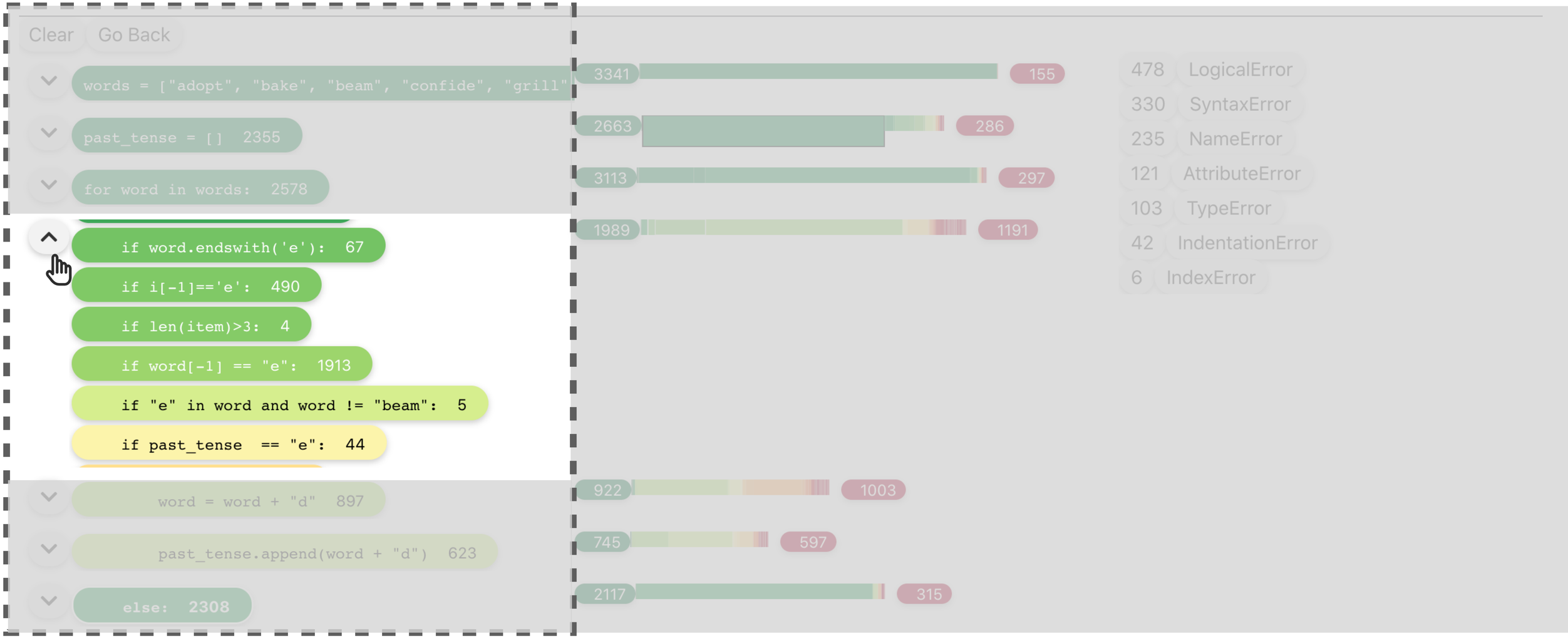
Semantic aggregation view

CFlow



Semantic aggregation view

CFlow



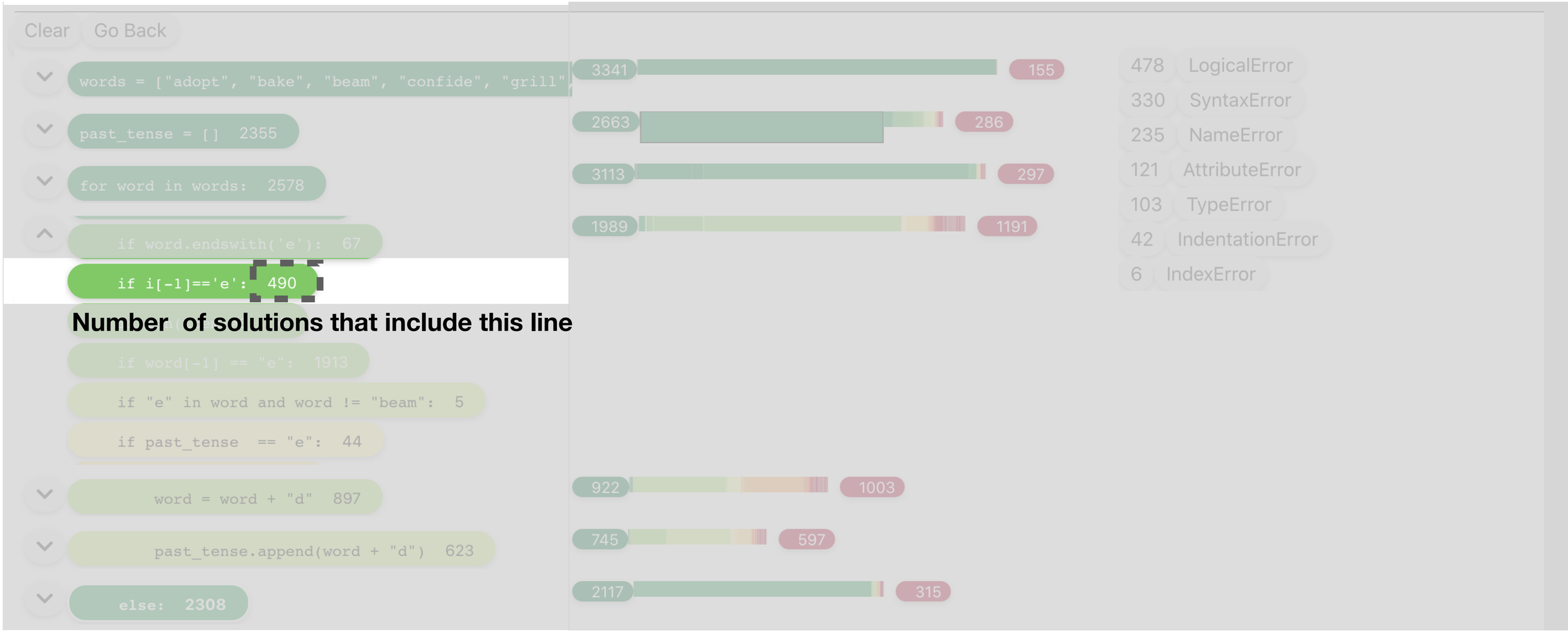
Semantic aggregation view

CFlow

A cluster of code lines

Semantic aggregation view

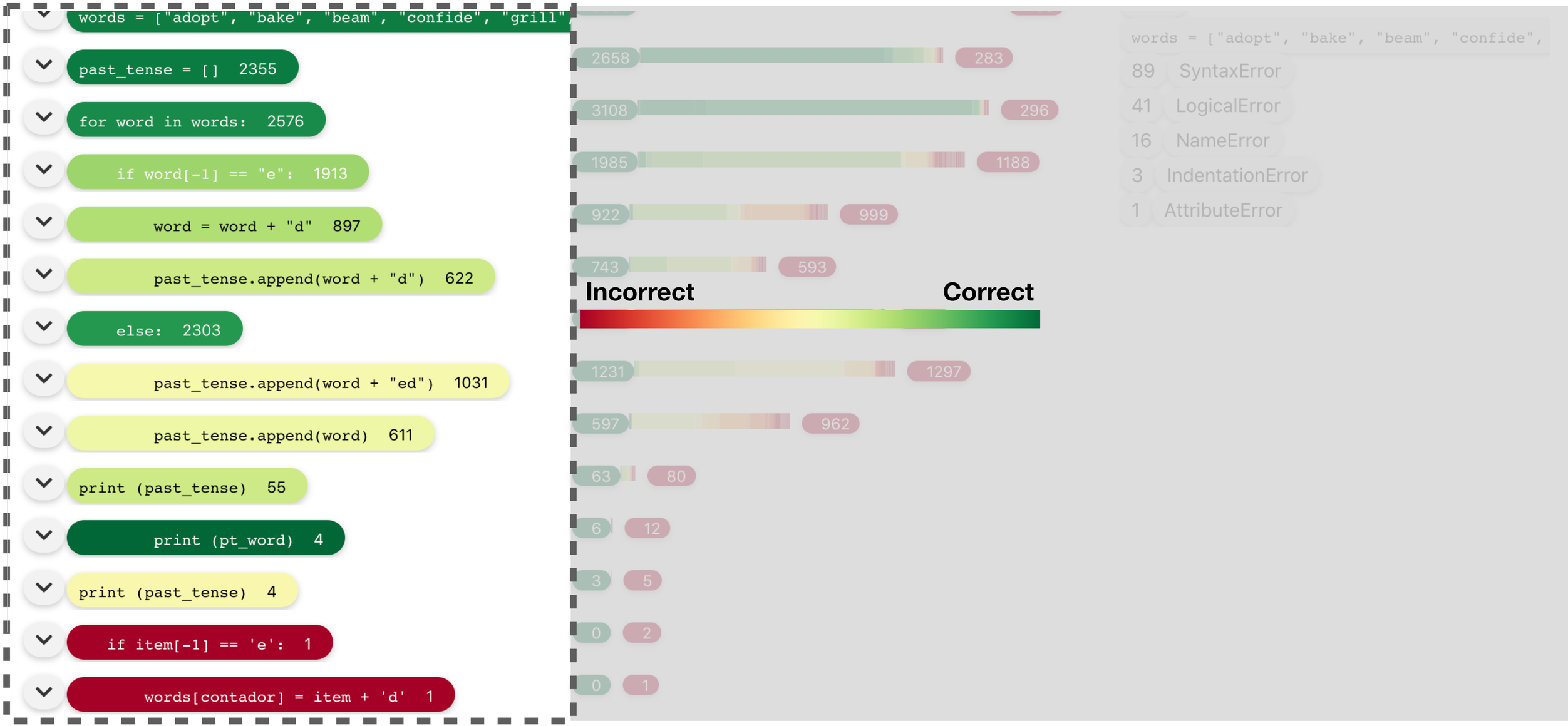
CFlow



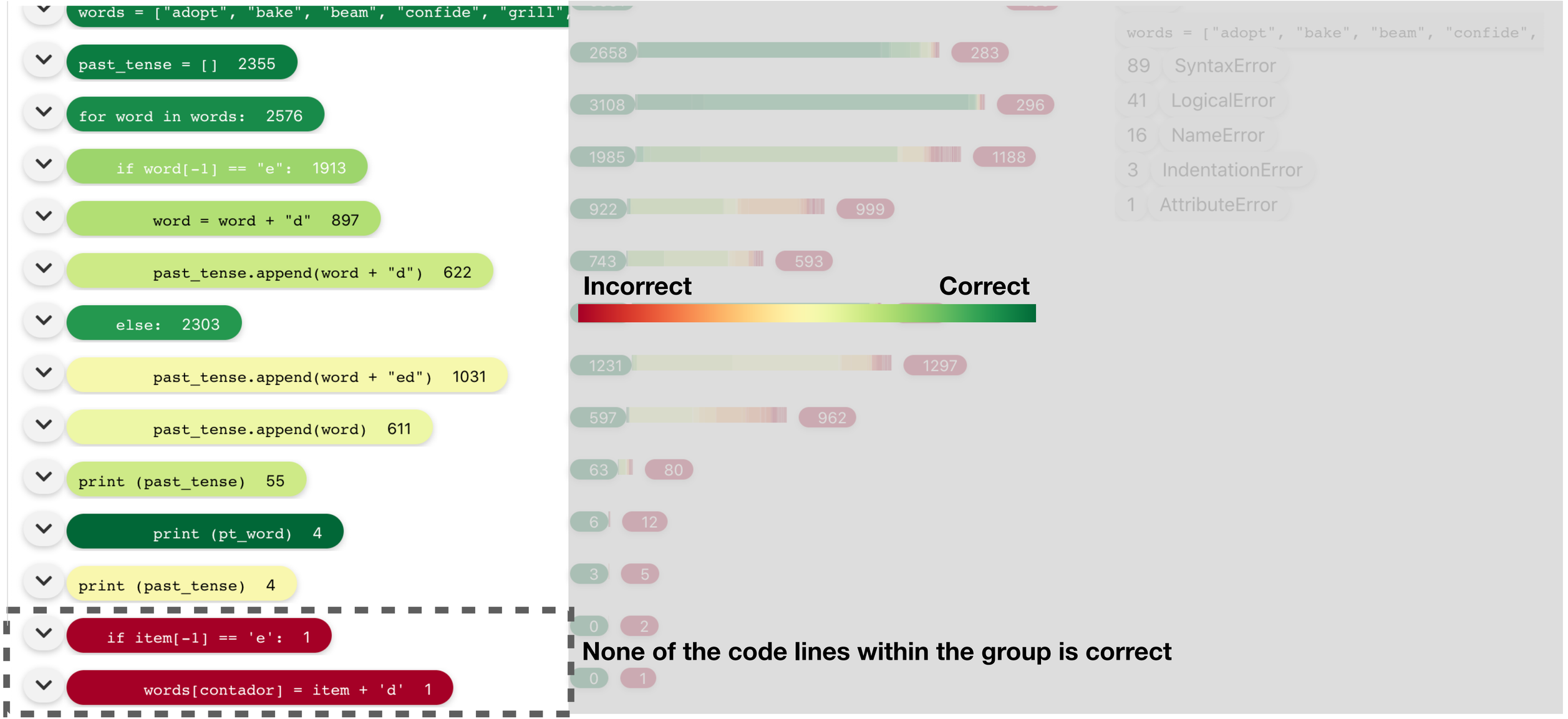
Number of solutions that include this line

Semantic aggregation view

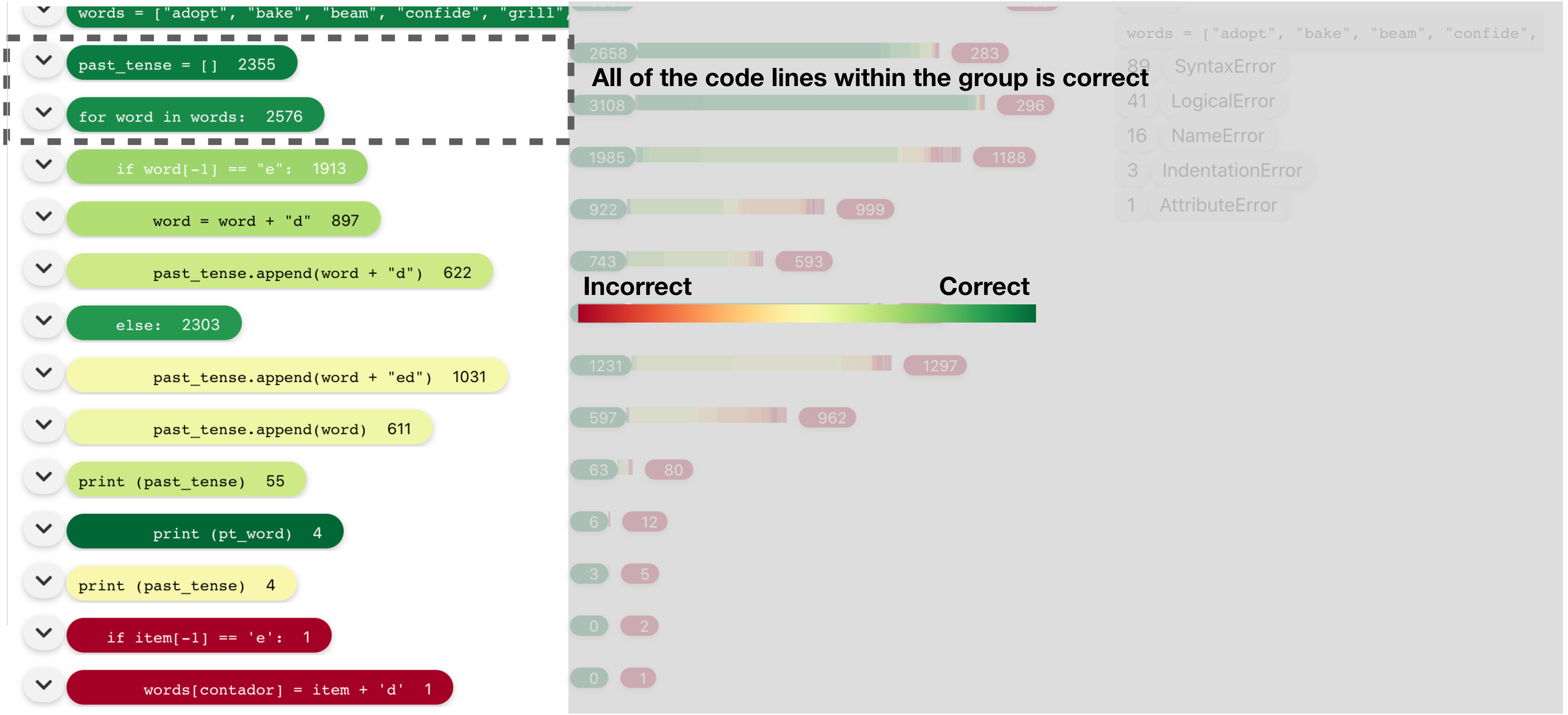
CFlow


words = ["adopt", "bake", "beam", "confide", "grill",
past_tense = [] 2355
for word in words: 2576
if word[-1] == "e": 1913
word = word + "d" 897
past_tense.append(word + "d") 622
else: 2303
past_tense.append(word + "ed") 1031
past_tense.append(word) 611
print (past_tense) 55
print (pt_word) 4
print (past_tense) 4
if item[-1] == 'e': 1
words[contador] = item + 'd' 1
2658
283
3108
296
1985
1188
922
999
743
593
Incorrect
Correct
1231
1297
597
962
63
80
6
12
3
5
0
2
0
1
words = ["adopt", "bake", "beam", "confide",
89 SyntaxError
41 LogicalError
16 NameError
3 IndentationError
1 AttributeError

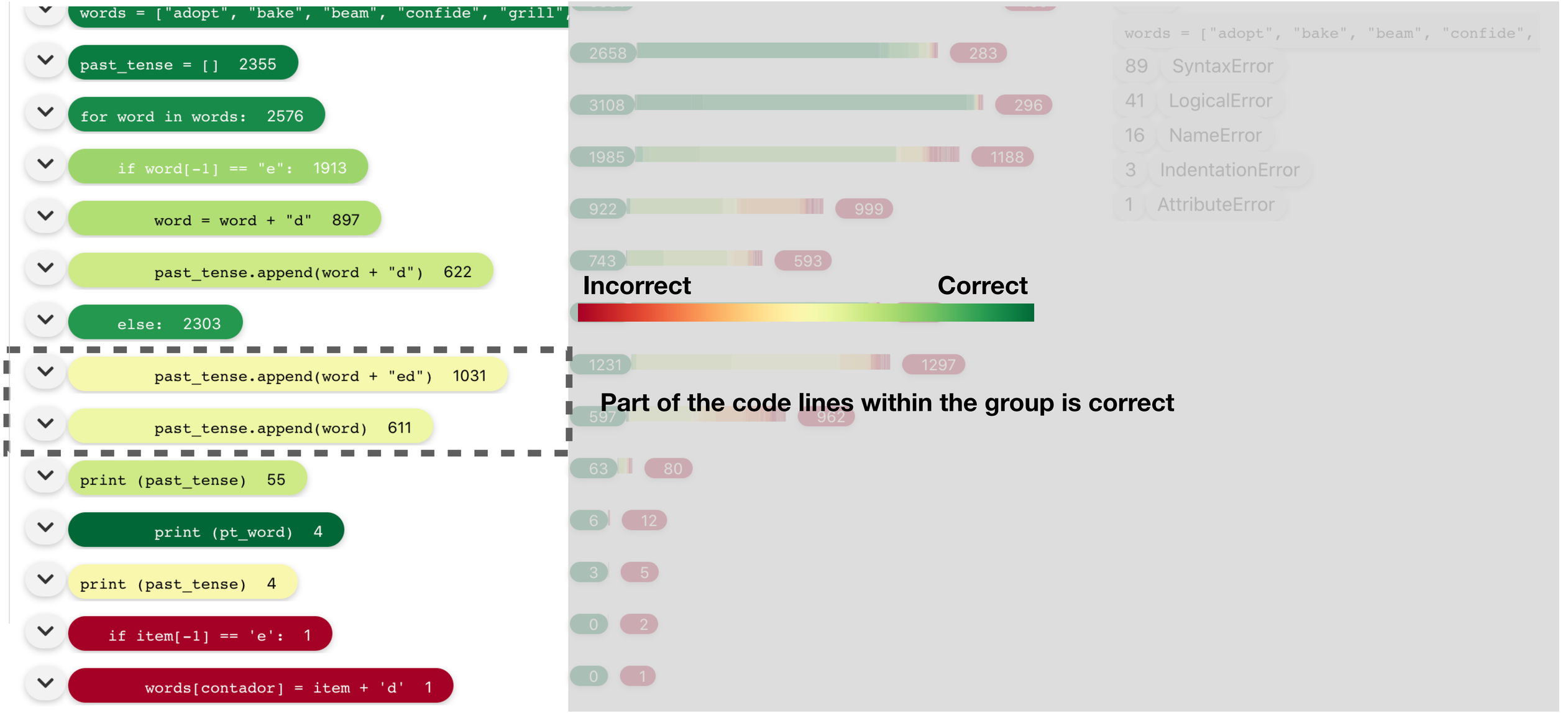
CFlow


words = ["adopt", "bake", "beam", "confide", "grill",
past_tense = [] 2355
for word in words: 2576
if word[-1] == "e": 1913
word = word + "d" 897
past_tense.append(word + "d") 622
else: 2303
past_tense.append(word + "ed") 1031
past_tense.append(word) 611
print (past_tense) 55
print (pt_word) 4
print (past_tense) 4
if item[-1] == 'e': 1
words[contador] = item + 'd' 1
2658 283
3108 296
1985 1188
922 999
743 593
1231 1297
597 962
63 80
6 12
3 5
0 2
0 1
Incorrect
Correct
None of the code lines within the group is correct
words = ["adopt", "bake", "beam", "confide",
89 SyntaxError
41 LogicalError
16 NameError
3 IndentationError
1 AttributeError

CFlow


words = ["adopt", "bake", "beam", "confide", "grill",
past_tense = [] 2355
for word in words: 2576
if word[-1] == "e": 1913
word = word + "d" 897
past_tense.append(word + "d") 622
else: 2303
past_tense.append(word + "ed") 1031
past_tense.append(word) 611
print (past_tense) 55
print (pt_word) 4
print (past_tense) 4
if item[-1] == 'e': 1
words[contador] = item + 'd' 1
2658
283
3108
296
1985
1188
922
999
743
593
1231
1297
597
962
63
80
6
12
3
5
0
2
0
1
All of the code lines within the group is correct
Incorrect
Correct
words = ["adopt", "bake", "beam", "confide",
89 SyntaxError
41 LogicalError
16 NameError
3 IndentationError
1 AttributeError

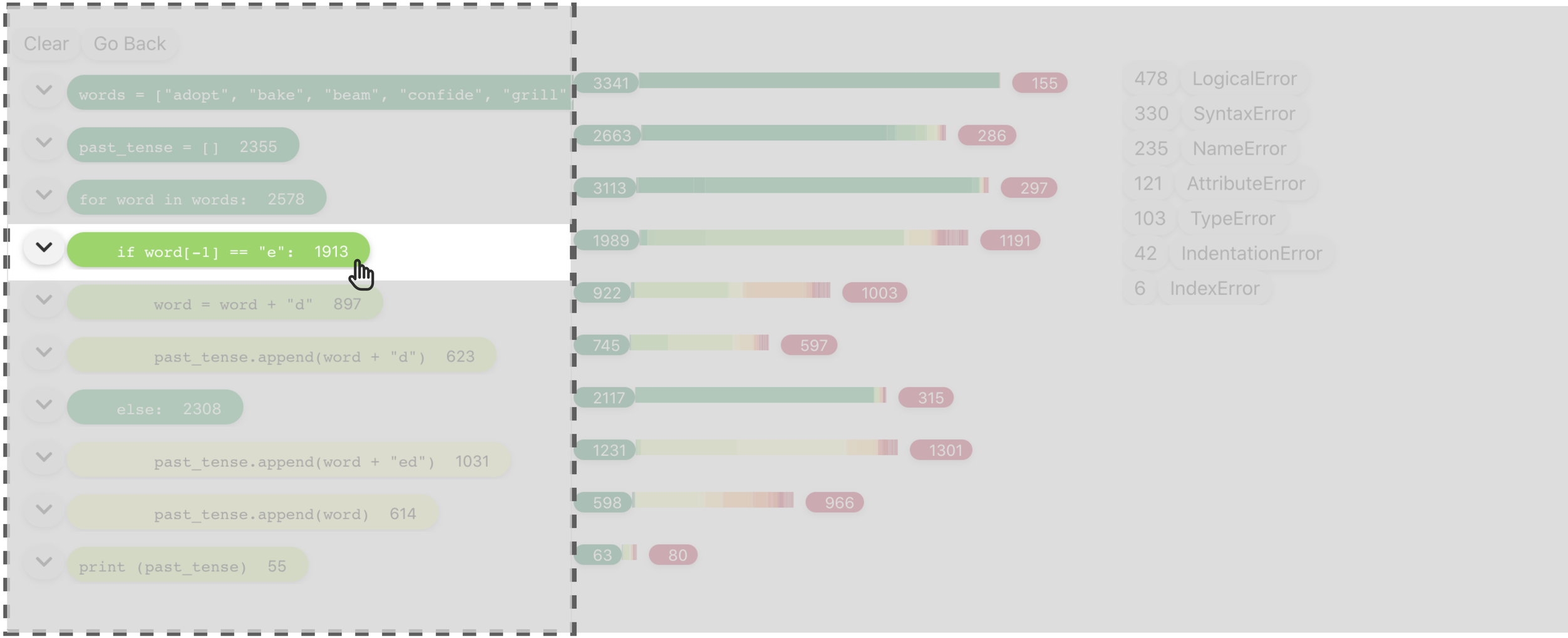
CFlow


words = ["adopt", "bake", "beam", "confide", "grill",
past_tense = [] 2355
for word in words: 2576
if word[-1] == "e": 1913
word = word + "d" 897
past_tense.append(word + "d") 622
else: 2303
past_tense.append(word + "ed") 1031
past_tense.append(word) 611
print (past_tense) 55
print (pt_word) 4
print (past_tense) 4
if item[-1] == 'e': 1
words[contador] = item + 'd' 1
2658
283
3108
296
1985
1188
922
999
743
593
1231
1297
597
962
63
80
6
12
3
5
0
2
0
1
words = ["adopt", "bake", "beam", "confide",
89 SyntaxError
41 LogicalError
16 NameError
3 IndentationError
1 AttributeError
Incorrect
Correct
Part of the code lines within the group is correct

CFlow

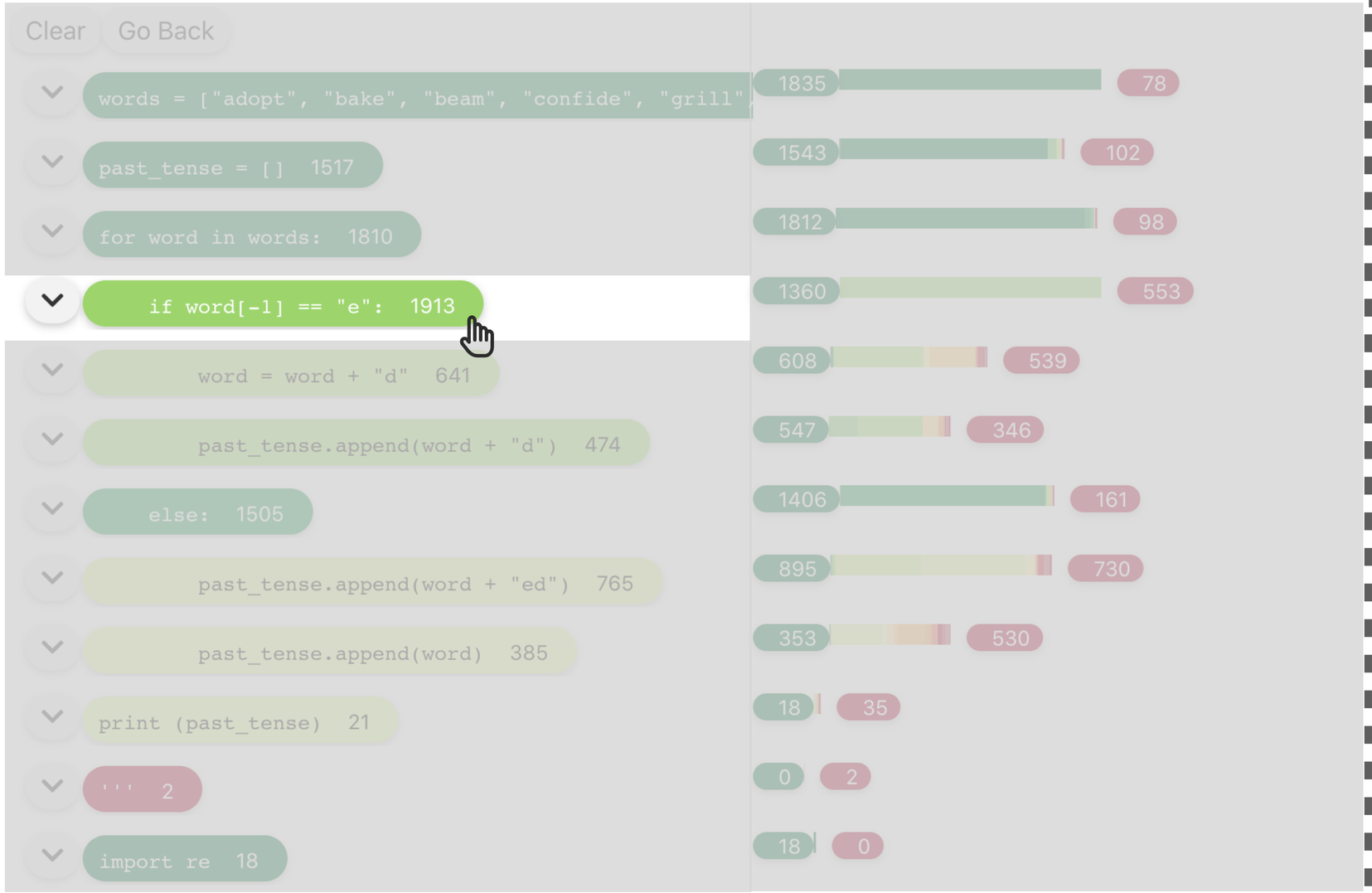
Semantic aggregation view

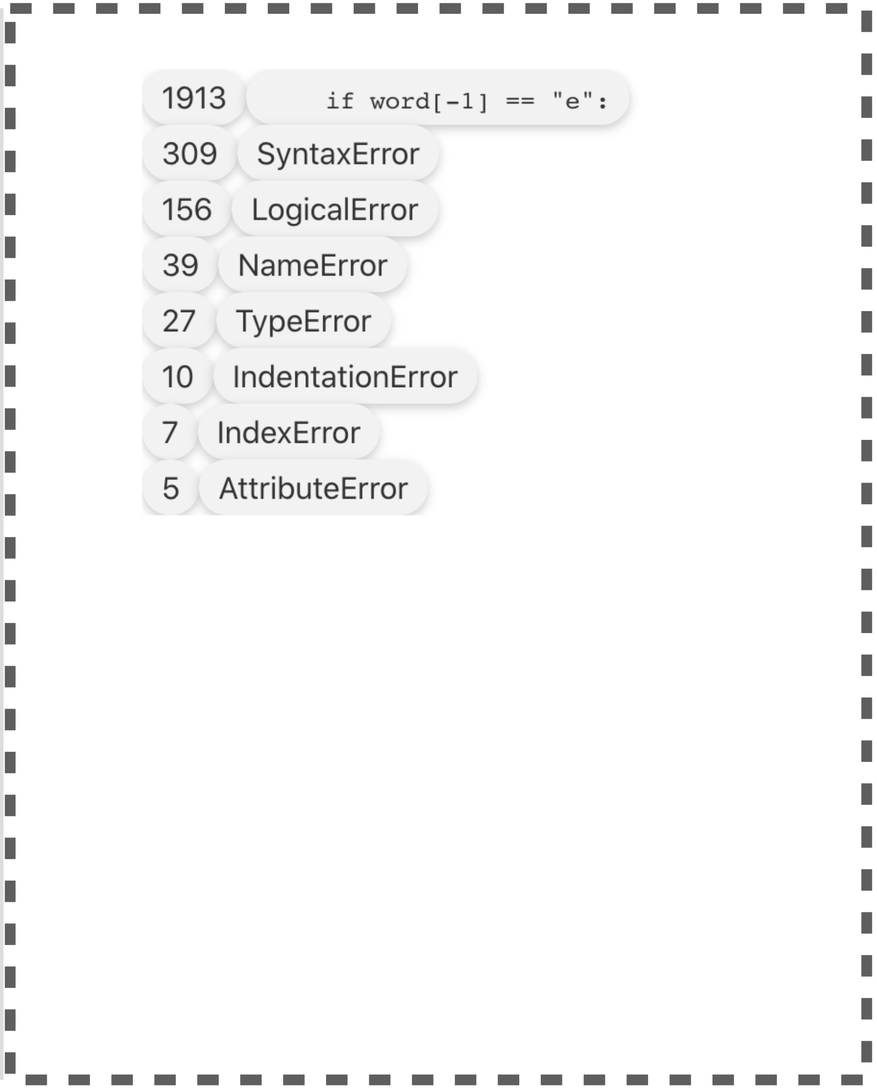
CFlow



Semantic aggregation view

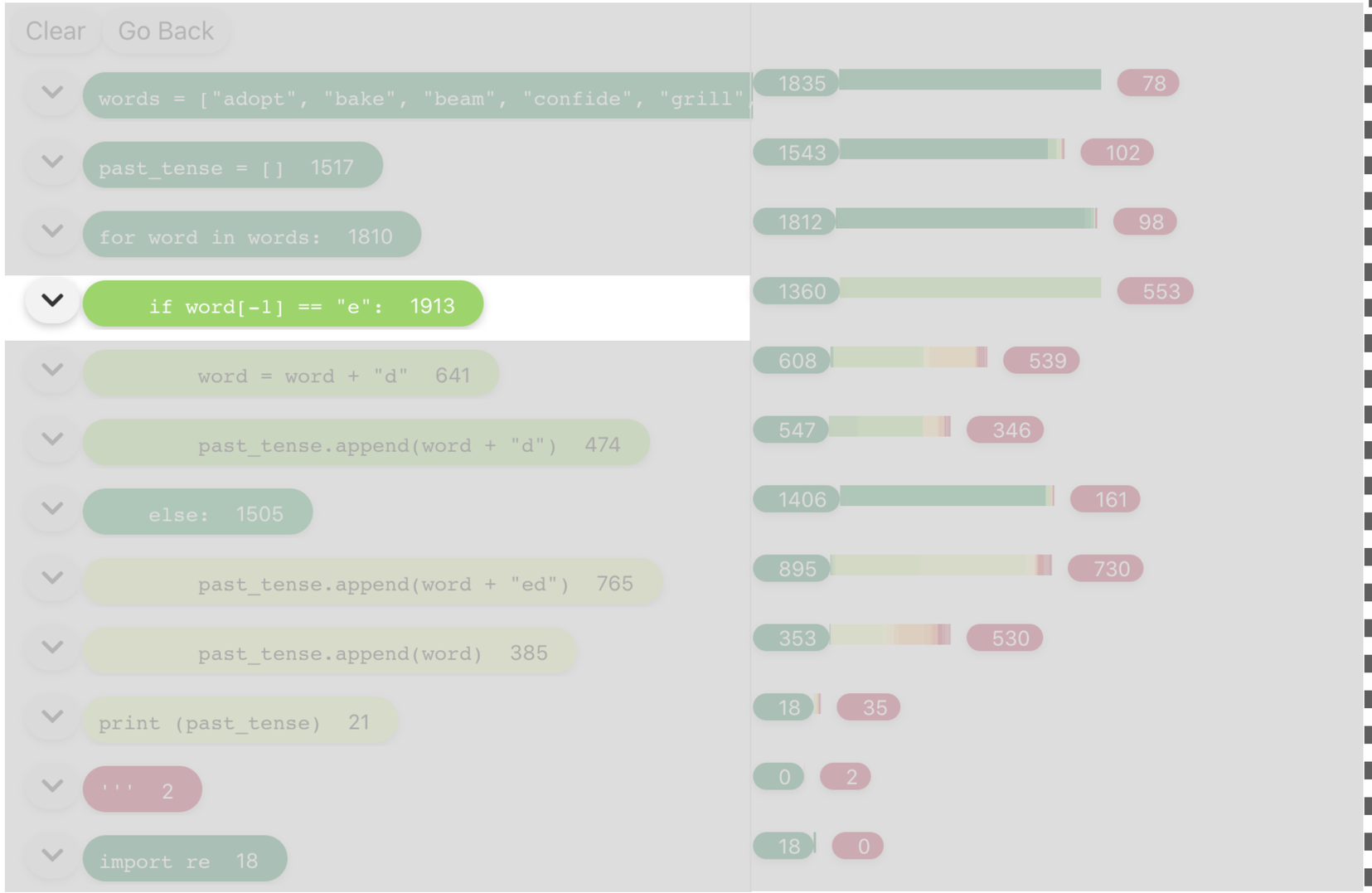
CFlow

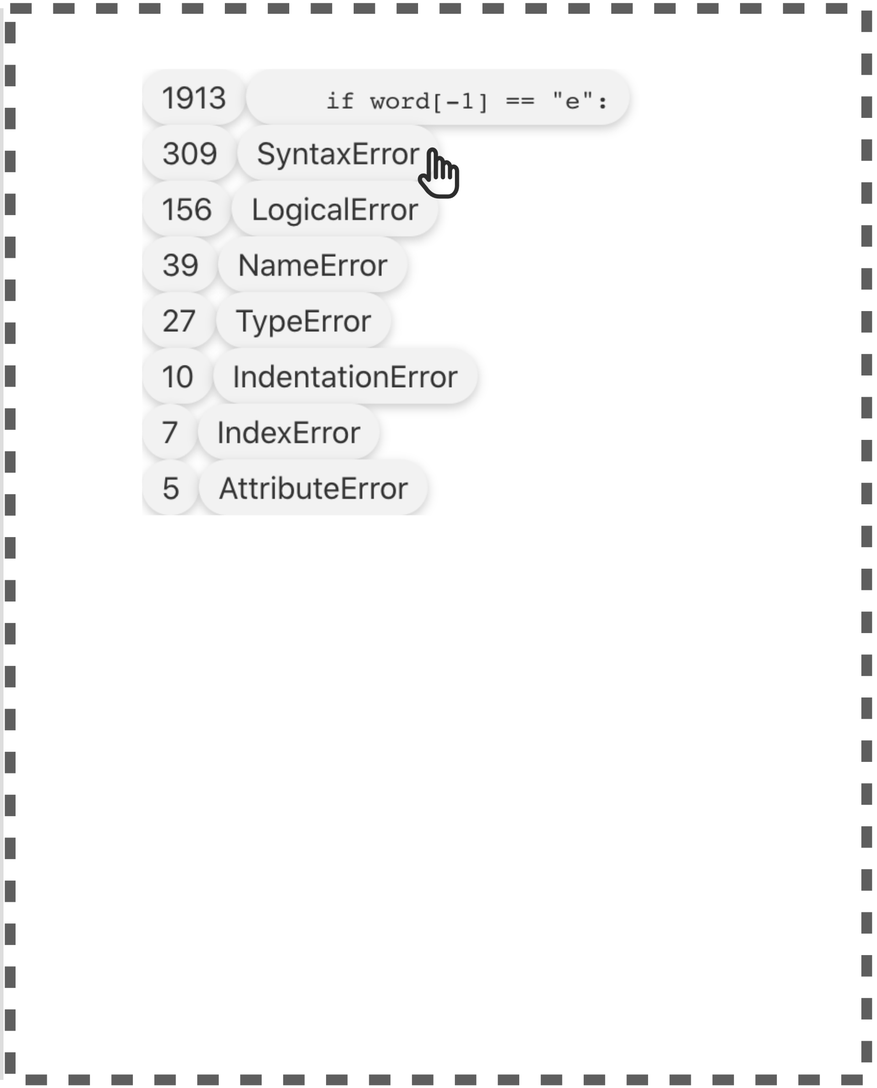




Code detailed view

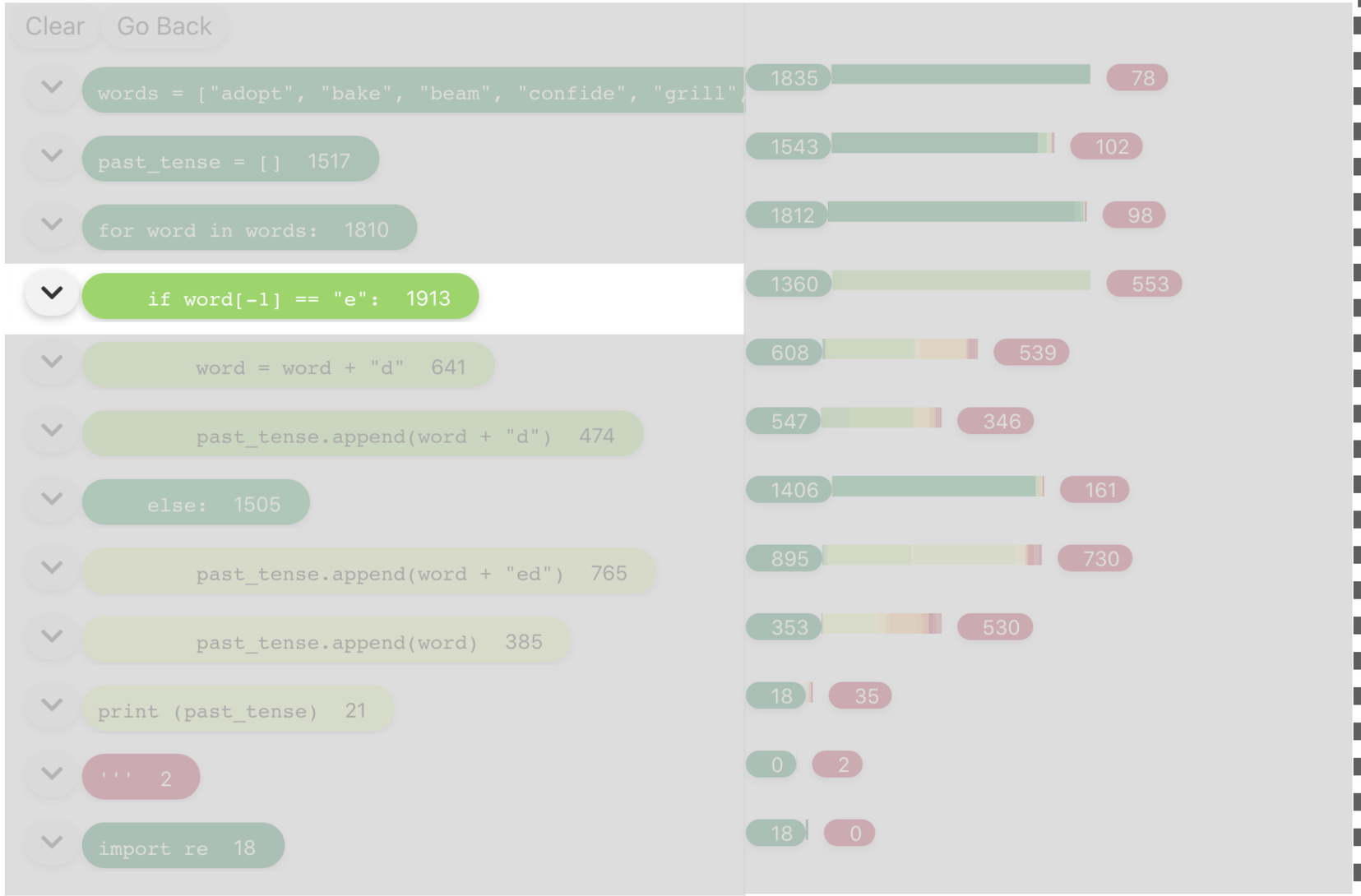
CFlow

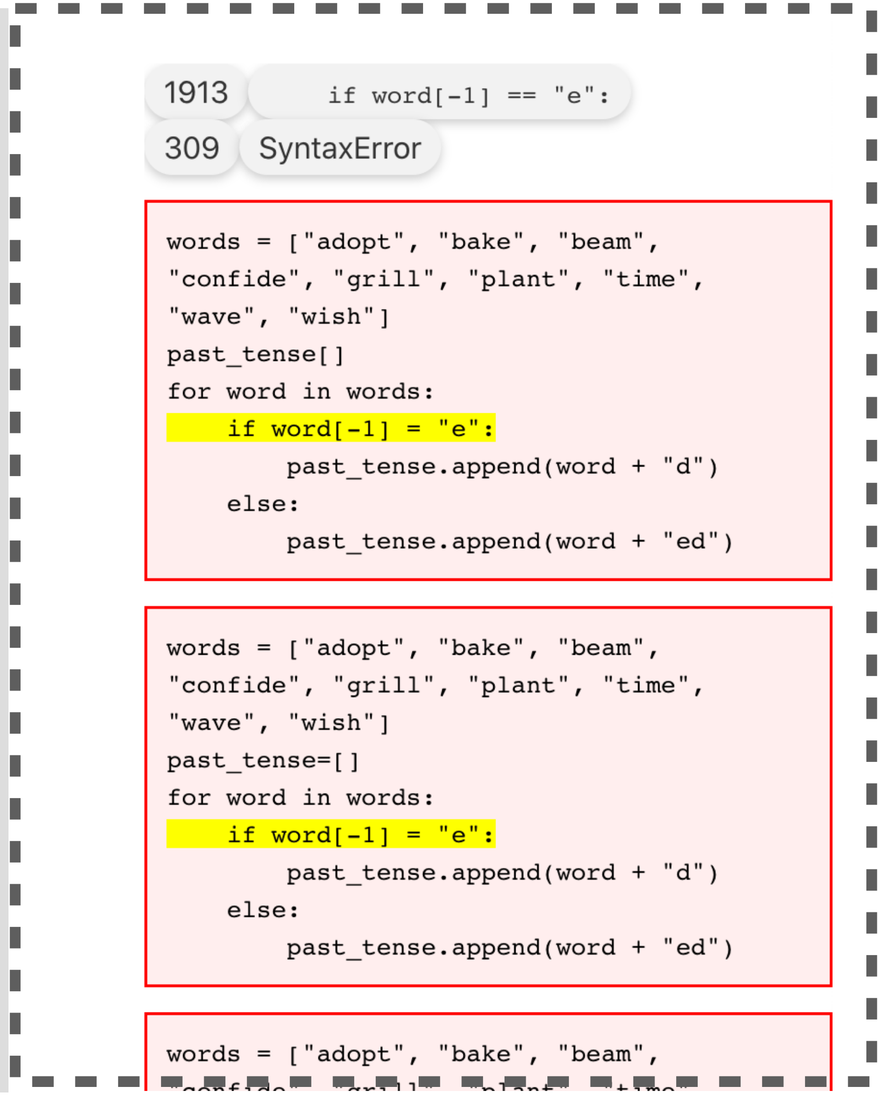




Code detailed view

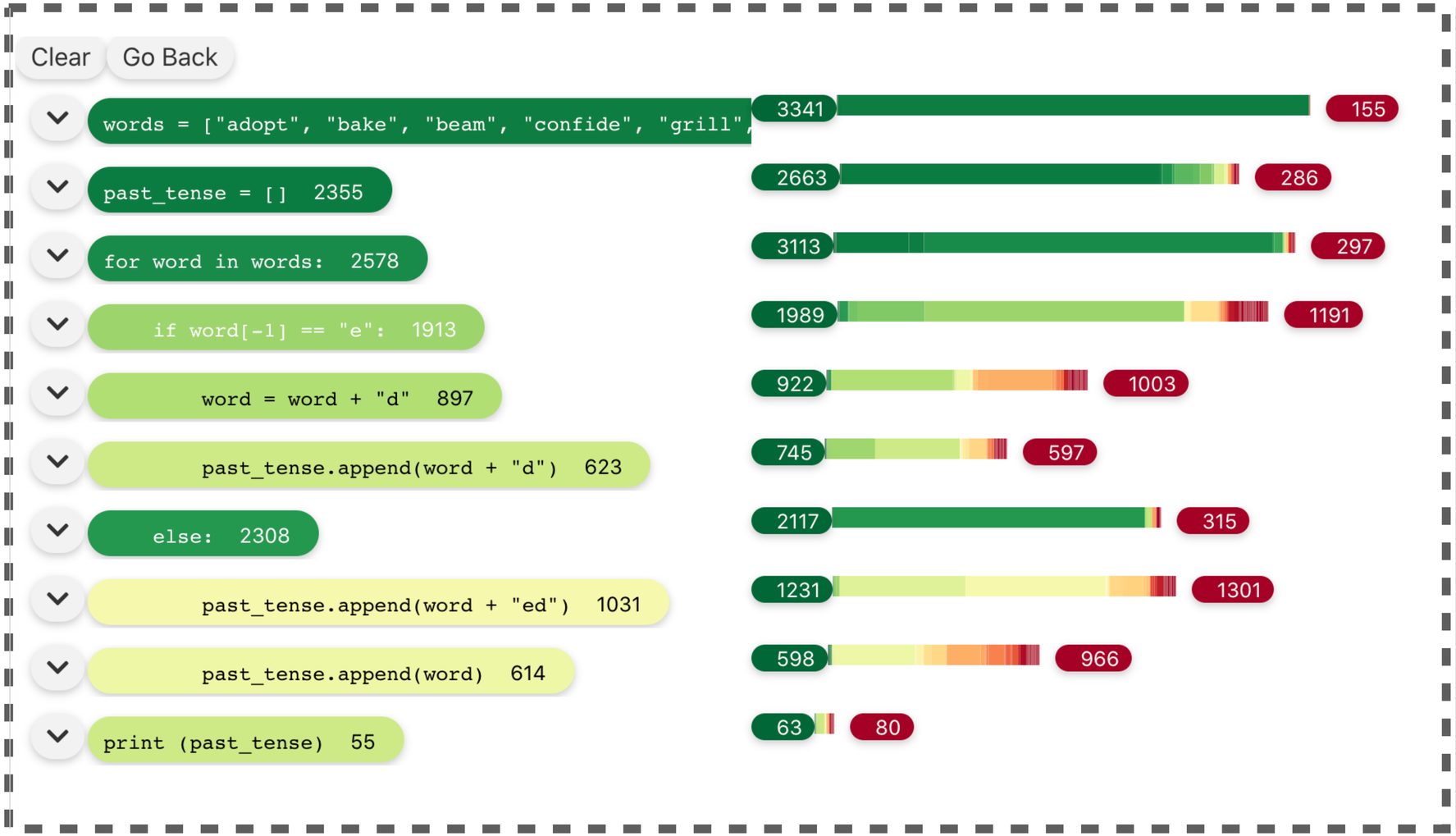
CFlow

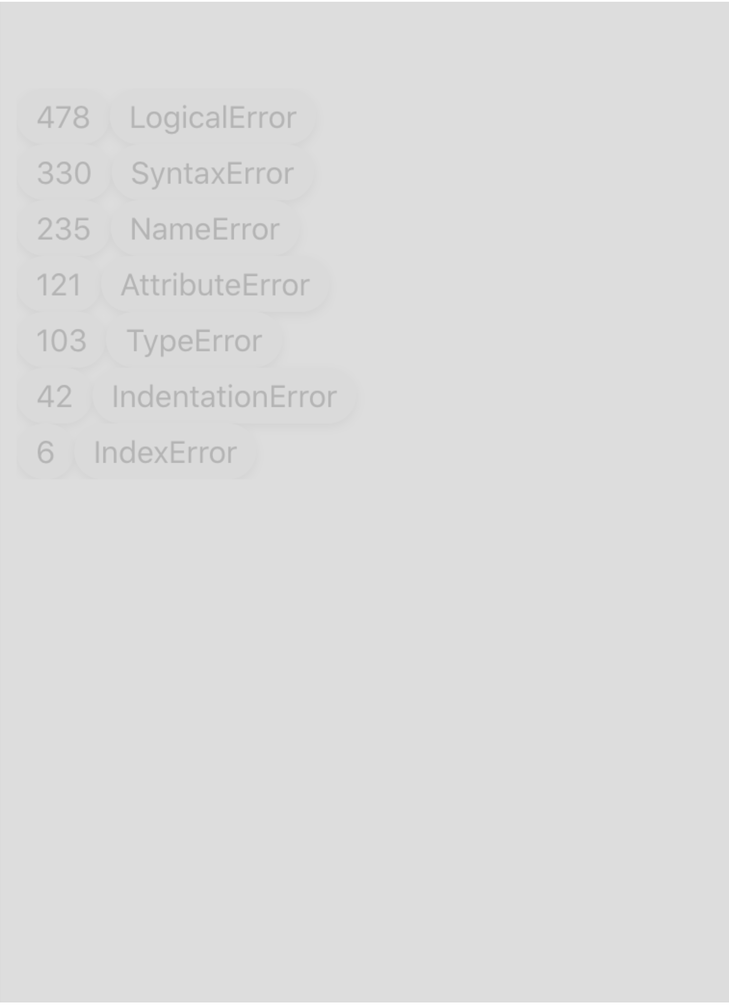




Code detailed view

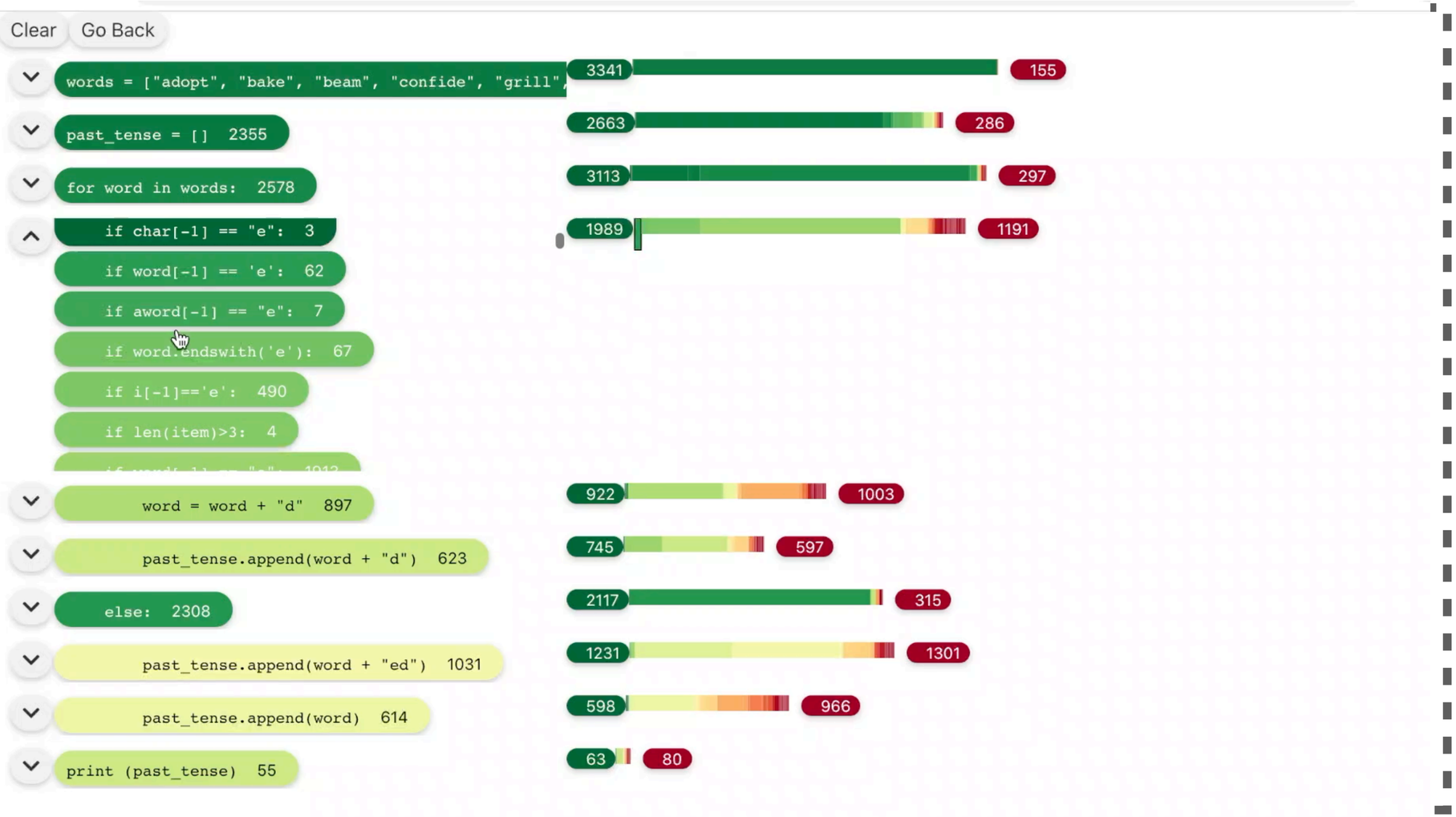
CFlow





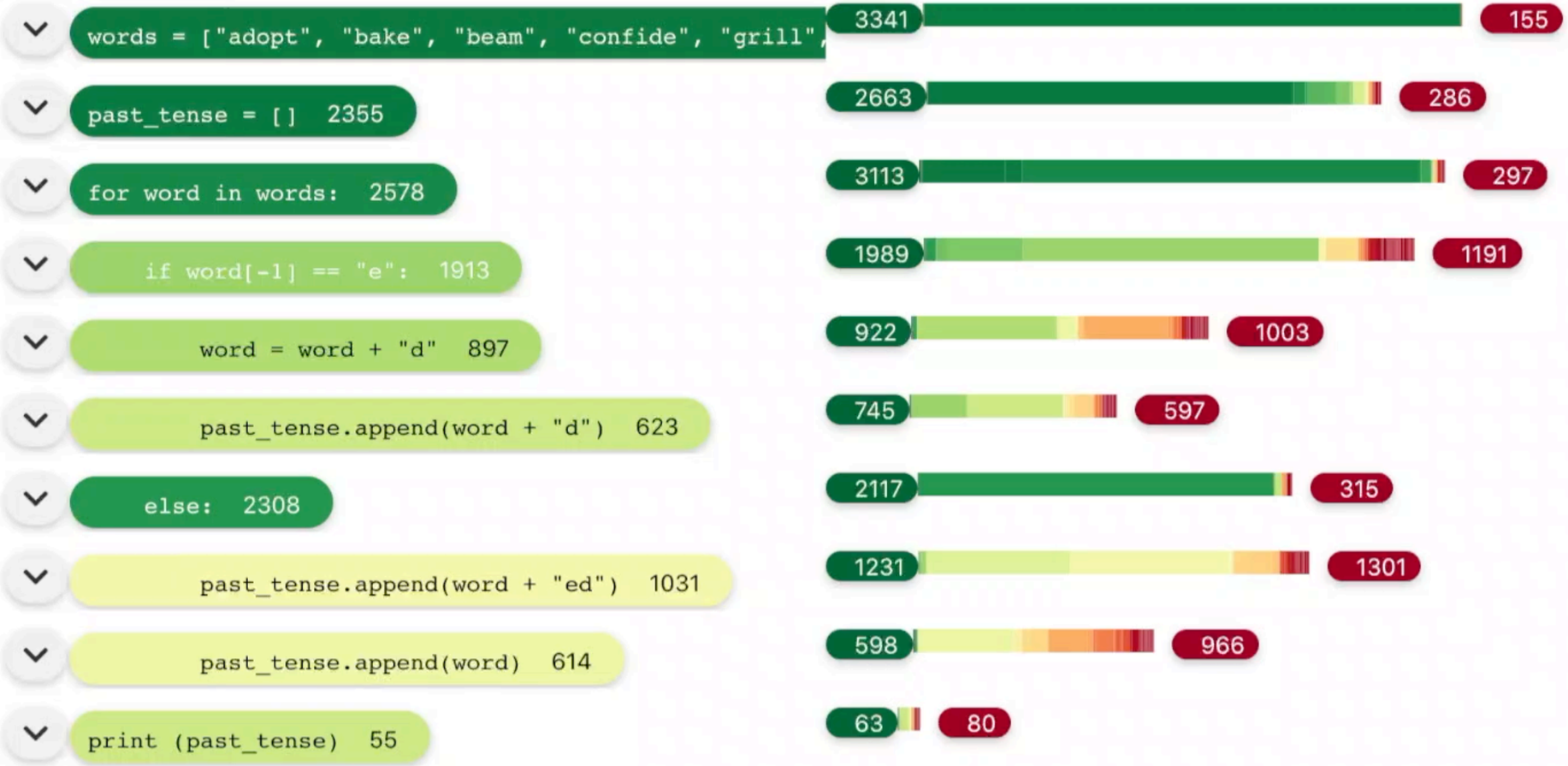
Semantic histogram view

CFlow



Semantic histogram view

Clear Go Back



Participants

Within-subjects study

16 participants

Teaching assistants (14/16)
Graduate students with > 3 years of experience using Python (2/16)

One 60-minute lab session

Baseline

Flow

View students' code
Answer quiz questions on students' mistakes and patterns

Questions

Is CFlow helpful in identifying students' mistakes?

How would CFlow influence instructor when viewing students' code?

Baseline

RunEx

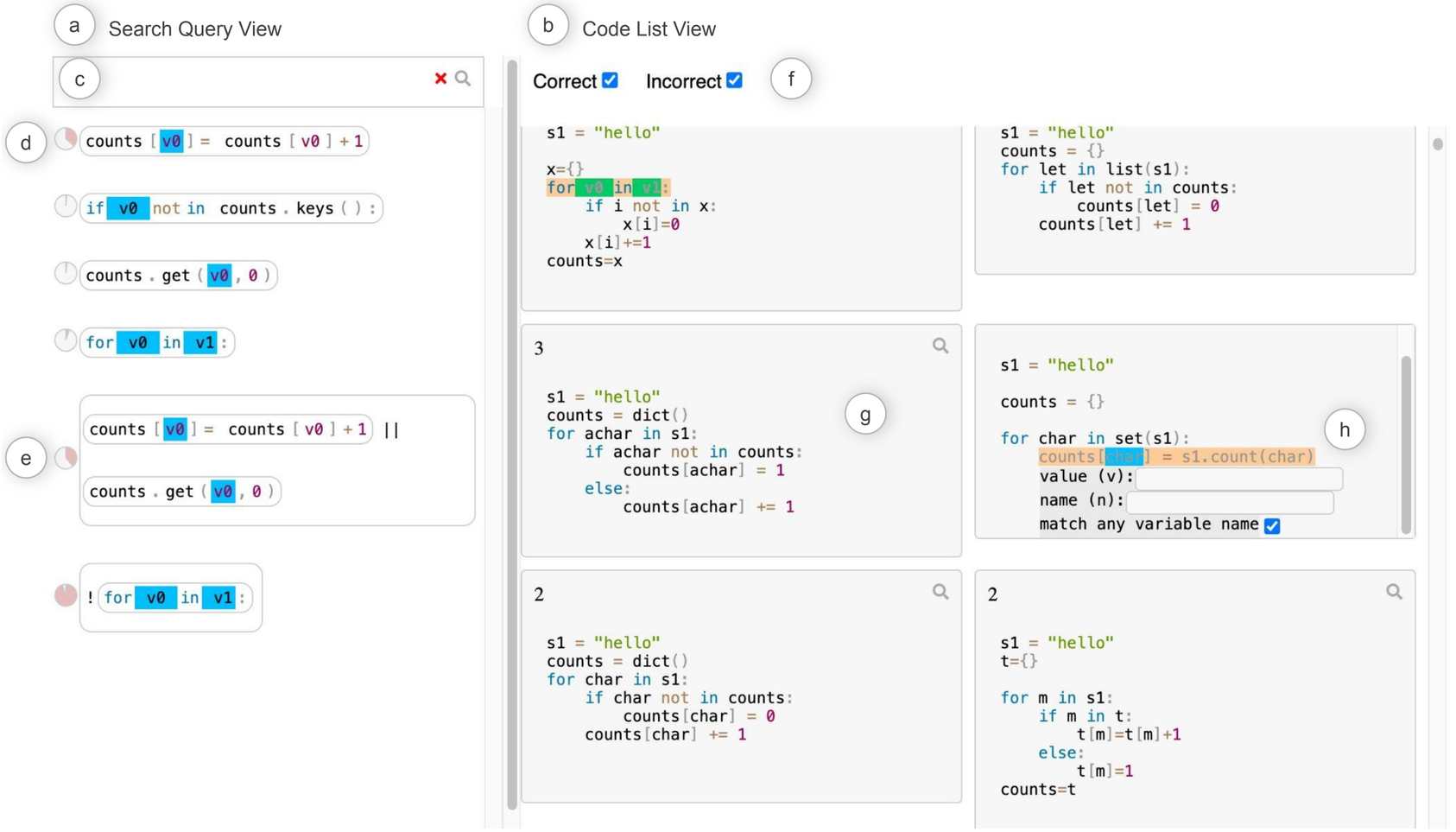
Group by code statements

+

OverCode

Group by entire code sample

Baseline


a
Search Query View
b
Code List View
c
Correct
Incorrect
f
d
counts [v0] = counts [v0] + 1
if v0 not in counts . keys ( ) :
counts . get (v0 , 0)
for v0 in v1 :
e
counts [v0] = counts [v0] + 1 ||
counts . get (v0 , 0)
! for v0 in v1 :
s1 = "hello"
x={}
for v0 in v1:
if i not in x:
x[i]=0
x[i]+=1
counts=x
s1 = "hello"
counts = {}
for let in list(s1):
if let not in counts:
counts[let] = 0
counts[let] += 1
3
s1 = "hello"
counts = dict()
for achar in s1:
if achar not in counts:
counts[achar] = 1
else:
counts[achar] += 1
g
s1 = "hello"
counts = {}
for char in set(s1):
counts[char] = s1.count(char)
value (v):
name (n):
match any variable name
h
2
s1 = "hello"
counts = dict()
for char in s1:
if char not in counts:
counts[char] = 0
counts[char] += 1
2
s1 = "hello"
t={}
for m in s1:
if m in t:
t[m]=t[m]+1
else:
t[m]=1
counts=t

Results

- Participants identify mistakes more accurately with less time using CFlow than using the baseline system

Accuracy (%)

	M	SD	P
CFlow	**93.02**	**0.06**	<0.0001
Baseline	52.40	0.18	

Time (second)

	M	SD	P
CFlow	**499.06**	**201.47**	<0.001
Baseline	817.50	264.85	

Results

- CFlow fosters an **exploratory and brainstorming** approach for instructors to understand students' code.

Baseline

(1) Random browsing
(2) Active searching without guidance

Flow

Semantic labels + color coded histograms

↓

Selectively dig deeper into the details

↓

Reason about students' misconceptions

Results

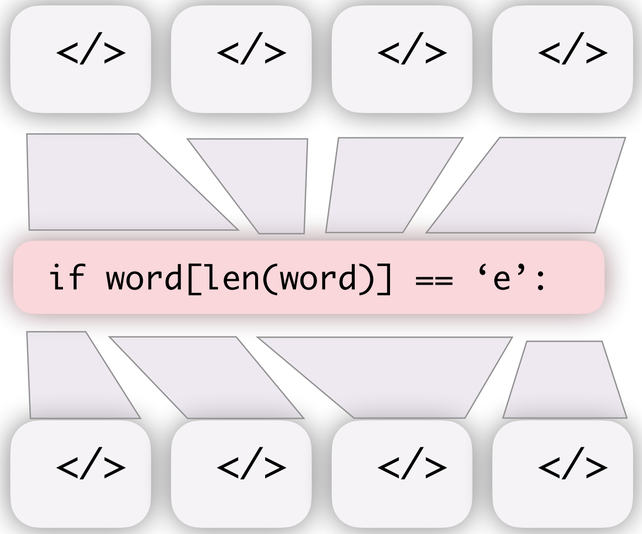
- CFlow requires **less context switch** when viewing the code.
  - Effortlessly locate the information they desire and seamlessly switch between abstraction and finer details

Design Takeaways

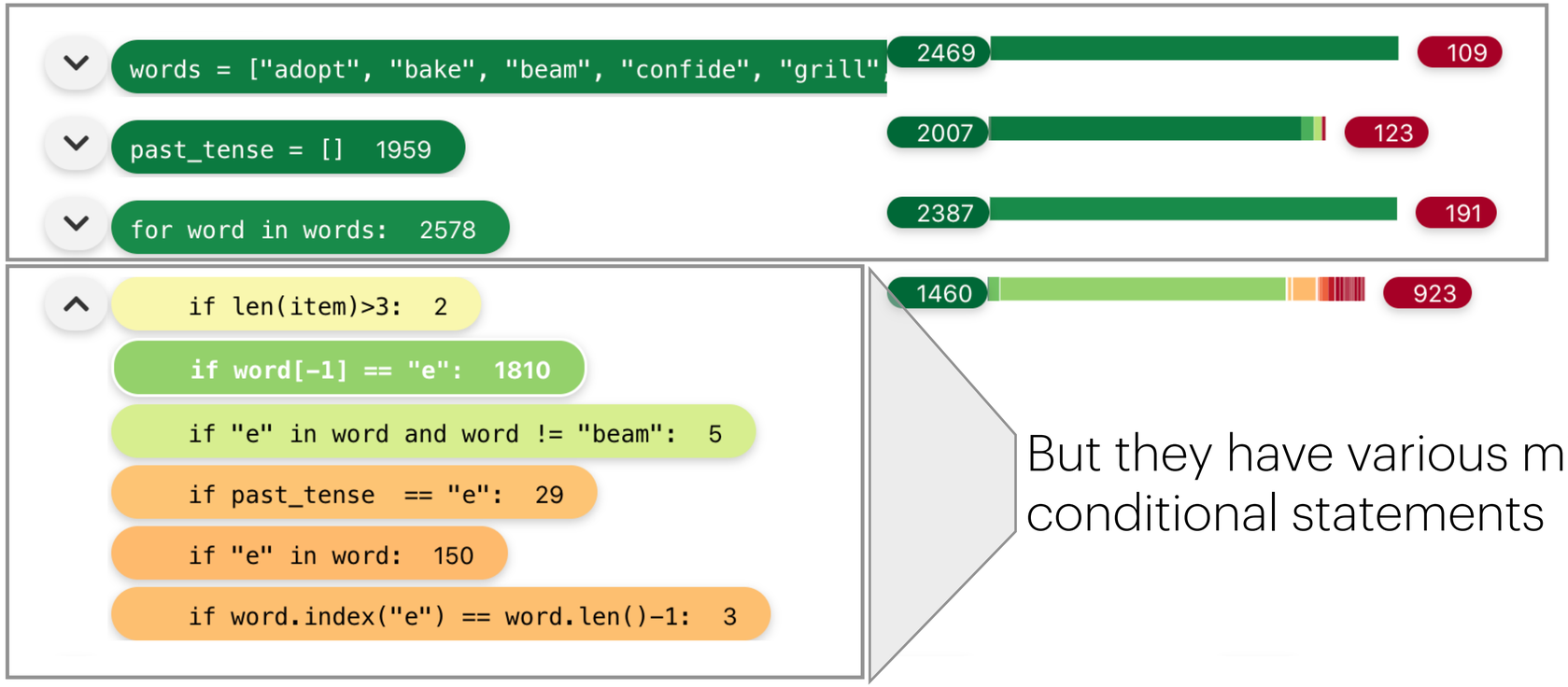
- Connect individual code samples with the entire dataset
  - Knowing the specifics of an individual submission is easy
  - Understanding different levels of variation is hard

```
words = [“egg”, “apple”, “tomato”]
e_words = []

for w in words:
    if word[len(word)] == ‘e’:
        e_words.append(w)
```

```
words = [“egg”, “apple”, “tomato”]
e_words = []

for w in words:
    if word[-1] == ‘e’:
        e_words.append(w)
```



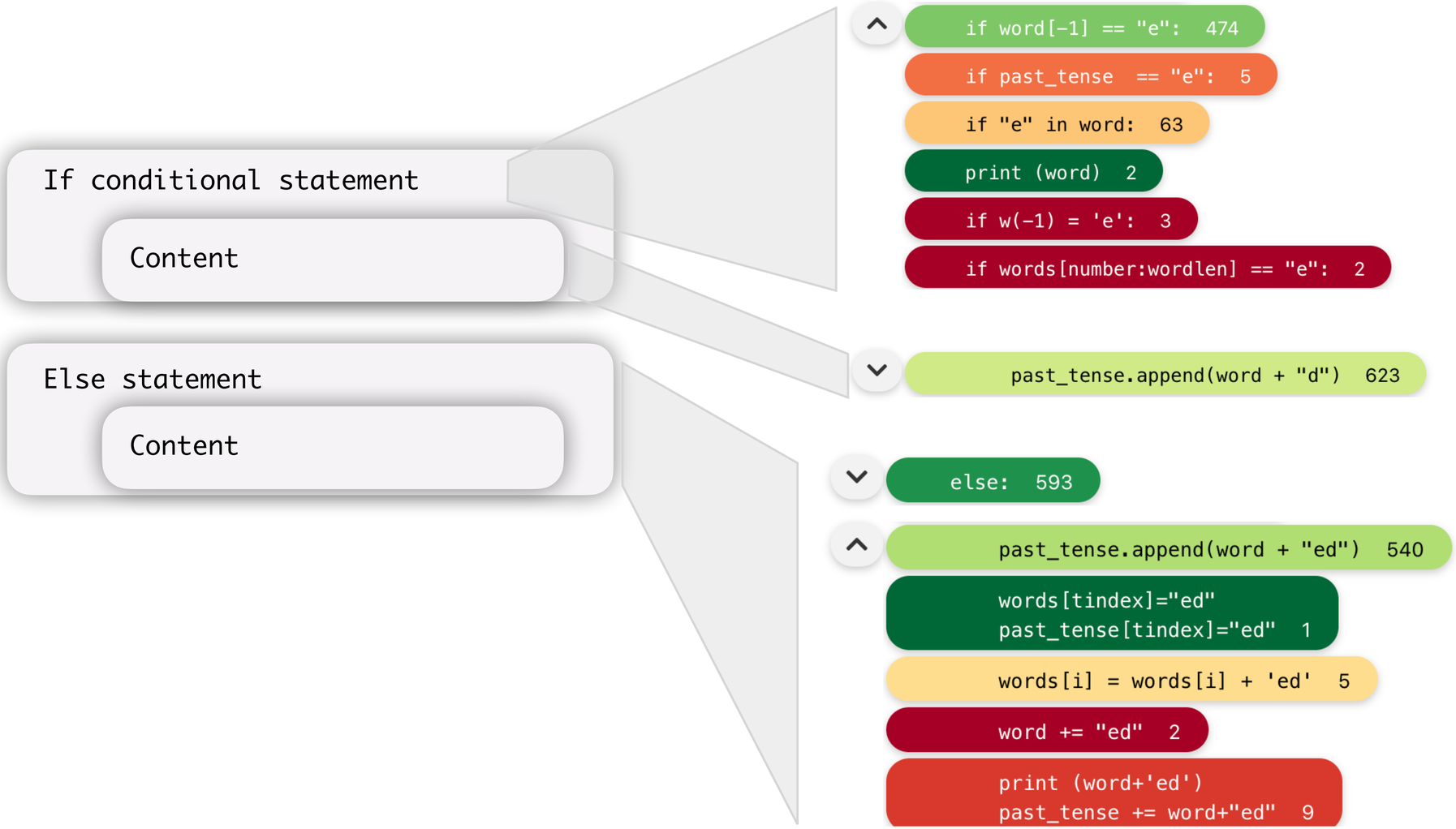
Design Takeaways



Most students get variable initializing and for loop correct

But they have various mistakes in the conditional statements

Design Takeaways



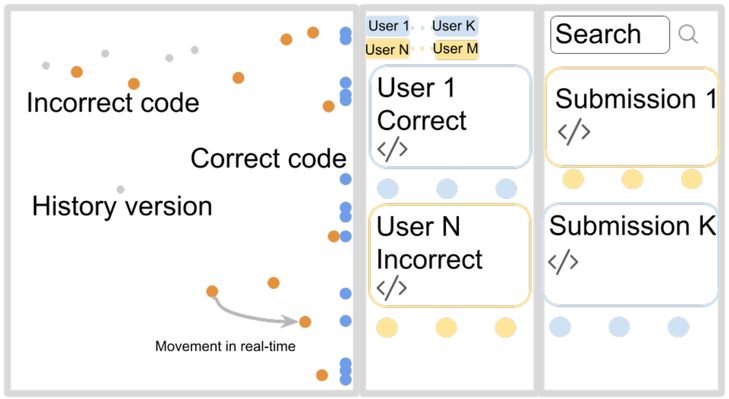
Design Takeaways

- Bridge abstraction and code details

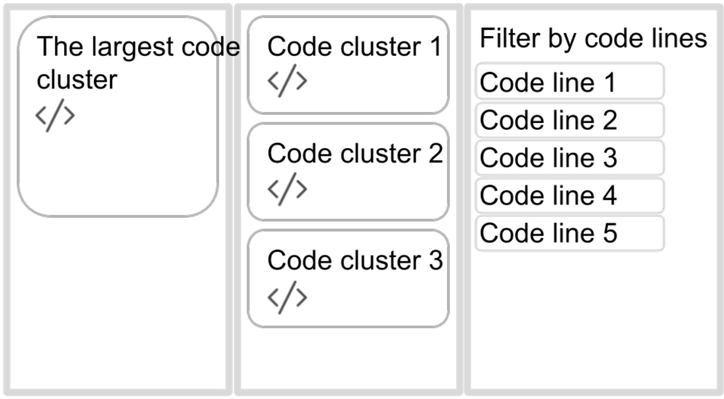
High-level overview of the entire solution set

Low-level details of individual solutions

VizProg



OverCode

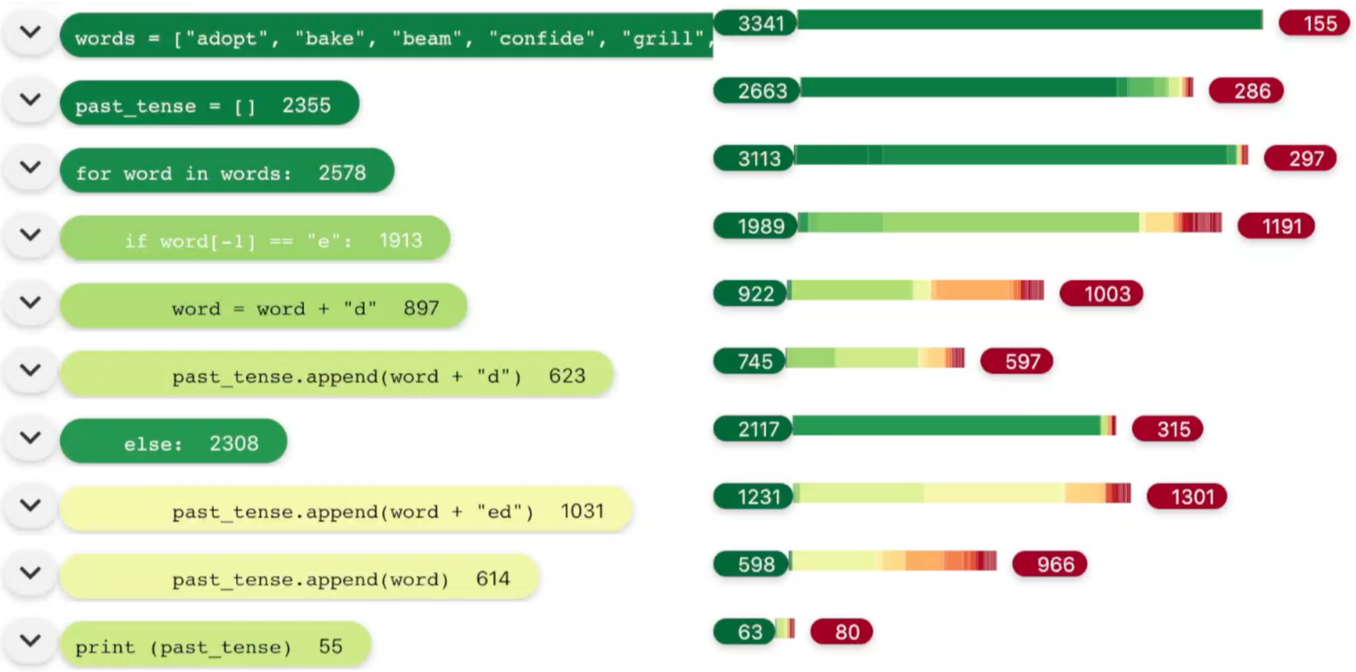


Design Takeaways

- Bridge abstraction and code details

High-level overview of the entire solution set

Low-level details of individual solutions



CFlow: Supporting **Semantic Flow** Analysis of Students' Code in Programming Problems **at Scale**

- A novel visualization approach, and implementation of the approach, that aggregate semantic patterns and code structures, to unearth complex in large-scale, multifaceted code submissions
- Evidence from an evaluation study showing that CFlow can assist users in identifying a variety of patterns and misconceptions in students' code.

Ashley Ge Zhang
University of Michigan

Xiaohang Tang
Virginia Tech

Steve Oney
University of Michigan

Yan Chen
Virginia Tech

https://gezhangrp.com

gezh@umich.edu