# Code improvement report

Roblox prime numbers:

Oleh Basystyi
Anna Stasyshyn
Artur Rudish
Anton Valihurskyi
Maksym Zhuk

April 2024

# 1 Introduction

In this little research, our team researched AI's capability to optimize Python code in different respects: *style*, *memory usage*, *runtime*. In this part we used only ***ChatGPT-3.5*** and ***Copilot***. Here we will provide only aggregate conclusions among different types of optimization techniques. Full list of used tasks with some statistical data as *lines of code before and after the optimization, runtime* and detailed summary to each task can be found in table on second sheet. Also, we have the GitHub with where you can find original and optimized versions of problems.

# 2 ChatGPT-3.5 vs Copilot

ChatGPT-3.5 vs Copilot At the beginning of this part, our team have decided to use these two AIs. And on average we can see that ChatGPT-3.5 fit more for code optimization than Copilot. Especially, you can see that in first seven rows in the table. So it is better to use Copilot only in code generation and avoid struggling

# 3 Style optimization

From our observations, we can say that ChatGPT is very good at improving code style and readability. It can analyze codebase, split it into separate blocks write code that will be far more readable than the original one. Also, Chat- GPT showed a great capability of rewriting code into functional paradigm. Also, AI is good at following the PEP8, with some exceptions, when it became irresponsible to pylint messages. Although some issues occurred. When AI rewrite complex or illogically written code, it tends to forget some details as writing module docstring or totally forgets about existence of particular function. Moreover, if precedence of operations was crucial in original code, it can rearrange them incorrectly. Despite listed issues, we think that AI's work with style improvement good enough, so they can be used to optimize style of great volumes of legacy code, if it has enough test coverage to avoid listed AI lags

# 4 Runtime & memory optimization

As we have seen in our survey, AI struggle in fully independent code optimization. It never gets send right code from the first try. As said many times before AI just forget about crucial detail to make code more readable and output just totally not-working code as in the

table #2, rows 14, 18. Hovever, after several manual directions on what it made wrong, we can improve performance, but it only works with problems where problem statement and overall code is not complex as in the table #2, row 18, but struggle to do the same thing in task at row 18.

# 5 Corner cases problem optimization

Also, we have tested how AI can work with Pandas in the table #2, rows 8-9 and OOP, row 10. In summary, ChatGPT manages to understand purpose of function and tries to improve them where possible. In both Pandas problems it improved long filters readability by dividing them into separate blocks, but as was said before sometimes it changed the order of operation and it had broken program. Also, ChatGPT find unnecessary parts of code and correctly eliminate them, which can improve code memory usage and runtime.

In OOP optimization, we notice the following AI behaviors:

- It tends to select parts of script (classes) where optimization is mostly needed and sends only changed version of that parts - and this is very convenient. On other hand, when AI does that it usually forget about existence of other classes in the script.
- It tend to forget to include all methods into the optimized version and to inherit all parent classes.
- It does not totally understand OOP principles and can move methods from base to child classes

So, when you try to optimize OOP through AI these issues should be taken into the consideration. And it's better to add the following prompts.

- Please optimize this code **{*code*}** this is problem statement **{*problem_statement*}** and required test cases **{*test_cases*}**

- Optimize code do not cut out distinct classes from it
- Do not copy methods from parent classes to child

# 6 Conclusion

In summary, AIs is not capable of full independent code optimization right now. They always forget something important or because unresponsive to different prompts and send the same solution with unchanged bugs. Despite this, AIs can make code far more readable, but do not guarantee that it will be 100% working. But, generally AIs can give a developer really nice directions for improvement which require a little fix to make code working. If you are up to using AI for code optimization, you should follow these Recommendations:

- In the beginning you should send all of **problem description, code, and test cases**
- Try not to give general promts like *"Optimize this code"*, but specify what part of program must be optimized
- If AI struggle to solve bug after several prompts (sends the same code) you should provide **function name** or **class name** where the error occured
- Sometimes AI struggle to solve even the PEP8 errors from error signatures, so you should send prompt like *"Split the documentation into two lines"*, because AI can be irresponsible to prompt like *"10th line is to long, make it shoter"*
- Do not rellay totally on AI you should only use it to search for optimizatioin hints, but not as complete tool fot code improvement.