Tampere University

Aleksi Postari

ANALYZING HOW WELL CHATGPT CAN COMPLETE A BEGINNER-LEVEL PYTHON COURSE

Faculty of Information Technology and Communication Sciences
M. Sc. Thesis
October 2023

ABSTRACT

Aleksi Postari: Analyzing How Well ChatGPT Can Complete a Beginner-Level Python Course
M.Sc. Thesis
Tampere University
Master's Degree Programme in Computer Science
October 2023

In recent years, Artificial intelligence (AI) has seen substantial improvements. ChatGPT, an advanced conversational chatbot trained on massive amounts of data, has sparked a lot of public attention due to its ability to solve complicated issues in various domains.

The primary research question of this thesis investigated the performance of various ChatGPT models in completing beginner-level Python course coding-related assignments and exams. The secondary research question examined the comparative performance of ChatGPT models and human students in tackling Python course assignments and exams. The tertiary research question explored the types of errors or incorrect responses produced by ChatGPT models when attempting Python course assignments and exams.

In this thesis, an experiment was carried out with ChatGPT using a "Programming Basics" course from Satakunta University of Applied Sciences. The results were spectacular; the latest model could complete the course with the highest grade of 5, with a completion score of 97%. Three different ChatGPT models were available during the experiment.

The findings revealed that each iteration of the GPT model did show improvement in completing beginner-level Python course coding-related assignments and exams. While comparing the models to humans, it was found that the models performed at the same level or even higher than humans. While categorizing the incorrect answers, it became apparent that the models struggled with fundamental programming problems, like slicing strings and handling inputs.

Keywords: ChatGPT, Artificial Intelligence.

The originality of this thesis has been checked using the Turnitin Originality Check service.

Preface

My personal motivation for investigating ChatGPT and its ability to solve programming questions for a programming course stems from my background. My professional journey began over a decade ago and has since included two start-ups in the game-based learning / phenomenon-based learning domain, and in the teaching field in general. I have been utilizing ChatGPT since its launch in late 2022 and have closely followed its development, as well as other advancements in the related field, on a daily basis.

How was Artificial Intelligence used

To aid in creating this thesis, two artificial intelligence-based tools were used: ChatGPT and Grammarly. ChatGPT was used to create ideas on the structure of the thesis and ideas on how to transform some parts of the text to a more scientific text format. Grammarly was used to correct the grammatical errors in this thesis. Nevertheless, artificial intelligence tools were only used as an assistive tool for creating this thesis, and all ideas, conclusions, and results are my own.

No artificial intelligence tools were used to write this section and abstract.

Contents

1 Motivation

The landscape of information access has evolved dramatically over the last decades. During the 19th century, formal educational establishments and libraries were instrumental in disseminating knowledge to the masses [Kranich, 2014]. Schatz [1997] already discussed 1997 that scientific literature and other material will be easily accessible through the Internet and its search engines. This digital revolution of all the media moving to the Internet has transformed the way people seek and consume knowledge.

In late 2022, a groundbreaking development in information retrieval emerged with the release of ChatGPT, a conversational artificial intelligence language model [Rudolph, Tan, Tan, 2023. OpenAI, 2022]. This innovative technology, which has swiftly gained popularity, is considered one of the most rapidly expanding web services [Hu, 2023]. ChatGPT has undoubtedly captured the attention of many users, with numerous individuals recognizing its potential to revolutionize the way information is sought, processed, and communicated. The CEO of Google even expressed significant concern by announcing a "Code Red" situation in response to the potential impact of ChatGPT on their business operations [Phelan, 2023].

The impact of ChatGPT is particularly noticeable due to its sophisticated conversational abilities, achieved through training on massive amounts of data [Rudolp, Tan, Tan, 2023]. ChatGPT is still a very novel technology, but it has garnered lots of attention and can potentially be the tool in the future that we use to retrieve and interact with information.

1.1 ChatGPT Disrupting the Formal Education

Rudolph and Tan and Tan [2023] present that there is a hype of ChatGPT transforming how we assess students in higher education. ChatGPT can generate text from which it is impossible to determine if it was written by a human or AI [Rudolp, Tan, Tan, 2023]. Some instructors are also worried that students will use these tools to complete their coursework, as the tool can create satisfactory text in seconds and bypass plagiarism detectors [Rudolp, Tan, Tan, 2023].

Malinka, *et al.* [2023] mention in their study that ChatGPT has surpassed its anticipated abilities, prompting considerations of its potential use and potential for misuse. The study made by Malinka, *et al.* [2023] presented the ability for ChatGPT to be used for misuse and concluded that ChatGPT has the ability to pass courses required for a university degree.

Jalil, *et al.* [2023] bring upon in their research that ChatGPT has spurred fervent discussion from educators in terms of fear of students using these AI tools to circumvent learning. "Findings still raise immediate concerns on how the use of ChatGPT might be detected to ensure that questions are meaningfully assessing students' understanding of course materials" Jalil, *et al.* [2023].

Building on the findings presented, it is clear that ChatGPT possesses the potential to influence and reshape the realm of formal education profoundly. Consequently, the research conducted in this thesis holds substantial importance and offers valuable insights for future studies within this domain.

1.2 Research Questions

The first research question of this thesis seeks to investigate the performance of various ChatGPT models in completing beginner-level coding-related Python course assignments and exams.

The secondary research question examines the comparative performance of ChatGPT models and human students in tackling Python course assignments and exams.

The tertiary research question delves into the types of errors or incorrect responses produced by ChatGPT models when attempting Python course assignments and exams.

1.3 Research Methods

This thesis employs a mixed-methods approach, combining experimental design and statistical analysis to address the research questions. More of the methods are introduced in Chapter 3. The experimental design is implemented within the context of a beginner-level university Python course, focusing exclusively on code-related questions throughout the experiment.

The structure of this thesis and the methods to conduct the experiment are inspired by the experimental design template proposed by Jedlitschka and Pfahl in the book Experimentation in Software Engineering [Wohlin, 2012].

1.4 Structure of Thesis

Chapter 2 of this thesis aims to familiarize the reader with the essential background information related to the experiment conducted. An overview of artificial intelligence,

AI-Assisted Software Development, the Python programming language, the ChatGPT technology, and a couple of prior studies will be introduced.

Chapter 3 provides the methodology employed in the study. This includes the material used, the procedure of the study, the data analysis, implications for test reproducibility, and hypotheses formed from previous studies.

Chapter 4 delivers the results from the experimental study, detailing the performance of each ChatGPT model in completing the programming exercises and exams, as well as the types of errors encountered during the process.

Chapter 5 offers a discussion of the results, comparing the performance of the different ChatGPT models, analyzing their strengths and weaknesses, and exploring the implications of these findings for both the field of AI and education. Additionally, this chapter explores the limitations of the study and future research.

Chapter 6 concludes the thesis by summarizing the main findings, discussing the limitations of the study, and suggesting potential future research in the field.

2 Background

This section presents background information on the subject. First, the basic concepts of artificial intelligence, machine learning, natural language processing, and large language models are presented.

Next, an introduction to AI-Assisted Software Development is given. The experiment conducted in this thesis focuses on exploring how well artificial intelligence can solve programming challenges. Thus, the section related to AI-Assisted Software Development provides a brief understanding of the background concepts of this field.

The chapter after AI-Assisted Software Development is focused on the Python programming language, as it is the language used in the course for which the experiment is conducted. As the last chapter providing theoretical understanding, ChatGPT is discussed, given its central role in the execution of the experiments. Subsequently, the various ChatGPT models employed in the study are presented and described.

Finally, a couple of relevant prior research studies are presented.

2.1 Artificial Intelligence

Since ancient times, great philosophers have grappled with the problem of thought process automation. This has not been possible until the twentieth century when the first computers were built. Alan M. Turing has been seen as the pioneer in the field of Artificial Intelligence. Turing came up with a so-called *imitation game*, an operational test of Artificial Intelligence. The imitation game works in a way that a human interrogator, another human, and another computer have a conversation together. According to Turing, if it is not possible to distinguish between a human and a computer during a conversation, then the computer's intelligence is the same as the intelligence of a human being. [Flasiński, 2016]

The birth year of artificial intelligence is usually seen as 1956. During the year, a conference was held in Dartmouth, where significant persons related to artificial intelligence were brought together. During the conference, the term "Artificial Intelligence" was first widely used and from which the concept and name of artificial intelligence were established. [Skinner, Elizabeth, 2012]

Even though the birth year can be considered as 1956, there has been research related to artificial intelligence dating even back to 1943. During the year 1943, Warren Mcculloch and Walter Pitts proposed a model where artificial neurons can be characterized to

reside in "on" or "off" state, and the state from "off" can be switched on by the stimulation of neighboring neurons. Pitts and Mcculloch demonstrated that any computable function is being able to be computed using a network of connected neurons. It was also stated by Pitts and Mcculloch that the created neuron network could learn by modifying the strength of connections between the neurons. This model is called *Hebbian learning*. [Russell, Norvig, 2010]

Artificial Intelligence has evolved significantly since its release in the 1950s [Russell, Norvig, 2010]. To date, Artificial Intelligence has branched to multiple sub-disciplines, including machine learning (ML), natural language processing (NLP), and more [Russell, Norvig, 2010]. The next sections will cover the basics of machine learning, natural language processing, and large language models.

2.1.1 Machine Learning

An algorithm of machine learning is a computational process that takes input data to achieve a desired outcome without having the values hard coded. The algorithms in machine learning adapt or alter the architecture through repetition to improve the desired outcome. The process where different kinds of input data alongside the desired outcome is provided is called *training*. The training part is the part where the learning happens. The algorithm should finally optimize itself so that it can produce desired outputs even from unseen data. [El Naqa, et al, 2015]

One example of machine learning in use is the detection of hand-written digits. For example, a 28x28 pixel image (see Figure 1) is being able to be represented as a vector that totals 784 numbers. This can be used to feed the algorithm vector x as input, and it will finally output digits 0 to 9. [Bishop, 2006]

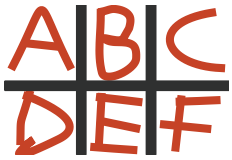

Figure 1: 6 hand-written alphabets in 28-pixel space each.

To achieve the results that the machine learning algorithm can detect the desired digit from the given hand-written image, a large set of digits will be used as a so-called *training set* to configure the parameters of the model. Typically the categories of the digits in the training set are known in advance, which involves a person inspecting and hand-labeling the digits. Once the model has been trained, it can be passed as input any new digit which did not belong to the original training set. This ability for the model to adapt

to new input data which did not belong to the original training set is called *generalization*. [Bishop, 2006]

Machine learning applications in which the input and corresponding output vectors are known are called *supervised learning*. For example, the case of handwritten digit recognition, where a data set of known hand-written digits are passed as inputs and outputs are already known. The digit recognition case, where there are a finite number of discrete categories, can be called a *classification problem*. [Bishop, 2006]

In other machine learning applications, where the input vector is known but there is no output vector provided, it is called *unsupervised learning*. The goal of these machine learning tasks is to usually automatically find similarities in the data and group similar data together, which is also called clustering. [Bishop, 2006]

The learning process in machine learning can be adapted by using a system called *reinforcement learning*. In reinforcement learning, a feedback system is created to increase the learning of the model by providing a reward for specific desired actions, for example, winning a game of checkers. [El Naqa, et al, 2015]

2.1.1.1 Neural Networks

Artificial Neural Networks (ANNs), or simply just neural networks, are computational models which are greatly inspired by the human brain's neuron structure. Neural networks mimic how the human brain processes and analyzes data. Neural networks are used in modern applications like self-driving vehicles, stock market price prediction, character recognition, and more. [Dastres, Soori, 2021]

Neural networks are represented similarly to the structure of a human brain, where neuron nodes are connected in a web-like fashion [Dastres, Soori, 2021]. Figure 2 demonstrates a simple neural network with an input layer, output layer, and hidden layer. Data enters the input layer, is processed in the hidden layers (this is usually one or multiple layers), and is finally outputted through the output layer [Dastres, Soori, 2021]. This behavior is similar to how the information enters the human brain and outputs [Dastres, Soori, 2021]. Each neuron in the network contains a weight that affects the signal intensity, and neurons that contain a threshold value only transfer information if the aggregate signal exceeds it [Dastres, Soori, 2021].

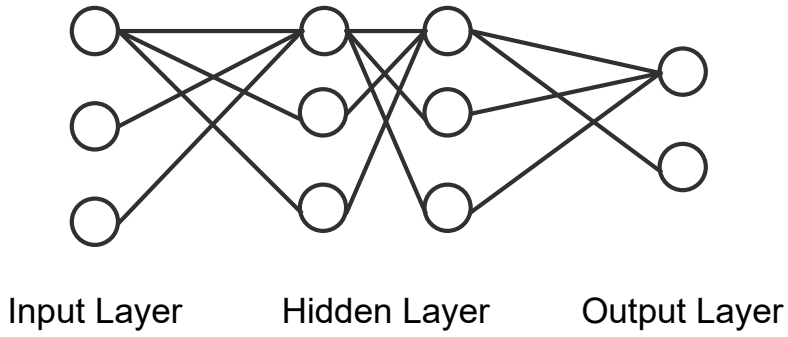

Figure 2: Simple demonstration of a neural network.

Neural network topologies can be generally categorized into two: *FeedForward* and *Feedback*. In the FeedForward neural network topology, the information travels only in one direction; from the input layer to the output layer. In the Feedback neural network topology, the information can flow back from the output layer to the input to maintain a memory of previous inputs. [Dastres, Soori, 2021]

2.1.1.2 Deep Learning

Deep learning is a subfield of machine learning which focuses on artificial neural network algorithms. Deep learning occurs when a neural network has more than three layers of neurons (including input and output layers). Deep learning allows for the representation of more complex layers than just with basic neural networks but requires more computing time and power. [Dastres, Soori, 2021]

While using deep learning, the model automatically adjusts the weights of the neurons in the neural network. Deep learning has brought breakthroughs in image, video, speech, and audio processing. Some modern applications of deep learning are, for example, to identify objects in images, match posts of user's interest, or find relevant search results. [LeCun, Bengio, Hinton, 2015]

2.1.2 Natural Language Processing

Natural language processing (NLP) is a field of research where computers are being enabled to perform tasks involving human language [Jurafsky, Martin, 2007]. These tasks can be, for example, human-machine communication, human-human communication, or the processing of text or speech [Jurafsky, Martin, 2007]. Generally, it could be stated that natural language processing is used for computers to understand and interpret using human language.

Knowledge of language is what distinguishes language processing applications from other data processing applications. For example, a Unix system has a program called *wc* that can be used to count the total number of words, bytes, and lines in a file. When it is used to count the number of bytes and lines, it is just a regular data processing application. However, when the program is used to count the number of words in a file, there is a requirement to be some kind of knowledge of what forms a word; thus, it becomes a language processing system. [Jurafsky, Martin, 2007]

2.1.2.1 History

Natural language processing as a research field has its roots in the 1950s as an intersection of artificial intelligence and linguistics [Nadkarni, Ohno-Machado, Chapman, 2011]. Kleene came up with finite automata in 1951 and regular expressions in 1956 [Jurafsky, Martin, 2007]. The sound spectrogram was developed in 1946, which also gave the possibility for the first machine speech recognizers to be developed in the early 1950s [Jurafsky, Martin, 2007]. In 1952 in the Bell Labs, a statistical system was built that could detect any ten digits from a single speaker [Jurafsky, Martin, 2007]. The system achieved a 97-99% accuracy [Jurafsky, Martin, 2007]. Historically natural language processing has been conducted by hand using rule-based systems [Manning, Schutze, 1999]. This approach only applies to very small domains and to languages with very limited semantics due to their manual nature [Manning, Schutze, 1999].

During the 1950s and 1960s, speech and language processing diverged into two paradigms: symbolic and stochastic. Symbolic systems worked mainly on single domains where a combination of keyword search and pattern matching was used for question-answering and reasoning. The stochastic systems worked on calculating the likelihood of observed letter sequences for the given words. [Jurafsky, Martin, 2007]

The research on natural language processing saw a surge in the 1970 to mid-1980s with paradigms that still influence the field to date. The stochastic paradigm was hugely popular during this period and saw advancements with the research done by Bahl, Mercer, and others with the development of the Hidden Markov Model and the metaphors of noisy channels and decoding. The field of *natural language understanding* also took off during this period with the work of Terry Winograd on the SHRDLU system. The program accepted natural language text commands as input (for example, a word provided by the user to move a red block on top of the smaller green one). [Jurafsky, Martin, 2007]

During the mid-1980s and mid-1995s, widespread adoption of probabilistic models for speech and language processing was seen. IBM Thomas J. Watson Research Center's work on probabilistic models of speech recognition was seen as the influencer of this adoption. [Jurafsky, Martin, 2007]

After the mid-1950s, the memory and speed of computers began to increase, which allowed commercial speech and language processing applications to be applied. Some examples of this are speech recognition and grammar checking. The rise of the Web also had a role in the importance of language-based information extraction. [Jurafsky, Martin, 2007]

In the early 2000s, the power of computers began to increase, allowing for larger natural language tasks to be completed. A large amount of spoken and written material was released by Linguistic Data Consortium (LDC) and other organizations. There also became a trend towards unsupervised machine learning techniques, which demonstrated that effective applications can be created even with data that has not been annotated.

2.1.2.2 Key Concepts

Generally, the process of natural language processing can be separated into the following process: surface text (input text) is given, tokenization, lexical analysis, syntactic analysis, semantic analysis, pragmatic analysis, and the speaker's intended meaning is outputted [Chapman, Hall, 2010]. Each of these steps has its own distinctive features, described in the chapters below.

Tokenization is the method of separating (tokenizing) words from text. Whitespace is usually used, at least in the context of the English language, but it is not always sufficient and accurate enough. For example, "New York" is two words, but tokenization would separate it into two different words, "New" and "York". [Jurafsky, Martin, 2007]

Lexical analysis is the process of performing text analysis at the level of the word. For example, the verb delivers is a cover term for multiple different words: delivers, deliver, delivering, delivered. The most basic operation of lexical analysis is to connect different words to their base form, also known as the *lemma*. The name of the process where the word is reduced to its basic form is called *lemmatization*. For example, given the example mentioned above, "delivered" and "delivering" can be reduced to "deliver". The lemmatized words are usually stored in a dictionary so that the meaning of the word can be looked up. [Chapman, Hall, 2010]

Syntactic Analysis is the process of analyzing how different words and phrases have a relation to each other and how they convey a meaning. During the process, a hierarchical, syntactic structure could be formed for further inspection. See Figure 3 for an example of a syntactic structure, where the word "the old man a ship" is represented in a syntax tree. Each node in the tree represents a grammar rule, and the leaves represent a word. [Chapman, Hall, 2010]

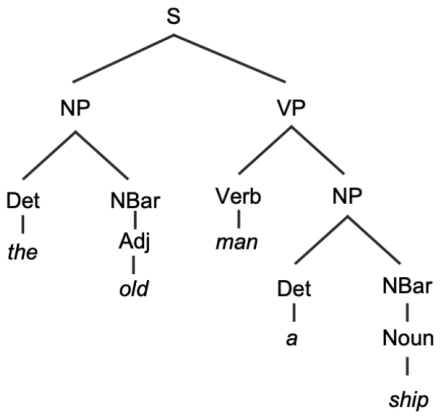

Figure 3: The word "the old man a ship" is represented in a syntax tree.

During the process of *Semantic Analysis,* the actual meaning of the sentences will be understood in their context. The meaning of individual words and phrases will be interpreted to understand the overall meaning of the whole text or parts of it. This process could involve, for example, text summarization, machine translation, and information retrieval. Pragmatic Analysis goes even further than semantic analysis by having the goal of understanding the goals and intentions of the speaker. [Chapman, Hall, 2010]

2.1.3 Large Language Models

Human communities have their own shared language, and they interact with other communities in the world. When a human infant is born, they will share and adapt to this shared language while growing up. Casual conversations, speech, or even email messages build upon these concepts of shared language learned from the community. [Shanahan, 2020]

Large language models work in a different way than humans [Shanahan, 2020]. A language model is a model that works on statistics to determine the following word(s) in sequence [Cooper, 2021]. Large language models do not have an understanding similar to humans, and their response is based on the highest probability based on the training set that they have been trained upon [Shanahan, 2020]. A large language model is the same as a language model but has just been trained with a large amount of data [Cooper, 2021]. Language models are being created by inputting a large amount of written data, and for example, GPT-3 model, introduced later in the thesis, was trained with 45 terabytes of text [Rudolp, Tan, Tan, 2023]. Large language models represent a significant evolution in the field of natural language processing and show impressive performance on a wide variety of tasks [Radford, *et al*, 2019].

In 2018 researchers at Google released a model called BERT (Bidirectional Encoder Representations from Transformers) [Devlin, *et al*, 2019]. The main improvement in BERT compared to previous models is that it can take into account the left and right sides of the given word and provide a better understanding of the context where the word appears [Devlin, *et al*, 2019]. Due to the improvements provided by BERT, it can be used to easily adapt to a task (fine-tuned to a specific purpose with the given dataset), like, for example, question answering and language inference [Devlin, *et al*, 2019]. BERT has been laying innovative groundwork for today's large language models.

Following BERT, company OpenAI initiated releasing their GPT (Generative Pre-trained Transformers) language models. GPT models utilize a massive dataset for training and can perform well on tasks not directly in the training data. [Brown, *et al*, 2020]

2.1.3.1 Key Concepts

The smallest unit with which large language models work are called *tokens*. The tokens can be words, subwords, or even characters. One example of a word that has been split into tokens, an operation so-called tokenization, is splitting the word "supersymmetrization" into tokens "super", "symmetry", and "ization". [Douglas, 2023]

GPT and BERT are large language models which utilize a transformer-based model [Devlin, et al, 2019; Brown, et al, 2020]. The transformer model was released in 2017 by Wasvani et al., and its development from previous models is the ability to handle long text in a sequence [Vaswani, et al, 2017]. The main innovation of the transformer model is a mechanism called self-attention [Wasvani, et al, 2017]. Self-attention makes sure that the importance of other words in the sequence is calculated, creating context-aware representations of the sentences as an output [Wasvani, et al, 2017].

Although both BERT and GPT are transformer-based models, they vary in a way that BERT is a pre-trained masked language model, and GPT is an autoregressive language model. In the autoregressive language model, the next token (the basic unit of text, like the word) predicted will depend on all the previous tokens. The masked language model (MLM) works in a different way so that some tokens in the training set are replaced with a special token [MASK], and the objective of the model is to detect with what token to replace the masked token. Due to this difference, it is easy to realize that both of these models work for their own purposes. [Liao, Jiang, Liu. 2020]

Large language models are not usually used as-is to output words with only the textual corpus it has been trained [Shanahan, 2020]. A concept of fine-tuning is usually applied to large language models to train them for certain objectives, like, for example, to be able to have human-like conversations with the model [Shanahan, 2020]. It is also possible to further optimize the language model by using a concept called reinforcement

learning from human feedback (RLHF), where a human gives a positive reward for the model based on their responses to input [Shanahan, 2020; Ziegler, et al, 2020].

2.2 AI-Assisted Software Development

For the last 30 years, the software development field has seen an increased amount of software development projects written again from scratch, and the efficiency of the projects has been waiting for an increment. Writing code has mainly been a task of a skilled human, which makes creating software an expensive and error-prone process. Recent developments in the field of AI have, however, shown significant improvements in overcoming these issues. [Korzeniowski, Goczyla, 2019]

Korzeniowski and Goczyla (2019) classify AI-software development (software development that is assisted by artificial intelligence) into three main categories: Full AI, Assisted AI, and Supplementary AI. All of these three categories have their own characteristics, represented below. [Korzeniowski, Goczyla, 2019]

2.2.1 Classifications

Full AI is classified to be a research direction where a program is automatically and fully generated based on specifications. The user interference is only required to specify the requirements for the software, and artificial intelligence, fully autonomously, creates the software. This research direction assumes that humans can be replaced in the process of creating software. [Korzeniowski, Goczyla, 2019]

Assisted AI means that there is a cooperation between the business user and AI, and program creation is semi-automated. User interference is required in every aspect of the software development process, and artificial intelligence is only used to assist in the development process. A human supervises AI and provides corrections to what AI generates. This research direction assumes that human is irreplaceable in the process of creating software, but this concept of Assisted AI can also vanish when Full AI matures. [Korzeniowski, Goczyla, 2019]

Supplementary AI determines that AI is used to improve the quality or performance of the manual software development process. Only the current manually conducted software development processes are being automated using AI. Some of these improvements could be to detect code defects, code improvement suggestions, automatically creating common libraries shared in different projects, and automatic documentation creation. [Korzeniowski, Goczyla, 2019]

It is still to be seen how extensively artificial intelligence can be utilized in software development. When it comes to the practical implementation of these categories, Supplementary AI can be seen as the most short-term solution, whereas Assisted AI is a mid-term solution, and Full AI is the long-term solution. [Korzeniowski, Goczyla, 2019]

2.2.2 Application

When it comes to software development, artificial intelligence can be applied to multiple tasks in different domains. Generally, the tasks into which artificial intelligence can be applied can broadly be categorized into code generation, code completion, code translation, code refinement, and code summarization. [Wong, et al, 2023]

Code generation, also known as program synthesis, is to automatically generate source code for programming language based on user inputs and constraints. Transformer-based large-language models like GPT have, in recent years, made impressive improvements in this field by utilizing a contextual understanding of large amounts of code. [Wong, et al, 2023]

Code completion, or autocompletion, is a feature found in software development IDEs to suggest user code completions as the user types. Transformer architecture has also shown significant improvements to this feature. [Wong, et al, 2023]

One difficult aspect of software development is to convert a codebase from one programming language to another. *Code translation* is the process of doing exactly this task, where a goal usually is to migrate older legacy software to a later programming language or framework. Artificial intelligence can provide a more automated approach to this tedious task. [Wong, et al, 2023]

Programmers are also human and tend to write code that can have problems or bugs. *Code refinement* is the process of automatically converting these problematic parts of the code to the correct format. Large language models, where the user inputs the faulty code, and the language model outputs a suggestion of a correct code, tend to show great results in this approach. [Wong, et al, 2023]

Finally, *code summarization* is the process of automatically documenting the source code, first at the function level, and finally, generating a large documentation of the codebase. In recent years, this same methodology has been applied to generate a natural language description of the whole program. [Wong, et al, 2023]

2.2.3 Practical Use – Github CoPilot

One of the modern tools which can be used to generate code automatically is called GitHub Copilot [Sobania, Briesch, Rothlauf. 2021]. It is an extension to the Visual Studio Code programming environment [Sobania, Briesch, Rothlauf. 2021]. The extension was released in June 2021 and uses the Codex model created by OpenAI, which has been trained with billions of lines of open-source GitHub code [Zhang, et al, 2023].

The current version of GitHub Copilot can accomplish multiple tasks. When opening a file and the extension is enabled, GitHub Copilot will automatically start finishing the text for the user (see Figure 4).

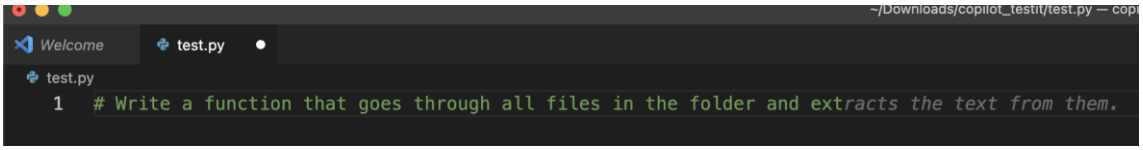

Figure 4: GitHub Copilot automatically suggests to the user the completion of the sentence.

GitHub Copilot also works for more extensive development. When, for example, starting to write a function, it can automatically suggest the function to be used (see Figure 5).

```
def extract_zip_files_in_folder(folder):
    """
    Extracts all zip files in a folder
    :param folder: folder to extract zip files from
    :return: None
    """
    for file in os.listdir(folder):
        if file.endswith(".zip"):
            with zipfile.ZipFile(folder + "/" + file, 'r') as zip_ref:
                zip_ref.extractall(folder)
```

Figure 5: GitHub Copilot automatically suggests the code for the function.

There also exist multiple additional tools to be used with GitHub Copilot, for example, the Copilot Labs. These consist of, for example, explaining code, fixing a bug, improving the readability of the code, or cleaning a code (see Figure 6).

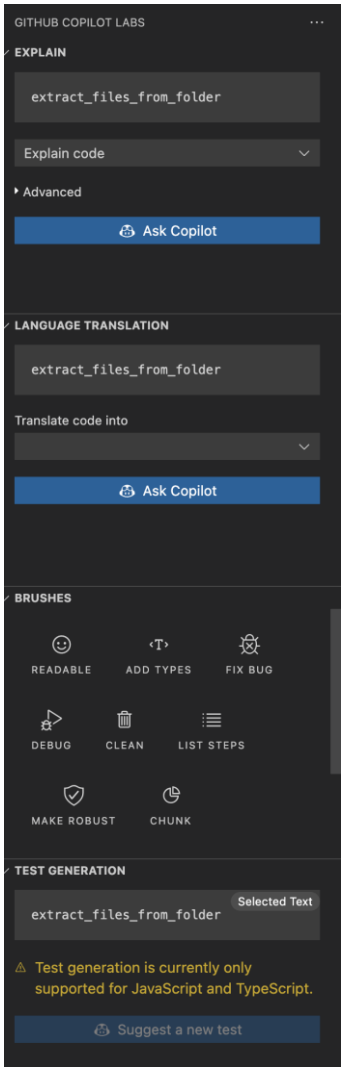

Figure 6: GitHub Copilot Labs feature enables you to, for example, explain a given code, make it more readable, fix a bug in the code, or clean the code.

2.3 Python Programming Language

In 1991 at Stichting Mathematisch Centrum, Guido van Rossum created Python, an interpreted high-level programming language. As open-source software, Python boasts a straightforward syntax (see Figure 7) that facilitates quick learning and makes it an appealing choice for rapid application development. [van Rossum, Drake, 2011]

```
fruits = ["apple", "banana"]
for fruit in fruits:
    print(fruit)
```

Figure 7: List and loop created with the Python programming language.

2.3.1 Popularity

Python enjoys widespread popularity among programming languages. In a 2022 Stack Overflow survey involving more than 70,000 developers, Python ranked as the second most sought-after and sixth most beloved language [Stack Overflow, 2022]. Lutz [2009], who has taught Python to over 3,000 students in a span of 12 years, asserts that software quality and developer productivity likely constitute the most persuasive advantages for Python users.

Python has not gained popularity only in recent years, but already in 2009, Lutz estimated that there are roughly 1 million Python users around the globe. Multiple companies around the globe use Python, and some of the prominent examples of Python in use in real-world applications are Youtube, Google Search, NASA for scientific programming tasks, the game EVE Online, and more. [Lutz, 2009]

2.3.2 Purpose

Python is a language that can be used for many general-purpose tasks. Commonly, it is defined as an object-oriented language, although the language itself embraces an array of paradigms, including procedural, object-oriented, and functional strategies [Lutz, 2009]. Perez, Granger, and Hunter [2011] state that Python is valid for high-level scientific code development.

Python has plenty of native types and is a dynamically typed language. Native types include some common types like strings, floating-point and common numbers, arbitrary-length integers, sets, and associative arrays known as dictionaries in Python. Python even includes low-level capabilities by having the ability to extend the language with C code and to be able to use third-party C and C++ libraries with the language. [Perez, Granger, Hunter, 2011]

2.3.3 Application

Python can be used to accomplish multiple real-world programming tasks in multiple domains. As it can be considered to be a general-purpose language, it can be used for the implementation of spacecraft control to build a simple website. [Lutz, 2009]

As Python directly integrates with the operating system, it is ideal for writing system administration tools and utilities. The programs can easily complete parallel processing, search for files and directories, and more. The standard library of Python supports some basic operating system tools, like, environment variables, command-line arguments, files, sockets, and more. [Lutz, 2009]

Python even comes with a built-in graphical user interface (GUI) tool called *Tkinter*. This allows Python applications to showcase GUI applications with a native look and feel of the operating system. Multiple external libraries are also available for Python to extend the features of the built-in GUI support. [Lutz, 2009]

Some other examples of the application of Python are gaming, rapid prototyping, numeric and scientific programming, database programming, and component integration [Lutz, 2009]. When reviewing the list mentioned, it can be easily determined that Python is truly a general-purpose language for tackling computational problems in multiple different domains.

2.3.4 Libraries and Frameworks

Python comes with multiple prebuilt libraries included, which are known as *standard libraries*. These libraries support tasks from text pattern matching to network scripting. Python even has support for extending the standard functionality with third-party libraries. Third-party libraries offer multiple new functionality, from serial port access to game development and more. [Lutz, 2009]

Python can be considered to be the most popular programming language for data science, where the amount of third-party data science libraries plays an important factor. The language can be used in all main areas of data science, like data preparation, data visualization, and more. When it comes to analyzing large amounts of data, big data tools are also supported with Python using third-party libraries. [Stancin, Jovic. 2019]

2.3.5 In Education

Guo [2014] reported that Python had become the predominant initial programming language at premier U.S. academic universities. Similarly, Bogdanchikov, Zhaparov, and Suliyev [2013] assert that Python is the language of choice for students in their initial year of study at most leading institutions. This widespread adoption may be attributed to Python's ease of learning, which enables new learners to develop programs within days or even hours for experienced programmers [Lutz, 2009].

2.4 ChatGPT

ChatGPT, an artificial-intelligence-driven chatbot, can engage in conversations akin to human interactions [Rudolph, Tan, Tan, 2023]. The launch of ChatGPT took place on November 30, 2022 [OpenAI, 2022]. ChatGPT can "answer follow-up questions, admit its mistakes, challenge incorrect premises, and reject inappropriate requests" [OpenAI, 2022].

ChatGPT gained huge popularity after its launch, attracting 1 million users within five days of its release [Brockman, 2022]. Roose [2022] claims that ChatGPT is the most advanced chatbot ever released to the general public. When describing the features of ChatGPT, Roose [2022] states: "It can write jokes (some of which are actually funny), working computer code, and college-level essays. It can also guess at medical diagnoses, create text-based Harry Potter games and explain scientific concepts at multiple levels of difficulty."

2.4.1 Models

ChatGPT, as its name suggests, utilizes a so-called GPT model (Generative Pre-trained Transformer) [Cooper, 2021] to be able to conversate and provide answers to questions. GPT employs deep learning techniques to produce text akin to human-generated content [Cooper, 2021].

ChatGPT can be used for free [OpenAI, 2022] or as a paid subscription service called "ChatGPT Plus" for $20 a month [OpenAI, 2023c]. The plus service offers perks like availability to the service during high-demand periods, swifter responses, and prioritized access to new features and improvements [OpenAI, 2023c].

ChatGPT currently includes three language models for its paid service: GPT-4, GPT-3.5 Default, and GPT-3.5 Legacy (see Figure 8) [OpenAI, 2023a].

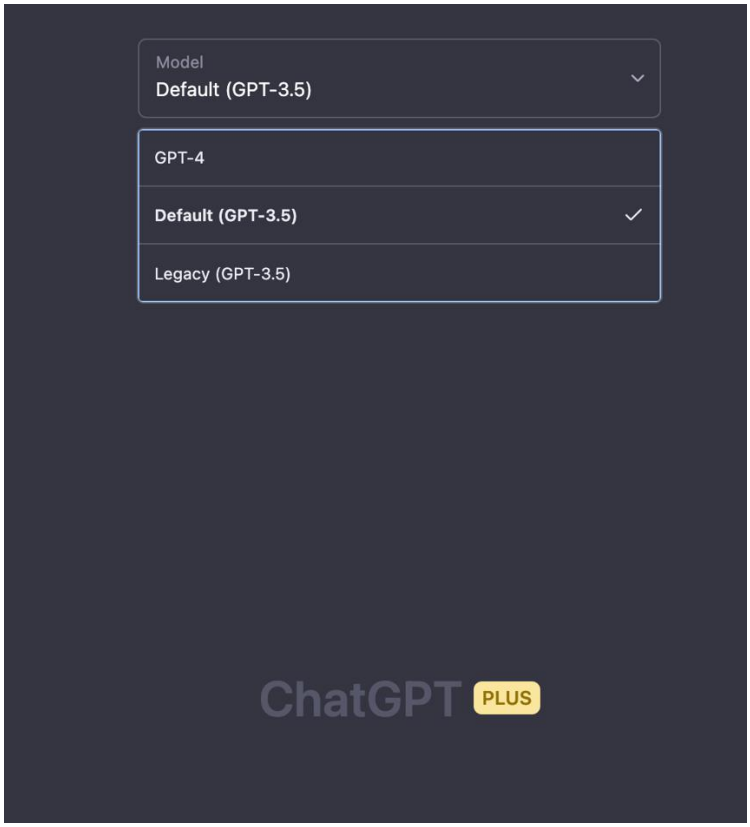

Figure 8: All ChatGPT models are offered for paid Plus subscription.

The free version of ChatGPT does not allow to selection of the model being used (see Figure 9).

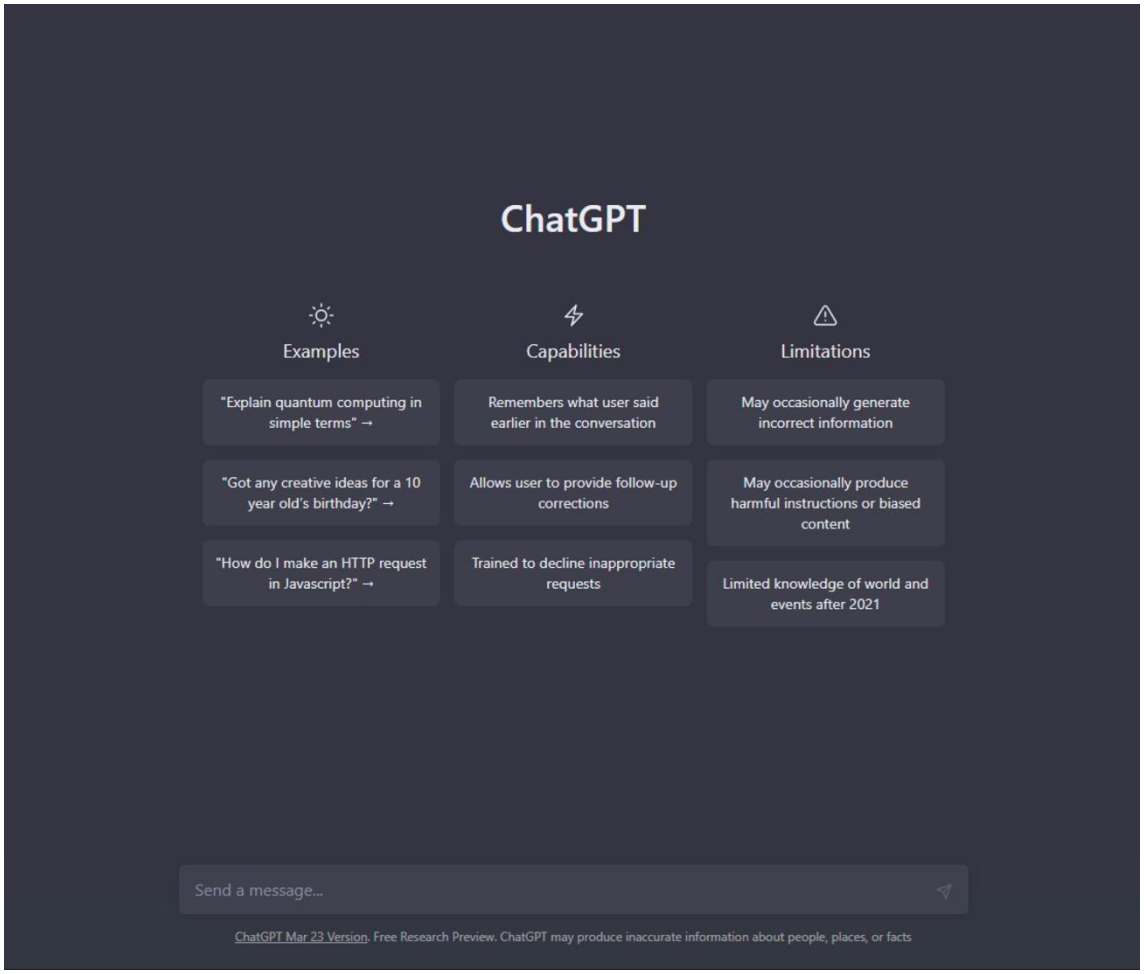

Figure 9: Free version of ChatGPT without a way to select the model being used.

2.4.2 Model Comparison

Table GPT_MODELS presents a comprehensive comparison of the available models, highlighting their distinctions [OpenAI, 2023a]. GPT-4, the most recent iteration, was launched on March 14, 2023, for paid Plus subscription plans, while GPT-3.5 models represent earlier versions [OpenAI, 2023a]. Intriguingly, the ChatGPT release notes [OpenAI, 2023a] only mention Default and Turbo GPT-3.5 models, yet the current web interface labels them as GPT-4, GPT-3.5 Default, and GPT-3.5 Legacy (refer to Table 1). Based on this observation, it can be deduced that the GPT-3.5 Turbo model corresponds to the Legacy model. This thesis will thereafter mention the models with names GPT-4, GPT-3.5 Default, and GPT-3.5 Legacy.

Model	Description	Reasoning	Speed	Conciseness
Turbo / Legacy (GPT-3.5)	The previous ChatGPT Plus model	3/5	2/5	1/5
Default (GPT-3.5)	Optimized for speed, currently available to Plus users	3/5	5/5	2/5

GPT-4	Our most advanced model, available to Plus subscribers. GPT-4 excels at tasks that require advanced reasoning, complex instruction understanding, and more creativity.	5/5	2/5	4/5

Table 1: Three different ChatGPT models alongside their overview and differences.

The GPT-3.5 model can already create text, like essays, stories, poems, and code [Rudolph, Tan, Tan, 2023]. GPT-3 generates high-quality, autonomous text on demand effortlessly and efficiently, and all it takes is composing a clear and concise prompt, such as a sentence or a question, to generate the desired output text [Floridi, Chiriatti, 2020].

Launched on March 14, 2023, the GPT-4 model serves as an enhanced version of its predecessor, GPT-3. Boasting multimodal capabilities, GPT-4 can process both text and images simultaneously, generating textual output based on the input provided. During an informal discussion, the nuances between GPT-3.5 and GPT-4 might not be readily apparent. However, as the task's intricacy increases to a certain level, GPT-4's superior reliability, inventiveness, and capacity to manage more sophisticated directives compared to GPT-3.5 become evident. [OpenAI, 2023b]

To understand the difference between the two models, OpenAI tested the models on a variety of benchmarks, including the simulation of assessment originally designed for humans. The results demonstrated a significant improvement across various evaluations. For instance, GPT-4 displayed a 22% enhancement in the Medical Knowledge Self-Assessment Program, a 14% increase in SAT math test scores, and a 21% boost in the Uniform Bar Exam (MBE+MEE+MPT) performance. [OpenAI, 2023b]

2.4.3 Interface

ChatGPT can be reached via the web address https://chat.openai.com. Once an account is created or a user logs in, they can immediately begin conversing with the language model (see Figure 10). The model selection dropdown menu is exclusively available to paid subscription plan users. The left sidebar houses a button for initiating a new chat conversation, while previous chat discussions can be accessed through the same sidebar.

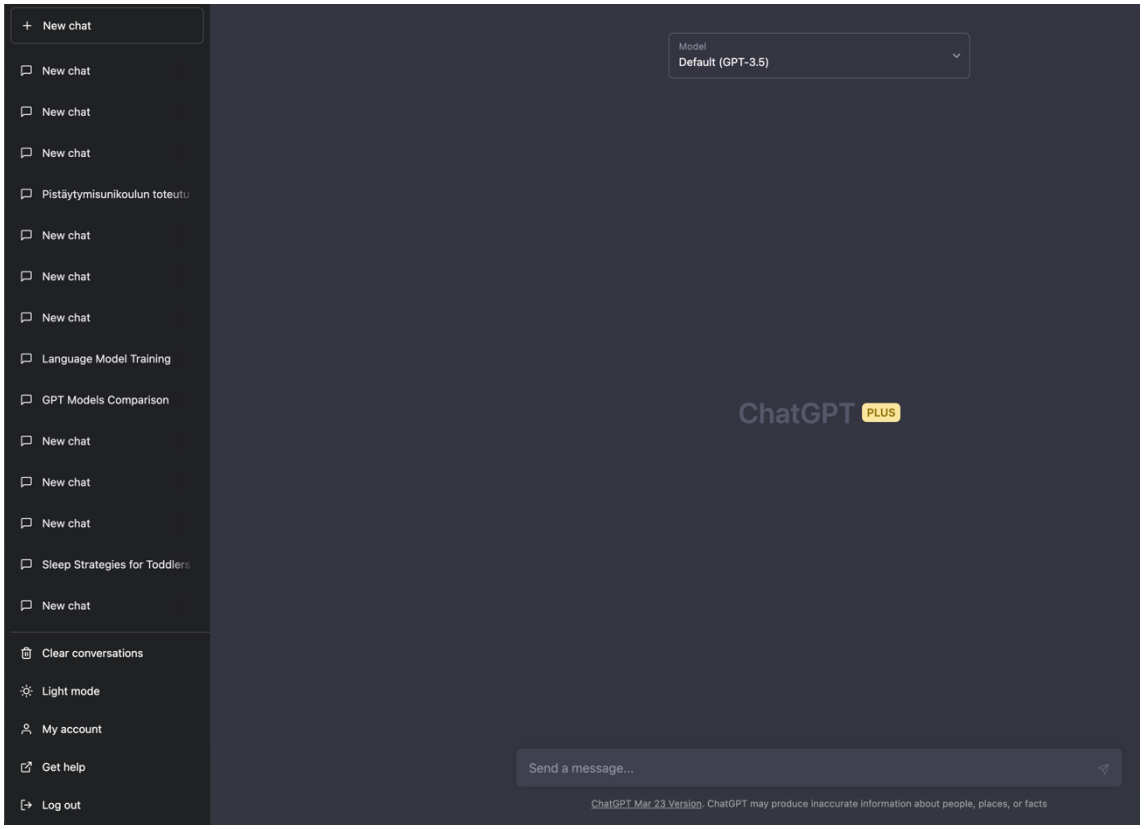

Figure 10. The ChatGPT interface upon successful login into the account with a Plus subscription.

Upon initiating a new conversation and entering "Explain quantum physics in layman's terms" in the text field at the bottom of the screen, the model promptly began formulating a response (see Figure 11).

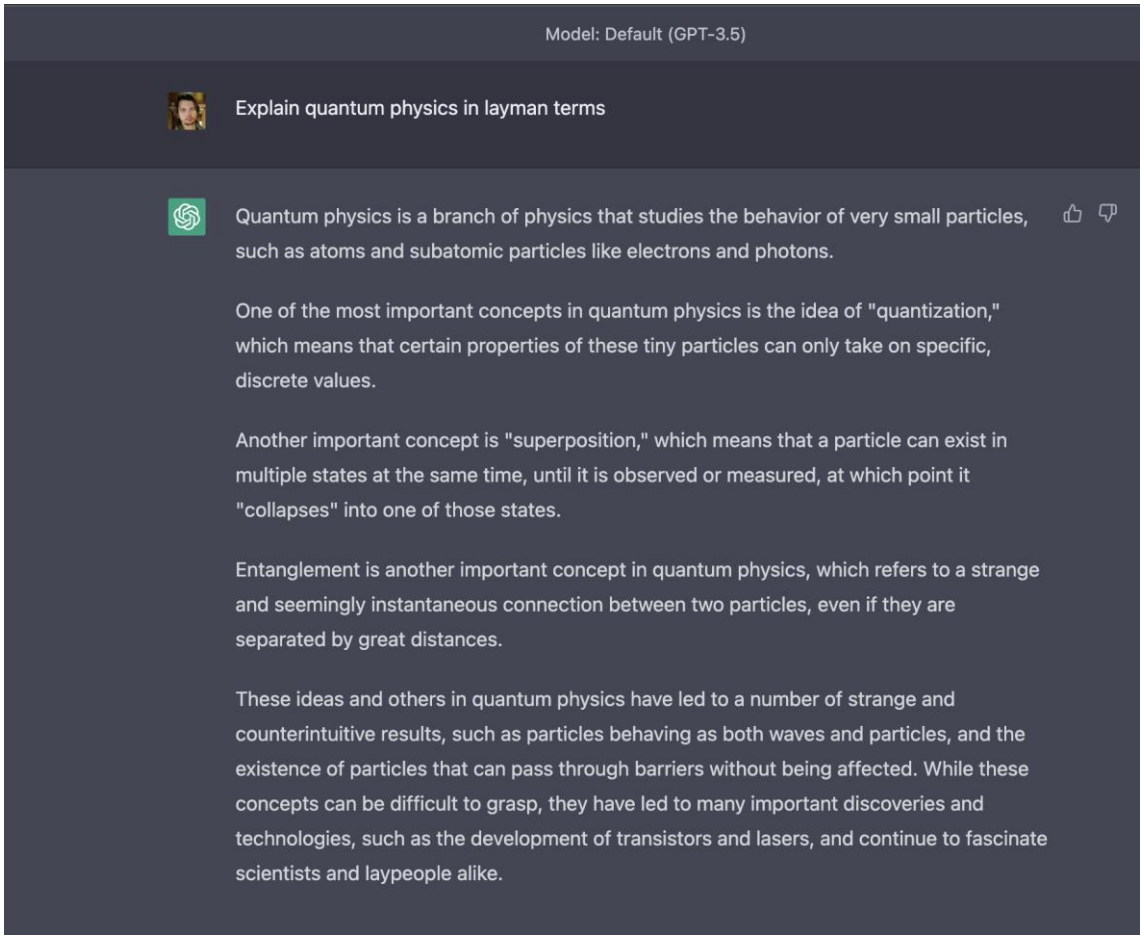

Figure 11. Asked, "Explain quantum physics in layman terms".

The dialogue may be extended by sending further messages (see Figure 13). Each model-generated reply can be rated with a thumbs up or thumbs down, and responses can be re-created as needed.

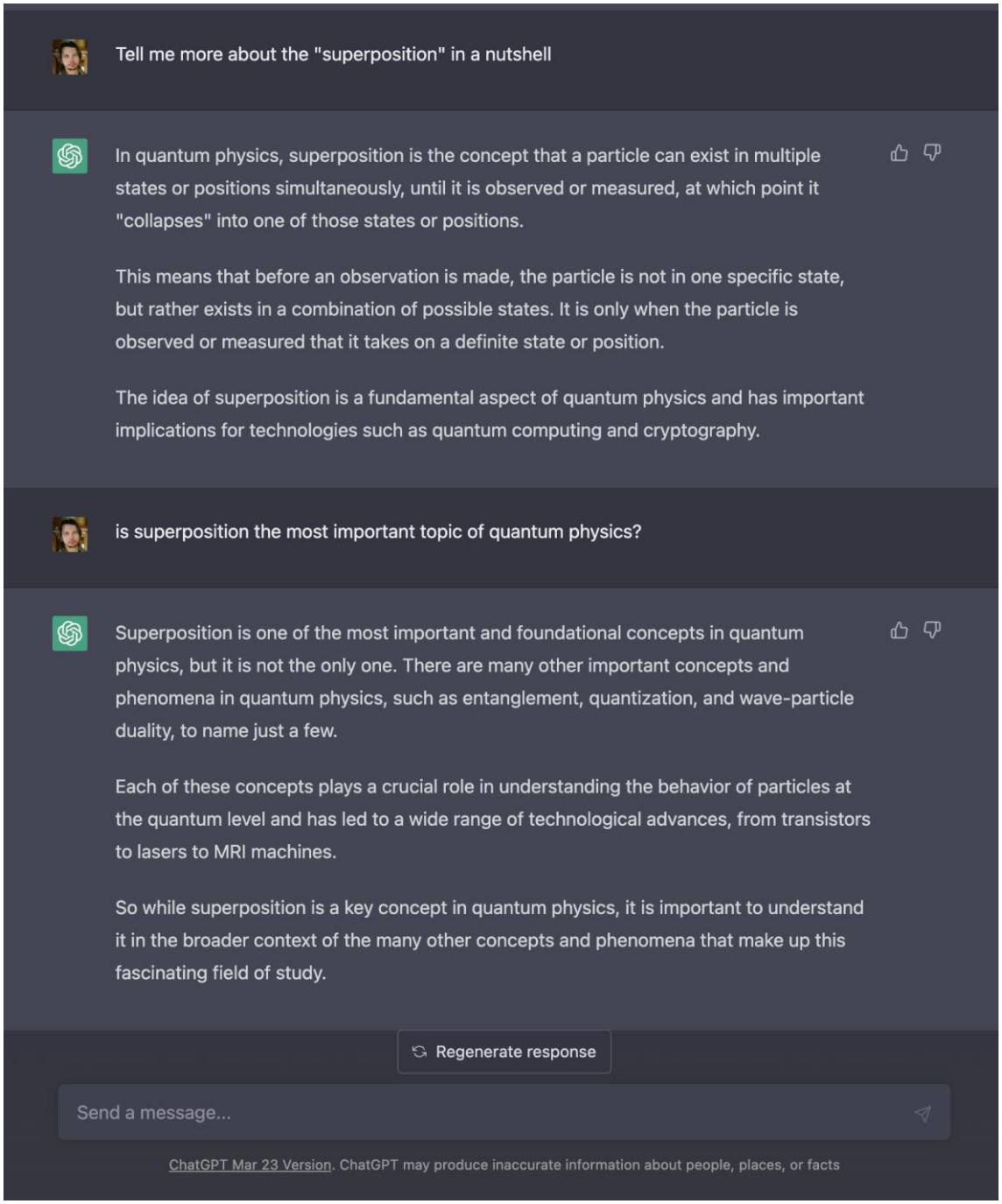

Figure 13. The dialogue can be sustained through additional messages.

Upon initiating a new chat and asking if the model recalled our prior discussion on quantum physics, it explained that every chat possesses its own unique context. (see Figure 14).

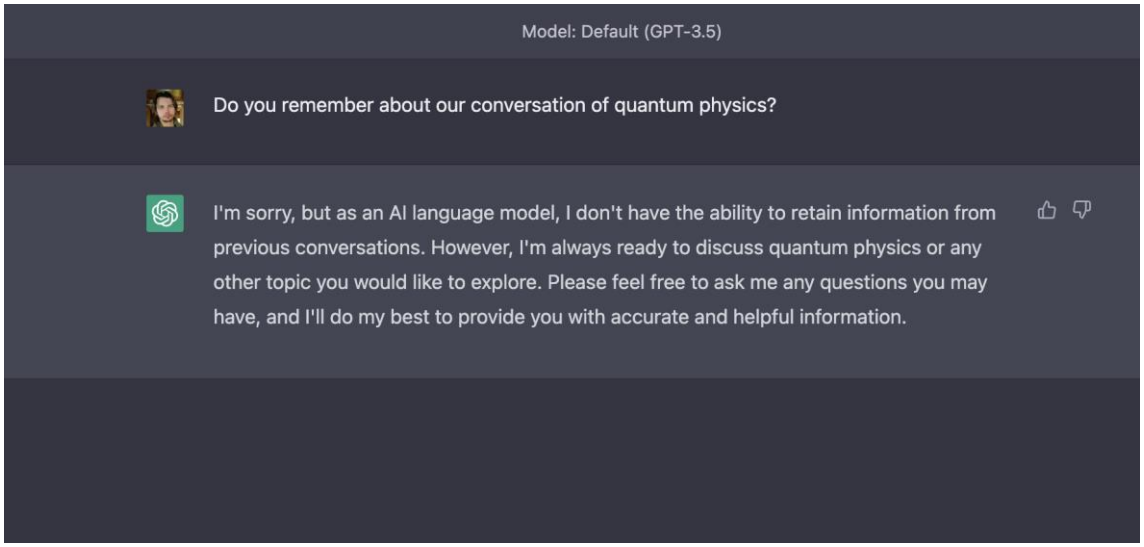

Figure 14. Inquiring about our previous discussion within a fresh chat conversation.

2.5 Previous Research

In this section, I will introduce two previous studies that have similarities to my research. Hypotheses for the experiment introduced in this thesis will be formulated based on the findings of the two relevant studies appearing in this section, providing a solid foundation for the investigation. Subsequently, the outcomes of the experiment will be analyzed and critically evaluated in the discussion section to contribute to the understanding of the research topic in a scientific context.

The first research, created by Rudolph and Tan and Tan [2023], titled "ChatGPT Participates in a Computer Science Exam", explores how well ChatGPT can complete an undergraduate computer science exam.

The second research, conducted by Jalil, *et al.* [2023], titled "ChatGPT and Software Testing Education: Promises & Perils", explores how well ChatGPT can complete practice questions related to software testing.

2.5.1 ChatGPT Completes an Exam

A study by Bordt and von Luxburg [2023] investigated the ability of ChatGPT to participate in an undergraduate computer science exam on the course "Algorithms and Data Structures." This chapter will discuss the methodology, findings, and implications of the study conducted by Bordt and von Luxburg [2023].

2.5.1.1 Methodology

Bordt and von Luxburg evaluated ChatGPT on an undergraduate computer science exam alongside 200 human students. The researchers prompted the entire exam to

ChatGPT and recorded the answers onto an exam sheet. To ensure impartiality and fairness in the evaluation process, the exam was then graded in a blind setup. The study aimed to determine the performance of ChatGPT in a challenging academic task. The study also assessed the improvements brought by GPT-4 compared to its previous version, GPT-3.5. [Bordt, von Luxburg, 2023]

2.5.1.2 Findings

The study found that ChatGPT could narrowly pass the exam, scoring 20.5 out of 40 points. This performance is impressive, given the complexity of the exam and its focus on algorithms and data structures. Additionally, the researchers discovered that GPT-4 would have achieved approximately a 17% increase in exam grades compared to GPT-3.5, attaining the same level of performance as the average human student. [Bordt, von Luxburg, 2023]

2.5.1.3 Implications and Limitations

While the results demonstrate ChatGPT's ability to succeed in challenging tasks like university exams, Bordt and von Luxburg [2023] caution against concluding that the AI model understands computer science. The exam questions were structurally similar to those found in other exams, solved homework problems, and teaching materials available online, which could have been part of ChatGPT's training data. As a result, the AI model's performance might be attributed to its huge training data rather than a profound grasp of the topic. [Bordt, von Luxburg, 2023]

2.5.2 ChatGPT Completes Software Testing Exercises

A study performed by Jalil, *et al.* [2023] titled "ChatGPT and Software Testing Education: Promises & Perils," examines the performance of ChatGPT in a software testing course, assessing its capabilities and limitations in an educational context.

2.5.2.1 Methodology

Jalil, *et al.* evaluated ChatGPT's performance by tasking it with answering practice questions in a popular software testing curriculum. The study aimed to determine the model's ability to respond to questions, provide accurate or partially accurate answers, and offer accurate or partially accurate explanations of answers. Additionally, the researchers explored the effect of prompting all questions in the same conversation. [Jalil, et al., 2023]

2.5.2.2 Findings

The study found that ChatGPT was capable of responding to 77.5% of the questions. It provided accurate or partially accurate answers in 55.6% of cases and accurate or partially accurate explanations of answers in 53.0% of situations. Furthermore, prompting

in the same conversation resulted in a higher rate of accurate answers and explanations. [Jalil, et al., 2023]

2.5.2.3 Implications and Limitations

As a result of the research conducted, Jalil, *et al.* assess the potential benefits and concerns while using ChatGPT in software testing education. The model's ability to respond to a significant proportion of questions and provide accurate or partially accurate answers and explanations suggests that it could serve as a valuable educational tool, enhancing learning experiences and supporting students and instructors. However, it also raises concerns about the potential abuse of the tool, such as students relying on ChatGPT to circumvent learning. [Jalil, et al., 2023]

3 Methods

Below are listed accurate descriptions of the materials, tools, and versions used and the procedure which can be followed to reproduce the test results.

3.1 Materials

All the programming exercises and exams for the experiment were located in the Satakunta University of Applied Sciences course *Programming Basics*. The course included a tool for automatically checking if the solution was correct or not. ChatGPT web interface was used to create answers.

3.1.1 Programming Basics Course

All exercises and exams were found in the Moodle Learning Management System (see Figure 15). The Moodle platform utilized a Coderunner plugin for automatic code checking, allowing for automatic exercise and exam grading.

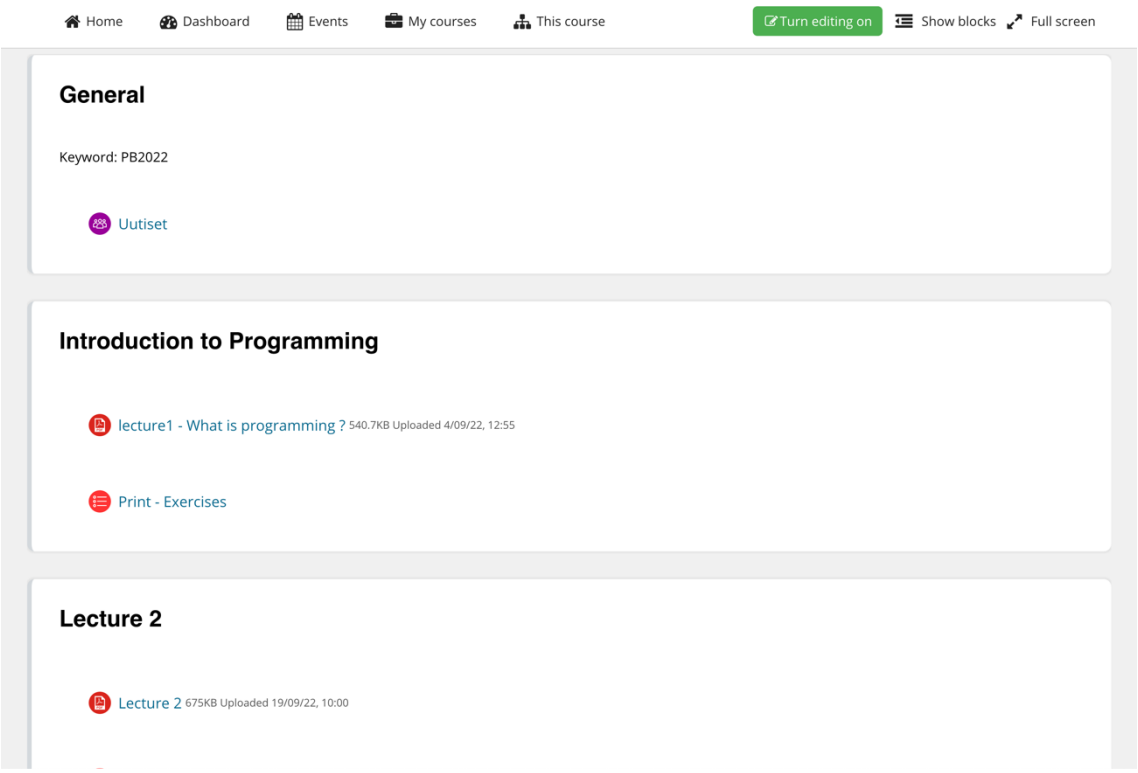

Figure 15. Moodle platform hosting the course Programming Basics

The course utilized Python as its programming language and contained 83 mandatory, automatically graded coding exercises and three exams comprising 8, 9, and 10 coding exercises, respectively. Each exam contained variating exercises. The exercises and exams covered various topics, including variables, data types, control structures, and functions. Taking into account both the exercises and exam questions, the total number of assessed questions amounts to 110.

The grading of the course consisted of grading the returned assignments and exams (see Appendix B). The course was graded based on weighted points, where the assignments had a weight of 30%, and the final exam had a weight of 70%. The best exam attempt was scored.

3.1.2 ChatGPT

ChatGPT's web interface, located at https://chat.openai.com/chat, was used to conduct the experiment. The version of the interface where the experiment was conducted is *March 23, 2023*. All three currently available models were used for the experiment (GPT-3.5 Legacy, GPT-3.5 Default, GPT-4).

3.1.2.1 Separate Conversations

Each chat conversation has its own context. The context refers to the information and previous conversation history that has been exchanged between the user and the AI model in a chat. The model also cannot be changed midway through the conversation.

Due to the fact that the context can influence the response from the model, each question was directed to a new, fresh chat conversation. Eventually, three chat windows were created for each programming exercise, as three different models were evaluated.

3.1.2.2 Tables to CSV

ChatGPT accepts only text-based input. As a result, to incorporate tables from many Moodle assignments, it was necessary to find an appropriate method for converting the table data into a suitable text format for ChatGPT. Numerous table cells in Moodle contained line breaks as well, necessitating a method to effectively feed this data to ChatGPT.

A solution was found by inquiring if ChatGPT (based on the GPT-4 model) could interpret CSV markdown syntax, using `<br>` (coming from HTML syntax) to represent line breaks (refer to Figure 16).

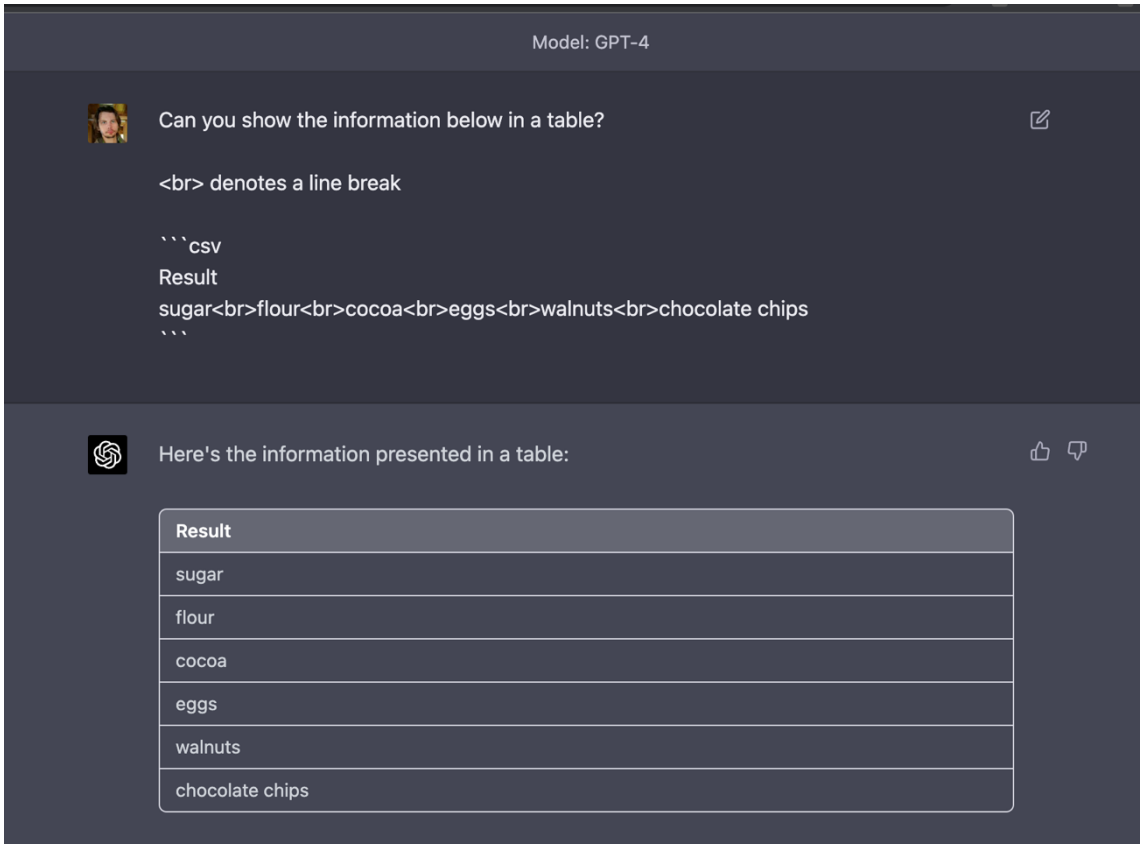

Figure 16: ChatGPT was asked to represent a markdown CSV in a tabular format.

3.1.3 Procedure

The teacher of the course provided access to the Moodle environment of the course, where all assignments and exams could be automatically completed. The process of completing the assignments is described below in linear order.

First, each assignment/exam was opened one by one in Moodle, starting from the first and ending at the last (see Figure 17)

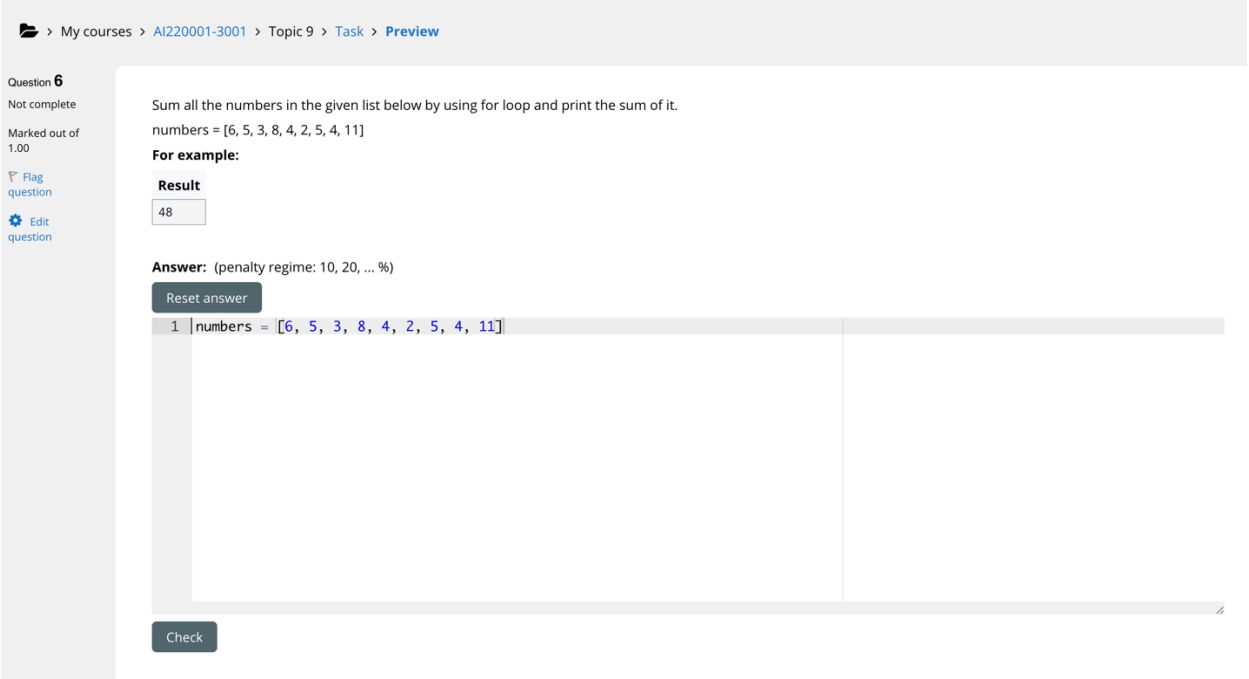

Figure 17: The environment where coding-related problems could be solved.

After this, a local markdown file was created. A premade markdown template was added to the file (see Appendix C). The assignment description was added to the file. All tables were changed to CSV format (see section "Tables to CSV" above). Finally, the file was saved as "[assignmentgroup]-[assignmentnumber].md", where *assignmentgroup* and *assignmentnumber* are variables denoting the group and assignment numbers (see Figure 18 for an example of a markdown file at this point).

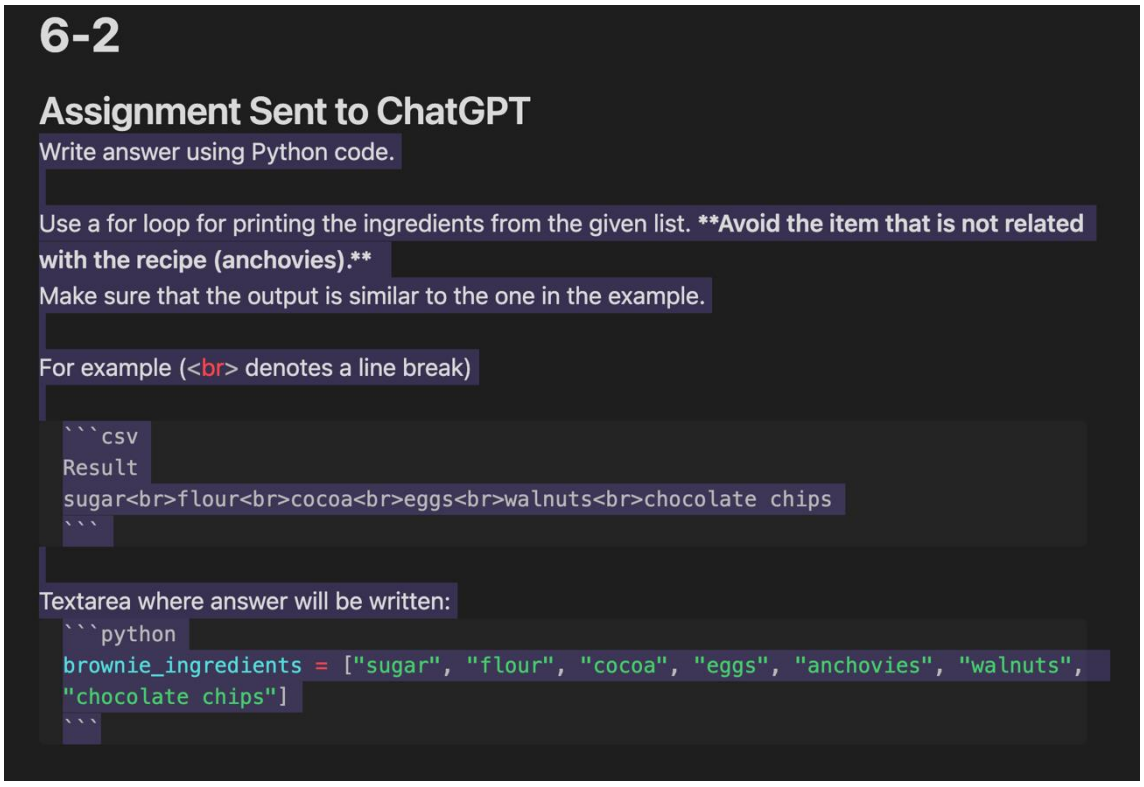

Figure 18: Example of a markdown file with the assignment description filled in.

Upon adding the assignment description to the file, a fresh chat session was initiated within the ChatGPT web interface using a selected model. The assignment description was directly transferred to ChatGPT, with the simple addition of "Write answer using Python code." as a prefix, and then submitted as input to ChatGPT (refer to Figure 19).

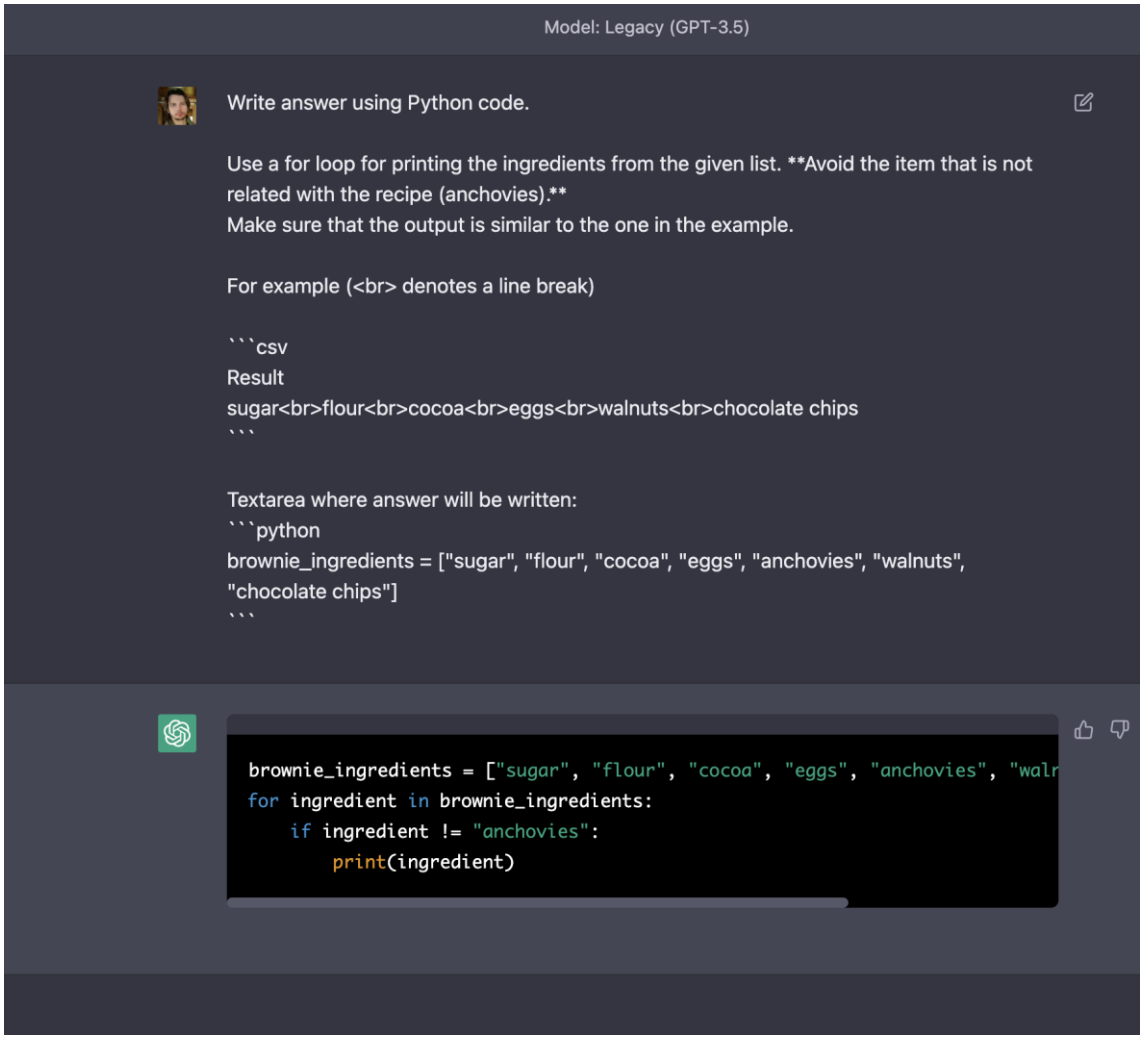

Figure 19: Question submitted to ChatGPT.

Occasionally, ChatGPT would offer explanations for the code; however, only the code segment of the response was imported into Moodle for automated assessment. Subsequently, Moodle's auto-grading system would indicate whether the submitted answer was accurate or not (refer to Figure 20 for a successful execution). Upon successful execution, the result would be recorded as *1* in a CSV file designated for outcomes (refer to Appendix A).

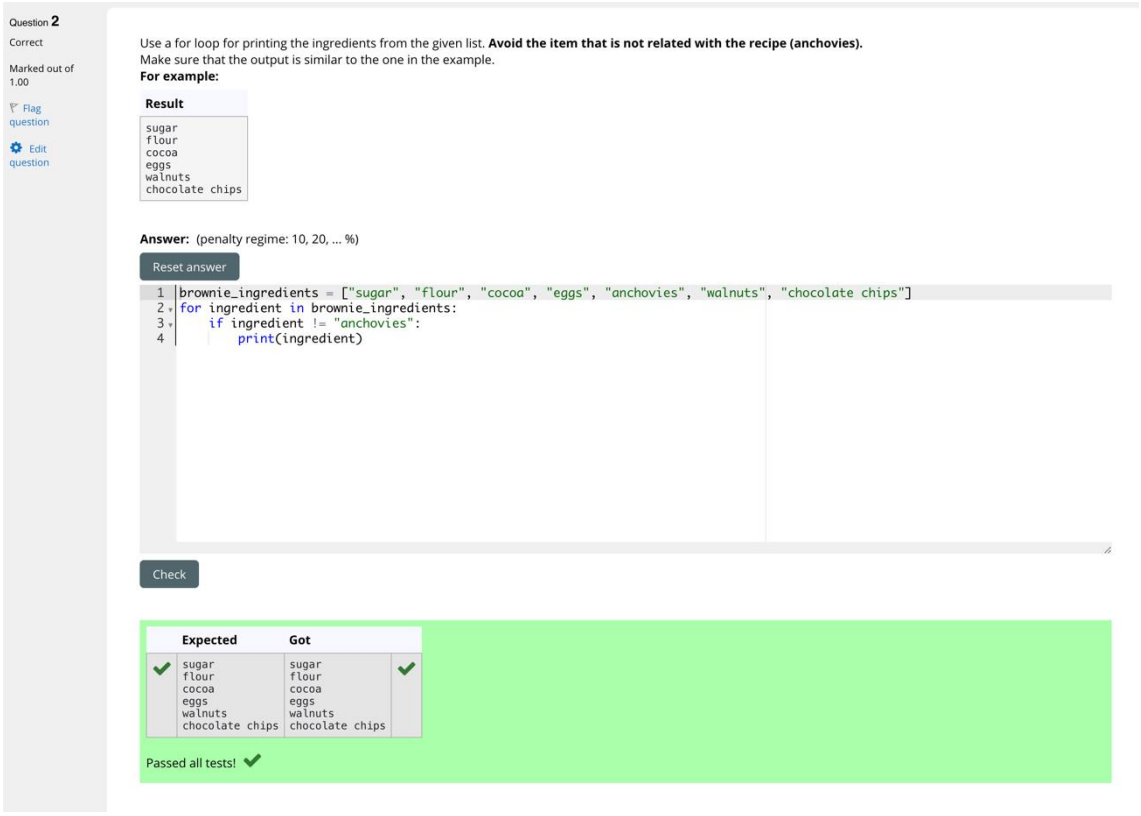

Figure 20: Code grading in Moodle passed successfully.

In the event that ChatGPT produces an incorrect response, it is granted a second chance to rectify the error by acknowledging its mistake and providing both the expected outcome and the initial output (refer to Figure 21). If the answer remains incorrect even after the second prompt, the result is recorded as 0 in a CSV file (see Appendix A). However, if the revised response is accurate, it is marked as 2 in the same CSV file (see Appendix A).

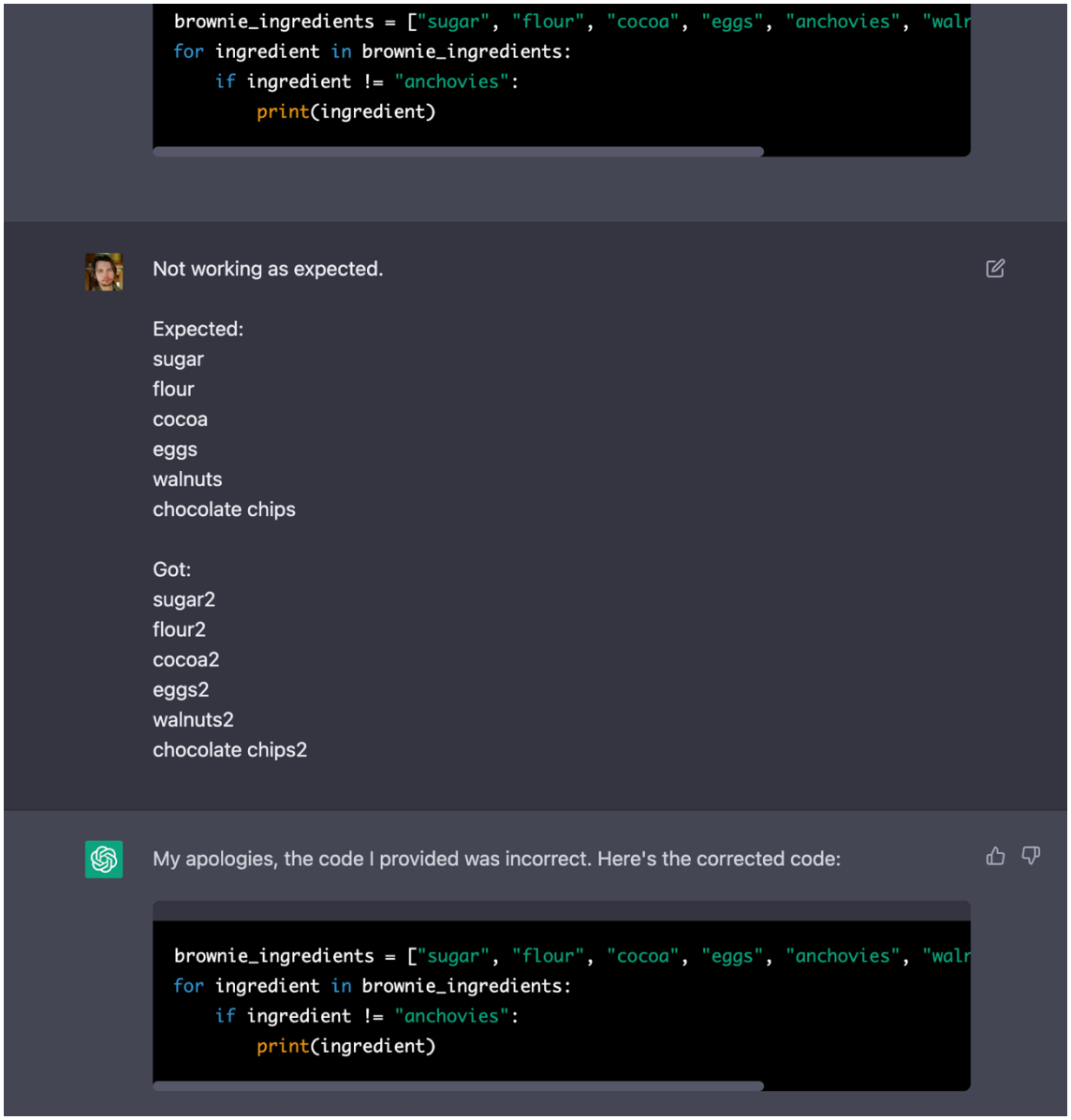

Figure 21: A second prompt was given to ChatGPT in case the answer did not work.

Upon testing each assignment, a thorough analysis and categorization of all unsuccessful tasks were carried out. More of this will be unveiled in Chapter 4.

3.2 Data Analysis

The data collected from the study included the numerical categories for each solution and the reason for failure, if applicable (see section "Procedure"). The data was analyzed using descriptive statistics to determine the proportion of correct solutions and solutions that required another prompt.

The possible outcomes of the test are:

e_0 = 0 = "Answer provided by ChatGPT did not solve the problem in Moodle even after second try"

e_1 = 1 = "Answer provided by ChatGPT solved the problem in Moodle on the first try"

e_2 = 2 = "Answer provided by ChatGPT solved the problem in Moodle on the second try"

The data is split into three different groups based on the model used: GPT-3.5 Legacy, GPT-3.5 Default, and GPT-4.

3.3 Implications for Test Reproducibility

One of the challenges associated with using large language models such as ChatGPT for programming exercises is that these models can generate different text each time they are used. This can make it difficult to reproduce the experiment and obtain consistent results.

The continuous updating of ChatGPT and its underlying models presents a challenge in terms of the reproducibility of test results. As the AI system evolves, it becomes increasingly difficult to replicate previous findings due to the potential impact of these updates.

3.4 Hypotheses

The hypotheses for this thesis are informed by previous research. Bordt and von Luxburg [2023] found that ChatGPT achieved 20.5 out of 40 points (51%) from an exam in the course "Algorithms and data structures". In comparison, the newer and more advanced GPT-4 model was projected to achieve a 17% higher score, resulting in a total of 68% [Bordt, von Luxburg, 2023].

Jalil, *et al.* [2023] found in their study that ChatGPT was capable of replying to 77.5% of the questions examined and presenting exact or partially exact answers in 55.6% of situations. The course related to the experiment conducted by Jalil, *et al.* [2023] was related to Software Testing. The study included conceptual and code-related questions, and ChatGPT could answer 83.3% of all code-related questions correctly.

The studies referenced earlier focused on more advanced computer science courses rather than beginner-level courses. As the course used in this thesis is designed for beginners, it is reasonable to anticipate that the success rates for ChatGPT will be higher compared to those reported in the previous studies.

Previous studies reported success rates of 51% and 83%, with an average of 67%. Taking into account the findings from these prior investigations and considering the lower level of difficulty in the course used for this thesis, I propose the following hypotheses: GPT-3 is expected to achieve approximately 10% higher performance compared to the average of the previous two studies, resulting in a success rate of around 77%, while GPT-4 is anticipated to perform about 10% better than GPT-3, yielding a success rate close to 88%

4 Results

This section presents the outcomes of the experiment. The efficacy of the models was tested, and each test had three possible outcomes: ChatGPT solved the programming exercise on the first attempt (e_1), ChatGPT solved the programming exercise on the second attempt (e_1), ChatGPT did not solve the programming exercise (e_0). The information that ChatGPT solved the exercise on the first or second attempt is aggregated to equation [WP]:

$WP = e_1 + e_2$

Equation WP

Where:

e_1 denoted, "Answer provided by ChatGPT solved the problem in Moodle on the first try"

e_2 denoted, "Answer provided by ChatGPT solved the problem in Moodle on the second try"

All the data from the experiment was collected in an Excel sheet (see Appendix A). The results of the data collected are presented in this chapter in tabular form. The results are further analyzed in section 5.

4.1 Mandatory Exercises

The course in which the experiment was conducted contained mandatory exercises which students were required to complete. Each mandatory exercise was inputted to ChatGPT as a prompt one by one. Table 3 contains the results of the experiment for mandatory exercises. The first column depicts the model used, where GPT-3.5 Legacy is the oldest model and GPT-4 is the latest model. The model, the result if the answer did not work, the result if the answer worked, if the answer worked with a prompt, and the combined result of working results + working with prompt are represented in the table. N is 83 for each test group.

Model	e_0 (Not working)	e_1 (Working)	e_2 (Working after reprompt)	WP
GPT-3.5 Legacy	17% (14 / 83)	80% (66 / 83)	4% (3 / 83)	83% (69 / 83)
GPT-3.5 Default	13% (11 / 83)	82% (68 / 83)	5% (4 / 83)	87% (72 / 83)
GPT-4	11% (9 / 83)	84% (70 / 83)	5% (4 / 83)	89% (74 / 83)

Table 3. Statistical summaries of mandatory exercises. N = 83.

EC1: Trend of improvement in each subsequent ChatGPT model on completing the course programming exercises.

EC2: The latest GPT-4 model performs better than older models in completing the course programming exercises.

4.2 Exams

The course for the experiment included three exams. Each of the exams was quite identical, with some variation in the questions. All programming exercises of the exams were again inputted to ChatGPT, each question at a time, and the performance of each model was determined.

4.2.1 Exam #1

Table 4 contains the results of the experiment for exam #1. The first column depicts the model used, where GPT-3.5 Legacy is the oldest model and GPT-4 is the latest model. The model, the result if the answer did not work, the result if the answer worked, if the answer worked with a prompt, and the combined result of working results + working with prompt are represented in the table. N is 8 for each test group.

Model	e_0 (Not working)	e_1 (Working)	e_2 (Working after reprompt)	WP
GPT-3.5 Legacy	25% (2 / 8)	75% (6 / 8)	0% (0 / 8)	75% (6 / 8)
GPT-3.5 Default	25% (2 / 8)	75% (6 / 8)	0% (0 / 8)	75% (6 / 8)
GPT-4	25% (2 / 8)	63% (5 / 8)	13% (1 / 8)	75% (6 / 8)

Table 4: Statistical summaries of exam #1. N = 8.

EC3: Each ChatGPT model demonstrates a very similar performance with exam #1.

4.2.2 Exam #2

Table 5 contains the results of the experiment for exam #2. The first column depicts the model used, where GPT-3.5 Legacy is the oldest model and GPT-4 is the latest model. The model, the result if the answer did not work, the result if the answer worked, if the answer worked with a prompt, and the combined result of working results + working with prompt are represented in the table. N is 9 for each test group.

Model	e_0 (Not working)	e_1 (Working)	e_2 (Working after reprompt)	WP
GPT-3.5 Legacy	0% (0 / 9)	100% (9 / 9)	0% (0 / 9)	100% (9 / 9)
GPT-3.5 Default	22% (2 / 9)	78% (7 / 9)	0% (0 / 9)	78% (7 / 9)
GPT-4	0% (2 / 9)	100% (9 / 9)	0% (0 / 9)	100% (9 / 9)

Table 5: Statistical summaries of exam #2. N = 9.

EC4: There is variation in the performance of the models, and the same trend seen in assignments is no longer seen within exam #2.

EC5: The variation in the performance of exams might result from a small number of observations within the exams.

4.2.3 Exam #3

Table 6 contains the results of the experiment for exam #3. The first column depicts the model used, where GPT-3.5 Legacy is the oldest model and GPT-4 is the latest model. The model, the result if the answer did not work, the result if the answer worked, if the answer worked with a prompt, and the combined result of working results + working with prompt are represented in the table. N is 10 for each test group.

Model	e_0 (Not working)	e_1 (Working)	e_2 (Working after reprompt)	WP
GPT-3.5 Legacy	30% (3 / 10)	70% (7 / 10)	0% (0 / 10)	70% (7 / 10)
GPT-3.5 Default	30% (3 / 10)	50% (5 / 10)	20% (2 / 10)	70% (7 / 10)
GPT-4	10% (1 / 10)	90% (9 / 10)	0% (0 / 10)	90% (9 / 10)

Table 6: Statistical summaries of exam #3. N = 10.

EC6: Two first models share the same completion percentage, and only the latest model is better than the older models when it comes to exam #3.

4.2.4 Statistical Inferences from Exams

The results of all three exams were aggregated into one result set for further analysis. Table 7 presents the results of statistical analyses conducted on exams. The *WP* equation for exams was utilized to compute the mean, range, standard deviation, min, max, and 95% confidence interval for the given data set.

Model	Mean	Range	Standard Deviation	Confidence Interval (95%)	Min	Max
GPT-3.5 Legacy	81.67%	30%	13.08%	53.49%,109.85%	70%	100%
GPT-3.5 Default	74.33%	8%	3.01%	70.03%,78.63%	75%	87%
GPT-4	88.33%	25%	12.74%	63.21%,113.45%	75%	100%

Table 7: The three exams are taken for each model and used to conduct statistical inferences.

EC7: The results are mixed when it comes to all exam attempts. The latest model is the most performant, but the second latest model (GPT-3.5 Default) performs worse than the oldest model.

EC8: The GPT-4 model performs better than other models with the exams.

4.3 Combined Results

For further analyzing the results from both the assignments and the exams, the results from both assignments and exams were aggregated into one result set. Table 8 contains the aggregated results of mandatory exercises, exam #1, exam #2, and exam #3 together. The first column depicts the model used, where GPT-3.5 Legacy is the oldest model and GPT-4 is the latest model. The model, the result if the answer did not work, the result if the answer worked, if the answer worked with a prompt, and the combined result of working results + working with prompt are represented in the table. N is 111 for each test group.

Model	e_0 (Not working)	e_1 (Working)	e_2 (Working after reprompt)	WP
GPT-3.5 Legacy	17% (19 / 111)	79% (88 / 111)	3% (3 / 111)	82% (91 / 111)
GPT-3.5 Default	16% (18 / 111)	77% (86 / 111)	5% (6 / 111)	83% (92 / 111)

Model	e_0 (Not working)	e_1 (Working)	e_2 (Working after reprompt)	WP
GPT-4	11% (12 / 111)	84% (93 / 111)	5% (5 / 111)	88% (98 / 111)

Table 8: Statistical summaries of mandatory exercises, exam #1, exam #2, and exam #3 together. N = 111.

4.3.1 Statistical Inferences from All Results

Statistical inferences were concluded from all the results, where both exams and assignments were combined into one aggregated result set. Table 9 presents the results of statistical analyses conducted on both exams and assignments. The *WP* equation for assignments and exams was utilized to compute the mean, range, standard deviation, min, max, and 95% confidence interval for the given data set.

Model	Mean	Range	Standard Deviation	Confidence Interval (95%)	Min	Max
GPT-3.5 Legacy	82%	30%	12.14%	68.92%,95.08%	70%	100%
GPT-3.5 Default	83%	17%	7.98%	68.75%,86.25%	75%	87%
GPT-4	88%	25%	9.57%	77.58%,99.42%	75%	100%

Table 9: All assignments and exams are taken for each model and used to conduct statistical inferences.

PEC1: The latest GPT-4 model performs better than other models when all results from the experiment are aggregated into one result set.

PEC2: Trend of improvement in each subsequent ChatGPT model on completing the programming exercises when all results from the experiment are aggregated into one result set.

4.4 Model Grades

Finally, the grades for each ChatGPT model were calculated. The calculation of the final grades was conducted by taking into account the weight of the best exam attempt and assignments (exam 60%, assignments 40%, see appendix B) by using a formula represented below:

GRADE = a / 100 * 0,6 + b / 100 * 0,6

Equation GRADE

Where:

a denoted the model's *WP* equation percentage score from the mandatory exercises
b denoted the model's top percentage score from the exam attempts

Table 10 contains the grades which each model would have achieved from the course based on the course weightings (see Appendix B). These are the same grades that the human student would have achieved with similar performance. The equation WP is used for each group.

Model	Percentage (WP)	Grade (WP)
GPT-3.5 Legacy	93%	5
GPT-3.5 Default	87%	4
GPT-4	97%	5

Table 10: Grades achieved for each model used during the experiment.

EC9: Surprisingly, the oldest model achieved a better grade compared to the second oldest model.

PEC3: The latest GPT-4 model achieved a very high 97% course completion score, which would result in a grade of 5.

4.5 Categorized Problems

After the experiment was conducted by inputting all the questions one by one to ChatGPT and the results were gathered, all questions which ChatGPT did not provide an answer for were further analyzed to try to categorize the problems into groups for statistical analysis.

Table 11 contains e_0 "Answer provided by ChatGPT did not solve the problem in Moodle even after second try" results of the experiment, categorized by commonly occurring problems. The first column depicts the model used, where GPT-3.5 Legacy is the oldest model and GPT-4 is the latest model. The following information is represented in the table:

- **Model**: The model used
- **Space**: Represents a case where the model did not understand a space character being used in the code or in the output
- **Input Print**: Represents a case where the model did not understand that Python's print command also outputs the given text to the console
- **Slice**: Represents a case where the model could not slice text array into new output using numeric indexes

- **Other**: Any other reason which does not belong to the other 3threecategories.

Model	Space	Input Print	Slice	Other
GPT-3.5 Legacy	22% (4 / 18)	11% (2 / 18)	11% (2 / 18)	56% (10 / 18)
GPT-3.5 Default	29% (5 / 17)	12% (2 / 17)	12% (2 / 17)	47% (8 / 17)
GPT-4	0% (0 / 12)	25% (3 / 12)	25% (3 / 12)	50% (6 / 12)

Table 11: Unsuccessful responses generated by ChatGPT categorized.

PEC4: All the ChatGPT models still struggled with very basic programming functionality, like being able to slice a string.

4.6 Empirical Conclusions

This chapter included four primary empirical conclusions and nine empirical conclusions. All these conclusions are visualized in this chapter as a table for easier reference. These conclusions are further discussed in section 5.

Table 12 presents the nine empirical conclusions that could be drawn from the results of the experiment.

Identifier	Empirical conclusion
EC1	The trend of improvement in each subsequent ChatGPT model on completing the course programming exercises.
EC2	The latest GPT-4 model performs better than older models in completing the course programming exercises.
EC3	Each ChatGPT model demonstrates a very similar performance on exam #1.
EC4	There is variation in the performance of the models, and the same trend seen in assignments is no longer seen within exam #2.
EC5	The variation in the performance of exams might result from a small number of observations within the exams.

EC6	The two first models share the same completion percentage, and only the latest model is better than the older models when it comes to exam #3.
EC7	The results are mixed when it comes to all exam attempts. The latest model is the most performant, but the second latest model (GPT-3.5 Default) performs worse than the oldest model.
EC8	The GPT-4 model performs better than other models with the exams.
EC9	Surprisingly, the oldest model achieved a better grade compared to the second oldest model.

Table 12: Empirical conclusions formed from the data.

Table 13 represents the four primary empirical conclusions drawn from the results of the experiment.

Identifier	Primary empirical conclusion
PEC1	The latest GPT-4 model performs better than other models when all results from the experiment are aggregated into one result set.
PEC2	The trend of improvement in each subsequent ChatGPT model on completing the programming exercises when all results from the experiment are aggregated into one result set.
PEC3	The latest GPT-4 model achieved a very high 97% course completion score, which would result in a grade of 5.
PEC4	All the ChatGPT models still struggled with very basic programming functionality, like being able to slice a string.

Table 13: Empirical conclusions formed from the data.

5 Discussion

The primary empirical conclusions (PECs) drawn in section 4.6 are reflected in this section. Students and teachers are the main roles affected by this study; thus, the implications for these two roles are also discussed in this section. Finally, a comparison to previous research is discussed.

5.1 Empirical Conclusions

It was found evident that the latest GPT-4 model performs better than other models (PEC1). This indicates that the models have seen improvements, at least in the small domain of basic Python programming, and there might even be improvement possibilities with potentially upcoming models.

It was noted in PEC2 that each ChatGPT model did show a trend of subsequent improvement. Again, it seems like there are future improvement possibilities available. Further on, it could be interesting to see what drives the improvements in the models. Has the architecture changed? Is it the training data? Does the RLHF concept improve the model?

GPT-4 model achieved the highest grade of 5 with a course completion score of 97 / 100 (PEC3). At the same time, this can be alarming and commendable. It can raise questions about academic integrity and the potential misuse of the tool. On the other hand, the tool could be beneficial for student learning, like, for example, for being a personal 24 / 7 tutor.

Even the latest GPT-4 model has its flaws, as demonstrated by PEC4. The tool still struggles with some basic programming tasks. This raises a question: why do the tools struggle with these specific tasks? Is it the training data or the architecture of the model? Could these problems be solved in future updates? Are the problems occurring due to too small context? Due to this, it could be interesting to have follow-up research that builds larger projects with the Python programming language to view if the problems still arise in those kinds of situations.

5.2 Implications for Students

Students are being able to access powerful artificial intelligence tools to help with their coursework and to even achieve the highest grade with the tool (PEC3), being there is no supervision provided by the institution for the completion of the course. Usually, the

assignments in academic situations are conducted unsupervised, and the exams are supervised; thus, in this kind of situation, the student could harm himself by completing the assignments using ChatGPT or a similar tool and by failing the supervised exam.

As the performance of the tool is very high (PEC3), the tool can be misused by the student. The tool is powerful and can still be used to help with harder assignments instead of just blatantly completing the exercises, but students now have more responsibility to think their selves when and how to use these kinds of tools to provide the best learning outcome. In short, the students can use these tools to aid in their learning, but have also the ability to conduct cheating.

ChatGPT models still have some limitations when it comes to programming tasks (PEC4), but as there is a trend of improvement (PEC2), the tools are likely developing, and the limitations found in this study are likely to be diminished. ChatGPT can already help students complete most of the exercises of the course to a certain extent.

While using tools like ChatGPT, the student should opt to use the latest GPT-4 model. The reason for this choice is that it was found in this study that the latest model performs better in comparison with the older models (PEC1).

5.3 Implications for Teachers

There is a trend of improvement with the ChatGPT models (PEC1), which indicates that the pace of artificial intelligence development in the field of large language models is consistent and fast. The pace of development is likely rising, and upcoming models in the future will certainly outperform the ones experimented with.

The latest GPT-4 model was found to be superior compared to older models (PEC3). The latest GPT-4 model could already complete the whole course with the best grade of 5, with a 97% completion score (PEC3). Teachers should take academic integrity into account while creating similar courses as students are currently being able to complete the course using artificial intelligence. This brings forward a need to validate that the student has the required knowledge of the course upon completing the course and that the student did not blatantly just use artificial intelligence to complete the course. Supervised exams are one concrete example of this.

ChatGPT still struggled with some basic programming functionality (PEC4). Based on this, the teacher might come up with a conclusion to adjust the course assignments to "trick" the artificial intelligence tools. However, as the tools still show improvement (PEC2), these kinds of workarounds could be a very short-term solution.

ChatGPT and other tools are now part of the society. Instead of worrying about students cheating through the courses with the tools provided, teachers should embrace the change and come up with solutions on how to adjust their courses to the availability of these new tools. In the future, it might not be the best approach to only test for completion of projects or assignments but to test for a more holistic approach where the student is required to demonstrate the larger learning process that was achieved during the whole project or while completing the assignments.

5.4 Comparison to Previous Research

Bordt and von Luxburg [2023] found that ChatGPT achieved 20.5 out of 40 points (51%) from an exam in the course "Algorithms and data structures". The experiment conducted in this thesis demonstrated that GPT-3 could achieve a higher success rate of around 83% for the beginner-level Python course (taking into account both assignments and exams). This can be attributed to the lower level of difficulty in the course used for this thesis, as well as the improvements in the GPT-3 model compared to the earlier ChatGPT models.

In the same study, Bordt and von Luxburg [2023] expected that the more advanced GPT-4 model would achieve a 17% higher score, resulting in a total of 68%. Our research showed that GPT-4 was able to perform even better, achieving a success rate of 88% in the beginner-level Python course (taking into account all exams and assignments). This indicates that the GPT-4 model has significantly improved compared to its predecessors, particularly when tackling less complex tasks.

Jalil, et al. [2023] found in their study that ChatGPT demonstrated the ability to respond to 77.5% of the examined questions, providing either completely correct or partially correct responses in 55.6% of cases. The course related to their experiment was focused on software testing, which is a more advanced subject compared to the beginner-level Python course used in this study. During the experiment conducted in this thesis, it was noticed that GPT-4 was able to answer a larger proportion of questions correctly, suggesting that the model performs better in tasks with lower complexity.

In conclusion, the study's findings support the hypothesis that both GPT-3 and GPT-4 models would perform better in a beginner-level Python course compared to previous studies involving more advanced computer science courses. The main reason is likely due to the fact that the level of requirements is less demanding in a beginner-level Python course. However, it is important to note that the models still face challenges in understanding specific Python concepts, such as the input command and slicing, which could be further improved in future iterations of the model.

6 Conclusion

This chapter concludes the whole thesis. The chapter includes answers to research questions, limitations of the research, and suggestions for future research.

6.1 Answers to Research Questions

The following were the research questions of the thesis:

1. How do various ChatGPT models perform in completing beginner-level coding-related Python course assignments and exams?

2. How does the performance of ChatGPT models measure to each other and that of human students in tackling Python course assignments and exams?

3. What types of errors or incorrect responses are produced by ChatGPT models when attempting Python course assignments and exams?

6.1.1 Performance in Assignments and Exams

Regarding research question 1, the performance of the models in completing beginner-level Python course assignments and exams shows a clear trend of improvement in each subsequent version of the model. GPT-4 excels GPT-3.5 Legacy and GPT-3.5 Default in every category with an 89% completion ratio (WP) in mandatory exercises and an overall completion rate of 88% across both exercises and exams.

The results from the exams appeared more mixed. The standard deviation and range widely varied between the exam attempts. This could be explained by such a small amount of exercises and also by a statement provided in Appendix B regarding that Exam #1 was much harder even for the students. GPT-3.5 Default also introduced fewer points than other models, which is against the results from the mandatory exercises.

6.1.2 Comparison to Human Students

In response to research question 2, the comparison between ChatGPT models and human students reveals that each ChatGPT model performed better than the average human student. As mentioned in Appendix B, the average grade for the students in the class was 4 (80%). ChatGPT models gained 87%, 93%, and 97% scores. GPT-4 performed the best, and surprisingly the GPT-3.5 default model performed the worst.

6.1.3 Error Patterns and Trends

Addressing research question 3, it was evident that certain recurring issues could be grouped into distinct categories, while the models encountered diverse types of challenges. Nevertheless, approximately 50% of the problems remained within the "Other"

category. All the models revealed a common challenge in understanding Python's input command and its associated output while trying to slice a string using numeric indexes.

The GPT-4 model exhibits noticeable improvements compared to its predecessors, particularly in eliminating issues related to the space character not being considered in the output. However, the GPT-4 model, like its older counterparts, continues to face challenges with problems related to slicing and handling inputs.

6.2 Limitations of the Research

The study conducted, as with any other research, can face certain limitations which should be taken into account when interpreting the results. Taking these limitations into account can give a better overview of the research conducted in this thesis and help guide future research on the topic.

Mainly, the model has the potential to generate different results each time. This has the potential to skew the results. Language models also generate different results based on the prompt given, so a more detailed prompt prepended to the question could also be explored for better results.

The Moodle course contained an adequate amount of programming exercises, but there were only a small amount of exam questions available. The way how the assignment descriptions are written is crucial, and more detailed, longer, or even shorter descriptions could change how the model could answer the question. The course selected for the experiment was also a beginner course, and the results can be different on any medium or advanced-level courses within the same subject.

6.3 Suggestions for Future Research

Taking the findings and limitations of this study into account, there are multiple approaches and recommendations that could be followed in the future. At first, the experiment could be conducted with a more advanced course. The model could also be queried multiple times (for example, three times) with the same prompt to determine if the answer changes.

The course could also be completed with different models. Some open-source models could be used to conduct the same experiment to explore their potential. Whenever a new version of GPT-4 comes up (like GPT-4.5), the same experiment could be repeated with that model, and the error trends could be further inspected. Fine-tuning a model to give more precise answers to the questions could also be explored.

The potential collaboration of humans and AI could also be explored. It would be interesting to research how humans could use the given AI-assisted tools during their studies. The potential benefits and drawbacks of this approach could be explored.

It would also be interesting to conduct research on why the models have improved. This study only concluded that GPT-4 performed better than older models, but the study only focused on statistical analysis. The factors that affect the development of the model could be explored, like the amount of training, the type of training data used, the RLHF method used, and the architecture of later models.

7 References

Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer.

Bogdanchikov, Zhaparov, Suliyev. 2013. *Python to learn programming*. Journal of Physics Conference Series 423(1):2027.

Bordt, von Luxburg. 2023. *ChatGPT Participates in a Computer Science Exam*. arXiv preprint arXiv:2303.09461.

Brockman. 2022. ”ChatGPT just crossed 1 million users; it's been 5 days since launch”. *https://twitter.com/gdb/status/1599683104142430208*.

Brown, et al. 2020. *Language Models are Few-Shot Learners*. NIPS'20: Proceedings of the 34th International Conference on Neural Information Processing Systems, Article No.: 159.

Chapman, Hall. 2010. *Handbook of Natural Language Processing - Second Edition*. CRC Press.

Cooper. 2021. OpenAI GPT-3: Everything you need to know. Springboard. https://www.springboard.com/blog/data-science/machine-learning-gpt-3-open-ai/

Dastres, Soori. 2021. *Artificial Neural Network Systems*. International Journal of Imaging and Robotics, 21(2):13-25.

Devlin, et al. 2019. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. Proceedings of NAACL-HLT 2019.

Douglas. 2023. *Large Language Models*. arXiv preprint arXiv:2307.05782.

El Naqa, Li, Murphy. 2015. *Machine Learning in Radiation Oncology*. Springer.

Flasiński. 2016. *Introduction to Artificial Intelligence*. Springer.

Floridi, Chiriatti. 2020. *GPT-3: Its Nature, Scope, Limits, and Consequences*. Springer Link. https://link.springer.com/article/10.1007/s11023-020-09548-1

Guo. 2014. Python is now the most popular introductory teaching language at top U.S. universities. *Communications of the ACM*. https://cacm.acm.org/blogs/blog-cacm/176450-python-is-now-the-most-popular-introductory-teaching-language-at-top-us-universities/fulltext

Hu. 2023. ChatGPT sets record for fastest-growing user base - analyst note. Reuters. https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/.

Jalil, Rafi, LaToza, Moran, Lam. 2023. *ChatGPT and Software Testing Education: Promises & Perils*. 2023 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW).

Jurafsky, Martin. 2007. *Speech and Language Processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Prentice Hall.

Korzeniowski, Goczyla. 2019. Artificial intelligence for software development — the present and the challenges for the future.

Kranich. 2004. *The Information Commons: a public policy report*. Brennan center for justice at NYU school of law.

LeCun, Bengio, Hinton. 2015. *Deep learning*. Nature 521(7553):436-44.

Liao, Jiang, Liu. 2020. *Probabilistically Masked Language Model Capable of Autoregressive Generation in Arbitrary Word Order*. Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics.

Lutz. 2009. *Learning Python*. O'Reilly.

Malinka, Perešíni, Firc, Hujňák, Januš. 2023. *On the Educational Impact of ChatGPT: Is Artificial Intelligence Ready to Obtain a University Degree?*. arXiv preprint arXiv:2303.11146.

Manning, Schutze. 1999. *Foundations of Statistical Natural Language Processing*. The MIT Press.

Nadkarni, Ohno-Machado, Chapman. 2011. *Natural language processing: an introduction*. Journal of the American Medical Informatics Association, Volume 18, Issue 5, September 2011.

OpenAI. 2022. *Introducing ChatGPT*. OpenAI. https://openai.com/blog/chatgpt.

OpenAI. 2023a. *ChatGPT — Release Notes*. OpenAI. https://help.openai.com/en/articles/6825453-chatgpt-release-notes.

OpenAI. 2023b. *GPT-4*. OpenAI. https://openai.com/research/gpt-4.

OpenAI. 2023c. *ChatGPT Plus*. OpenAI. https://openai.com/blog/chatgpt-plus.

Perez, Granger, Hunter. 2011. *Python: An Ecosystem for Scientific Computing*. Computing in Science & Engineering (Volume: 13, Issue: 2, March-April 2011).

Phelan. 2023. *How ChatGPT Suddenly Became Google's Code Red, Prompting Return Of Page And Brin*. Forbes. https://www.forbes.com/sites/davidphelan/2023/01/23/how-chatgpt-suddenly-became-googles-code-red-prompting-return-of-page-and-brin/

Radford, et al. 2019. *Language Models are Unsupervised Multitask Learners*. Technical report, OpenAI.

Roose. 2022. *The brilliance and weirdness of ChatGPT*. The New York Times. https://www.nytimes.com/2022/12/05/technology/chatgpt-ai-twitter.html

Rudolph, Tan, Tan. 2023. *ChatGPT: Bullshit spewer or the end of traditional assessments in higher education?*. Journal of Applied Learning & Teaching, Vol 6, No 1 (2023).

Russell, Norvig. 2010. *Artificial Intelligence: A Modern Approach*. Third Edition. Pearson.

Schatz. 1997. *Information Retrieval in Digital Libraries: Bringing Search to the Net*. Science, vol 275, issue 5298.

Shanahan. 2022. Talking About Large Language Models. arXiv preprint arXiv:2212.03551.

Skinner, Elizabeth. 2012. *Building the Second Mind: 1956 and the Origins of Artificial Intelligence Computing*.

Sobania, Briesch, Rothlauf. 2021. *Choose Your Programming Copilot*. GECCO '22: Proceedings of the Genetic and Evolutionary Computation Conference, July 2022.

Stackoverflow. 2022. *Developer survey*. Stackoverflow. https://survey.stackoverflow.co/2022/

Stancin, Jovic. 2019. *An overview and comparison of free Python libraries for data mining and big data analysis*. 2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO).

van Rossum, Drake. 2011. *The Python Language Reference Manual*. Network Theory LTD.

Vaswani, et al. 2017. *Attention Is All You Need*. Advances in Neural Information Processing Systems 30 (NIPS 2017).

Wohlin, et al. 2012. *Experimentation in Software Engineering*. Springer.

Wong, et al. 2023. *Natural Language Generation and Understanding of Big Code for AI-Assisted Programming: A Review*. Statistical Machine Learning with High-Dimensional Data and Image Analysis, Entropy 2023, 25(6), 888.

Ziegler, et al. 2020. Fine-Tuning Language Models from Human Preferences. arXiv preprint arXiv:1909.08593.

Zhang, et al. 2023. Practices and Challenges of Using GitHub Copilot: An Empirical Study. arXiv preprint arXiv:2303.08733.

8 Appendixes

8.1 Appendix A

Results of the experiment in CSV format:

Type,GPT-3.5 Legacy,GPT-3.5 Default,GPT-4,gpt35legacyReason,gpt35defaultReason,gpt4reason

Assignment,1,1,1,,,
Assignment,1,1,1,,,
Assignment,1,1,1,,,
Assignment,1,1,1,,,
Assignment,1,1,1,,,
Assignment,1,1,1,,,
Assignment,1,1,1,,,
Assignment,1,1,1,,,
Assignment,1,1,1,,,
Assignment,1,1,1,,,
Assignment,1,1,1,,,
Assignment,0,2,1,other,other,
Assignment,2,0,1,space,space,
Assignment,0,0,1,other,other,
Assignment,0,0,1,other,other,
Assignment,1,1,1,,,
Assignment,1,1,1,,,
Assignment,1,1,1,,,
Assignment,1,1,1,,,
Assignment,0,0,0,slice,slice,slice
Assignment,1,1,1,,,
Assignment,1,1,1,,,
Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,1,1,2,,,br

Assignment,0,0,0,slice,slice,slice

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,1,1,0,,,inputPrint

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,1,1,0,,,br

Assignment,1,1,1,,,

Assignment,1,2,1,,other,

Assignment,0,0,1,space,space,

Assignment,0,0,0,inputPrint,inputPrint,inputPrint

Assignment,0,0,0,inputPrint,inputPrint,inputPrint

Assignment,1,1,1,,,

Assignment,0,1,0,other,,other

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,1,2,2,,br,br

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,1,2,1,,br,

Assignment,1,1,2,,,other

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,0,1,0,other,,other

Assignment,1,1,1,,,

Assignment,0,0,2,other,other,space

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,0,0,0,slice,slice,slice

Assignment,2,1,1,slice,,

Assignment,1,1,1,,,

Assignment,0,0,1,space,space,

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,2,1,1,other,,

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Assignment,0,1,1,other,,

Assignment,1,1,1,,,

Assignment,1,1,1,,,

Exam 1,0,0,0,space,space,other

Exam 1,1,1,1,,,

Exam 1,0,0,0,other,other,other

Exam 1,1,1,2,,,other

Exam 1,1,1,1,,,

Exam 1,1,1,1,,,

Exam 1,1,1,1,,,

Exam 1,1,1,1,,,

Exam 2,1,1,1,,,

Exam 2,1,0,1,,other,

Exam 2,1,0,1,,other,

Exam 2,1,1,1,,,

Exam 2,1,1,1,,,

Exam 2,1,1,1,,,

Exam 2,1,1,1,,,

Exam 2,1,1,1,,,

Exam 2,1,1,1,,,

Exam 3,1,1,1,,,

Exam 3,1,1,1,,,

Exam 3,1,1,1,,,

Exam 3,1,1,1,,,

Exam 3,1,2,1,,br,

Exam 3,1,2,1,,other,

Exam 3,0,0,0,other,other,other

Exam 3,0,0,1,space,space,

Exam 3,0,0,1,other,other,

Exam 3,1,1,1,,,

8.2 Appendix B

Email correspondence with Sara Kfouri Koskinen, instructor of the "Programming Basics" course utilized for the experiment in this study:

Email from Aleksi Postari to Sara Kfouri Koskinen

Hi Sara!

Below are some questions about my thesis related to your Programming Basics course. Could you answer these? Thank you!

Question 1

I would like to use your course to experiment with checking how well ChatGPT completes assignments for the course. Is this OK with you?

Question 2

When was the course implemented? Meaning, when did it start, and when did it end?

Question 3

Who were the students who participated in the course (from which program they participated in the class)?

Question 4

How many students completed the course, and what was the average points & grade students got from the course?

Question 5

Exam #1 seemed more complicated than exams #2 and #3. Was this also the case for the students?

Reply from Sara Kfouri Koskinen

Question 1

I would like to use your course to experiment with checking how well ChatGPT completes assignments for the course. Is this OK with you?

> A: Yes, I agree.

Question 2
When was the course implemented? Meaning, when did it start, and when did it end?

> It started : 01.08.2022. It ended : 31.12.2022

Question 3
Who were the students who participated in the course (from which program they participated in the class)?

> First-year students of Artificial Intelligence program from 2022

Question 4
How many students completed the course, and what was the average points & grade students got from the course?

> 32 students and the average points was 80%. Grades average 4.

Question 5
Exam #1 seemed more complicated than exams #2 and #3. Was this also the case for the students?

> Yes, it was. I changed the exams level because the first was too hard for the students.

Question 6
What were the grading criteria for the students? What percentage they needed to get 1, and what for 2...

> The evaluate criteria was 1 >= 50% , 2 >= 60 % , 3 >= 70% , 4 >= 80% , 5 >= 90%

Question 7

And what weights were used? So how much weight did the automated assignments have, and how much weight did the exam have?

> The exam had a weight of 60% and the exercises 40%.

8.3 Appendix C

Template employed to carry out the experiment for each programming task:

Assignment Sent to ChatGPT

Write answer using Python code.

For example (
 denotes a line break)

```csv
```

Textarea where answer will be written:

```python

```

GPT-3.5 (Legacy)

gpt35legacy::

Answer

GPT-3.5 (Default)

gpt35default::

Answer

GPT-4

gpt4::

Answer