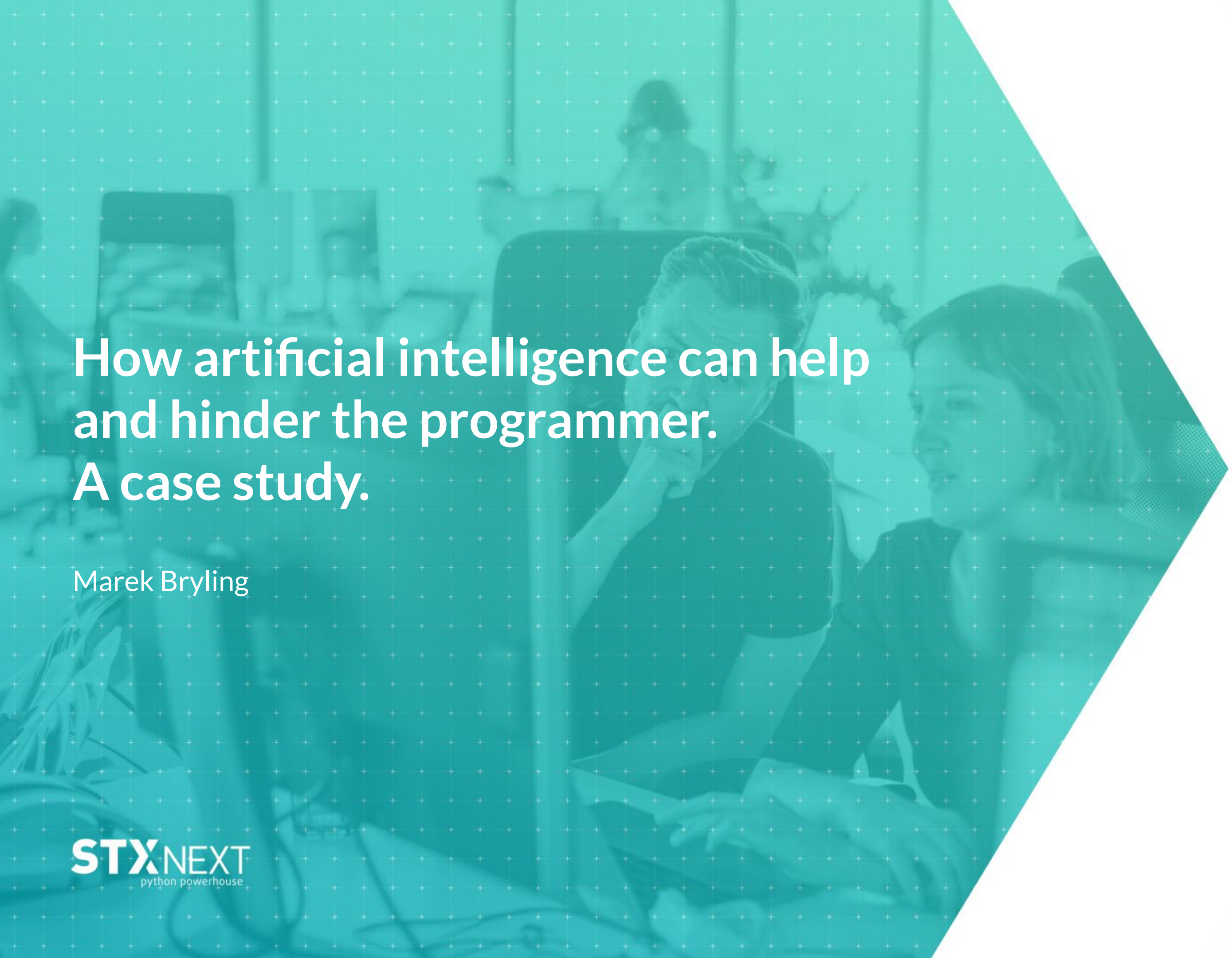


How artificial intelligence can help and hinder the programmer. A case study.

Marek Bryling

STXNEXT
python powerhouse

Disclaimer

Most of the slides in this presentation were generated by ChatGPT and translated into English by DeepL.

STXNEXT
python powerhouse

About the author

Marek Bryling

Practice Lead at STX Next

- Team support
- Introducing new practices and technologies
- Improving knowledge sharing
- Overseeing R&D projects
- Company-wide engagement
- 30 Years of IT industry experience

”

- “Code churn,” or the percentage of lines thrown out less than two weeks after being authored, is on the rise and expected to double in 2024. The study notes that more churn means higher risk of mistakes being deployed into production.
- The percentage of “copy/pasted code” is increasing faster than “updated,” “deleted,” or “moved” code. “In this regard, the composition of AI-generated code is similar to a short-term developer that doesn’t thoughtfully integrate their work into the broader project,”

GitClear founder Bill Harding

STX NEXT
python powerhouse

Introduction

Definition of AI in the context of a programmer's work

https://chat.openai.com/, https://copilot.microsoft.com/, https://codium.ai/, https://snyk.io/, https://coderabbit.ai/

Copilot

Your everyday AI companion

limerick about me

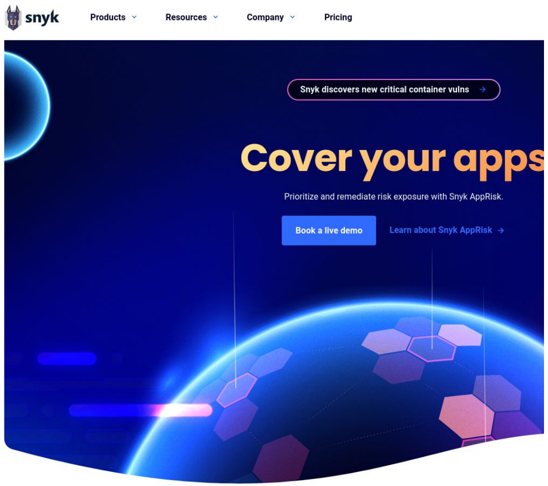
Create a personal website using HTML/Javascript/CSS with my biography, resume, and contact d...

What's something fun th tonight?

Generating code reviews for busy devs

With CodiumAI, you get non-trivial suggestions (and trivial, too!) suggested right inside your IDE or Git platform, so you can code smart and stay confident when you push.
Code, as you meant it.

Get your FREE Codiumate now:



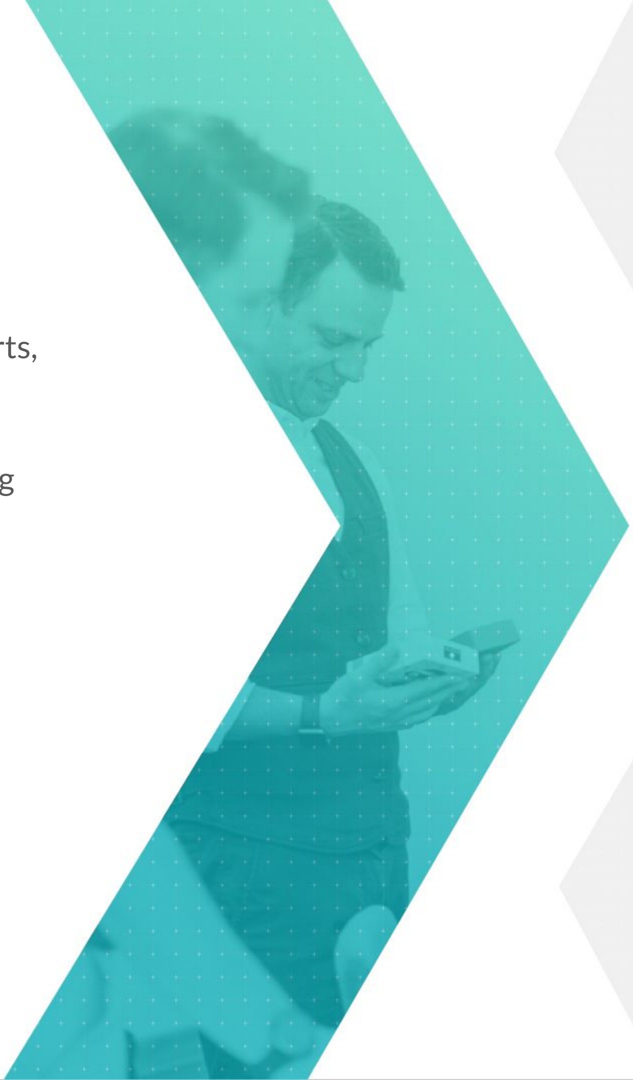
Snyk discovers new critical container vulns

Cover your apps

Prioritize and remediate risk exposure with Snyk AppRisk.

How AI can help in a programmer's work

- Automate routine tasks such as testing code, generating reports, and optimizing processes
- Increase efficiency through data analysis and trend forecasting
- Facilitate decision-making with machine learning algorithms



Generated by ChatGPT

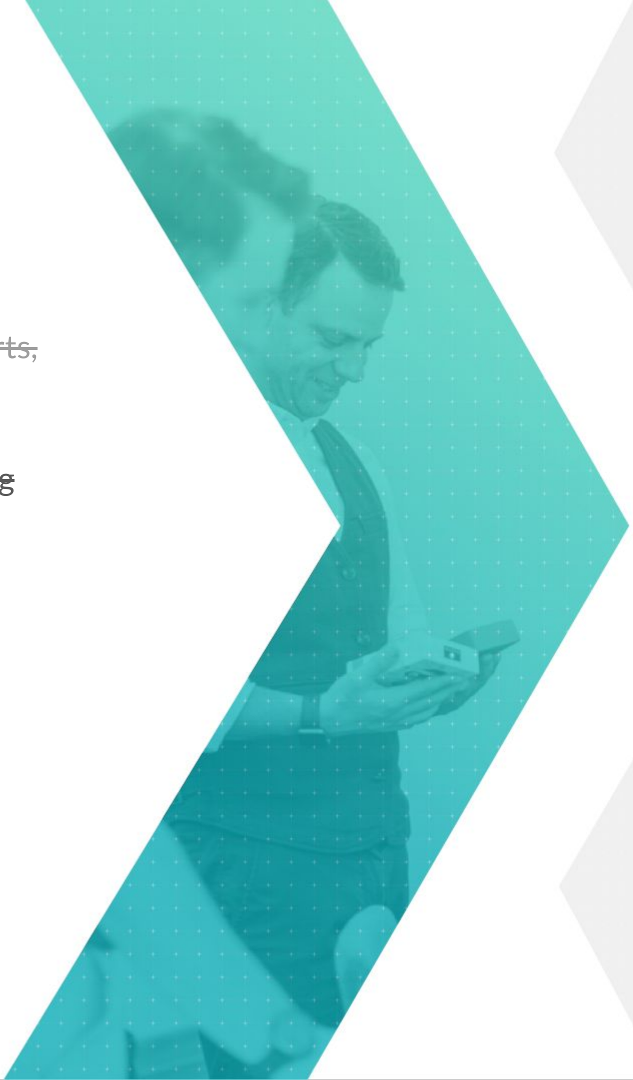
How AI can help in a programmer's work

- ~~Automate routine tasks such as testing code, generating reports, and optimizing processes~~
    - Code hinting
    - Code prompting
    - Code generation
    - Code optimizations
    - Finding bugs in code
    - **Using new and unfamiliar technologies**
- Increase efficiency through data analysis and trend forecasting
- Facilitate decision-making with machine learning algorithms

Generated by ChatGPT

How AI can help in a programmer's work

- ~~Automate routine tasks such as testing code, generating reports, and optimizing processes~~
- ~~Increase efficiency through data analysis and trend forecasting~~
    - Code profiling
    - Optimizations
    - Finding vulnerabilities
- Facilitate decision-making with machine learning algorithms



Generated by ChatGPT

How AI can help in a programmer's work

- ~~Automate routine tasks such as testing code, generating reports, and optimizing processes~~
- ~~Increase efficiency through data analysis and trend forecasting~~
- ~~Facilitate decision-making with machine learning algorithms~~
    - Automated code review
    - A new solution to the problem
    - Suggestions for new libraries
    - I don't know how to do it, AI will tell me

Generated by ChatGPT

How AI can spoil into a programmer's job

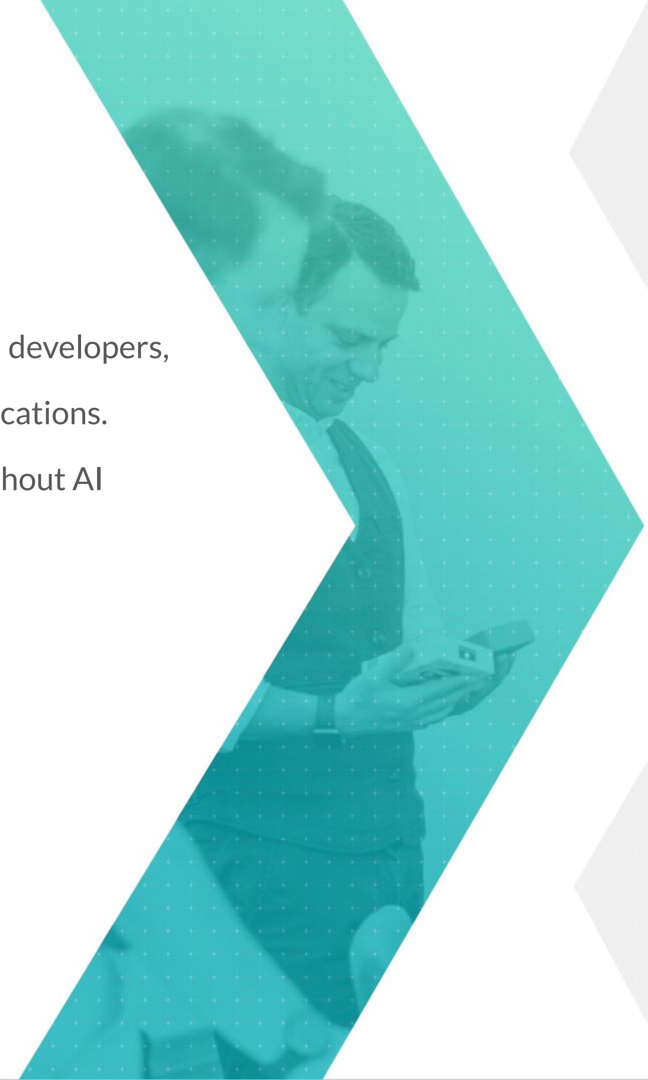
- The risk of errors in data interpretation and algorithm performance
- Potential for AI-based systems to replace the work of programmers
- Potential limitation of programmers' creativity and inventiveness through dependence on smart tools

Generated by ChatGPT

How AI can spoil into a programmer's job

- It is estimated that for about ~~30-50%~~ 30-33%* of junior-level developers, AI can be a problem when working with more advanced applications.
- The customer doesn't want AI, but the developer can't do without AI
- The generated code is of poor quality
  - Code review hell
  - Prompting hell
  - Security issues
  - Unreadable code
  - more …

Generated by ChatGPT



Research

[Coding on Copilot: 2023 Data Suggests Downward Pressure on Code Quality]

2023 Data Shows Downward Pressure on Code Quality

150m lines of analyzed code + projections for 2024

… GitHub's research finding on this point says "developers are 75% more fulfilled when using Copilot." …

AI is like a "brand new credit card here that is going to allow us to accumulate technical debt in ways we were never able to do before," Armando Solar-Lezama, a professor at MIT

Research

[The Impact of AI Tool on Engineering at ANZ Bank An Empirical Study on GitHub Copilot within Corporate Environment](), [Blog post with a summary]()

The findings were notable: the group using Copilot completed their tasks 42.36% faster than the control group. Specifically, the control group took an average of 30.98 minutes per task, while the Copilot group averaged 17.86 minutes.

Data generated by ChatGPT

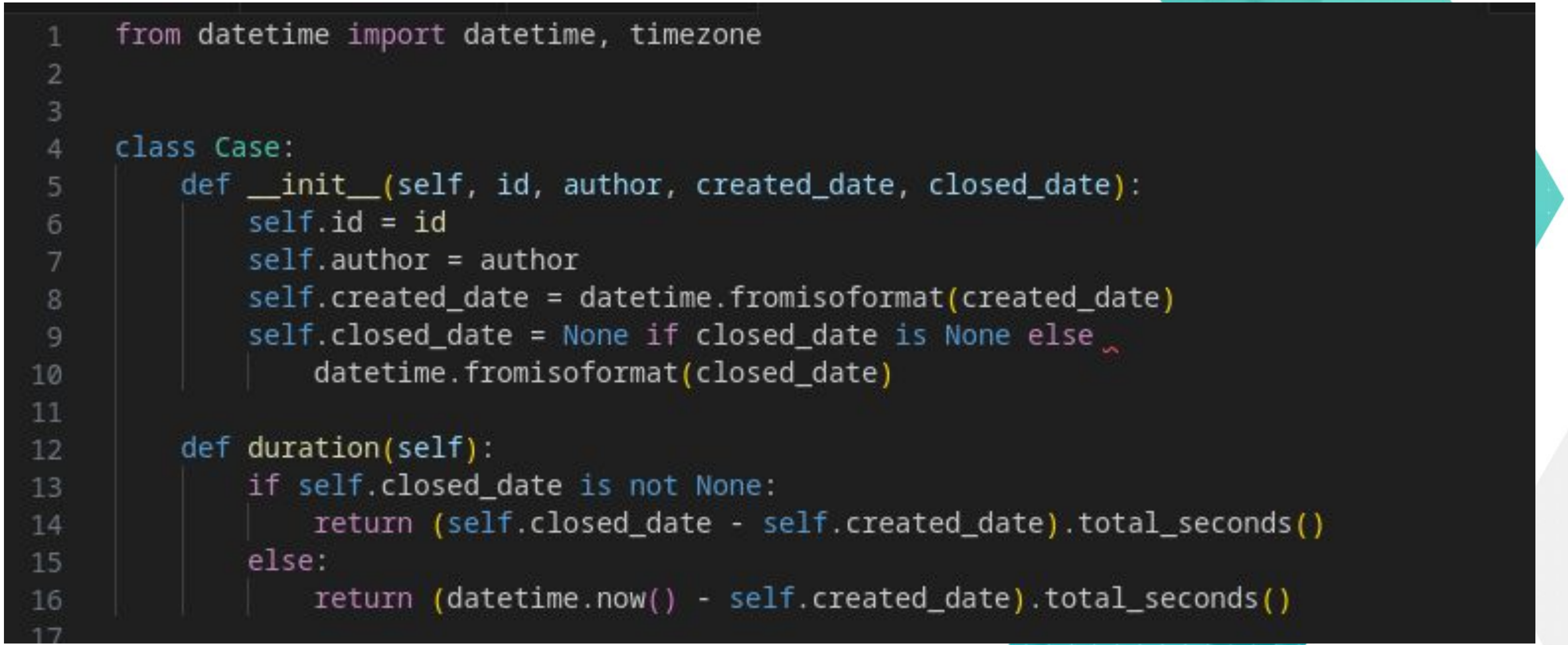
Case study

Create a Case class that can be initialized from data like the example(s) below.

Add functionality whereby the elapsed open time (duration) for any Case may be retrieved (in seconds).

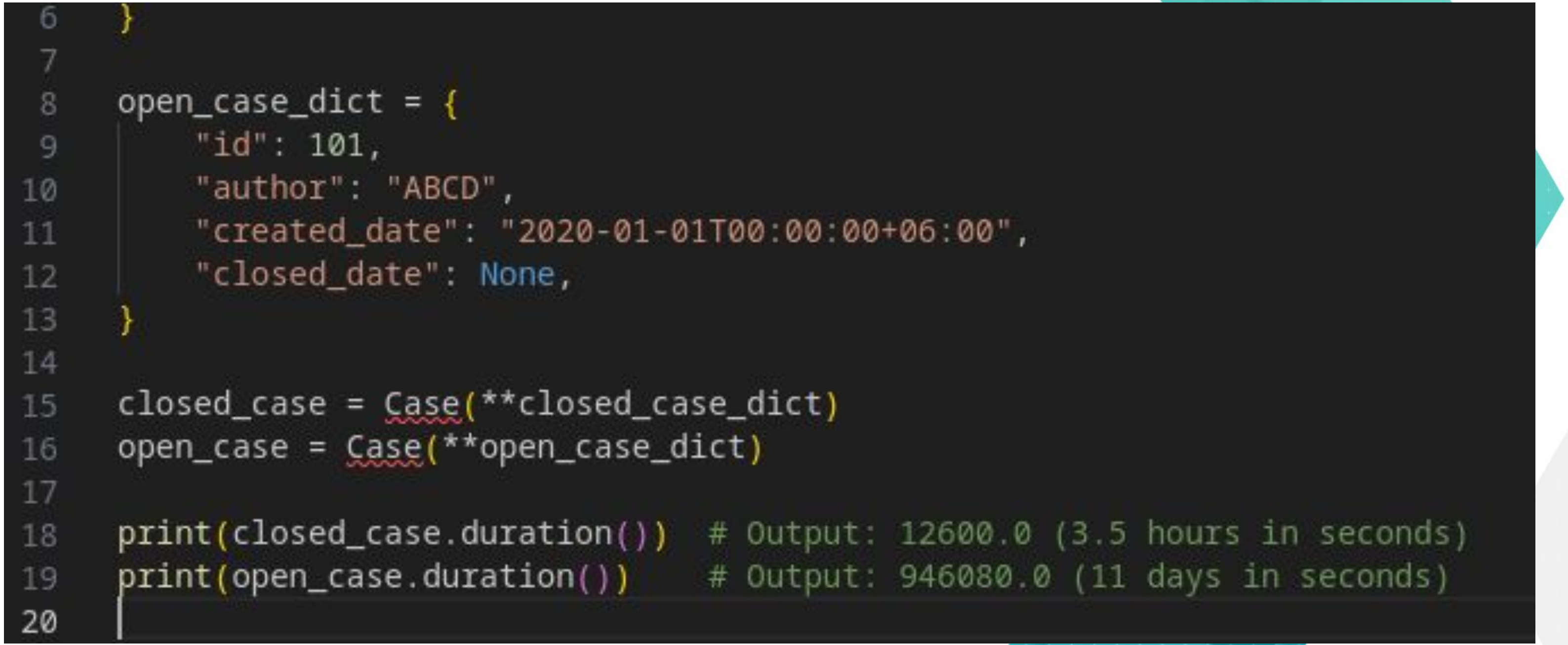
```
closed_case_dict = {“id": 101,
                "author": "ABCD",
                "created_date": "2020-01-01T00:00:00+00:00",
                "closed_date": "2020-01-01T12:00:00+00:00"}
open_case_dict = {"id": 101,
                "author": "ABCD",
                "created_date": "2020-01-01T00:00:00+00:00",
                "closed_date”: None}
```

Case study (ChatGPT 3.5)



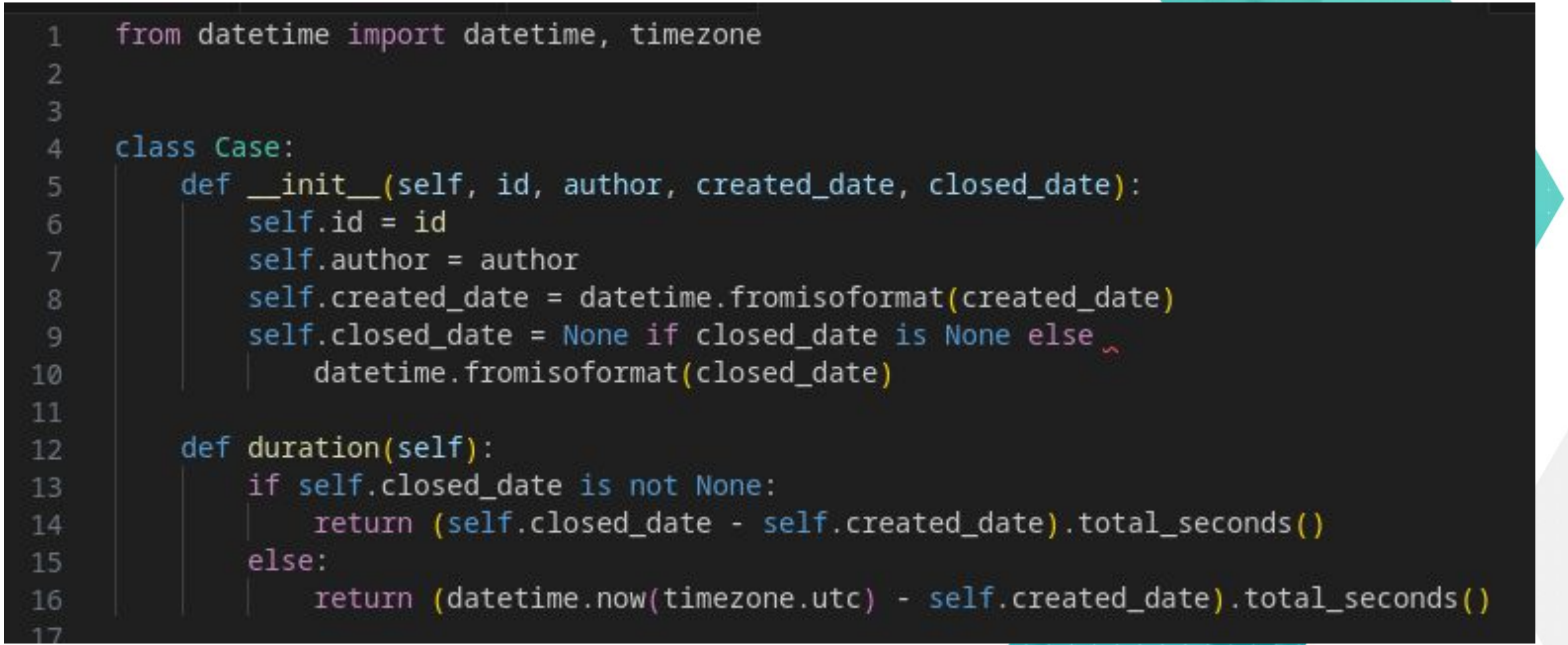
```python
from datetime import datetime, timezone


class Case:
    def __init__(self, id, author, created_date, closed_date):
        self.id = id
        self.author = author
        self.created_date = datetime.fromisoformat(created_date)
        self.closed_date = None if closed_date is None else
            datetime.fromisoformat(closed_date)

    def duration(self):
        if self.closed_date is not None:
            return (self.closed_date - self.created_date).total_seconds()
        else:
            return (datetime.now() - self.created_date).total_seconds()
```

Case study - test (ChatGPT 3.5)



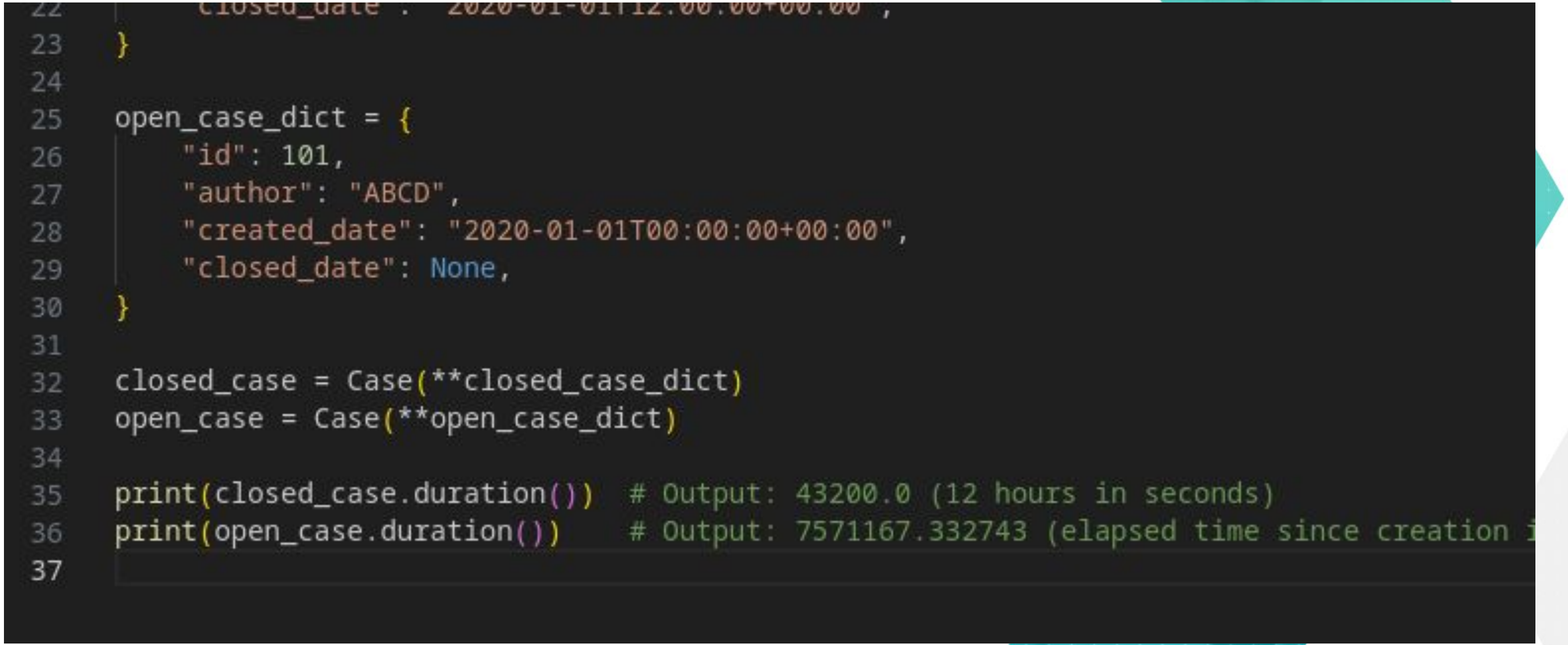
```python
}

open_case_dict = {
    "id": 101,
    "author": "ABCD",
    "created_date": "2020-01-01T00:00:00+06:00",
    "closed_date": None,
}

closed_case = Case(**closed_case_dict)
open_case = Case(**open_case_dict)

print(closed_case.duration())  # Output: 12600.0 (3.5 hours in seconds)
print(open_case.duration())    # Output: 946080.0 (11 days in seconds)

```

Case study (ChatGPT 3.5-turbo)



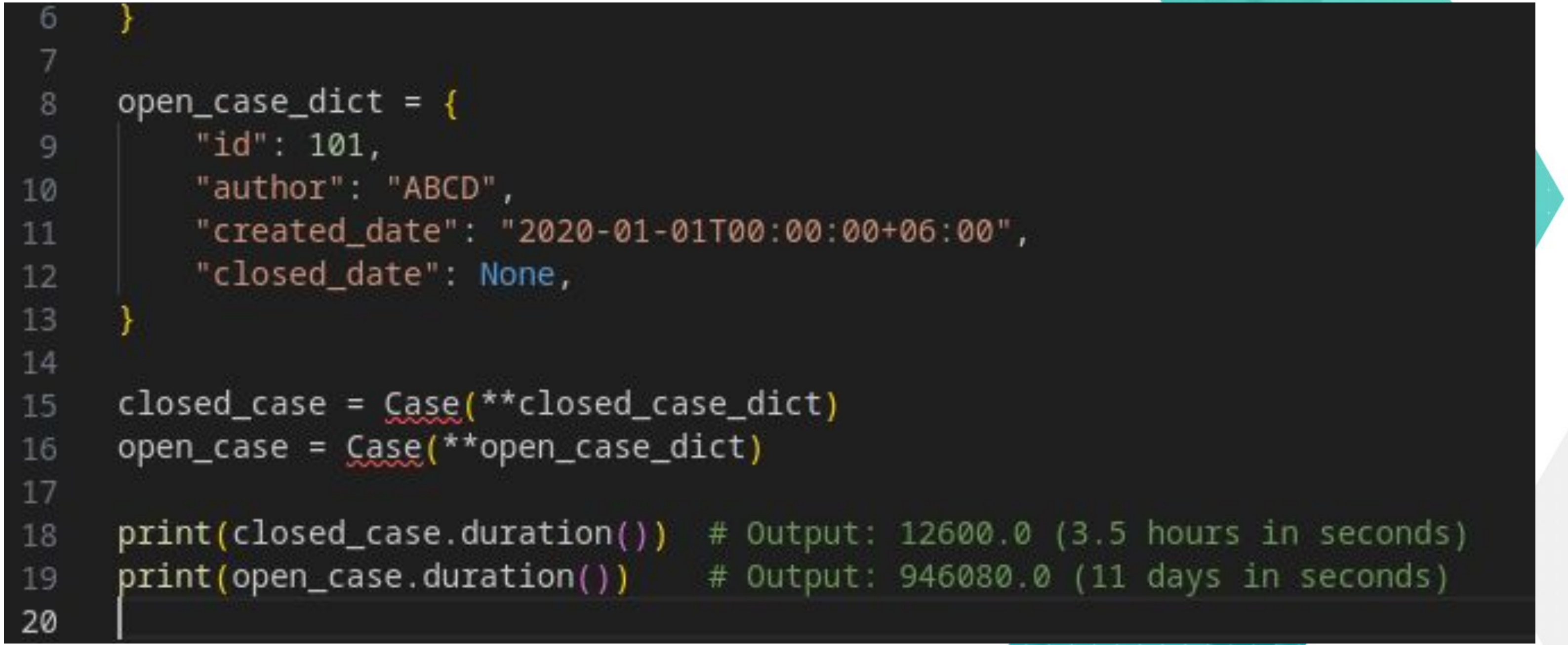
```python
from datetime import datetime, timezone


class Case:
    def __init__(self, id, author, created_date, closed_date):
        self.id = id
        self.author = author
        self.created_date = datetime.fromisoformat(created_date)
        self.closed_date = None if closed_date is None else
            datetime.fromisoformat(closed_date)

    def duration(self):
        if self.closed_date is not None:
            return (self.closed_date - self.created_date).total_seconds()
        else:
            return (datetime.now(timezone.utc) - self.created_date).total_seconds()
```

Case study (ChatGPT 3.5-turbo)



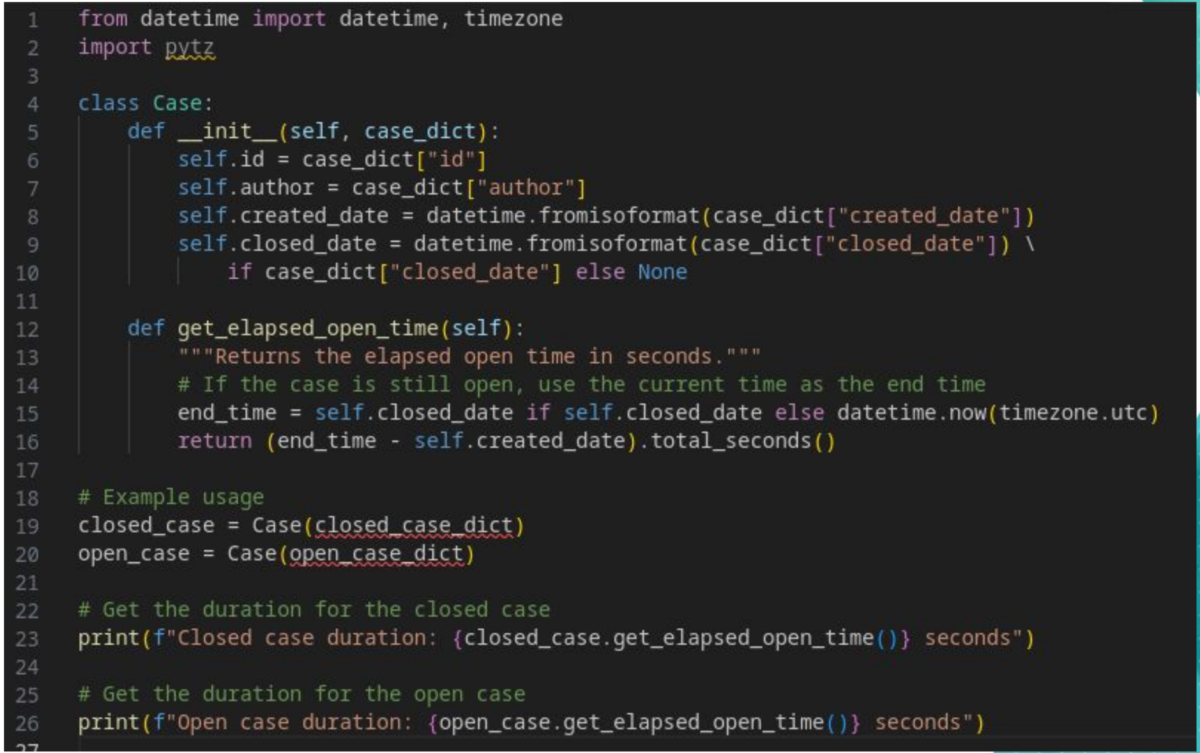
```
    "closed_date": "2020-01-01T12:00:00+00:00",
}

open_case_dict = {
    "id": 101,
    "author": "ABCD",
    "created_date": "2020-01-01T00:00:00+00:00",
    "closed_date": None,
}

closed_case = Case(**closed_case_dict)
open_case = Case(**open_case_dict)

print(closed_case.duration())  # Output: 43200.0 (12 hours in seconds)
print(open_case.duration())    # Output: 7571167.332743 (elapsed time since creation i
```

Case study - test (ChatGPT 3.5-turbo)

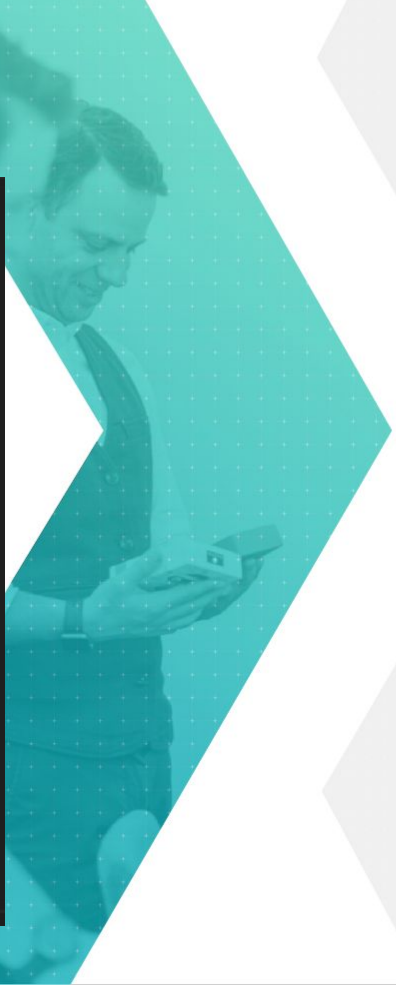


```
}

open_case_dict = {
    "id": 101,
    "author": "ABCD",
    "created_date": "2020-01-01T00:00:00+06:00",
    "closed_date": None,
}

closed_case = Case(**closed_case_dict)
open_case = Case(**open_case_dict)

print(closed_case.duration())  # Output: 12600.0 (3.5 hours in seconds)
print(open_case.duration())    # Output: 946080.0 (11 days in seconds)

```

Case study (ChatGPT 4)



```python
from datetime import datetime, timezone
import pytz

class Case:
    def __init__(self, case_dict):
        self.id = case_dict["id"]
        self.author = case_dict["author"]
        self.created_date = datetime.fromisoformat(case_dict["created_date"])
        self.closed_date = datetime.fromisoformat(case_dict["closed_date"]) \
            if case_dict["closed_date"] else None

    def get_elapsed_open_time(self):
        """Returns the elapsed open time in seconds."""
        # If the case is still open, use the current time as the end time
        end_time = self.closed_date if self.closed_date else datetime.now(timezone.utc)
        return (end_time - self.created_date).total_seconds()

# Example usage
closed_case = Case(closed_case_dict)
open_case = Case(open_case_dict)

# Get the duration for the closed case
print(f"Closed case duration: {closed_case.get_elapsed_open_time()} seconds")

# Get the duration for the open case
print(f"Open case duration: {open_case.get_elapsed_open_time()} seconds")
```



The Future

- DeepNext www.deepnext.ai
- The tools listed above will get better and better
- Programmers will still be needed,

  but their work will look different.
- At some stage, we will no longer understand the code.

Questions (or not)?