INVESTIGATION OF LARGE LANGUAGE MODELS FOR TEACHING SOFTWARE PROGRAMMING TO NON-PROGRAMMERS

Javier Morales

A Research Paper Submitted for
Direct Individual Study

Summer I and II

Department of Computer Science

University of North Carolina Wilmington

July 17, 2023

Abstract

Investigation of Large Language Models for Teaching Software Programming to Non-Programmers. Morales, Javier, 2024. Direct Individual Study Research Paper, University of North Carolina Wilmington.

Large Language Models (LLMs) have made significant news since the announcement of ChatGPT's release in November 2021. ChatGPT has sparked interest in the adoption and deployment of AI solutions. One particular area is in reimagining software development where an AI would support training in software development to hands-free development of software based on engineered prompts. The research will investigate the potential of LLM, in particular ChatGPT, for training a non-programmer to program. More precisely this investigation will demonstrate the effectiveness and impact of Large Language Models (LLM) being used as tools for teaching Software Programming to Non-Programmers. This work does touch upon teaching methodologies with a comparison between the traditional methods of teaching first programming with an LLM (ChatGPT) approach to programming.

The goals of this research project include:

- Investigation of a large language model for teaching programming to non-programmers.
- Curation of training and tutorials for teaching software programming to non-programmers.
- Review of LLMs as a instructional aid in teaching programming to non-programmers.

To facilitate the goals outlined above I will survey the literature to identify resources using LLMs identify, review, and recommend videos, projects and conversational prompts to learn software programming with little to no knowledge.

Table of Contents

Page

Chapter 1: Introduction

1.1 Large Language Models

It may seem that the word ChatGPT and AI are heard more and more frequently and very interchangeably but the reality is that AI or artificial intelligence and ChatGPT are different. Artificial intelligence is “a set of technologies that enable computers to perform a variety of advanced functions, including the ability to see, understand and translate spoken and written language, analyze data, make recommendations, and more.” as described by Google [1] while ChatGPT is a conversational language model that uses the technologies of AI to develop a tool/product.

Further, AI is an area of research and development that encompasses several distinct fields of study from machine learning (ML) to expert systems to reinforcement learning to artificial general intelligence (AGI). A formal definition of AI is " the intelligence of machines or software, as opposed to the intelligence of human beings or animals” (Wikipedia [2]). The field of AI is broad, so it is important to understand the spectrum.

Weak AI, also known as Narrow AI, is what defines the most common AI deployed today. Weak AI is defined by Techopedia [3] as “a form of AI specifically designed to be focused on a narrow task and to seem very intelligent at it”. Examples of this AI include smart assistants like Siri, Alexa, and Cortana, face recognition, lane keeping found in automobiles, and large language modeled chatbots. These models tend to have limited memory which leads to limited capabilities and specific tasks.

Don't let the prefix “weak” fool you as these AI are still very efficient as we see them being integrated in today's society. An example of this can be seen in ChatGPT– lets ask ChatGPT how to code a simple game and see how it handles a task.

Figure 2. OpenAI’s ChatGPT is given the assignment to write a simple game of its choice in Python. (OpenAI, 2023)

In this example, we use OpenAI’s ChatGPT to demonstrate the efficacy of “Weak AI” in large language models. Large language models or LLMs are “are deep learning algorithms that

can recognize, summarize, translate, predict, and generate content using very large datasets." as described by Nvidia[4].

The attempt is to demonstrate if LLM can develop a simple game of its choice written in Python code. First, we need to help it understand the approach, then we follow the actual request. While we now understand weak AI is able to write simple code in a matter of seconds, it is revolutionizing but it is important to realize the end goal is still the same – Strong AI.

Strong AI is more capable to not mimic but behave like a normal human can. Behavioral examples may include for the AI to be able to generalize concepts on its own and apply them to future environments to further build its knowledge, emotional and logical sense of its own. It is important to remember that we have only reached "Weak AI" and thus the only concept we can grasp to understand what "Strong AI" is from fictional films and comics. Some of these "Strong AI" you may recognize from films in example Tony Stark's J.A.R.V.I.S. from Marvel's Iron Man, Plankton's Wife from SpongeBob or Skynet from The Terminator franchise.

Now if we infer how chatbots are able to write simple code via a user's request, the possibilities with strong AI would be astronomical. It is important to remember this concept is still theoretical but the application of them could be truly revolutionizing. In today's increasingly technologically advancing world, the ability to code has become increasingly popular and with common tools in weak AI, the ability to learn how to code has never been more diverse.

Chapter 2: Analysis & Review of Large Language Models

2.1 Software Programming

Software programming can be seen anywhere and everywhere, its an integral part of society. Software programming refers to the discipline involving computer science, engineering and coding to develop a software in other words “a profession within the computer technology field that primarily deals with writing code.” (Learn.org[5])

As the demand for software continues to grow, the need for software programmers will also grow. Software programming is an essential skill that can be used to an advantage as it can open a broad field of career opportunities.

In the early days, software was programmed using simple rules and algorithms to create software. However, as software became more complex, it became clear that a more sophisticated approach was needed. This led to the development of artificial intelligence (AI) techniques, such as machine learning and natural language processing.

AI techniques be used in automation to handle software programming tasks that were thought out as difficult before in a matter of time. These techniques are also able to create new software like LLMs that would not be possible with traditional programming methods. With time given, the potential of software programming could be truly revolutionizing, this is why it is important for the next generation to continue to keep learning software programming.

2.2 Conversational Chatbots

The rise of popularity amongst LLMs has led to many conversational “chatbots”,

including Google's BARD, OpenAI's ChatGPT or Bing's Chat but have you ever stopped yourself to ask how these LLMs know how to create a Python program that can sort data with the best efficiency? How are these LLMs able to act as tutors, teachers and even more at the same time? In this portion of the research, the development of large language models for conversational chatbots will be explored and the capabilities will test the efficacy of these language models.

Large language models (LLMs) can be traced back to the 1950s, when Alan Turing proposed the Turing Test [6], a benchmark for determining whether a computer can exhibit intelligent behavior equivalent to, or indistinguishable from, that of a human.

Turing's proposal was inspired by the imitation game, a parlor game in which a human interrogator attempts to distinguish between a human and a machine by engaging in natural language conversation with both.

In the Turing Test, a human interrogator engages in natural language conversation with two other parties: a human and a machine. The interrogator's goal is to determine which party is the machine and which is the human. If the interrogator cannot reliably distinguish between the two parties, then the machine is said to have passed the Turing Test.

Following the proposal of Alan Turing, Joseph Weizenbaum (1964), MIT Computer Scientist. used this newfound knowledge and created a computer program for the study of natural language communication between man and computer – the first attempt at a "chatbot". ELIZA [7] was the product of this – a program that simulated the presence of psychotherapy sessions. Using advanced source code and early deep machine-learning techniques, ELIZA was able to analyze user prompts to provide a written response mimicking with best efforts as a psychotherapist would.

You might think this is insane, how is a machine able to understand the user's mental or physical problems by simply typing and sending prompts? Think of it like this – imagine there is a culinary student that is going to the library (let this act as the computer) and the student needs a book that shows him how to cook oatmeal cookies (now this is going to act as the user's personal issues). The student then proceeds to the librarian (ELIZA) to request oatmeal cookie recipes (user's personal request) and the librarian (ELIZA) will make its best efforts to find relevant information to what the chef (user) had requested.

At first, it might seem out of this world that a machine could understand a user's mental or physical problems by simply typing and sending prompts. However, it's not so different from how a customer service representative might help a customer find a product.

Imagine a customer who calls a company to ask about a product they are interested in. The customer service representative asks the customer some questions to further understand their needs, and then the representative does their best to find relevant information.

Here is where things get more complex for the LLM. The LLM processes the request from the user to further understand what the request is. Language modeling aims to generate text based on preceding words. Removing short texts from the dataset helps reduce noise and enables better modeling of text dependencies. Additionally, when working with learning-based models, preprocessing and dividing large documents into non-overlapping spans can be beneficial for efficient handling and processing.

Here are the most important stages as the LLM process a user's request [8].

Tokenizing Datasets:

The LL first needs to understand the user's request as it does not simply comprehend statements in full therefore the LLM breaks down the user's request by

individual words or phrases called tokens. Tokens are the building blocks of input for the model. They can be single characters, words, parts of words, or other segments of text or code.

Encoding and Decontamination:

Once the user's text request has been tokenized into a format that the LLM can understand, the tokens must be encoded and decontaminated. The encoding process typically involves assigning each token a unique identifier. This is done by using a tokenization scheme, which is a set of rules for converting tokens into unique identifiers.

After encoding the tokens, the LLMs move forward through the decontamination process. This involves removing any tokens that are not relevant to the task at hand. This can include stop words, punctuation, and other tokens. These stop words are not very informative, so they are often removed from the text before it is processed by the LLM. Punctuation is also often removed from the text, as it is not necessary for the LLM to understand the meaning of the text [9].

Processing:

After tokenization and encoding, the LLM performs a semantic parsing step to understand the meaning of the question. It then expands the query with synonyms and related words to find more relevant information.

The LLM then retrieves the answer from the knowledge base or corpus by matching the query to concepts and facts. The answer is then ranked based on its relevance to the question.

Finally, the LLM generates a response based on the ranked answer in the form for the user to best understand. The LLM provides the response.

Large language models are still under development, but they have the potential to revolutionize the way we interact with computers. They could be used to create more creative content, provide personalized assistance and more popular conversational chatbots. They could be used to create personalized learning plans, to provide feedback on student work, and to even help students tackle often complex problems. The future of LLMs is bright, and we can only imagine what they will be able to do in the years to come.

2.3 Large Language Models for Software Programming

LLM's have been deployed as conversational technologies that can author quite compelling dialogue with and for humans. These models are able to create quite complex writings based on materials acquired from the internet. These tools have been used to write compelling articles for college entry, marketing materials for global giants, scientific manuscripts, books of fiction and non-fiction, poems, songs, and more.

It should not be a stretch to imagine that these tools can also be used in the development of software programs. In this example, I have asked ChatGPT to write a program to add a set of numbers and return their sum. Included below in the figures are the results of this prompt for multiple programming languages to include Python, C/C++, Java, and LC3 Assembly.

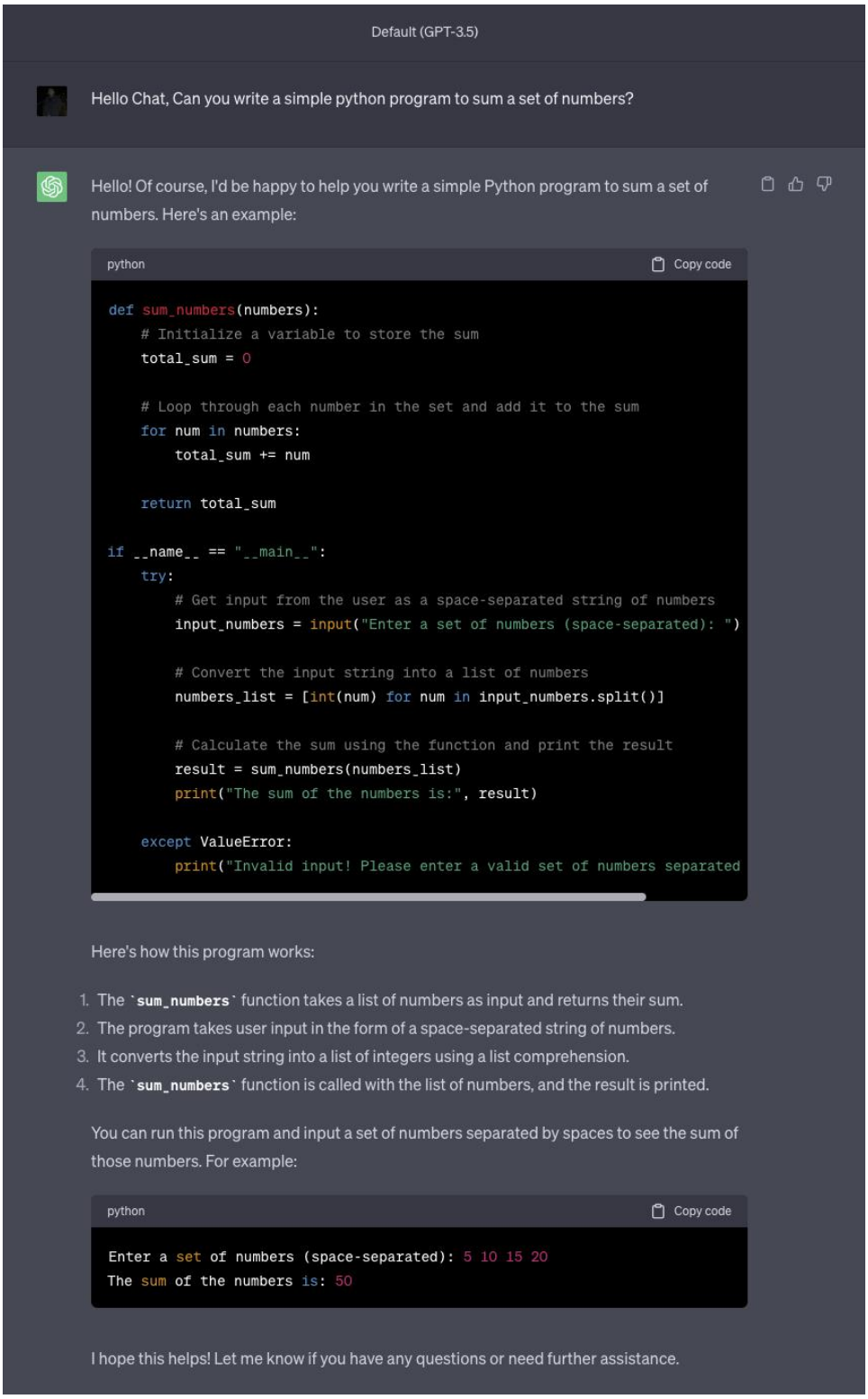


Figure 1. Simple Python Program to sum a set of numbers (OpenAI, 2023)

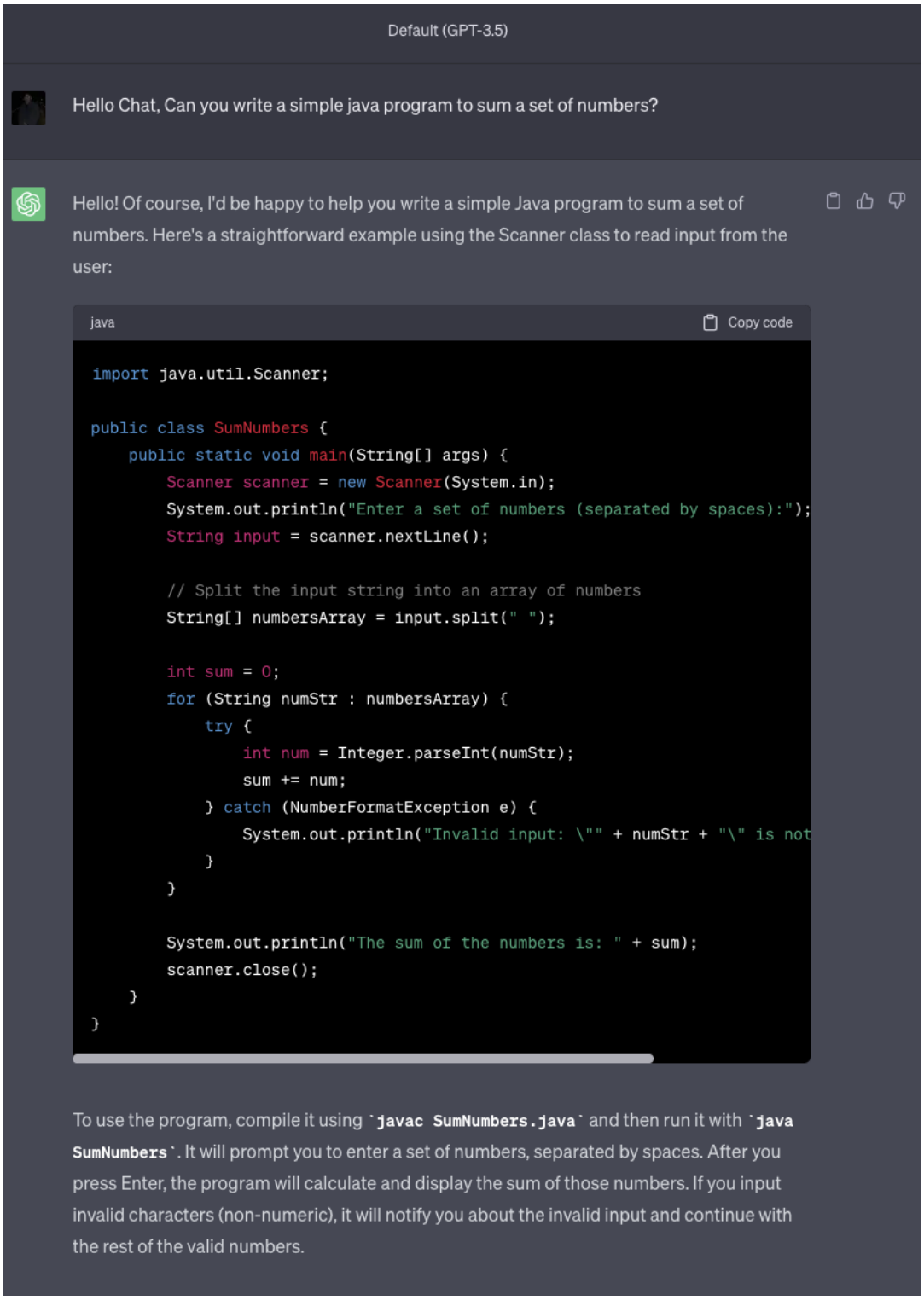


Figure 2. Simple Java Program to sum a set of numbers (OpenAI, 2023)

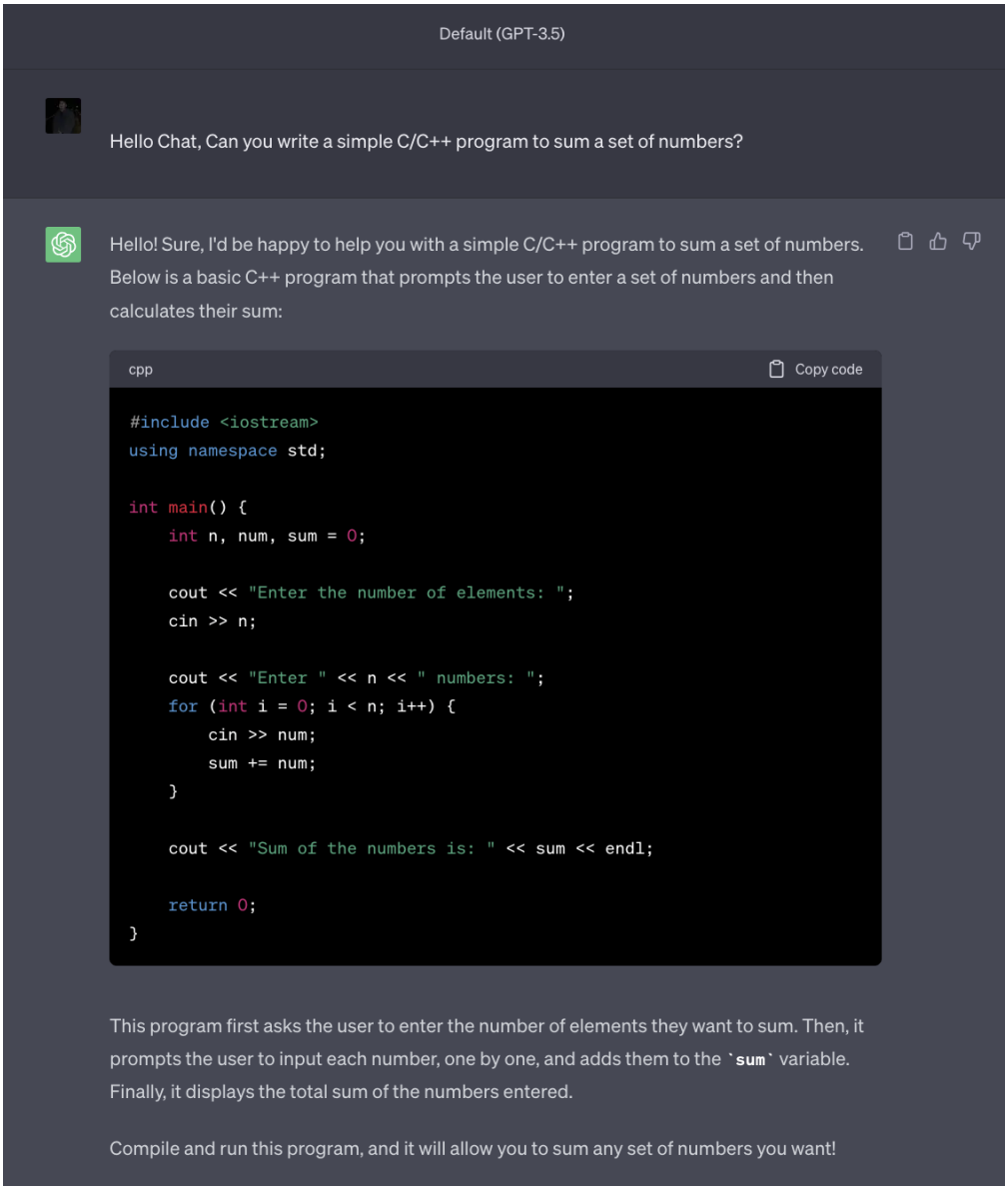


Figure 3. Simple C/C++ program to sum a set of numbers (OpenAI, 2023)

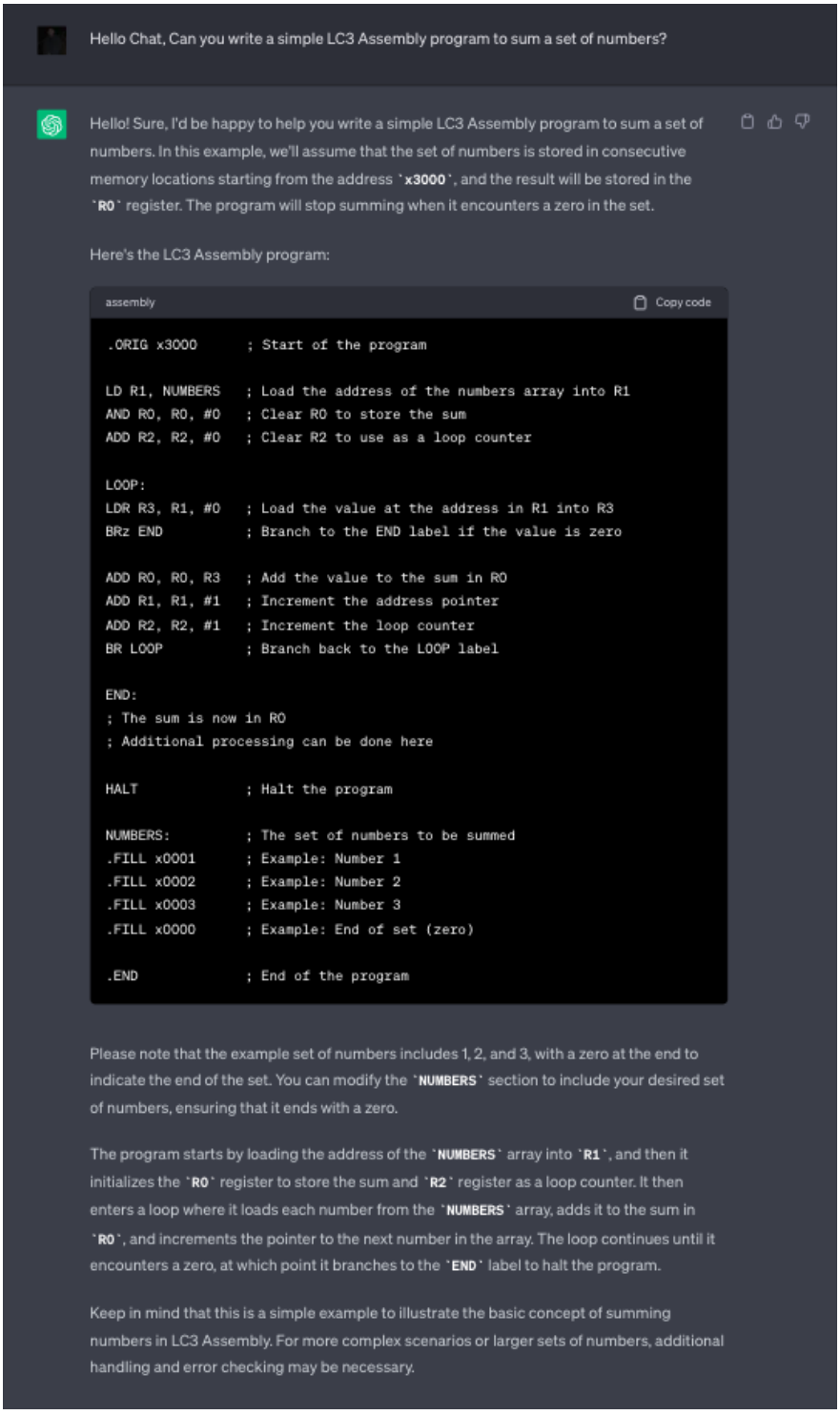


Figure 4. Simple LC3 Assembly program to sum a set of numbers (OpenAI, 2023)

It is quite compelling how LLMs can be used to create code that at some point in any

programmer's career would have taken more than 2 minutes as compared to an LLM. With this knowledge it is quite impossible to imagine the limitations for an LLM or how large the projects can reach with the power of this tool.

Chapter 3: Review of Learning Software Engineering for Non-programmers

3.1 Education Curriculums

The current state of education is facing a number of challenges. Educators need to be aware of this and start planning for how they will incorporate LLMs into their teaching. The student population is becoming increasingly diverse, and the rise of technology in the classroom is changing the way that students learn. Additionally, the growing emphasis on standardized testing is putting pressure on teachers to perform.

One of the biggest challenges that educators face is dealing with large class sizes. This can make it difficult to provide personalized attention for each student. Additionally, keeping up with the latest educational technology can be a challenge, as new technologies are constantly emerging.

Another challenge that educators face is preparing students for the 21st century job market. This requires teaching students' skills in modern critical thinking, problem solving, and collaboration. Additionally, educators need to find ways to engage students in their learning and keep them motivated.

LLMs have the potential to help address some of these challenges. LLMs can be used to personalize learning for each student, provide feedback on student work, and create engaging and interactive learning experiences. For example, LLMs can be used to create personalized learning plans for each student, based on their individual needs and interests.

LLMs can also be used to provide feedback on student work [10], pointing out areas where students could improve. This feedback could be tailored to each student's individual needs, and it could be provided in a timely manner. Additionally, LLMs can be used to create engaging and interactive learning experiences for students. This has helped to keep students motivated and engaged in their learning.

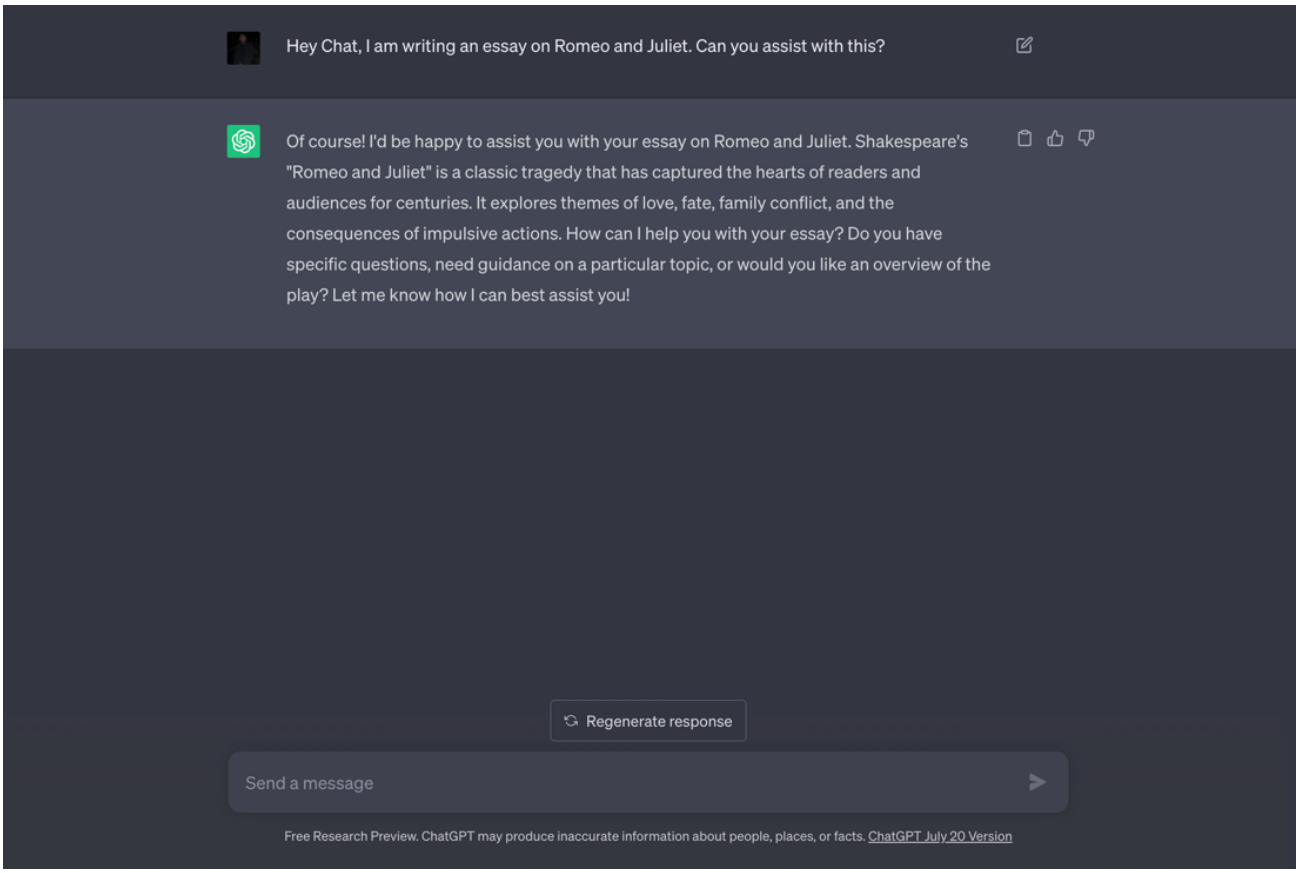


Figure 1. Example of ChatGPT chatbot being requested for brainstorming. (OpenAI, 2023)

Overall, LLMs have the potential to be a powerful tool for educators. They can help to personalize learning, provide feedback, and create engaging learning experiences. LLMs have the potential to revolutionize education, and as they continue to develop, it is likely that they will play an increasingly important role in the classroom.

3.2 Integration of Large Language Models

In the field of software programming, we have been adhering to the same curriculum for many years without any significant changes. However, it is crucial that we start advancing our approach to keep up with the rapidly evolving technology, particularly the emergence of large language models. The world is quickly moving forward with LLMs, and the standard curriculums need to as well.

LLMs offer many benefits in regard to education. One significant aspect of LLMs is their capability to create highly personalized learning experiences for students. By comprehending the individual strengths and weaknesses of each student, LLMs can tailor content and activities to cater to their specific needs. Consequently, this personalized approach enables students to learn more effectively and efficiently.

Furthermore, LLMs possess the ability to adapt and evolve over time, continuously refining their algorithms and enhancing their capacity to deliver precise and targeted educational content. As technology advances and more data becomes available, LLMs have the potential to revolutionize the global education system.

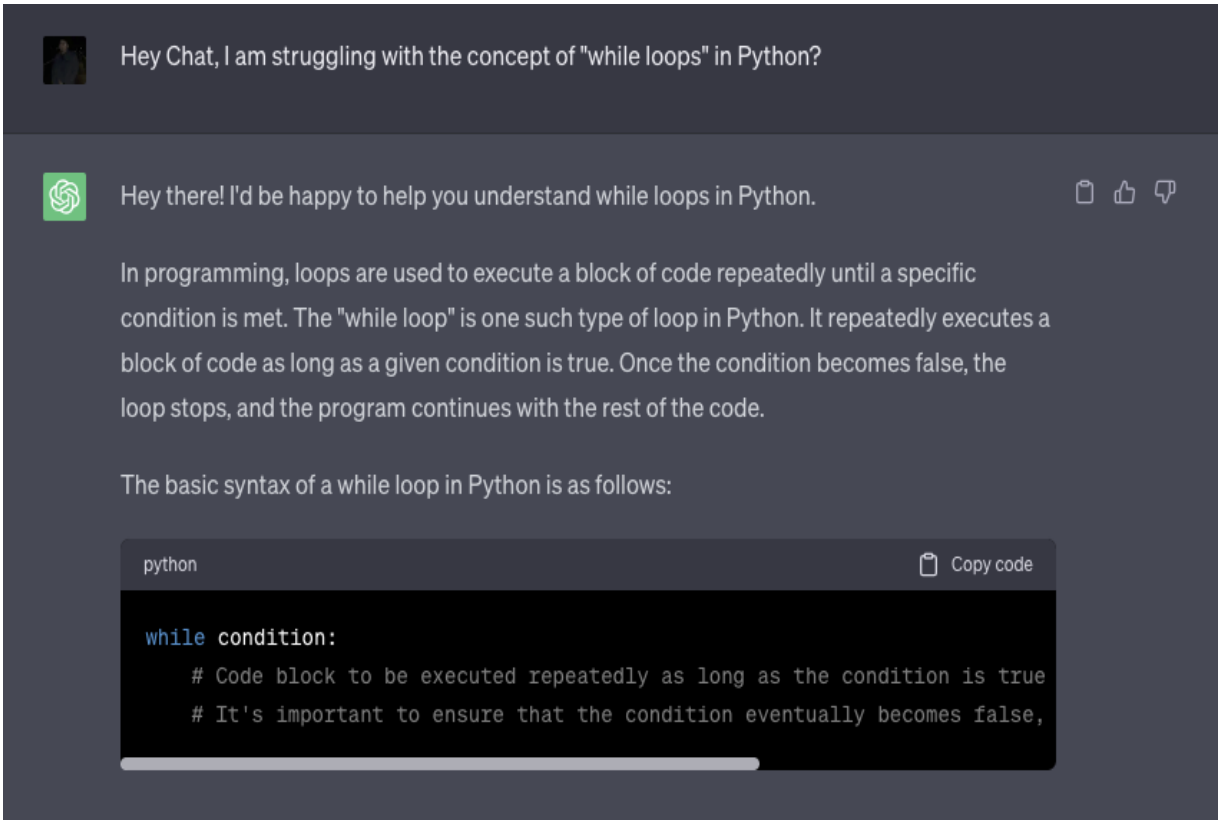


Figure 1. Example of ChatGPT being used as a tool to enhance education. (OpenAI, 2023)

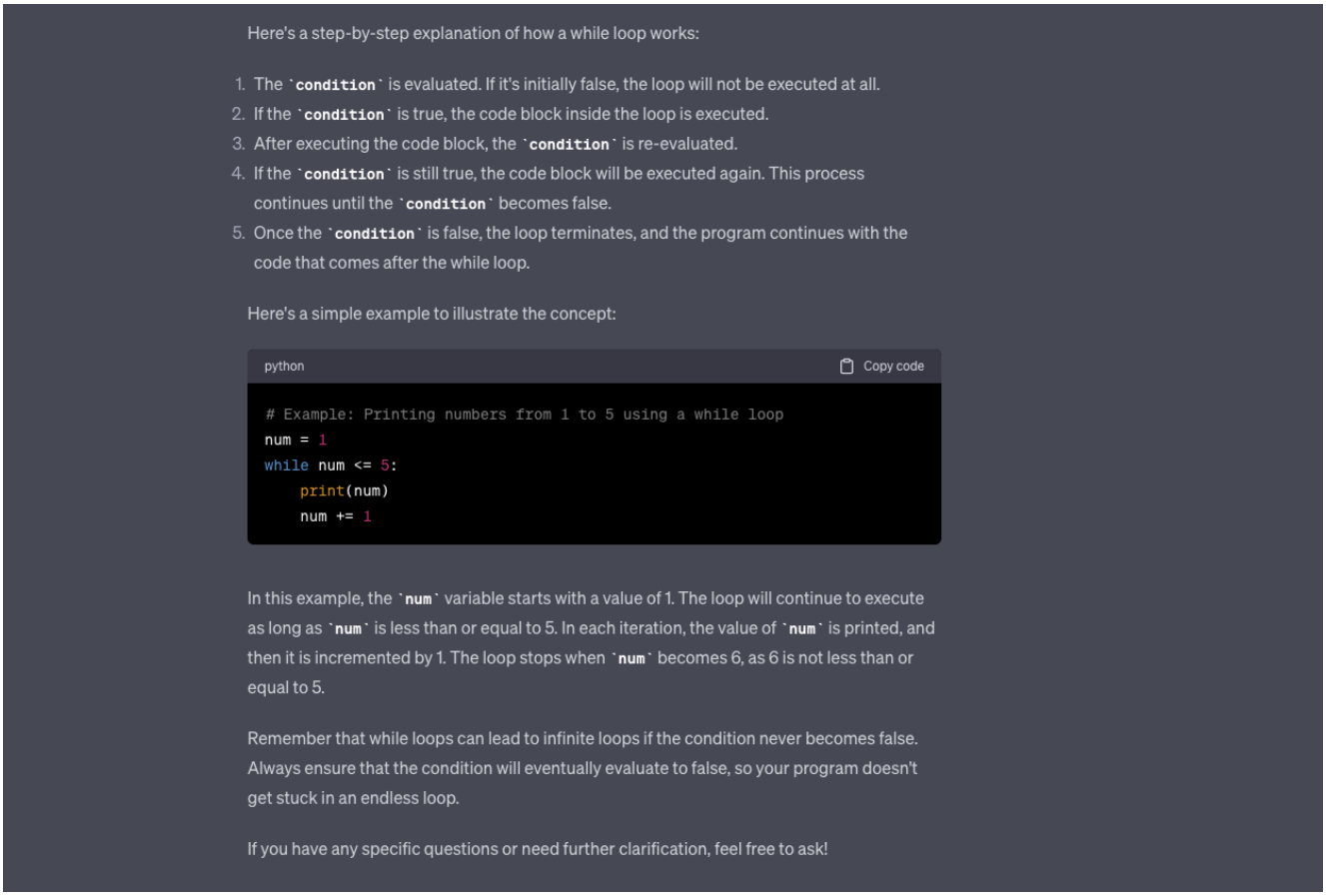


Figure 2. ChatGPT provides a continued detailed response to the user's request. (OpenAI, 2023)

To some change is scary – the question is how we can implement complex tools like large language models in today's classrooms. The public's first reaction following the release of this powerful tool, ChatGPT was a mix of excitement and panic.

In early January, the New York City Department of Education's representative, Jenna Lyle, released a statement to the Washington Post, "While the tool may be able to provide quick and easy answers to questions, it does not build critical--thinking and problem-solving skills, which are essential for academic and lifelong success," [11]. This initial panic and uncertainty not only spread to the education sector but nationally as well.

The reality is we have already done it before but with calculators. Calculators have been

used as a tool to solve even more complex problems, for example, the average student would never be able to solve integral problems as fast or ease as we do today.

Professor Rob Reich from Stanford University proposed the same statement and noted that it enables students "to quickly solve complex computations, graph equations and solve problems" [12]. He also does add that even though this tool enables students to quickly skip towards the solution to problems, it is still necessary for students to learn about "how the world works from first principles" [12] thus shifting the long hard thinking process to the LLM and making students to engage even deeper in critical thinking and creativity.

It is important to realize that LLMs should be used to supplement students' learning, not to replace it. Teachers play an essential role in providing guidance, support, and feedback to students but with the help of LLMs, students can learn more effectively and efficiently.

3.3 Methods of Learning

LLMs are still in their early stages of being used to help non-programmers learn software programming. However, there is a growing body of research that suggests that LLMs have the potential to be a valuable tool for helping non-programmers learn software programming effectively.

With careful planning and implementation, LLMs can help to make software programming education more accessible and inclusive for non-programmers. In this section of this research paper, I will discuss the curation of training and tutorials for teaching software programming to non-programmers.

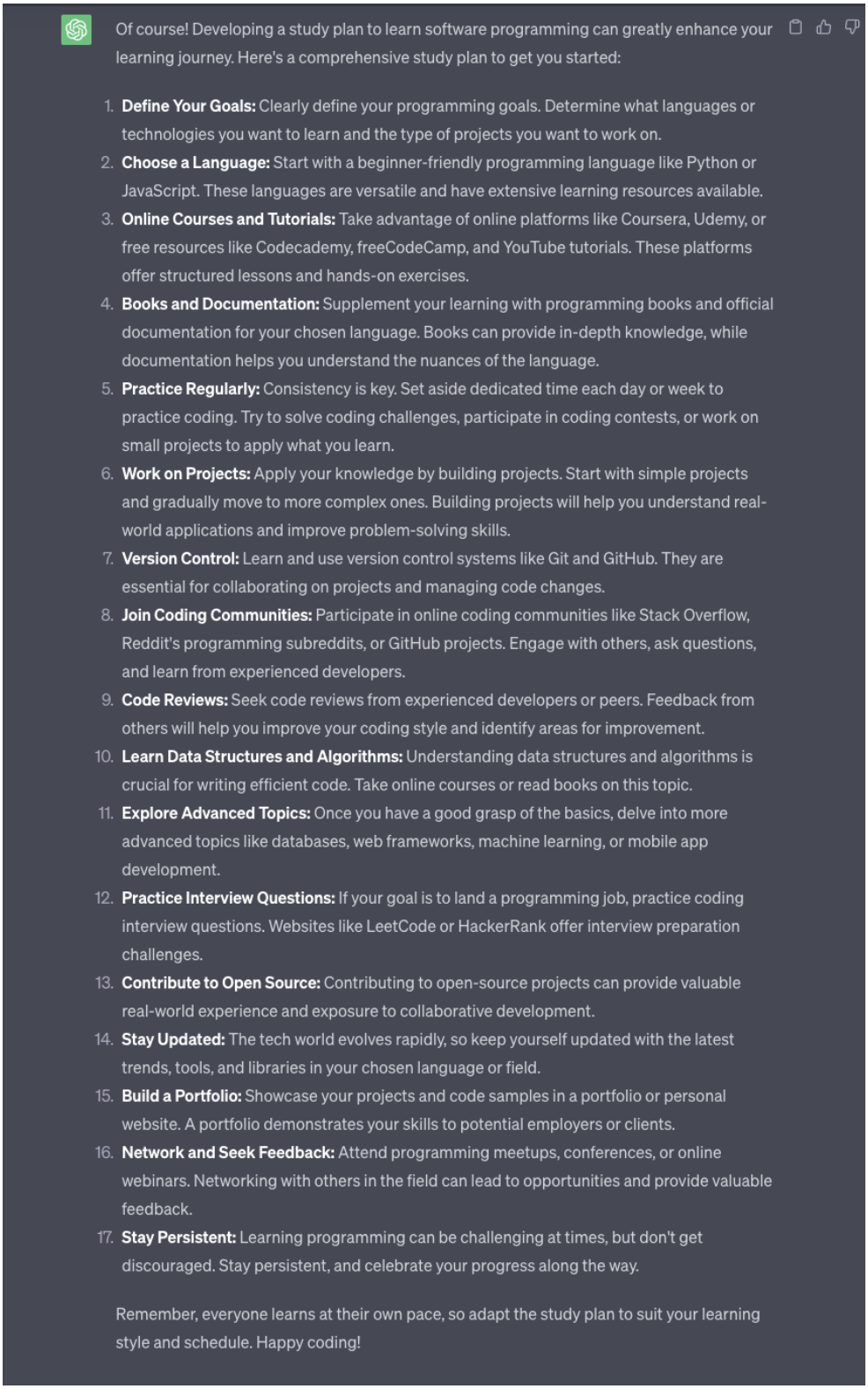
3.3.1 Development of Study Plan

It is important to keep in mind that LLMs are tools not teachers, so therefore to complete this research, I had to approach ChatGPT in a specific way to design a study plan that a

nonprogrammer could follow and accomplish the goal of learning software programming [12-14].

As I was researching the first step, I did was to ask the simple and straightforward question of “Hey Chat, can you develop a study plan to learn software programming?”. The response was quite decent as it introduced a general overview of learning software programming and how to achieve it but was still very vague.

As you can tell from the figure below ChatGPT lacks specific resources, tutorials or timelines in regard to when I can complete this goal. ChatGPT does provide some feedback on things we can improve on within the request. For example, the first action ChatGPT recommends is to define the goals. We should keep in mind that users need to be specific in their requests when prompting an LLM.



Of course! Developing a study plan to learn software programming can greatly enhance your learning journey. Here's a comprehensive study plan to get you started:

1. **Define Your Goals:** Clearly define your programming goals. Determine what languages or technologies you want to learn and the type of projects you want to work on.
2. **Choose a Language:** Start with a beginner-friendly programming language like Python or JavaScript. These languages are versatile and have extensive learning resources available.
3. **Online Courses and Tutorials:** Take advantage of online platforms like Coursera, Udemy, or free resources like Codecademy, freeCodeCamp, and YouTube tutorials. These platforms offer structured lessons and hands-on exercises.
4. **Books and Documentation:** Supplement your learning with programming books and official documentation for your chosen language. Books can provide in-depth knowledge, while documentation helps you understand the nuances of the language.
5. **Practice Regularly:** Consistency is key. Set aside dedicated time each day or week to practice coding. Try to solve coding challenges, participate in coding contests, or work on small projects to apply what you learn.
6. **Work on Projects:** Apply your knowledge by building projects. Start with simple projects and gradually move to more complex ones. Building projects will help you understand real-world applications and improve problem-solving skills.
7. **Version Control:** Learn and use version control systems like Git and GitHub. They are essential for collaborating on projects and managing code changes.
8. **Join Coding Communities:** Participate in online coding communities like Stack Overflow, Reddit's programming subreddits, or GitHub projects. Engage with others, ask questions, and learn from experienced developers.
9. **Code Reviews:** Seek code reviews from experienced developers or peers. Feedback from others will help you improve your coding style and identify areas for improvement.
10. **Learn Data Structures and Algorithms:** Understanding data structures and algorithms is crucial for writing efficient code. Take online courses or read books on this topic.
11. **Explore Advanced Topics:** Once you have a good grasp of the basics, delve into more advanced topics like databases, web frameworks, machine learning, or mobile app development.
12. **Practice Interview Questions:** If your goal is to land a programming job, practice coding interview questions. Websites like LeetCode or HackerRank offer interview preparation challenges.
13. **Contribute to Open Source:** Contributing to open-source projects can provide valuable real-world experience and exposure to collaborative development.
14. **Stay Updated:** The tech world evolves rapidly, so keep yourself updated with the latest trends, tools, and libraries in your chosen language or field.
15. **Build a Portfolio:** Showcase your projects and code samples in a portfolio or personal website. A portfolio demonstrates your skills to potential employers or clients.
16. **Network and Seek Feedback:** Attend programming meetups, conferences, or online webinars. Networking with others in the field can lead to opportunities and provide valuable feedback.
17. **Stay Persistent:** Learning programming can be challenging at times, but don't get discouraged. Stay persistent, and celebrate your progress along the way.

Remember, everyone learns at their own pace, so adapt the study plan to suit your learning style and schedule. Happy coding!

Figure 1. Example of ChatGPT being used develop a study plan to learn software programming. (OpenAI, 2023)

To further gain a more elaborate and detailed study plan I incorporated a couple of variables to define my goals. For this example, I want to learn Python for software programming in a timeline of about ten weeks. I also would like to add that I have no previous experience as we are attempting to learn from the perspective of a nonprogrammer in the request as well as the addition of more resources. In the figure below, the results are shown.

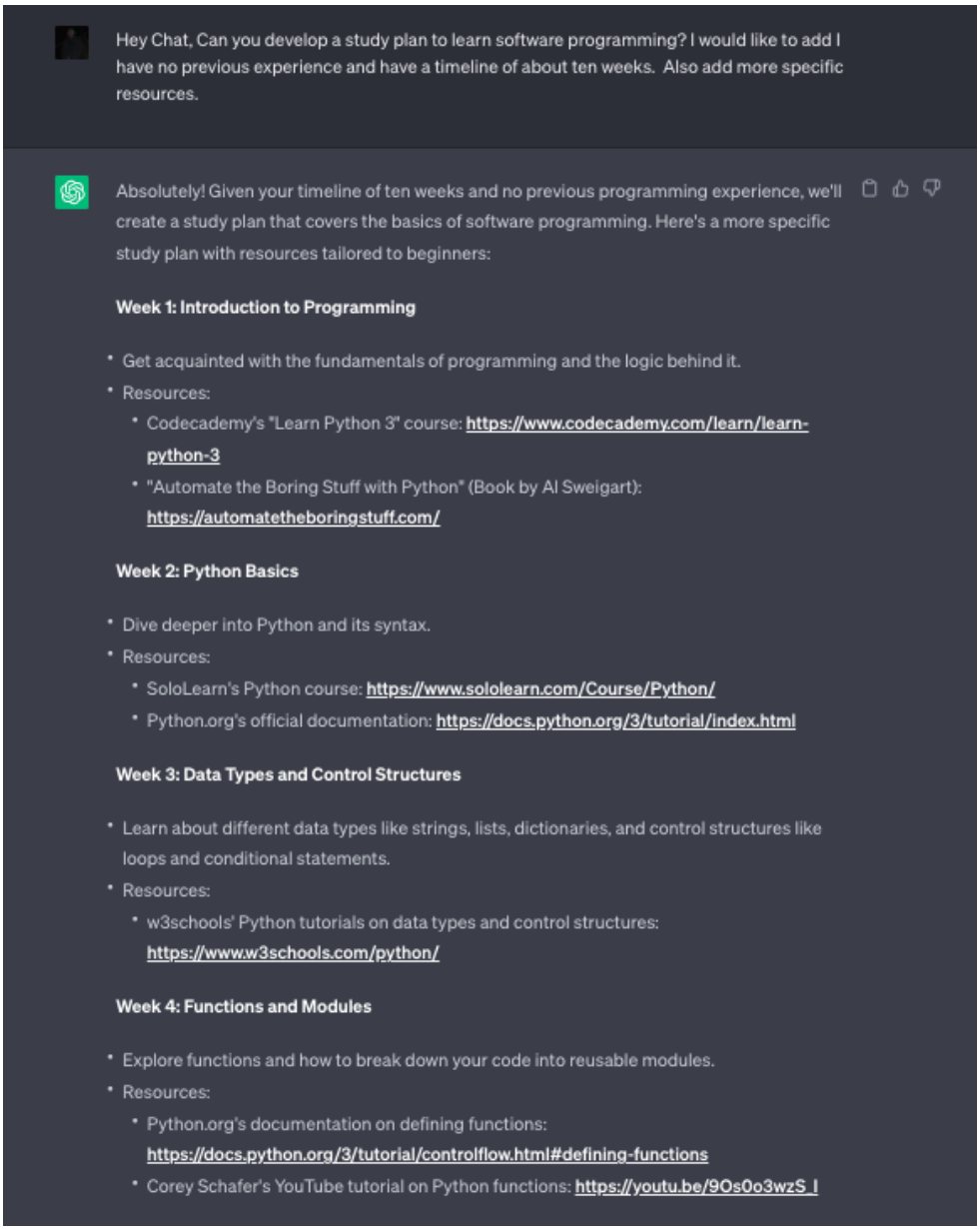


Figure 1. Example of ChatGPT being used develop a more detailed study plan to learn software programming. (OpenAI, 2023)

Week 5: Introduction to HTML, CSS, and JavaScript

- Get started with web development by learning the basics of HTML, CSS, and JavaScript.
- Resources:
  - freeCodeCamp's "Responsive Web Design Certification" course: https://www.freecodecamp.org/learn/responsive-web-design/
  - Mozilla Developer Network's (MDN) web development guide: https://developer.mozilla.org/en-US/docs/Learn

Week 6: Building a Simple Website

- Combine your HTML, CSS, and JavaScript skills to build a basic website.
- Resources:
  - Codecademy's "Build Your Own Webpage" project: https://www.codecademy.com/learn/paths/web-development

Week 7: Introduction to Git and Version Control

- Learn about version control systems like Git to manage your code.
- Resources:
  - Atlassian's Git tutorial: https://www.atlassian.com/git/tutorials/what-is-version-control

Week 8: Introduction to Object-Oriented Programming

- Understand the concepts of object-oriented programming (OOP).
- Resources:
  - Python.org's tutorial on classes and objects: https://docs.python.org/3/tutorial/classes.html

Week 9: Building a Simple Python Project

- Apply your Python knowledge to build a small project of your choice, such as a text-based game or a simple web scraper.
- Resources:
  - "100 Python Projects" (GitHub repository with project ideas): https://github.com/karan/Projects

Week 10: Review and Reflect

- Review all the concepts you've learned and work on any areas that need improvement.
- Reflect on your progress and decide on the next steps in your programming journey.

Remember, consistent practice and hands-on coding are essential for learning programming effectively. Don't hesitate to explore additional resources or ask for help from programming communities if you encounter difficulties. Best of luck with your ten-week programming journey!

Figure 2. Example of ChatGPT\continued from figure 1 (OpenAI, 2023)

The results shown are much more resourceful as it now provides links to different tutorials and guides on how-to's. This leads our research to believe that ChatGPT does take into processing that we added the user as a non-programmer. It is very important for the LLM to understand our level of background knowledge on this request as it could have mentioned topics way to advanced for a beginner such as the time efficacy of algorithms or complex data structures. Another positive about this result is that it fits perfectly in our ten-week timeline which ChatGPT hopes for the user to accomplish.

3.3.2 Study Plan Frameworks

While this is a remarkable achievement in our research, there is still progress to be made since the overview of the timeline can be scattered or a bit disoriented to users. In this next approach, our research attempts to adapt a study plan framework [13].

In this portion of the research, ChatGPT is asked to change the timeline completely and instead of utilizing tutorials for material in each week in the timeline, we have to test the user further complete understanding. The user achieves this through the implementation of projects replacing the tutorials.

This allows for the user have a more oriented path to having a curriculum and hands on learning. While having projects that the user needs to complete, the user understands if I can complete this with complete understanding, I can move on to the next project. In the figure below we see an understanding of how the study plan changes.

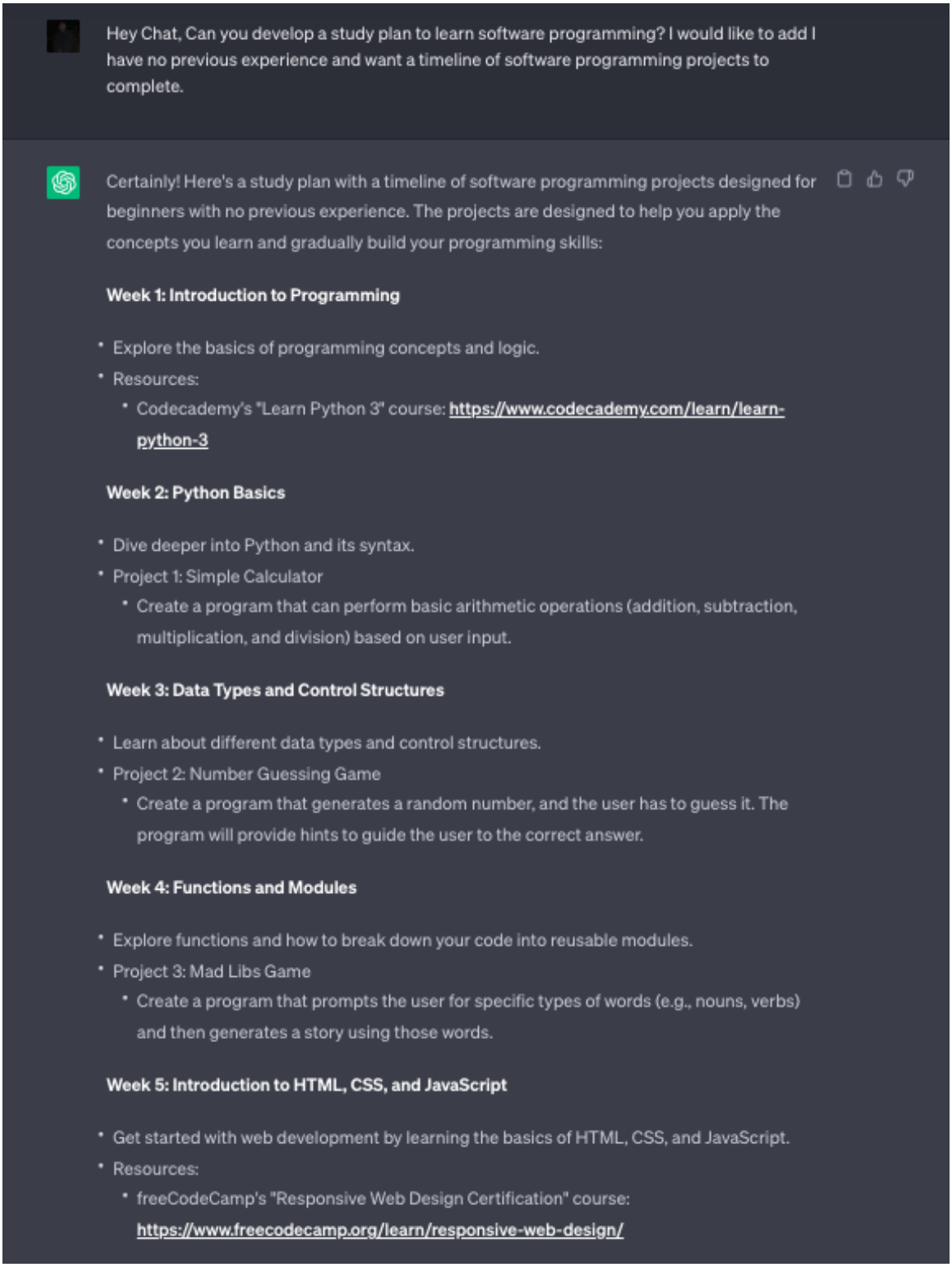


Figure 2. Example of ChatGPT applying a study framework plan based on project completion (OpenAI, 2023)

Week 6: Building a Simple Website

- Combine your HTML, CSS, and JavaScript skills to build a basic website.
- Project 4: Personal Portfolio Website
  - Create a simple portfolio website showcasing your projects and information about yourself.

Week 7: Introduction to Git and Version Control

- Learn about version control systems like Git to manage your code.
- Resources:
  - Atlassian's Git tutorial: https://www.atlassian.com/git/tutorials/what-is-version-control

Week 8: Introduction to Object-Oriented Programming

- Understand the concepts of object-oriented programming (OOP).
- Project 5: To-Do List Application
  - Build a to-do list application that allows users to add, edit, and delete tasks.

Week 9: Building a Simple Python Project

- Apply your Python knowledge to build a small project of your choice.
- Project 6: Text-Based Adventure Game
  - Create a text-based adventure game where users make choices that affect the outcome of the story.

Week 10: Review and Reflect

- Review all the concepts you've learned and work on any areas that need improvement.
- Reflect on your progress and decide on the next steps in your programming journey.

Throughout this study plan, don't forget to practice regularly, seek help from programming communities or forums if needed, and stay curious about new technologies and concepts in the field of software programming. Happy coding!

Figure 2. Continued example of ChatGPT applying a study framework plan based on project completion (OpenAI, 2023)

As we refine our study plan framework, we must be mindful of setting realistic

expectations, especially for nonprogrammers. Learning multiple topics of software programming in just ten weeks might be overly ambitious and impractical. Therefore, to ensure the success of this specific research, I propose a more focused approach.

Let's pinpoint clear and achievable goals while selecting a specific branch of software programming. In doing so, we can emphasize the vast potential of Python and its applications in software programming. By narrowing our focus and concentrating on Python's versatility, participants can gain a deeper understanding of this language and its relevance in the software development domain. This approach will allow them to build a solid foundation while preparing them for more advanced learning beyond the initial ten weeks.

For this next portion of study framework, I want to introduce the idea of the LLM to act as a “tutor”. This will allow for ChatGPT to be more understanding of its environment and what the vision is for the final product. Lets use a similar prompts as mentioned in the Youtube video by Tina Huang which is as follows - “Act as a coding tutor that creates study plans to help people learn to code. You will be provided with the goal of the student, their time commitment, and resource preferences. You will create a study plan with timelines and links to resources. Only include relevant resources because time is limited. My first request - "I want to become a data scientist but I do not know how to code. I can study 10 hours per week and only want video resources. I want to learn to code in python. Create a study plan for me”. We will use this prompt but slightly modify to be appropriate for this research.

In this next prompt, I would also like to includethe request for ChatGPT will be to develop a Python-oriented study plan framework in the field of software programming. The request will continue the project style timeline to be implemented although in this request since a more practical approach is needed. In the figure below ChatGPT demonstrates the final study

plan framework.

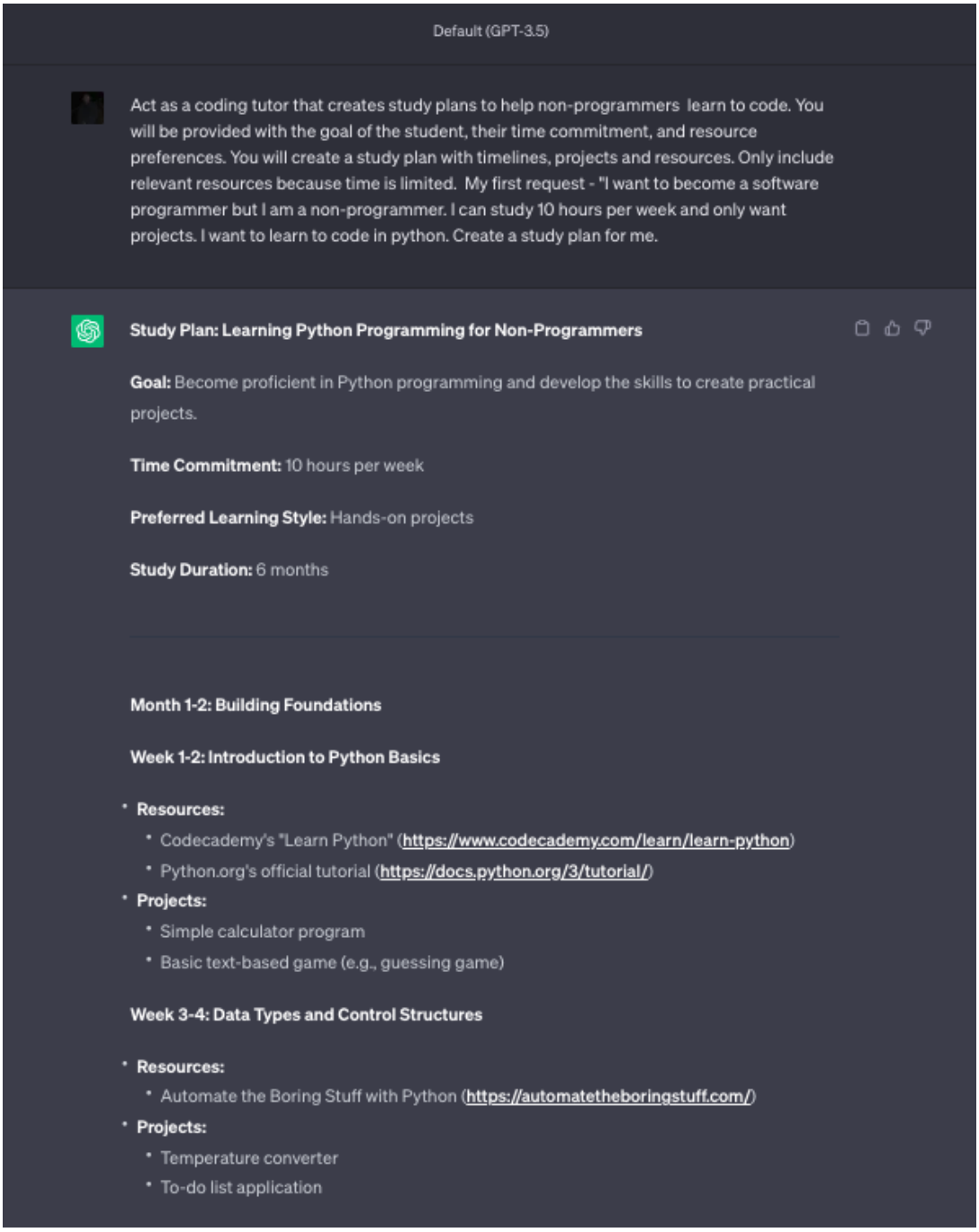


Figure 1. Example of ChatGPT study framework plan with a more detailed approach. (OpenAI, 2023)

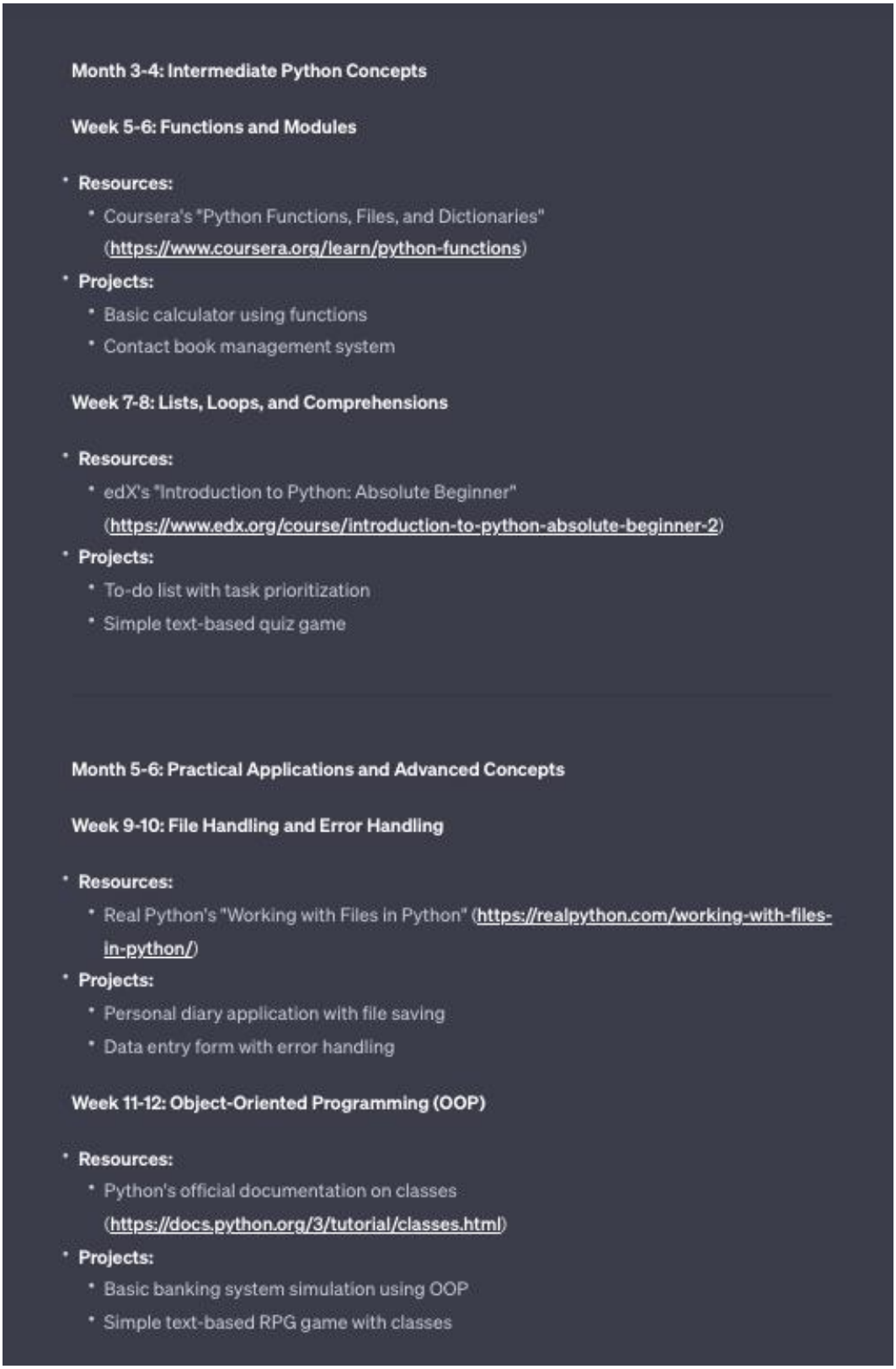


Figure 2. Continued example of ChatGPT study framework plan with a more detailed approach. (OpenAI, 2023).

Project-Focused Learning (Months 7-12)

Month 7-8: Web Development with Flask

- **Resources:**
  - The Flask Mega-Tutorial (https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-i-hello-world)
- **Projects:**
  - Personal website/portfolio using Flask
  - Simple blog application

Month 9-10: Data Analysis with Pandas

- **Resources:**
  - Kaggle's "Pandas Tutorials" (https://www.kaggle.com/learn/pandas)
- **Projects:**
  - Analyzing and visualizing a dataset of your choice
  - Basic data-driven report generator

Month 11-12: Capstone Projects and Refinement

- **Projects:**
  - Choose two complex projects that interest you the most. These could involve web applications, automation tools, data visualization, etc.
  - Focus on refining your code, optimizing performance, and enhancing the user experience.

Remember, consistency is key to successful learning. Set aside dedicated time each week, practice regularly, and challenge yourself with projects that align with your interests. Feel free to explore additional resources and tutorials as you progress. Good luck on your journey to becoming a proficient Python programmer!

Figure 3. Continued example of ChatGPT study framework plan with a more detailed approach. (OpenAI, 2023)

Chapter 4: Conclusion

4.1 Ethical Issues

As a new technology, there is also the risk of misuse, such as for harmful, misleading, or offensive purposes. The content being generated could be used to mislead students, spread misinformation, or even incite plagiarism in code [15].

We should also consider the potential consequences of our actions before we use these models. It is important to be aware of these privacy and safety concerns and to take steps to mitigate them.

4.2 Legal Issues

LLMs are able to generate just about any type of content per the user's request but where does this content come from? Can the user claim the content as their own? Is the LLM liable for misinformation that could lead to plagiarism? The use of LLMs for teaching nonprogrammers for software programming raises a number of ethical concerns, including the potential for bias, privacy violations, and the misuse of generated content [16].

Imagine this: a large monopoly research company buys out the biggest large language model. This company also makes educational materials, such as textbooks, online courses, and assessments. The company could use the LLM to generate inaccurate code or misleading content in these materials. For example, the company could generate content that reinforces existing stereotypes or that promotes the company's own products or services. This could have a negative impact on nonprogrammers' learning and could lead to the perpetuation of bias in society.

In reference to LLMs, this means that it is crucial to integrate the use of these models to better society as a whole, and in this case, not just for the benefit of the company that owns the model. For example, if a company is using an LLM to generate biased or misleading content in

educational materials, it is important to consider whether the use of the model will benefit students or whether it will simply serve to perpetuate existing biases in society.

4.3 Security Issues

LLMs are trained in large dataset of text, code and information from all over the internet. This means that they can be exposed to a wide variety of content, including both good and bad, in order to provide your request in an informative way therefore posing a number of security risks [17].

One of the biggest security risks with LLMs is data privacy. In order for the LLMs to keep expanding their knowledge to answer user requested questions – it has to keep collecting information. LLMs are trained on massive datasets of text and code, which can include personal information such as names, addresses, and email addresses.

This information can be collected from a variety of sources, including public websites, social media platforms, and user-generated content. If the information is not collected and stored securely, it could be accessed by unauthorized individuals, which could lead to privacy violations. Michael Schade from OpenAI promises that when "you share your data with us [OpenAI], it helps our models become more accurate and better at solving your specific problems and it also helps improve their general capabilities and safety."

4.4 Conclusion

Large Language Models (LLMs) can be detrimental to educating nonprogrammers but overall, the impact that it could have on our future generations is positive. Large Language Models are far from perfect, but we must acknowledge that we have only scratched the surface with AI, Chatbots and Language Models.

There are also some challenges that need to be addressed before LLMs can be fully

integrated into education. One particular area that is crucial to be addressed would be within development, specifically potential security and misinformation. By taking steps to address the challenges, we can ensure that LLMs are used in a safe and responsible manner, and that they can help students to reach their full potential.

Overall, the utilization of LLMs for educating nonprogrammers offers substantial potential benefits. By taking steps to address the challenges, educators can create a supportive learning environment that maximizes the advantages of LLMs while safeguarding students' privacy and data security.

Additionally, providing appropriate training and professional development opportunities for teachers will enable them to effectively integrate LLMs into their teaching methods. As we navigate this digital transformation, it is vital to ensure that our educational practices embrace the potential of LLMs while upholding the principles of safety, security, and ethical usage.

Chapter 5: LLM Training Material

5.1 Video Tutorials

GPT-4 - How does it work, and how do I build apps with it? - CS50 Tech Talk

Overview

The CS50 tech talk discussed the growing interest in artificial intelligence (AI) and OpenAI, and provided resources for developers who want to learn more about these technologies. The talk also featured presentations from McGill University and Steamship, which discussed how to make it easier to build and deploy AI-powered applications.

Ted and Still, two researchers from McGill University, gave a presentation on the research behind language models like GPT. They discussed how GPT works and how it can be used to create successful applications. They also provided examples of apps that are already using GPT, such as Bard, a large language model chatbot developed by Google AI.

The speaker from Steamship, a company that provides cloud computing services, discussed their perspective on building AI apps on AWS. They highlighted the recent hackathon

they hosted, which focused on using AI to solve real-world problems. They also emphasized the importance of understanding GPT and how to build with it as a developer.

GPT is a large language model that uses neural networks to predict and assist in writing text. It is trained on a massive dataset of text and code, which allows it to generate text that is both fluent and coherent. GPT has been used to create a variety of applications, including chatbots, text generators, and translation tools.

In conclusion, the CS50 tech talk provided a comprehensive overview of the latest trends in AI and open AI. It also featured presentations from leading experts in the field, who shared their insights and advice on how to build and deploy AI-powered applications.

[How to learn to code FAST using ChatGPT (it's a game changer seriously)]

Overview

Tina Huang, former data scientist at Meta, attempts to learn Python via ChatGPT due to hearing how useful of a tool ChatGPT is [1]. Huang describes ChatGPT as a very knowledgeable tool but not smart. This refers to the lack of understanding of what the user needs at times.

Huang attempts to ask ChatGPT to ask for a study plan to learn Python or Data Science. The LLM gives a decent response for timeline of learning curriculum but lacks a sense of reality since it assumes the user knows a lot of terms in the cs field.

Huang attempts to ask again but this time adds in the request to provide the same plan but with resources and a timeline. The LLM provides resources like W3Schools, hackerrank, and Codecademy per week but aims to do better by using a very specific prompt, “Act as a coding tutor that creates study plans to help people learn to code. You will be provided with the goal of the student, their time commitment, and resource preferences. You will create a study plan with timelines and links to resources. Only include relevant resources because time is limited. My first request - "I want to become a data scientist, but I do not know how to code. I can study 10 hours per week and only want video resources. I want to learn to code in Python. Create a study plan for me."

The LLM provides a solid plan with more aggressiveness near the end. Huang plans to tackle the topics with projects that allow applied learning.

Huang asks ChatGPT, “Act as an expert data scientist and create an exploratory data analysis python data science project about Naruto the anime.” The LLM provides a project outline with objectives, data and a reflection. Huang then prompts to ask for the code used for the project and tests for functionality but fails as it lacks files mentioned in the code.

Huang attempts to understand a common difficult topic that csc students tend to stumble upon, inheritance versus polymorphism and continues to ask the LLM. The LLM successfully acts as a tutor when it comes to specific requests, revising and explaining code.

ChatGPT is a very useful and powerful tool that can be used to learn programming but does not need the assistance of revalidating that information from other resources. Remember to always be specific in your requests.

[How to Learn Python FAST with ChatGPT?]

Overview

Sundas Khalid focuses on learning Python via ChatGPT and refers to ChatGPT as a smart coworker who you can ask whatever you want, whenever you want without the "embarrassment" [14].

Khalid begins the roadmap for Python by asking ChatGPT "Can you give me a study plan to learn Python for data science" and the LLM provides a generic response. Khalid narrows the roadmap by adding that the timeline for it is 8 weeks (about 2 months) and receives a more specific response. She continues to request links and resources but still thinks it's too generic.

Khalid uses the prompt "Act as a Berkeley Data Science Professor and provides new students with 10 weeks (about 2 and a half months) plan to learn Python for data science and data analysis. Students don't have any background in coding and the time is limited. Include resources and links for learning so students can learn in their own time and not in class." and receives a more detailed plan.

Khalid believes hands-on learning is essential for learning something new, especially coding. She continues to request projects in the same prompt and receives more effective results.

Chapter 6: References

[1] Google, “What is Artificial Intelligence (AI)?,” Google, https://cloud.google.com/learn/what-is-artificial-intelligence.

[2] Wikipedia. (2023, August 7). *Artificial Intelligence*. Wikipedia. https://en.wikipedia.org/wiki/Artificial_intelligence#:~:text=Artificial%20intelligence%20(AI)%20is%20the,of%20human%20beings%20or%20animals.

[3] Rouse, M. (2023, June 17). *Weak Artificial Intelligence*. Techopedia. https://www.techopedia.com/definition/31621/weak-artificial-intelligence-weak-ai

[4] Lee, A. (2023, June 26). *What Are Large Language Models Used For?*. NVIDIA Blog. https://blogs.nvidia.com/blog/2023/01/26/what-are-large-language-models-used-for/

[5] *Software Programming Definition*. Learn.org. (n.d.). https://learn.org/articles/What_is_Software_Programming.html

[6] GeeksforGeeks. (2023, February 22). *Turing test in Artificial Intelligence*. GeeksforGeeks. https://www.geeksforgeeks.org/turing-test-artificial-intelligence/

[7] Wikimedia Foundation. (2023a, August 3). *Eliza*. Wikipedia. https://en.wikipedia.org/wiki/ELIZA

[8] Rickard, M. (2023, June 5). The Problem with Tokenization in LLMs. https://matt-rickard.com/the-problem-with-tokenization-in-llms

[9] Maeda, J., & Bolanos, M. (n.d.). *What are Tokens?*. LLM AI Tokens | Microsoft Learn. https://learn.microsoft.com/en-us/semantic-kernel/prompt-engineering/tokens

[10] Abramson, A. (n.d.). *How to use ChatGPT as a learning tool*. Monitor on Psychology. https://www.apa.org/monitor/2023/06/chatgpt-learning-tool

[11] Douglas Heaven, W. (2023, April 6). ChatGPT is going to change education, not destroy it. https://www.technologyreview.com/2023/04/06/1071059/chatgpt-change-not-destroy-education-openai/#:~:text=%E2%80%9CWhile%20the%20tool%20may%20be,Washington%20Post%20in%20early%20January.

[12] Chen, C. (n.d.). *AI Will Transform Teaching and Learning. Let’s Get it Right.* Stanford HAI. https://hai.stanford.edu/news/ai-will-transform-teaching-and-learning-lets-get-it-right

[13] *How to learn to code FAST using ChatGPT (it's a game changer seriously)*. (2023). *YouTube*. Retrieved August 9, 2023, from https://youtu.be/VznoKyh6AXs.

[14] YouTube. (2023b). *How to use ChatGPT to easily learn any skill you want*. *YouTube*. Retrieved August 9, 2023, from https://www.youtube.com/watch?v=MnDudvCyWpc&ab_channel=bridoesthings.

[15] Gordon, C. (2023, May 2). *AI Ethicist Views On ChatGPT*. Forbes. https://www.forbes.com/sites/cindygordon/2023/04/30/ai-ethicist-views-on-chatgpt/

[16] Armitage, P. (2023, July 21). *5 Ethics Issues for ChatGPT and Design*. The Fountain Institute. http://www.thefountaininstitute.com/blog/chat-gpt-ethics

[17] Dash, B., & Sharma, P. (n.d.). *Are CHATGPT and deepfake algorithms endangering the cybersecurity ...* Are ChatGPT and Deepfake Algorithms Endangering the Cybersecurity Industry? A Review. https://www.researchgate.net/profile/Bibhu-Dash-5/publication/368838115_Are_ChatGPT_and_Deepfake_Algorithms_Endangering_the_Cybersecurity_Industry_A_Review/links/63fcd167b1704f343f889dd9/Are-ChatGPT-and-Deepfake-Algorithms-Endangering-the-Cybersecurity-Industry-A-Review.pdf

Chapter 7: Acknowledgement

The author would like to express my sincere gratitude to Dr. Karl Ricanek for their wonderful guidance and support in the creation of this paper. Their expertise and knowledge in the field of large language models was invaluable, and I am grateful for their patience and encouragement throughout the research and writing process.

The author would also like to thank the UNCW M2S LSAMP program for providing me with the financial support and resources necessary to conduct this research. Their assistance was essential in helping me to identify a research investigation and to further a passion for learning more about software programming and artificial intelligence.

Finally, the author would like to thank the researchers, doctors, scientists, and other authors mentioned in this paper for their work on the use of Large Language Models For Teaching Software Programming to Non-Programmers. Their insights and findings have been instrumental in shaping my understanding of this important issue. This work would not have been possible without the support of these individuals and organizations. The author is deeply grateful for their contributions.