Iyer, S. et al. (Eds.) (2022). Proceedings of the 30th International Conference on Computers in Education.
Asia-Pacific Society for Computers in Education

The ABC Workbook: Adapting online judge systems for introductory programming classes

Aldrich Ellis ASUNCION[a*], Brian Christopher GUADALUPE[b] & Gerard Francis ORTEGA[c]
[a]*Department of Mathematics, Ateneo de Manila University, Philippines*
[b]*Department of Information Systems and Computer Science, Ateneo de Manila University, Philippines*
[c]*Scientific Committee, National Olympiad in Informatics – Philippines, Philippines*
*aeasuncion@ateneo.edu

Abstract: The online judge is an established method of automatic assessment for programming tasks. Recognizing the relative lack of problems and resources for introductory-level programming classes, we present the ABC Workbook project, a workbook containing a comparatively large selection of basic programming exercises, accompanied by an application written to check the exercises in the workbook. The contents of the workbook, an overview of the app, and strategies for integration in a class are discussed. The accessibility of the app for use in remote and developing areas is emphasized. The project is expected to improve introductory-level programming instruction, by facilitating immediate feedback and providing access to a bank of curated problems. Future work involves application in classrooms, as well as implementation of problem creation features.

Keywords: Computer science education, automated assessment, textbooks, introductory programming.

1. Introduction

Introductory programming classes rely on the students' ability to receive feedback for their work. These classes have two primary learning objectives: programming knowledge, via the syntax and semantics of the course's chosen language; and problem-solving strategies (Malik and Coldwell-Neilson 2017). These objectives are reinforced by assigning practice exercises to students, usually which require them to write a simple program which can perform a desired task.

However, the effectiveness of these exercises relies on the ability of an external entity (e.g. a teacher, a software program) to give feedback to the students. A novice programmer cannot just self-evaluate that their code “seems correct.” Practice in algorithmic thinking requires stricter supervision because correctness and efficiency may not be apparent to a beginner. For example, the student might be overlooking an edge case, or their logic may contain a faulty argument that had gone unnoticed.

As a resource for such exercises, traditional textbooks are lacking in two regards. First, traditional textbooks focus heavily on content delivery of programming knowledge, and not as much on exercises and problem-solving skills (Demilie 2020). Second, even if a traditional textbook had many exercises, it still could not be effectively self-studied, as an instructor is required in order to give feedback. Even with an instructor, the workload of checking the code submissions of each student for each exercise scales poorly to upwards of hundreds of students, which may lead to the instructor not assigning too many programming exercises in their classes, which is also not ideal.

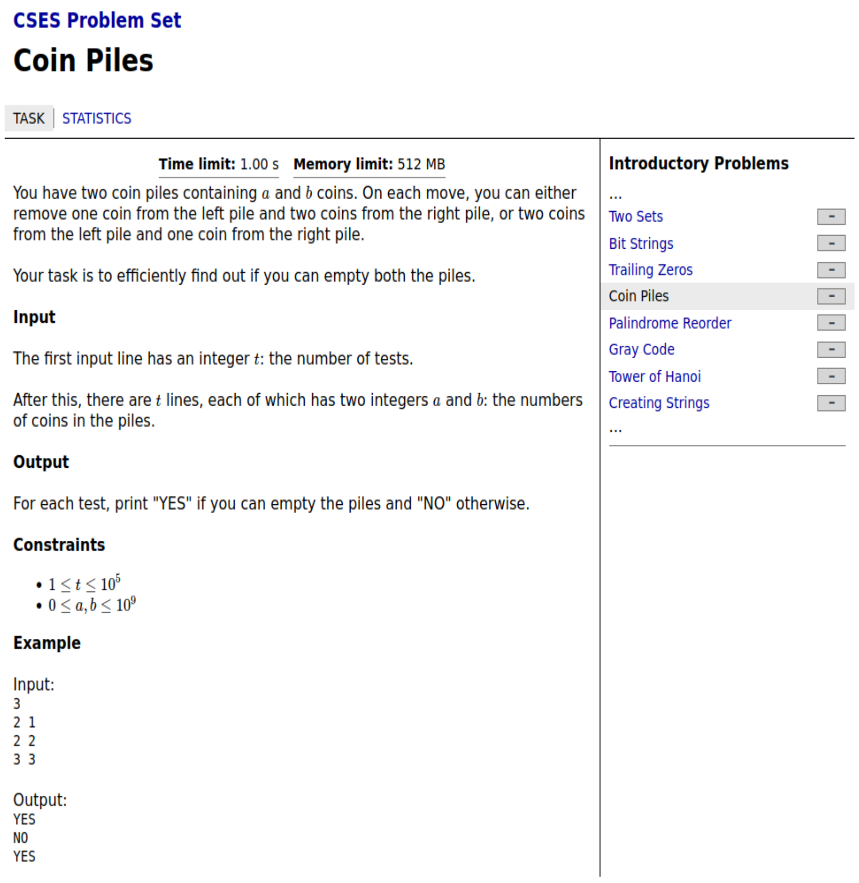
With the shift to e-learning, there is a clear need for a resource which is capable of giving automated feedback to programming students. Fortunately, such a system already exists and has been used by the competitive programming community for more than two decades now: the online judge. Some well-known examples of online judges are: Codeforces, UVa Online Judge[1], and CSES.

1.1 The Online Judge

[1] Since the passing of the site's maintainer, Miguel Revilla, the site is now simply called Online Judge, as it is no longer associated with the University of Valladolid.

The online judge is an automated assessment tool. They have been successfully used as an effective assessment tool for data structures and algorithms courses (Enstrom et al. 2011; Garcia-Mateos and Fernandez-Aleman 2009). CSES is itself used in facilitating an algorithmic programming course in the University of Helsinki (Laaksonen and Talvitie 2020).

Each task on an online judge typically involves the creation of a program whose specifics are laid out in the scenario given in a problem statement. As an example, in Figure 1, we show the problem "Coin Piles" from the online judge CSES. The desired behavior of the program is stated using generic parameters. In Figure 1, we see that the coin piles' sizes are given by variables a and b, and the number of test cases is also some variable t. The contestant's program is given, as input, particular values of e.g. t and each a and b, representing some concrete instance of the input parameters. The program should then output the correct answer for this instance of the problem.


CSES Problem Set

Coin Piles

TASK | STATISTICS

Time limit: 1.00 s **Memory limit:** 512 MB

You have two coin piles containing a and b coins. On each move, you can either remove one coin from the left pile and two coins from the right pile, or two coins from the left pile and one coin from the right pile.

Your task is to efficiently find out if you can empty both the piles.

Input

The first input line has an integer t: the number of tests.

After this, there are t lines, each of which has two integers a and b: the numbers of coins in the piles.

Output

For each test, print "YES" if you can empty the piles and "NO" otherwise.

Constraints

- $1 \le t \le 10^5$
- $0 \le a, b \le 10^9$

Example

Input:
```
3
2 1
2 2
3 3
```

Output:
```
YES
NO
YES
```

Figure 1. Sample task "Coin Piles" from the online judge CSES

The problem author prepares several test files. To test the correctness of the submitted code, it is run with each of these test files as input, and a judge program automatically checks whether the contestant's program produces the correct output for each one. The author writes a model solution to the problem, and the judge checks that the output of the submission exactly matches the output of the model solution. For tasks with multiple possible answers, the problem author may instead write a custom checker program which verifies the correctness of the contestant's output. A time limit and memory limit are also given in the problem statement, and if a submission consumes more time or resources than is allotted to them, it is also marked incorrect.

Constraints are given which limit the kinds of values which may appear in the input. Strict formats on input and output are imposed so that the feedback process can be automated by the online judge. Assuming that the problem author's test data is robust in handling all the different cases and detecting incorrect solutions, a student is able to automatically receive feedback on the correctness of the programs they write through an online judge.

1.2 Issues with Online Judges

There are a handful of factors that limit the accessibility of existing online judges for use in introductory programming classes. First, problems in existing online judges were not designed to be exercises for introductory programming classes. Sites like Codeforces and UVa are archives of ACM ICPC-style contest problems, and so even their "easy" problems may require finding mathematical insights that are outside the scope of the class, or require concepts that would appear in a data structures and algorithms course, not in an introductory programming course.

Second, it is not easy for an instructor to create and upload their own problems to many online judges, if it is possible at all. There is a nontrivial technical barrier in authoring a problem using Codeforces' problem-creation platform, and in giving students access to these custom problems. Administrator access is required to upload problems to UVa and CSES.

Third, all of these online judges are dependent on the student having a stable and persistent internet connection. This makes these platforms inaccessible to students in remote or developing areas without a stable internet connection.

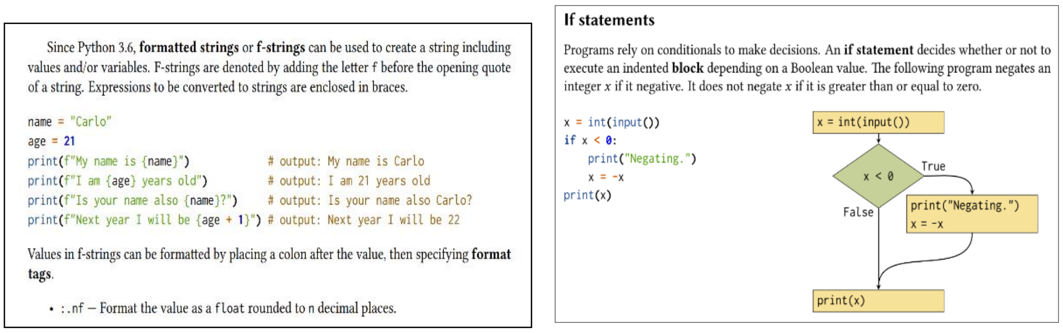
2. The ABC Workbook Project

The ABC Workbook project is intended to address the issues we have identified in applying online judges to an introductory Python programming class. The project is comprised of two parts: the *ABC Workbook* and *AutoJudge++*.

2.1 The ABC Workbook

The ABC Workbook is a problem bank for introductory Python programming, intended to supplement an introductory programming or computer science course by providing a curated collection of exercises. It is planned to contain exercises under the following topic areas: variables, data types, input and output, conditionals, loops and nesting, built-in data structures, functions, recursion, classes, and basic design patterns. These topic areas align with most of the topics in the Software Development Fundamentals (SDF) knowledge area in the ACM/IEEE curriculum guidelines for computer science, which characterize many introductory computer science sequences (ACM Computing Curricula Task Force 2013). As a result, the workbook does not assume any prior programming knowledge.

Each section of the workbook begins with a summary of Python syntax and constructs, accompanied with examples (Figure 2). This discussion is largely adapted from the official Python tutorials and language specification but with additional examples used to illustrate common usage patterns of programming constructs. The discussion is short, as most of the book's content is deferred to the exercises.



Since Python 3.6, **formatted strings** or **f-strings** can be used to create a string including values and/or variables. F-strings are denoted by adding the letter `f` before the opening quote of a string. Expressions to be converted to strings are enclosed in braces.

```
name = "Carlo"
age = 21
print(f"My name is {name}")            # output: My name is Carlo
print(f"I am {age} years old")         # output: I am 21 years old
print(f"Is your name also {name}?")    # output: Is your name also Carlo?
print(f"Next year I will be {age + 1}") # output: Next year I will be 22
```

Values in f-strings can be formatted by placing a colon after the value, then specifying **format tags**.

- `:.nf` – Format the value as a `float` rounded to `n` decimal places.

If statements

Programs rely on conditionals to make decisions. An **if statement** decides whether or not to execute an indented **block** depending on a Boolean value. The following program negates an integer *x* if it negative. It does not negate *x* if it is greater than or equal to zero.

```
x = int(input())
if x < 0:
    print("Negating.")
    x = -x
print(x)
```

Figure 2. Samples from the explanatory portions of the ABC Workbook.

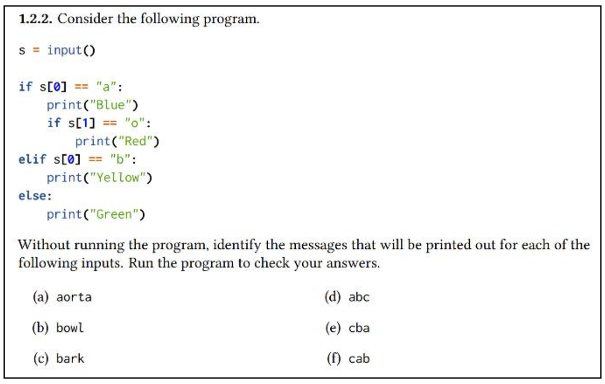
The discussion is followed by a collection of practice exercises. While a handful of the exercises are conceptual (Figure 3a), majority of the exercises are programming tasks, formatted similarly to those found on online judges (Figure 3b). Each exercise is written so that answers may be easily checked by the reader, either through an answer key, or using the accompanying AutoJudge++ application. This ensures that the workbook is suitable for self-study and remote use.

Compared to a number of introductory programming textbooks, the ABC Workbook contains a large quantity of problems. The book contains over 80 problems devoted to the use of basic data types, variables, and conditionals, while over 140 problems are devoted to iteration and lists. The full book is estimated to have at least 500 exercises. Table 1 provides a comparison with various introductory programming books.

Table 1. *Approximate Number of Exercises of Various Introductory Programming Books*

Book	Exercises (approximate)
Think Python, 2nd Edition (Downey 2016)	70
Introduction to Programming in Python (Sedgewick et al. 2015)	160
Python Programming: An Introduction to Computer Science, 3rd Edition (Zelle 2017).	180
C How to Program, 8th Edition (P. J. Deitel and H. M. Deitel 2016)	200
Big Java: Early Objects, 7th Edition	250

While the volume of exercises in the ABC Workbook may seem large in comparison, this is not unusual compared to many online judges. CSES has over 200 problems and is planned to contain 1000 problems (Laaksonen and Talvitie 2020). The ABC Workbook is comparable to a typical precalculus or calculus textbook, which often contain a similarly large number of basic exercises.



1.2.2. Consider the following program.

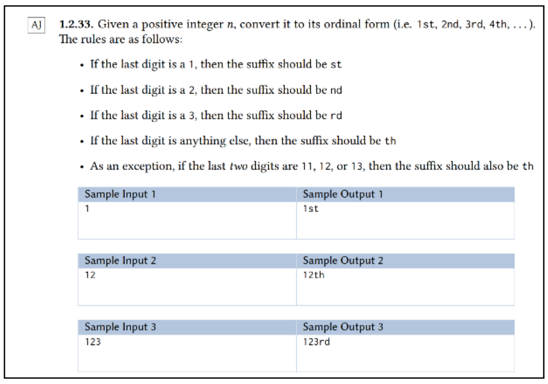
```python
s = input()

if s[0] == "a":
    print("Blue")
    if s[1] == "o":
        print("Red")
elif s[0] == "b":
    print("Yellow")
else:
    print("Green")
```

Without running the program, identify the messages that will be printed out for each of the following inputs. Run the program to check your answers.

(a) aorta
(b) bowl
(c) bark
(d) abc
(e) cba
(f) cab

(a)



AJ **1.2.33.** Given a positive integer *n*, convert it to its ordinal form (i.e. 1st, 2nd, 3rd, 4th, . . .). The rules are as follows:

- If the last digit is a 1, then the suffix should be st
- If the last digit is a 2, then the suffix should be nd
- If the last digit is a 3, then the suffix should be rd
- If the last digit is anything else, then the suffix should be th
- As an exception, if the last *two* digits are 11, 12, or 13, then the suffix should also be th

Sample Input 1	Sample Output 1
1	1st

Sample Input 2	Sample Output 2
12	12th

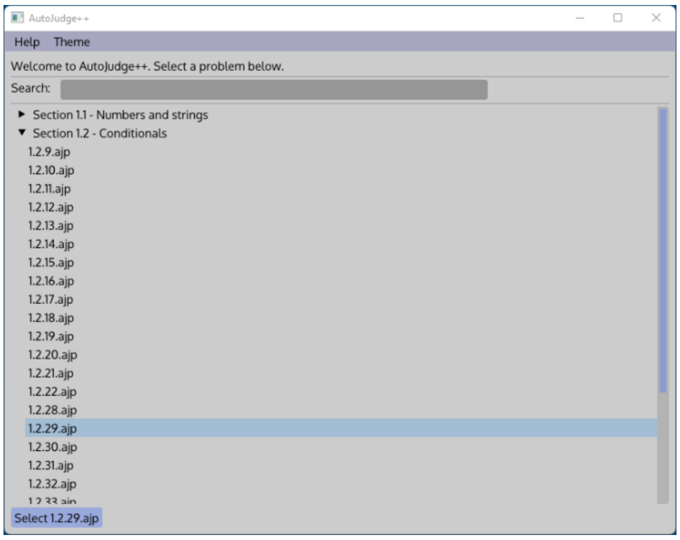
Sample Input 3	Sample Output 3
123	123rd

(b)

Figure 3. Sample exercises from the ABC Workbook

2.2 AutoJudge++

Accompanying the ABC Workbook is a desktop application, AutoJudge++. Like an online judge, it allows for the automated checking of programs. The application is preloaded with most of the programming problems listed in the ABC Workbook, allowing those using the workbook to check their answers using the application. Unlike online judges, AutoJudge++ is built to function completely offline, running on the user's computer instead of an online judge server. This allows those without a stable persistent internet connection to use the app unimpeded.



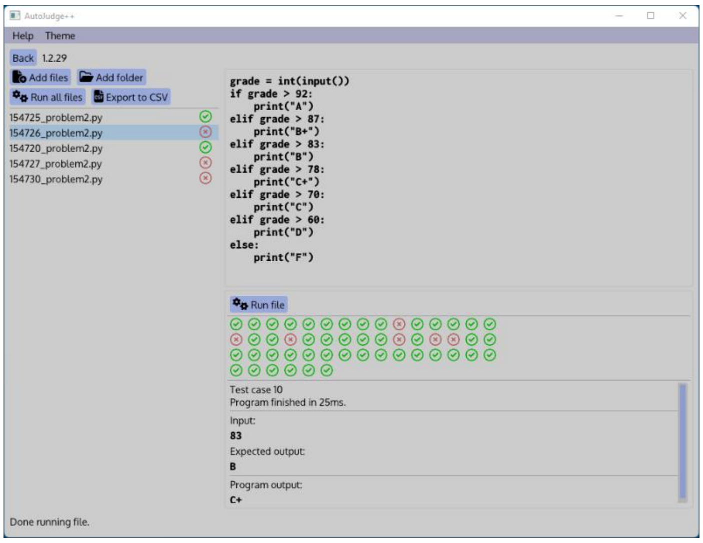
Welcome to AutoJudge++. Select a problem below.

- Section 1.1 - Numbers and strings
- Section 1.2 - Conditionals
  - 1.2.9.ajp
  - 1.2.10.ajp
  - 1.2.11.ajp
  - 1.2.12.ajp
  - 1.2.13.ajp
  - 1.2.14.ajp
  - 1.2.15.ajp
  - 1.2.16.ajp
  - 1.2.17.ajp
  - 1.2.18.ajp
  - 1.2.19.ajp
  - 1.2.20.ajp
  - 1.2.21.ajp
  - 1.2.22.ajp
  - 1.2.28.ajp
  - 1.2.29.ajp
  - 1.2.30.ajp
  - 1.2.31.ajp
  - 1.2.32.ajp

Figure 4. The problem selection menu of AutoJudge++

Upon loading the application, the user is presented with a list of problems, corresponding to marked problems in the ABC Workbook (Figure 4). Selecting a problem allows the user to then add Python file submissions. Students using the app can add a Python file containing their solutions, while teachers using the app can add a directory of Python files.

Each problem in AutoJudge++ is set up with a number of test files, covering a variety of possible inputs and edge cases depending on the problem. The app can then run the Python submissions on each test input, and automatically check if the output of the submission exactly matches the expected output. These results are summarized in a visual display (Figure 5). The user can inspect individual test cases to view the execution time, the input for the test case, the expected output, and the submission's output. Teachers using the app may also export a CSV summary of the verdicts for all submissions to a problem, which they can then incorporate into their own workflow for checking assessments.



154725_problem2.py ✓
154726_problem2.py ⊗
154720_problem2.py ✓
154727_problem2.py ⊗
154730_problem2.py ⊗

```
grade = int(input())
if grade > 92:
    print("A")
elif grade > 87:
    print("B+")
elif grade > 83:
    print("B")
elif grade > 78:
    print("C+")
elif grade > 70:
    print("C")
elif grade > 60:
    print("D")
else:
    print("F")
```

```
Test case 10
Program finished in 25ms.
Input:
83
Expected output:
B
Program output:
C+
```

Done running file.

Figure 5. A submission for a problem, checked using AutoJudge++. The panel on the left shows all submissions. The panel at the bottom displays details for each test case for the current submission.

2.3 Integration and use

As the ABC Workbook project is a resource which simply contains a problem bank and automatic assessment tool, without any online dependence or defined submission workflow, it can be readily adapted for use in existing assessment workflows. It is recommended to combine both automated and manual assessment systems, with the automatic system providing ease-of-use and efficiency to the assessment process (Ihantola et al. 2010). We present several ideas adapted from existing literature here.

The AutoJudge++ application, like online judge systems used in education today, may be used with *test-driven education* (Enstrom et al. 2011). Teachers assigning programming tasks can ask their students to have their solutions checked by AutoJudge++ before presenting their solutions to the teacher. In this approach, students are allowed to have their program checked by the app any number of times, with the app serving as an adversary. This form of feedback encourages students to develop their programs based on concrete tests. When students present their program to the teacher, the teacher can focus on higher levels of assessment, by asking students to explain their understanding of their code, as well how they addressed issues during program development.

The test-driven education approach lends itself well to approaches such as *pair programming* (Waite and Sentance 2021). In pair programming, two people work simultaneously on a single project, with one person actively using the computer, while the other reviews the work done by their collaborator and offers suggestions. As students are able receive immediate, independent feedback through the AutoJudge++ app, students are constantly confronted with problems mid-development as they try to pass all test cases, sparking discussion and activity among the students. This is an integral part of the teamwork involved in the competitive programming environments, and this can be adapted for use in introductory programming education.

As we have established that existing introductory textbooks contain relatively few programming exercises, the ABC Workbook project may be used as a problem bank for use in both assignments and exams, with the AutoJudge++ app serving to manage the workload of assessing many submissions for many exercises. Students may also use it as a self-study resource, due to the feedback provided by the project and the lack of dependence on online connectivity. The ABC Workbook may be distributed as a PDF file, while the AutoJudge++ app may be distributed as a zip file with no installation required.

3. Conclusion and Recommendations

In this paper, we presented the ABC Workbook project, a workbook and companion app for introductory Python programming. We demonstrated how these online judges can be adapted for an introductory resource. Through a brief survey of textbooks, we also established the need for larger volumes of exercises, to give students more opportunities to learn by practicing programming directly. Lastly, we presented methods by which these resources may be integrated in the classroom. We hope that this paper inspires related resource creation for introductory programming classes and furthers interest in designing accessible automatic assessment methods for programming assignments.

The ABC Workbook project is currently still under development. We are currently designing additional features for the AutoJudge++ application to facilitate easier *problem creation*. The goal is to allow teachers to easily create problems which can be checked by AutoJudge++, even without prior contest problem-setting experience. Further development can also focus on the security of the application, to ensure submissions cannot maliciously affect the host computer (Ihantola et al. 2010). Further work on this project may also involve a detailed examination of exercises in recent introductory textbooks and workbooks, as well as an investigation into the effectiveness of applying the ABC Workbook project in the classroom.

References

ACM Computing Curricula Task Force (Ed.). (2013). *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science.* ACM, Inc. https://doi.org/10.1145/2534860

Deitel, P. J., & Deitel, H. M. (2016). *C How to Program* (8th ed.). Pearson.

Demilie, W. B. (2020). Why University Students Fail in Most Computer Programming Courses: The Case of Wachemo University-Student-Teacher Perspective. *Computer Engineering and Intelligent Systems*. https://doi.org/10.7176/CEIS/11-2-02

Downey, A. (2016). *Think Python* (2nd ed.). O'Reilly Media.

Enstrom, E., Kreitz, G., Niemela, F., Soderman, P., & Kann, V. (2011). Five years with Kattis – Using an automated assessment system in teaching. *2011 Frontiers in Education Conference (FIE)*, T3J–1–T3J–6. https://doi.org/10.1109/FIE.2011.6142931

Garcia-Mateos, G., & Fernandez-Aleman, J. L. (2009). A course on algorithms and data structures using on-line judging. *Proceedings of the 14th annual ACM SIGCSE conference on Innovation and technology in computer science education - ITiCSE '09*, 45. https://doi.org/10.1145/1562877.1562897

Guttag, J. (2021). *Introduction to Computation and Programming Using Python* (3rd ed.). MIT Press.

Horstmann, C. S. (2018). *Big Java: Early Objects* (7th). John Wiley & Sons, Inc.

Ihantola, P., Ahoniemi, T., Karavirta, V., & Seppälä, O. (2010). Review of recent systems for automatic assessment of programming assignments. *Proceedings of the 10th Koli Calling International Conference on Computing Education Research - Koli Calling '10, 86–93.* https://doi.org/10.1145/1930464.1930480

Laaksonen, A., & Talvitie, T. (2020). CSES – Yet Another Online Judge. *Olympiads in Informatics,* 105–111. https://doi.org/10.15388/ioi.2020.08

Malik, S. I., & Coldwell-Neilson, J. (2017). A model for teaching an introductory programming course using ADRI. *Education and Information Technologies*, 22(3), 1089–1120. https://doi.org/10.1007/s10639-016-9474-0

Sedgewick, R., Wayne, K. D., & Dondero, R. (2015). *Introduction to Programming in Python: An Interdisciplinary Approach*. Addison-Wesley.

Waite, J., & Sentance, S. (2021). *Teaching programming in schools: A review of approaches and strategies* (tech. rep.). Raspberry Pi Foundation.

Zelle, J. M. (2017). *Python Programming: An Introduction to Computer Science* (3rd ed.). Franklin, Beedle & Associates Inc.