

Elicast: Embedding Interactive Exercises in Instructional Programming Screencasts

Jungkook Park KAIST

Yeong Hoon Park University of Minnesota

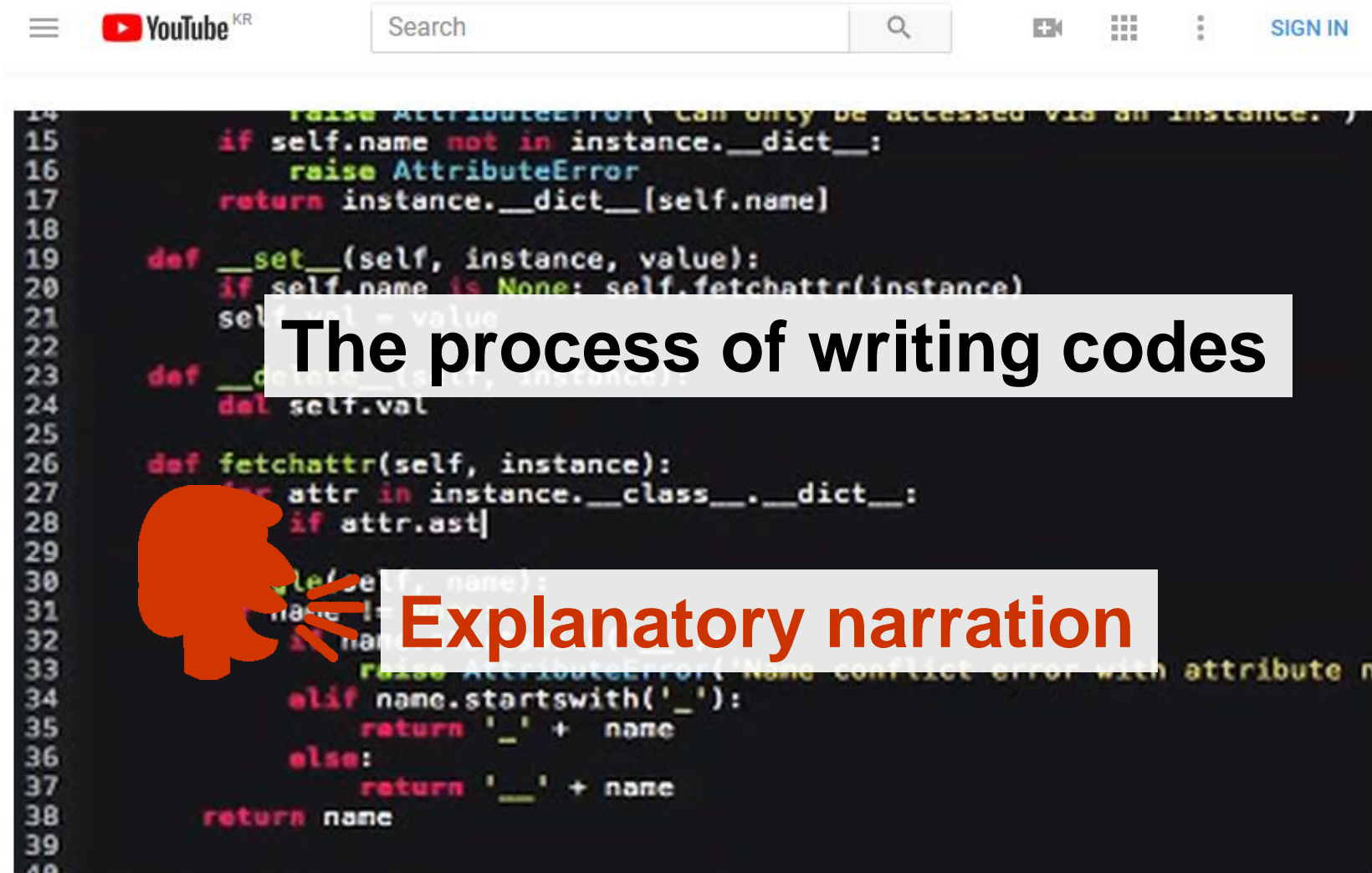
Jinhan Kim KAIST

Jeongmin Cha KAIST

Suin Kim KAIST

Alice Oh KAIST

Instructional Programming Screencast



The image shows a YouTube video player interface. At the top, there is a search bar with the text "Search" and a magnifying glass icon. To the right of the search bar are icons for a camera, a grid, and a vertical ellipsis, followed by a "SIGN IN" button. The main content of the video is a Python code snippet for implementing descriptors. The code is displayed on a dark background with syntax highlighting. Two white text boxes with black text are overlaid on the code. The first box, containing the text "The process of writing codes", is positioned over lines 20-22 of the code. The second box, containing the text "Explanatory narration", is positioned over lines 30-32 of the code. An orange silhouette of a person's head and shoulders is positioned to the left of the second text box, with lines radiating from it towards the text, suggesting a narrator is speaking.

```
14         raise AttributeError('Can only be accessed via an instance.')
15     if self.name not in instance.__dict__:
16         raise AttributeError
17     return instance.__dict__[self.name]
18
19     def __set__(self, instance, value):
20         if self.name is None: self.fetchattr(instance)
21         self.val = value
22
23     def __delattr__(self, instance):
24         del self.val
25
26     def fetchattr(self, instance):
27         for attr in instance.__class__.__dict__:
28             if attr.ast|
29
30     def __get__(self, name):
31         name = name.lstrip('_')
32         if name in self.__dict__:
33             raise AttributeError('Name conflict error with attribute n
34         elif name.startswith('_'):
35             return '_' + name
36         else:
37             return '__' + name
38     return name
39
40
```

Python Screencast - Descriptors

5,899 views

43 0 SHARE

```
14 raise AttributeError('can only be accessed via an instance.')
15 if self.name not in instance.__dict__:
16     raise AttributeError
17     return instance.__dict__[self.name]
18
19 def __set__(self, instance, value):
20     if self.name is None: self.fetchattr(instance)
21     self.val = value
22
23 def __delete__(self, instance):
24     del self.val
25
26 def fetchattr(self, instance):
27     for attr in instance.__class__.__dict__:
28         if attr.as|
29
30 def nangle(self, name):
31     if name != None:
32         if name.startswith('__'):
33             raise AttributeError('Name conflict error with attribute n
34         elif name.startswith('_'):
35             return '_' + name
36         else:
37             return '__' + name
38     return name
39
40
```

Python Screencast - Descriptors

5,899 views

43 0 SHARE

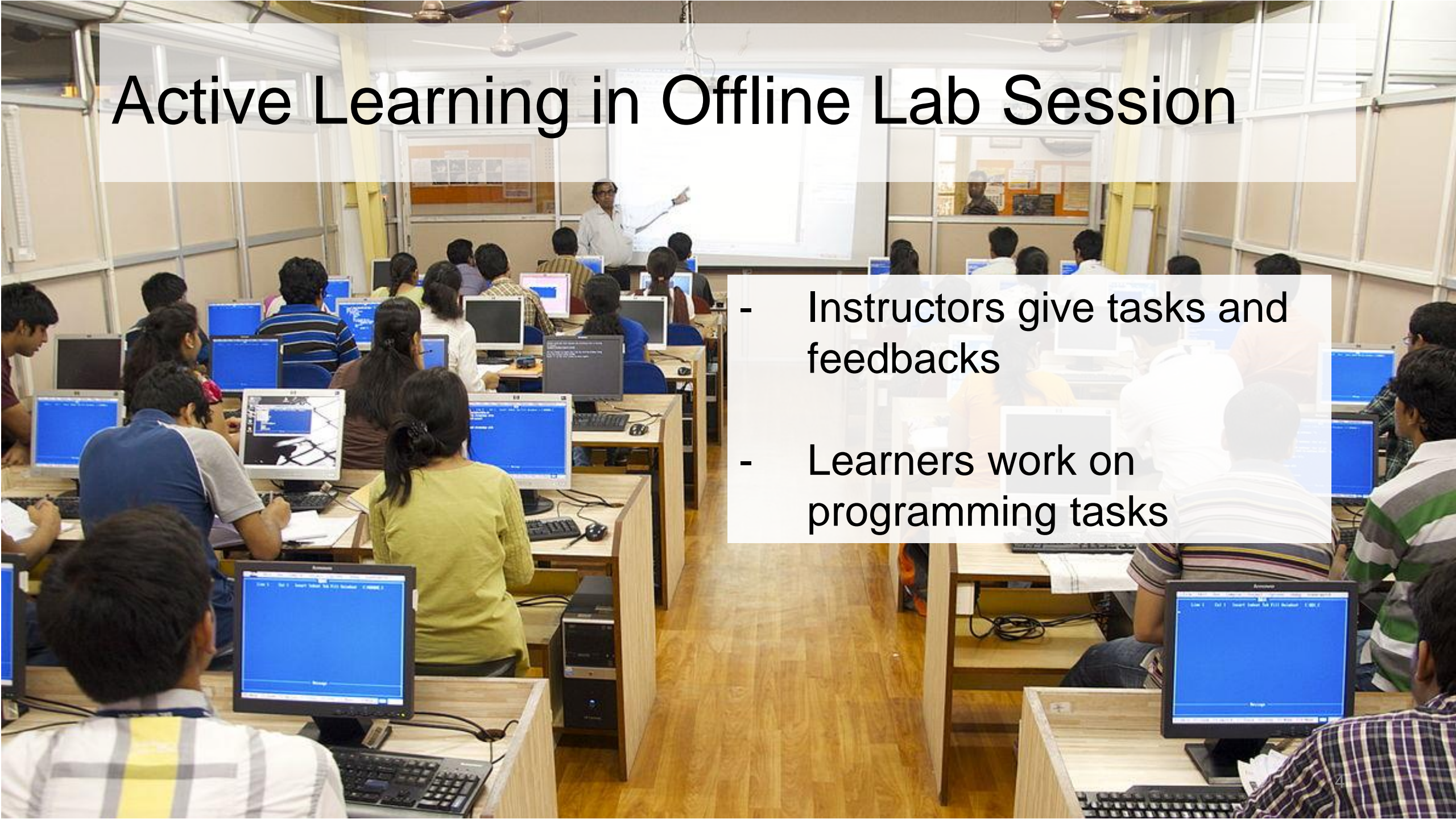
Lack of Support for Active Learning

- Limited support for interaction with the content
- Separated experience between learn and practice

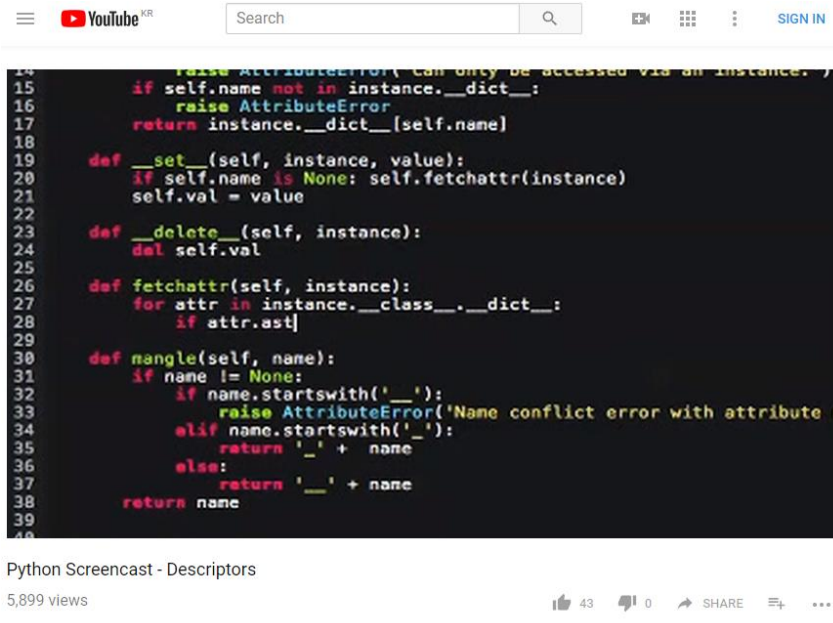


Active Learning in Offline Lab Session

- Instructors give tasks and feedbacks
- Learners work on programming tasks



Promoting Active Learning in Instructional Programming Screencast



> Demonstration of the process of writing code



> Hands-on programming experience

Elicast: Embedding Interactive Exercises in Instructional Programming Screencasts

```
/* Elicast */  
Python lambda, map, filter  
5 return x + 1  
6  
7 print(add_1(3))  
8  
9 add_1_lambda = lambda x: x + 1  
10 print(add_1_lambda(5))  
11  
12 l = [1, 3, 2, 4]  
13 l2 = []  
14 for e in l:  
15     l2.append(e + 1)  
16  
17 print(l2)  
18  
19 print(list(map(add_1, l)))  
20 print(list(map(lambda x: x + 1, l)))  
21  
22 l_twice_2 = list(map( /* Write your answer here */ , l))
```

Load

>_ Run Check Answer Skip Exercise

Output

```
4  
6  
[2, 4, 3, 5]  
[2, 4, 3, 5]  
[2, 4, 3, 5]
```

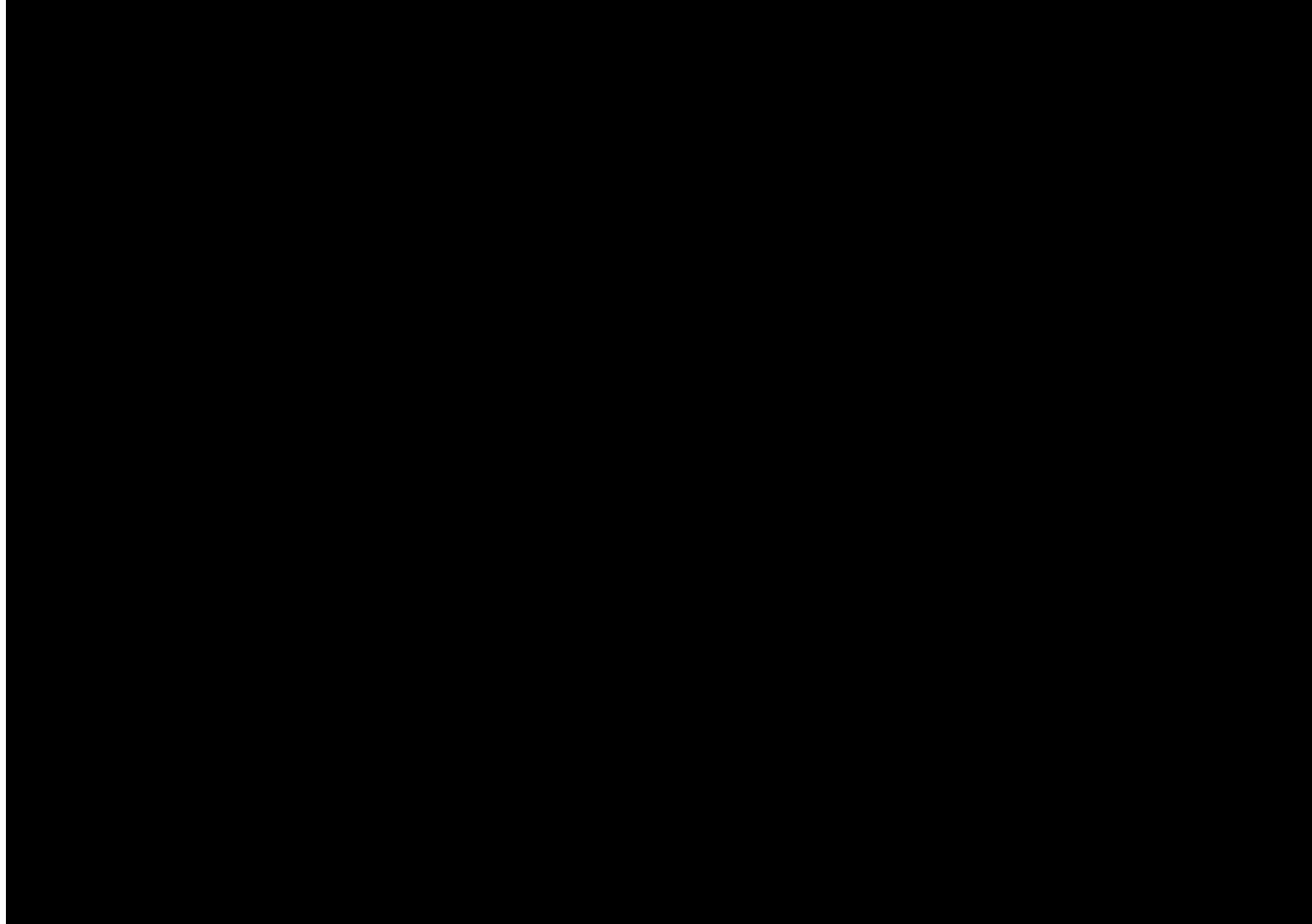
5:59 / 12:22

Text-based
Screencast

Automated
Assessment

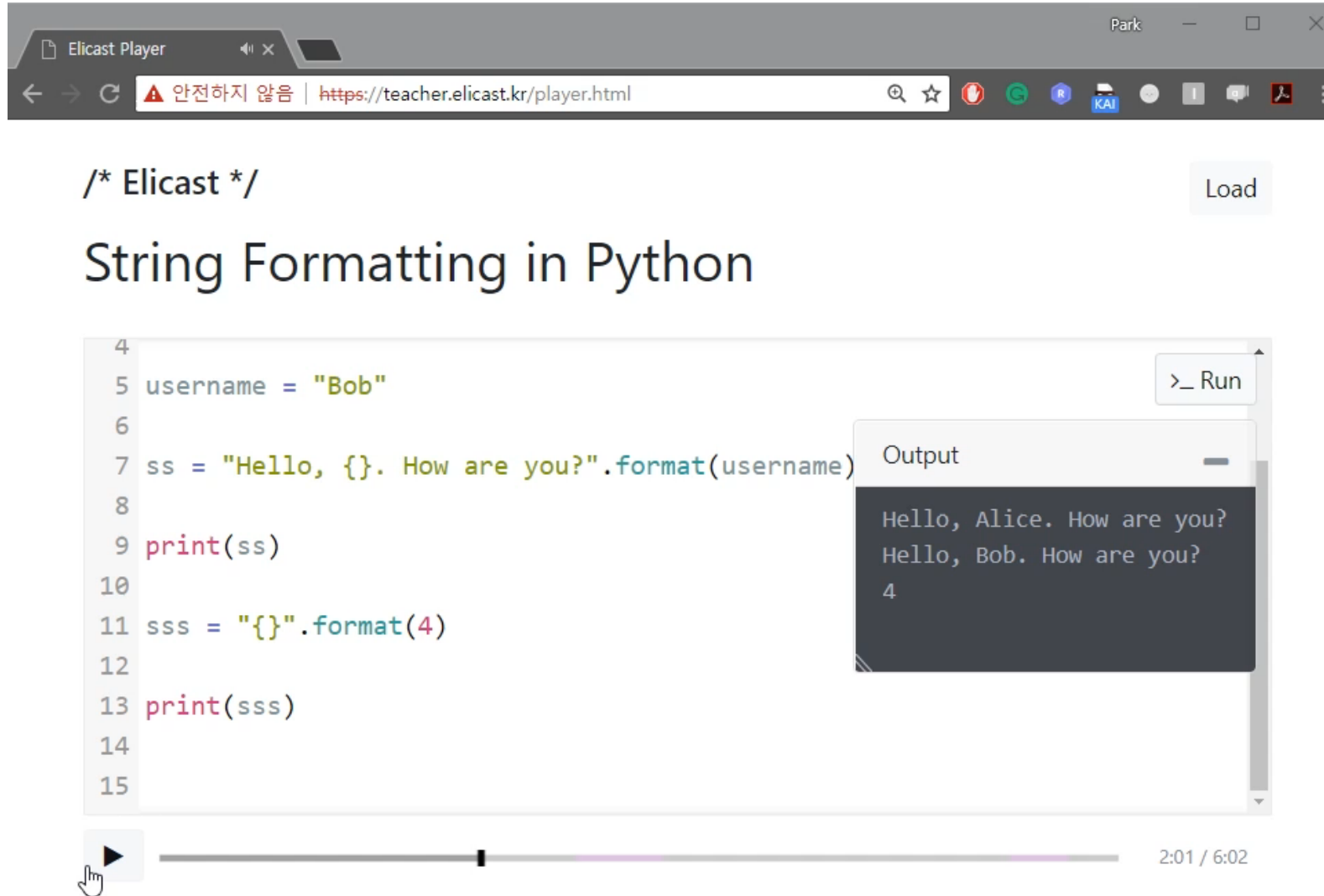
Embedded
Exercise

Text-based Programming Screencast



<https://youtu.be/dKWlqDLgsm8>

Embedded Interactive Exercise



The screenshot shows a web browser window titled "Elicast Player" with the URL <https://teacher.elicast.kr/player.html>. The page content includes a comment `/* Elicast */`, a "Load" button, and the title "String Formatting in Python". Below this is a code editor with the following Python code:

```
4
5 username = "Bob"
6
7 ss = "Hello, {}. How are you?".format(username)
8
9 print(ss)
10
11 sss = "{}".format(4)
12
13 print(sss)
14
15
```

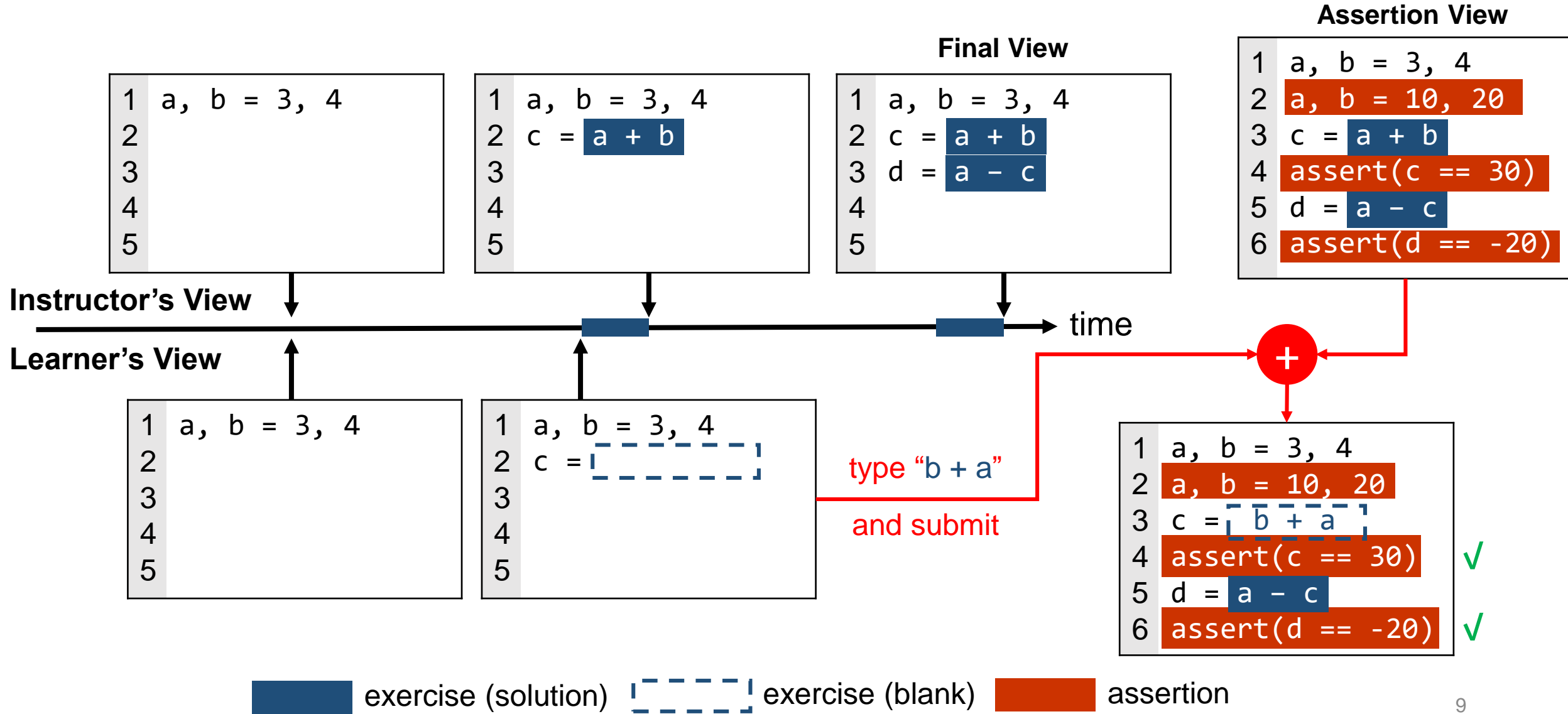
To the right of the code editor is an "Output" window showing the results of the code execution:

```
Hello, Alice. How are you?
Hello, Bob. How are you?
4
```

At the bottom of the interface is a video player with a play button, a progress bar, and a timestamp of 2:01 / 6:02.

<https://youtu.be/KZZIvBtDwXU>

Assertion-based Automated Assessment



Exploratory Study

Study 1. Instructors record exercise embedded screencast

Q) how instructors make use of embedded exercise in creating screencast lectures?

Study 2. Learners watch screencast and engage in exercises

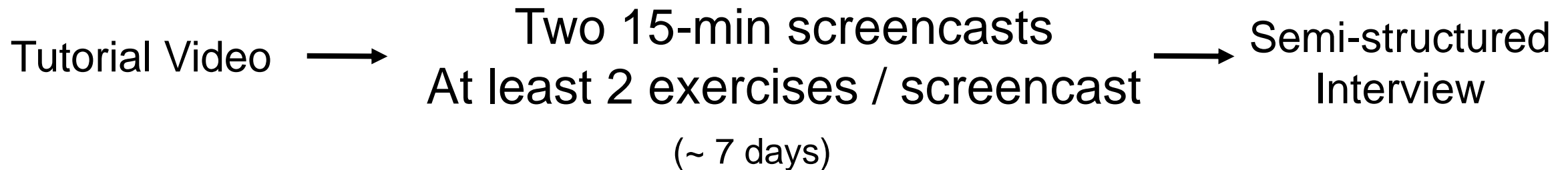
Q) how learners engage with the exercise embedded screencasts?

Study 1. Instructors Record Exercise Embedded Screencasts

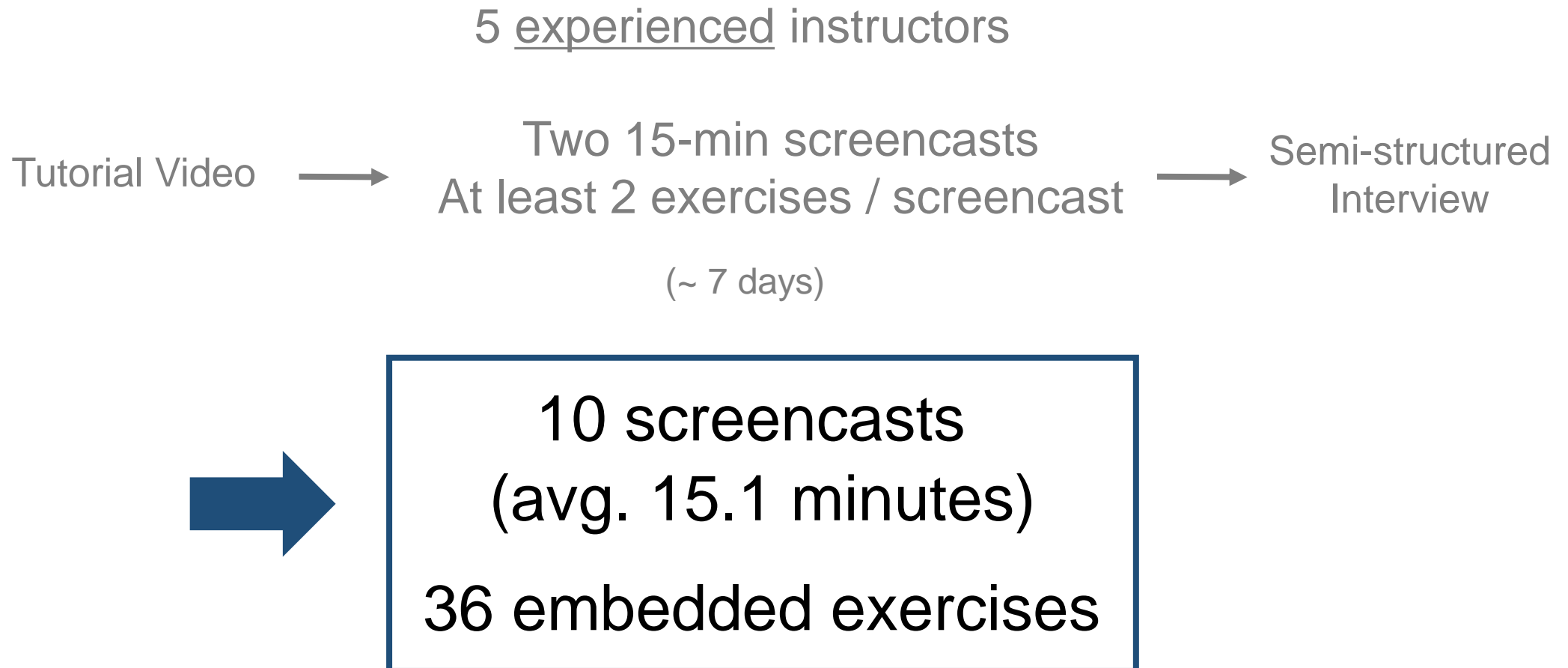
5 experienced instructors

Study 1. Instructors Record Exercise Embedded Screencasts

5 experienced instructors



Study 1. Instructors Record Exercise Embedded Screencasts



Findings From Study 1

Modularized, checkpoint-style learning units

Assertions are easy-to-create yet limited

Expectation of pedagogical benefits

Findings From Study 1

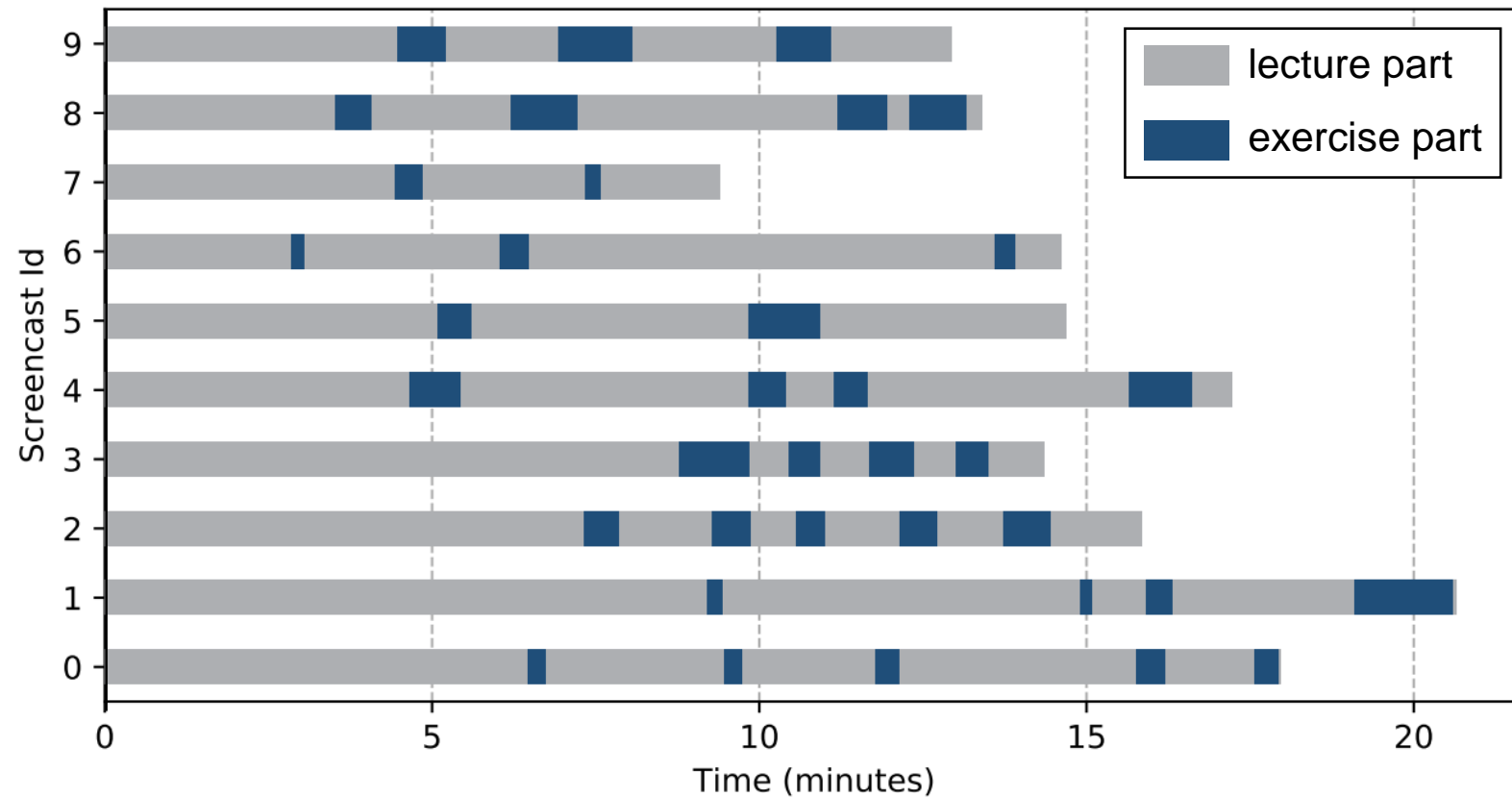
Modularized, checkpoint-style learning units

Assertions are easy-to-create yet limited

Expectation of pedagogical benefits

Modularized, Checkpoint-style Learning Units

Instructors tended to organize each screencast lecture into smaller learning units.



Modularized, Checkpoint-style Learning Units

“With Elicast, what I felt different from the conventional lecture style was that I could define finer-grained goals of the lecture...” (instructor 1)

```
2 def isValidPlace (state, place):
3     (x, y) = place
4     /* Write your answer here */
5     return True
6
7
8 def dummy_player (state):
9     return (0, 0)
10
11 def player1 (state):
12     for i in range(3):
13         for j in range(3):
14             if state[i][j] == ".":
15                 return (i, j)
```

Output

```
(0, 0)
(0, 1)
```

6:28 / 17:57

Two independent sub-goals

```
3 def isValidPlace (state, place):
4     (x, y) = place
5     if x < 0 or y < 0 or x > 2 or y > 2:
6         return False
7     elif state[x][y] != ".":
8         return False
9     return True
10
11 def getFlippedState (state):
12     /* Write your answer here */
13
14 def dummy_player (state):
15     return (0, 0)
16
```

Output

```
0
```

9:28 / 17:57

Findings From Study 1

Modularized, checkpoint-style learning units

Assertions are easy-to-create yet limited

Expectation of pedagogical benefits

Assertions Are Easy-to-Create Yet Limited

The instructors spent a median of 1.82 minutes on writing assertions.

```
20
27 def sum_all(*args):
28     sum = 0
29     for i in args:
30         sum += i
31     return sum
32
33 assert(sum_all(1,2,3) == 6)
34 assert(sum_all(1,2,3,4,5,6) == 21)
35
36 print(sum_all(1,2,3))
37 print(sum_all(1,2,3,4,5,6))
38
39 def sum_multiply(a, b):
40     s = a + b
```

Unit test style

Record Assert

Output

```
김재원
10
25
김재원
체리
토끼
김재원
6
21
(15, 50)
```

16:21 / 16:21

```
41 print(q.size()) # 2 # q안에 몇 개의 사료가 들어있는가?
42 q.__init__()
43 q.push(1)
44 q.push(2)
45 q.push(3)
46 q.pop()
47 q.push(4)
48 q.push(5)
49 assert(q.front() == 2)
50 assert(q.size() == 4)
51
52 q.pop()
53 assert(q.front() == 3)
54 assert(q.size() == 3)
55 q.pop()
```

Functional test style

Record Assert

Output

```
3
4
2
```

16:27 / 16:27

Assertions Are Easy-to-Create Yet Limited

“If I wanted to test a condition in an if statement, then it would be quite difficult. There are certain places I can set as an input field, ...”
(Instructor 1)

“Some things cannot be test with assertions., especially when assessing based on how well the student formed the code structure. This is essential when we teach novice students, ...” (Instructor 4)

Findings From Study 1

Modularized, checkpoint-style learning units

Assertions are easy-to-create yet limited

Expectation of pedagogical benefits

Expectation of Pedagogical Benefits

While some instructors felt recording with Elicast took more time and effort, all expected that Elicast would be pedagogically beneficial to students.

“I like the fact that students would feel they are writing code with me, rather than repeating after me ... I like how students would feel they’re learning together.” (Instructor 1)

“... this tool allows the instructor to quickly create small-sized exercises, which is intuitive to both instructor and students, and students can check if they actually understood the lecture – by doing.” (Instructor 4)

Study 2. Learners Watch Screencast and Engage in Exercises

63 undergraduate students

The majority (46/63) had taken only one CS course before

Study 2. Learners Watch Screencast and Engage in Exercises

63 undergraduate students

The majority (46/63) had taken only one CS course before

For each student,

Id	Title	Duration	# exercises
L1	Max Machine	15:51	5
L2	Queue	14:21	4
L3	Python RE	20:38	4

randomly
select

Screencast A
Screencast B

randomly choose one
and remove exercises

Screencast A'
Screencast B'
or
Screencast A
Screencast B'

Study 2. Learners Watch Screencast and Engage in Exercises

63 undergraduate students

The majority (46/63) had taken only one CS course before

For each student,

Id	Title	Duration	# exercises
L1	Max Machine	15:51	5
L2	Queue	14:21	4
L3	Python RE	20:38	4

randomly
select

Screencast A
Screencast B

randomly choose one
and remove exercises

Screencast A'
Screencast B
or
Screencast A
Screencast B'

Pre-survey → Pre-test → Screencast → Post-test → Post-survey

↑
Repeat for

Screencast A' and B
or
Screencast A and B'

Findings From Study 2

Active engagement in lectures

Preliminary evidence on higher learning gains

Learning by doing

Findings From Study 2

Active engagement in lectures

Preliminary evidence on higher learning gains

Learning by doing

Learners Engage in Lectures Actively



Tried at least once
(Correct-1, Correct-N, Give-Up)

90.44%

Correctly answered
(Correct-1, Correct-N)

73.16%

Learners Engage in Lectures Actively

The number of video navigations per student
(play, pause, seeking)

25.16

w/ Exercise

>>

unequal var. t-test
*p < 0.005

16.30

w/o Exercise

Learners Engage in Lectures Actively

The number of video navigations per student
(play, pause, seeking)

25.16

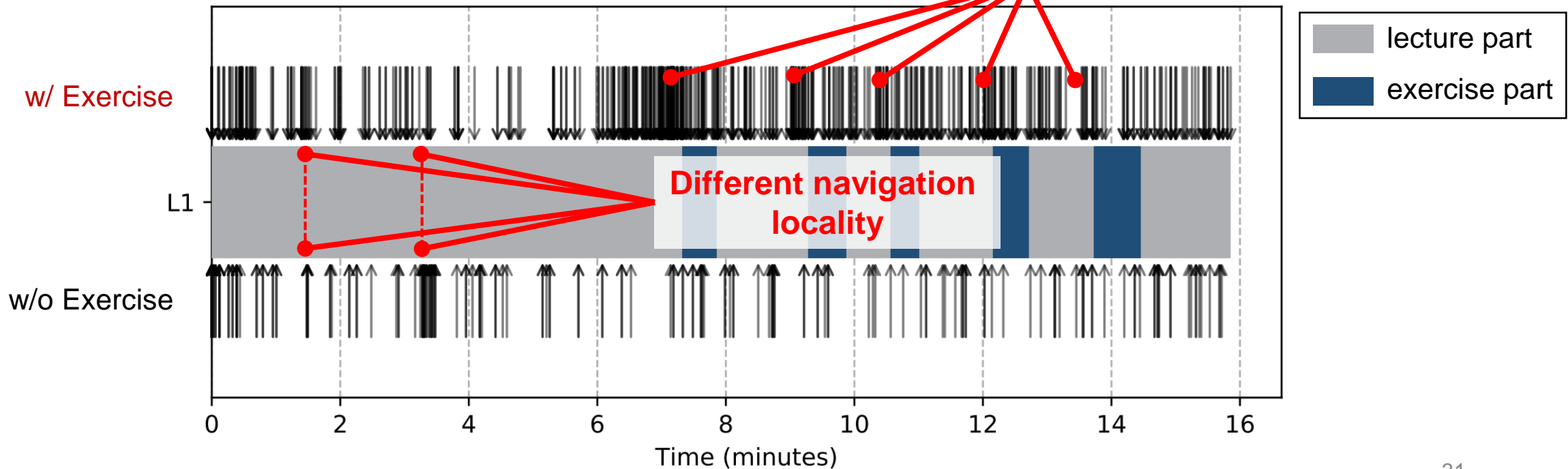
w/ Exercise

>>

unequal var. t-test
*p < 0.005

16.30

Dense navigations
around exercise parts



Learners Engage in Lectures Actively

13 students mentioned that they were able to stay focused and be engaged throughout the lecture because of the embedded exercises.

“Online lectures are usually disengaging, but I stayed focused this time in order to solve the problems.” (Student 17)

“It made me take time to write code and apply things that I might have overlooked otherwise.” (Student 56)

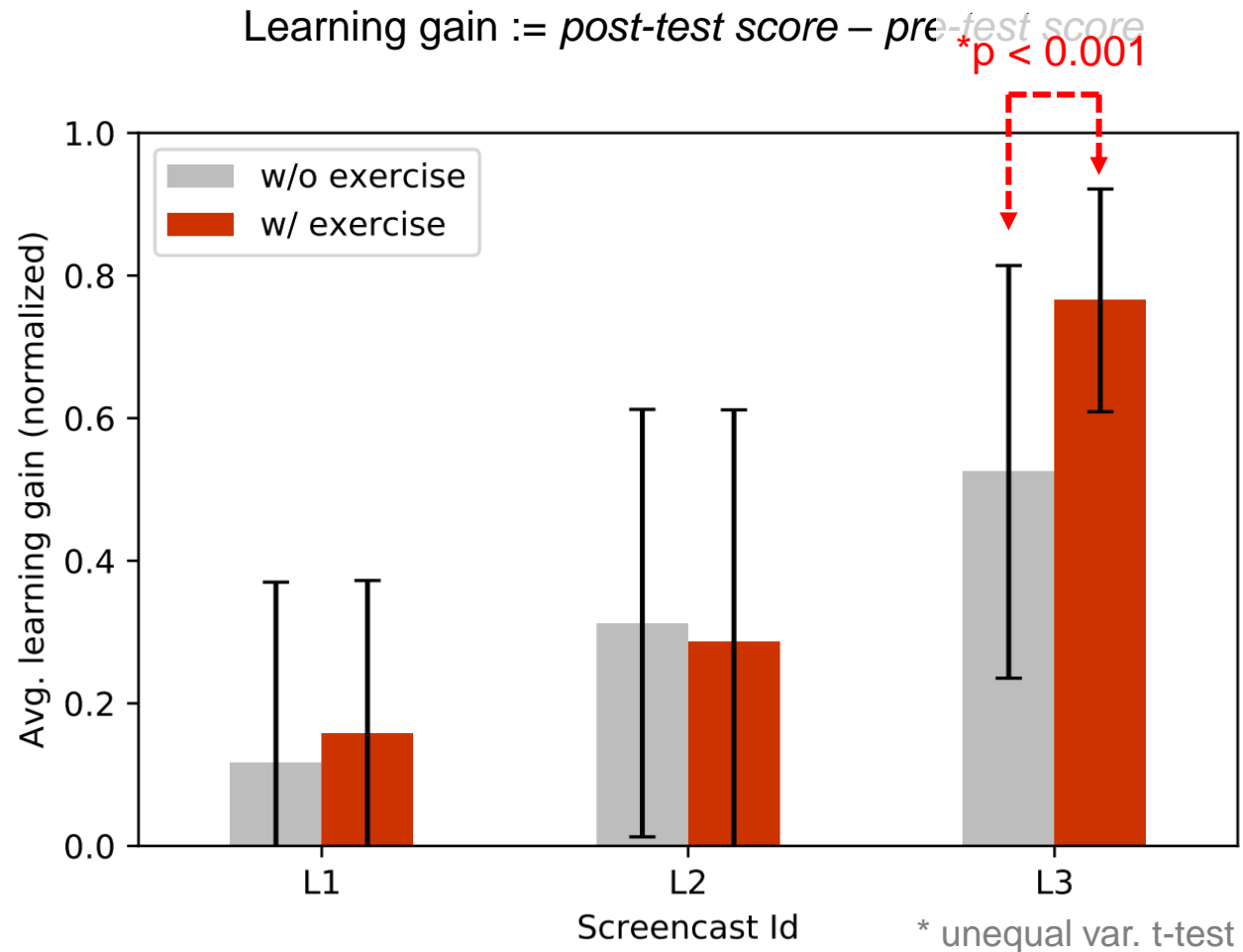
Findings From Study 2

Active engagement in lectures

Preliminary evidence on higher learning gains

Learning by doing

Preliminary Evidence on Higher Learning Gains



Preliminary Evidence on Higher Learning Gains

- Applying What I just Learned (11)* “I realized that understanding something conceptually is quite different from applying it in practice” (Student 43)
- Checking for Understanding (8)* “... It gave me a chance to think twice about the contents that I was going to go through in confusion.” (Student 58)
- Memorizing (6)* “Solving the exercises during the lecture, I was able to take control of my own learning, and I will probably remember longer through repetition of the concept” (Student 11)

Findings From Study 2

Active engagement in lectures

Preliminary evidence on higher learning gains

Learning by doing

Elicast Promotes Learning by Doing

The number of code executions
(excluding submissions for exercises)

5.14

w/ Exercise

>>

unequal var. t-test
*p < 0.001

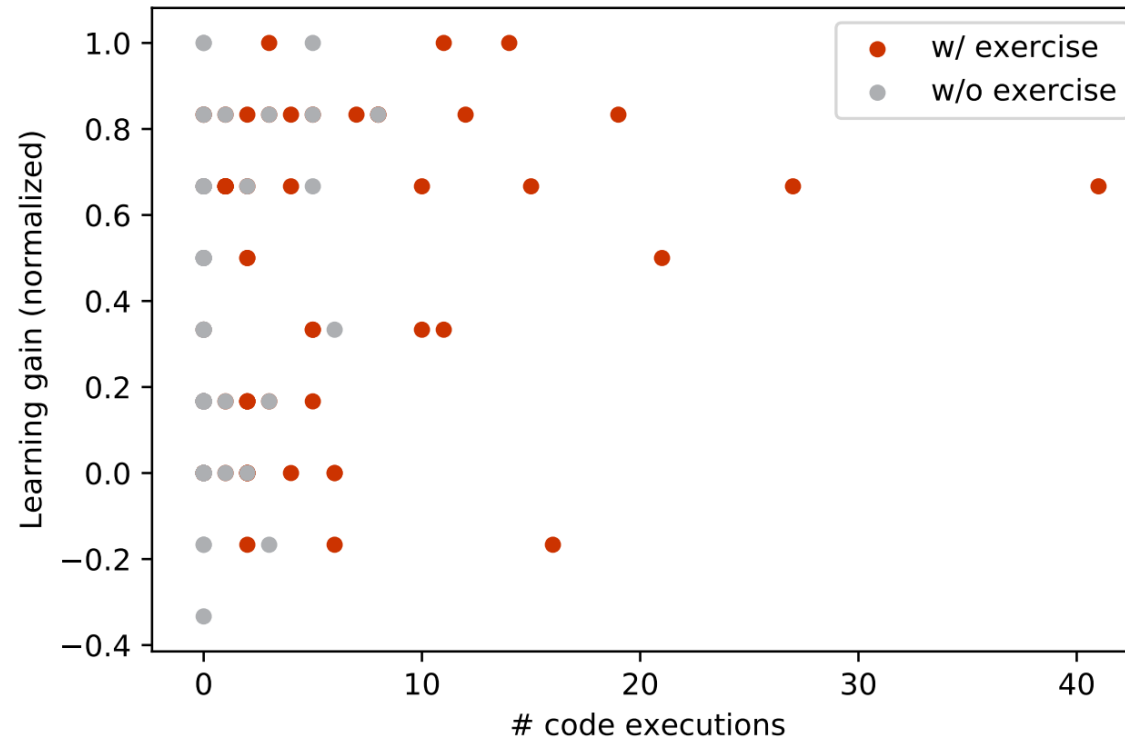
0.71

w/o Exercise

Elicast Promotes Learning by Doing

(Pearson's $r = .26$)

Positive correlation between the number of code executions and learning gain



Future Direction

Provide learners' activity as feedback to instructors

“The most skipped exercise would be my primary interest. Then I would improve my lecture based on that data.” (Instructor 2)

“... Lecturers could know who did not understand which part of the lecture. ... This could not be done in my past lecture experiences.” (Instructor 4)

Provide more guidance for exercises to learners

“... but I thought I needed some hints that would guide me in solving problems and lead me to the intended direction ...”
(Student 31)

Summary

- We present **Elicast**, a screencast tool for recording and viewing programming lectures with **embedded programming exercises**
- Elicast positively influenced the behaviors of both instructors and learners
 - Instructors – smaller learning units using embedded exercises as checkpoints
 - Learners – active engagement in the lecture with embedded programming exercises

Code: <https://github.com/elicast-research/elicast>

Elicast: Embedding Interactive Exercises in Instructional Programming Screencasts

/* Elicast */

Load

Python class

```
1 class Animal:
2
3     def __init__(self, name, num_legs):
4         self.name = name
5         self.num_legs = num_legs
6
7 cat = Animal('Cat', 4)
8
9 class Car:
10     # model, num_wheels
11     /* Write your answer here */
```

>_ Run

Check Answer

Skip Exercise

Output



5:04 / 14:41