Spring 2016

Interactive Computer Science Exercises In edX

Hong Le
San Jose State University

Follow this and additional works at: https://scholarworks.sjsu.edu/etd_projects

Part of the [Other Computer Sciences Commons](#), and the [Software Engineering Commons](#)

Recommended Citation

Le, Hong, "Interactive Computer Science Exercises In edX" (2016). *Master's Projects*. 473.
DOI: <https://doi.org/10.31979/etd.hrr2-y8rc>
https://scholarworks.sjsu.edu/etd_projects/473

This Master's Project is brought to you for free and open access by the Master's Theses and Graduate Research at SJSU ScholarWorks. It has been accepted for inclusion in Master's Projects by an authorized administrator of SJSU ScholarWorks. For more information, please contact scholarworks@sjsu.edu.

Interactive Computer Science Exercises In edX

A Project Report

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Computer Science

By Hong Le

May 2016

© 2016

Hong Le

ALL RIGHTS RESERVED

The Designated Thesis Committee Approves the Thesis Titled
Interactive Computer Science Exercises In edX

by
Hong Le

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE
SAN JOSÉ STATE UNIVERSITY

May 2016

Dr. Horstmann	Department of Computer Science
Dr. Pearce	Department of Computer Science
Dr. Teng	Department of Computer Science

ACKNOWLEDGEMENTS

I have an immeasurable pleasure to present my acknowledgement and gratitude to everyone who has given me his or her hand in this project. I am very grateful to my project advisor, Dr. Cay Horstmann for giving me valuable guidance and constantly encouraging me throughout the project. I am thankful to my committee members, Dr. Jon Pearce and Dr. Teng Moh, for their encouragement and recommendations. Without all of that, this project would not have been completed.

ABSTRACT

This project focuses on improving online learning courses for Computer Science. My approach is to create a platform in which interactive exercises can be implemented for students to work on. Methodology includes creating plugins for interactive exercises using XBlock, a component architecture for building independent online courses on edX. The exercises are based on existing exercises like CodeCheck and Wiley's InterActivities Exercise System. In order to integrate these exercises, I implemented CodeCheck XBlock and Interactive XBlock. These Xblocks allow students to work on interactive exercises on edX, and instructors to view and download students' submissions.

Table of Contents

d tTable of Contents.....	1
List of Figures	3
List of Tables.....	4
List of Acronyms and Terms	5
1. Introduction.....	6
2. Extending Learning Management Systems	7
2.1. Closed Source LMS.....	8
2.1.1. Coursera.....	8
2.1.2. Udacity.....	8
2.2. Open Source LMS.....	9
2.2.1. Moodle.....	9
2.2.2. Canvas.....	10
2.2.3. edX.....	10
2.3. Why edX	10
3. Codecheck And Interactive Exercises.....	11
3.1. CodeCheck.....	11
3.2. Interactive Exercises	12
4. Introduction To Xblocks	15
4.1. The XBlock Component Architecture	15
4.2. Existing XBlocks.....	16
5. Open edX Overview	18
5.1. Open edX.....	18
5.2. Devstack.....	19
5.2.1. Installing a DevStack Instance.....	19
5.2.2. Run a DevStack Instance.....	20
5.3. FullStack.....	20
5.3.1. Installing a FullStack Instance on AWS.....	20
5.3.2. Running a FullStack Instance on AWS.....	21
6. Implementing XBlocks.....	22

6.1. Creating a Sample XBlock.....	22
6.2. Running an XBlock on the XBlock SDK.....	22
6.3. Installing and Running an XBlock on DevStack and FullStack	24
6.3.1. <i>Installing an XBlock on DevStack</i>	24
6.3.2. <i>Installing XBlock on FullStack</i>	24
6.3.3. <i>Running an XBlock on DevStack and FullStack</i>	24
6.4. XBlock Structure.....	25
7. The Codecheck XBlock.....	26
7.1. Overview.....	26
7.2. Implementation	27
7.2.1. <i>Studio View</i>	27
7.2.2. <i>LMS View</i>	29
8. The Interactive XBlock.....	35
8.1. Overview.....	35
8.2. Implementation	36
8.2.1. <i>Studio View</i>	36
8.2.2. <i>LMS View</i>	37
8.3. Third-party tools	40
9. Evaluation.....	41
9.1. Instructor Responses.....	41
9.2. Student Responses.....	43
10. Challenges and Conclusion	45
Appendix A	47
Appendix B	48
Reference	51

List of Figures

Figure 1 - LMS Market Share in US & Canada [4]	9
Figure 2 - CodeCheck Question	12
Figure 3 - CodeCheck Report	12
Figure 4 - Code Tracing Interactive Exercise	14
Figure 5 - Object Diagram Interactive Exercises	14
Figure 6 - Rearrange Code Interactive Exercises	15
Figure 7 - Sample Computer Science online exercise	17
Figure 8 - CodeBoard.io XBlock	18
Figure 9 - XBlock SDK Server	23
Figure 10 - Sample XBlock File Tree	26
Figure 11 - CodeCheck XBlock Staff View on LMS	27
Figure 12 - CodeCheck XBlock Studio View	28
Figure 13 - CodeCheck XBlock Workflow	29
Figure 14 - CodeCheck Student View on LMS	30
Figure 15 - Display Report on LMS	32
Figure 16 - CodeCheck XBlock Student submission table on LMS - Staff view	33
Figure 17 Pop up with student report	33
Figure 18 - CSV file's content	35
Figure 19 - Interactive XBlock Staff View on LMS	36
Figure 20 - Interactive XBlock Studio View	36
Figure 21 - Interactive XBlock Workflow	37
Figure 22 - Interactive XBlock Student View on LMS	38
Figure 23 - Student's grade on LMS	39
Figure 24 - Interactive XBlock Staff View Table Submission on LMS	40

List of Tables

Table 1 - LMS Platform Comparison 8

List of Acronyms and Terms

AJAX - asynchronous javascript and xml

API - Application Programming Interface

CMS - Course Management System

CSS - Cascading Style Sheets

CSV - Comma Separated Values

HTML - Hyper Text Markup Language

JSON - Javascript Object Notation

LMS - Learning Management System

LTI - Learning Tools Interoperability

MOOC - Massive Open Online Course

Moocs - Massive Open Online Courses

URL – Uniform Recourse Identifier

ORA - Open Response Assessments

SDK - Software Development Kit

1. Introduction

A Massive Open Online Course (MOOC) is a way to connect, collaborate and learn in the networked world. MOOCs are online courses, which offer learning materials that can be modified, reused, and redistributed to reach massive communities [1]. Nowadays, there are many existing MOOC platforms and providers for students to participate in; among them is edX. edX is not only a MOOC provider, but also a platform that hosts online courses in many different categories. Although edX platform is compatible with other MOOC providers, many universities, corporations, non-government organizations, government and entrepreneurs are using edX as MOOC provider as well as the platform to host online courses.

However, there are some disadvantages of taking online courses. According to Gou [2], taking online courses makes it easier for students to skip through the lecture content than traditional in-person classroom. The study also shows that students who paid for online courses only view on average seventy-eight percent of the course's content and skip twenty-two percent of it. It also shows that viewers who take free online courses would skip even more of the course content. This has become a big concern for professors since their students do not express a good understanding of the materials being taught because they do not watch the lecture seriously. One of the best ways for students to learn is not just by reading but also by doing interactive problems and working on the code by themselves. The more students practice, the better they get. Unlike traditional classrooms, in the online classroom, students don't have many opportunities to interact with professors or to ask questions during the lectures. Not only that, most online questions in computer science are mainly multiple-choice questions, there aren't many programming questions that students can actually code and get feedback on like they would in a traditional setting.

The goal of this project is to create edX exercise components, which will allow professors to create interactive problems for students to work on. For online courses in computer science, there are many things needed to achieve a successful course, such

as good lecture videos, detailed lecture notes, student-professor or student- student communication, and course exercises. After students are introduced to new material, they need to work on practice exercises to apply the material they learned to better understand it. For computer science students, they need to work on code comprehension, algorithm visualization, code tracing, writing pseudocode and programming problems to later study the lecture's material to improve their skill and understanding of the materials. Because of that, I want to embed those interactive problems on edX so computer science students will be able to work on interactive problems when they are taking online courses with the edX platform.

I divided this project into two parts: interactive exercises and programming problems. Interactive exercises ask students to work on conceptual problems such as tracing code or drawing a tree structure or linked list based on given code. Students need to successfully work on each step of the exercise by interacting with a drawing to achieve the correct answer. Programming problems use unit tests and automatic evaluation, which gives the student a question and a textbox to fill in their code. After the student fills in their answer in the text box, it will give students instant feedback on their code to see wherever it is correct or not. In this project, I embed interactive exercises and programming problem to edX.

2. Extending Learning Management Systems

A Learning Management System (LMS) is a platform for hosting a course, which could be a MOOC, a small or closed online course, or a course that is not online [2]. Nowadays, there are many existing LMS platforms, such as Moodle, Canvas, Coursera, Udacity, or edX. Many students and universities are using those platforms. Some features of these systems are shown in Table 1.

Function	Moodle	Coursera	Canvas	edX	Udacity
Video lecture	✗	✓ (stored on Coursera)	✗	✓ (stored on YouTube)	✓ (stored on YouTube)
Quiz integrated with video	✗	✓	✗	✗	✓
Store and calculate grade	✓	✓	✓	✓	✓
Open sources	✓	✗	✓	✓	✗

Table 1 - LMS Platform Comparison

2.1. Closed Source LMS

2.1.1. Coursera

Coursera is an educational for-profit website. It is a MOOC platform and the course is like a classroom model. Coursera has a large number of university partners from all around the world. It also provides a mobile app for learners to access the course on their phone. Many of its courses are not free; students have to pay a fee to be able to access the course. Universities pay Coursera a fee to host courses. It is not possible for universities to modify the Coursera platform for supporting new exercise types.

2.1.2. Udacity

Udacity is a MOOC platform that offers both free and paid courses in Computer Science. Udacity courses are asynchronous courses, which students can start anytime that they want [3]. A unique feature about Udacity is that its video lectures include many quizzes that show up every few minutes. The quiz content is related to the video content. This feature helps to check if students follow the video lecture. Udacity is no

longer working with many universities. Like Coursera, it is not possible for third parties to modify their platform for supporting new exercise types.

2.2. Open Source LMS

Figure 1 - LMS Market Share in US & Canada [4]

2.2.1. Moodle

Moodle is an open source that is used widely now as shown in Figure 1. It is a modular system based on activity modules. There are many kinds of activity modules that help professors build the course. In terms of course structure, Moodle is a folder structure type, which means it organizes all materials as a list of categories or a list of topics on

course homepage. It also provides collaborative and communicative activity modules for students to connect with instructors or other students [5].

If one wants to create a new activity module that supports exercises different from the ones that are already provided on Moodle, there is a question type plugin API.

Developers can implement their own Moodle question type plugin using PHP code [6].

2.2.2. Canvas

Canvas is also an LMS system, which was developed by an educational technology business based in Salt Lake City, which is also widely adopted, as shown in Figure 1.

Canvas does not have an API for adding question types. For Canvas, to create a new custom question type, one has to use the Learning Tools Interoperability (LTI) standard [7]. LTI is a standard way to integrate external tool with education platform such as LMS. If one wants to create a new exercise type that can be used in Canvas, one has to develop it as a separate application that supports the LTI standard, and then integrate it into Canvas. However, Canvas does not allow an assignment to have more than one LTI problem. It is not possible to make an assignment in Canvas that is made up of multiple LTI problems or a mixture of LTI and supported problem types.

2.2.3. edX

edX is a MOOC provider [8]. Unlike Udacity and Coursera, edX is a non-profit organization and runs on open-source platform, Open edX [8]. Its platform includes an LMS (the student interface) and Studio (a course authoring website). In order to create custom activities, one uses the XBlock plugin architecture [8]. It is equivalent to the question type plugin in Moodle.

2.3. Why edX

Coursera and Udacity are closed source platforms; their vendors host all the system development. They don't allow developers to build and customize their own course. Usually, Udacity or Coursera will partner with a university to get the course content from

the university. After that, they will build the course entirely by themselves. For those reasons, I'm not considering them for my project.

Interactive exercises can be developed on Moodle using the question type plugin API. It can also be built on Canvas as an LTI external tool. However, LTI is for creating big and complex content. You can't create an LTI problem, which can display many interactive exercises at the same time. For example, in one quiz problem, the instructor can only put one interactive exercise. They can't combine many problems together using LTI for problems in the quiz. Most interactive exercises are small problems and students can work on many interactive exercises in one assignment.

Moreover, both Canvas and Moodle are not following a course structure, but they use a folder structure, which is not popular anymore. In other words, these systems can't get back to the last activity stream where the user left off before. For example, suppose a student logs out after they finished reading lesson 3. When they log in again next time, students can't get back to where they were the last time and have to look for lesson 3 again.

3. Codecheck And Interactive Exercises

3.1. CodeCheck

CodeCheck [9], which was developed by Cay Horstmann and his students, is a unit-testing library for automatic evaluation of programming problems. Instructors can give an assignment on CodeCheck to students. Students fill in their coding answers in the textbox and press the Submit button. After that they will be given a score and evaluation of their answer.

For students to work on a CodeCheck problem, they have to access the CodeCheck website using a specific link, which is provided by instructors. Each problem is on one individual page, which includes the problem's question, the answer textbox and Submit button.

Figure 2 - CodeCheck Question

Figure 3 - CodeCheck Report

3.2. Interactive Exercises

For his electric textbook [12], Cay Horstmann designed interactive exercise (called InterActivities by his publisher). In the past, computer science students would study

based on reading a textbook or listening to lectures. They can also watch algorithm animations or videos online. However, watching animations or videos is not as effective as letting students actively participate [10]. Interactive exercises allow students to practice pre-coding activities with immediate feedback and automatic scoring. There are many types of interactive exercises such as code tracing, object diagram, and rearranging code.

A code tracing exercise is a type of exercise where students have to fill in the values of variables and trace the flow of a program by going to the next line at branch points as shown in Figure 4. Whenever students make an error by going to the wrong line or fill in the wrong variable's value, an error message will be displayed. A hint button, which will show the next correct step, will be displayed when the student makes two errors in a row.

For object diagram exercises, students track stack variables and dynamically allocated objects. Figure 5 shows an object diagram and code related to the diagram. The diagram is about inserting node into a linked list. Students should drag the arrow to the next step `newNode.next = first`.

The rearrange code exercise is a puzzle for producing programs from prepared statements. Students need to rearrange lines of code in the correct order and indentation as shown in Figure 6. The code may also contain distractors.

11. Trace through the following loop.

Please click on the next line.

```
int n = 1796;
int count = 0;
while (n > 0)
{
    int digit = n % 10;
    if (digit == 6 || digit == 7)
    {
        count++;
    }
    n = n / 10;
}
System.out.println(count);
```

n	count	digit	Output
1796	0		

0 errors

Start over

Figure 4 - Code Tracing Interactive Exercise

1. Trace through the following code for adding a new element to the beginning of a linked list. Assume the value of the element parameter is Peter.

```
public class LinkedList
{
    ...
    public void addFirst(Object element)
    {
        Node newNode = new Node(); ① ②
        newNode.data = element; ③
        newNode.next = first; ④
        first = newNode; ⑤
    }
    ...
}
```

④ Initialize the next instance variable of the new node

Draw the arrow.

2 correct, 0 errors

Start over

Figure 5 - Object Diagram Interactive Exercises

3. Your task is to design a method

```
int[] allMatches(int[] values, int size, int a, int b)
```

that returns all values between a and b inclusive in the partially filled array `values`. For example, if an array `a` contains 11, 3, 9, 4, 2, 5, 4, 7, 6, 0, then the call `allMatches(a, 8, 3, 7)` should return an array containing 3, 4, 5, 4, 7. Note that the 6 is not included because it is past the size of the partially filled array.

Arrange the pseudocode below in the right order.

Order the statements by dragging them into the left window. Use the guidelines for proper indenting.

<pre>count=0</pre>	<pre>return result</pre>
<pre>for each v in values</pre>	<pre>for each v in values</pre>
<pre> if a ≤ v ≤ b</pre>	<pre> i++</pre>
<pre> count++</pre>	<pre> if a ≤ v ≤ b</pre>
<pre>result = new int[count]</pre>	<pre> i=0</pre>
	<pre> result[i] = v</pre>

Done

0 correct, 0 errors

Start over

Figure 6 – Rearrange Code Interactive Exercises

4. Introduction To Xblocks

4.1. The XBlock Component Architecture

For edX, there are many different kinds of exercises that are provided for instructors to use in their course. Multiple choice, fill in the blank, or true/false are some examples. Not all types of exercises are provided on edX such as interactive exercises that have been described in section 3.2. However, developers can develop a new type of exercise using the edX platform. As mentioned, different learning management systems have different mechanisms for adding a new exercise type. The XBlock architecture is the mechanism that is used by edX.

The XBlock architecture is a component architecture for building online coursework components that go into course content, and that allow that domain experts to author instances of components. An XBlock is a web application fragment that can be aggregated into a larger page. Instead of building an entire web page, an XBlock developer can build a small component that is focused on a particular exercise type.

The LMS container can host many XBlock components in a hierarchical tree. Each of them can provide content in HTML, CSS and JavaScript [11].

4.2. Existing XBlocks

There are a few existing XBlocks in the Open edX platform. Among them are XBlocks for writing different types of problem, embedding video, dealing with chemical formula and symbolic mathematic:

1. The Problem XBlock is developed so the domain expert can add questions like open-ended, multiple choice, numerical, or free text. Students can enter their answers and get corrections [12].
2. The Video XBlock enables developers to add video content, which can be played as a single video or a playlist [12].
3. The Chemtools XBlock is developed for chemistry courses where students can input the shapes of molecules [12].
4. The Symmath XBlock is used to evaluate mathematical expressions [12].

Those XBlocks are for general study, chemistry, and mathematics.

Programs

(2 points possible)

Which is the value of the variables `q` and `r` after running the following code segment?

```
int p = 1;
int q = 2;
int r = 3;
if (r > p + q) {
    r = 2 * p;
}
else if (r < p + q) {
    r = 2 * q;
}
else {
    q = 2 * r;
}
```

Enter the value of `r` :

?

Enter the value of `q` :

?

FINAL CHECK

SAVE

You have used 0 of 1 submissions

Figure 7 - Sample Computer Science online exercise

For computer science courses, there is an XBlock called CodeBoard.io [13], which is a web based IDE to teach programming on edX. Instructors can create exercises in Java, Python or C and link them to the project. Students can view the project and work directly on edX. Codeboard works as a compiler, so students can compile their code as many times as they want to practice.

However, a downside of this XBlock is that instructors have to work in both Codeboard and edX. Instructors have set up and/or edit the project on the Codeboard website. Next, instructors have to integrate the project from Codeboard to edX. Instructors also need go back to the Codeboard website to view students' submission because they can't view students' submission directly on edX. Also, instructors and students must have their own Codeboard accounts in order to work on the project on edX.

Figure 8 - CodeBoard.io XBlock

5. OPEN EDX OVERVIEW

5.1. Open edX

The Open edX platform is a free and open source CMS system that is developed by the edX company. It's used to host MOOCs, small courses, and training modules. There are several options to install Open edX environment, referred to as DevStack, FullStack, Native, Manual, and Bitnami. For this project, I used both DevStack and FullStack. DevStack is used for developers to run and test their code. FullStack is used as a production-like installation for testing [14]. I installed DevStack on my local machine for development and FullStack on an AWS instance for demonstration.

5.2. Devstack

DevStack is a Vagrant instance that works on the local machine. It includes simple components for developers to work on their products: The LMS interface, the studio interface, discussion forums and the ORA component. The LMS interface is a student facing website where students and course teams interact with the course content. The studio interface is a course authoring platform where the course team and instructors author and build courses. Students are not able to access the studio interface. In the discussion forum, students can ask questions and interact with other students or staff members on the LMS. ORA is used for the course team and instructors to collect grades, machine learning grading and staff grading [15].

5.2.1. Installing a DevStack Instance

To install DevStack, we need to install VirtualBox (<https://www.virtualbox.org/wiki/Downloads>), a cross-platform virtualization application, and Vagrant (<https://www.vagrantup.com/downloads.html>), the command line utility for building and distributing the virtual machines. First, you need to ensure the nfsd client is running by using the nfsd status command:

```
sudo nfsd status
nfsd service is enabled
nfsd is running (pid 313, 8 threads)
```

If nfsd is not running, start it:

```
sudo nfsd start
Starting the nfsd service
```

Commands to create the devstack directory is:

```
mkdir devstack
cd devstack
```

Then you can download the vagrantfile and create the vagrant instance:

```
curl -L
https://raw.githubusercontent.com/edX/configuration/master/vagrant/release/devstack/Vagrantfile > Vagrantfile
vagrant plugin install vagrant-vbguest
```

To create and start the devstack virtual machine use command:

```
vagrant up
```

It can take a couple of hours to download a base box. If you delete or destroy the virtual machine, the system will use the base box that you downloaded. It would not take as long as the first time that you downloaded the base box.

5.2.2. Run a DevStack Instance

To connect to the devstack virtual machine, we need to get to the devstack directory and use the SSH command:

```
Vagrant ssh
```

Then connect to the edXapp user:

```
Sudo su edXapp
```

That command will point to the current working directory `/edx/app/edxapp/edx-platform` and if you want to run the LMS use command:

```
paver devstack lms
```

After that, the LMS server runs at: `http://localhost:8000/`. If you want to run edX Studio, use the command:

```
paver devstack studio
```

After that, the Studio server runs at: `http://localhost:8001/`

5.3. FullStack

FullStack is a vagrant instance that works on the server. It is designed for deploying all edX services on one single server. Besides all the DevStack components, FullStack also includes three components: Xqueue, Discern and Ease. Xqueue is a queuing server, which uses RabbitMQ for external graders. Discern is a machine-learning-based automatic textual classification API service. Ease is a library for the cataloging of textual content [16].

5.3.1. Installing a FullStack Instance on AWS

The steps to install FullStack on the local machine are very similar to DevStack that I mentioned in section 5.2.1. For this project, I run FullStack on Amazon Web Services (AWS), so that other users are be able to view and use my XBlocks. First, you need to

have an AWS account and create an EC2 instance — a virtual server in the Amazon Elastic Compute Cloud for running application on AWS. The EC2 server should support Ubuntu 12.04, have a minimum of 4GB memory, one 2.00GHz CPU, and 25GB of free disk. An m3.medium server is recommended [17]. After launching the server and connecting to the server on the terminal, update your Ubuntu packages using the following command:

```
sudo apt-get update -y
sudo apt-get upgrade -y
sudo reboot
```

Use the OPENEDX_RELEASE command to determine the FullStack version you want to run:

```
export OPENEDX_RELEASE="{Git tag}"
# for example: export OPENEDX_RELEASE=named-release/dogwood
```

Run the following command as root via sudo to install FullStack:

```
wget https://raw.githubusercontent.com/edx/configuration/master/util/install/ansible-
bootstrap.sh -O - | sudo bash [17]
wget
https://raw.githubusercontent.com/edx/configuration/$OPENEDX_RELEASE/util/install/sandbox.sh
-O - | bash [17]
```

After that, the installation is finished.

5.3.2. Running a FullStack Instance on AWS

In order to run the LMS on AWS, we need to launch the EC2 instance on the AWS website first. Then, we use the EC2 public DNS address such as `ec2-52-9-161-122.us-west-1.compute.amazonaws.com` or public IP address such as `52.9.161.122` to log in.

In order to run Studio on AWS, we need to add `18010` on the public DNS or public IP address to log in such as `ec2-52-9-161-122.us-west-1.compute.amazonaws.com:18010` or `52.9.161.122:18010`.

6. IMPLEMENTING XBLOCKS

6.1. Creating a Sample XBlock

Before installing the XBlock SDK, you need to make sure that you installed Python, pip, git, Django, and VirtualEnv on your computer. First, you need to create a directory for XBlock development on your computer:

```
mkdir xblock_development
cd xblock_development
```

Next, create and activate the virtual environment:

```
virtualenv venv
$ source venv/bin/activate
```

Then, clone the XBlock SDK from the github repository:

```
git clone https://github.com/edX/xblock-sdk.git
cd xblock-sdk
```

After that, we install XBlock SDK 's requirements:

```
pip install -r requirements/base.txt
```

You only need to clone and install the XBlock SDK once. To create a second XBlock, you can skip these steps and only need to active the virtual environment. To create a new XBlock, go to `xblock_development` directory and run the command:

```
xblock-sdk/bin/workbench-make-xblock
```

It will display the guidelines to create an XBlock. You need to enter the XBlock short name and class name to create it. The short name must be a single word, all lower-case. For example, to create a sample XBlock, we can name it as "survey". The class name is a valid Python class name. It must end with "XBlock". For example, the class for a survey XBlock can be named `SurveyXBlock`. After that, a skeleton of the XBlock is created in the survey directory.

6.2. Running an XBlock on the XBlock SDK

When an XBlock is freshly created, we can view its content on the XBlock SDK server in our local machine. The XBlock SDK server has an interface that displays how the

XBlock will look like on the LMS interface. However, there is no Studio interface for XBlock the SDK server. Using the XBlock SDK server is helpful because it quickly displays the latest version after the source code has changed.

First, we go to the `xblock_development` directory that we created earlier for XBlock and install the new XBlock on SDK server:

```
pip install -e name_of_the_xblock
```

Then we run the following commands to create the database and start the server:

```
python xblock-sdk/manage.py syncdb
python xblock-sdk/manage.py runserver
```

Next, we will see the command to as user to use port 8000 to run XBlock SDK Server:

```
System check identified no issues (0 silenced).
April 12, 2016 - 21:56:55
Django version 1.8.11, using settings 'workbench.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C.
```

After that, we go to the URL `http://127.0.0.1:8000/` to access the server and we can see the new sample XBlock that we just created as shown in Figure 9.

Figure 9 - XBlock SDK Server

6.3. Installing and Running an XBlock on DevStack and FullStack

6.3.1. Installing an XBlock on DevStack

After an XBlock has been implemented, we can install it on DevStack to test it. Turn on the DevStack virtual machine on the devstack directory using these commands:

```
Vagrant up  
Vagrant ssh
```

Then connect to edXapp user:

```
sudo su edXapp
```

Then, our current working directory is `/edx/app/edxapp/edx-platform`. We can copy the XBlock on the current working directory or clone it from the git repository. For example, if I want to put the survey XBlock that I created in section 6.1 to the directory `/edx/app/edxapp/edx-platform/survey`, I can use the `git clone` command to clone the survey XBlock directly from the github repository to the current working directory. After downloading the XBlock, use the following command to install it to DevStack:

```
sudo -u edXapp /edx/bin/pip.edXapp install /path/to/your/xblock
```

6.3.2. Installing XBlock on FullStack

To install the new XBlock, you need to run the following commands:

To get to the directory which stores the new XBlock:

```
cd /edx/app/edxapp
```

Next, download the new XBlock from github:

```
sudo -u edxapp git clone github-link
```

After that, install the new XBlock to FullStack:

```
sudo -u edxapp /edx/bin/pip.edxapp install yourXBlock/
```

In order for the system to receive the new XBlock, we need to restart that system:

```
sudo /edx/bin/supervisorctl -c /edx/etc/supervisord.conf restart edxapp:
```

6.3.3. Running an XBlock on DevStack and FullStack

After an XBlock is installed, you must enable it by doing the following steps:

In `"edx-platform/lms/envs/common.py"`, uncomment `XBLOCK_SELECT_FUNCTION = prefer_xmodules`.

In “edX-platform/cms/envs/common.py”, uncomment `XBLOCK_SELECT_FUNCTION = prefer_xmodules`.

In “edX-platform/cms/envs/common.py”, change ‘`ALLOW_ALL_ADVANCED_COMPONENTS`’ change from `False` to `True`

You only need to do these steps for the first time. Then, you need to go to the Studio interface, add the XBlock on your course’s advanced settings by going to Setting -> Advanced Setting, add the XBlock short name to the `Advanced_Modules` box. For example, the Survey XBlock in section 6.1 will be “survey” when added to the `Advanced_Modules` box. After that, to create the new XBlock, you can find the XBlock in the “advanced” button in your studio interface to add it in your course.

6.4. XBlock Structure

As shown in Figure 10, an XBlock project includes a Python file, an HTML file, a CSS file and a JavaScript file. The Python file has fragments, fields, a view method, a handler method, and workbench scenarios. XBlock fragments are all of the resources that are used to display the XBlock in a web page like HTML, JavaScript, and CSS. XBlock fields are Python attributes, which store user and XBlock states as JSON data. XBlock view methods act as Python methods. To render the XBlock, it is invoked by the XBlock runtime. XBlock handle methods are Python methods to interact between the server side of your XBlock and the viewer side. It receives HTTP requests and returns HTTP responses. Workbench scenario methods provide a quick way to load XBlock instances in scenarios. In the HTML file, all the contents from the HTML files are added to a fragment, which is defined in the Python file. The HTML content can indicate the XBlock fields, which are also defined in the Python file. The JavaScript file is used to manage the interaction between users and the XBlock that are added to the fragment. The CSS file is used to style the XBlock’s display and will also be added to the fragment [18]. When the fragment is returned to the view method in the Python file, the HTML content will be displayed.

Figure 10 - Sample XBlock File Tree

7. The Codecheck XBlock

7.1. Overview

The CodeCheck XBlock is an XBlock component, which manages questions and students feedback from CodeCheck. The CodeCheck XBlock also stores each student's latest submission and displays it on the staff website for professors and staff to view. The CodeCheck XBlock has two views to display: LMS and Studio. The studio view displays an edit box, which allows an exercise author to provide the problem title and the URL of the CodeCheck problem. On the LMS, in the student view, the XBlock shows the problem description, the answer box and a submission button. Students can work on the problem and get instant feedback through this interface. There is also a staff view part on LMS interface as shown in Figure 11. It displays a table with a list of student grades and their code report. It also allows staff or instructors to download all students' submission and information in a CSV file.

Figure 11 – CodeCheck XBlock Staff View on LMS

7.2. Implementation

7.2.1. Studio View

First, I need to create an edit box in Studio so instructors can create and edit the XBlock on an HTML for Studio interface. The edit box takes two fields: problem display name and problem URL. The problem display name field is the exercise name that will be displayed in LMS. The problem URL field is the CodeCheck link, which is the correspond exercise that instructors assign to students.

```
#studio.html
<-- Create Edit box in Studio Editor -->
<ul class="list-input settings-list">
  <li class="field comp-setting-entry is-set">
    <div class="wrapper-comp-setting">
      <label class="label setting-label" for="display_name">Title</label>
      <input class="input setting-input" name="display_name" id="display_name"
value="{self.display_name}" type="text"/>
    </div>
    <span class="tip setting-help">The title of the CodeCheck problem that is displayed to
the user</span>
```

```

</li>
<li class="field comp-setting-entry is-set">
  <div class="wrapper-comp-setting">
    <label class="label setting-label" for="url">Problem URL</label>
    <input class="input setting-input" name="url" id="url" value="{self.url}" type="text"
  />
  </div>
  <span class="tip setting-help">The URL of CodeCheck problem</span>
</li>
</ul>
<div class="xblock-actions">
  <ul>
    <li class="action-item">
      <a href="#" class="button action-primary save-button">Save</a>
    </li>
    <li class="action-item">
      <a href="#" class="button cancel-button">Cancel</a>
    </li>
  </ul>
</div>

```

After creating the edit box, I wrote a view method named `studio_view` in `Cocheck.py` to render the XBlock in the Studio editor and allow instructors to access the edit box by selecting Edit in the Studio as shown in Figure 12.

The screenshot shows a web interface titled "Editing: CodeCheck Problem". It contains a form with two main sections. The first section is labeled "Title" and has a text input field containing "Compute the average length". Below this is a small text description: "The title of the CodeCheck Problem that is displayed to the user". The second section is labeled "Problem URL" and has a text input field containing "http://codecheck.it/codecheck/files?repo=bj4cc". To the right of this field is a small text description: "The URL of CodeCheck Problem". Below the form is a blue link that says "Add problems". At the bottom of the interface are two buttons: "Save" and "Cancel".

Figure 12 – CodeCheck XBlock Studio View

In order to process the information that instructors entered in the edit box from studio interface to display on LMS interface, I implemented a handler method in Python to get the submitting information from `studio_view`. After getting the problem's URL, I fetched the CodeCheck content from the URL and removed the submit button on the CodeCheck web page.

```

def _display_problem_content(self):
    """
    Get codecheck problem's content and removes the 'submit' button
    :return: codecheck's content only
    """

```

```

if self.url:
    res = requests.get(self.url)
    match = self.get_html_body(res.content)
    if match:
        # remove 'submit' button
        new_cont = match.replace('<input type="submit"/>', '')
        return new_cont
    return "No Content Available"

```

Then, I created a view method called `studio_submit` to send the problem content and display name to display on the LMS View.

```

@XBlock.json_handler
def studio_submit(self, data, suffix=''):
    """
    Called when submitting the form in Studio to get the display name and the url.
    """
    self.display_name = data.get('display_name')
    self.url = data.get('url')
    return {'result': 'success'}

```

7.2.2. LMS View

Figure 13 - CodeCheck XBlock Workflow

First of all, as shown in Figure 13, edX gets CodeCheck's content from the CodeCheck website and displays it on the edX interface for students to see. In order to display the content of CodeCheck problem, I have to fetch the content from the edx XBlock backend. After that, I wanted to be able to replace the submit button on CodeCheck with a custom one. The custom submit button allows sending a POST request to the CodeCheck server and controlling the response. After getting the entire problem content

from the handler, I created a JSON template codecheck.html to show the problem content on the student view as shown in Figure 14:

```
#codecheck.html
<div class="codecheck_block">
  <!--display the codecheck problem content-->
  {{ content }}
  <br>
  <!--check if the url works or not-->
  {% if self.url %}
    <div>
      <!--create new Submit button-->
      <button id="submit-issue">Submit</button>
    </div>
    <br>
    <!--display the Codecheck report here-->
    <div id="response"></div>
  {% endif %}
</div>
```

Figure 14 – CodeCheck Student View on LMS

After students submit the problem, I make an AJAX POST request to get the feedback. After students receive their grade and feedback like in Figure 13, I make AJAX calls on

each student's submission to get his/her submission data, then store it in the database. The submitted data includes student id, student username, full name, the latest report, submission time and the actual and maximum score of the problem:

```
'module_id': module.id,
'student_id': s.student_id,
'submission_id': sub['uuid'],
'username': module.student.username,
'fullname': module.student.profile.name,
'timestamp': sub['created_at'].strftime(
    '%Y-%m-%d %H:%M:%S'
),
'score': sub['answer']['score'],
'max_value': sub['answer'].get('max_value'),
'needs_approval': instructor and needs_approval,
```

If the data already was requested once, I do not request it again, but go to the cache to get the data. Moreover, instead of directly going to a new page to display the student's report like on the CodeCheck server, I want to show the report on the same page where students submitted the code. I get the problem response from the CodeCheck server using an AJAX call and display the report below the submit button as in Figure 15.

```
#codecheck.js
function updateResponse(result) {
    // display codecheck problem result
    $('#response', element).html(result.response);
}

// process click on "Submit" button
$(element).find('#submit-issue', element).click(function () {
    var form = $(element).find('form');
    var data = form.serializeArray();
    // send form data to codecheck.it and display the result
    $.ajax({
        type: "POST",
        url: handlerUrl,
        data: JSON.stringify({"form": data}),
        success: updateResponse
    });
});
```

Figure 15 - Display Report on LMS

In order to send a score to the edX grading component, I need both the actual and maximum score of each problem. I used an XBlock handler method to publish a grade event:

```
self.runtime.publish(self, "grade", {'value': self.score, 'max_value': max_value})
```

Students can check their scores for the problem in the process tab where students can view the grades of the entire course. However, it's not easy for staff members or instructors to view all student grades. When staff members or instructors want to get the grade report, which includes student information and grades for all problems, it will take edX up to several hours to download the CSV grade report file. Therefore, I created a table, which shows a list of students who submitted the answer. Each row of the table includes the username, name, last modified time, the score, maximum score of the problem and a view button as shown in Figure 16.

STUDENTS SUBMISSIONS

Username	Name	Last modified	Grade	Max value	Actions
staff	staff	2016-04-06 18:20:55	0	1	View
CayHorstmann	Cay Horstmann	2016-04-06 17:13:24	0	2	View

Figure 16 – CodeCheck XBlock Student submission table on LMS - Staff view

Figure 17 Pop up with student report

The view button pops up the student's latest submission report. The data have been saved in the database inside of the submission model. I made an AJAX call for every student to the backend to get his/her submission data and also used a cache to get the data if the requested was made before. The pop up window is written on codecheck.html using a JSON Template and JavaScript to display the report:

```
#codecheck.html
<!-- popup report script-->
<script type="text/template" id="result-templ">
    <% for(item in report) { %>
        <% var code_string= report[item] %>
        <i><%= item %></i>
        <br>
        <div style="padding-left: 15px;">
            <pre><%= code_string?code_string:'No code provided' %></pre>
    <% } %>
```



```

        </div>
        <hr>
        <% } %>
        <br>
        Score: <%= score %>
    </script>

#codecheck.js
// render popup with student's submission
function renderPopup(answer) {
    var username = {username: answer.username};
    $(element).find('#username')
        .html(unTemplate(username));
    $(element).find('#grade-info')
        .html(template(answer)).height($('html').height()*0.6);
}
// display results
renderPopup(answer);

```

Also, in the edX grade report, the CSV file only includes student information and the grade without the student code. However, the instructor would want to get a copy of the student code for each problem for reviewing purpose. I created a “Save to CSV” link that generates a CSV file for each problem which includes student information, grades, and submission reports. In order to create a CSV file, I grabbed all student information and used the Python CSV module to write the data into a CSV file format:

```

import csv
# modify each file name based on problem's name
file_name = "%s-%s.csv" % (self.display_name, self.report['problem'])
response = Response(
    content_type='text/csv',
    content_disposition='attachment;filename=' + file_name
)
# write the data in dictionary form
writer = csv.DictWriter(response, fieldnames=csv_header)
# write a row with field name
writer.writeheader()
# write all the rows - each row is a student's submission information
for row in data:
    writer.writerow(row)

```

After writing all student data to a CSV file, I allow staff and instructors to download the CSV file to their local computer.

Place through the following code for adding a new element to the beginning of a linked list. Assume the value of the element parameter is Peter.

```

public class LinkedList
{
    ...
    public void addFirst(Object element)
    {
        Node newNode = new Node(); 1 2
        newNode.data = element; 3
        newNode.next = first; 4
        first = newNode; 5
    }
    ...
}

```

Press start to begin.

[Start](#)

[Submit](#)

STUDENTS SUBMISSIONS

Username	Name	Last modified	Grade	Max value
staff	staff	2016-03-16 18:15:59	2	4

Figure 19 - Interactive XBlock Staff View on LMS

8.2. Implementation

8.2.1. Studio View

In the Studio editor, I wrote an HTML script based on JSON Template to create the edit box similar to section 7.2.1.

Editing: Interactive Problem

Title The title of the interactives problem that is displayed to the user

interactives url The url of the interactives problem

[Save](#) [Cancel](#)

Figure 20 - Interactive XBlock Studio View

8.2.2. LMS View

Interaction Interactive Diagram

Figure 21 - Interactive XBlock Workflow

The interactive elements are very complex, and they have many dependencies. The easiest way to display them is in an iframe. However, there is the challenge of retrieving the scores. The interactive elements, unlike CodeCheck, are scored on the client side. Communication with an iframe is very limited. Also, even though there are several different kinds of interactive exercises, and we have to figure out one way of dealing with all of them. In order to get the score from Interactive exercise's iframe, I made a call using `iframe.contentWindow.postMessage("scores", "*")` to send a message to the iframe to get requested data. After that, I will receive the response message that corresponds with includes score value such as the maximum score of the exercise and the score that I get for the exercise.

As shown in Figure 21, I fetched the exercise content using an iframe and created a costumed Submit button to display on LMS view on an HTML file.

```

<div class="interactivities_block">
  <iframe style='width:100%; height: 40em;' id='iframe' src="{self.href}"></iframe>
  <div>
    <button id="submit-issue">Submit</button>
  </div>
  <div id='response'></div>

```

Figure 22 - Interactive XBlock Student View on LMS

Once students finish an exercise, they can submit their answers and get the responses immediately. To get the responses, I used iframe messaging to get the computed score. The response message has four different values: the maximum score of exercise, the correct grade that students get, the number of errors that students made, the activity name of the problem, and the element.

```
#interactive.js
//post message to get the score from iframe
$(element).find('#submit-issue', element).click(function (eventObject) {
    window.addEventListener("message", receiveMessage, false);
    document.getElementById('iframe').contentWindow.postMessage("scores", "*");
    return false;
});
//receive the response message from interactive website
function receiveMessage(event) {
```

```

$.ajax({
    type: "POST", url: handlerUrl, data: JSON.stringify(event.data),
    success: function(result){
        $('#iframe[src*=' + event.data.activity +
    ']').closest('div').find('#response').html(result.response);
        window.removeEventListener('message', receiveMessage, false);
    }
})
}

```

I only display the correct score and the maximum value of the exercise for students to see as seen in Figure 23.

```

# interactive.py
@XBlock.json_handler
def submit_issue(self, data, suffix=''):
    # get student's return data: score, max score, student's id
    value = data.get('correct', 0)
    max_value = data.get("maxscore", 1)
    student_id = self.student_submission_id()
    # store submission in submissions
    submissions_api.create_submission(
        student_id,
        {
            'code': data.get('element'),
            'score': value,
            'max_value': max_value
        }
    )
    # Display the response message
    return {
        'response': "Your result :  %s out of %s correct" % (value, max_value)
    }

```

Figure 23 - Student's grade on LMS

After that, I send the score to Open edX Grading by publishing a grade event using an XBlock handler method in Python as I did in the CodeCheck XBlock. Students can view their grade of the exercise on the Process tab.

For the staff/instructor view on the LMS, I also created a table for them to view all student submissions as shown in Figure 24. The submission table is similar as the table I explained in section 7.2.2 but it does not contain the code view column.

STUDENTS SUBMISSIONS

Username	Name	Last modified	Grade	Max value
staff	staff	2016-04-11 22:01:39	3	4
hongnle	hongle	2016-04-11 22:23:22	4	4

Figure 24 - Interactive XBlock Staff View Table Submission on LMS

This XBlock also allows instructors and staffs to download all student information and grade in CSV file to save on their local computer similar to 7.2.2.

8.3. Third-party tools

The Interactive XBlock does not only work for Wiley Interactive exercises. It can also work for other tools. A third-party tool that wants to use the Interactive XBlock must make instances available with unique URLs. The XBlock will get the exercise content to display on edX from the URL. For the exercise to get the score to display on edX, it needs to reply to a message sent to the iframe. The third-party tool would need to send back the response by calling `window.postMessage`, sending an object of the format: `{correct: correct_value, maxscore: maxscore_value}`.

The Wiley Interactive exercises use a strategy where they set the score and maxscore as attributes on the DOM element. In that case, the message handler has to find the values in the DOM element and send them:

```
function receiveMessage(event){  
  
    var origin = event.origin || event.originalEvent.origin;  
  
    // For Chrome, the origin property is in the event.originalEvent object.  
  
    $(document).find('div').filter(function(i, e) { // Supply a filter for locating the  
interactive element on the page                }).each(function(i, e) {  
var response = { correct: e.correct, maxscore: e.maxscore}  
event.source.postMessage(response, '*'); })}window.addEventListener("message",  
receiveMessage, false);
```

Therefore, a third-party tool has the choice of publishing scores in that way and simply pasting in the generic message handler. After that, the third-party tool can use Interactive XBlock to integrate their tool on edX. The Interactive XBlock architecture makes it possible for third party tools to be supported with a minimum amount of adaptation by the tool provider.

9. Evaluation

In order to measure the utility of adding interactive exercises and programming projects to edX, I surveyed instructors from the San Jose State University – computer science department with experience teaching introductory computer science courses. I also surveyed students from CS046A then indicate the response rates.

9.1. Instructor Responses

1. How do you rate the presentation as a sequence of many short videos with frequent student interactions?

• Very useful	100%
• Somewhat useful	0%
• Not very useful	0%
• Not at all useful	0%
2. This format is intended to have students read the book at home, a section at a time, and to reinforce the reading by engaging students with the material covered in the section. How do you rate the usefulness of this format?

• Very useful	87.5%
• Somewhat useful	12.5%
• Not very useful	0%
• Not at all useful	0%
3. The "code walkthrough" question type in Problem 2 is meant to engage students more deeply than a simple multiple choice question such as Problem 1. How do you rate the usefulness of this question type?

• Very useful	87.5%
• Somewhat useful	12.5%
• Not very useful	0%
• Not at all useful	0%
4. Unlike a web page or an eBook, the edX environment tracks the student progress

on exercises. As instructor, you will be able to monitor which students did the exercises, and how well they did. How do you rate the usefulness of this edX feature?

- Very useful 87.5%
- Somewhat useful 12.5%
- Not very useful 0%
- Not at all useful 0%

5. How do you rate the efficacy of such a worksheet?

- Very useful 100%
- Somewhat useful 0%
- Not very useful 0%
- Not at all useful 0%

6. The edX environment tracks the student progress on worksheets such as this one. As instructor, you will be able to monitor which students used the worksheets, and how well they did. How do you rate the usefulness of this edX feature?

- Very useful 87.5%
- Somewhat useful 12.5%
- Not very useful 0%
- Not at all useful 0%

7. How do you rate the user interface of the auto grader of the edX environment, compared with plain CodeCheck?

- Much better 25%
- Somewhat better 25%
- The same 0%
- Somewhat worse 0%
- Much worse 0%
- I don't know CodeCheck 50%

8. The edX environment stores the student scores and responses. For example, a student can do Part A and later come back to work on Part B. How do you rate the usefulness of this edX feature?

- Very useful 62.5%
- Somewhat useful 37.5%
- Not very useful 0%
- Not at all useful 0%

9. The edX environment enables instructors to download a spreadsheet with the student scores on the homework assignments. How do you rate the usefulness of this edX feature?

- Very useful 87.5%
- Somewhat useful 12.5%
- Not very useful 0%
- Not at all useful 0%

10. Additional comments?

See Appendix A

From the survey's responses, instructors think that the edX interface is very helpful and makes it easy to keep track of students' work process. All of the respondents say that working on many CodeCheck problems on edX is either very helpful or somewhat helpful since edX environment stores the student scores and responses. And 87.5% of the responses agree that the feature to enable instructors to download a spreadsheet with the student scores on the homework assignments is very helpful for them. Overall, the surveyed instructors rate the utility of the features that I implemented in this project very highly.

9.2. Student Responses

1. How many problems did you complete in the worksheets?

Hours	Array minima and maxima worksheet	Finding and removing array elements worksheet	Traversing two-dimensional arrays worksheet
0	0%	7.7%	11.5%
1	1.3%	2.6%	0%
2	5.1%	15.4%	6.4%
3	21.8%	16.7%	23.1%
4	20.5%	17.9%	6.4%
5	51.3%	39.7%	52.6%

2. How much time did you spend approximately with all problems of each worksheet?

Times	Array minima and maxima worksheet	Finding and removing array elements worksheet	Traversing two-dimensional arrays worksheet
Less than an hour	33.3%	46.2%	43.6%
About an hour	37.2%	24.4%	33.3%
About two hours	21.8%	14.1%	12.8%

About three hours	0%	7.7%	6.4%
More than three hours	7.7%	7.7%	3.8%

3. Each worksheet has a progression of problems that starts out with making you understand an algorithm. Then you are asked to construct related program from pre-made lines of code. Finally, you write your own code with CodeCheck. How do you rate the effectiveness of this progression?

- Very useful 37.3%
- Moderately useful 52%
- Slightly useful 10.7%
- Not useful 0%

4. How do you rate the effectiveness of the worksheets overall for reviewing the course material and preparing for the final exam?

- Very useful 58.7%
- Moderately useful 28%
- Slightly useful 13.3%
- Not useful 0%

5. If you were offered additional worksheets of this kind in a programming course that you could work at without receiving course credit, how likely is it that you would use them?

- Very useful 30.7%
- Moderately useful 49.3%
- Slightly useful 12%
- Not useful 8%

6. Additional comments?

See Appendix B

From the survey response, it appears that most students think that the CodeCheck problems on edX are very useful or moderately useful. About 1/2 of the responses said that the interactive problems on edX are very useful for reviewing the course material and preparing for the exam. And 80% of responses are willing to work on interactive problems on edX again in their programming courses. Based on the responses, students mostly spent around an hour per worksheet and worked about 80-100% of the problems. In general, students perceived the worksheets as useful, but somewhat less so than instructors.

10. Challenges and Conclusion

The goal of this project was to implement two XBlocks supporting CodeCheck and interactive exercises. The CodeCheck XBlock displays CodeCheck problems and grades them inside edX. Moreover, the CodeCheck XBlock also displays the student submission table and allows saving all submissions to CSV file in the staff LMS interface. The Interactive XBlock displays interactive exercises and grades them on inside edX. It also has a student submission table on the staff's LMS interface. This project was designed to help instructors create and assign auto-graded problems and interactive problems for students easily on edX. Adding the CodeCheck and Interactive XBlocks were demonstrated to be simple for edX users.

There were many challenges during the time that I have worked on this project. First of all, Open edX is under constant development and is not a bug-free system. During the installation, I have discovered more than 20 bugs in both DevStack and FullStack. Moreover, the company keeps releasing new versions and each new version has different bugs. During the updating time, the system usually crashed and I would lose my data. Then, I had to reinstall the entire system again with a different kind of bugs every time. Next, the XBlock API is fairly new and there isn't much guidance to help in building it. The guide to the XBlock API doesn't explain clearly all the functionalities of XBlock and doesn't include the extended version of XBlock. Since XBlock is new and there aren't many people are using it, you can rarely find the answer that you are looking for online. Most of the time, I have to ask the question on the edX Google group to seek answers from edX developers. Another challenge that I have encountered is storing and creating a pop up window to view the CodeCheck problem report in CodeCheck XBlock to overcome the inefficient report generation In Open edX. I created a grade book table at the staff LMS interface as a quick access method for instructors. However, it was very difficult to understand how to store submission on XBlock because they have not been mentioned in any of XBlock documents. I figured out that I could use CMS to read a student's submission and store the latest submission in the cache by

looking at existing XBlocks. After that, I needed to figure out how to only display the submission table on the staff interface.

In summary, it was a technical achievement to learn how to run edX instances, understand the extension mechanism, and successfully integrate new XBlocks to edX.

This project will be helpful for instructors who want to run computer science online courses in edX or use edX to provide supplementary course materials. Instructors will be able to assign interactive exercises and auto-graded programming problems on edX easily and view students' works quickly.

APPENDIX A

This Appendix shows the instructor's comments about the edx interactive exercises.

1. In videos, don't use a black stylus; black text nearly vanishes when overlaid over this stylus. I don't know how appropriate this tech would be for 46B. But I would be very happy if my 46B students came out of 46A having trained on this platform. I really like the feature in the videos where the speaker starts to write a word or phrase, and then the video skips forward to the point where the word/phrase has completed.
2. I was really confused at first. I did not notice the right arrow to continue. Once I figures that out, everything went smoothly Does the instructor have a way of seeing all the submission in one place as we currently do so comment can be made for student improvement?
3. Line numbers in the worksheets would be helpful. Also, the code wrapped a little awkwardly in come cases. Making the text box a little bigger might help. Neat idea. Looking forward to hearing more about it.
4. I think there was bug on Cay + 1 > Sara problem. I've never used the system of turning in an intermediate stage of an assignment, so I'm not sure how useful this is. I would think that it would be less bother to simply complete the assignment early and get some extra credit than to come back to a partial solution 2 days later. But I haven't tried either technique so I don't know. I thought the exercise of arranging code using drag-and-drop was very cool. Lastly, I wonder about the slow pace of progress. I get impatient watching videos and would rather read.

APPENDIX B

This Appendix shows the student's comments about the edx interactive exercises.

1. There should be more practice problems like these to compel students to think and learn more. Great Incentive!
2. May be making the text box thing bigger. It's hard to scroll and look at everything when it's so small
3. The instructions or the way it works is pretty confusing.
4. I really like the algorithm/trace problems here. I think the interactivity makes it work really well because it's real-time guidance on how it works. I don't like the drag and drop ones. I personally feel that type of problem is very hard to understand for beginners, and that was one of the types of problems I hated in the Udacity videos. Drag and drop doesn't work so well with the computers and the { } brackets are confusing. Overall, I think this is useful and presents the material for newbies a lot better than some of the exercises in our current course curriculum.
5. More different algorithms for arrays and array lists.
6. The drag and drop questions were a little hard to interact with while using a laptop touch pad.
7. The typing program for the code was kind of annoying, it should follow be something like from Udacity. The drag and drop was also pretty annoying, I think it should just be prompts for what comes next.
8. More examples of arrays and array lists.
9. Thank for the extra credit!

10. It was very cool, and very fun, made me question myself a bit, and go back to review what I didn't know. Also, having things that go before Array's, like loops, ifs and else's would be cool too
11. Wish it covered more concepts! It's great! Honestly, I enjoy how I'm introduced to how an algorithm works before I start doing problems using that concept. It really helps seeing what your strengths and weaknesses are before an exam. I would definitely wish that computer science students had these before exams so that you can see what you need to review in instead of having to only read the book and redo past homework assignments. This is great practice and I would definitely say it was well done. The problems where you trace code or rearrange code to create a method were a little bit of a learning curve since I wasn't used to the format but once I understood it, it was great practice!
12. Good job on this. Problems were very useful and thought-provoking.
13. Good Work. Was a bit difficult at first how to use the worksheet but figured that out. Nicely done.
14. Great Practice! Looking forward for more.
15. I really like the concept of having an example and then providing a problem that puts it to practice. One thing that could improve is elaboration. Even with the directions giving the section number in the book for more information on how to program a certain algorithm, sometimes it didn't help. I'd like more information to elaborate on the problem and what steps I can take to successfully code the algorithm correctly.
16. I liked how you referenced a section from the CS textbook for reference. Overall, the problems were pretty challenging and great practice for the exam
17. In the second section, one of the worksheets has a glitch. Even though the answer is correct, it wouldn't let me pass until I put in another line that was not supposed to be there. Though I'm not sure if this is a mistake or I just missed

something. But these were really good! It must have taken a lot of work and time to put these together. They are really well done. Thank you so much!

18. This was very helpful, thank you!

19. Thanks!

20. The worksheet was well made and helps you step by step through the code.
Thank You.

Reference

- [1] Stephen Colbran and Gilding Anthony, "MOOCs and the Rise of Online Legal Education," *Journal of Legal Education*, vol. 63, no. 3, pp. 405–428, Feb. 2014.
- [2] Bryant Nielson. (2015, May) Your Training Edge. [Online]. <http://www.yourtrainingedge.com/whats-the-difference-between-a-mooc-and-an-lms/>
- [3] (2013, Apr.) The Good MOOC. [Online]. <http://www.thegoodmooc.com/2013/04/a-review-of-udacity.html>
- [4] Phil Hill. (2016, Apr.) e-Literate. [Online]. <http://mfeldstein.com/state-higher-ed-lms-market-spring-2016/>
- [5] Marija Blagojević and Danijela Milošević, "Massive Open Online Courses: edX vs Moodle MOOC," Technical Sciences Čačak, University of Kragujevac, Serbia, 2015. [Online]. <http://www.baektel.eu/documents/conferences/Blagojevic%20Milosevic%20ICIST%20final.pdf>
- [6] (2014, Oct.) moodle. [Online]. https://docs.moodle.org/dev/Question_types
- [7] Ims Global. [Online]. <https://www.imsglobal.org/activity/learning-tools-interoperability>
- [8] edX. [Online]. <https://open.edx.org/about-open-edx>
- [9] Cay Horstmann. Codecheck. [Online]. <http://horstmann.com/Codecheck/>
- [10] David Taylor et al., in *SIGCSE Conference 2009*, 2009.
- [11] Srinu Kolukuluri, "Indian Institute of Technology," Department of Computer Science and Engineering, Bombay, R&D Report 2014.
- [12] Admin. (2014) Extension Engine. [Online]. <http://blog.extensionengine.com/what-is-a-xblock/>
- [13] Chair of Software Engineering at ETH Zurich. Codeboard. [Online]. <https://codeboard.io>
- [14] Fred Smith. (2016, Mar.) Open edX Installation Options. [Online]. <https://openedx.atlassian.net/wiki/display/OpenOPS/Open+edX+Installation+Options>
- [15] Edx. (2015) Installing, Configuring, and Running the Open edX Platform. [Online]. <http://edx.readthedocs.org/projects/edx-installing-configuring-and-running/en/latest/>
- [16] (2016) Open edX Platform Installation Options. [Online]. https://github.com/edx/edx-documentation/blob/master/en_us/install_operations/source/installation/installation_options.rst

- [17] Fred Smith. (4, 2016) Atlassian. [Online].
<https://openedx.atlassian.net/wiki/display/OpenOPS/Native+Open+edX+Ubuntu+12.04+64+bit+Installation>
- [18] (2015) Edx. [Online]. <http://edx.readthedocs.org/projects/xblock-tutorial/en/latest/index.html#>
- [19] Cay S. Horstmann, *Big Java*. United States: Wiley, John & Sons, 2012.
- [20] Ned Batchelder. (2015, Oct.) github. [Online]. <https://github.com/edx/edx-platform/wiki/Installing-a-new-XBlock>
- [21] Cay Horstmann. InterActivities. [Online]. <http://horstmann/private/apps/worksheet.html>