Interactive Extraction of Examples from Existing Code

Andrew Head, Elena L. Glassman,
Björn Hartmann, and Marti A. Hearst

UC Berkeley

Code Examples Are Everywhere

1.7 Recursive Functions

The sum of the digits of 18117 is `1+8+1+1+7 = 18`. Just as we can separate the number, v
this sum into the last digit, 7, and the sum of all but the last digit, `1+8+1+1 = 11`. This sepa
algorithm: to sum the digits of a number `n`, add its last digit `n % 10` to the sum of the digits
There's one special case: if a number has only one digit, then the sum of its digits is itself.
be implemented as a recursive function.

```
>>> def sum_digits(n):
        """Return the sum of the digits of positive integer n."""
        if n < 10:
            return n
        else:
            all_but_last, last = n // 10, n % 10
            return sum_digits(all_but_last) + last
```

222

This might be of interest:

```
function isElement(obj) {
  try {
    //Using W3 DOM2 (works for FF, Opera and Chrome)
    return obj instanceof HTMLElement;
  }
  catch(e){
    //Browsers not supporting W3 DOM2 don't have HTMLElement and
    //an exception is thrown and we end up here. Testing some
    //properties that all elements have (works on IE7)
    return (typeof obj==="object") &&
      (obj.nodeType===1) && (typeof obj.style === "object") &&
      (typeof obj.ownerDocument ==="object");
  }
}
```

It's part of the DOM, Level2.

12:54 PM **wmcgrath** Hey, per discussion in the group chat, when you have some ti
"focus on the next time that line runs" in the timeline? We had the func
complicated - here's the code from Bifrost

12:55 PM **wmcgrath** added this JavaScript/JSON snippet: **Untitled**

```
1       if(!codeEditedSinceLastTrace){
2          var orig_line = row + 1;
3          // console.log(orig_line)
4
5          var beginningEventIndex;
6          if(cur_line==undefined){
7            beginningEventIndex = 0;
8          } else {
9            beginningEventIndex = cur_line.event_index;
10         }
```

GitHubGist Search... All gists GitHub New gist

spint / disable-back-swipe.js

Created 5 years ago • Report gist

Star 5 Fork 0

Code Revisions 1 Stars 5 Embed <script src="https://gist Download ZIP

Disabling back history navigation with swipe (chrome)

disable-back-swipe.js Raw

```
1  // http://stackoverflow.com/questions/15829172/stop-chrome-back-forward-two-finger-swipe
2  $(document).on('mousewheel', function(e) {
3    var $target = $(e.target).closest('.scrollable-h');
4    if ($target.scrollLeft () <= 4) {
5      $target.scrollLeft(5);
6      return false;
7    }
8  });
9
10
```

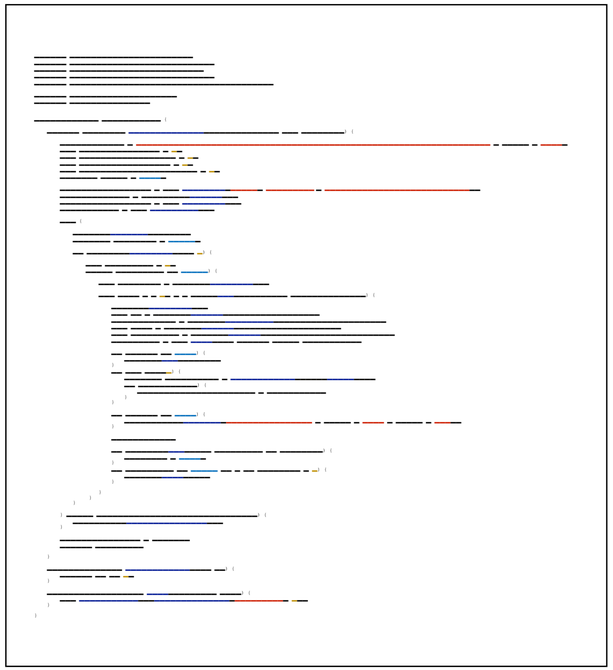
Why Examples Matter

People learn more and do more work by copying and following examples and by working their way through exercises than by any other single activity...

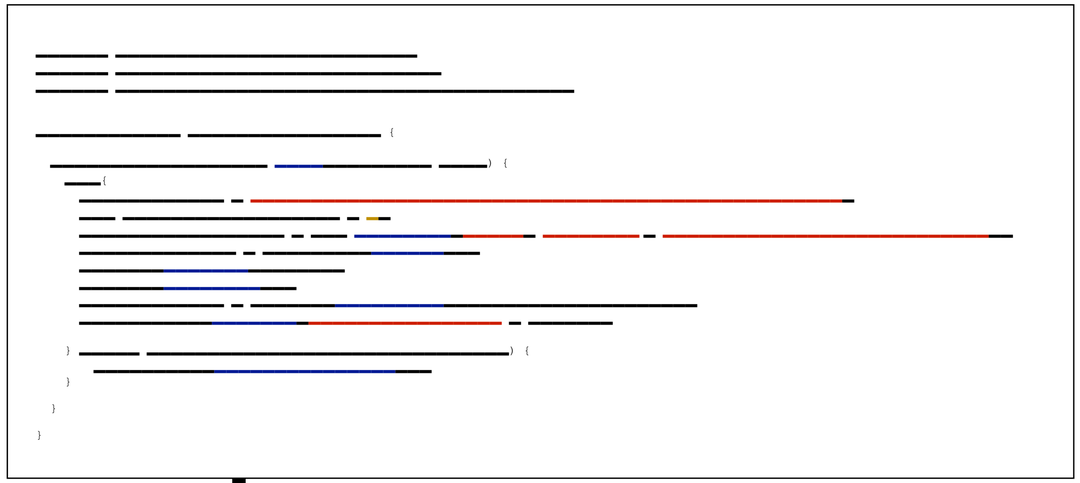
Marc Sacks, *On-the-Job Learning in the Software Industry*, 1994

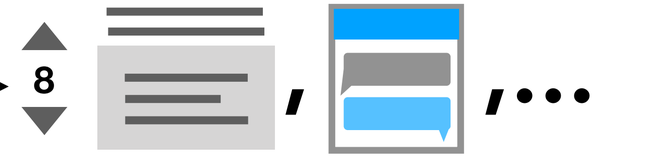
How can tools make it easier for programmers to share examples from their own code?

Detailed, personal code



Concise, self-contained example





Post online, share locally, ...

```
for (____ _ = _; _ < ____.___(________, ________); ++
    cursor.fetchone();
    int id = cursor.getInt(________________);
    ______ _____ = ______._______(_________________
    ___ ____ = ______.______(________________);
    ___ _______ = ______.______(________________)
    Book book = new Book(id, _____, ____, ________);
```

A row of data is fetched from the database.

```
for (____ _ = _; _ < ____.___(________, _______); ++
    cursor.fetchone();
    int id = cursor.getInt(________________);
    _____ _____ = ______._______(_________________
    ___ ____ = _____.______(________________);
    ___ ________ = ______.______(_____________)
    Book book = new Book(id, _____, ____, ________);
```

A row of data is fetched from the database.

This is data for a book.

```
for (int i = 0; i < Math.min(rowCount, maxBooks); ++
  cursor.fetchon
  int id = cursor.getInt(COLUMN_INDEX_ID);
  String title = cursor.getString(COLUMN_INDEX_T
  int year = cursor.getInt(COLUMN_INDEX_YEAR);
  int num_pages = cursor.getInt(COLUMN_INDEX
  Book book = new Book(id, title, year, num_pag
```

Video link: https://youtu.be/sIpSS-F1Ltg

Example Code Is Often...

Inadequate: Coders frequently face API learning obstacles due to inadequate examples (Robillard 2009).

Incomplete: It takes years to document all members of well-known APIs (Parnin et al. 2012).

Broken: Often, code examples don't compile (Terragni 2016) and lack important details (Treude and Robillard 2017).

Supporting the Sharing of Example Code

Example Extraction:	Buse and Weimer 2012, Montandon et al. 2013, Moreno et al. 2015
Example Creation:	Oezbek and Prechelt 2007 , Ginosar et al. 2013, Kojouharov et al. 2004
Example Search:	Hoffman et al. 2007, Stylos et al. 2007, Brandt et al. 2010
Example Reuse:	Cottrell et al. 2008, Oney and Brandt 2012, Wightman et al. 2012, Doerner et al. 2014

Supporting the Sharing of Example Code

Example Extraction: Buse and Weimer 2012, Montandon et al. 2013, Moreno et al. 2015

Example Creation: Oezbek and Prechelt 2007 , Ginosar et al. 2013, Kojouharov et al. 2004

Example Search: Hoffman et al. 2007, S

Example Reuse: Cottrell et al. 2008, O al. 2012, Doerner et a

CodeScoop blends these two areas!

It uses **both automation and human input**, to produce concise, executable examples.

```
33
34  int rowNumber = 0;
35  while (finished == false) {
36
37      int rowCount = cursor.rowCount();
38
39      for (int i = 0; i < Math.min(rowCount, maxBooks); ++i) {
40
41          cursor.fetchone();
42          int id = cursor.getInt(COLUMN_INDEX_ID);
43          String title = cursor.getString(COLUMN_INDEX_TITLE);
44          int year = cursor.getInt(COLUMN_INDEX_YEAR);
45          int num_pages = cursor.getInt(COLUMN_INDEX_NUM_PAGES);
46          Book book = new Book(id, title, year, num_pages);
47
48          if (title != null) {
49              titles.add(title);
50          }
51          if (id != -1) {
```

Scoop

Undo

Run

Reset

```
int rowNumber = 0;
while (finished == false) {

    int rowCount = cursor.rowCount();

    for (int i = 0; i < Math.min(rowCount, maxBooks); ++i) {

        cursor.fetchone();
        int id = cursor.getInt(COLUMN_INDEX_ID);
                       etString(COLUMN_INDEX_TITLE);
                       t(COLUMN_INDEX_YEAR);
                       getInt(COLUMN_INDEX_NUM_PAGES);
                       , title, year, num_pages);

        if (id != -1) {
```

(1) **User** selects tasty pattern

Scoop

Undo

Run

Reset

```
33
34  int rowNumber = 0;
35  while (finished == false) {
36
37      int rowCount = cursor.rowCou
38
39      for (int i = 0; i < Math.min
40
41          cursor.fetchone();
42          int id = cursor.getInt(C
43
44
45
46
47
48
49
50
51          if (id != -1) {
```

(1) **User** selects tasty pattern
(2) **Editor** creates example,

```
1  public class ExtractedExample {
2
3      public static void main(String[
4
5          int id = cursor.getInt(COLUMN
6
7      }
8
9  }
```

Scoop

Undo

Run

Reset

```
33
34  int rowNumber = 0;
35  while (finished == false) {
36
37      int rowCount = cursor.rowCou
38
39      for (int i = 0; i < Math.min
40
41          cursor.fetchone();
42          int id = cursor.getInt(C
43
44
45
46
47
48
49
50
51          if (id != -1) {
```

(1) **User** selects tasty pattern
(2) **Editor** creates example,
(3) Flags errors,

```
1  public class ExtractedExample {
2
3      public static void main(String[
4
5          int id = cursor.getInt(COLUMN
6
7      }
8
9  }
```

Scoop

Undo

Run

Reset

```
21
22      Database database = new Databa
23      Cursor cursor = database.curs
24      Booklist booklist = new Bookl
25      List titles = new ArrayList()
26
27      try {
28
29          cursor.execute(QUERY);
30          boolean finished = false;
31
32
33
34
35
36
37
38
39                  for (int i = 0; i
```

(1) **User** selects tasty pattern
(2) **Editor** creates example,
(3) Flags errors,
(4) Suggests code fixes,

```
1  public class ExtractedExample {
2
3      public static void main(String[
4
5          int id = cursor.getInt(COLUMN
6
7      }
8
9  }
```

Add code | **Stub out**

Line 23

Scoop

Undo

Run

Reset

```
40
41                      cursor.fet
42                      int id = c
43                      String tit
44                      int year =
45                      int num_pa
46                      Book book
47
48                      if (title
49                          titles
50
51
52
53
54
55
56
57
58                      if (DEBUG
```

(1) **User** selects tasty pattern
(2) **Editor** creates example,
(3) Flags errors,
(4) Suggests code fixes,
(5) Suggests simplifications,

```
1   dExample {
2
3   main(String[] args) {
4
5   etInt(0);
6   ook(id, title, year, num_pages);
7
8
9
10
```

Define

Scoop

Undo

Run

Reset

```
40
41                      cursor.fet
42                      int id = c
43                      String tit
44                      int year =
45                      int num_pa
46                      Book book
47
48                      if (title
49                          titles
50
51
52
53
54
55
56
57
58                      if (DEBUG
```

(1) **User** selects tasty pattern
(2) **Editor** creates example,
(3) Flags errors,
(4) Suggests code fixes,
(5) Suggests simplifications,

```
1   dExample {
2
3   main(String[] args) {
4
5   etInt(0);
6   ook(id, title, year, 330);
7
8
9
10
```

Add code | Set value

250

200

330

Scoop

Reset

```
36
37                  int rowCount
38
39                  for (int i =
40
41                      cursor.f
42                      int id =
43                      String t
44                      int year
45                      int num_
```

(1) **User** selects tasty pattern
(2) **Editor** creates example,
(3) Flags errors,
(4) Suggests code fixes,
(5) Suggests simplifications,
(6) And makes automatic fixes.

```
54
```

```
1  import org.acme.database.Databas
2  import org.acme.database.Cursor;
3  import org.acme.database.Book;
4
5  public class ExtractedExample {
6
7    public static void main(String
8
9      Database database = new Data
10     Cursor cursor = database.cur
11     cursor.execute("SELECT id, t
12     cursor.fetchone();
13     int id = cursor.getInt(0);
14     Book book = new Book(id, "Da
15
16   }
17
18 }
```

Scoop

Undo

Run

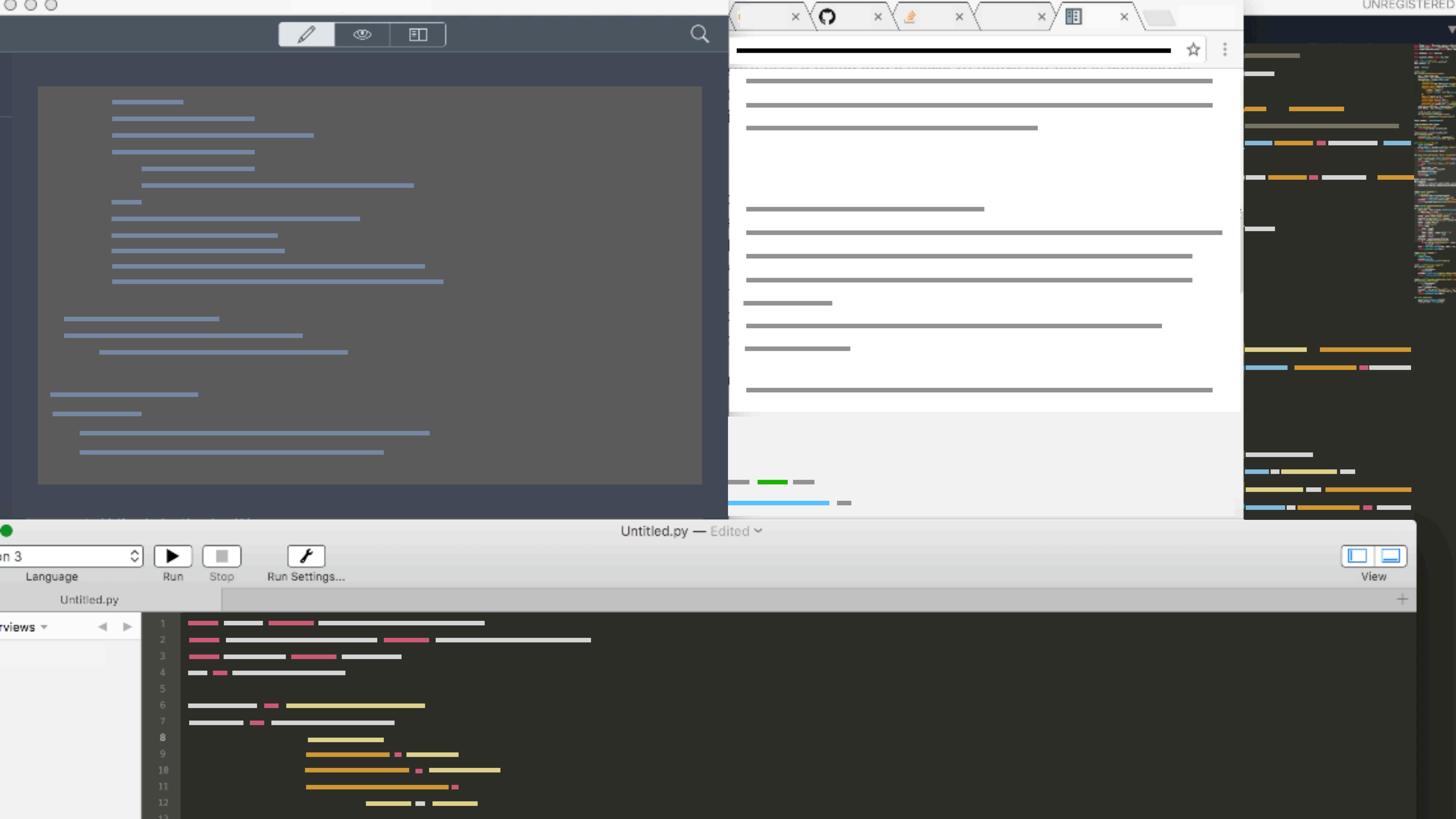
Reset

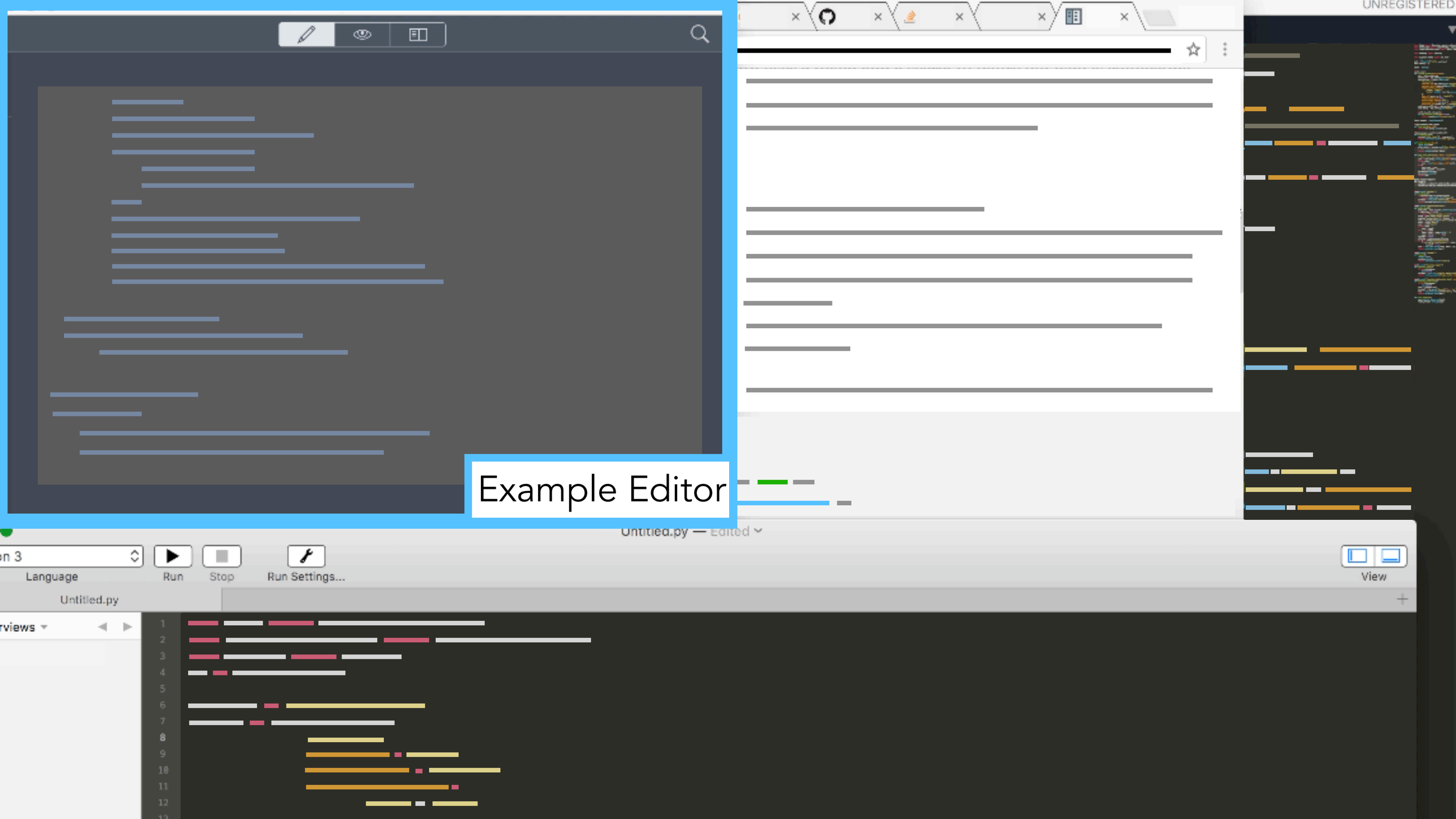
This Talk

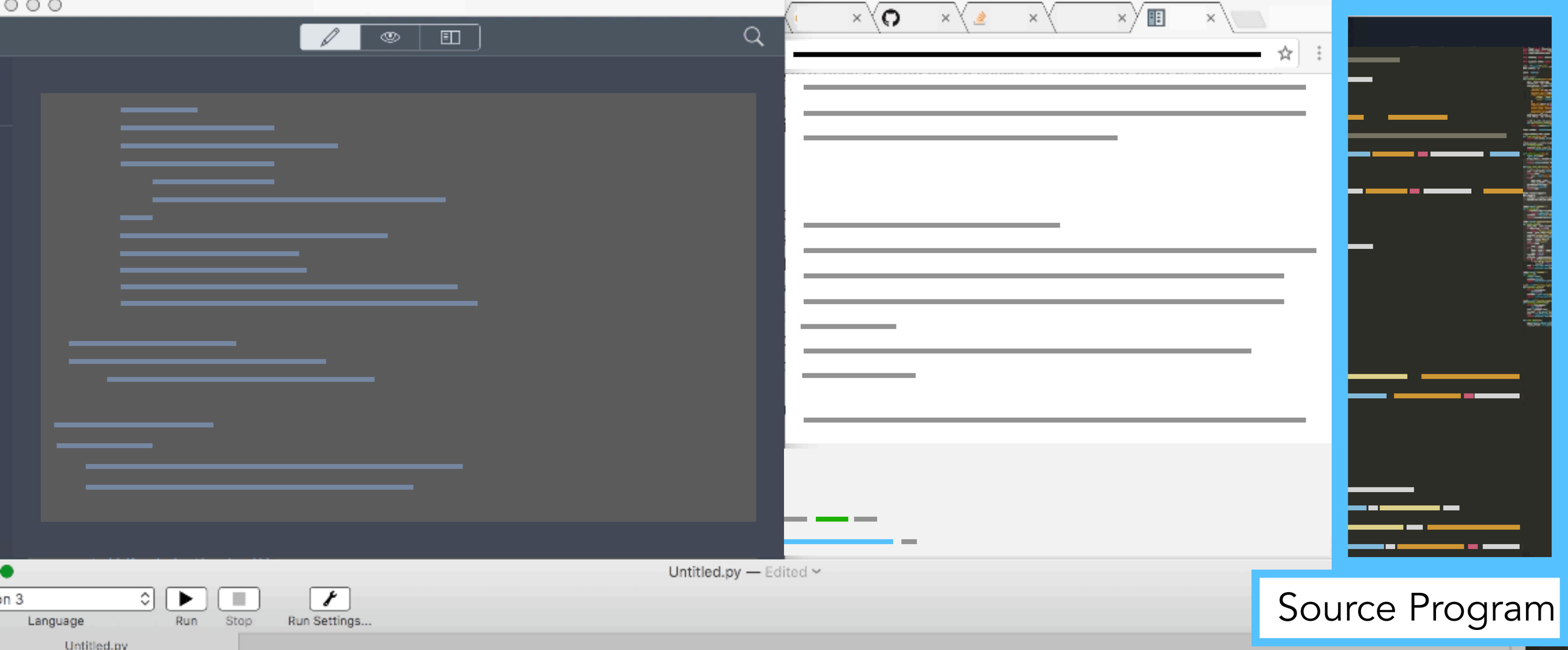
- ~~Motivation~~
- ~~Related Work~~
- ~~Tool Preview~~
- Formative Study
- Prototype
- Evaluation

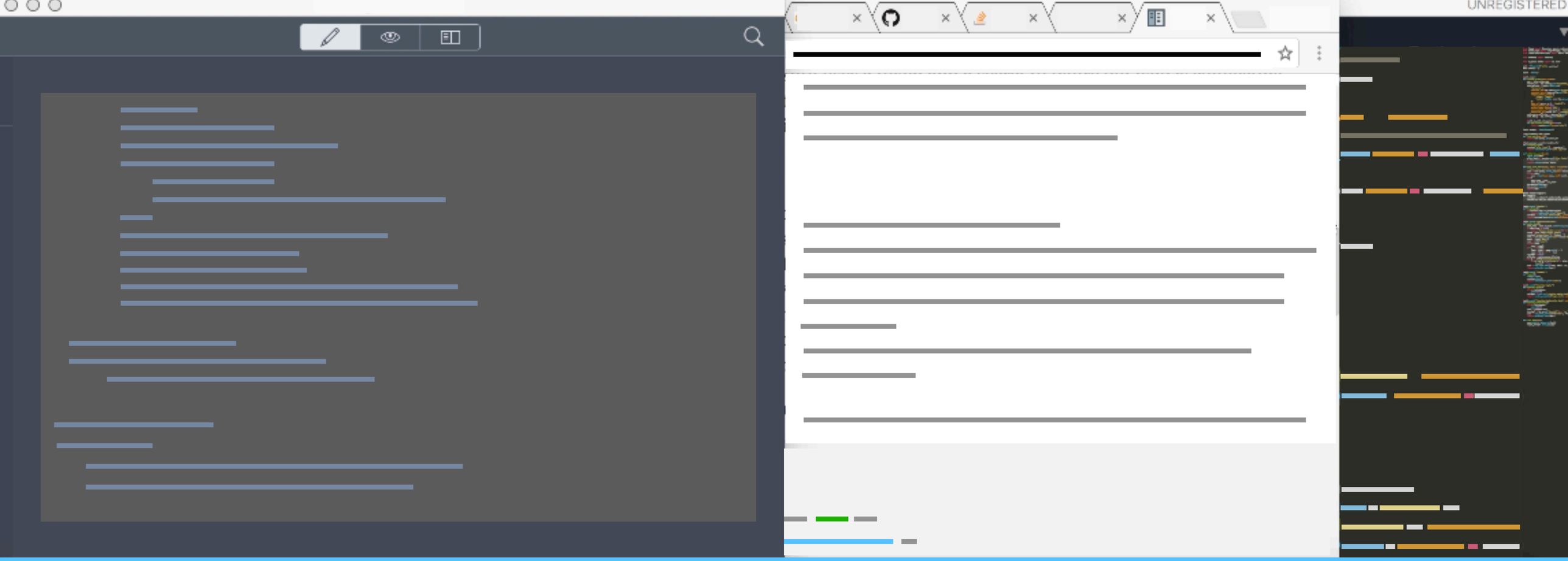
Formative Study

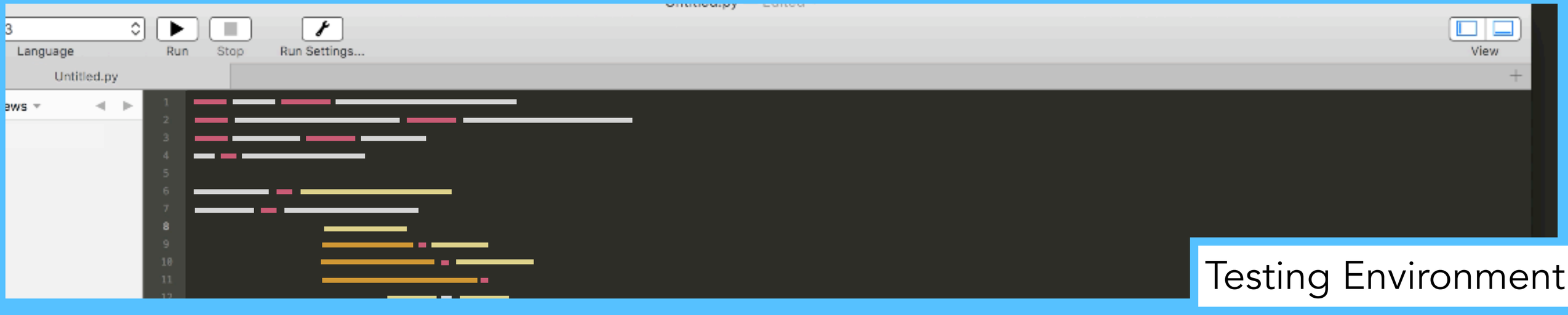
12 programmers creating examples from existing code.


Untitled.py — Edited
Language
Run
Stop
Run Settings...
View
Untitled.py

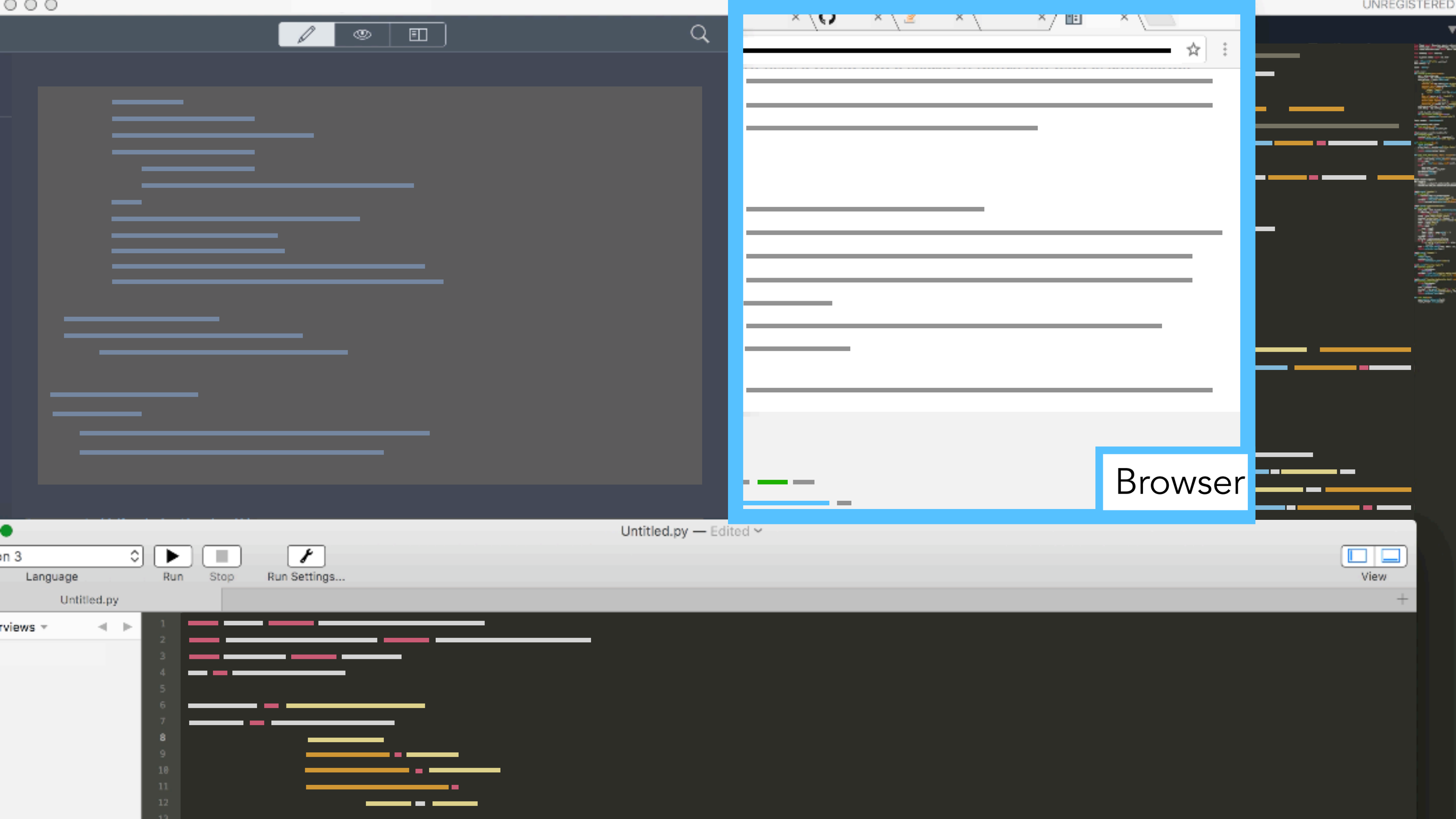

Example Editor
Untitled.py — Edited
Language
Run
Stop
Run Settings...
View
Untitled.py

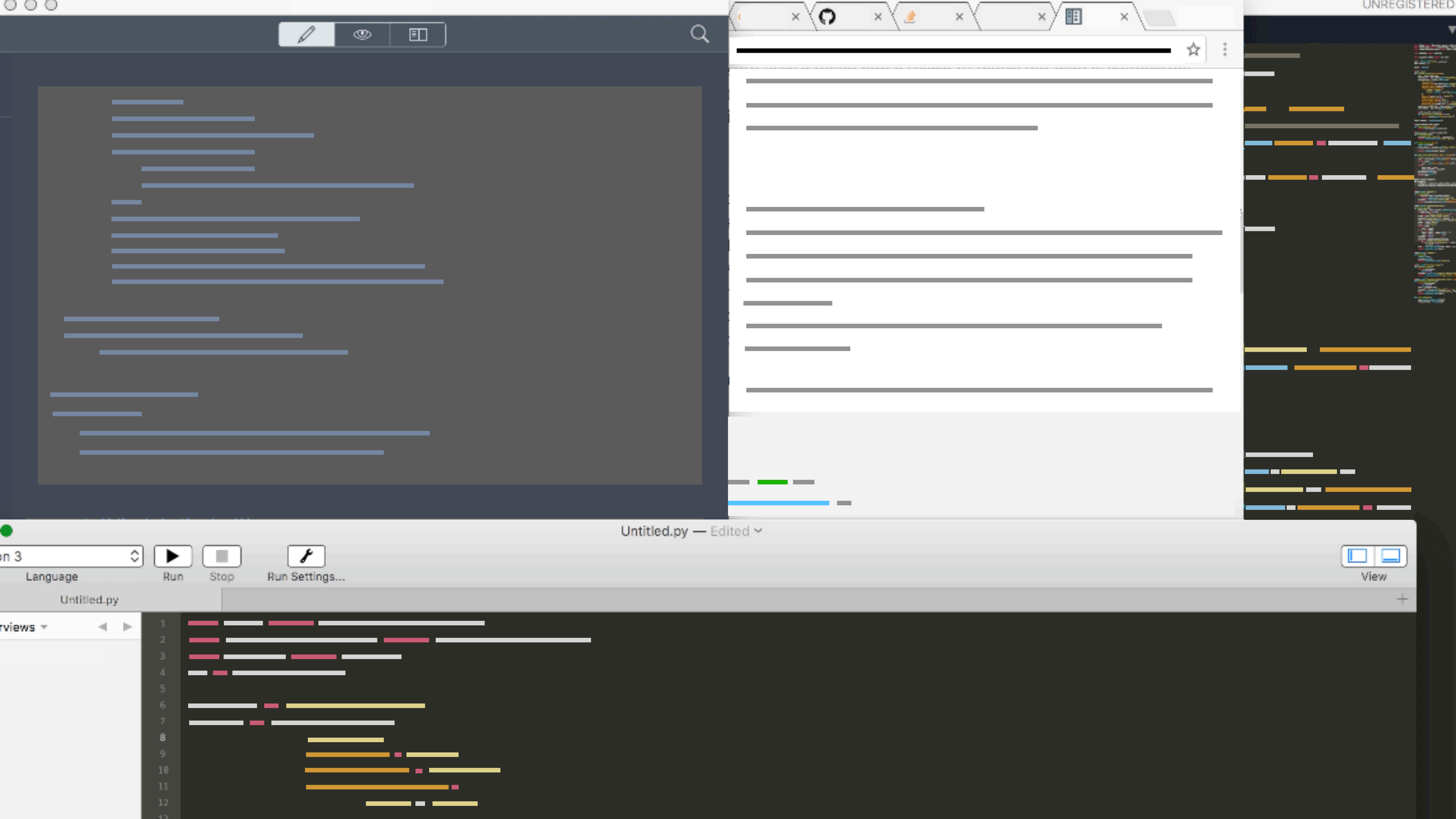

Untitled.py — Edited
Language
Run
Stop
Run Settings...
Untitled.py
Source Program

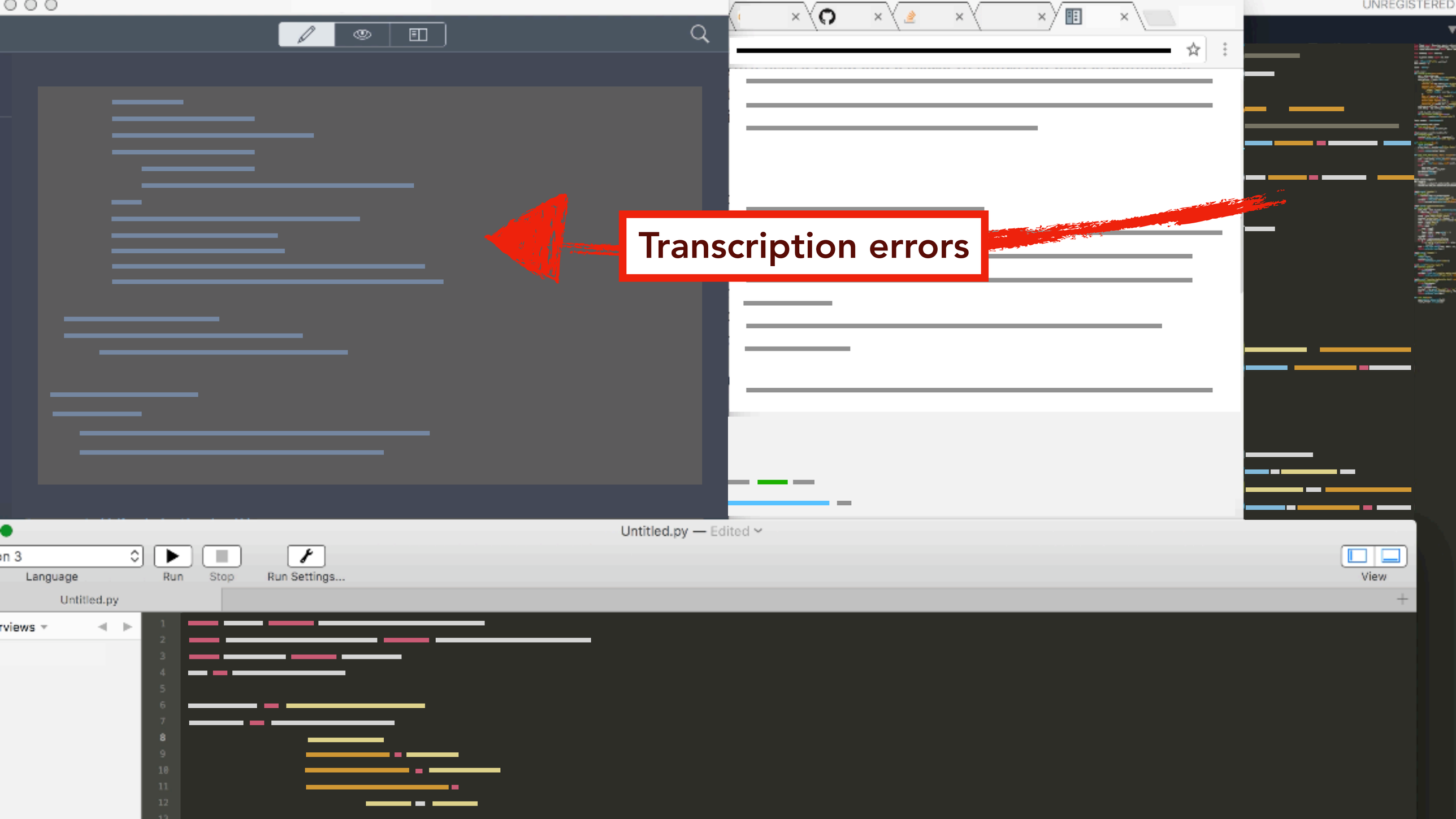


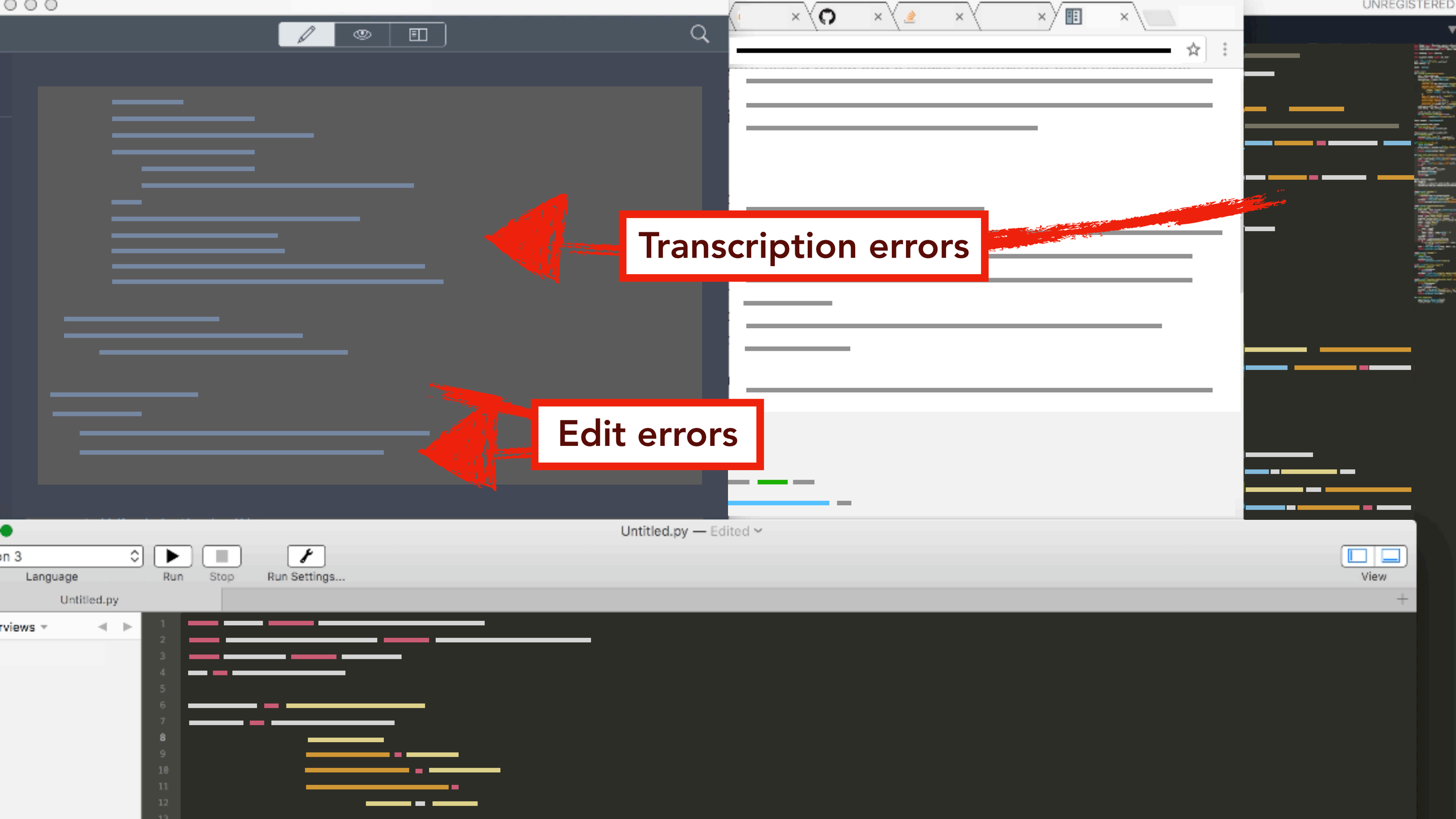


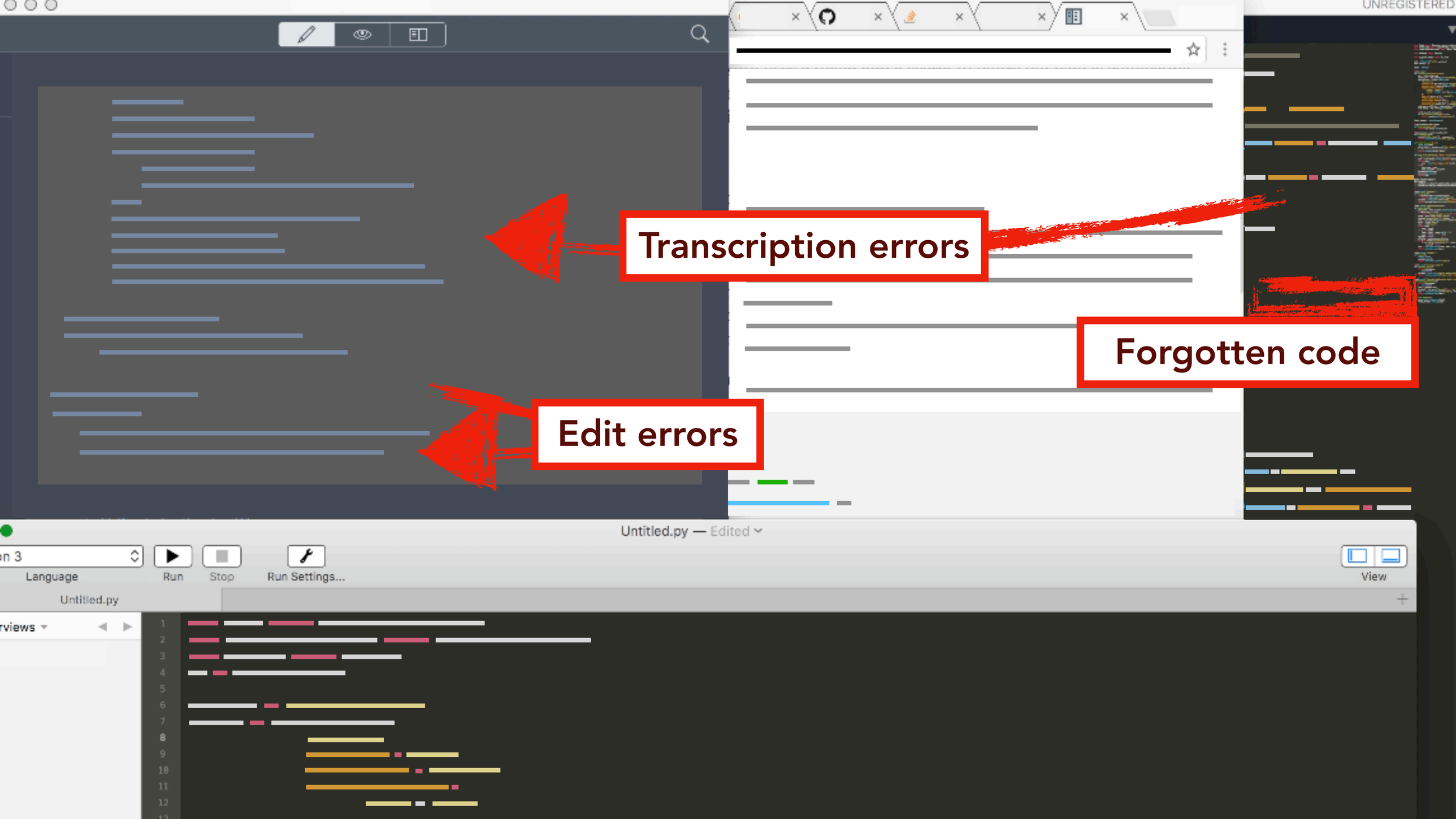
Testing Environment

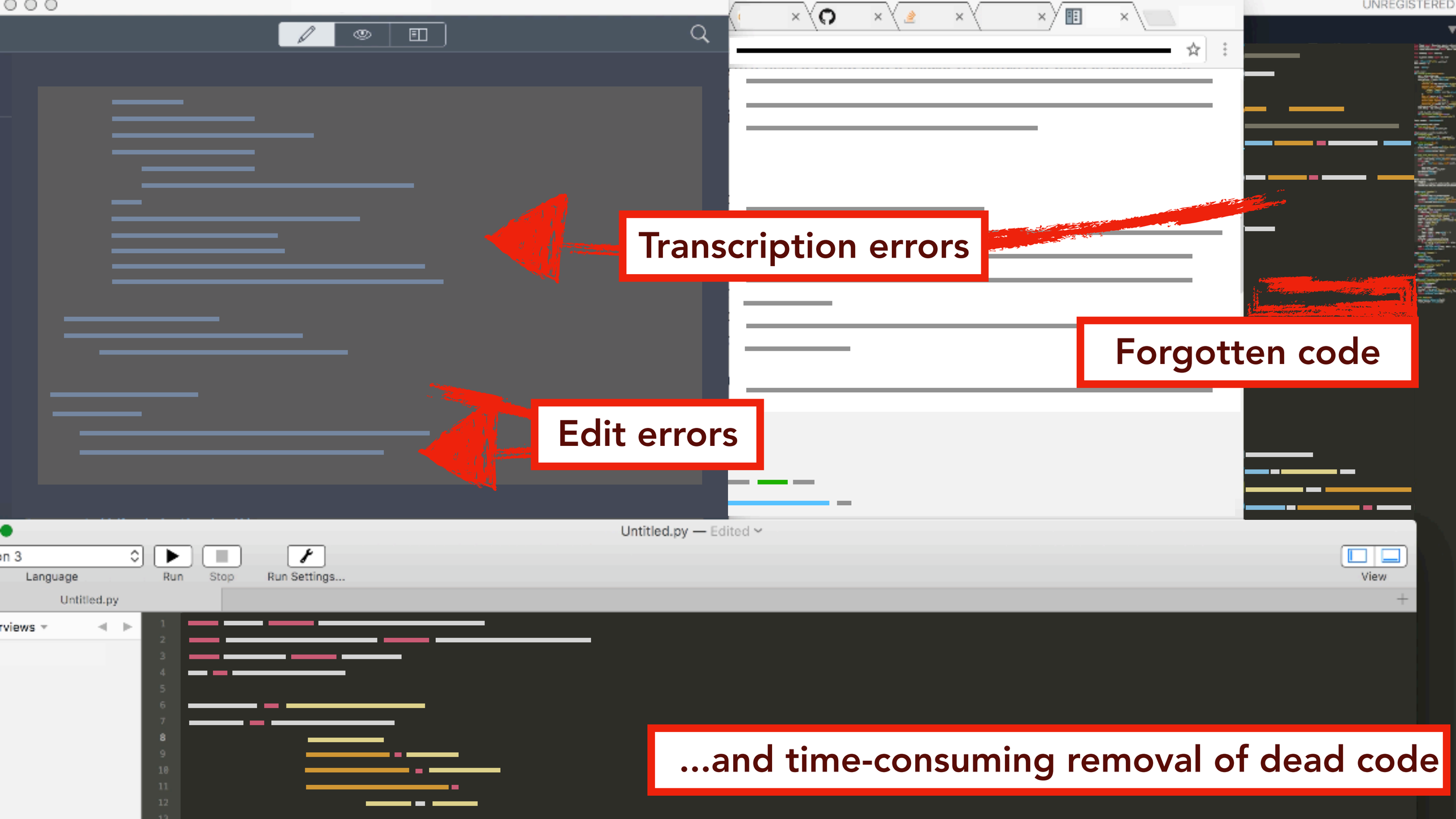

Browser
Untitled.py — Edited
Language
Run
Stop
Run Settings...
View
Untitled.py


Untitled.py — Edited
Language
Run
Stop
Run Settings...
View
Untitled.py


Transcription errors
Untitled.py — Edited
Language
Run
Stop
Run Settings...
View
Untitled.py


Transcription errors
Edit errors
Untitled.py — Edited
Language
Run
Stop
Run Settings...
View
Untitled.py


Transcription errors
Forgotten code
Edit errors
Untitled.py — Edited
Language
Run
Stop
Run Settings...
View
Untitled.py

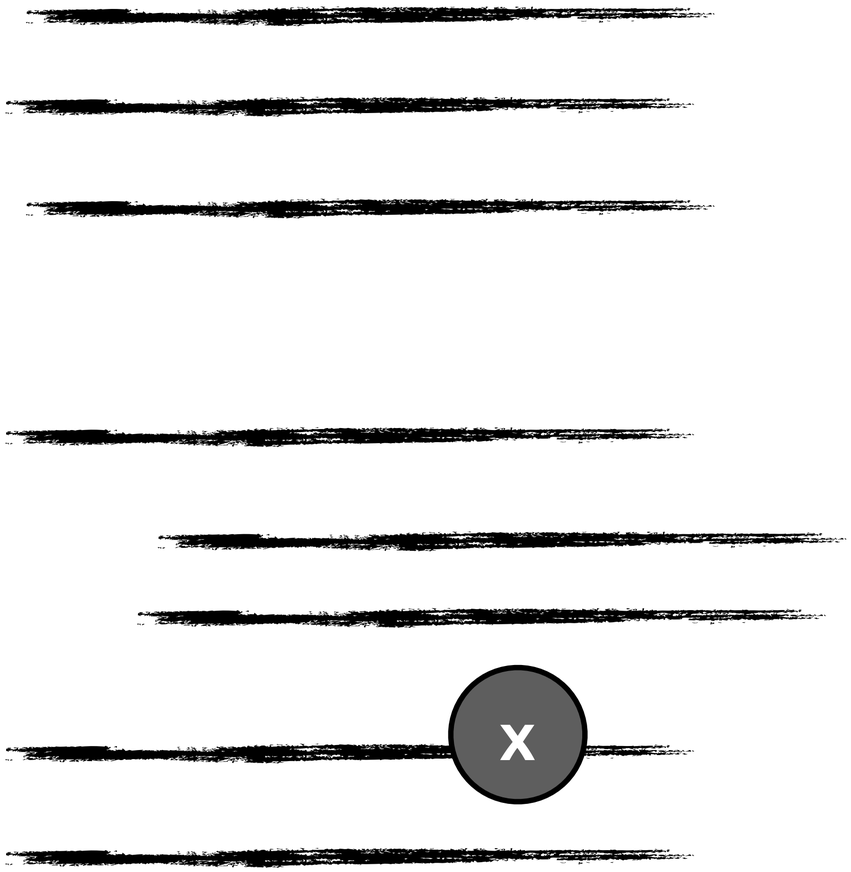

Transcription errors
Forgotten code
Edit errors
Untitled.py — Edited
Language
Run
Stop
Run Settings...
View
Untitled.py
...and time-consuming removal of dead code

RQ. How can tools make it easier for programmers to share examples from their own code?

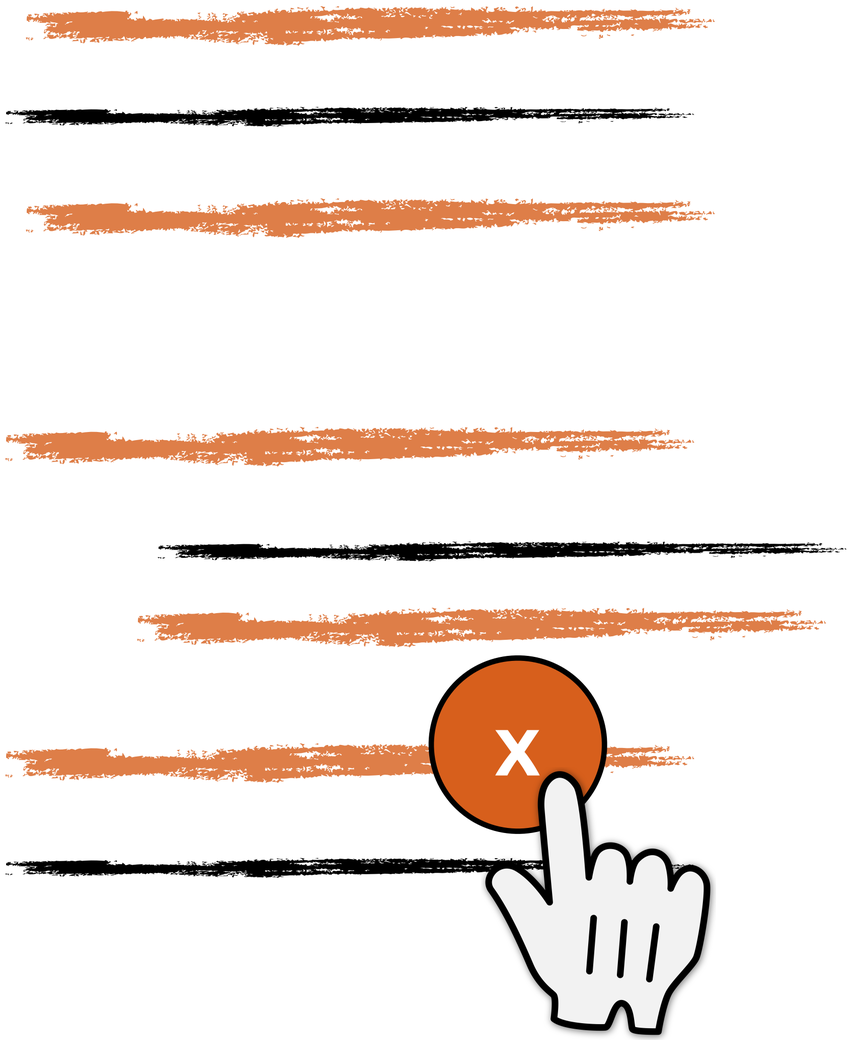
We hypothesize that tools should:

- Suggest lines of code that the current example needs to run
- Constrain manual code edits
- Enable early and frequent testing
- Omit code except for explicit code selections and fixes

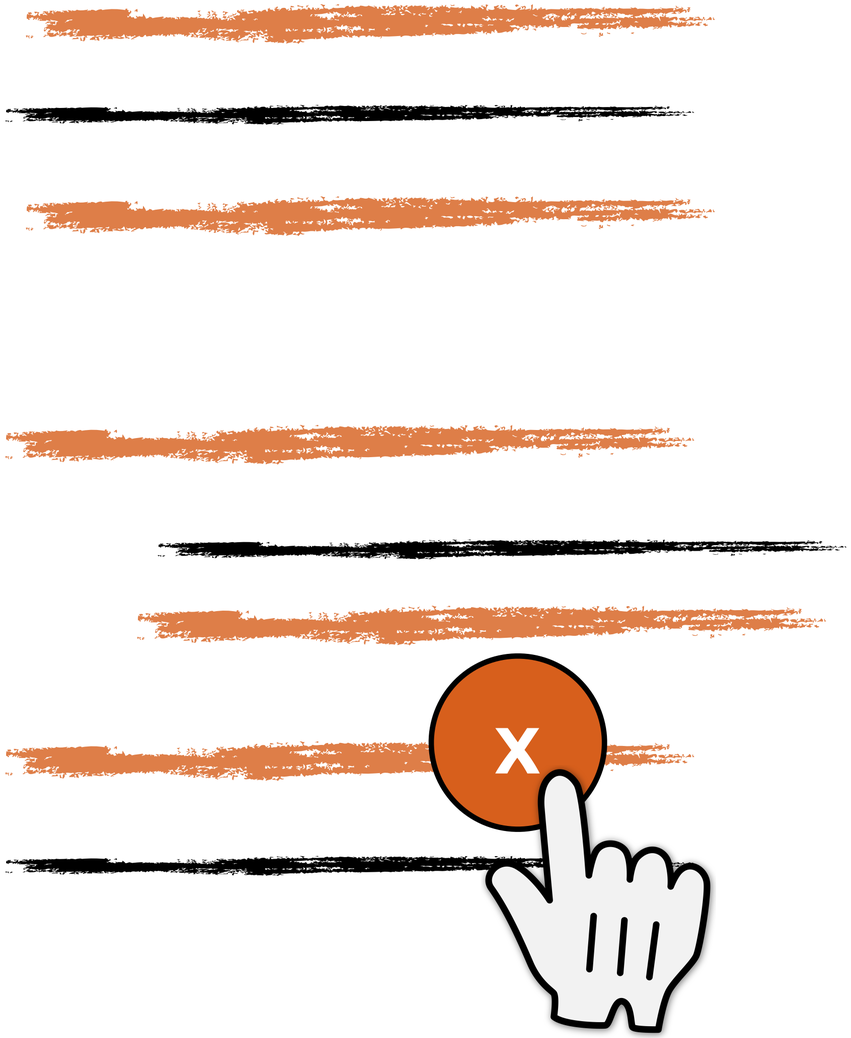
Program **Slicing**



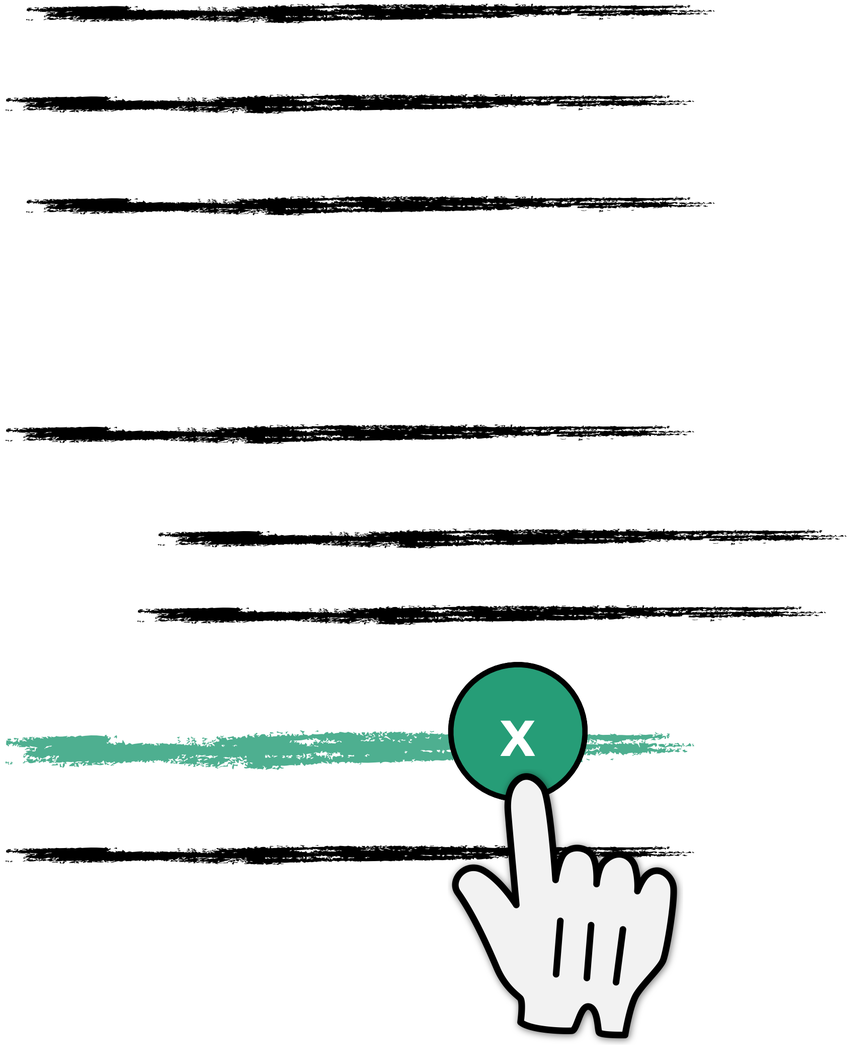
Program **Slicing**



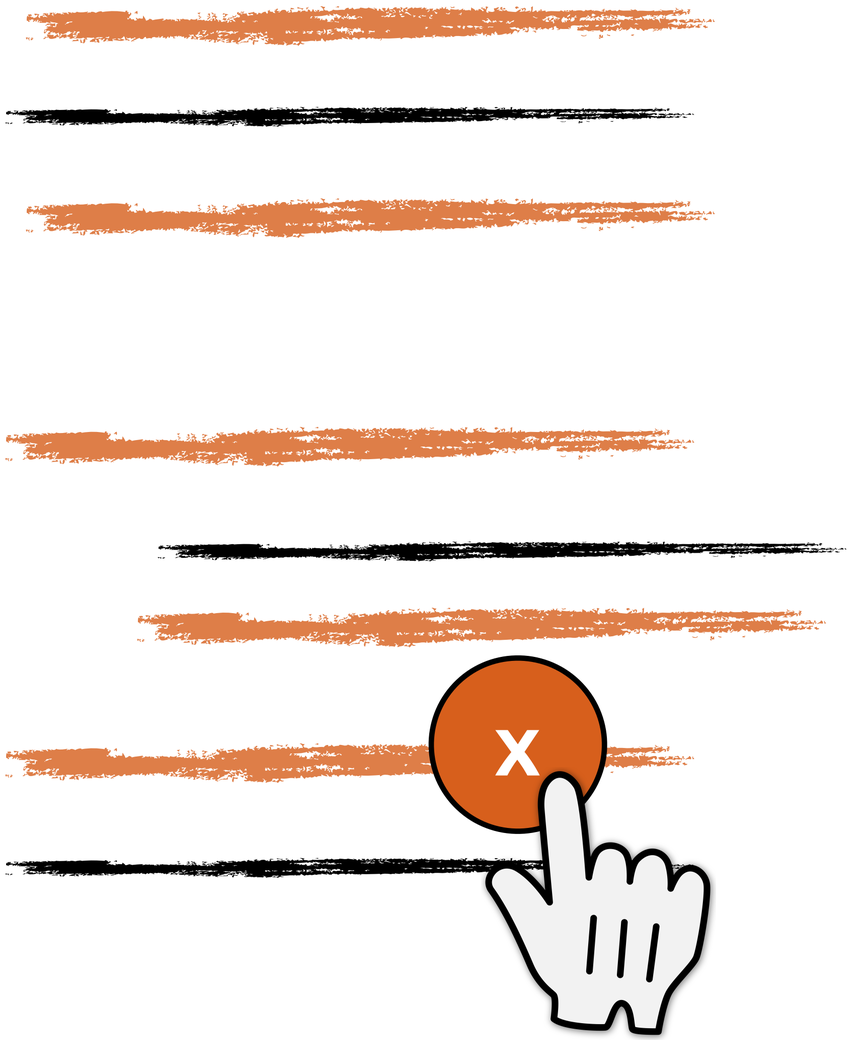
Program **Slicing**



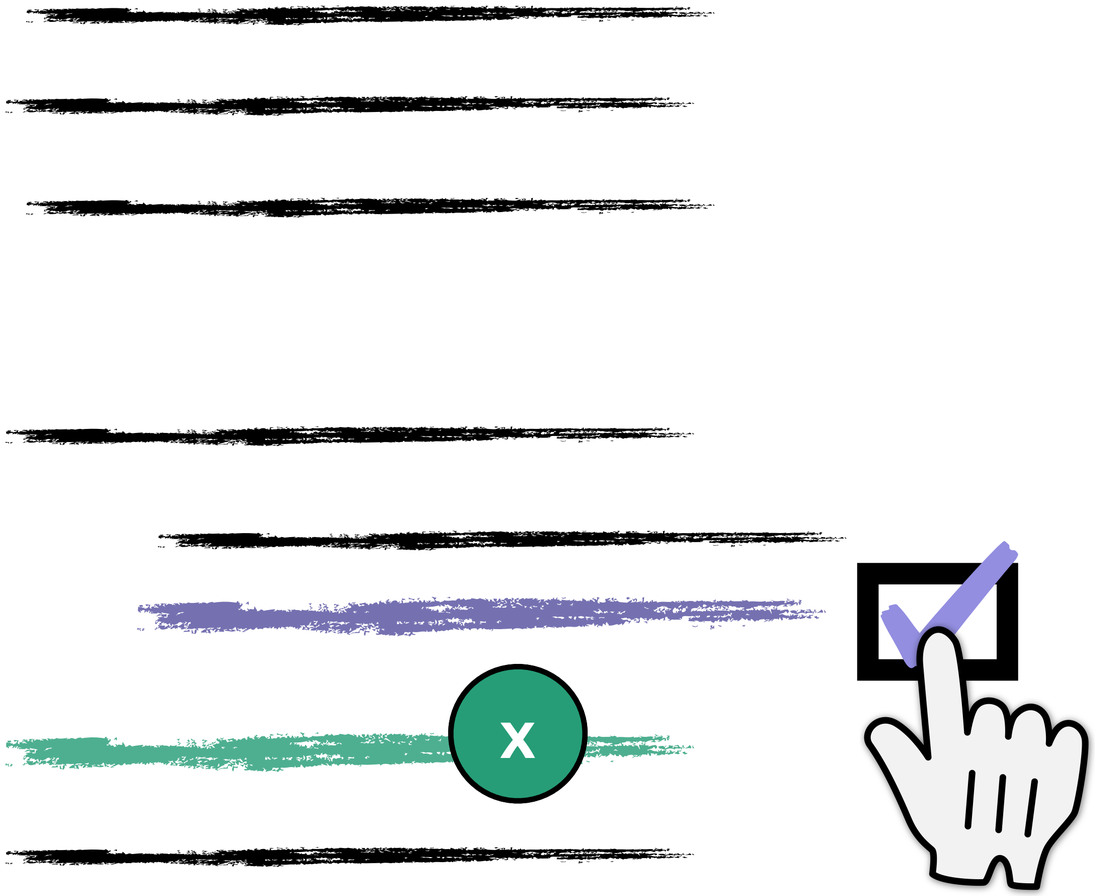
Program **Scooping**



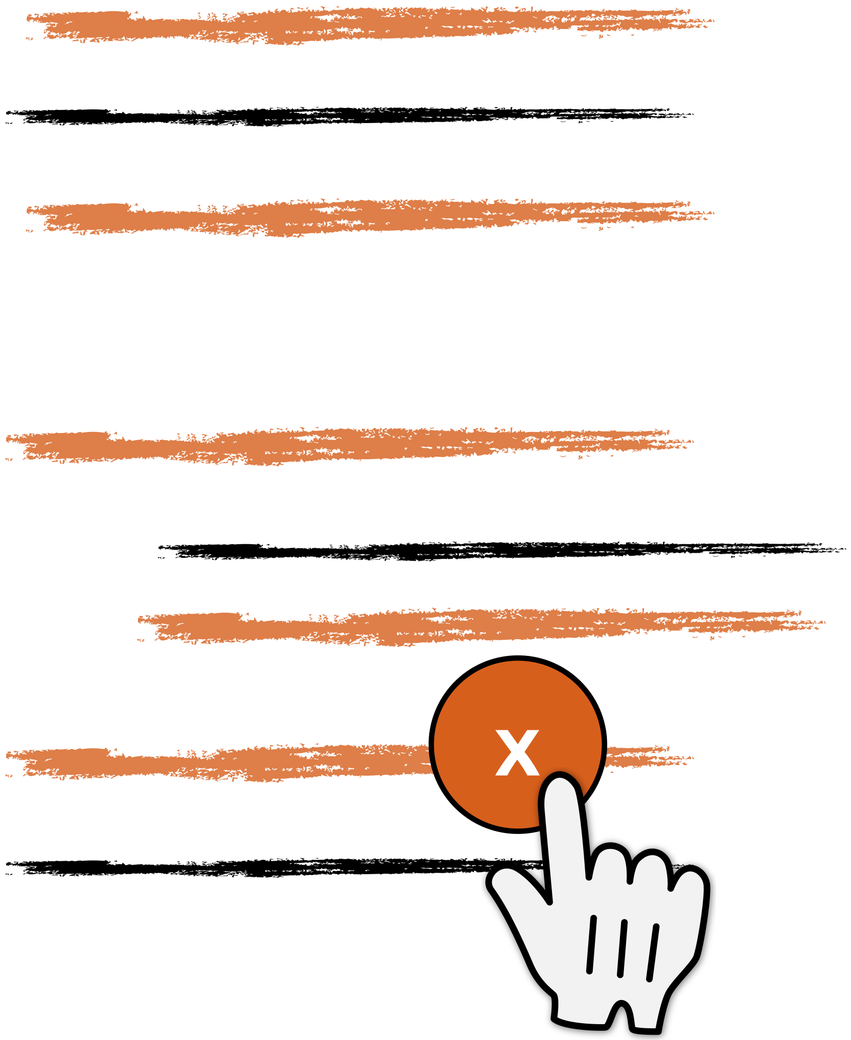
Program **Slicing**



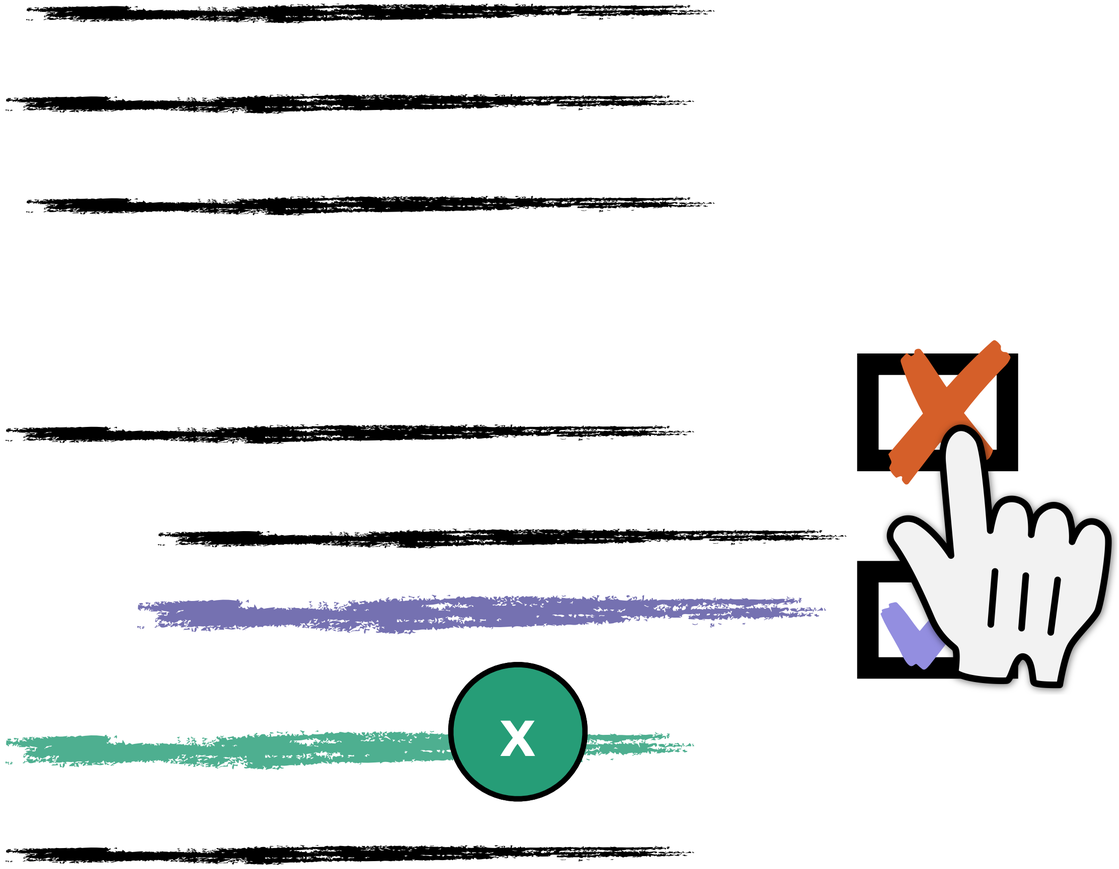
Program **Scooping**



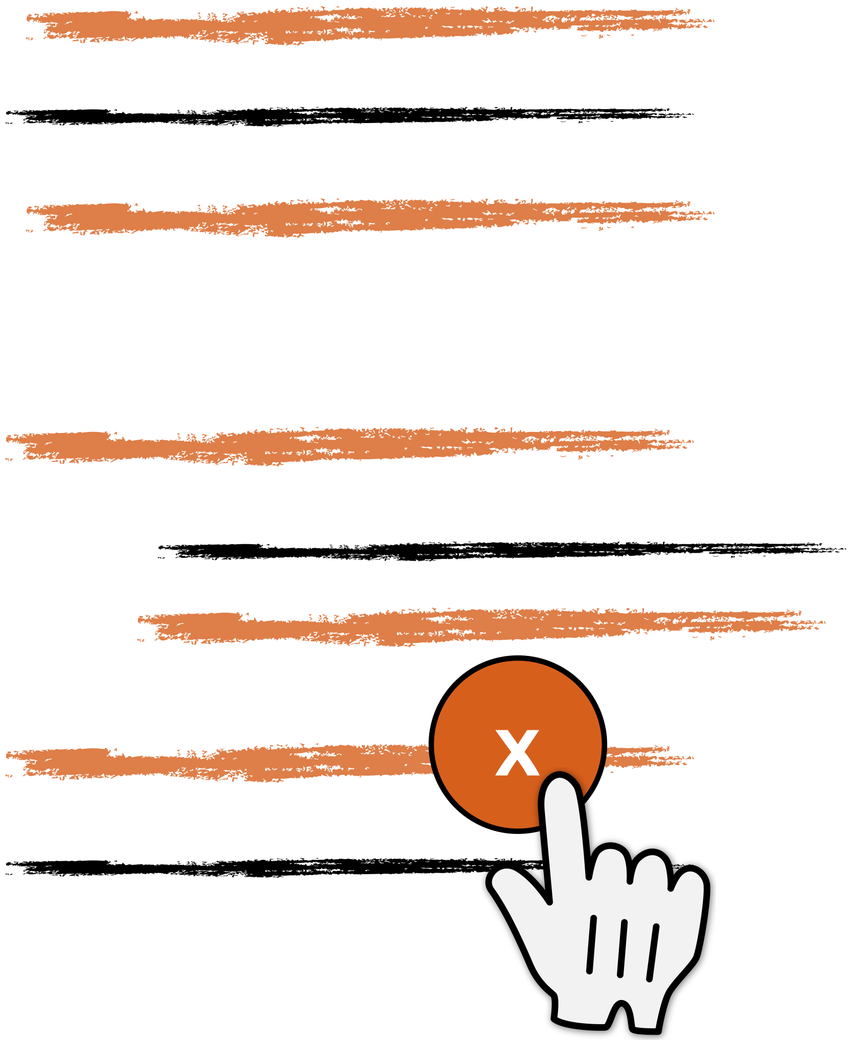
Program **Slicing**



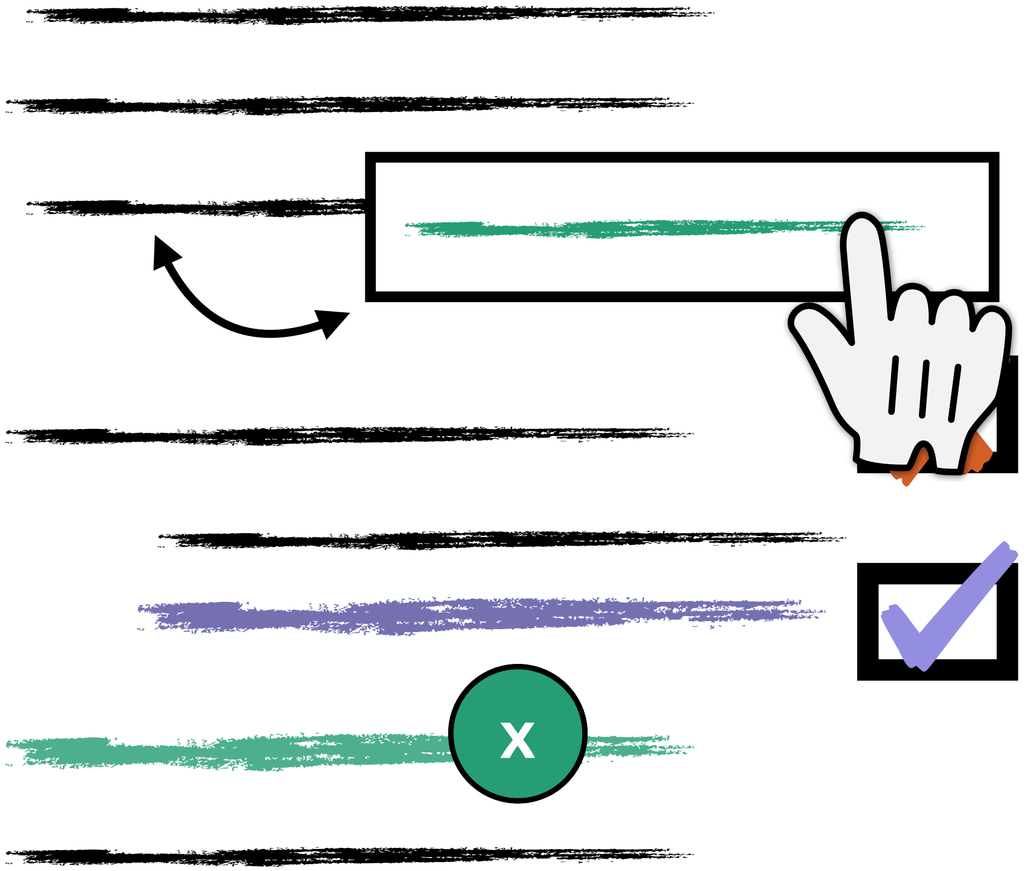
Program **Scooping**



Program **Slicing**

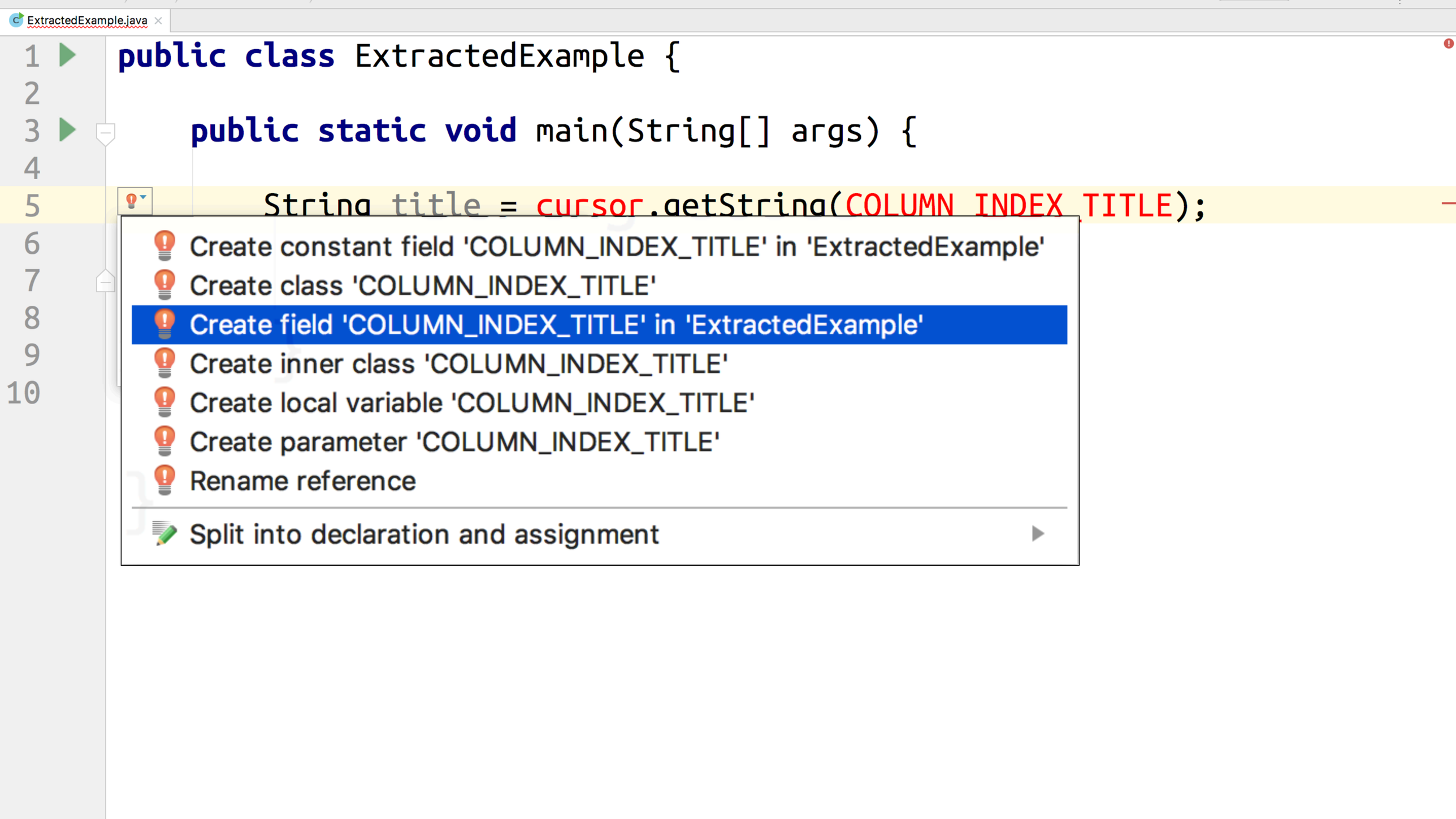


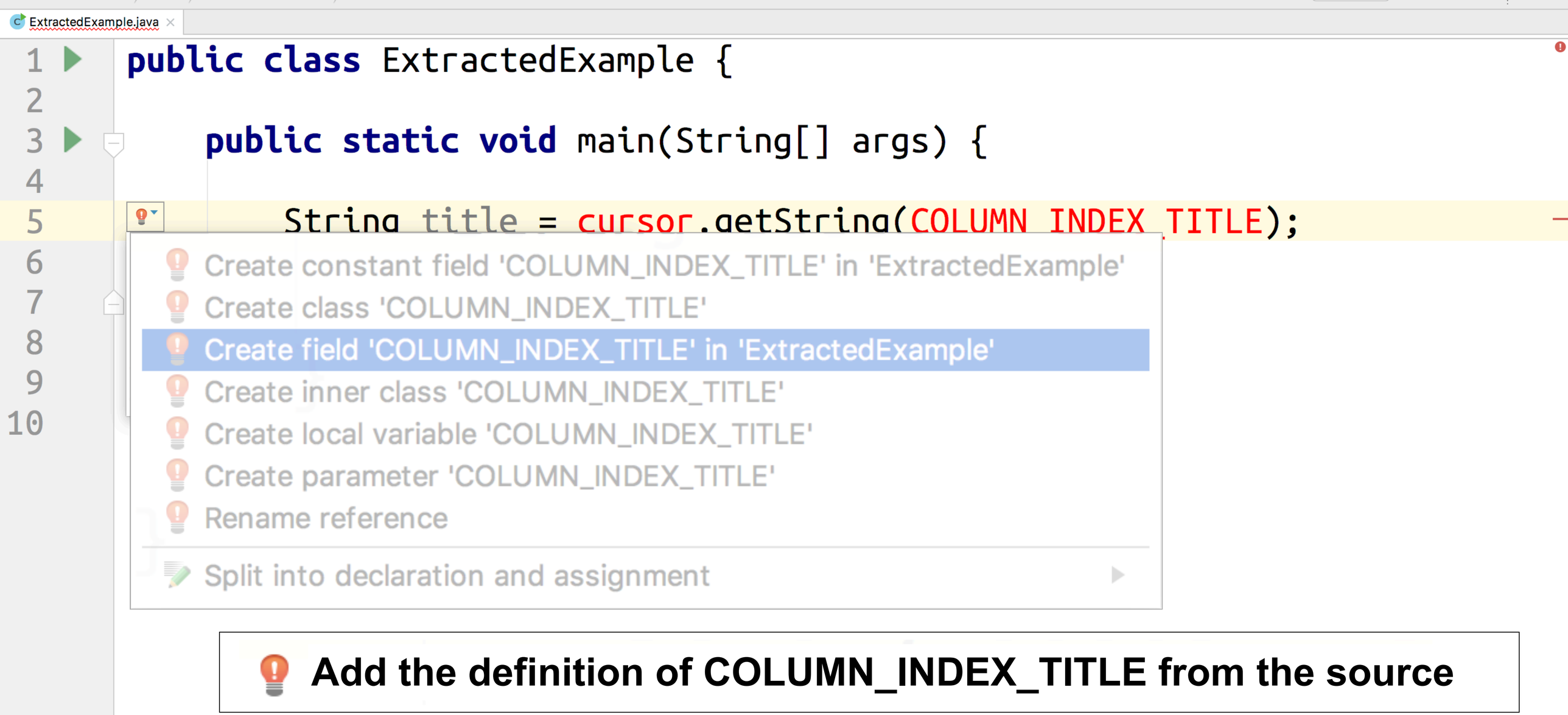
Program **Scooping**



ExtractedExample.java

```
1  public class ExtractedExample {
2
3      public static void main(String[] args) {
4
5          String title = cursor.getString(COLUMN_INDEX_TITLE);
6
7      }
8
9  }
10
```


ExtractedExample.java
public class ExtractedExample {
public static void main(String[] args) {
String title = cursor.getString(COLUMN_INDEX_TITLE);
Create constant field 'COLUMN_INDEX_TITLE' in 'ExtractedExample'
Create class 'COLUMN_INDEX_TITLE'
Create field 'COLUMN_INDEX_TITLE' in 'ExtractedExample'
Create inner class 'COLUMN_INDEX_TITLE'
Create local variable 'COLUMN_INDEX_TITLE'
Create parameter 'COLUMN_INDEX_TITLE'
Rename reference
Split into declaration and assignment



Add the definition of COLUMN_INDEX_TITLE from the source

Sometimes the most useful fixes come from the source program (*Euklas*, Dörner et al. 2014)

```
34              int rowNumber = 0;
35              while (finished == false) {
36
37                  int rowCount = cursor.rowCount();
38
39                  for (int i = 0; i < Math.min(rowCount, maxBooks); ++i) {
40
41                      cursor.fetchone();
42                      int id = cursor.getInt(COLUMN_INDEX_ID);
43                      String title = cursor.getString(COLUMN_INDEX_TITLE);
44                      int year = cursor.getInt(COLUMN_INDEX_YEAR);
45                      int num_pages = cursor.getInt(COLUMN_INDEX_NUM_PAGES);
46                      Book book = new Book(id, title, year, num_pages);
47
48                      if (title != null) {
49                          titles.add(title);
50                      }
51                      if (id != -1) {
52                          boolean bestseller = isBestseller(book.getId());
53                          if (bestseller) {
54                              booklist.hasBestseller = bestseller;
55                          }
56                      }
57
58                      if (DEBUG == true) {
```

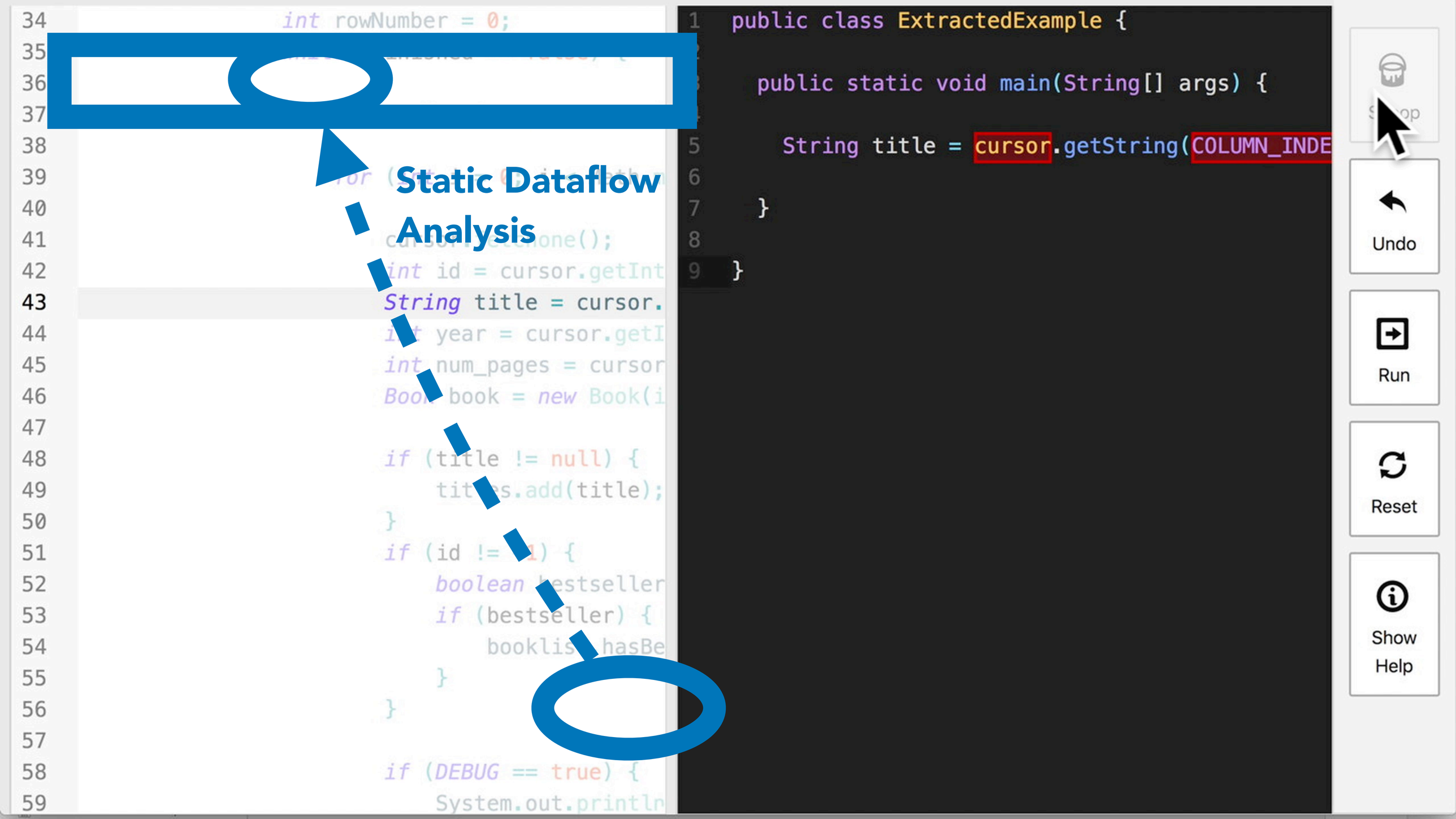
Scoop

Undo

Run

Reset

Show Help


Static Dataflow Analysis
public class ExtractedExample {
public static void main(String[] args) {
String title = cursor.getString(COLUMN_INDE
}
}
Undo
Run
Reset
Show Help

```
21
22        Database database = new Database("lou"
23        Cursor cursor = database.cursor();
24        Booklist booklist = new Booklist();
25        List titles = new ArrayList();
26
27        try {
28
29            cursor.execute(QUERY);
30            boolean finished = false;
31
32            if (cursor.rowCount() > 0) {
33
34                int rowNumber = 0;
35                while (finished == false) {
36
37                    int rowCount = cursor.rowC
38
39                    for (int i = 0; i < Math.m
40
41                        cursor.fetchone();
42                        int id = cursor.getInt
43                        String title = cursor.
44                        int year = cursor.getI
45                        int num_pages = cursor
46                        Book book = new Book(i
```

```
1  public class ExtractedExample {
2
3    public static void main(String[] args) {
4
5      String title = cursor.getString(COLUMN_INDE
6
7    }
8
9  }
```

Add code | Stub out

Line 5

Scoop

Undo

Run

Reset

Show Help

```
21
22          Database database = new Database("lou"
23          Cursor cursor = database.cursor();
24          Booklist booklist = new Booklist();
25          List titles = new ArrayList();
26
27          try {
28
29              cursor.execute(QUERY);
30              boolean finished = false;
31
32              if (cursor.rowCount() > 0) {
33
34                  int rowNumber = 0;
35                  while (finished == false) {
36
37                      int rowCount = cursor.rowC
38
39                      for (int i = 0; i < Math.m
40
41                          cursor.fetchone();
42                          int id = cursor.getInt
43                          String title = cursor.
44                          int year = cursor.getI
45                          int num_pages = cursor
46                          Book book = new Book(i
```

```
1  public class ExtractedExample {
2
3    public static void main(String[] args) {
4
5      Cursor cursor = database.cursor();
6      String title = cursor.getString(COLUMN_IND
7
8    }
9
10 }
```

Include 'try' structure? **Accept** **Reject**

Scoop

Undo

Run

Reset

Show Help

```
21
22          Database database = new Database("lou"
23          Cursor cursor = database.cursor();
24          Booklist booklist = new Booklist();
25          List titles = new ArrayList();
26
27          try {
28
29              cursor.execute(QUERY);
30              boolean finished = false;
31
32              if (cursor.rowCount() > 0) {
33
34                  int rowNumber = 0;
35                  while (finished == false) {
36
37                      int rowCount = cursor.rowC
38
39                      for (int i = 0; i < Math.m
40
41                          cursor.fetchone();
42                          int id = cursor.getInt
43                          String title = cursor.
44                          int year = cursor.getI
45                          int num_pages = cursor
46                          Book book = new Book(i
```

```
1  public class ExtractedExample {
2
3    public static void main(String[] args) {
4
5      Cursor cursor = database.cursor();
6      try {
7        cursor.execute(QUERY);
8        cursor.fetchone();
9        String title = cursor.getString(COLUMN_I
10     } catch (Connecti                    tion) {
11     }
12
13   }
14
15 }
```

Do you want any of these uses of `cursor`?

- Line 32
- Line 37
- Line 42
- No, Skip

Scoop

Undo

Run

Reset

Show Help

```
15        String QUERY = "SELECT id, title, year
16        int COLUMN_INDEX_ID = 0;
17        int COLUMN_INDEX_TITLE = 1;
18        int COLUMN_INDEX_YEAR = 2;
19        int COLUMN_INDEX_NUM_PAGES = 3;
20        boolean DEBUG = true;
21
22        Database database = new Database("lou"
23        Cursor cursor = database.cursor();
24        Booklist booklist = new Booklist();
25        List titles = new ArrayList();
26
27        try {
28
29            cursor.execute(QUERY);
30            boolean finished = false;
31
32            if (cursor.rowCount() > 0) {
33
34                int rowNumber = 0;
35                while (finished == false) {
36
37                    int rowCount = cursor.rowC
38
39                    for (int i = 0; i < Math.m
40
```

```
1   ExtractedExample {
2
3  tic void main(String[] args) {
4
5  ursor = database.cursor();
6
7  .execute(QUERY);
8  .fetchone();
9   title = cursor.getString(1);
10 (ConnectionException excep
11
12
13
14
15
```

Add code | Set value

Scoop

Undo

Run

Reset

Show Help

```
15          String QUERY = "SELECT id, title, year
16          int COLUMN_INDEX_ID = 0;
17          int COLUMN_INDEX_TITLE = 1;
18          int COLUMN_INDEX_YEAR = 2;
19          int COLUMN_INDEX_NUM_PAGES = 3;
20          boolean DEBUG = true;
21
22          Database database = new Database("lou"
23          Cursor cursor = database.cursor();
24          Booklist booklist = new Booklist();
25          List titles = new ArrayList();
26
27          try {
28
29              cursor.execute(QUERY);
30              boolean finished = false;
31
32              if (cursor.rowCount() > 0) {
33
34                  int rowNumber = 0;
35                  while (finished == false) {
36
37                      int rowCount = cursor.rowC
38
39                      for (int i = 0; i < Math.m
40
```

```
1  public class ExtractedExample {
2
3    public static void main(String[] args) {
4
5      Cursor cursor = database.cursor();
6      try {
7        cursor.execute                    year,
8        cursor.fetchon
9        String title =                  tString(1);
10     } catch (Connectio                ception exception) {
11     }
12
13   }
14
15 }
```

Add code | Stub out

line 22

Scoop

Undo

Run

Reset

Show Help

Scooping Summary

First selections

Scooping Summary

Scooping Summary

Optional control

Code fixups

First selections

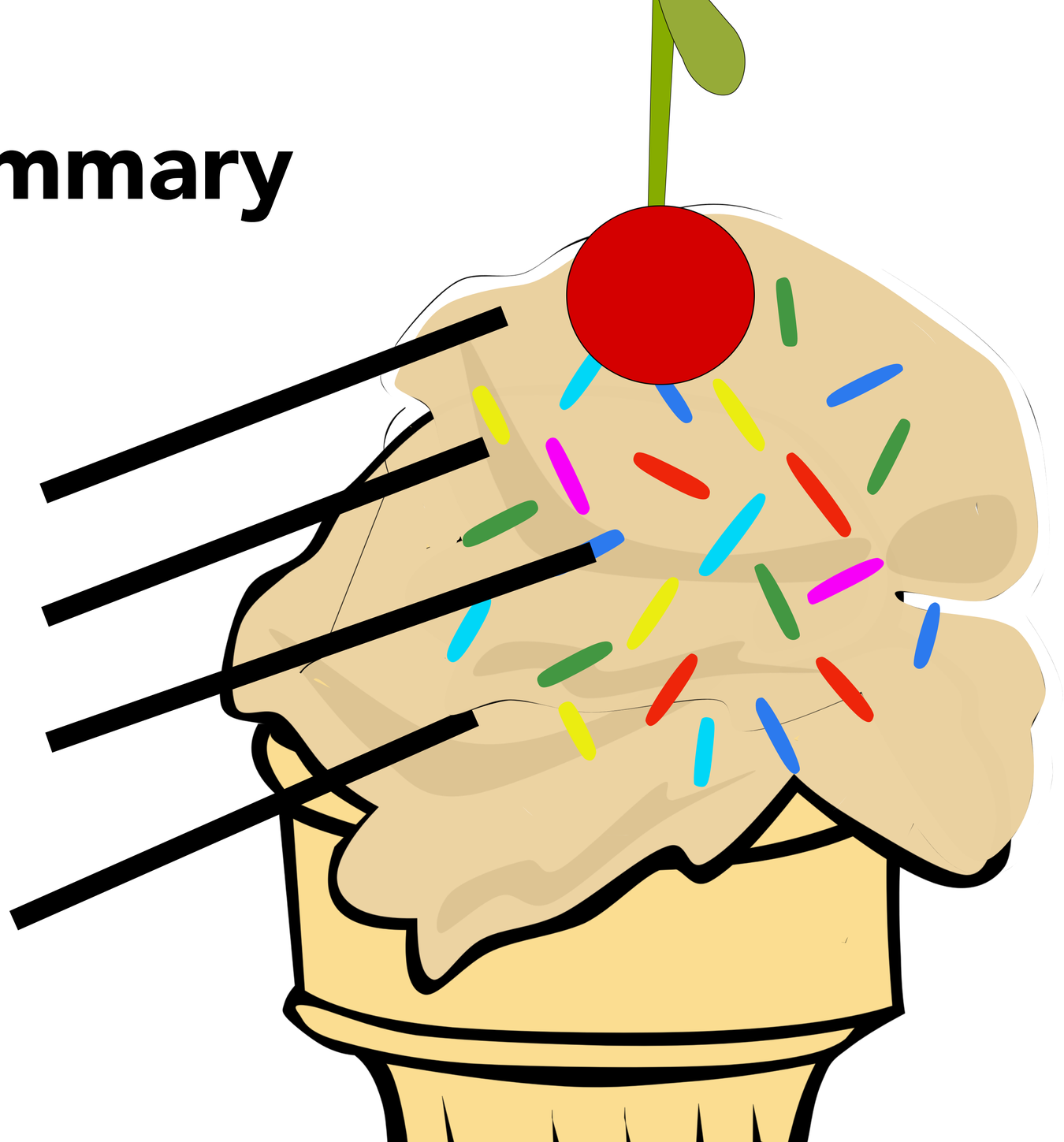
Scooping Summary

Variable substitutions

Optional control

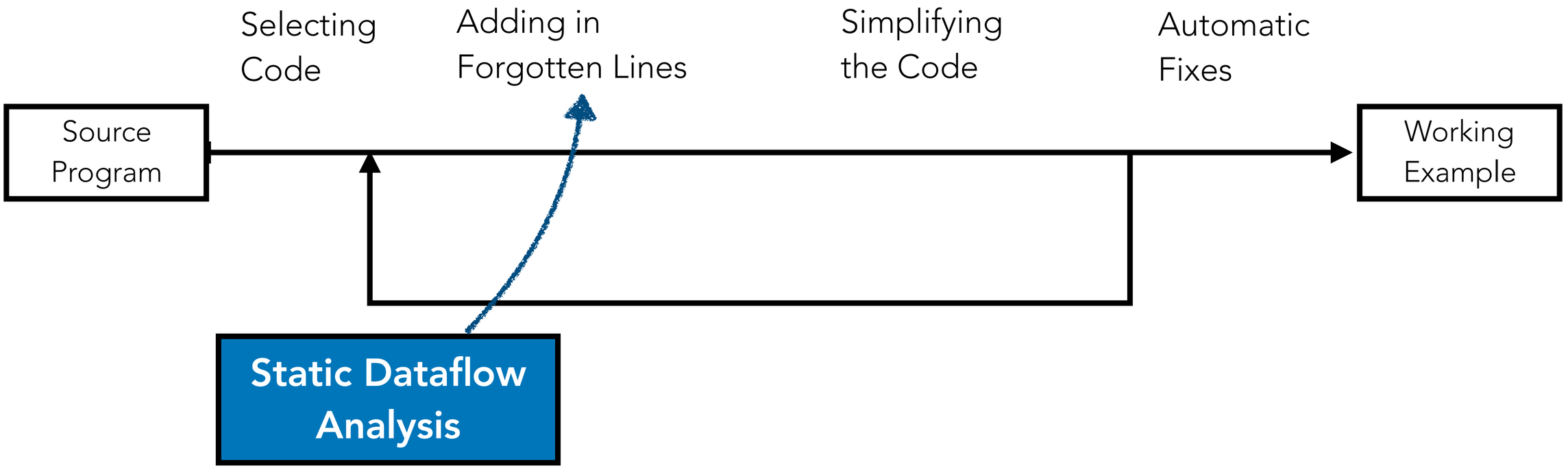
Code fixups

First selections

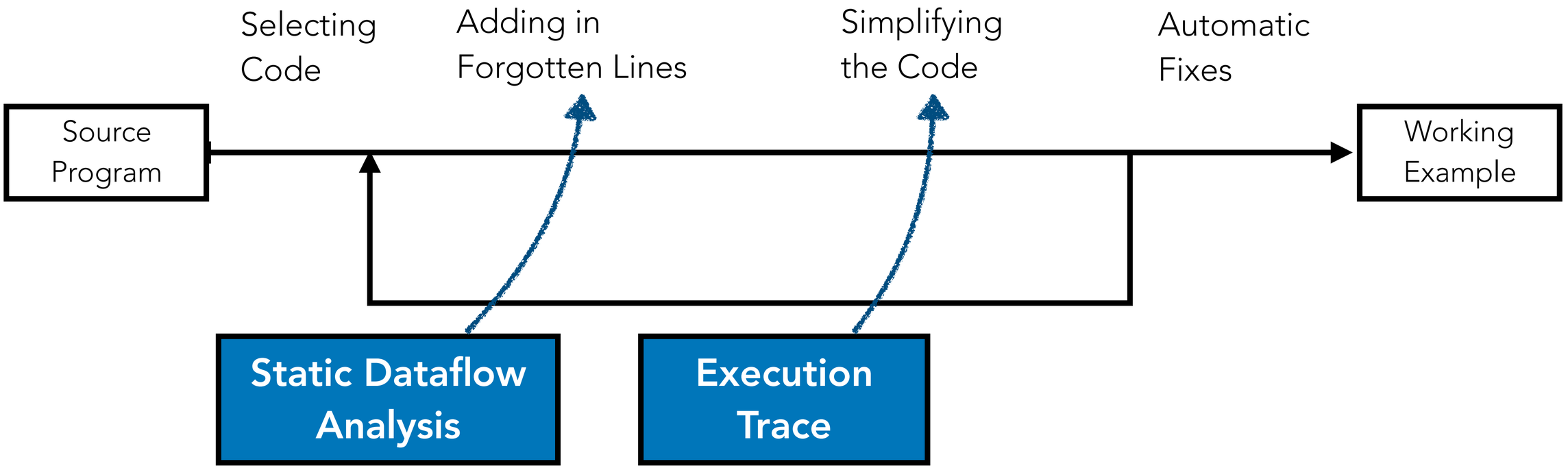


Combining Code Analyses into an Integrated Tool

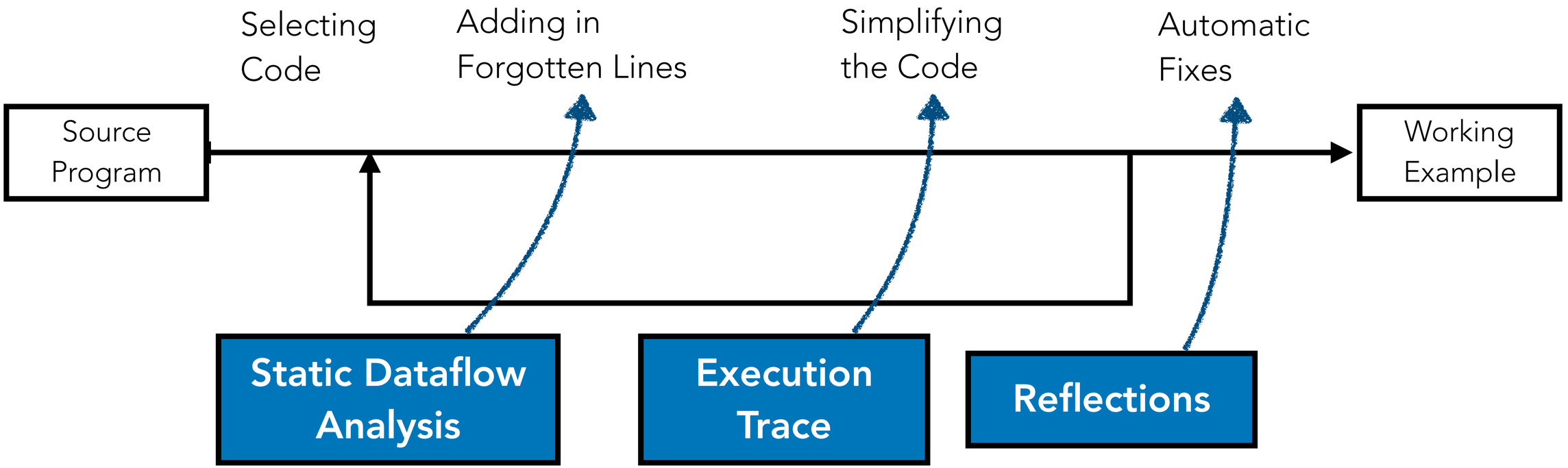
Combining Code Analyses into an Integrated Tool



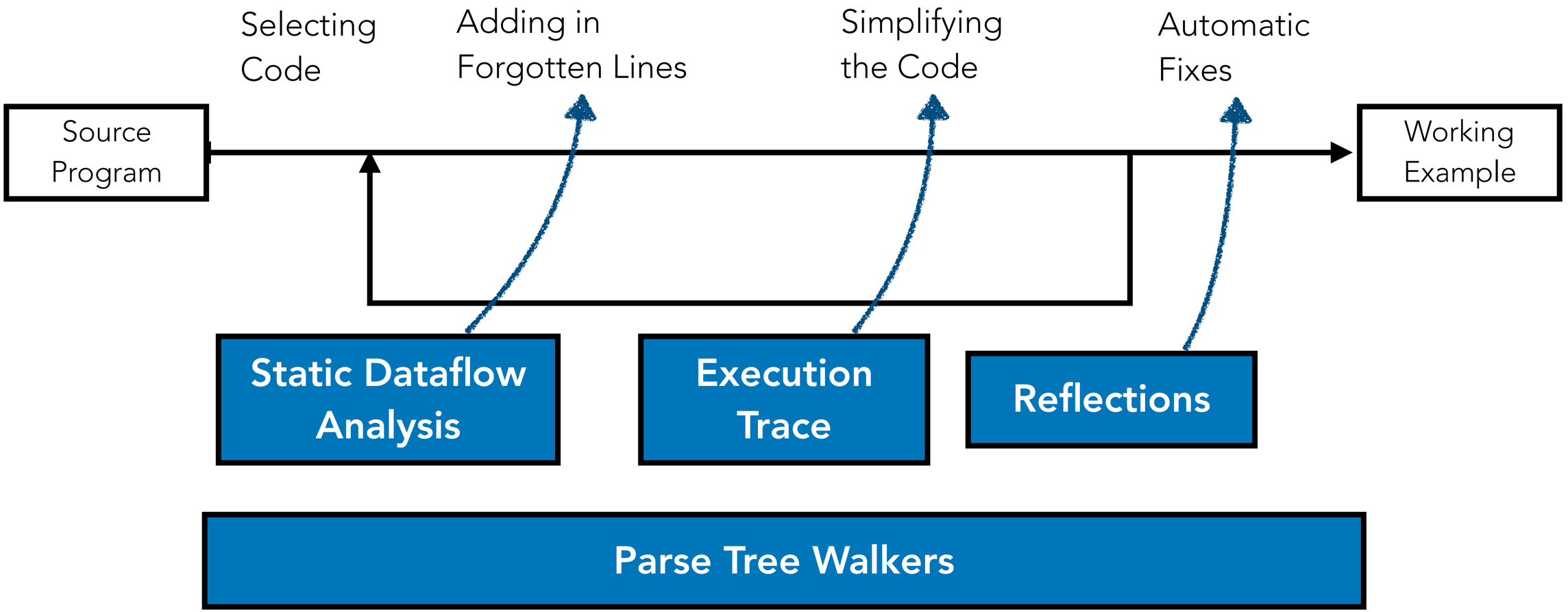
Combining Code Analyses into an Integrated Tool



Combining Code Analyses into an Integrated Tool



Combining Code Analyses into an Integrated Tool



Evaluating CodeScoop

Q1. How does CodeScoop compare to a standard text editor for extracting example code?

Q2. Should CodeScoop be making decisions about fixing code automatically?

...

A Pilot Study about Example Code Extraction

Participants: *N = 19* undergraduate student programmers

Main Task: Create examples from existing code

Measurements: Usability of CodeScoop vs. baseline text editor, Preference for their scoop vs. an automatic slice, time to extract an example, ...

Qualitative Feedback: Survey and Interview

Was CodeScoop, vs. the baseline...?

Faster to use?	**Yes**.	5.8 min vs 9.6 min	$p < .001$
Easier to use?	**Yes**.	$\Delta = 3$ (7-pt scale)	$< .01$
More **enjoyable**?	**Yes**.	$\Delta = 3$ "	$< .01$
Producing more **satisfying** examples?	**Yes**.	$\Delta = 2$ "	$< .01$

"[CodeScoop's features] made creating an example a lot easier because I just had to look at the relevant code and see if I needed it or not instead of having to manually add them in."

CodeScoop provided a median of...

12 automatic corrections

5 suggestions of optional code

2 suggestions of error fixes

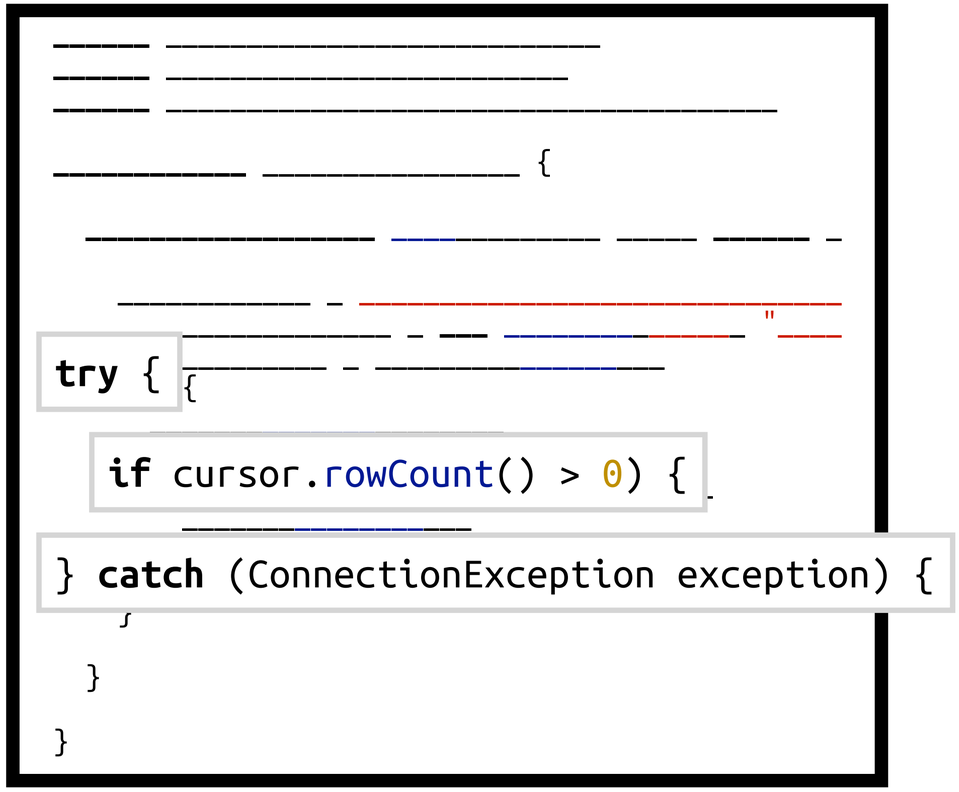
CodeScoop Enabled Meaningful Simplifications

For Task 3, participants made an important simplification:

Slice (101 lines)

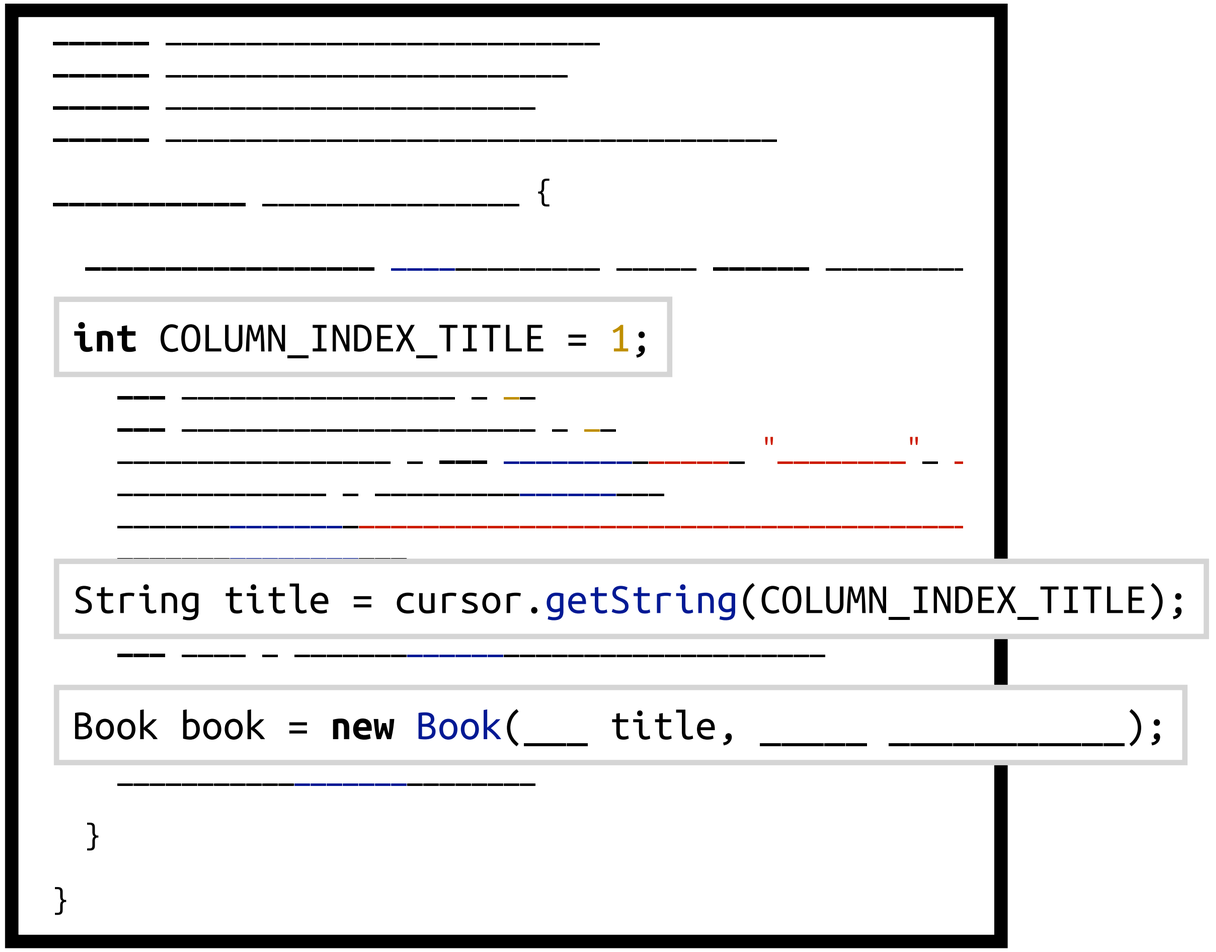
Scoop (*median* = 36 lines)

No Consistent Heuristics for Automating Extraction



```
try {
    if cursor.rowCount() > 0) {
} catch (ConnectionException exception) {
  }
}
```

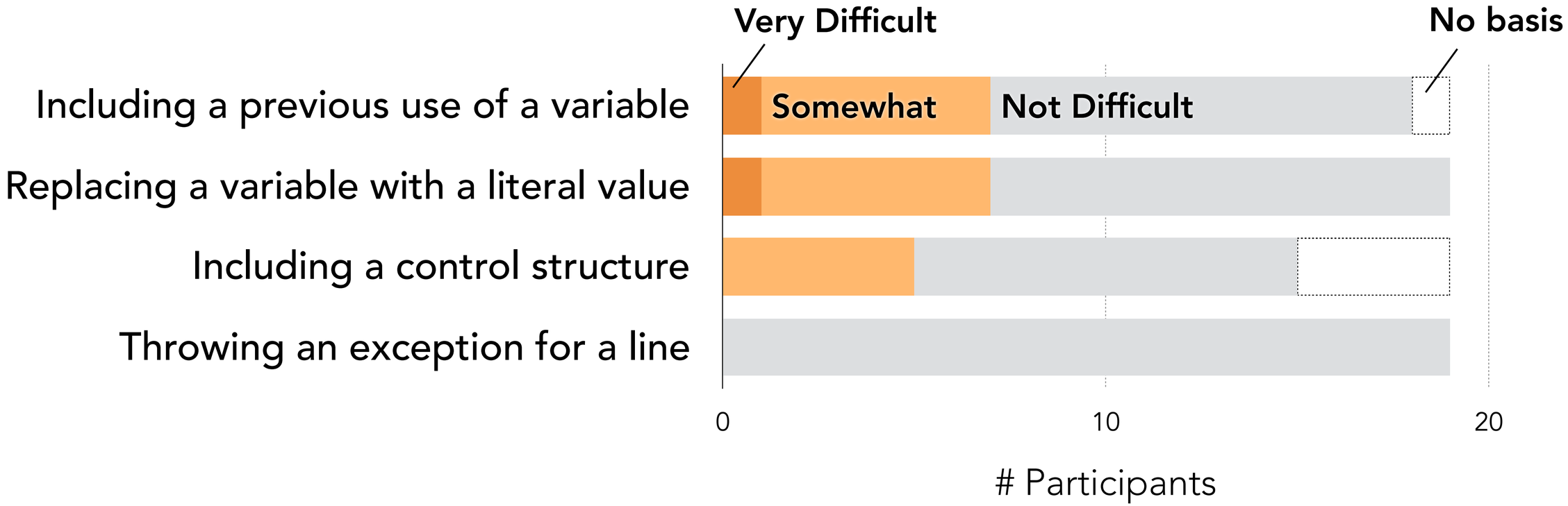
Choice: **Error checking** through exceptions and postconditions.



```
int COLUMN_INDEX_TITLE = 1;
String title = cursor.getString(COLUMN_INDEX_TITLE);
Book book = new Book(___ title, _____ ___________);
 }
}
```

Choice: Column **variable names**, saving results to **Book** object.

Not All Authoring Decisions Were Easy

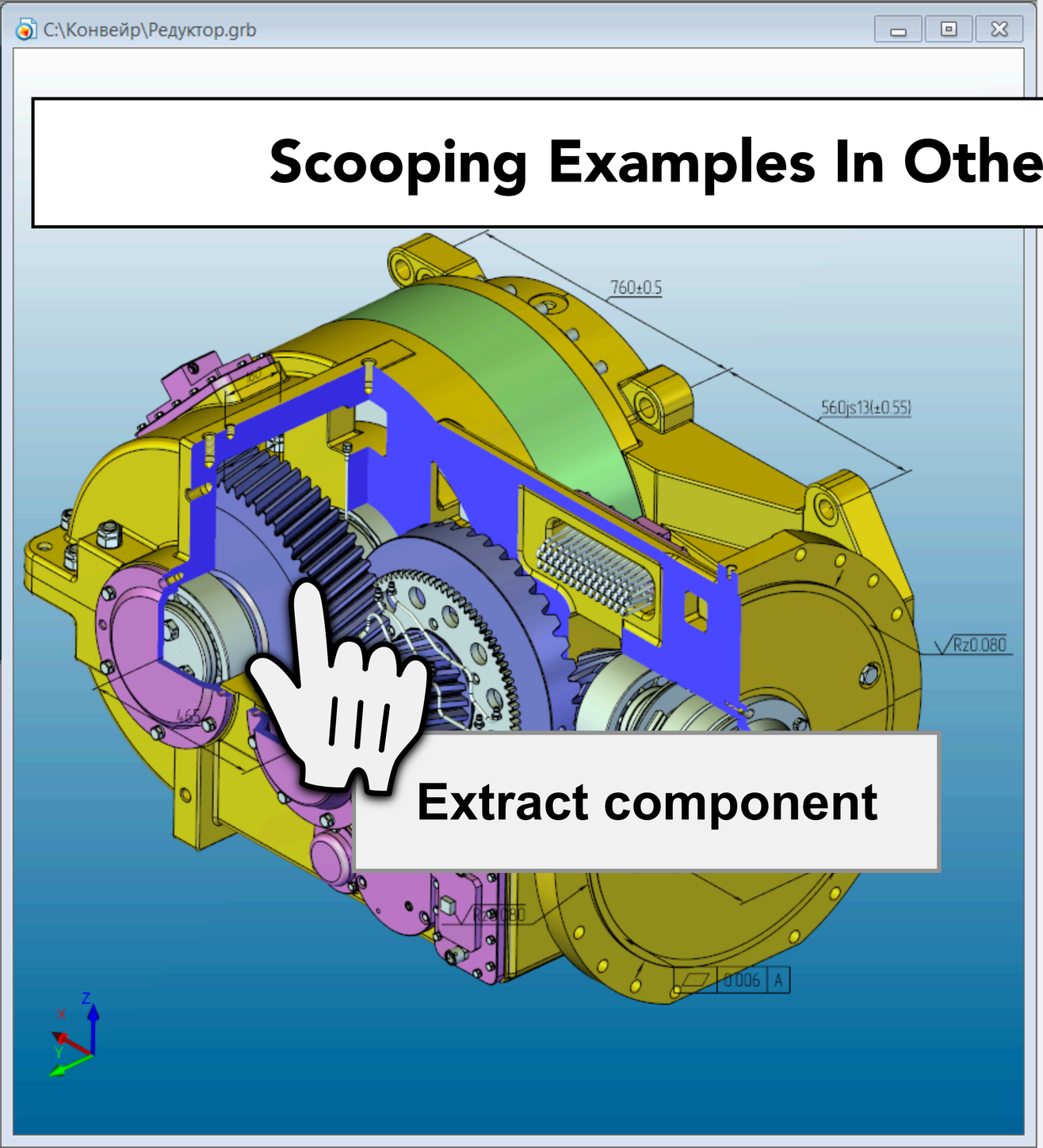


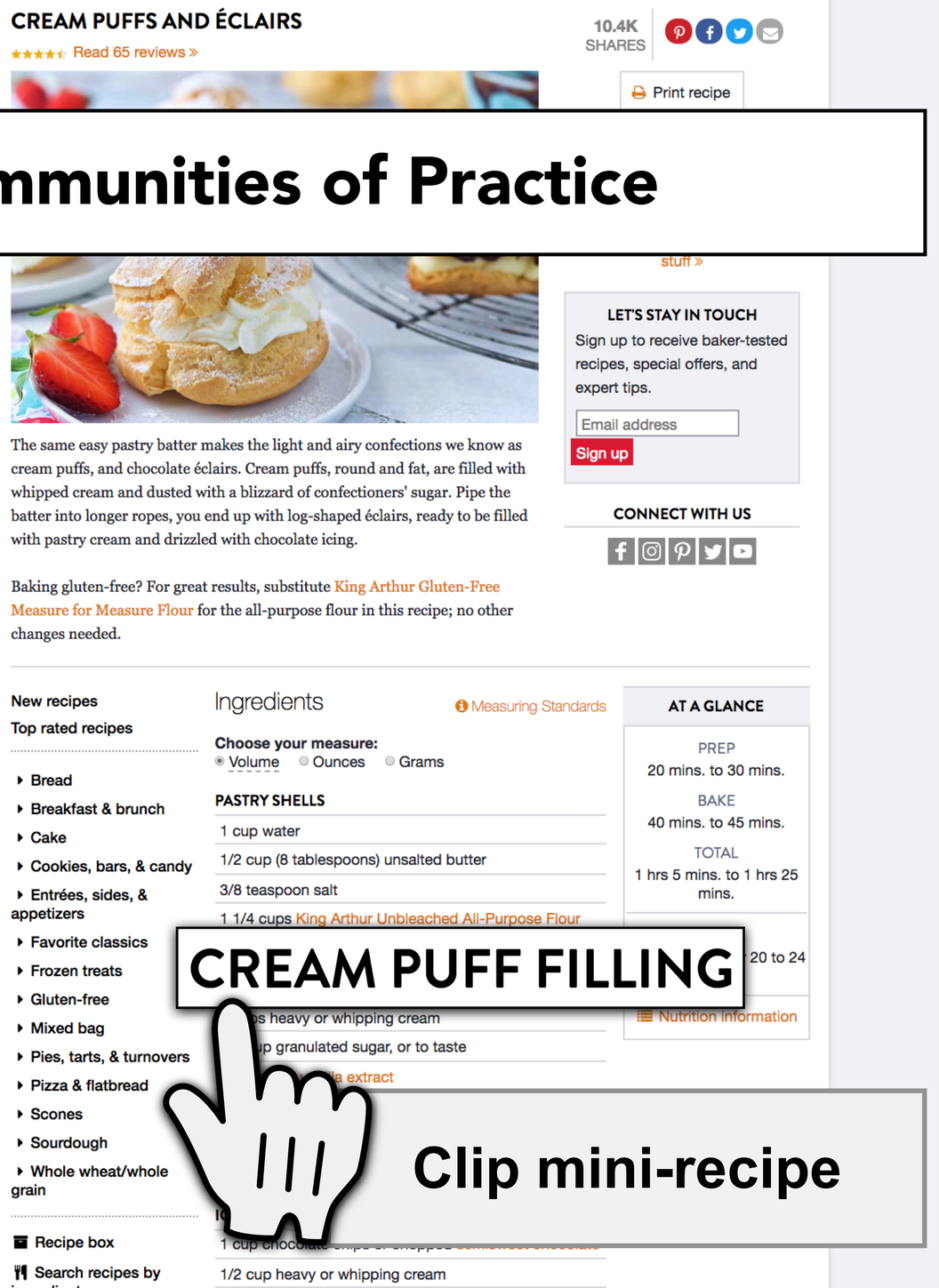
Takeaways from Study

Q1. Scooping with our prototype was easier than using the baseline text editor.

Q2. Scooping has advantages over slicing: authoring choices, and sometimes conciseness.

Scooping Examples In Other Communities of Practice





Video link: https://youtu.be/sIpSS-F1Ltg

Demo, paper, and auxiliary material @ codescoop.berkeley.edu

Demo, paper, and auxiliary material @ **codescoop.berkeley.edu**

Was CodeScoop, compared to the baseline editor...?		
Faster to use?	**Yes.** (5.8 vs. 9.5 mins.)	$p < .001$
More enjoyable to use?	**Yes.** ($\Delta = 3$, on 7-point scale)	$p < .01$
Easier to use?	**Yes.** ($\Delta = 3$, on 7-point scale)	$p < .01$
Producing more satisfying examples?	**Yes.** ($\Delta = 2$, on 7-point scale)	$p < .01$

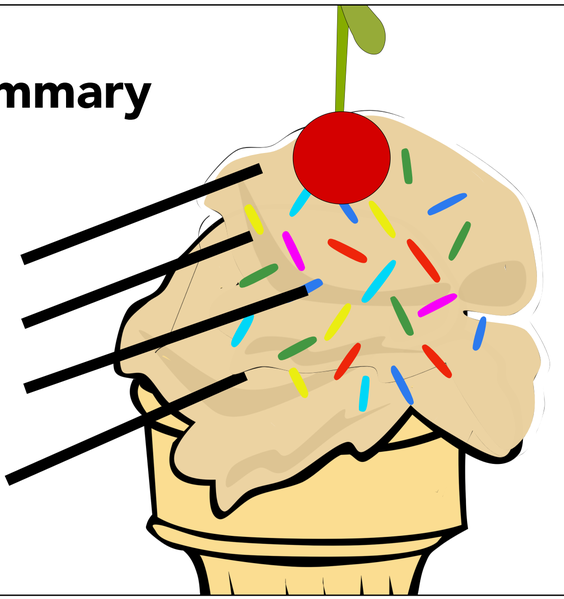
Scooping Summary

Variable substitutions

Optional control

Code fixups

First selections



Backup slides

Example Authoring Process and Choices

Authors made examples by...	Tools should help authors...
Copying the original code and pasting into example editor	• Create examples from text selections • Add lines from original code at any time
Replacing variables with meaningful literal values	• Review and insert literal values that preserve program behavior
Tweaking comments and code format for readability	• Directly edit code to add comments, group lines, and add `print` statements

We describe more choices in the auxiliary material!

```
private class Database {

    public AnonymousClass1 cursor() {
        return new AnonymousClass1();
    }

}

private class AnonymousClass1 extends org.acme.database.Cursor {

    private int rowCountCallCount = 0;
    private int endCallCount = 0;
    private int getStringCallCount = 0;
    private int fetchoneCallCount = 0;
    private int getIntCallCount = 0;

    public int rowCount() {
        rowCountCallCount += 1;
        if (rowCountCallCount == 1) {
            return 2;
        } else if (rowCountCallCount == 2) {
            return 2;
        } else {
            return 1;
        }
    }

    public boolean end() {
        endCallCount += 1;
```

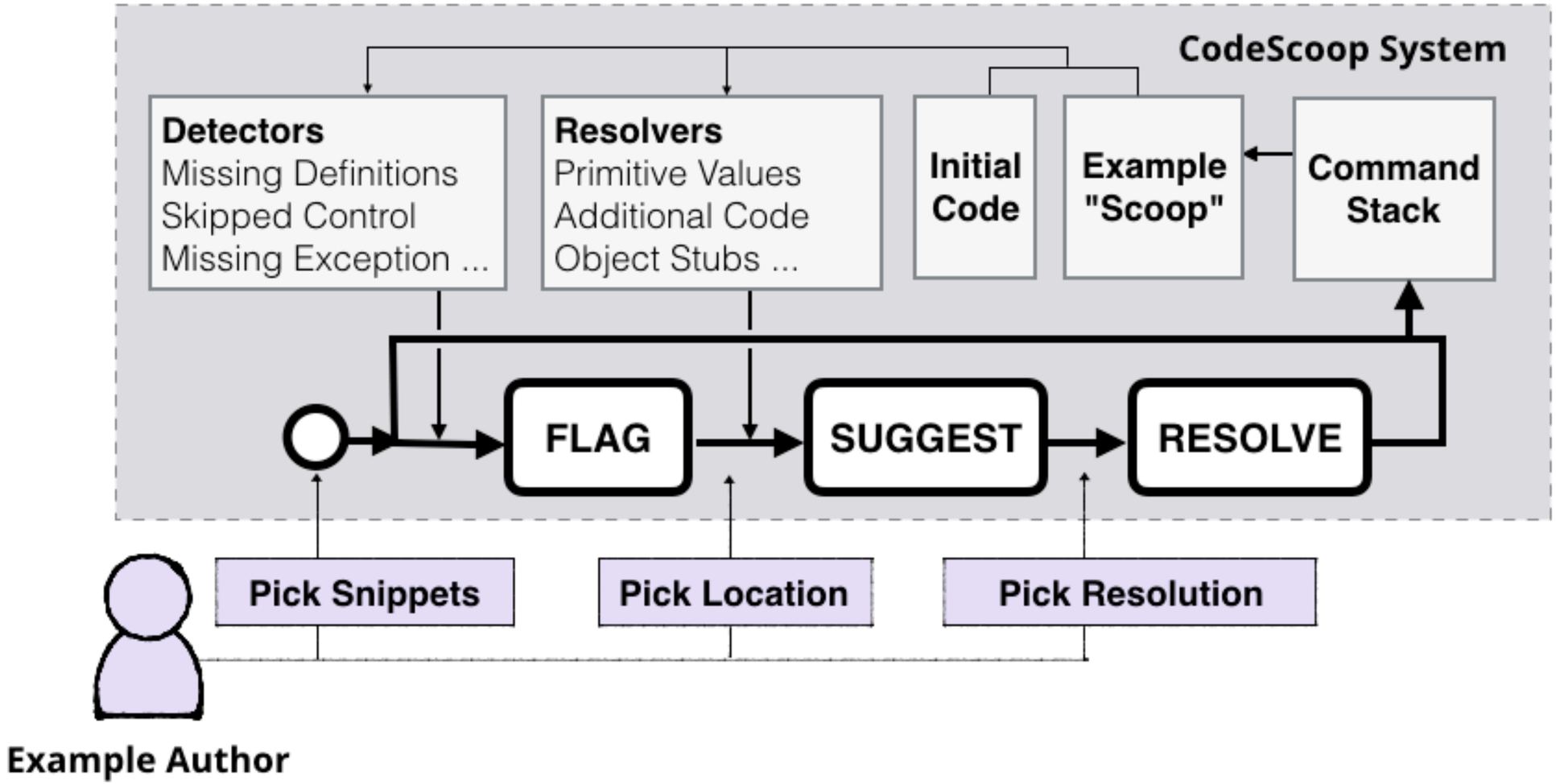
```
public class ExtractedExample {

  public static void main(String[] args) {

    Cursor cursor = (new Database()).cursor();
    try {
      cursor.execute
      cursor.fetchone();
      String title = cursor.get
    } catch (ConnectionException exception) {
    }

  }

}
```

Add code | Stub out

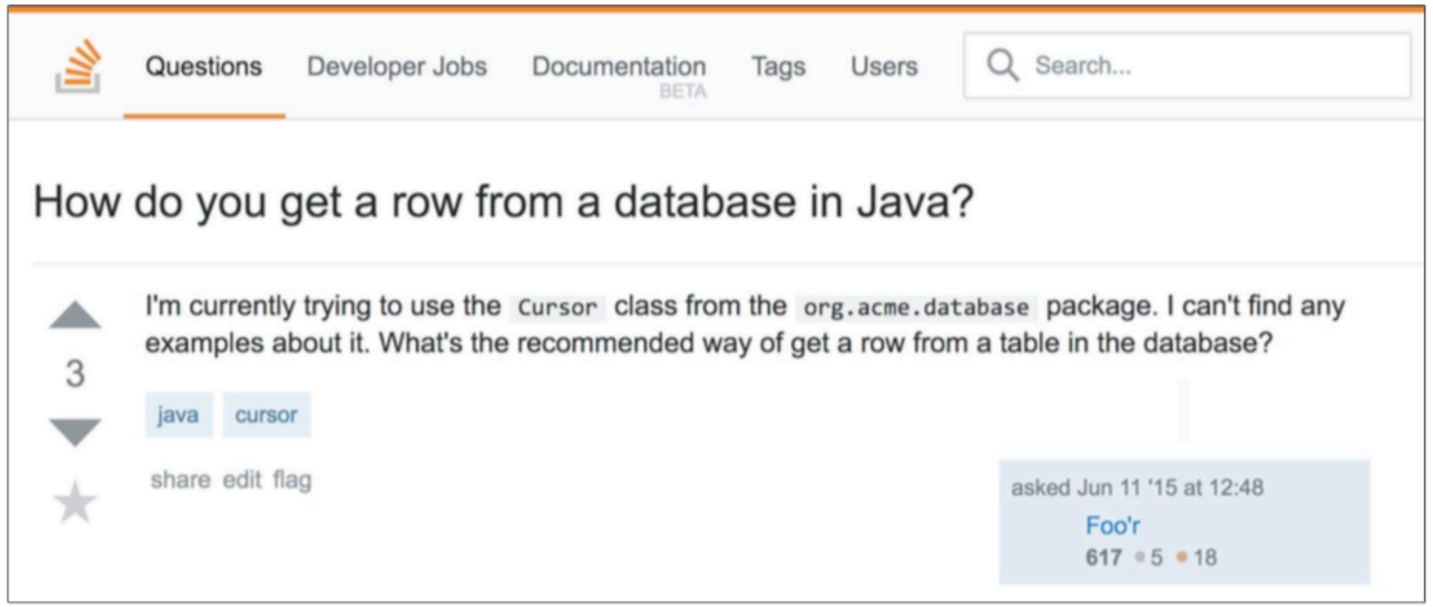
← Preview Stub 1

Generating object stubs.

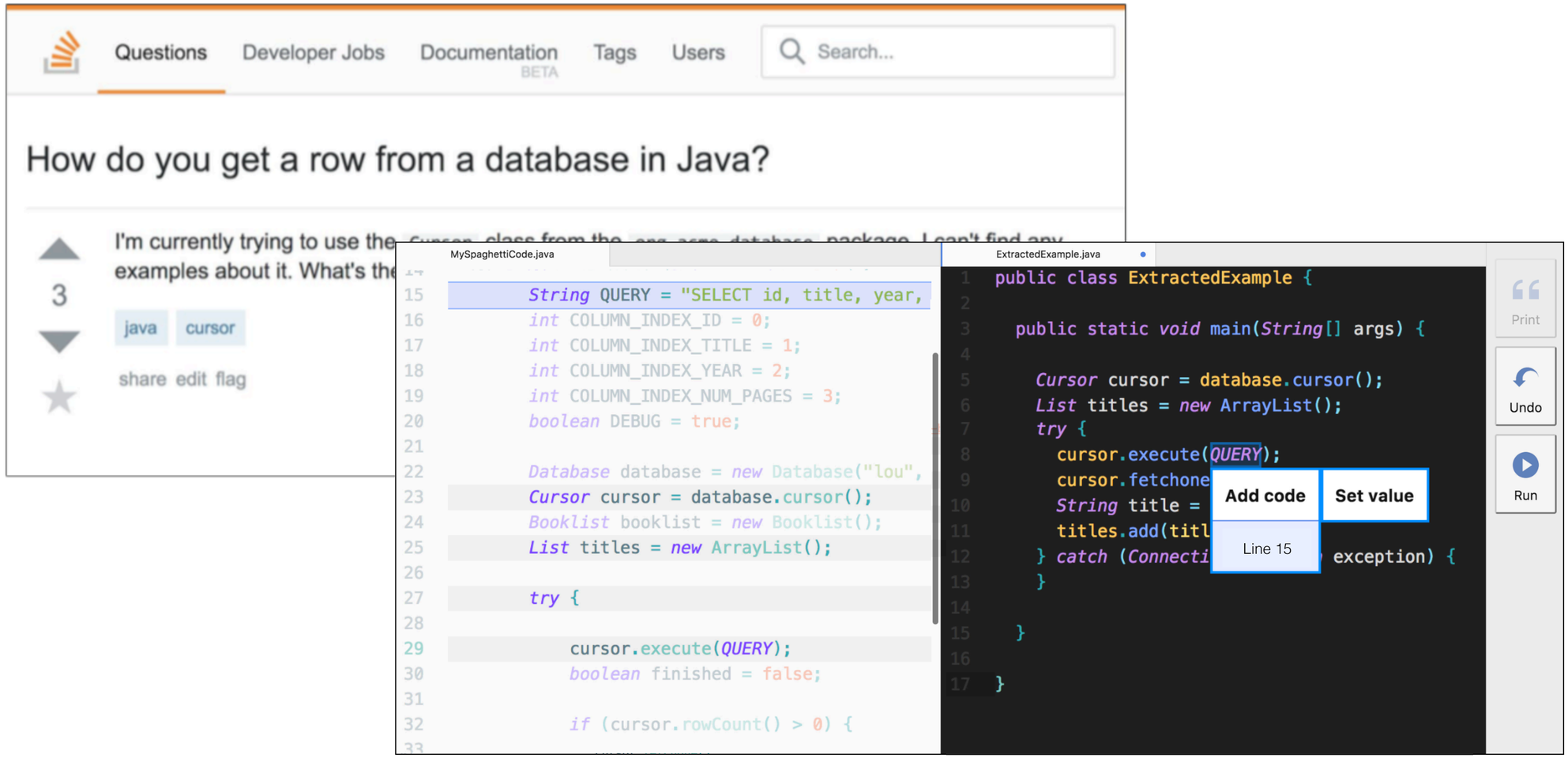
Detailed CodeScoop System View



Pilot Study Task Design



Pilot Study Task Design



Source Program

Their Example

High-Level Utility of CodeScoop Tool

18 / 19 participants would prefer to use CodeScoop instead of a text editor in the future.

16 / 19 participants successfully extracted an example with CodeScoop, vs. 11 / 19 with the text editor.

Comparing Scoop Length to Slice Length

Task	Scoop Length	Slice Length
1	22.5	22
2	21	37
3	36	101

Quick Fixes Are Useful!

"[CodeScoop] did a lot of the grueling work for me, such as importing any libraries or packages that I needed to work with."

"[CodeScoop] saved me the trouble of having to go through and find things like undeclared variables, missing import statements, and unchecked exceptions, which prevented my [...] code from compiling [in the baseline text editor]."

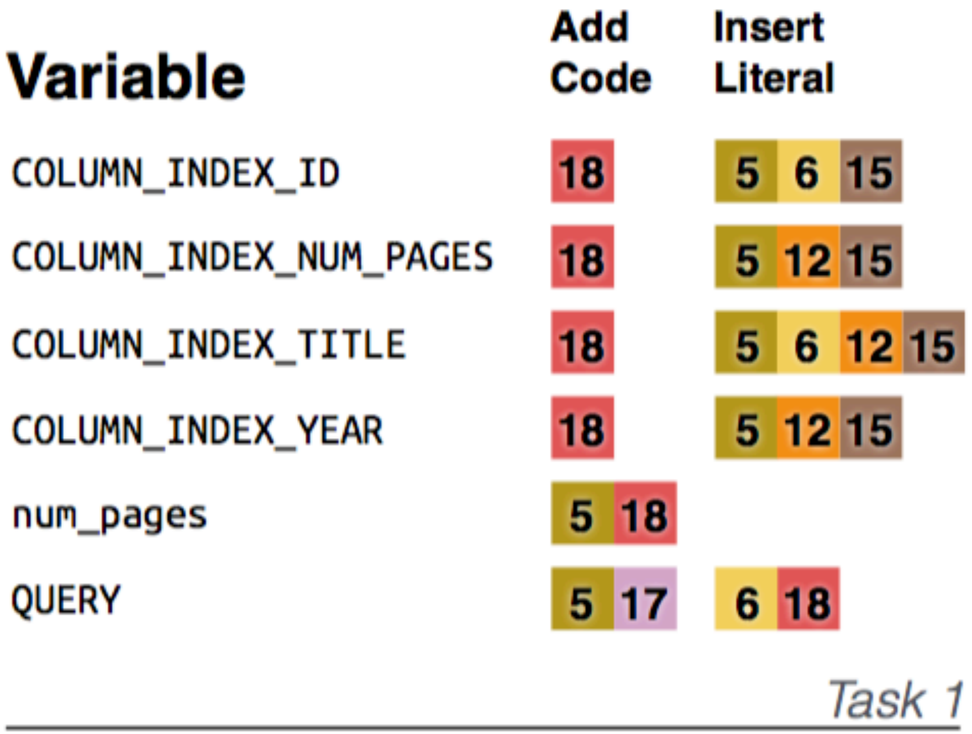
Fixes Anchored In the Source Program Are Useful Too

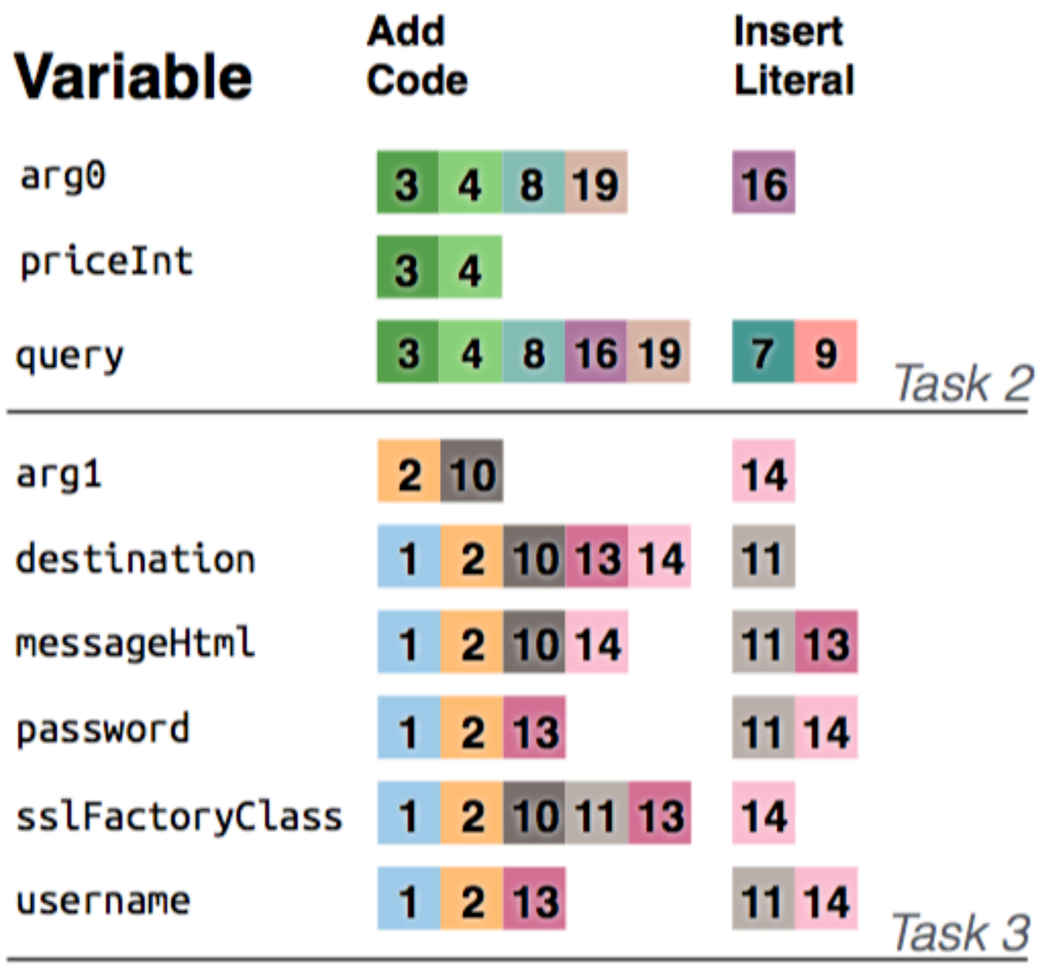
"For me, the features of highest importance were making sure that my code would compile and run with no important left-out lines."

"[CodeScoop] fills in a lot of things that people usually don't really think about (exceptions, variables/constants) and saves a lot of time spent just searching and copy/pasting."

"Often the hardest part about writing code is finding variables and the relationships they have with the other parts of the code in the sea of text that is a program."

Divergence and Convergence on a Fine-Grained Authoring Choice: How to Fix Missing Variables





Annotated Examples from CodeScoop Study

For each example that a participant produced with CodeScoop, we display the example with annotations of how it was constructed.

The marks in the left gutter indicate author interactions. There are three types of interactions:

- An author's first code selections when starting CodeScoop.
- Additional lines added to the example, without any prompting from CodeScoop.
- Lines an author included in response to a suggestion from CodeScoop.

The color of a line indicates the source of the line:

- *Boilerplate*: this code shows up in every example so the example can compile.
- *Automatic*: CodeScoop inferred that this line was required, and added it automatically.
- *Prompted*: The participant included these lines when reviewing a prompt from CodeScoop .
- *Manual*: The participant included these lines without any prompting.

Task 1, Participant 18

```
import org.acme.database.Database;                                          automatic import
import org.acme.database.Cursor;                                            automatic import
import org.acme.database.Book;                                              automatic import
import org.acme.database.ConnectionException;                               automatic import

public class ExtractedExample {                                             boilerplate

  public static void main(String[] args) throws ConnectionException {       boilerplate + prompted throws

    int COLUMN_INDEX_ID = 0;                                                prompted selection (def)
    int COLUMN_INDEX_TITLE = 1;                                             prompted selection (def)
    int COLUMN_INDEX_YEAR = 2;                                              prompted selection (def)
    int COLUMN_INDEX_NUM_PAGES = 3;                                         prompted selection (def)
    Database database = new Database("lou", "PA$$W0RD", "https://acme-books.com/db");   initial selection
    Cursor cursor = database.cursor();                                      selection
    cursor.execute("SELECT id, title, year, num_pages FROM table WHERE title LIKE '%romance%'");   prompted selection (use)
    cursor.fetchone();                                                      [illegible]
    int id = cursor.getInt(COLUMN_INDEX_ID);                                prompted selection (use)
    String title = cursor.getString(COLUMN_INDEX_TITLE);                    prompted selection (use)
    int year = cursor.getInt(COLUMN_INDEX_YEAR);                            prompted selection (use)
    int num_pages = cursor.getInt(COLUMN_INDEX_NUM_PAGES);                  prompted selection (def)
    Book book = new Book(id, title, year, num_pages);                       selection

    System.out.println(title);                                              add print statement

  }                                                                         boilerplate

}                                                                           boilerplate
```

The labels in the right gutter are fine-grained descriptions each line's source. There are three variants of *prompted selections*:

- def: adding code to define a variable
- use: adding a previous use of a variable
- control: adding a control structure that surrounds a statement
- throws: throwing an exception from the main method signature

A line is generated with the "add print statement" label when an author clicks the "Print" with a variable name selected.

Where an author replaced an undefined variable with a literal, the literal is **highlighted in green, underlined, and bolded**.