



Norwegian University of
Science and Technology

Learning How to Program With a Self-Evaluation System

A Study on Motivational Aspects and Learning
Effect

Fredrik Christoffer Berg
Håkon Ødegård Løvdal

Master of Science in Informatics

Submission date: May 2017

Supervisor: Guttorm Sindre, IDI

Norwegian University of Science and Technology
Department of Computer Science

Abstract

Learning how to program is difficult for many students, as it separates itself from many of the traditional technological fields like mathematics, chemistry, and physics. Additionally, many students have never programmed before and have no prior experience in the field. That poses challenges when designing introductory courses in programming. To avoid having students dropping out of these courses, it is important that professors have the tools necessary to motivate students. This report describes the work done as a master's thesis in Informatics from August 2016 to June 2017. It consists of research done on motivational factors as well as learning outcome with a self-evaluation system for Python programming. The initial version of the system was developed over the course three months. Upon completion, it was used to generate both qualitative and quantitative data on its effects. The system was distributed to first-year students taking *TDT4110 - Information Technology - Introduction* at NTNU as a voluntary addition towards the end of the semester. Motivational factors were measured through questionnaires as well as analysis of usage patterns. Learning effect was studied through an experiment with 56 students, where some had prior programming experience with MatLab, while others had no experience with programming.

Sammendrag

Å lære seg å programmere er vanskelig for mange studenter, da det skiller seg fra flere av de tradisjonelle teknologiske feltene som for eksempel matematikk, kjemi og fysikk. Videre er det mange studenter som ikke har erfaring med programmering da de begynner å studere. Dette åpner for utfordringer når det kommer til å utforme og planlegge introduksjonskurs innen programmering. Derfor blir det vanskelig for foreleserne å tilby nødvendige verktøy og materiell for å redusere sannsynligheten for at studenter dropper ut av kurset. Denne oppgaven beskriver arbeidet som ble utført i forbindelse med en masteroppgave i informatikk fra august 2016 til juni 2017. Oppgaven inneholder en detaljert beskrivelse av forskningen som ble utført på læringseffekten og de motiverende faktorene ved bruk av et selvevalueringssystem for Python-programmering. Den initielle utgaven av systemet ble utviklet i løpet av tre måneder. Når systemet var ferdig, ble det brukt til å produsere kvantitative og kvalitative data om effektene av systemet. Systemet ble distribuert til førsteårsstudenter som var oppmeldt i faget *TDT4110 - Informasjonsteknologi, grunnkurs* ved NTNU. Det ble distribuert som et frivillig tillegg til eksamenslesningen på slutten av semesteret. Motiverende faktorer ble målt ved hjelp av spørreundersøkelser med tilbakemeldinger fra brukere, i tillegg til en analyse av bruksmønstre. Læringseffekten ble målt gjennom et eksperiment med 56 deltagende studenter, som hadde litt eller ingen tidligere programmeringserfaring.

Preface

This thesis and its associated source code is the result of the research conducted from the fall of 2016 to the spring of 2017. It concludes our degree in Master of Science at the Norwegian University of Science and Technology.

We would like to extend our sincere gratitude to everyone who contributed to this thesis, including those who proofread it. Especially Prof. Guttorm Sindre, our supervisor, who followed our work in close cooperation throughout the whole period. Additionally, we would like to thank the Department of Computer Science at the Norwegian University of Science and Technology for funding our experiment. Finally, we would like to thank everyone who participated the experiment, without whom it would not have been possible to complete our work.

Fredrik Christoffer Berg and Håkon Ødegård Løvdal
Trondheim, May 29, 2017

Table of Contents

Abstract	i
Sammendrag	iii
Preface	v
Table of Contents	x
List of Tables	xi
List of Figures	xiv
Abbreviations	xv
1 Introduction	1
1.1 Background	2
1.1.1 Motivation	3
1.2 Problem Description	3
1.3 Nygårds Master's Thesis	4
1.4 Related Work	5

1.4.1	Ways of Experiencing the Act of Learning to Program: A Phenomenographic Study of Introductory Programming Students at University	6
1.4.2	Five Years with Kattis - Using an Automated Assessment System in Teaching	7
1.4.3	Individualized Exercises for Self-Assessment of Programming Knowledge: An Evaluation of QuizPACK	7
1.4.4	Experimental Validation of the Learning Effect for a Pedagogical Game on Computer Fundamentals	8
1.4.5	Similar Software	9
1.5	Report Structure	10
2	Method	11
2.1	Research Strategies	11
2.1.1	Research Strategy for Question One	12
2.1.2	Research Strategy for Question Two	14
2.2	Data Collection for Research Question One	15
2.2.1	Qualitative	15
2.2.2	Quantitative	16
2.3	Data Collection for Research Question Two	17
2.4	Bias and Data Validity	19
2.4.1	Social Desirability Bias	19
2.4.2	Acquiescence Bias and Demand Characteristics	20
2.4.3	Non-Response Bias	20
2.4.4	Missing Data With Google Analytics	21
2.4.5	Human Involvement in Experiments	21
2.5	Ethical Issues	22
3	Test and Experiment Design	25
3.1	Initial Prototype Test Design	25
3.1.1	Test Structure	26

3.2	Distributing learnpython to Gather Quantitative Data	27
3.2.1	Remote Evaluation with Google Analytics	28
3.2.2	Django Administration Panel	29
3.3	Experiment Design for Testing Learning Outcome	30
3.3.1	Statistical Methods	32
3.3.2	Method Selection	34
3.3.3	Null Hypotheses	35
4	Development	37
4.1	Inherited Prototype	37
4.1.1	Differences from the Inherited Prototype	38
4.2	Requirements	38
4.3	Architecture	40
4.4	Development Strategy	41
4.5	Technology	42
4.5.1	React	43
4.5.2	Django	44
4.5.3	Docker	44
4.5.4	Scaleway	45
4.6	Using the System	46
5	Results	49
5.1	Initial Prototype Test Results	49
5.2	Test Results from Distributing learnpython	51
5.2.1	Usage Statistics from learnpython	52
5.2.2	Progress Questionnaire	53
5.2.3	Questionnaire Distributed After the Final Examination	56
5.3	Experiment Results	57
5.3.1	ANOVA Report	59
6	Discussion	61
6.1	User Experience	61

6.1.1	Changes Based on the Initial Prototype Test	62
6.2	Distribution of learnpython	63
6.2.1	Progress Questionnaire	63
6.2.2	Usage and Falloff Rate	64
6.2.3	Questionnaire Distributed After the Final Examination	65
6.3	Measuring Learning Outcome	66
6.3.1	Threats to Validity	67
6.4	Addressing the Research Questions	70
6.4.1	Research Question One	70
6.4.2	Research Question Two	71
6.5	Future Work	72
6.5.1	Further Experiments on Learning Outcome	72
6.5.2	Improvements to learnpython	73
6.6	Conclusion	74
	Bibliography	76
	Appendix	81
	A Tests Used in the Experiment	83
	B Raw Data from the Experiment	91

List of Tables

2.1	Pre- Post-Test Control Group Design Outline	18
2.2	Solomon Four-Group Design Outline	18
2.3	Comparison of Pre- Post-Test Control Group and Solomon Four-Group Design	19
3.1	Experiment Timeline	31
3.2	Null Hypotheses	35
4.1	Non-Functional Requirements	39
5.1	Key Usage Numbers During Distribution	52
5.2	Descriptive Statistics	59
5.3	Levene’s Test	59
5.4	ANOVA Report	60
5.5	Tukey-Kramer Post-Hoc Analysis	60
6.1	Descriptive Statistics with Outliers Removed	67
B.1	Raw Data Gathered During the Experiment	93

List of Figures

1.1	The Code Editor in Nygårds Prototype	5
2.1	Research Strategy for RQ1	12
2.2	The Design Science Research Cycle	13
2.3	Research Strategy for RQ2	14
3.1	Google Analytics Example	29
3.2	Django Administration Panel Example	30
4.1	System Architecture	41
4.2	Error Messages from Transpiled JavaScript and Native Python	42
4.3	The Landing Page in learnpython	47
4.4	Ongoing Quiz in learnpython	47
4.5	Sequence Diagram for Quiz Execution	48
5.1	Example of Run and Submit Buttons in the Initial Prototype	50
5.2	Example of a Simple For-Loop Assignment	51
5.3	Usage Statistics from learnpython During Distribution	52
5.4	Histogram: Primary Way of Learning	53
5.5	Histogram: Prior Knowledge Level	54
5.6	Histogram: Motivation from Solving Assignments	55

5.7	Histogram: Relevance of the System	55
5.8	Scatter Plot of Pre- and Post-Test Results	58
5.9	Box Plot of Change Scores	58
6.1	Run Code, Skip, and Next Task Functionality	62

Abbreviations

ANCOVA	=	Analysis of Covariance
ANOVA	=	Analysis of Variance
HTTP	=	Hypertext Transfer Protocol
HTTPS	=	Hypertext Transfer Protocol Secure
IaaS	=	Infrastructure as a Service
ITI	=	Information Technology, Introduction
JSON	=	JavaScript Object Notation
NTNU	=	Norwegian University of Science and Technology
ORM	=	Object-Relational Mapper
REST	=	Representational State Transfer
SaaS	=	Software as a Service
SQL	=	Structured Query Language
VPS	=	Virtual Private Server
XML	=	Extensible Markup Language

Chapter 1

Introduction

Many technological studies at university level have mandatory introductory programming courses. Unlike traditional courses such as mathematics and physics, where students typically have experience from middle- and high school, students enrolled in introductory programming courses have varying degrees of experience [1]. Even with the existence of powerful online tools, books, and lectures, lesser experienced students tend to struggle with learning fundamental concepts required to understand how to create simple programs [2], [3].

In a society with high demand for IT-solutions, programming is an important skill to have in most technological lines of work. It is no longer an ability only limited to computer scientists and enthusiasts. Biologists might use it to simulate species development given certain parameters. Computational chemistry is a large field, with applications such as modeling the behavior of atoms and molecules. Due to the importance of programming in a wide range of professions, it is important that professors motivate their students to learn it. Getting hands on experience is necessary to become a good programmer [4]. Thus, it is important that professors present their students with tools that are both motivating and efficient, and that makes them want to write code.

1.1 Background

At the Norwegian University of Science and Technology (hereby NTNU), most students starting their technology studies have to take an introductory course in programming called *Information Technology, Introduction* (hereby ITI). The course comprises basic computer science theory, as well as an introduction to procedure oriented programming in either Python or MatLab, depending on the study program. Python is a high-level, cross-platform programming language with an emphasis on code readability. Compared to languages like Java and C++, it has simple, compact syntax to allow for programmers to express code in few lines, making it a suitable language for introductory classes. MatLab on the other hand, is more similar to traditional programming languages in terms of syntax, with an emphasis on mathematics. Students enrolled in ITI have widely different backgrounds. Some students have a broad understanding of computers and are experienced programmers, while other students have never programmed before. That leads to a tendency where experienced students tend to cruise through the course, while those less experienced typically struggle.

The primary goal of the course is to make sure students learn the basic principles of programming. To achieve that, it is important to cater to those less experienced. As of now, that is achieved through lectures, a mandatory set of exercises, and study groups for those who need it. The exercises are distributed through the semester and consist of programming tasks as well as theory questions. Although the exercises are a good way to get the basic understanding that is required to pass the course, a considerable amount struggle with understanding the concepts required to solve the tasks. Not understanding how to solve the tasks in an exercise tempts some students to copy other students answers to pass the exercise. This effectively removes most of the learning outcome the students would otherwise get by spending time trying to understand how to solve the exercises. Additionally, it leads to students developing bad habits of cheating on exercises. In the end, a considerable amount of students fail or get poor grades in the course.

1.1.1 Motivation

Both authors of this thesis completed ITI as part of their first semester at NTNU. None of them had any prior experience with programming. Many of the exercises were difficult, and it took a long time to understand core programming concepts. Getting to the point where understanding how functions, loops, and control structures all fit together was challenging. Over the course of five years of studying, the authors have observed other first graders struggle with the course. Hence, the possibility to do research on new and innovative learning tools is interesting. Being able to contribute to research that aims to give people new and fun ways of learning how to program is motivating for the authors.

1.2 Problem Description

The challenges described previously forms the basis of the research. The primary focus is on motivational factors and learning outcome of different learning methods, making students want to spend time learning how to program and do so efficiently. The research is performed through an incrementally developed self-evaluation system called learnpython. The system is created primarily for research purposes. However, if the user testing yields satisfactory results, it might be used as a supplement to exercises and lectures in ITI in the future.

The targeted user group are students taking ITI at NTNU. User experience, as well as an implementation of features derived from user testing, is emphasized. The incremental approach allows new or changed features to be properly tested and adjusted according to user feedback. When a functional and satisfying system tailor made to fit the ITI course is in place, it will be used to conduct research. The goal is to find out what motivates students to use such a system, and how much they learn from using it. Eventually, it is desired to incorporate such software in ITI. With that in mind, a natural beginning is to find out whether it can be effectively used as a learning tool, which is the overall goal of the research. With the design and creation of a self-evaluation system in mind, and given that it is functional and easy to use, the following research questions are presented:

RQ1: Is the system motivating to use as a way of learning how to program?

RQ2: Is the use of the system an efficient way of learning how to program?

RQ2.1: Is the use of the system a more efficient way of learning how to program than traditional reading?

1.3 Nygård's Master's Thesis

This thesis is a continuation of the work done by Sindre Haneset Nygård in the fall of 2016 [5]. As part of his master's thesis, he conducted research on the use of self-evaluation systems in introductory programming courses, attempting to find design elements that would make students want to use such software. The prototype he proposed and created was a web based solution, in which users got assignments they could solve by programming in Python. The code is written in a text editor, and the output is shown in a text area. When submitting a correct answer, the user is presented with the next assignment. Figure 1.1 shows the central part of the prototype, where users can write code, submit it and see the output. After completing a certain amount of assignments, the user proceeds to the next level, meaning that more challenging assignments are presented. The idea behind it is simple; the more assignments a user solves correctly, the more difficult the tasks get. An achievement system is also implemented, notifying users after they have reached certain milestones, such as solving five assignments in a row.

Before creating his prototype, he created a questionnaire in an attempt to discover features that should be included in the system. He found that students wanted the system to automatically adjust the difficulty of the tasks based on individual skill level. Additionally, students wanted short assignments that does not necessarily need too much context to them.

Having created his prototype, Nygård carried out user tests on students. The results indicated that gamification elements in the form of the achievement system made the students inclined to continue using the system. His level system, where users were presented with more challenging tasks the more they solved, was also positively received. Due to the positive feedback from the tests, his prototype is used for initial testing and further development. The research conducted in this project will, however, have some differences. His

LEARNPYTHON
Logout

Functions

Assignment: Level difficulty: 1

Create the body of function `f` so that `f` returns the sum of `num1` and `num2`

```

1 - def f(num1, num2):
2     return num1 + num2
3
4 print(f(4,5))

```

Run code
Reset code editor

The expected output is

9

SUBMIT

Please refer to question ID 27 if there are any problems with the question it self.

Figure 1.1: The code editor in Nygård's prototype

project emphasized viable design elements in such systems, while this project has focused on motivational effects and learning outcome, with good design being a prerequisite for achieving that.

1.4 Related Work

In later years, there has been a growing interest around renewing of learning methods. Several new online platforms have arisen, with the goal of providing efficient learning tools for an array of different subjects and fields. Classroom teaching is also changing, with teachers and professors attempting to incorporate new and innovative tools to make lectures more fun and exciting, e.g., Kahoot¹ created by a professor at NTNU. This project falls within the category of such tools, with the scope set to programming. Following is a list of relevant research done on the field of learning how to program. It ranges from research done with specific software in mind, to more theoretical papers on learning in

¹Available at <https://getkahoot.com>

general. The papers are meant to provide a context in terms of what data has been gathered from previous research. The work listed is what was deemed the most relevant research done in the field for this thesis. In addition, the last section presents additional software related to the field of learning how to program.

1.4.1 Ways of Experiencing the Act of Learning to Program: A Phenomenographic Study of Introductory Programming Students at University

In the paper *Ways of Experiencing the Act of Learning to Program: A Phenomenographic Study of Introductory Programming Students at University*, Bruce et al. [6] discuss the ways in which novice programmers first learn how to program. The primary motivation behind the study is to provide empirical insight into this phenomenon, in order to influence the way professors teach programming. The research is mainly based on semi-structured interviews with first-year students from the Bachelor of Information Technology degree. Each interview consists of questions regarding learning, as well as how they see programs and program code.

Based on the results from the interviews, students were placed into different categories based on how they learn and how they experience the act of learning. On the one hand, there are those who seek to learn by doing the minimal requirements, focusing on tasks that affect the grade. These kinds of students are highly affected by the structure of the course, showing the importance of well defined and good exercises. Their underlying motivation to learn is the desire to pass the course. On the other side of the scale, there are those who are genuinely motivated by the problem they are attempting to solve. For them, merely completing an exercise might not be enough. They seek to understand why certain aspects of the code they have written works in the way it does, often with guidance from teachers and teaching assistants. This basically means that they spend more time perfecting the work they are doing, learning in the process. Several categories are presented that lies in between the two, representing groups of people who have different motivations and approaches to courses.

The importance of this paper lies in the different ways students approach programming

courses. This poses implications on the curriculum of introductory programming courses, as well as the learning environment and resources. To reach and motivate as many students as possible, the difference in learning styles need to be considered.

1.4.2 Five Years with Kattis - Using an Automated Assessment System in Teaching

Kattis is an automated assessment tool developed at the Royal Institute of Technology in Stockholm, where it is used to evaluate programming exercises [7]. The software is able to test programs written in several languages for correction, by running one or more pre-defined tests based on different input. The underlying motivation for creating the software in the first place was to remove the need for teaching assistants to verify program correctness manually. This allowed them to focus more on assessing problem-solving and programming skills with individual students.

After the initial release of Kattis, the effect of it was studied from 2006 to 2010. The research focused on two things primarily; the impact the software had on students pass rate for exercises and the students satisfaction with the software. The pass rates for students were tracked in two exercises in the *Algorithms, Data Structures, and Complexity* course. The introduction of Kattis lead to the pass rates falling by approximately 10% on average. However, Enström et al. argue that Kattis tested programs more thoroughly than the manual testing had before, thus setting stricter requirements for submissions [7].

Another interesting discovery was that the system appealed to the competitive side of students. They were motivated by the fact that their submissions were timed and tracked in a high score list. Several students continued to attempt to climb the high score list, even after having an accepted submission. This shows a motivational effect in competition, even when it comes to learning tools like Kattis.

1.4.3 Individualized Exercises for Self-Assessment of Programming Knowledge: An Evaluation of QuizPACK

QuizPACK is a programming quiz system developed at the University of Pittsburgh [8]. The system presents questions related to code understanding, rather than code writing.

Questions are designed to help students understand how values of variables are manipulated with different control structures. A typical question involves some lines of code, and the task is to guess the value of some variable after the code has run. Additionally, the system emphasizes simple creation of tasks through the use of parameterized questions. That is, the system can automatically create different parameters for the same task, avoiding cycling through the exact same problems multiple times.

The usage of QuizPACK was evaluated at the University of Pittsburgh in 2002 and 2003. Although the data is old and the system has gone through changes since then, the data collected is interesting. An evaluation of the results showed that students who used the system benefited strongly. However, less than a third of the students used the system actively. This relationship indicates that while the usage of such systems benefit students, it is difficult to make the students use them.

1.4.4 Experimental Validation of the Learning Effect for a Pedagogical Game on Computer Fundamentals

In 2003, a game called Age of Computers was introduced in the course *Computer Fundamentals* at NTNU. The game is a question/answer based game where players need to navigate around and solve exercises related to the curriculum. Following its introduction, an experiment was conducted to find the effect of having such a game as a mandatory exercise, compared to reading and more traditional, paper-based exercises [9]. The study revealed that the game had a significant impact on the learning outcome of students. However, it did not prove to be more efficient than other studying methods. On the other hand, informal talks with students and teaching assistants during the first semester gave the impression that it was motivating to use. The motivational factor is important, as people are more willing to spend time doing something that is motivating to do.

Age of Computers is a game which is quite dissimilar to the software created as part of this research project. Because it is a game, they cannot be directly compared in terms of learning outcome and motivation. The research is still relevant to this project, as it shows how new and innovative learning methods can be a viable strategy to improve the motivation of students. Although the game might not be a more efficient way of learning,

Bruce et. al [6] claim that students learn in different ways and have different approaches to learning. Thus, playing Age of Computers might be more efficient and motivating for some students. This paper is in many ways similar and relevant to the work done in this project.

1.4.5 Similar Software

Following is a list of software aimed at teaching people how to program. Although no concrete research is presented concerning these solutions, these websites showcase efforts that have been made in the field.

Python Tutor

Python Tutor² is an open source software for web-based visualization of programming [10]. The motivation behind this software is to help students learn the fundamentals of programming. This is achieved by simplifying teaching through automatically visualizing heap- and stack diagrams together with step-by-step code execution in the web browser. Additionally, it supports live programming, enabling collaborative coding sessions so that instructors or friends can join in, to help and discuss the code. Python Tutor may also be embedded into other web pages through JavaScript.

Runestone Interactive

Runestone Interactive³ is a website that provides free educational books about programming. These books are interactive, containing elements such as code editors (using Python Tutor) and multiple choice tasks within the chapters. The interactive elements can be utilized by the reader to test the concepts while reading about them. This effectively allows readers to get practical experience while learning theoretical concepts. These books are used as part of courses by over 500 teaching institutions around the world, according to their authors.

²Available at <http://pythontutor.com/>

³Available at <http://runestoneinteractive.org>

Codecademy

Codecademy⁴ is a website aimed at teaching people how to program, by providing programming tasks in the web browser. The tasks are linearly increasing in level of difficulty, meaning you follow a progression curve. The website offers courses in multiple programming languages. Typically, a course starts off with simple concepts, gradually progressing to the more advanced. Towards the end of a course, users are typically tasked with creating programs using the concepts that have been presented. Codecademy has tasks from several programming languages, including Python, Java, JavaScript, and more. The site has gained more than 25 million users since its launch in 2011, and it is still widely popular. In many ways, Codecademy is similar to Khan Academy⁵, a site that focuses on online teaching in a wide range of subjects.

1.5 Report Structure

Following the initial introduction, the rest of the thesis is structured as follows; the research strategy, as well as the means of data collection and analysis, is discussed in chapter 2. Chapter 3 elaborates on the usage of the test methods described in chapter 2 and explains how the tests were executed. Potential hazards and their respective precautions with regards to data validity and bias are also considered. Chapter 4 is concerned with the development of the software artifact used as part of the research. Development strategy, the technologies used, and the reasoning behind these decisions, are discussed. It also contains an explanation of how the system can be used from a user perspective. The results gathered during data collection are listed and explained in chapter 5. Chapter 6 is the discussion, meant to provide answers to the research questions listed in section 1.2. An interpretation of the results in chapter 5 are included, as well as any concerns with data validity. Additionally, any remaining work and proposed future work are discussed. Finally, a conclusion is drawn from the overall work done during the project period.

⁴Available at <https://www.codecademy.com>

⁵Available at <https://www.khanacademy.org>

Chapter 2

Method

This chapter describes the research methods used to gather the information required to address the research questions outlined in section 1.2. The different research strategies utilized for the two research questions are described at the beginning of the chapter. Following is a description of the data collection approaches for the research questions. The end of the chapter describes potential hazards that might affect the data in different ways, and any ethical issues that need to be considered.

2.1 Research Strategies

The research questions are quite different in terms of the approaches required to answer them. While research question one relies heavily on personal opinions about how fun and motivational such a system is to use, research question two relies on actual measurements of learning outcome. Due to these differences, two separate research strategies were used to answer the questions. Subsequently, there was a need to address the questions one after the other. Research question two is reliant on the system being developed as part of research question one. Thus, research question one was addressed and answered, and the research method terminated before the work with question two started.

2.1.1 Research Strategy for Question One

Research question one is about students motivation to use such software as part of their learning experience and reads: *“Is the system motivating to use as a way of learning how to program?”* In light of the involvement of a system as the primary vehicle for the research, design and creation by Oates [11] was chosen as the research strategy. As opposed to other strategies such as surveys and case studies, it incorporates the use of an artifact to aid the research. Having the need to create and test such an artifact (software in this case), it suits the needs of this project better than its alternatives. Design and creation is commonly used in research related to software. The research question(s) form the purpose of the system and is typically used to gather data in some way. Figure 2.1 outlines the research approach for research question one.

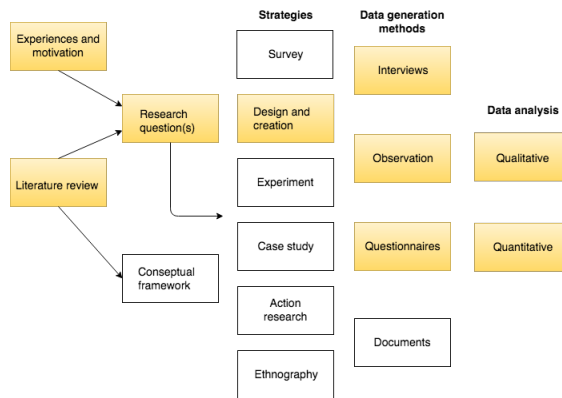


Figure 2.1: A visual presentation of the research strategy for research question one

The research strategy involves designing and analyzing an artifact to address some research objective(s), in this case, the difficulties and motivational factors involved in learning how to program. Design and creation outputs tentative designs, artifacts, result, and performance measures used in the process. The artifact involved in this research is a self-evaluation tool for use in ITI. After the creation of the artifact, it is used to present justification, analysis, explanation, critical evaluation, and new knowledge. The new knowledge is presented through the final results and discussion of the research questions, whereas the justification and analysis are part of the process of analyzing the usage of the system.

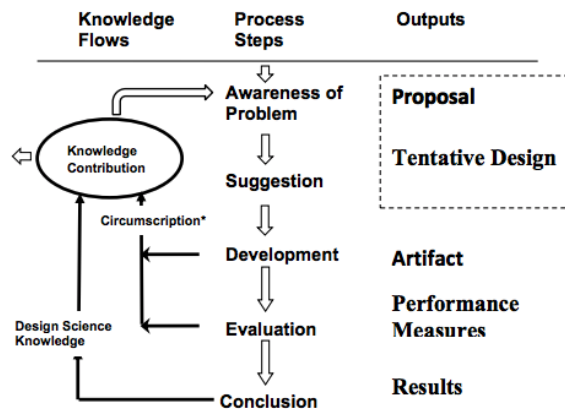


Figure 2.2: The Design Science Research Cycle by Vaishnavi and Kuechler [12]

To execute the research strategy successfully, the research question needs to be closely related to the purpose of the artifact. The system in its context of use should address the research question in some way. Upon testing the system on the targeted user group, it should be possible to get feedback directly linked to the research question. Typically, the design and creation process is an iterative process consisting of five steps. The steps do not need to be followed step-wise, but they form a guide for an iterative cycle, leading towards the goal of the research. Figure 2.2 shows the general design and creation approach. The five process steps are described below:

- Awareness is about identifying and recognizing the problem. This is achieved either through literature studies, field research or client requests for a new technological solution.
- Suggestion is the initial brainstorming, where a tentative idea on how to solve the problem is presented.
- Development is the creation of the IT artifact. This artifact is most often developed according to the idea presented in the suggestion step.
- Evaluation is an examination of the artifact and it is evaluated according to a criteria or evaluation measure.
- Conclusion is where the results are gathered, and knowledge is identified. This may be the final step if no further cycles are to be executed.

The initial literature study, as well as the background for the research, forms the awareness step in this context. As an inherited system was available, the suggestion step consists of identifying both flaws found during user testing and known issues with the prototype. Furthermore, ideas for improving or correcting the flaws and issues are suggested, leading to the development step. The evaluation and conclusion steps are the results gathered from distributing the system to students, along with the interpretation of said results.

2.1.2 Research Strategy for Question Two

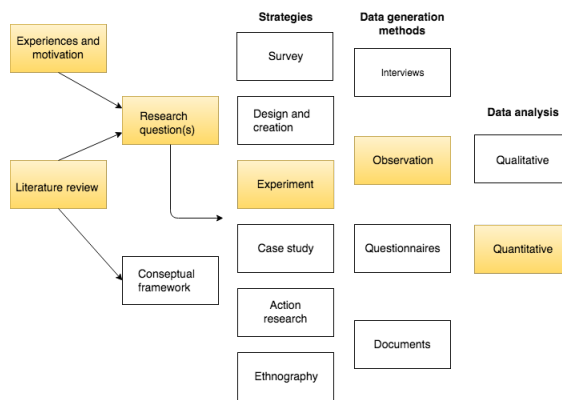


Figure 2.3: A visual presentation of the research strategy for research question two

Research question two is about the act of learning and reads: *“Is the use of the system an efficient way of learning how to program?”* Additionally, there is a sub-question, in which the system is compared to traditional reading as a way to learn: *“Is the use of the system a more efficient way of learning how to program than traditional reading?”* When starting to work on research question two, the design and creation process was terminated. An experiment was chosen as the most suitable approach to address it. An experiment outputs observed numerical data, which can be used to analyze the results statistically. This approach gives actual, observed data about learning effect, as opposed to some perceived effect. Figure 2.3 shows the general approach for research question two. The experiment is conducted in an attempt to discover the learning effect of the system. Using observations as a term for the data generation method is somewhat vague. In this case, observations are tests that subjects take, to measure what they have learned. Finally, the data collected is

analyzed in a quantitative manner, using statistical analysis and meta-analysis to uncover learning effect and the strength of the analysis.

2.2 Data Collection for Research Question One

The data collection approach for research question one is a combination of quantitative and qualitative data collection. The results of the data collection lead to suggestions, meant to form a way of solving the problems found in the data collection. The two data collection approaches are used in different phases of the research project. The qualitative approach was used early on in the project to identify issues with the prototype created by Nygård. Any errors and inconsistencies found were fixed, and the issues with the user interface lead to a redesign, forming the new system learnpython. Quantitative data collection was used to conduct research on motivational aspects related to the new system.

2.2.1 Qualitative

The qualitative methods used to collect data are discussed in this section. The qualitative part was conducted through observations and interviews, meant to provide data about the type of functionality that should be included in the system. Observations and interviews were conducted in the early stages of the research project.

Observations

Observing allows researchers to notice and understand things that might otherwise be missed [13]. This is an important aspect when studying how students think and feel about using software. It opens the possibility of seeing the reactions and emotions of test subjects. Observations can be further categorized as structured, semi-structured, and unstructured observations [11], [13]. While structured observations have a set agenda with specific events that are looked for and enumerated, the unstructured approach allows observers to review observational data before suggesting an explanation. Having the need to observe the reactions and emotions of users when using the system, a semi-structured approach is utilized in this research project.

Interviews

In this research project, interviews are used on each test subject after observations. The interviews are planned and conducted following the procedural guidelines of Jacob and Furgerson [14]. When conducting interviews, it is important that the questions are related to the research questions, and do not stray into unrelated topics. When interviewing a test subject, questions are often predefined. In this case, some questions are based on observational data gathered. Thus, the interviews consist of two parts, one predefined and one with open follow-up questions related to observations. Due to that, the data gathered will inevitably be interpretive and unstructured. The follow-up questions are expansive and open-ended, allowing the test subject to elaborate freely upon the topics discussed. All observations and answers to questions are written down. These notes are means of identifying problems with the design of the user interface. The problems identified are then subject to change and further testing.

2.2.2 Quantitative

Quantitative data collection is often used to acquire numerical data and generate statistics. In this case, it is desirable to generate numerical data about the system at hand and uncover usage patterns. It is important to uncover whether students are willing to use such systems as part of the learning experience. Additionally, this part of the research seeks to discover to what degree students are motivated to use the system. The quantitative research was conducted through remote evaluation and questionnaires.

Remote Usability Evaluation

The core concept of remote evaluation (asynchronous testing) is testing where the observers and test subjects can be separated in both time and space. Remote evaluation gives test subjects the opportunity to use the system without the observer(s) present, while the observer(s) simultaneously gather data from the usage. The user is able to use his or her own computer to use the software in a familiar environment. Martin et al. conclude that results from remote evaluation are comparable or of higher quality than results from traditional laboratory testing [15].

In this thesis, the tool used for remote evaluation is Google Analytics¹. It is a highly customizable web analytics service offered for free by Google. The primary functionality of Analytics is, however, sales and marketing. This aside, Analytics can with configuration be used to capture custom user events and time spent on a single page at the client-side.

Questionnaires

Two questionnaires were distributed to the subjects. One was presented as part of learnpython after a user had successfully completed five tasks. The other was sent by email after the final examination in ITI. These questionnaires measured the following aspects:

1. The methods they used to learn to program.
2. Motivational aspects of learnpython.
3. The parts of learnpython that made it useful.
4. What could be improved in learnpython.

Questionnaires were chosen for this thesis since they are an efficient method to collect quantitative data from the users of the system [11]. They are practical and allow researchers to gather large amounts of data in a relatively short amount of time. With questionnaire results, it is possible to identify if the users understand the system and whether they are motivated by using it. Since these questionnaires are self-administered, and there is no researcher present, the quality of the questions are important.

2.3 Data Collection for Research Question Two

Cohen et al. present the pre- post-test control group design as a solid experimental design, which is suitable within educational experimentation [13]. The design is similar to the one group pre- post-test design, where a test group is measured before and after receiving some treatment. In the first case, a control group is introduced. The control group is used to negate issues with internal validity, such as differences in the pre- and post-test. The design in general consists of two tests, and a categorical independent variable or treatment distributed in between the tests. Subjects are assigned to groups at the beginning of the

¹Available at <https://analytics.google.com>

experiment. Each group then take the same pre-test, typically using some continuous scale to measure or evaluate the results of the test. The treatment group(s) then undergo their respective treatments for a set amount of time. At this point, the control group performs no activities that might accidentally influence the dependent variable (post-test results). In medicine, this usually involves taking some placebo drug for instance. Finally, all subjects undergo the same post-test, which is usually equal, or at least similar to the pre-test. In the context of this research, two treatment groups are required, with one control group.

Group	Pre-test	Treatment	Post-test
Treatment Group 1	O	X	O
Treatment Group 2	O	X	O
Control Group	O		O

Table 2.1: Outline of the pre- post-test control group design. O represents the act of measurement, while X represents the exposure to some treatment.

Group	Pre-test	Treatment	Post-test
Treatment Group 1	O	X	O
Treatment Group 2	O	X	O
Control Group	O		O
Treatment Group 1'		X	O
Treatment Group 2'		X	O
Control Group'			O

Table 2.2: Outline of the Solomon four-group design. O represents the act of measurement, while X represents the exposure to some treatment.

While the before-mentioned design generally has solid internal validity, it does not account for the potential impact the pre-testing might have on the subjects, which limits the ability to generalize the study. The Solomon four-group design [13], [16] accounts for this. The Solomon four-group design introduces additional groups to control for the potential effect of the pre-test. These groups do not partake in the pre-test but undergo the same treatment as their counterpart. Although this method is solid in terms of validity, there are some issues with it. First and foremost, it requires more subjects, and the experiment is more complex in terms of execution. Additionally, it is more difficult to analyze the results of such a design statistically. With these disadvantages in mind, using a traditional pre- post-test control group design was deemed sufficient for this experiment. The classic

	Pre- Post-Test Control Group Design	Solomon Four-Group Design
Internal Validity	Uses the control group to control for issues with internal validity.	Stronger internal validity due to the introduction of three additional groups, not performing the pre-test, to control for the effect of the pre-test.
External Validity	Uses random selection from the population.	Similar in that the subjects are randomly selected from the population.
Complexity	Fewer groups means that more subjects can be placed in each group, reducing complexity.	More groups make it more difficult to control the experiment environment.

Table 2.3: A comparison of the pre- post-test control group design and the Solomon four-group design

pre- post-test control group design is outlined in Table 2.1, while the Solomon four-group design is outlined in Table 2.2. Both tables describe the experiments conducted with two treatment groups and one control group. Also, a summary of both designs is found in Table 2.3.

2.4 Bias and Data Validity

The different data collection methods described in this chapter have certain threats to their data integrity associated with them. The following sections describe the threats that were deemed the most important, how they occur and the consequences they can have. The steps taken to reduce the impact of these issues are covered in each tests respective section in chapter 3.

2.4.1 Social Desirability Bias

Observations and interviews are subjective by nature, given that it is an exchange between two people. This causes issues related to social factors. Social desirability bias is a type of bias that is threatening when conducting interviews and distributing questionnaires [17]. In the context of interviews, it means that the interviewee will lean toward trying to please the interviewer by altering responses in an attempt to give the answer they think the inter-

viewer seeks, and thus making themselves more favorable to the interviewer. The same goes for questionnaires, where results might be skewed due to overly extreme answers. Bruner and King claim that *"A major source of response distortion identified early on by researchers in this area is the tendency for subjects to "fake good" or "fake bad" responses to personality test items"* [18, p. 81]. Although this refers to the field of psychology, the concept is transferable to user testing and interviews. The consequence is that the gathered data might be more extreme than what it should be, as the subject subconsciously wants to "please" the interviewer. Asking non-leading questions is the most important factor to mitigate the consequence of this. The exact impact it has on a specific case is difficult to measure.

2.4.2 Acquiescence Bias and Demand Characteristics

Using questionnaires as a data collection method poses some additional implications related to the answers that are received. One of the main issues is acquiescence bias, or so-called "yay-saying" [17]. This type of bias is based on peoples tendency to find it easier to agree with statements than to disagree. The phenomenon might lead to some people not answering entirely honestly, skewing the results of a questionnaire in some direction. In turn, that leads to inaccurate data. Additionally, it is also somewhat related to demand characteristics, a type of bias in which participants alter their answers because they know they are part of some research [19]. Participants might consciously or subconsciously attempt to figure out the purpose of the research and alter their answers thereafter. In the end, it leads to another source of potentially inaccurate data.

2.4.3 Non-Response Bias

Non-response bias is a type of bias where results from respondents of a questionnaire can differ from the potential answers of those who did not answer [20]. This can occur when a low percentage of those asked to respond, actually answers. When this occurs, the answers are less representative of the targeted user group. Thus, having a high response rate is an important factor when conducting research with questionnaires. This issue is difficult to deal with in an efficient manner. One way to make people more likely to answer is to

incentivize them in some way. If respondents simply refuse to answer, there are in most cases little that can be done to change this. Forcing or excessively requesting answers from individuals is deemed unethical, and thus unacceptable.

2.4.4 Missing Data With Google Analytics

Commonly used ad-blocking software and web browser extensions actively block Google Analytics by default. When people use such software, Analytics is unable to track information about that user at all. Thus, the data being generated from this source is incomplete, lacking usage data from some percentage of the user base. It is hard to measure how much data is lost due to ad-blocking software, as the number of people who visit a site with ad-blocking software can not be easily measured. However, it is possible to map the number of users registered a system and compare it to the users tracked by Analytics.

2.4.5 Human Involvement in Experiments

When conducting experiments, bias might occur from human involvement, be it the researcher(s) themselves or the human test subjects. Conducting experiments with a hypothesis to test might lead to a subconscious choice to either influence the participants of the experiments, or treat the results in a manner that supports the hypothesis. This kind of influence is known as the observer-expectancy [21] effect and might occur at any stage of an experiment. To avoid bias when evaluating results, Oates et al. presents a solution to this issue where the researcher is "blind" [11]. That is, the researcher does not know which group the results belong to when evaluating them. As for conducting the experiment itself, it is important that all subjects are treated equally. There are ways to effectively avoid bias when conducting the experiment as well, such as the double-blind approach [22]. The approach is an extension of the blind approach in a way, where the researcher does not know who are in which group during the experiment either. The approach is difficult to conduct in this case, as those leading the experiment need to know who are in which group.

Oates et al. presents several bullet points for potential issues when dealing with human test subjects [11]. First and foremost, the subjects need to be relevant to the experiment that is performed. It is also preferable to have subjects that are of equal suitability to

the experiment. If a person's age might have an impact on the results of the experiment, the participants should be of similar age. Finally, it is preferable to use control groups to ensure that the outcome of the experiment is due to the treatment that the subjects are exposed to, rather than external factors.

2.5 Ethical Issues

In order to preserve the privacy of individuals, awareness of potential ethical issues is important when conducting research. In this research project, it is mainly related to the data collection in terms of; how to collect the data, what data will be collected, and how this data is handled. The first issue to address is the usage of Google Analytics, where the users were not informed of the tracking. Consequently, unless the user used ad-blocking services, they were unable to opt-out. This issue is partially handled in the core of Google Analytics. For Analytics to be able to determine whether separate events belong to the same client, every client is provided with a unique identifier. The identifier is created as a randomly generated string stored in the client's browser cookies. Hence, there was no need to trace a specific event to a username or other identifier.

To use the system, the users had to register using an email address. The email address was collected to contact each user asking whether they would be willing to answer a questionnaire. During registration, users were presented with information about this questionnaire and were given the possibility to opt-out. Thus, the questionnaire was not forced upon them after they had finished using the system. The email addresses stored in the system's database are not distributed or shared and are to be deleted upon completion of this project.

Finally, steps were taken to protect privacy during the experiment. All the information the subjects needed to know prior to the experiment were distributed through a contact person for the different organizations test subjects were recruited from. The contact persons were not part of the actual experiment, and the researchers possessed no personal information about the subjects, apart from what organization they were recruited from. The experiment subjects were given practical information before showing up, making sure they knew what they signed up for. Additionally, each subject was randomly assigned a

generated identifier upon starting the experiment. Hence, it was not possible to map a specific person to the identifier after the experiment was conducted. Upon showing up, subjects were informed that the purpose of the experiment was to assess different learning methods, not individual performance. All of the subjects recruited to the experiment were treated with respect and dignity.

Test and Experiment Design

This chapter is tightly coupled with chapter 2, but with a focus on how the general research methods and data collection approaches are used in this project. In the first part of this chapter, the initial prototype test that was performed is discussed. The planning and execution of the test are described, as well as the recruitment of users for the test. Following is a description of the planning that went into distributing the system and how it was marketed to students. Both the questionnaire implemented in the system and the questionnaire distributed to students after their examination is elaborated upon. Additionally, how data was collected from remote evaluation and the Django administration panel is emphasized. Furthermore, the chapter contains an evaluation of different statistical methods evaluated, and the final choice of method to analyze the experiment data is elaborated upon. Finally, the null hypotheses for the experiment are presented.

3.1 Initial Prototype Test Design

The initial prototype test was conducted with the inherited prototype created by Nygård. However, it was slightly modified in an attempt to address some of the issues he uncovered. The primary goal of this test was to discover problems with the user interface, as well as questioning students about what sorts of features they would want in such a system. The test is similar to the final prototype test Nygård performed. He was not able to conduct the

test on students currently enrolled in ITI however. Thus, conducting a new prototype test made it possible to carry out the test on the targeted user group.

When conducting the test, subjects were allowed to use the system freely. As the test did not target any particular features in the system, it was unnecessary to have predefined tasks that subjects were to do. In general, it was important to determine the current state of the system. Any potential bugs or unintuitive parts of the system needed to be uncovered to maximize the chance of students using it for an extended period of time.

3.1.1 Test Structure

Each test was conducted with one test subject and lasted approximately 35 minutes. Before the test, subjects were informed about the observation in accordance with the ten step guideline designed by Apple Computer [23]. The guideline is intended to make test subjects understand the environment they are being observed in, why they are test subjects and that the goal is to test the system, not them. The tests were conducted in two parts; an observation and a semi-structured interview. During the observation part, the test subject used the system freely while the observers captured issues and thoughts the subject had. The observer did not help the subject during the test, apart from addressing some minor technical issues. The observation part of the tests lasted approximately 20 minutes. After the observation, a semi-structured interview followed. First of all, the subject was asked some predefined questions. Then an open conversation was held to allow exploration based on what the subject said, and to address any reactions or emotions the observer had seen. Each interview lasted approximately 15 minutes.

The test was conducted on a total of five test subjects. All of the test subjects were enrolled in the ITI course during fall of 2016 but had different backgrounds. Two of the test subjects were female, while the other three male. Two of them had coding experience prior to the course, a male and a female test subject. The three other had no experience other than what they had learned so far in the course. The diversity of test subjects was deemed important, due to the possibility of them noticing different things in the software. The test subjects were members of the student organization Online¹, and recruited through

¹More information about the student organization is available at <https://online.ntnu.no/>

the public chatting channels of the organization.

3.2 Distributing learnpython to Gather Quantitative Data

Before distributing learnpython, a hallway test was conducted on five students, testing the system after its completion. The tests focused on the new features implemented, making sure they were well understood. They were held in a normal work environment at a computer lab. After using the system, the subjects were asked if they found any particular issues with the user interface. Test subjects were recruited in the same fashion as those in the initial prototype test.

By distributing the system to students, it was possible to gather quantitative data about the usefulness and motivational aspects of such a system. The data was gathered through remote evaluation, in addition to analysis of data collected in a database. Additionally, a questionnaire is implemented in the system, which is presented to students having successfully completed five assignments. This questionnaire is aimed at gathering information about how they feel about the system, both in terms of motivation to use it, and if they actually learn from using it. Answering questions while using the system yields information that comes fresh from their memory. Two of the questions gather data about the user group, and which sorts of students use the system. The final two are about motivation and students perceived learning outcome. The questions and the results of the questionnaire are further elaborated upon in chapter 5.

To distribute learnpython, a note about the option to use the system was posted on ITT's wiki page, visible to all students taking the course. Additionally, the solution was marketed during one of the last lectures of the fall semester, reaching approximately 100 students. It is important to note that it was clearly marketed as an optional way of preparing for the exam, meaning that students who signed up did so voluntarily. In order to reach as many students as possible and to vouch for the validity of the system, the opportunity to reach out through the courses official channels was important.

An additional questionnaire was sent to students in early January 2017. Upon registering to the system, students were asked if they wanted to participate in a questionnaire regarding their usage of the system. The questionnaire was only distributed to students who

wanted to participate, in an effort to avoid having students who did not want to participate after their examination feel forced to do so. The questions regarded the features students found useful, and if they had any particular problems with the system. With the other questionnaire being implemented in the system, there is a possibility that users changed their mind after answering it. Having the latter questionnaire allow users to elaborate upon that.

In an attempt to avoid acquiescence bias, the questions were formulated in a neutral manner. That means that they were formulated as questions with a range of answers (e.g., *"On a scale from 1 to 5, to which degree do you feel this software is relevant to learning Python?"*). The contrary to that is formulating statements, such as *"I liked this system."* to which you can agree with on a scale from one to five. Although it is not a surefire way of avoiding the issue entirely, it negates some of its impacts by not letting a user simply "agree" with a statement. Tracking how many assignments users have solved also backs up questionnaire data. Seeing how many assignments users have solved serves as an indication of how much they like the system. Users who like it, will in most cases solve more assignments than those who do not.

3.2.1 Remote Evaluation with Google Analytics

One of the methods of data collection used was Google Analytics. It allows administrators to track data about several aspects of the users visiting the site. The data represents usage patterns in a meaningful and valuable matter. It is difficult to deal with the lack of data due to ad-blocking software. Awareness is important, in knowing that the data is in most cases incomplete. In this research project, Google Analytics is used to collect statistics about:

1. Which assignments a user solve and if the solution was correct or wrong.
2. Average time spent in learnpython (length of a session).
3. Total page views and unique visitors.
4. Bounce rate (percentage of users leaving without progressing from the first page).

These statistics provide general information about usage patterns. The amount of time people spend using the site, along with the number of assignments they solve, can be used

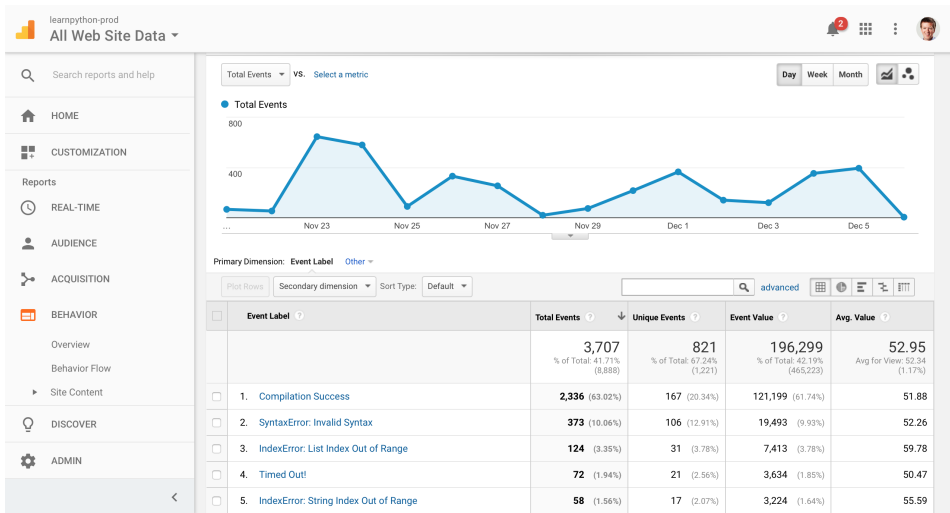


Figure 3.1: An example of Google Analytics showing data about assignment compilation requests

to determine how much effort people are willing to put into learning by using the system. However, it is important to keep in mind that the data collected about time spent using the site may not be entirely precise. This is due to an issue with Google Analytics, as it is not able to measure the time spent on the last visited page, thus being aware of potential data loss is important when interpreting this data. Figure 3.1 shows an example of a statistics page within Google Analytics.

3.2.2 Django Administration Panel

The Django administration panel is an administration solution included in Django by default. In its simplest form, it is an interface allowing administrators to view, create and change data in the database. Making students have to register an account to use the system, opens the opportunity to see how many assignments each user solve, which assignments users tend to struggle with, etc. Most importantly, it is possible to track how many students sign up, and how many assignments they solve. It is also possible to see how much individuals use the system, whether or not they quickly give up, and so forth. Although this type of data is subject to interpretation, it does give indications about usage patterns and how students feel about the system. In combination with remote evaluation

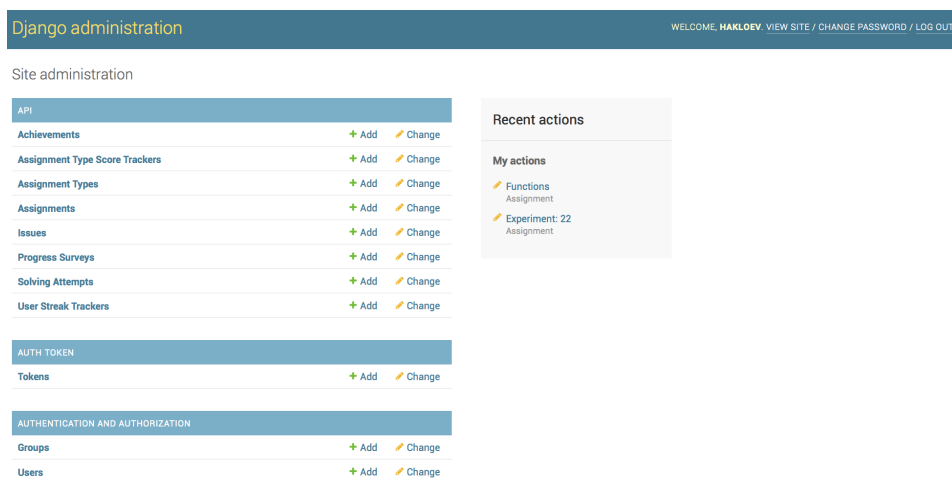


Figure 3.2: The main page of the administration panel in learnpython

and the questionnaires, it is possible to draw conclusions about motivational factors and how much students are willing to use the system. Figure 3.2 shows the main page of the administration panel in learnpython.

3.3 Experiment Design for Testing Learning Outcome

Questionnaires and remote evaluation provide an indication of students perceived learning outcome from using learnpython. The pre- post-test control group design was used as described in chapter 2.3, to gather empirical data on the learning outcome. 57 students were recruited to conduct the experiment. There were differences in both educational background and prior programming experience. Some of the subjects had never programmed before, while others had been through an introductory programming course using MatLab in ITI, two years ago. None of the subjects were actively programming as part of school work or during their spare time however. The overall programming skill level among the test subjects were low. The subjects were recruited from student organizations, both social and athletic. The subjects were not payed individually or based on their individual performance during the experiment. Instead, the organizations were payed a fixed amount of NOK 225 or approximately USD 27 for each student who participated in the experiment.

The experiment lasted 90 minutes in total. The subjects were given 15 minutes to complete each test, and 50 minutes of learning time in between the tests. The remaining 10 minutes were used to organize the tests and different treatments (see the outline presented in Table 3.1). The pre- and post-test consisted of nine tasks each, with the possibility to get three points on each task (27 points maximum). This assessment system gave some leeway to attribute points, even though the answer might not be completely, syntactically correct. The tests used are included in appendix A. Subjects were not allowed to communicate with each other during the tests, nor use any other aids. To follow a blind approach, the number identifying which group each test belonged to was hashed. Having no idea which test belonged to which group, helped to assure that the tests were assessed in an equal and fair manner. After all of the tests were graded and the results noted, the hashes were reversed to identify the group affiliation of each test subject.

Activity	Time
Introduction	4 min
Pre-Test	15 min
Setting up learning activities	5 min
Learning Activity	50 min
Post-Test	15 min
Completion	1 min
Total:	90 min

Table 3.1: The timeline for the experiment execution

As mentioned, the experiment followed a pre- post-test control group design. The participants were randomly assigned to one of three groups. Group 1 (G_1) were to use learnpython as the primary way of gaining knowledge of programming in Python. Group 2 (G_2) were to use relevant parts of a textbook² on the subject. Finally, the control group (G_3) would not perform any activities related to programming. Both learning groups were allowed to use the Internet as a tool. The Internet is a natural tool for students to use when learning. The decision to allow it was a conscious choice made to resemble a real learning situation.

²The complete book is available at https://en.wikibooks.org/wiki/Non-Programmer's_Tutorial_for_Python_3

The pre-test and the post-test were designed in a way that made them slightly different from each other. Most of the questions were similar, having different parameters, or other small modifications that made them distinct from each other. When answering such tests, it is possible that subjects might learn simply from doing the test. This leads to an effect where answers to the post-test scores may be better. Although having slightly different test does not remove the problem completely, it does make the learning effect less prominent. Additionally, having a control group helps to negate the issue.

It is important that the control group does not partake in any learning activities during the experiment. Given that the subjects take a pre-test, it might come naturally for curious minds to attempt to find the solutions to the problems they faced in the test. In an attempt to avoid that members of G_3 attempted to gain new knowledge, the subjects were asked to compete in an online game³. The game is in no way related to programming, and the effort was merely an attempt to make them think about something else than the test they had undergone.

3.3.1 Statistical Methods

Performing such experiments usually entails the use of some statistical analysis method to analyze the results. To find a suitable analysis strategy, it was important to do research on different statistical analysis methods. This was done to discover which methods fit the data set at hand, as well as which methods were relevant and fit the needs of this experiment. Eventually, three statistical methods were found that suited this experiment; t-test, Analysis of Variance (hereby ANOVA) and Analysis of Covariance (hereby ANCOVA) [24], [25]. The following sections elaborate upon the strengths and weaknesses of the different methods. To get a fair understanding of the methods, the structure of the data set generated in this particular experiment is presented. The data set consists of four variables; two nominal values and two continuous values. These values are id, group, and pre- and post-test scores respectively. The interesting aspect to analyze is the change in scores for each subject between the pre- and post-test, along with the mean differences between groups. Different analysis methods have different assumptions about the data set, and the choice of

³The game is available at <http://paper-io.com>

method is highly reliant upon these assumptions. Choosing an inappropriate method will often lead to errors when assessing the null hypothesis. Therefore, the statistical methods were analyzed in order to find the most suitable method for the statistical analysis, with respect to the data set.

T-Test

A *t*-test is a popular and versatile statistical method. With respect to a continuous dependent variable, it determines whether the means of two independent groups are statistically significantly different from each other. Additionally, it outputs how significant the differences between the groups are. That is, the probability of the differences occurring due to chance. A *t*-value is a ratio that describes the difference between the two groups and the difference within each one. Larger *t*-values indicate that the groups differ. Consequently, smaller values indicate that the groups are similar. However, the *t*-value alone is not enough to draw conclusions about the difference between the groups. Every *t*-value is related to a *p*-value. The *p*-value describes the probability that the observations made are the same, or more extreme than actual observed results. In other words, when assuming that the null hypothesis is true, the *p*-value indicates the probability of a difference that large occurring *p*-% of the time. The *p*-value threshold is typically chosen based on the consequences it would have if the observations do not match the actual observed results. For instance, low *p*-value thresholds are often used in the medical field, due to the consequences it might have to draw wrong conclusions about the effect of certain medications. In most other fields, however, a *p*-value of 5% is commonly used and accepted unless there is a specific reason for setting a different threshold [26].

ANOVA

ANOVA is a common term used to describe several, different methods of statistical analysis. Generally, an ANOVA uncovers mean differences between different groups (between-subjects) or differences within a group (within subjects). To test for these differences, statistical significance is considered, which is obtained from calculating a *p*-value. This *p*-value is the same as the one in a *t*-test. As opposed to *t*-tests, ANOVA's are desirable to

use when considering more than two independent variables. They are more conservative, thus reducing the chance of making type-1 errors. That is, rejecting the null hypothesis even though it should, in reality, have been accepted. Similarly to the t-test, an ANOVA also tests for differences under the assumption (null hypothesis) that means between all groups are equal ($G_1 = G_2 = G_3 \dots = G_n$)

ANCOVA

ANCOVA is used to describe a general linear model blending ANOVA and regression. In its essence, ANCOVA is used to express the effects of a categorical independent variable consisting of multiple levels with regards to a dependent variable, while controlling for a covariate. The covariate acts as a control variable for the dependent variable, serving as a way to interpret the dependent variable based on the value of it. The regression part of ANCOVA is basically the relationship between the dependent variable and the independent variable. ANCOVA is preferred and suitable when there is a strong correlation between an independent variable and the dependent variable. If that is not present, regression curves become non-linear, invalidating an assumption that is required to get valid data from the ANCOVA.

3.3.2 Method Selection

ANCOVA would use the pre-test as a covariate, meaning that one would consider post-test scores while controlling for the pre-test scores. Having subjects with different backgrounds and learning ability leads to a high probability of getting varying scores on both pre- and post-tests. This could, in turn, invalidate the method, as ANCOVA has an assumption that requires the covariate to be linearly related to the dependent variable (post-test scores). Due to the differences described above, this would fail in situations where there are large differences in performance.

A one-way ANOVA is generally used to calculate differences in a dependent variable with three or more groups, with one independent variable. Using this method, the change scores are considered to be the dependent variable, while the different learning methods are considered the independent variable. A t-test could also successfully be used as an

alternative to a one-way ANOVA, by testing each group's relationship to one another. A t-test is less conservative than an ANOVA, thus leading to a larger chance of making type-1 errors. The choice fell on using a one-way ANOVA. The dataset matches its assumptions, and it is more relevant than an ANCOVA in terms of desired output. The method is also widely used in similar experiments, the output is concise, and the ways of reporting the results are well documented.

3.3.3 Null Hypotheses

Using statistical methods for analysis usually entails presenting one or more null hypotheses. These hypotheses serve as a default position, where accepting them would mean there is no relationship between measured phenomena. Table 3.2 shows the null hypotheses proposed for this experiment.

H_{0_1} :	There is no significant difference in mean pre-test scores across all three groups (i.e., all groups performed equally well on the pre-test).
H_{0_2} :	There is no significant difference between pre-test and post-test scores for any of the groups (i.e., none of the groups learned anything).

Table 3.2: The proposed null hypotheses for the experiment

The first hypothesis is concerned with whether each group is approximately equally able prior to the experiment. Randomly assigning subjects to groups should, in theory, ensure that they are. However, either group might get students with more knowledge by chance. Although it is not vital to the validity of the ANOVA, having equally able groups is desirable to be able to describe the learning effect between groups more accurately. The second hypothesis defines a null result in ANOVA. Accepting it means that no significant learning effect for any of the groups can be concluded with. Several alternative hypotheses can be proposed to H_{0_2} , in different directions. For instance, post-test scores could be better or worse for either of the groups, or some group will perform better than another group. Presenting possible alternative hypotheses is of less importance as there are many possibilities, and conclusions can be drawn from the results regardless of having presented them beforehand.

Chapter 4

Development

This chapter comprises the development of learnpython, which was created as a vehicle for the research project. The primary focus of this chapter is on the technological and architectural decisions that was made, including the reasoning behind these decisions. The majority of the development was done during the fall of 2016, after the initial prototype testing. The inherited prototype is presented at the beginning of the chapter, as well as any differences between the prototype and the system developed. Architectural decisions and the choice of technology is then discussed. Finally, a description of how the system is used is included.

4.1 Inherited Prototype

The web application created by Nygård during the spring of 2016 forms the foundation on which learnpython is built [5]. The core code and concept of Nygårds software is still present in the new software, but due to architectural changes, some elements have been completely changed and adapted. The changes were primarily made for two reasons; to suit the needs of this research project, and to adapt to the feedback received from the initial prototype testing. Nygårds software was built with Django¹, a Python-based web framework for developing web applications (elaborated upon in section 4.5.2).

¹Available at <https://www.djangoproject.com>

4.1.1 Differences from the Inherited Prototype

There are several differences between learnpython and the inherited prototype. Some of the changes are based on known issues in the prototype, others on the results from user testing, and some to comply with the needs of this research project. Finally, some additional features were important to have, because the system was to be released to students, allowing them to use it outside of a controlled environment. First of all, several new assignments were made, some of which were targeted towards exam practice. Due to students being nearly done with ITI, these assignments were more comprehensive and made to be similar to assignments that might appear on an exam. A form where users could submit issues they found with the system was created, to be able to identify and fix critical errors in the system quickly. Having the wish to track the amount of users, as well as how many assignments were solved, user registration was introduced. The user interface and button layout were changed somewhat, to address issues found during user testing of the inherited prototype. To reduce page reloads and get output and stack traces from the native Python interpreter, the architecture had to be changed.

4.2 Requirements

Table 4.1 shows the non-functional requirements that were emphasized when making decisions about the architecture of the system, as well as which technologies and frameworks to use. The different requirements are referred to later when discussing these decisions, as a way to link the decisions to the requirements.

Availability is important in this context, as the system is a web-based solution. Users typically expect to be able to access a website at any time, and potential downtime might drive users away from the site. The modifiability requirement is in this case mainly for development purposes, as users do not directly make modifications to the system. If someone else inherits the code base in the future, a modular and structured code base makes the transition easier. There is some focus on performance, in particular on the feature that allows users to compile their code. Response time should be kept to a minimum, while not negatively affecting other requirements. Usability is important in this case and plays a

ID	Requirement	Description
NFR1	Availability	Availability is a measure of the degree to which a system is operable at any time. In this context, it can be seen as the uptime of the system.
NFR2	Modifiability	Modifiability is the degree to which a system can easily be changed, either by the creator or a user.
NFR3	Performance	Performance is the amount of work that can be done by a computer. Typically measured in response time, throughput, low utilization of computing resources, etc.
NFR4	Usability	Usability describes to which degree a system can be used by a user group, to perform certain tasks and successfully accomplish objectives effectively.
NFR5	Security	Security is the degree to which a system is protected from unintended or unauthorized access, as well as protected from harmful change and destruction.

Table 4.1: The derived list of non-functional requirements for learnpython.

significant role in the project as a whole. It is vital that users are able to understand how to use the system properly.

When designing a web application, multiple security risks need to be considered. It is important to protect the personal information of users and the integrity of the system. First of all, running code server side poses a risk of malicious code being executed. This could potentially harm the system in multiple ways; either leaving it unable to answer requests from other users or destroy the server operating system entirely. Another risk to address is SQL (Structured Query Language) injections. These kinds of attacks are performed by a user who executes malicious SQL-statements to get data (e.g., usernames and passwords) from the database. SQL injections can also potentially wipe or destroy an entire database. All data sent over the Internet using HTTP (Hypertext Transfer Protocol) are subject to man-in-the-middle attacks (most commonly eavesdropping), as the data exchanged is unencrypted.

The system created as part of this project can hardly be defined as a complete solution. The project focused on the students using the system and the effect it had on them. In a complete solution, there should be some way to allow the academic staff to be able to create assignments within the system easily. It is possible as of now, but it is a time-consuming process to create each assignment in the Django administration panel individually. Hence, it should be possible to submit assignments through JSON (JavaScript Object Notation)

or XML (Extensible Markup Language) and possibly allow users to submit assignments themselves, which the staff can evaluate and possibly include. To reduce the time needed to create single assignments further, a solution for creating parameterized assignments using a template could be incorporated.

4.3 Architecture

Separation of responsibility was deemed the most important factor when choosing an architecture. To achieve high modifiability (NFR1) and modularity, the architecture should distinguish between the presentation layer (hereby frontend) and the data access and modification layer (hereby backend). Due to the nature of pure Django applications, these layers are tightly coupled. Django is implemented as an MTV-architecture – that is, model, template, and view. The architecture defines the model as an abstraction of the database layer, including a programmatic representation used for modification and fetching of data. The template is equal to the frontend (view in MVC-architecture). All logic regarding the presentation and visualization of data is performed within this layer. The business logic is located in the view (similar to the controller in MVC-architecture). It is used as a bridge, connecting the model and the template, handling both fetching and updating of the model.

The MTV-architecture and Django allow developers to develop stable applications quickly. On the other hand, there are limitations when it comes to developing comprehensive and dynamic web applications. To address these issues, a variant of the client-server model was used to develop learnpython. Figure 4.1 displays a simplified model of the software architecture. By moving the frontend into a separate web application, it was possible to create a single maintainable service, handling only the view part of the architecture. This layer was created using React², a JavaScript-library. The backend of the project was created as a REST (hereby Representational State Transfer) web service. This was achieved by using Django in addition to the third-party library Django REST-framework. Django REST-framework replaces the views in Django with its own view sets that serialize models and returns a RESTful representation. The communication between the two layers is made possible through HTTP requests. The client issues HTTP requests

²Available at <https://facebook.github.io/react/>

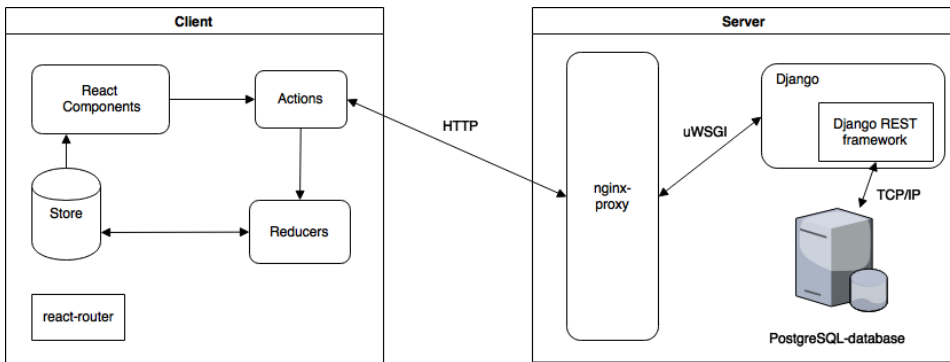


Figure 4.1: A simplified overview of the system architecture.

through what is called actions in React, to which Django sends responses back. Django accepts the HTTP requests through its alternated views. All the data is stored in PostgreSQL, an open-source relational database. Alterations and fetching of data from the database are done by the model.

4.4 Development Strategy

The development strategy used during the development was quite simple. Having two developers with constant contact, and an architecture enabling separation of the frontend and backend, a formal development methodology was deemed unnecessary. The project was split into tasks, to obtain an overview of what was to be done, and to keep track of how much time was needed. The tasks were a way to ensure that the software was good enough to justify distributing it to ITI students.

Version control is important when working on development projects. Git³ was used in collaboration with GitHub⁴. GitHub is a cloud hosted, Software as a Service (hereby SaaS), distributed revision control system that allows team members to access the code. Git provides a complete history and version control, with complete offline support, as the Git working directory acts as a complete repository. GitHub was important both to have a backup, and to have a history of code that had been added. Additionally, branches were

³Available at <https://git-scm.com>

⁴Available at <https://github.com>

used to develop new features, to keep the core code clean and free of errors.

4.5 Technology

Not being able to execute native Python code in the browser was one of the issues in the inherited prototype. Nygård introduces the idea of executing Python scripts inside isolated Docker containers. The main advantage of executing code within such a container is the possibility to execute code with the actual Python interpreter. In the inherited prototype, the code is executed by transpiling the program to JavaScript before execution. During the initial testing of his prototype, it became apparent that this approach introduces the possibility of faulty transpiling. Writing faulty code also causes issues, as error messages are from JavaScript, not Python. For the novice programmer attempting to learn Python, this output is more difficult to understand, as it is syntactically inconsistent with regular Python output. Assessing this issue was one of the main factors when choosing technologies, as it directly affects the usability (NFR4) of the system. An example of the different outputs from transpiled JavaScript and Python can be seen in Figure 4.2.

```
Traceback (most recent call last):
  module editor line 31
    exec(src, ns)
  module editor line 31
    exec(src, ns)
  module exec_43 line 4
    printf(4,5)
  module editor line 31
    exec(src, ns)
  module editor line 31
    exec(src, ns)
  module exec_47 line 4
    printf(4,5)
NameError: printf
```

(a) JavaScript

```
Traceback (most recent call last):
  File "pyt.py", line 4, in <module>
    printf(4,5)
NameError: name 'printf' is not defined
```

(b) Python

Figure 4.2: Examples of the different error messages from transpiled JavaScript (a) and the native Python interpreter (b)

Being able to reuse code was deemed beneficial to save time during development. Having a functioning prototype and its source code serving several of the same features planned in the system affected the choice of technologies. Apart from code reuse, each technological choice has advantages that are useful in this project. The full list of technologies is described in the following sections.

4.5.1 React

React is an open source JavaScript library developed by Facebook. React was created as a library for developing user interfaces with continuously changing data. Also, React introduces a virtual document object model to increase the speed of dynamical web applications. In React, the user interface is divided into reusable components, all encapsulated with their individual state and responsible for their individual representational output. That makes it easy to change or replace certain components, without affecting other parts of the system, making the frontend easily modifiable (NFR1). One can think of a component as a mathematical function $V = f(d)$, accepting an input d , and returning an output V . The final user interface are composed of all the individual components. The data, d , is provided to a component either directly from a state store or through properties (called props in React) passed into the component. These components are declarative, which means that components update when new data is provided. The frontend becomes dynamical, modular, and separated from the backend.

React makes it easy to build a dynamic user interface responsive to user input. When creating the feature that allows users to interpret and execute Python code, that comes in handy. Whenever code is executed, the output component changes to a "compiling" state and displays a circular progress bar. As soon as the server returns the code output, the component updates to display the output. All of this happens without the need for a page reload, which makes the user interface seem continuous. Building the frontend in React makes the code execution feel more native in the browser. Performance (NFR3) is also improved, as the responsible component is the only part of the document object model that needs to re-render when receiving the data.

4.5.2 Django

Django is an open-source web framework for creating web applications. When used together with the Django REST-framework, it becomes a RESTful application. A powerful object-relational mapper (hereby ORM) is included in the core of Django, which is a code library that automates the communication between objects, classes, and data stored in a relational database. The ORM allows developers to write queriesets (Python code) instead of raw SQL-queries to perform operations on the database. SQL-statements created by the queriesets are escaped prior to being executed, which is a security advantage, as it protects against SQL injections (NFR5).

Django REST-framework is an extension to Django, which provides code for serializing data. It extends Django in a way that makes it possible to build a RESTful application programming interface out of the Django ORM data. Django also includes an administration panel by default. The panel can display data from the database to a user flagged as an administrator, as seen in Figure 3.2 in section 3.2.2. The functionality is handy when analyzing data and usage statistics. Writing user events to the database enables the possibility to track what each individual registered user has done in the system, e.g., the number of solved tasks. Adding Django REST-framework to Nygårds software did not require extensive alterations to the core code. The main change was replacing the pre-compiled HTML views presented by Django with data serialized through REST, using Django REST-frameworks view sets.

4.5.3 Docker

Docker⁵ is an open-source software used to create containers. Containers are single, stand-alone executable software with a minimal footprint. A container includes all that is needed to execute it, like required system libraries and third-party software. It has advantages such as resource isolation and allocation, security, and a self-contained system, without needing to bundle a full operating system. A container is similar to a virtual machine but shares the operating system kernel with any other containers that might exist. One of the main advantages of Docker is that containers are created using immutable images of the required

⁵Available at <https://www.docker.com>

system libraries. As a consequence, the container is guaranteed to produce the same result if the container is executed twice or more, which is important for this system. If a user executes the same code twice, the output of the executed code needs to be the same, and not be affected by errors or other factors in the container. Another advantage of Docker is that the containers may be completely isolated, disconnected from any network and given a certain amount of hardware resources. As a result, it is possible to execute user scripts within a container, without that being a security risk to the origin operating system. The containers may also be given a timeout threshold as an argument when starting. If set, this argument will make the container stop executing when reaching the threshold, ensuring that no code (e.g., infinite loops) will block the origin operating system from performing other tasks. Both the isolation of containers and the timeout threshold helps mitigate issues with security (NFR5), in particular running malicious code or access the origin operating system.

The simple container created for executing user scripts in this project could be started, completed, and removed within approximately 600 milliseconds on the hardware used in this project. On average this has given a response time of approximately one second, where the last 400 milliseconds are HTTP requests, as well as a small overhead with database calls. The response time is measured from the initial HTTP request for code execution, until the response is visible to the user. Although using containers slightly reduces performance (NFR3) as opposed to transpiling, it has a clear advantage both in terms of usability (NFR4) and security (NFR5).

4.5.4 Scaleway

Scaleway⁶ is an Infrastructure as a Service (hereby IaaS) platform. The service provides virtual private servers (hereby VPS) to the consumer market. By using this service, the time spent on infrastructure could be kept to a bare minimum, freeing up time to focus on development. An advantage of using IaaS is the ability to re-size VPS' during heavy load. During the initial HTTP load-testing and benchmarking of learnpython, it was apparent that the system could be resource demanding during heavy load. The primary reason

⁶Available at <https://www.scaleway.com/>

being launching, executing user scripts, and shutting down containers. Even though the image of the containers are the same, the process of starting and stopping the containers was demanding. By using a VPS, this issue could be solved by scaling the VPS' available hardware resources, thus avoiding a potential server crash during heavy load and ensuring system availability (NFR1).

When using HTTP, all of the data that is exchanged between a client and a server are unencrypted. To mitigate that issue, the connection was secured using an SSL (Secure Sockets Layer) certificate, enabling HTTPS. Also, enabling HTTP Strict Transport Security ensured that all non-HTTPS communication were prevented by the client after the initial request. HTTPS ensures that all data sent are securely encrypted (NFR5). Even if somebody managed to access the connection (e.g., eavesdropping), they would not be able to decrypt any of the data.

4.6 Using the System

Upon accessing the system for the first time, the user is presented with a simple sign in form. Users are forced to register to use the system. The registration is done at the `/register`-route, available through a button. During registration, users must mark whether or not they accept to receive a questionnaire after using the system. After successfully registering and signing in, the `/start`-route is presented, as shown in Figure 4.3. This page shows the different quizzes the user may start and simple statistics of assignment streaks.

The statistics display how many assignments the user has completed in a row and the highest streak of correctly solved assignments. The available quizzes are a general quiz within different subjects and a quiz with exam questions. The general quiz consists of introductory tasks within the subjects loops, control structures, and functions. The exam questions are more difficult, requiring users to combine knowledge from a broader set of subjects. The system contains 45 assignments which are presented in a random order from the set of subjects that the user has selected. When having solved all assignments in a given category, the list of solved assignments resets, meaning that the user will receive the same assignments again in a new, randomized order.

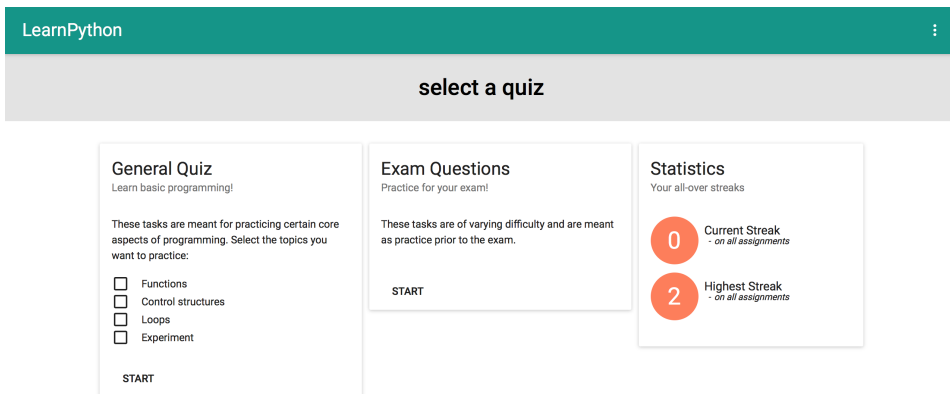


Figure 4.3: The landing page, with the quiz selector in learnpython.

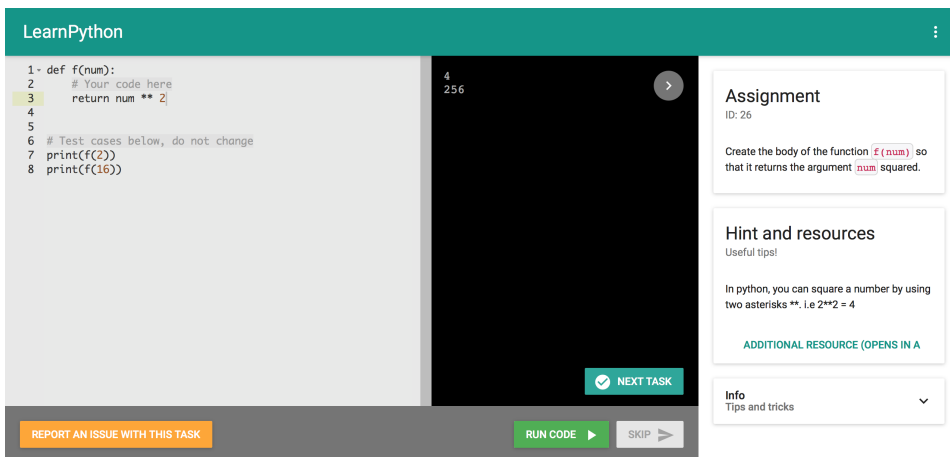


Figure 4.4: Presentation of an ongoing quiz in learnpython.

When a quiz is selected and started, a `startQuizRequest`-action is dispatched in React, and an assignment from the quiz is requested from the server. The action marks the start of the quiz loop, as presented in Figure 4.5. From here on, and until exiting the quiz, the system loops this sequence.

The quiz is presented through the `/quiz`-route, as seen in Figure 4.4. There is an editor on the left side of the view, where users can write code. The editor has syntax highlighting for Python, but auto-completion is purposely disabled. The output console

is located to the right of the editor. The console displays all output (both results and errors) from the code execution, as well as a loading indicator while executing the code. In the case of a code execution timeout, a timeout error is displayed in the console. The assignment text itself is located furthest right, presented within a collapsible column. Users may choose to close the assignment text, to enlarge the code editor and to remove any distractions from the programming itself. A taskbar is located at the bottom of the page. The taskbar may be used to report issues with the specific assignment, execute code, and to skip the assignment. When a correct answer to an assignment is submitted, a new button appears, used to fetch a new assignment.

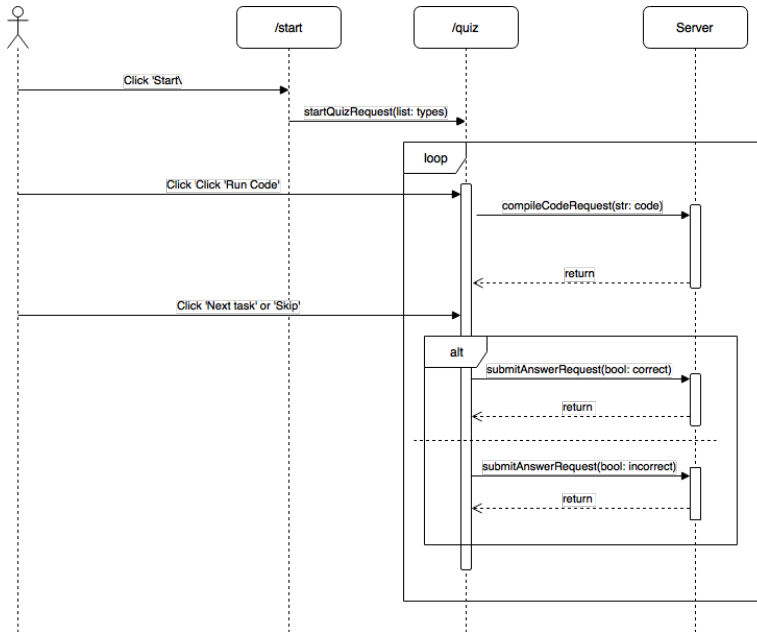


Figure 4.5: Sequence diagram of a quiz execution.

Results

This chapter contains the results from all stages of the project. The first part describes the feedback gained from user testing the software created by Nygård. Following that is a presentation of the quantitative data gathered from distributing learnpython to ITI students. The data comes from remote evaluation, the Django administration panel, and questionnaires. Finally, the results from the experiment are reported.

5.1 Initial Prototype Test Results

The results presented stem from the user tests done with Nygård's prototype at the start of this project. A total of five students were recruited for the tests. The problems shown below are the things noticed by test subjects that were deemed most important to the usability of the software.

Run and Submit

When a user attempted to execute code that was the correct answer to an assignment, the submit button in Figure 5.1 would turn green, meaning that the assignment was successfully completed. The majority of users did not notice this, however, leading to confusion regarding whether or not they could submit their answer. The test subjects were simply

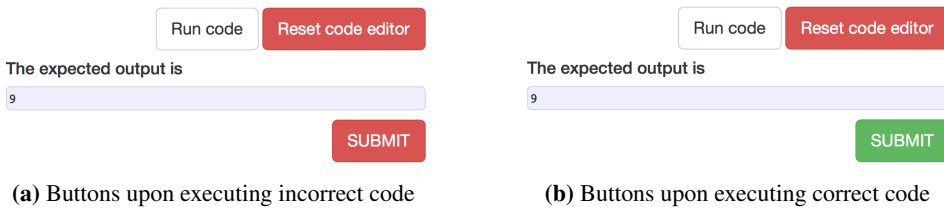


Figure 5.1: The "Run" and "Submit" buttons in the initial prototype by Nygård.

not able to tell whether their answer was correct after executing their code. There should be a clearer indication of whether the submitted code is correct. Some subjects suggested having a *Next Task* button appear when they had submitted a correct answer. Some also wanted a clearly defined way of skipping a question, in case they were unable to answer it.

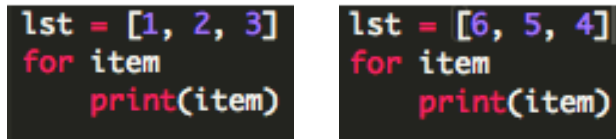
Achievements

When questioned about the achievement system and the motivational factors related to it, only one of the five subjects was fond of it. The rest had a neutral stance on the matter, saying that it did not influence the system in any meaningful way. One of the subjects stated that to be meaningful, there should be some broader context to the achievements, making them more exciting. One of the suggestions was letting users compare themselves with other users in terms of which achievements they had gathered. A leaderboard could be implemented to make that possible.

Assignments

The majority of users complained about the assignments being repetitive. During the test, some assignments could be presented to the user as many as three times. This made some of the users unsure about their progress. Also, some assignments had subtle parametric differences that the users did not notice, as seen in Figure 5.2.

The majority of the test group had no issues with the first assignments that were presented to them. This was discussed in the interview afterward, revealing a wish for more control over the difficulty of the assignments that were presented to them. Although they had the option to choose the topic of the assignment, they also wanted some way to choose



```
lst = [1, 2, 3]
for item
print(item)
```

```
lst = [6, 5, 4]
for item
print(item)
```

Figure 5.2: An example of simple for-loop assignments with subtle parametric differences, in the initial prototype by Nygård

difficulty level themselves. Finally, users wanted assignments which to a larger degree combined different topics. Being stuck with assignments that only focuses on one topic was boring to users in the long run.

User Experience

Generally, test subjects were satisfied with the general look and feel of the site, apart from the run/submit functionality. There was, however, an issue with hints and additional resources, where users having screens with low resolution were required to scroll down on the page in order to see it. One of the testers did not notice this feature at all during the testing.

5.2 Test Results from Distributing learnpython

After the changes from the initial prototype test had been implemented, the newly adapted system was hallway-tested to identify bugs and unintuitive design decisions. When the tests yielded satisfying feedback, stable enough for distribution, it was distributed to the students enrolled in ITI. The usage of the system was carefully monitored through the distribution period. Table 5.1 contains some key data gathered during the period the system was in use by students.

The table shows that 88 students chose to try the system over the 13-day distribution period. Each user spent 37 minutes using the system on average, out of everyone who visited the site. If only those registering a user are accounted for, the average user spent 1 hour and 24 minutes with the system as their learning method. In total, the system had 190 unique visitors over the 13 day duration, explaining the large difference in average time

Data	Value
Total amount of days in distribution:	13
Total amount of registered users:	88
Average time spent per user, counting all users (190):	0:37:25
Average time on site per user, counting registered users (83):	1:23:54
Total amount of assignments attempted to be solved:	1507
Total amount of assignments correctly solved:	729

Table 5.1: Key usage numbers after the distribution of learnpython.

spent on the site. The total amount of assignments attempted was 1507, with 729 being correctly solved and 778 skipped.

5.2.1 Usage Statistics from learnpython

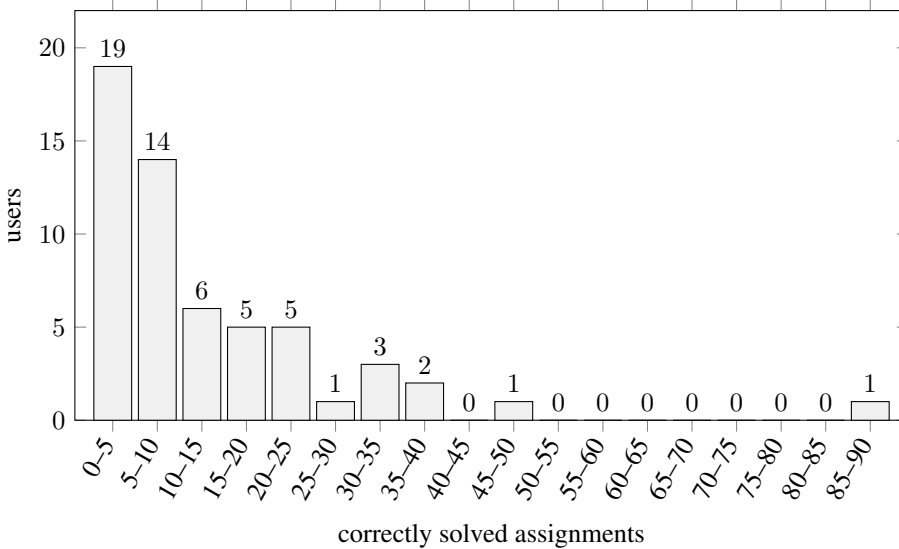


Figure 5.3: Histogram presenting the number of assignments solved by users during the distribution of learnpython. The x-axis indicates the amount of assignment solved in segments of five. The y-axis indicates the number of users within a given interval.

Figure 5.3 is an overview of how many assignments the users solved, only accounting for those who solved at least one assignment correctly. The x-axis indicates the number of assignments solved in segments of five, while the y-axis indicates the number of users who fell into these categories. A substantial amount of users solved few assignments and fell off

quickly. The majority of users did, however, solve more than ten assignments. One person solved as much as 87 assignments correctly (with a total of 129 attempts), meaning that the user solved every assignment in the system more than once. Out of the 88 registered users, 24 users registered an account, but never attempted to solve any assignments.

5.2.2 Progress Questionnaire

The following bar charts show the results gathered from the questionnaire that was distributed to students having successfully completed five assignments. In total, 34 people answered the questionnaire out of the 38 that successfully completed five assignments. While the data from section 5.2.1 is gathered from all users, this questionnaire was presented to users upon solving five assignments. The questionnaire consisted of four questions, two of which are related to students background and two of which are related to motivation and relevance of the system.

Primary Way of Learning

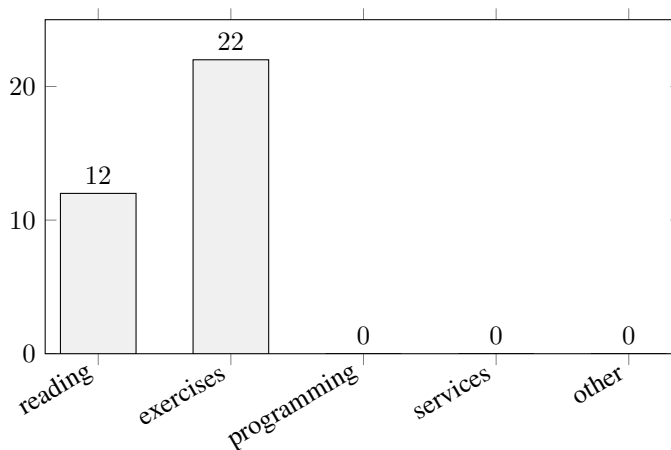


Figure 5.4: Histogram of the result from the question about the students primary way of learning.

This question was related to the student's way of gathering new knowledge and learning. It states *"Over the course of this semester, what was your primary way of learning Python?"* Figure 5.4 shows that all of the students who used the system learned from either reading

course material or the exercises. The students were given five alternatives; reading course literature, solving mandatory exercises, programming aside from the mandatory exercises, services like Codecademy and Project Euler, and lastly – other resources/techniques.

Prior Knowledge Level

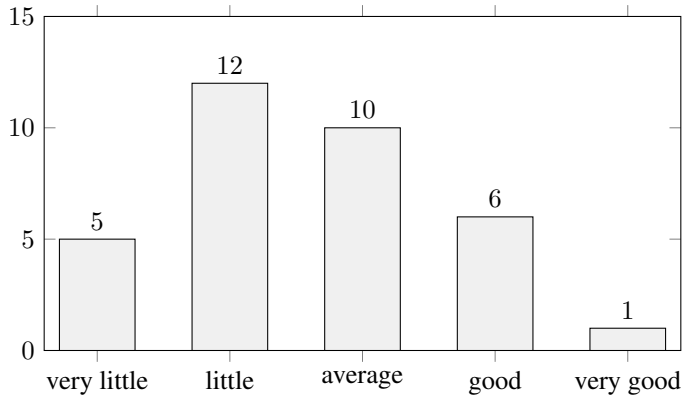


Figure 5.5: Histogram of the result from the question about the students prior knowledge level.

The question was *"On a scale from 1 to 5, how would you rate your own skill level in programming?"* Asking this question is an attempt to map the student's knowledge of programming. The point of it is to find out if the system appeals to a wide array of students, or if it only appeals to one side of the scale. The results are presented in Figure 5.5. The distribution of users is approximately normalized, meaning that users with different programming skills have used the system.

Motivation from Solving Assignments

The next question asked was *"When you successfully complete a task, to what degree does that motivate you to keep working?"* The meaning of this question was to get an indication of how motivated students get by completing assignments. The question was formulated in that manner, as successfully being able to solve assignments is seen as the primary motivational factor in the system. The degree to which students are motivated by solving assignments is a key element to the success of this system. The results are presented in

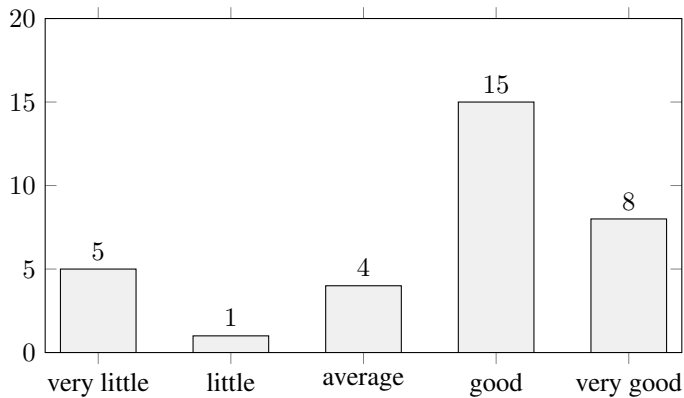


Figure 5.6: Histogram of the result from the question about the students motivation from solving assignments.

Figure 5.6, and show that the majority of users find it motivating.

Relevance of the Software

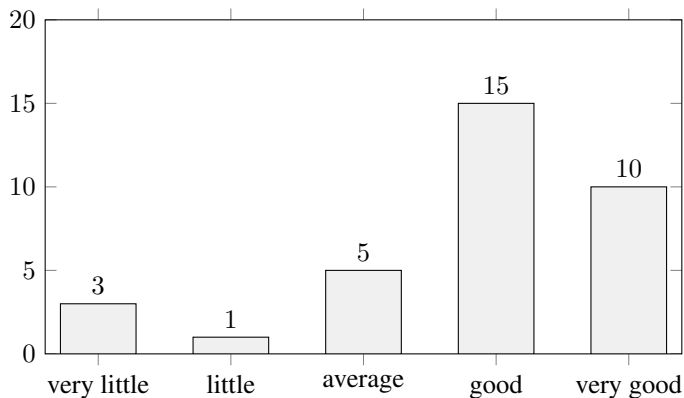


Figure 5.7: Histogram of the result from the question about the relevance of the system.

This question concerns the relevancy of the website with regards to the ITI course. The question asked was *"On a scale from 1 to 5, to which degree do you feel this software is relevant to learning Python?"* This question addresses the usefulness of the system, as perceived by the users. If students do not feel that the system is relevant, it is unlikely to be well received. Figure 5.7 shows the results from the question.

5.2.3 Questionnaire Distributed After the Final Examination

17 students answered the questionnaire out of the 38 it was sent to. Overall, their responses indicate that the system is indeed useful. There are also some issues that should be fixed in future iterations of the system, elaborated upon in section 6.5.2. The questionnaire was short and emphasized aspects of the system that could have been improved and aspects that were good.

Points of Improvement

The open question regarding what could have been improved in the system shows that more work should be done creating assignments. Several of the answers indicate that the assignments were not well enough thought through, and lacked better context, such as hints. Feedback such as *"Examples of solutions to similar assignments"*, *"More different types of assignments"* and *"A lot of the assignments were very similar"* shows the importance of good assignments and providing context to them.

Several students also complained that even though they had the correct answer to an assignment, the system did not accept it as correct. This stems from how the evaluation of the answers was implemented. It only compares answers to the solution, meaning that the solution needs to be exactly right. As an example; if a user submitted 10.0 as an answer, the system would interpret it as wrong if it expected the number as an integer (10 in this case). The same logic applies for uppercase and lowercase letters, incorrect orderings of numbers in a list, and so forth.

Particularly Liked Features

This open question asks if there are any features that users liked in particular, and is meant to provide some clues to what makes such a system a viable way of learning. Feedback such as *"The learning method in general, works better than lectures"* and *"It is nice being constantly provided with new tasks in an online editor, I learned a lot"* suggest that users enjoyed it as a learning method. Additionally, the feedback suggested that the exam questions were helpful to practice for the exam, mainly because they were similar to questions that tend to appear on exams. Finally, several of the respondents emphasized

that they liked the user interface, and that it was easy to use.

5.3 Experiment Results

Figure 5.8 displays a scatter plot of the pre- and post-test results. The raw data used to generate the scatter plot is included in appendix B. Out of the 57 subjects recruited for the experiment, 56 of them showed up. Each mark on the plot represents a subject, where the x-axis represents the score on the pre-test, and the y-axis represents the post-test score. The subjects were spread as evenly as possible between groups, with the learnpython and reading group having 19 subjects, while the control group had 18. Subjects were randomly assigned to groups, and each subject got a note with an identifier and their group belonging at the start of the experiment. In total, it was possible to obtain 27 points on each of the two tests. The scores were transformed into an interval between 0 and 1 for convenience, where 0 is no correct answers, and 1 is all correct. Furthermore, Figure 5.9 show the box plots of the results, displaying the spread of change scores across each group. All of the calculations in this section were performed with SPSS Statistics¹.

Table 5.2 displays descriptive statistics generated by SPSS. The table shows the mean results of both the pre- and post-test, along with the change score, for all groups. Additionally, it displays the standard deviation from the mean.

Table 5.4 shows calculations from the one-way ANOVA [27] that were conducted to calculate significance levels between groups. The sum of squares is calculated by taking the squared differences from the mean and summing them. Degrees of freedom (df) is the number of independent factors that went into calculating the results and is shown in Table 5.3. The mean square is calculated as the sum of squares divided by the degrees of freedom. The F-score is based on the ratio of squared means between the between groups and within groups measurements, and is a measure of variance in the data set. Finally, the significance (*p*-value) yields some value that estimates the probability of the results occurring due to randomness, as opposed to the effect the independent variable has on the dependent variable.

¹In this project IBM SPSS Statistics, version 24.0.0.0, was used for statistical calculations. More detailed information about this software is available at <https://www.ibm.com/us-en/marketplace/spss-statistics>

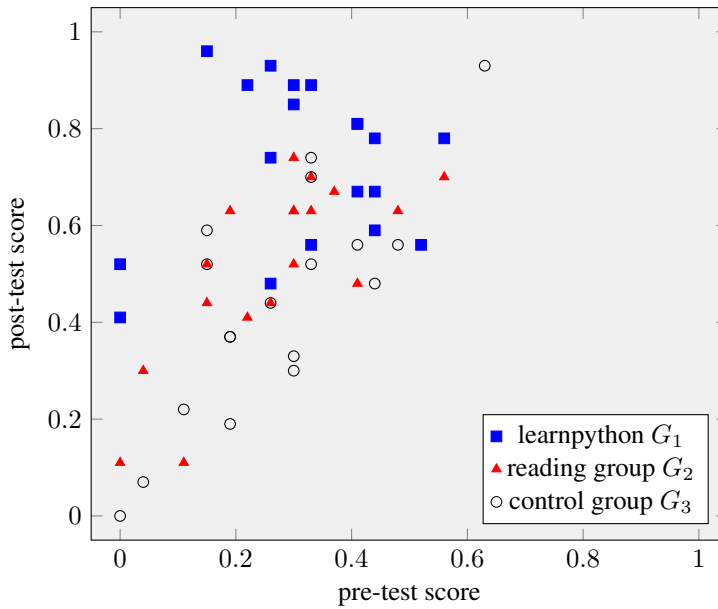


Figure 5.8: Scatter plot of pre- and post-test results for each user, sorted by treatment group. Each point indicates a given subject’s score on the pre-test (x-axis) and post-test (y-axis).

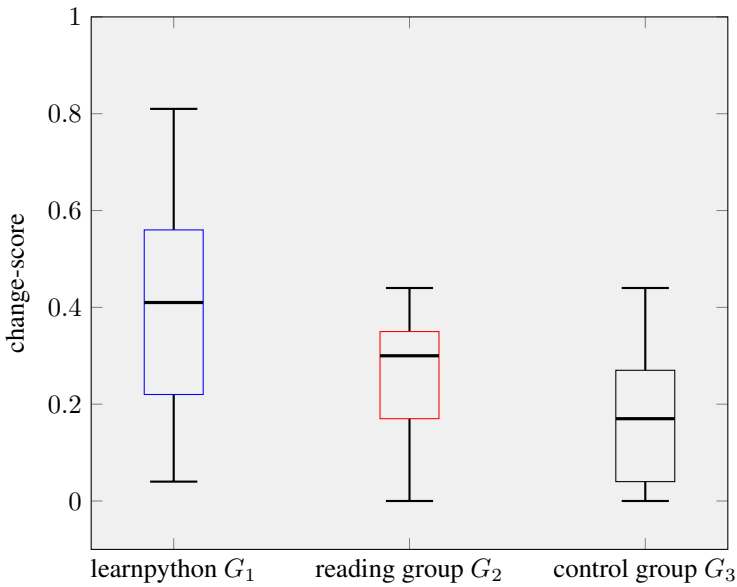


Figure 5.9: Box and whisker plots representing the change score between the pre- and post-test, sorted by treatment group. The whiskers show the minimum and maximum score, the horizontal lines within the boxes show the median, and the boundaries of the boxes show the first and third quartile.

Given these results, it is improbable that all groups learned equally much. H_{02} ("There is no significant difference between pre-test and post-test scores for any of the groups.") is rejected, as $p = .000207$ is below the threshold of .05. Furthermore, there was no large difference in pre-test scores between the three groups (.04% at most). Given that all groups performed approximately equally well on the pre-test, H_{01} ("There is no significant difference in mean pre-test scores across all three groups.") is accepted.

Group		Mean	N	Std. Deviation
learnpython, G_1	post	.7258	19	.16588
	pre	.3179	19	.15285
	change	.4079	19	.20490
Reading Group, G_2	post	.5316	19	.19563
	pre	.2742	19	.14473
	change	.2574	19	.12648
Control Group, G_3	post	.4383	18	.23593
	pre	.2683	18	.15983
	change	.17	18	.15010

Table 5.2: Descriptive statistics; means and standard deviations of standardized test scores

Levenes Statistic	df1	df2	Sig.
2.481	2	53	.093

Table 5.3: Levene's test for equality of variances.

5.3.1 ANOVA Report

The one-way ANOVA was conducted to determine if all learning methods were equally effective, which was determined by change scores between pre- and post-test. The box plots in Figure 5.9 show no outliers, calculated by interquartile ranges. Levene's test shows that there were homogeneity of variances ($p = .093$), which is above the .05 threshold. The change scores were statistically significantly different between the learning methods, with an F-score of $F(2, 53) = 9.999$ and $p < .0005$. The ω^2 score was .24, which is a measure of effect size in the population, and was calculated in accordance with [27]. The change scores were .17 for the control group, .26 for the reading group and .41 for the learnpython group, with standard deviations of .15, .13 and .20 respectively. The Tukey-

Kramer post-hoc analysis (see Table 5.5) revealed that the mean increase from control group to the learnpython group (.24, 95% CI [.11, .37]) was statistically significant ($p < .0005$). The increase from reading group to the learnpython group (.15, 95% CI [.02, .28]) was also significant ($p = .018$). The increase from control group to the reading group (.09, 95% CI [.04, .22]) was not significant however ($p = .247$).

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	.538	2	.269	9.999	.000207
Within Groups	1.427	53	.027		
Total	1.965	55			

Table 5.4: ANOVA for standardized test scores

<i>i</i>	<i>j</i>	Mean (<i>i</i> - <i>j</i>)	Std. Error	Sig.	95% Confidence Interval	
					Lower	Upper
G_1	G_2	.15053	.05323	.018	.222	.2789
	G_3	.23789	.05397	.000150	.1078	.3680
G_2	G_3	.08737	.05397	.247	.0428	.2175

Table 5.5: Tukey-Kramer post-hoc analysis of standardized test scores, sorted by treatment group.

Chapter 6

Discussion

This chapter contains a discussion of the results gathered during the whole project period. The start of the chapter discusses the results of the initial prototype test, and the changes done to the system based on those results. Following is a discussion of the results gathered during the days in which learnpython was distributed to students. Both questionnaires, the usage data from the remote evaluation, and the Django administration panel, is evaluated. Then the results of the experiment are discussed, and the concerns with regards to the validity of the experiment are considered. Finally, the potential future work is elaborated upon, and the project as a whole is concluded.

6.1 User Experience

User experience and useful features were some of the primary focus points during the design and creation approach. As previously discussed, it is nearly impossible to build a user base if people do not understand how to use a system. Tending to the wishes of the users during the first user test allowed the creation of a more intuitive system. The hallway tests yielded results that were sufficient to justify distributing the system. No critical errors were found, and the test subjects generally found the system useful and easy to use.

6.1.1 Changes Based on the Initial Prototype Test

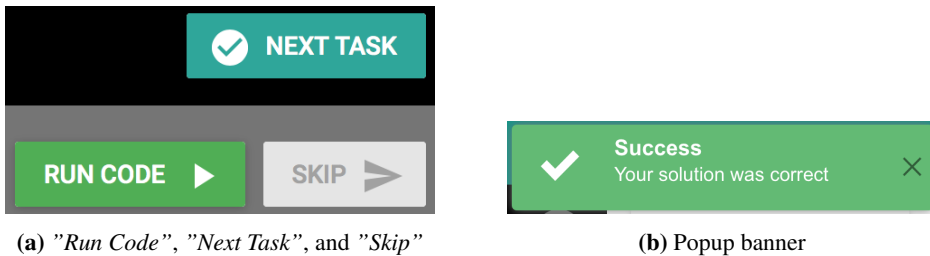


Figure 6.1: Functionality for notifying users about whether their answer is correct.

The suggestions made in this section are functionality that was modified or implemented in the prototype. The changes were based on the results of the user tests described in section 5.1. Architectural decisions were based on the considerations listed in Table 4.1 in section 4.2.

First of all, the functionality for running and submitting code was changed. Figure 6.1 shows how the functionality for running and submitting code is implemented. When a user submits code that is correct, a *Next Task* button appears (Figure 6.1a). A notification also appears in the top right corner (Figure 6.1b), displaying whether the submitted code was correct or incorrect. Additionally, an option to skip the current assignment was implemented, letting users go to the next assignment if they were not able to do the current one. The hints and additional resources section were moved to the right side of the screen, in an attempt to avoid scrolling on the page. Finally, the leveling system was replaced in favor of having basic assignments and assignments targeted towards exam practice.

The changes made to the system was hallway tested on five students. The testing uncovered no issues related to the design of the system, and the subjects were able to understand the system well. Based on these results, the user experience and current features of the system were deemed sufficient for release. Hence, the premise for the research questions as presented in section 1.2 was also seen as fulfilled. There are always certain things that can be improved in a system, but given that all of the test subjects understood the system well, it was deemed sufficient. The rest is related to how well designed the assignments in the system are, and if users are able to solve them.

6.2 Distribution of learnpython

The primary motivation behind distributing the system to students was to gather numerical data about usage. As the results show, there were a substantial amount of users registering, but also quite a significant fall off rate. It is important to remember that the whole idea behind the software is to use it as a voluntary addition to the exercises and curriculum. Although it is positive if a lot of students find the system useful, it is not required that everyone must do so. The most notable discovery during this phase is the fact that the system was able to attract a substantial user base by simple marketing in lectures, in addition to the note posted on the courses wiki page. Given these circumstances, it is fair to argue that having 88 users sign up was a promising result.

The system was available as a website, and it is not guaranteed that all of the people signing up were actually students enrolled in ITI. The system was not marketed to anyone else, and discovering the website without knowledge of its existence would be difficult. The possibility that someone with other backgrounds might have registered is present either way. If it were to be the case, it is unlikely that it represents more than a small percentage of the user base. Thus, it is not considered a significant issue with respect to the integrity of the collected data.

Auto-completion was purposely disabled in the editor. Auto-completion is at its most useful when using extensive class libraries, where it is difficult to remember all available methods. That makes it useful when writing comprehensive, object-oriented programs. For simple programs in Python, which is the current usage area of this software, it is important that students learn through writing their own code rather than get suggested relevant methods and constructs.

6.2.1 Progress Questionnaire

The results presented in section 5.2.2 are positive. Students who used the system were motivated to keep using it and felt that they were learning from using it. It was not limited to students falling within a certain skill category either, as the prior programming skill seemingly ranged across the whole specter. It is important to note that these results are not a definitive indication that the usage of this system is exclusively positive. First of all, the

answers are merely students personal feelings about it, which are limited in terms of conclusiveness. Secondly, only students having completed five assignments were presented with this questionnaire. Some students quit after having completed fewer assignments than five, or no assignments at all in some cases. As an example, it is fair to assume that a student who successfully completed five assignments were more motivated to use it, than a student who quit after completing one. Thus, the questionnaire probably yielded more positive results in its current form, than if it was distributed to absolutely everyone who had used the system.

The overall results are promising. The fact that several users solved a lot of assignments in the system is positive, and an indication of its usefulness to students. This is further backed up by the questionnaire that was presented to them. The majority answered positively to both the question about motivation and learning outcome. As previously stated, these results do not give definitive answers to whether or not this is an effective way of learning. It does, however, prove that several of the students who were willing to try it, actually liked and appreciated the functionality it offers, and thought that it was an effective way of learning.

6.2.2 Usage and Falloff Rate

The system did suffer from users falling off in early stages, as about half of the users who visited the site did not register an account. Furthermore, of the users who registered, almost 20 did not solve more than five assignments, indicating that they did not get much out of their stay. The reason people fall off prior to registration is often related to the effort it takes to register an account, meaning that a lot of users will not bother, unless they know that they will use it. After registering, there is a wider range of possible explanations. One is that they do not understand the system well enough due to the design choices. Another hypothesis is that a user who is not able to solve the assignments they are presented with gets demotivated by that. The data gathered from the database backs up this hypothesis. Several users who fell off early have registered some failed attempts and then stopped using the software. Usage of the system was only monitored for 13 days. Releasing the system earlier could have made more people sign up. Students may feel that trying a

new system is not worth their time so close to the exam and that they would rather use traditional methods of practicing. Given the time required to develop a good system, time constraints did not allow for earlier distribution.

The average time spent on the site per unique visitor was 37 minutes. Counting only registered users, each user spent on average one hour and 24 minutes on the site. This is time spent having the web-browser with the system as the active tab. If a user walks away from the computer, the time does count as active regardless. Thus, the numbers might be somewhat unrealistic, and a little higher than they should be. The results should be regarded as good anyway. Comparing these results to those that show the number of assignments solved (see Figure 5.3 in section 5.2.1) show that a large number of users spent a significant amount of time using the website.

6.2.3 Questionnaire Distributed After the Final Examination

When asked about problems with the system, it became apparent that assignment design was the biggest issue. Thus, when designing assignments, it is important to adjust the difficulty of assignments properly, in addition to having a concise description of the assignment and any hints related to the assignment. It was clear that not understanding the assignment at hand was the number one reason students stopped using the system. It is also a possible explanation to the large number of students solving so few assignments.

Among the respondents, none claimed that they missed functionality in the system. All the suggestions were related to functionality already in the system, and mainly to the structure of questions and the hints and additional resources related to it. This indicates that the way the system is structured is satisfying, and does not require large changes. Additionally, the question regarding what features were liked in the system indicates that this is an enjoyable way to practice. It seemed that the respondents generally liked that way of learning. Although the system was only distributed during the exam period, students expressed a positive attitude towards having it as an addition to exercises and lectures. That statement matches the goal of the software, showing that the software as it is, fulfills its intended purpose in ITI.

Non-Response Bias

Non-response bias refers to the possibility of having potential answers from those who did not answer, that differ from the answers of those who did. Only 17 students responded to the questionnaire, out of the 38 it was sent to. This yielded a response ratio of 45%. Since the questionnaire was sent after the final examination in ITI, students might have ignored it due to feeling finished with the course. One hypothesis is that only those who found the system especially interesting or useful actually cared to answer. Generally, a low response rate leads to non-response bias, which might have a significant impact on the results. Berg claims that it is still possible to accept the results in many cases, as long as one is being sensitive to the potential impact the low response rate might have had [20]. The questionnaire consisted of textual questions in this case. In questionnaires where answers are continuous (e.g., a scale from one to five), the potential answers from those who did not answer might change the outcome of the questionnaire. Due to the nature of textual questions, more answers simply grant more insight into both issues and good features.

6.3 Measuring Learning Outcome

The results gathered from the ANOVA show that learnpython had a significant learning effect compared to the reading group. Something to note, is that the control group improved quite a lot from pre- to post-test. This might stem from an effect where students learn from simply doing the pre-test. Although the control group was playing an online game during the learning period, some might have processed their experience from doing the pre-test. This could lead to an effect where some subjects were able to make sense of some aspects of the test they had not gotten right the first time. The post-test could also have been slightly easier than the pre-test, leading to subjects naturally scoring more points. While most subjects in the control group did slightly better from pre- to post-test, there were four subjects who scored around 40% better, which seems suspicious, although they do not strictly qualify as outliers as assessed by box plot 5.9 in section 5.3. One theory is that they searched for answers to the pre-test questions out of curiosity, thus naturally

Group		Mean	N	Std. Deviation
Control Group, G_3	post	.4383	18	.23593
	pre	.2683	18	.15983
	change	.17	18	.15010
Control Group, G'_3	post	.3814	14	.23399
	pre	.2764	14	.17491
	change	.1050	14	.09338

Table 6.1: Descriptive statistics; means and standard deviations of standardized test scores after removing outliers.

performing better on the second test.

Table 6.1 show descriptive statistics similar to those found in Table 5.2 in section 5.3, after removing the four subjects from the control group who improved an exceptional amount. Unsurprisingly, the standard deviation went down quite a lot. More interestingly, the control group to reading group relation is now significant at the .019 level, showing that both learning groups actually have improved over the relatively short learning time.

6.3.1 Threats to Validity

The following sections present potential threats to validity and how they were handled. Wohlin et al. identify four different types of threats related to data validity in experiments; conclusion validity, internal validity, construct validity and external validity [25].

Conclusion Validity

The results show a significant spread in individual student's learning capabilities. This is especially visible in the learnpython group, where the change scores covered a large amount of the range. Although the results are valid in terms of significance, that itself is not enough to draw conclusions about the results. It is also advantageous to look at the suggested sample size of the study in terms of effect size. Effect size is a calculation of the strength of a phenomenon, based on the difference in means and standard deviation. Evaluating effect size with *Cohen's d* yields an effect size for the change score between learnpython and the control group of 1.32, which is huge (well above one standard deviation). Thus, the power of study when considering the learning effect of learnpython is

satisfying. When comparing change scores between the reading group and the learnpython group, the effect size is 0.88. When comparing means, Cohen suggests sample sizes based on effect size [26]. For large effect sizes, $d > 0.8$, 26 subjects are suggested to obtain satisfying power in such a study. Given that both groups had 19 subjects, it is slightly too low to justify that learnpython worked better than reading.

Internal Validity

Internal validity is the extent to which the observed outcome of an experiment is due to a given treatment, or other factors. Several precautions were taken to avoid effects from other factors. Subjects were randomly assigned, and a pre-test was used to prevent selection bias [28]. The tests were also assessed with no knowledge of which treatment group they belonged to, in order to negate the observer-expectancy effect. All groups took the tests under identical conditions and encouragement, in a classroom setting.

The pre- and post-test play a vital role in terms of internal validity. It was important to design the tests in such a way that they did not favor any particular group. Most importantly, none of the assignments presented as part of learnpython were equal to a task in either of the tests. Similarly, reading group subjects were presented with textbook material about general, basic programming concepts, rather than exact information about how to solve the tasks in the tests. Both groups were presented with material relevant to the tasks, but none of them were provided with answers.

It was possible to get three points on each task. That makes it easier to grade the quality of an answer compared to assigning one or zero points for each task. The downside is that it makes the grading somewhat subjective. Although it is not considered a large factor, it could have been avoided by having multiple persons assessing the tests, where the final grading is an average of all the assessments. The available resources were limited, making it difficult to involve more people in the process. It would also be necessary to have people who could be trusted to do a thorough job.

Construct Validity

Construct validity is the degree to which the results from some measurement reports what it is supposed to report. Differences in pre- versus post-test scores were the primary measurements in the study. The pre- and post-test being different might have harmed the construct validity to some degree. Having a control group taking both tests does control for that matter when having the possible learning effect from taking the pre-test in mind. Additionally, each group was presented with the same tests, and while the questions were not equal in both, the overall post-test scores of each group are likely not affected by having different questions.

External Validity

External validity is the degree to which a study can be generalized to a general situation or with a general population. There is, of course, a limit to what can be concluded with this study. Given the short learning period and limited curriculum, the experiment does not reflect long-term learning effects of the two different learning methods. This study merely reveals students abilities to learn the very basics of simple concepts, over a short period of time, in a controlled environment. The students recruited were not currently enrolled in ITI, as the course is only lectured during the fall. Most of the students recruited were third-year students, which makes them more experienced than first-year students. Although an effort was made to make sure that none of the subjects were too competent programmers, it is difficult to evaluate to what degree these students represent the population of first-year students. The two years of experience might make a difference with regards to how quickly they are able to understand programming concepts. Then again, the ability to quickly understand concepts is individual, and the two years of university experience might not play a huge role. Given the background of the subjects, however, it is fair to assume that they represent the targeted user group well.

Finally, it is important to address whether or not the learning part of the experiment relates to a real learning situation. There is of course some time pressure involved, as the subjects knew they had 50 minutes to learn. That could stress some of the test subjects. It was clearly stated that individuals ability to learn were of no importance to the experiment

and that they were to do their best and not be results-oriented. Thus, it seems unlikely that this was a prominent factor. Although respectively reading and using learnpython were the main ways of learning for each of the two learning groups, both were allowed to use the Internet as a tool. In this day and age, the Internet is a part of almost any legitimate learning situation for most students. Hence, it is fair to assume that the learning part represents a real, any day learning situation.

6.4 Addressing the Research Questions

This section discusses possible answers to the research questions with regards to the data gathered and the analysis of said data. The context of the research questions is discussed in chapter 1.2. To answer the research questions in the first place, designing and creating a user-friendly and functional system was regarded a prerequisite.

6.4.1 Research Question One

Research question one states: *"Is the system motivating to use as a way of learning how to program?"* This research question was addressed during the design and creation phase, with remote evaluation and questionnaires. Whether something is motivating to use is based on personal opinion, and thus there was no need to measure it in any way mathematically. Data gathered from the questionnaire distributed to those who had completed five assignments showed the following:

- 5 got very little motivation from completing assignments.
- 1 got a little motivation from completing assignments.
- 4 answered neutrally.
- 15 got much motivation from completing assignments.
- 8 got very much motivation from completing assignments.

The answers to this question are positive. The majority seem to get motivated by solving assignments. Additionally, results from the questionnaire distributed after the final exam revealed that several users answered that they enjoyed learning with such a system.

Finally, usage statistics show that several users were willing to use the system for an extended amount of time.

All in all, the conclusion is yes; these sorts of systems are motivating to use as a way of learning how to program. Given the data at hand, there is no indication that says otherwise. Both usage statistics and questionnaire results clearly indicate that students want such a system as part of ITI and are motivated to use it. It is important to note that it might not fit the needs of everyone. As previously stated, students learn in different manners, meaning that it is difficult to suit the needs of everyone. A significant portion of the student base used it, however, and the majority of the users who tried it liked it and were motivated by using it. The usage statistics tells a story in of itself, showing that many of the users who registered solved many assignments. Thus, one can argue that the conclusion is fair, even though the system had weaknesses with the assignments during testing.

6.4.2 Research Question Two

Research question 2 states: *"Is the use of the system an efficient way of learning?"* The results gathered from measuring learning outcome through the learning experiment clearly show that it, in fact, is an effective way to learn. The subjects using learnpython improved by .4079 ($\approx 40\%$) from pre- to post-test, while the control group improved by .17 (17%) on average. The difference in change scores is in itself a strong indication that the subjects using learnpython learned a lot over a short amount of time. Supported by the significance of the result and the power of the study, the conclusion is definitive.

Research question 2.1 states: *"Is the use of the system a more efficient way of learning than traditional reading?"* This also seems to be the case. In the environment of the experiment, students using learnpython did learn more than the reading group (average change scores of .4079 versus .2574). These results showed to be significant, meaning that there is a high probability that the correct conclusion was drawn. It is important to remember that these answers are relevant within the bounds of the experiment. Although the learnpython group learned the most, it does not mean the same answers could be reproduced in a long term experiment. Additionally, the calculations made in section 6.3.1 shows that the number of test subjects were slightly too low to justify the conclusion fully. Thus, the

answer to the research question is not entirely conclusive and is subject to further work.

6.5 Future Work

To make learnpython viable as a supplement to lectures and exercises in ITI, several things should or could be done. These are things that have not been done during this project or was out of the scope, while still being deemed important. The following sections discuss both aspects of learnpython as a product that has a potential for improvement, as well as additional research that should be conducted in order to ensure its learning effect on students.

6.5.1 Further Experiments on Learning Outcome

While motivational effects were studied on students enrolled in ITI during the fall, learning effect was studied during the spring. That made it more difficult to find test subjects with relevant backgrounds for the experiment. As previously discussed, mostly third-year students were recruited for the experiment, and age difference and university experience might make a difference when it comes to how fast they are able to understand certain programming concepts.

Conducting the same, or at least a similar experiment on students early in ITI would be beneficial. It would reveal data about how the system works on new students with all kinds of programming backgrounds. At the same time, it could say something about the difference it has on students who are new, and those who has already had a course in information technology. Although the results gathered should correctly reflect the capabilities such online tools have, it is preferable to conduct the experiment on the targeted user group.

Conducting an experiment with more subjects, possibly enrolled in ITI, over a longer period of time would be preferable. That would make the experiment reflect the learning effect of different methods over a longer time span, with a larger amount of the curriculum involved. A more significant number of subjects would also lead to results with more statistical power. Overall, it would more accurately reflect students ability to learn with

the use of different methods.

6.5.2 Improvements to learnpython

In the current version of learnpython, assignments submitted by users are validated using regular expressions, to verify whether the output equals the correct output. The validation is performed in the frontend code after the response from the server has arrived, which is a weakness. Users can just print the answer using a single statement, yielding a correct result. The issue was known during development but deemed as less important and too time-consuming to change. In future versions it would be wise to move this logic to the backend, enabling the possibility to do more comprehensive validations. For instance, it would be possible to define different test cases to check the code using different input arguments.

Creating assignments is a time-consuming task in such a system. Throughout the project period, a substantial amount of time were used to create and refine assignments for learnpython. Some sort of functionality should be in place to facilitate for simpler, less time-consuming assignment creation. One such method is parameterized assignment creation, meaning the system is able to create similar assignments based on some template, for example, different parameters for an assignment regarding conditionals, like `if/else`. Another possibility is to create functionality that allows others to submit assignments to the system through a form. Although it does not make the creation itself simpler, it allows others to contribute to the project in a meaningful way, which in turn removes some of the work required of the academic staff.

Gamification elements have been somewhat neglected during this project. The achievement system as is, seemed to have too little context for it to be fun to users. Implementing leaderboards as a gamification element could have an impact on the competitive side of people. Leaderboards could function as a way to track both how many assignments other users have solved, how many achievements they have gotten, who has the fastest code execution time, etc. Such functionality may make users want to spend more time solving assignments and optimizing code, which in turn makes them learn more.

Allowing users to choose the difficulty of the assignments they want to solve was a

feature that was wanted by several users. In its current state, the system does not allow that. Along with many of the other possible extensions to the system, it would make assignment creation more difficult and time-consuming. On the other hand it would allow end users to have more control of what sorts of assignment they want to solve.

6.6 Conclusion

During the project period, a web-based self-evaluation system for Python programming called learnpython was created as a research artifact. The system contains programming assignments that users can solve with an editor on the same page. Correctly solving an assignment allows a user to proceed to the next assignment, earning different achievements as more assignments are solved. The system is based on a similar system created by Sindre Haneset Nygård as part of his master's thesis. The system was developed as part of the design and creation strategy to conduct the research, and data was mainly collected in a quantitative manner. After creating the system and user testing functionality along the way, the system was distributed to first-year students as a voluntary option for exam practice. Results from questionnaires showed that the use of such a system as a way of learning was motivating and a fun activity. Analysis of usage data from Google Analytics revealed that a significant amount of users were willing to use the system for an extended period of time, solving a lot of assignments.

To measure the learning effect, 57 students who had little to no programming knowledge were recruited to an experiment. The experiment compared the use of the system to traditional reading in terms of learning efficiency. The experiment was designed as a traditional pre- post-test experiment, having a control group to control for any differences in the two tests. Statistical analysis of the results showed that while both groups showed progress during the learning period, the learnpython group were able to learn more. There is room for skepticism, in terms of the statistical power of the results gathered from the tests, primarily due to the number of test subjects. Additionally, there are issues with generalizing the effects, due to the experiment being limited in time and conducted in a controlled environment.

The study indicates strongly that such web-based tools can be efficiently used as a sup-

plement to traditional teaching methods at a university. Although further research should be done on the subject, the results show that learnpython can be efficiently used for learning. Additionally, the system itself could also be improved. Assignment creation is tedious work, and poorly designed assignments can easily demotivate users. Easier ways to create assignments and better evaluation methods should be implemented to cope with these challenges. Overall, the potential of using such a system in ITI shows promise both in terms of motivation and learning effect for students. That complies with the primary goal of this research; if such a system can be effectively used as a learning tool.

Bibliography

- [1] E. Holden and E. Weeden, “The impact of prior experience in an information technology programming course sequence,” in *Proceedings of the 4th Conference on Information Technology Curriculum*, ser. CITC4 '03, Lafayette, Indiana, USA: ACM, 2003, pp. 41–46, ISBN: 1-58113-770-2.
- [2] L. Thomas, M. Ratcliffe, J. Woodbury, and E. Jarman, “Learning styles and performance in the introductory programming sequence,” in *ACM SIGCSE Bulletin*, ACM, vol. 34, 2002, pp. 33–37.
- [3] M. McCracken, V. Almstrum, D. Diaz, M. Guzdial, D. Hagan, Y. B.-D. Kolikant, C. Laxer, L. Thomas, I. Utting, and T. Wilusz, “A multi-national, multi-institutional study of assessment of programming skills of first-year cs students,” in *Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education*, ser. ITiCSE-WGR '01, ACM, 2001, pp. 125–180.
- [4] A. Haataja, J. Suhonen, and E. Sutinen, “How to learn introductory programming over web,” in *Proceedings of the 7th International Conference of European University Information Systems-(EUNIS2001)*, Humboldt-Universität zu Berlin, 2001.
- [5] S. H. Nygård, “Automatic self-evaluation system for novice python developers,” Master’s thesis, Norwegian University of Science and Technology, Jun. 2016.
- [6] C. Bruce, L. Buckingham, J. Hynd, C. McMahon, M. Roggenkamp, and I. Stoodley, “Ways of experiencing the act of learning to program: A phenomenographic study

-
- of introductory programming students at university,” *Transforming IT education: Promoting a culture of excellence*, pp. 301–325, 2006.
- [7] E. Enström, G. Kreitz, F. Niemelä, P. Söderman, and V. Kann, “Five years with katis—using an automated assessment system in teaching,” in *Frontiers in Education Conference (FIE), 2011*, IEEE, 2011, T3J–1.
- [8] P. Brusilovsky and S. Sosnovsky, “Individualized exercises for self-assessment of programming knowledge: An evaluation of quizpack,” *Journal on Educational Resources in Computing (JERIC)*, vol. 5, no. 3, p. 6, 2005.
- [9] G. Sindre, L. Natvig, and M. Jahre, “Experimental validation of the learning effect for a pedagogical game on computer fundamentals,” *IEEE Transactions on Education*, vol. 52, no. 1, Feb. 2009.
- [10] P. J. Guo, “Online python tutor: Embeddable web-based program visualization for cs education,” in *Proceeding of the 44th ACM Technical Symposium on Computer Science Education*, ser. SIGCSE ’13, Denver, Colorado, USA: ACM, 2013, pp. 579–584, ISBN: 978-1-4503-1868-6.
- [11] B. J. Oates, *Research Information Systems and Computing*. SAGE Publications Ltd, 2006, pp. 108–124, 126–140, 219–232, ISBN: 1412902231.
- [12] V. Vaishnavi and W. Kuechler, “Design science research in information systems,” Jan. 2014, Available at <http://www.desrist.org/design-research-in-information-systems/>.
- [13] L. Cohen, L. Manion, and K. Morrison, *Research in Education*. RoutledgeFalmer, 2000, pp. 213–214, 306–315, ISBN: 0415195411.
- [14] S. A. Jacob and S. P. Furgerson, “Writing interview protocols and conducting interviews: Tips for students new to the field of qualitative research,” *The Qualitative Report*, vol. 17, no. 42, pp. 1–10, 2012.
- [15] R. Martin, M. Shamari, M. E. Seliaman, and P. Mayhew, “Remote asynchronous testing: A cost-effective alternative for website usability evaluation,” *International Journal of Computer and Information Technology*, vol. 3, no. 1, pp. 99–104, 2014.

-
- [16] D. Ary, L. Jacobs, A. Razavieh, and C. Sorensen, *Introduction to Research in Education*, ser. SOE Curriculum Lab. Wadsworth, Cengage Learning, 2009, pp. 309–310, ISBN: 9780495601227.
- [17] J. A. Krosnick and S. Presser, “Question and questionnaire design,” *Handbook of survey research*, vol. 2, no. 3, pp. 263–314, 2010.
- [18] M. F. King and G. C. Bruner, “Social desirability bias: A neglected aspect of validity testing,” *Psychology and Marketing*, vol. 17, no. 2, pp. 79–103, 2000.
- [19] M. T. Orne, “Demand characteristics and the concept of quasi-controls,” *Artifacts in Behavioral Research: Robert Rosenthal and Ralph L. Rosnow’s Classic Books*, vol. 110, pp. 110–137, 2009.
- [20] N. Berg, “Non-response bias,” *ENCYCLOPEDIA OF SOCIAL MEASUREMENT*, vol. 2, pp. 865–873, 2005, Available at <https://ssrn.com/abstract=1691967>.
- [21] D. F. Balph and M. H. Balph, “On the psychology of watching birds: The problem of observer-expectancy bias,” *The Auk*, vol. 100, no. 3, pp. 755–757, 1983.
- [22] A. Padhi and N. Fineberg, *Encyclopedia of Psychopharmacology*, I. P. Stolerman, Ed. Springer Berlin Heidelberg, 2010, pp. 418–418, ISBN: 978-3-540-68706-1.
- [23] Apple Computer, Inc, *Macintosh Human Interface Guidelines*. Addison-Wesley Publishing Company, 1992, pp. 43–46, ISBN: 0201622165.
- [24] D. Howell, *Statistical Methods for Psychology*, ser. PSY 613 Qualitative Research and Analysis in Psychology Series. Wadsworth, Cengage Learning, 2012, pp. 90–97, 101–102, 577–580, 594–604, ISBN: 9781111835484.
- [25] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering*, ser. Computer Science. Springer Berlin Heidelberg, pp. 68–69, 138–139, 143–144, 184–186, ISBN: 9783642290442.
- [26] J. Cohen, “A power primer,” *Psychological bulletin*, vol. 112, no. 1, pp. 155–159, 1992.

-
- [27] Lund Research Ltd. (2015). One-way anova using spss statistics. Available at <https://statistics.laerd.com/premium/spss/owa/one-way-anova-in-spss.php>, (visited on 03/27/2017).
- [28] J. J. Heckman, "Sample selection bias as a specification error (with an application to the estimation of labor supply functions)," 1977.

Appendix

Appendix **A**

Tests Used in the Experiment

In this appendix, the pre- and post-test used to measure the different groups' learning effect are presented. The tests were given before and after receiving some treatment during the experiment. They are presented in the order they were given during the experiment.

PRE-TEST

PARTICIPANT-ID: _____

In the tasks marked with (code), you are supposed to write code that fulfills the task description. In the tasks marked with (quiz), you are supposed to give a textual answer to the question. It is encouraged to attempt to answer a question, even if you do not know the exact answer. In the coding tasks, consider the answer boxes to be an editor, meaning a correct answer should compile without errors. Code is not supposed to be written in Python's interactive mode.

Task 1a (code):

Assign the string `'Test'` to a variable, then print that variable to the screen:

Task 1b (code):

Store the mathematical expression $(20 + 10) + (30 * 10) - 3$ in a variable called `exp`, then print that variable to the screen:

1 / 6

PRE-TEST

Task 1c (quiz):

Write down which type each of the following variables belong to:

```
foo = True
bar = False
baz = 10
qux = 2530, 54
quux = "demp"
```

Task 2a (code):

Assume that a variable `foo` of the type integer is defined in the code. Create a conditional statement using `if/else` that checks if `foo` contains a number between 10 and 100. Print `'Yes'` to the screen if it does and `'No'` if it does not.

2 / 6

PRE-TEST

Task 2b (code):

Assume that two variables `foo` and `bar` of the type boolean are defined in the code. Print `Both` if both variables are true, `print One` if one of them is true and print `None` if both are false.

Task 3a (quiz):

What is printed when the following code is executed:

```
a = True
b = False
if not (a and not b):
    print('Awesome')
else:
    print('Super')
```

3 / 6

PRE-TEST

Task 3b (quiz):

What is printed when the following code is executed:

```
a = 5
b = 10
if ((not b / 2 == a) and (a > b)):
    print('First')
elif (a == b) or (a != b):
    print('Second')
else:
    print('Third')
```

Task 4a (code):

Create a for-loop and use it to print every odd (1, 3, 5, 7 and so forth) number between 0 and 100:

4 / 6

PRE-TEST

Task 4b (code):

Assume that a variable `foo` of the type integer is defined in the code and currently holds the value 0. Create a while-loop and increment `foo` by 1 and print the value for each iteration. Stop looping once `foo` reaches the value 42.

5/6

PRE-TEST

Built in functions and useful structures

```
print(statement)
```

Displays (prints) the `statement` on the screen. `statement` can be text, number, boolean values and more.

```
range(start, stop[, step])
```

Can be used in for-loops in order to iterate over a range of integers. `start` is an integer used to define the start of the range, `stop` is used to define the end of the range. `step` is optional, and defines how far each step of the loop is (default value is 1).

Assigning a variable:

```
Variable_name = value
```

False/true syntax:

```
If expression:
    statement(S)
elif expression2:
    statement(S)
else:
    statement(S)
```

for-loop syntax:

```
for iterating_var in sequence:
    statement(S)
```

while-loop syntax:

```
while expression:
    statement(S)
```

6/6

POST-TEST

PARTICIPANT-ID: _____

In the tasks marked with (code), you are supposed to write code that fulfills the task description. In the tasks marked with (quiz), you are supposed to give a textual answer to the question. It is encouraged to attempt to answer a question, even if you do not know the exact answer. In the coding tasks, consider the answer boxes to be an editor, meaning a correct answer should compile without errors. Code is not supposed to be written in Pythons interactive mode.

Task 1a (code):

Assign the string `'Another test'` to a variable, then print that variable to the screen.

Task 1b (code):

Store a floating point number to a variable called `floating_point`, then print that variable:

1 / 6

POST-TEST

Task 1c (quiz):

Write down which type each of the following variables belong to:

```
foo = "TEST"  
bar = False  
baz = "10"  
qux = 34.2345
```

Task 2a (code):

Assume that a variable `foo` of the type string is defined in the code. Create a conditional statement using `if/elif/else` that checks if the variable contains the letter `c`. Print `Yes!` if it does and `No!` if it does not.

2 / 6

POST-TEST

Task 2b (code):

Assume that two variables `foo` and `bar` of the type integer are defined in the code. Print `First` if `foo` is larger than `bar` and `second` if `bar` is larger than `foo`. Print `Equal` if the two variables are equal.

Task 3a (quiz):

What is printed when the following code is executed:

```
a = False
b = True
if not (a and not b):
    print('Awesome')
else:
    print('Super')
```

3 / 6

POST-TEST

Task 3b (quiz):

What is printed when the following code is executed:

```
a = False
b = False
if ((a or b) and (not (a and b))):
    print('Harry')
else:
    print('Hermione')
```

Task 4a (code):

Create a for-loop and use it to print every even (2, 4, 6, 8 and so forth) number between 0 and 100:

4 / 6

POST-TEST

Task 4b (code):

Assume that two variables `foo` and `bar` are defined in the code, where both are integers and `foo` is larger than `bar`. Create a while-loop and increment `bar` until both variables holds the same value. Print the values of both `foo` and `bar` in each iteration of the loop.

5/6

POST-TEST

Built in functions and useful structures

```
print(statement)
```

Displays (prints) the `statement` on the screen. `statement` can be text, number, boolean values and more.

```
range(start, stop[, step])
```

Can be used in for-loops in order to iterate over a range of integers. `start` is an integer used to define the start of the range, `stop` is used to define the end of the range. `step` is optional, and defines how far each step of the loop is (default value is 1).

Assigning a variable:

```
Variable_name = value
```

False/False syntax:

```
If expression:  
    statement(S)  
elif expression2:  
    statement(S)  
else:  
    statement(S)
```

for-loop syntax:

```
for iterating_var in sequence:  
    statement(S)
```

while-loop syntax:

```
while expression:  
    statement(S)
```

6/6

Appendix **B**

Raw Data from the Experiment

The following table contains the raw data generated in the experiment. Group 1 represents the learnpython group, group 2 is the reading group and group 3 is the control group.

Group	Pre-score	Post-score	Change-score
1	0.41	0.81	0.41
1	0.26	0.48	0.22
1	0.26	0.93	0.67
1	0.33	0.56	0.22
1	0.33	0.89	0.56
1	0.30	0.85	0.56
1	0.44	0.67	0.22
1	0.26	0.74	0.48
1	0.44	0.78	0.33
1	0.41	0.81	0.41
1	0.56	0.78	0.22
1	0.44	0.59	0.15
1	0.15	0.96	0.81
1	0.52	0.56	0.04
1	0.30	0.89	0.59
1	0.00	0.52	0.52
1	0.22	0.89	0.67
1	0.41	0.67	0.26
1	0.00	0.41	0.41
2	0.22	0.41	0.19
2	0.48	0.63	0.15
2	0.33	0.70	0.37
2	0.30	0.63	0.33
2	0.41	0.81	0.41
2	0.30	0.63	0.33
2	0.56	0.70	0.15
2	0.26	0.44	0.19
2	0.19	0.63	0.44
2	0.30	0.74	0.44
2	0.41	0.48	0.07
2	0.33	0.63	0.30
2	0.37	0.67	0.30
2	0.15	0.44	0.30
2	0.15	0.52	0.37
2	0.04	0.30	0.26
2	0.30	0.52	0.22
2	0.00	0.11	0.11
2	0.11	0.11	0.00

Group	Pre-score	Post-score	Change-score
3	0.26	0.44	0.19
3	0.48	0.56	0.07
3	0.30	0.33	0.04
3	0.63	0.93	0.30
3	0.44	0.48	0.04
3	0.19	0.19	0.00
3	0.30	0.30	0.00
3	0.41	0.56	0.15
3	0.19	0.37	0.19
3	0.33	0.70	0.37
3	0.15	0.52	0.37
3	0.15	0.59	0.44
3	0.04	0.07	0.04
3	0.00	0.00	0.00
3	0.33	0.74	0.41
3	0.33	0.52	0.19
3	0.19	0.37	0.19
3	0.11	0.22	0.11

Table B.1: Raw Data Gathered During the Experiment