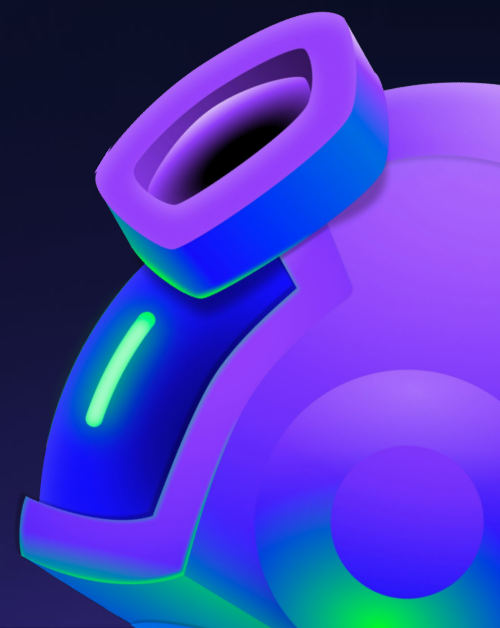




# GitHub Copilot

## Workshop





# 本日の講師



ダニエル・チョ

シニアソリューションズエンジニア  
GitHub



# 本日のアジェンダ

01

GitHub Copilot  
現状

30分

02

質疑応答

20分

03

ハンズオン

60分

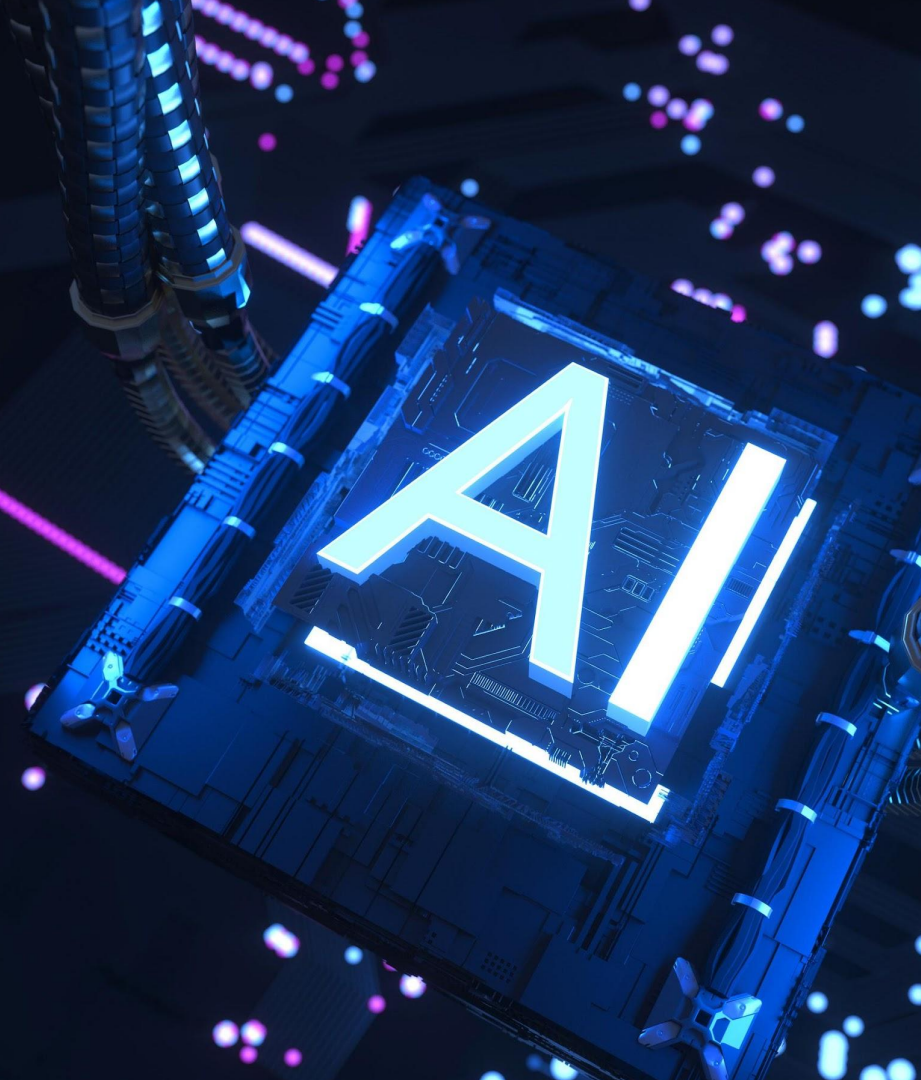
04

今後の動向 &  
まとめ

25分

第1部

# GitHub Copilot 現状



# GitHub Platform

安全なソフトウェアをビルド・  
スケール・リリースできる  
AI原動の開発基盤

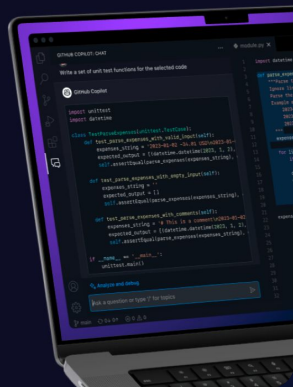




# 開発 エンジニアの 第一候補



生成AI系開発  
ツール利用中の  
55%の開発エン  
ジニアはGitHub  
Copilotを選ぶ



75%の開発エン  
ジニアが仕事に  
より満足



75%の開発者  
は 来年試した  
いと云及

## SUGGESTIONS

🔍 Loading suggestions...

Ask a question here. Select code for context.

GitHub Copilot  
を使って  
コード業務を 55%  
速く遂行

利用者数  
150万ユーザ以上

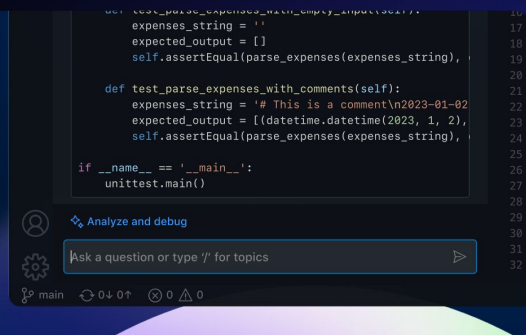


# AIの企業採用 の先駆者

20%以上  
のFortune 500  
企業はGitHub  
Copilot を利用  
(リリースから1年未満時点)



37,000+ の企業は  
GitHub Copilotと  
ともにソフトウェア  
を開発



duolingo



stripe



# GitHub Copilot

自分のプロジェクトに特化した  
ステートレスコード提案を提示

✔ コメントをコードに変換

✔ 繰り返しコードの自動補完

✔ 代替手段の提示

TS sentiment.ts write\_sql.go parse\_expenses

```
1 #!/usr/bin/env ts-node
2
3 import { fetch } from "fetch-h2";
4
5 // Determine whether the sentiment of
6 // Use a web service
7 async function isPositive(text: string)
8   const response = await fetch(`http://
9     method: "POST",
10    body: `text=${text}`,
11    headers: {
12      "Content-Type": "application/x-www
13    },
14  });
15  const json = await response.json();
16  return json.label === "pos";
17 }
```

 Copilot



# GitHub Copilot Chat

エディタ上のコンテキストを踏まえた、Copilotとのチャット

✓ コーディングに関する質問への回答

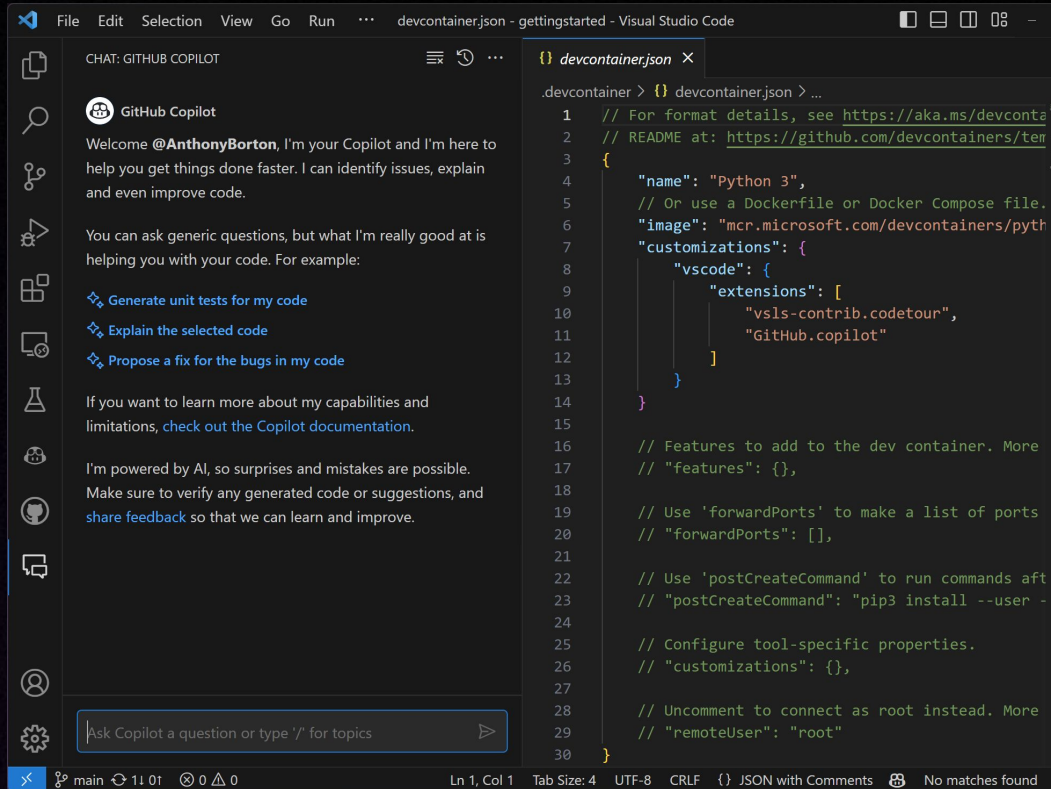
✓ コードの説明

✓ ユニットテストの生成

✓ コードの修正提案

✓ 可読性の向上

✓ 言語変換



The screenshot shows the Visual Studio Code interface with the GitHub Copilot Chat sidebar on the left and a Dockerfile open in the editor on the right.

**CHAT: GITHUB COPILOT**

**GitHub Copilot**

Welcome @AnthonyBorton, I'm your Copilot and I'm here to help you get things done faster. I can identify issues, explain and even improve code.

You can ask generic questions, but what I'm really good at is helping you with your code. For example:

- Generate unit tests for my code
- Explain the selected code
- Propose a fix for the bugs in my code

If you want to learn more about my capabilities and limitations, [check out the Copilot documentation](#).

I'm powered by AI, so surprises and mistakes are possible. Make sure to verify any generated code or suggestions, and [share feedback](#) so that we can learn and improve.

Ask Copilot a question or type '/' for topics

**devcontainer.json**

```
devcontainer > {} devcontainer.json > ...
1 // For format details, see https://aka.ms/devconta
2 // README at: https://github.com/devcontainers/ter
3 {
4   "name": "Python 3",
5   // Or use a Dockerfile or Docker Compose file.
6   "image": "mcr.microsoft.com/devcontainers/pyth
7   "customizations": {
8     "vscode": {
9       "extensions": [
10        "vs1s-contrib.codetour",
11        "GitHub.copilot"
12      ]
13    }
14  }
15
16 // Features to add to the dev container. More
17 // "features": {},
18
19 // Use 'forwardPorts' to make a list of ports
20 // "forwardPorts": [],
21
22 // Use 'postCreateCommand' to run commands aft
23 // "postCreateCommand": "pip3 install --user -
24
25 // Configure tool-specific properties.
26 // "customizations": {},
27
28 // Uncomment to connect as root instead. More
29 // "remoteUser": "root"
30 }
```

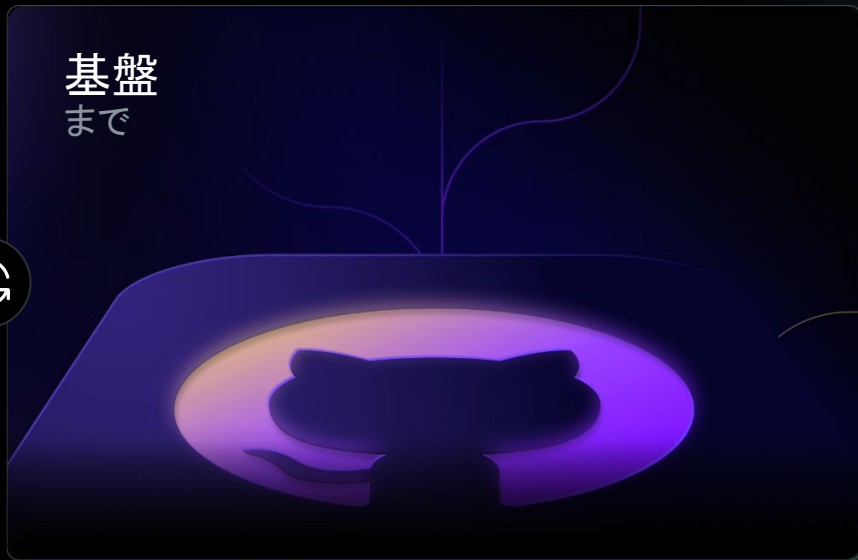


# 世界で最も人気なIDE向けの拡張機能

IDEと  
ターミナル  
から



基盤  
まで



# GitHub Copilot



LLM



文脈

提案



Visual Studio



Neovim



VS Code



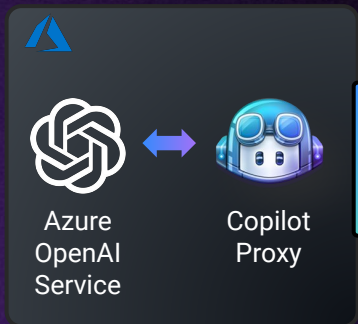
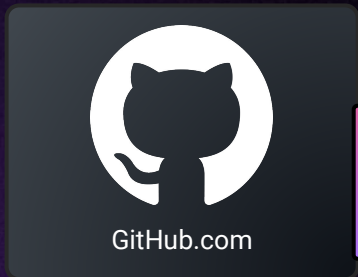
JetBrains IDEs

```
runtime.go course.rb time.js lsPrimeTest.java
1 package main
2
3 type Run struct {
4     Time int // in milliseconds
5     Results string
6     Failed bool
7 }
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
```



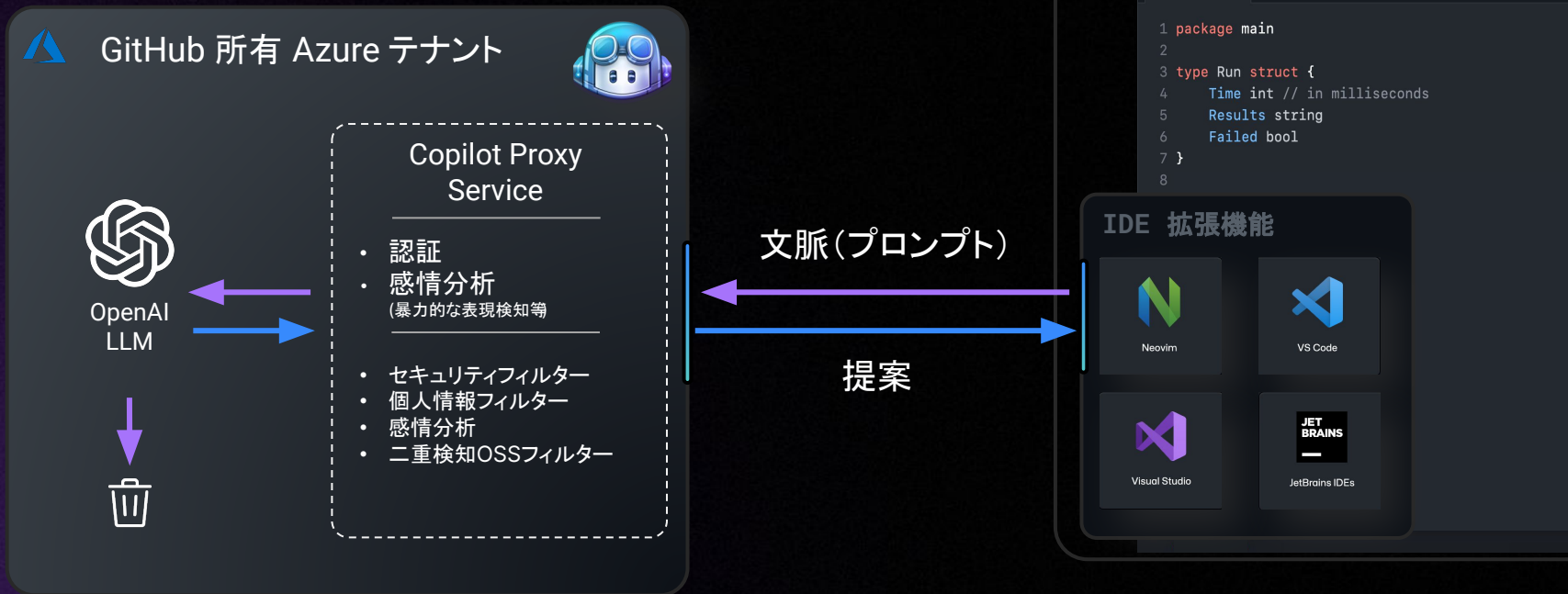


# 認証





# データの流れ



# デモ

GitHub Copilot

# Accenture社内の450人の開発エンジニアに対し 6ヶ月間かけた GitHub Copilot 効果の調査結果



活動量

94%

フロー状態が維持できたと報告した割合

90%

調査に割いた時間の削減を実現した割合



生産性

90%

より品質が高いコードがかけたと報告した割合

88%

Copilot が提案した内容のうち利用した割合



効率性

50%

ビルドの増加率

84%

成功したビルドの増加率



満足度

96%

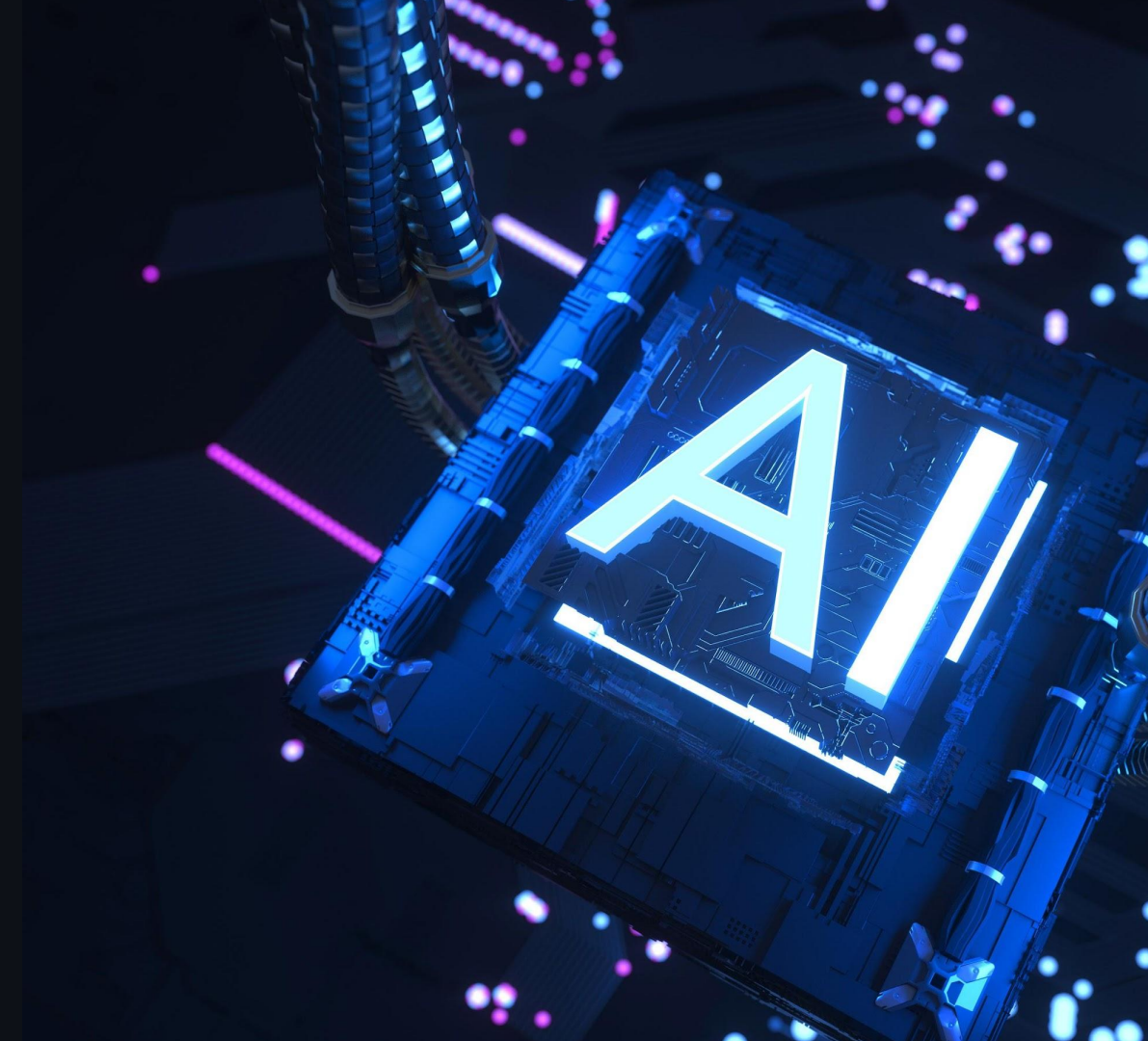
初日から成功した割合

90%

より満足できた割合

第2部

# 質疑応答





# よくあるご質問

出典:

<https://github.com/features/copilot/>

<https://resources.github.com/ja/copilot-trust-center/>

## Q. GitHub Copilotが収集したデータは学習に使われますか？

- A. いいえ、GitHub Copilot Businessの場合は提案を生成するために収集した顧客データ(コード等)を学習やサービス改善に一切利用しません。  
なお、GitHub Copilot Individualsの場合は契約時または個人設定にてテレメトリの設定を変更することによって収集したデータをサービス改善に利用される可能性があります。

**Q. GitHub Copilotは学習したデータをそのまま検索してユーザーに提示する仕組みでしょうか？**

A. いいえ、GitHub CopilotはOpenAI社にて開発されたGPT系モデルにて稼働している生成式事前学習済みの言語モデル型AIになります。GitHub Copilotにて生成される提案は全てコードやコメント等の文脈を考慮したものであり、学習データに対して検索をかけて提案として表示する検索エンジンではございません。

## Q. GitHub Copilotの学習データは何でしょうか？

- A. GitHub CopilotはOpenAI社のGPT系モデルを利用しております。学習データはOpenAI社にて定めており、GitHubやMicrosoftによる介入や選別等は行っておりません。

## Q. GitHub Copilotが生成したコードの所有者はGitHubになりますか？

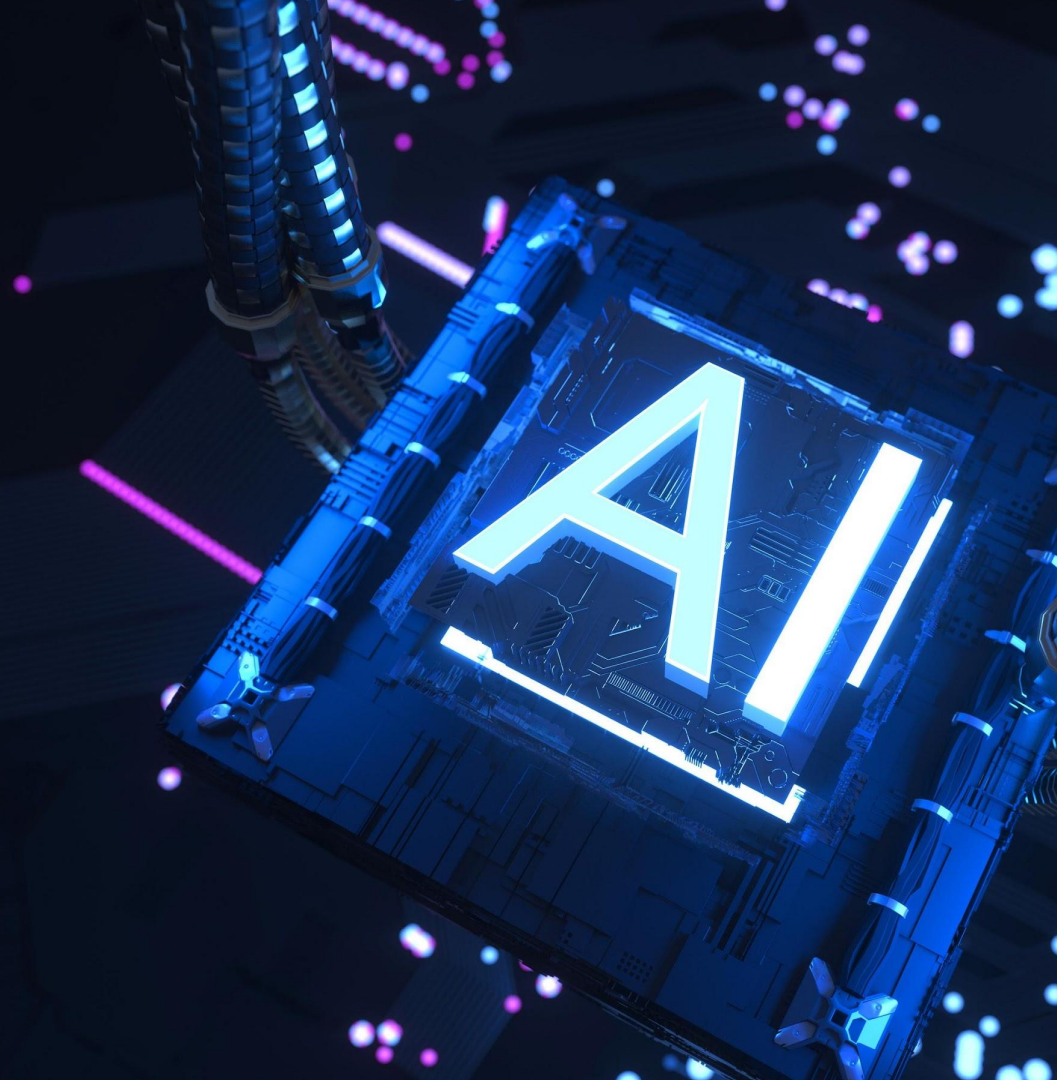
- A. GitHub Copilotはコンパイラやペンのような道具です。GitHubはGitHub Copilotが提案する内容の所有権を一切所持しておりません。GitHub Copilotを利用して作成しているコードの責任は作成者のものであり、本番環境で利用する前に緻密にコードをテスト、評価、そしてレビューすることを推奨しております。

## Q. GitHub Copilot は外部認証を受けていますか？

- A. 2024年6月時点ではGitHub Copilot BusinessはSOC 2 Type 1認証及びGitHubのISO/IEC 27001:2013認証の対象になっております。今後は更にGitHubのSOC 2 Type 2の認証対象に加えることが予定されております。

第3部

ハンズオン





# ハンズオン

最初の始め方の紹介

注意: Copilotは  
確率的モデルであり、  
決定論的モデルではない



GitHub Copilot  
Your AI pair programmer







ハンズオン シナリオ

## Pythonでじゃんけんゲームを 作成

まずはじゃんけんゲームを作りながら様々なCopilotの  
使い方を学びます

別資料の案内に従いできるだけ多くの内容をカバーしま  
す

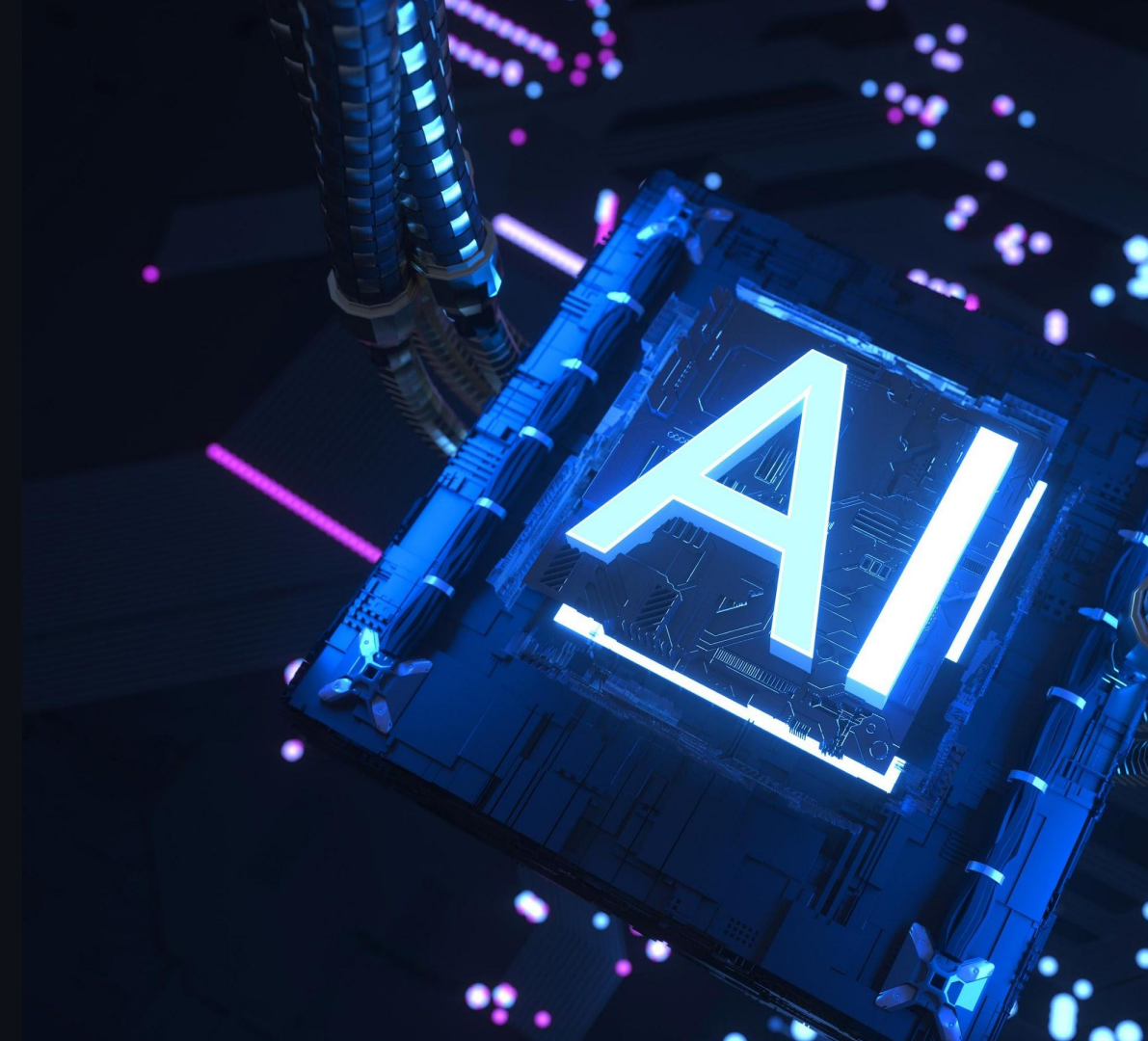
ハンズオン資料:

[https://github.github.com/workshops  
/copilot-for-dev-ja/index.html#1](https://github.github.com/workshops/copilot-for-dev-ja/index.html#1)



第4部

# 今後の動向





# GitHub のビジョン



統合された一つの  
エンタープライズ  
向け基盤



すべての工程に  
セキュリティを  
実装



開発エンジニア向け  
業界最先端の  
コラボレーションツール



AI 原動





# GitHub Copilot Business

ソフトウェアを  
AIと一緒に  
開発するおともに

55%

早くコードを書く

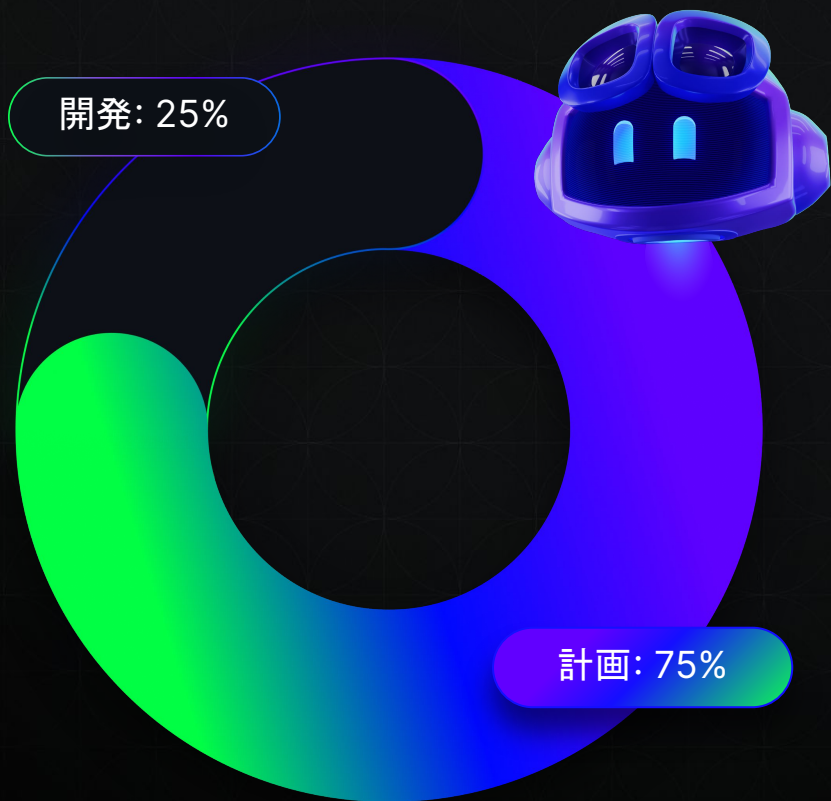
46%

のコードはAIが作成

15%

高いコード品質

エンジニアの  
業務時間のうち  
75% は開発以外の  
タスク



# 乗り越えるべき課題

複雑なコードベース

断片的な知識共有・処理

長いオンボード期間

厳しい納期

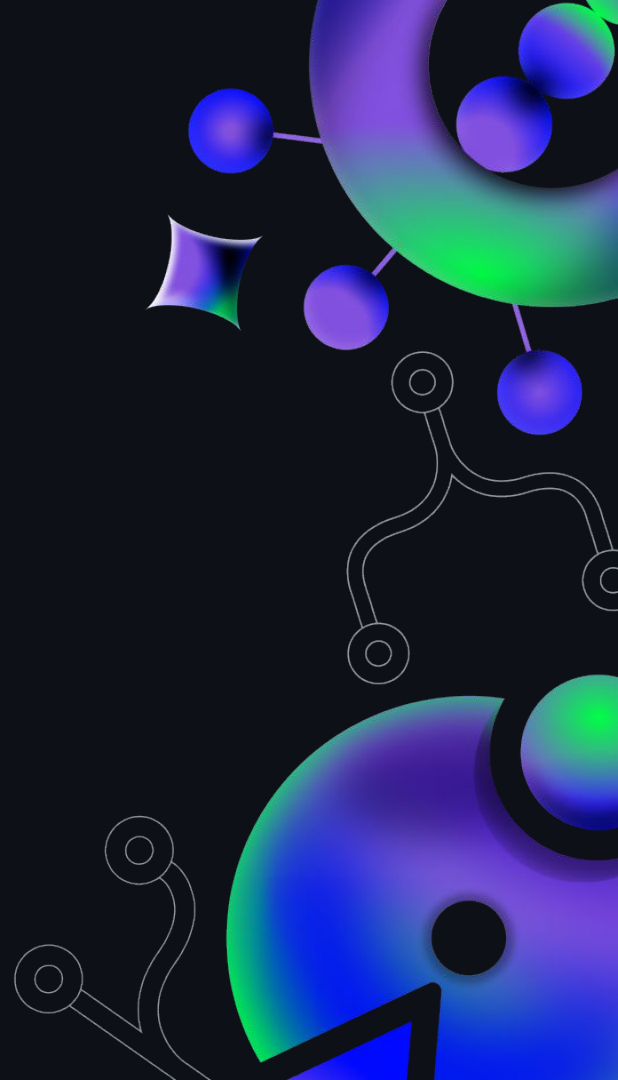
反復的な作業

存在時間が長い歴史的なリポジトリ

知識の発見・共有・コラボレーション

不慣れなコードベースでの作業

コード再利用





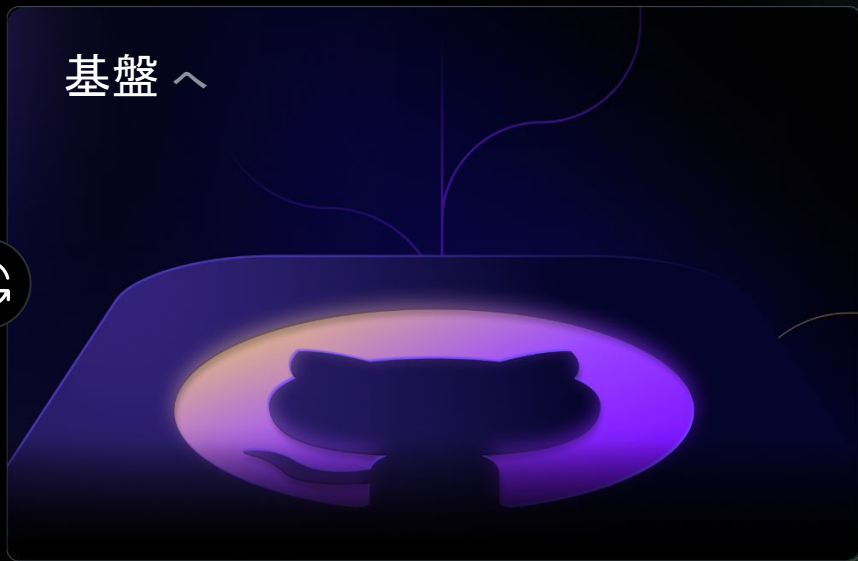
# AI駆動なソフトウェア開発向けの GitHub Copilot Enterprise

IDE, CLI,  
Mobile

から



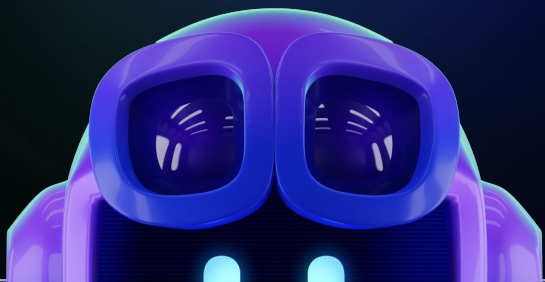
基盤 ^



GA: 一般提供

# Copilot Enterprise

コードとナレッジベースから  
ドメインの専門知識を  
Copilot Chatにもたらす



将来

# Custom models

SDK、API、および  
あまり普及していない言語に  
関する知識を Copilot の  
コード補完モデルに追加

+ Add-on

2025年初期

# Plugins

お好きなサードパーティのツール、  
データベース、サービスをCopilotに  
統合します



# Copilot を自分の組織内にある リポジトリとナレッジベースに特化



## 発見

この部分のコードはすでにどこかにありますか？

細かい手順で説明してください



## 解説

このコードは何をしていますか？

なぜこれが必要ですか？



## 解析

どこに注目すべきですか？

アンチパターンを使っている部分がありますか？





 GitHub Copilot

# 新たな ユースケースで より大きな物を

Copilot Enterprise



## 価値の実現までの時間短縮

- 開発ベロシティ
- 要件収集やプロジェクト計画
- InnerSource

## 運用効率の改善

- コード品質やベストプラクティスの遵守
- アプリケーションのモダナイズ
- 技術的負債の評価とパフォーマンス最適化

## より効果的な人材流出防止

- 同僚の支援とコラボレーション
- アップスキリングと学び
- メンタリングとコーチング



## Copilot Business

コーディング環境内の Copilot

\$**19** per user/month

- コード補完
- IDE と Mobile 内のチャット
- CLI 補助
- 脆弱性フィルター
- コード引用検知
- 公開コードフィルター
- 第三者による訴訟に対する補償
- エンタープライズレベルのプライバシー、ポリシー、管理制御

## Copilot Enterprise

組織のソフトウェア開発ライフサイクル全体においてパーソナライズされた Copilot

\* 要GitHub Enterprise Cloud

\$**39** per user/month

- **Copilot Businessの全機能に加えて:**
- コードベースに特化したチャット体験 \*
- ドキュメント検索や要約
- Pull request 要約
- コードレビュー
- ファインチューンモデル \*\*

\*リポジトリをインデックス化し、RAGにて文脈向上

\*\*Copilot Enterpriseのお客様向けのアドオン機能  
リリース日未定

## Copilot Individual

個人開発者やフリーランサー向けのコード補完やチャット機能等の Copilot

\$**10** per month/\$100 per year

認証済みの学生、教師、そして人気の OSSプロジェクトメンテナ向けには無償で提供



Resources    GitHub を使用する理由    トピックス    学ぶ    イベントとウェビナー    🔍    Enterprise トライアル    営業担当に問い合わせる

# GitHub Copilot Trust Center

GitHub は、GitHub Copilot を開発してイテレーションを行うときに、セキュリティ、プライバシー、コンプライアンス、および透明性を最優先することによって、開発者と組織がその可能性を最大限に引き出すことを可能にします。

<https://resources.github.com/ja/copilot-trust-center/>

# Appendix

プロンプト  
クラフティング



**旅行会社と一緒に海外  
への旅行を計画して  
いると仮定...**

下記の情報が無い限り適切な計画を作成することができないはずです:

- 予算
- 起点
- 旅行先
- 出発日・帰国日
- 直行便・乗り換え便の希望等



**プロンプトクラフティング:** プログラム (例えばGitHub Copilotの裏にある大規模言語モデル) に対し自然言語による明確な指示を与えるプロセス





## Copilotは何を プロンプトとして 認識しているか？

- 質問、文書、またはCopilotにて実現してほしい説明文
- ファイル内のコメントやCopilot Chat に対する入力値の形で受理



**Copilot は  
確率論的であり、  
決定論的ではない**

Copilotの裏にある大規模言語モデルは確率論に基づいたエンジンになるため、同じプロンプトに対して毎回同じ結果が返ってこない可能性があります

# 効果的なCopilotプロンプトの原則

## 文脈

Copilotに要求する  
タスクをより  
理解されるために  
プロンプトに  
含めている情報

## 意思


プロンプトを作成  
する際に自分が  
持っている特定の  
目標や目的

## 鮮明

曖昧でなく  
理解しやすい

## 具体的

プロンプト内の  
詳細や精度の  
レベル



# Copilotを助ける コード内のパターン

## 変数名

自分の意思が伝わるように  
分かりやすい変数名を利用

```
total_attendees = 5
```

## メソッドシグネチャー

明確なパラメータ名や型を  
メソッドシグネチャーで定義

```
calculateAverage(int  
num)
```

## 命名規則

変数名や関数名を  
規則的につける

例: 変数名では必ず  
camelCase (キャメルケース)を  
利用

# リファクタリング時はChatに訊く！

Before

```
function getZoomAttendees() {  
  var x = [];  
  var list = document.getElementsByClassName("participants-item__display-name");  
  for (var i = 0; i < list.length; i++) {  
    x.push(list[i].innerText);  
  }  
  return x;  
}
```

After



zack-evans

refactor variable names to be more descriptive in what they represent



GitHub Copilot

Sure, here's a suggestion for refactoring the variable names in the `getZoomAttendees` function:

```
function getZoomAttendees() {  
  var attendees = [];  
  var attendeeElements = document.getElementsByClassName;  
  for (var i = 0; i < attendeeElements.length; i++) {  
    attendees.push(attendeeElements[i].innerText);  
  }  
  return attendees;  
}
```

I renamed `x` to `attendees` to make it clear what the

これらの原則は  
プロンプト作成に  
どう関連しているか？

## 入力・出力の型

入力値・出力値の型を  
明確に記載

「入力値として文字列型の配列  
を使い、回文である場合は true  
のブーリアン型を返す関数を作  
成して」

## エラー処理

エラー処理が必要な場面を  
明確に説明


「整数型の入力値が空の場合は  
exit処理を行い、入力値が整数  
ではない場合はエラーを発生さ  
せて」

## 制御構造

制御構造を説明

「整数の配列を入力値としてある  
場合、最初の素数を返す while  
文を作成して。素数が配列にな  
い場合はそのままループから  
exitするように処理して」





**Copilot**の提案精度を  
最大限にするための  
アドバイス

# ファイルの冒頭に 要件のハイレベルな 説明文を含める

明確な手順をその後に記載すると  
より効果的である

```
JS weather.js > ...
1 //Create a class that will retrieve weather metrics from the US National
2 //Include the ability to retrieve the current weather, 7 day forecast,
3 //API calls should be handled asynchronously
4 //1.Import axios to make API calls
5 //2.Import the conversion.js file and use the functions to convert the
6 //3.Store API endpoint as a constant
7 //4.Functions should take a zip code as a parameter where applicable
8 //5.Handle exceptions and return appropriate error codes with user friendly
9 //6.Functions should return the data in a format that can be used by the
10
11 const axios = require('axios');
12 const conversion = require('./conversion');
13
14 const BASE_URL = 'https://api.weather.gov/points/';
15
16 // define Weather class
17 class Weather {
18   constructor(zip) {
19     this.zip = zip;
20   }
21 }
```

期待している内容を  
明確にするため  
プロンプトに  
いくつかの例を含める

```
calculations.py > ...
1  """
2  Create a Python function, `sum_array_values`, that finds
3  and returns the sum of integers in an array.
4
5  1.Verify if the array is empty and exit if it is.
6  2.Throw error if the array contains non-integer values.
7  3.Return the sum of the array values if the array is valid.
8
9  Please consider the following example data:
10 Example 1: [5, 10, 15, 20, 8]
11 Example 2: [2, 4, 6, 1, 9, 12]
12 The generated code should work correctly for arrays like above
13 """
14 #define sum_array_values function
15 def sum_array_values(array):
16     # 1.Verify if the array is empty and exit if it is.
17     if len(array) == 0:
18         return 0
19     # 2.Throw error if the array contains non-integer values.
20     for i in array:
21         if type(i) != int:
22             raise TypeError("The array contains non-integer values.")
23     # 3.Return the sum of the array values if the array is valid.
24     return sum(array)
```

操作中のファイルの  
要件に関連する  
プロジェクト内の  
別ファイルを開いたままにする

```
J BookDatabaseImpl.java • J BookDatabase.java J BookService.java J BookServiceException.java
src > main > java > com > github > demo > service > J BookDatabaseImpl.java > BookDatabaseImpl > getBook
146 @Override
147 public List<Book> getBooksByAuthor(String author) throws BookServiceException {
148     // 1.create list of type Books
149     // 2.create Prepared statement and query to get books by Author
150     // 3.iterate through results and add to list
151     // 4.close connection if prepared statement is not null
152     // 5.return list
153
154     List<Book> books = new ArrayList<Book>();
155
156     if (!isValid()) {
157         throw new BookServiceException(message:"Database connection is not valid, check
158     }
159
160     PreparedStatement ps = null;
161
162     try {
163         ps = connection.prepareStatement(sql:"SELECT * FROM books WHERE author LIKE ?")
164         ps.setString(parameterIndex:1, "%" + author + "%");
165
166         ResultSet results = ps.executeQuery();
167
168         while (results.next()) {
169             Book book = new Book(
170                 results.getString(columnLabel:"author"),
171                 results.getString(columnLabel:"title"),
172                 results.getString(columnLabel:"image"));
173             books.add(book);
174         }
175     } catch (SQLException e) {
176         logger.error("Failed while searching for {}", author);
177         throw new BookServiceException(e);
```

最初に成功  
しなかったら  
もう一度試す!

## 最初のプロンプト

```
1 //Write a JavaScript function that finds
2 //the maximum value in an array.
3 function max(array) {
4     return Math.max.apply(null, array);
5 }
```

## 二回目のプロンプト

```
1 ✓ //Create a JavaScript function, `findMax`, that takes an array of
2 //and returns the string with the maximum length.
3 //Ensure that the function works correctly for arrays that are em
4 //as well as arrays that contain both strings and numbers.
5 ✓ function findMax(array) {
6     var max = array[0].length;
7     array.map(v => max = Math.max(max, v.length));
8     result = array.filter(v => v.length == max);
9     return result;
10 }
```



**Chat**を活用して  
**Copilot**体験を  
より一層高める

# GitHub Copilot Chat: 文脈を認識した ChatGPTらしい 会話

Copilotのコード補完機能と同様、Copilot Chatは文脈を認識し  
対話形式で提案を表示

The screenshot displays the GitHub Copilot Chat interface within an IDE. The chat window is titled "CHAT: GITHUB COPILOT" and shows a conversation between a user and the Copilot agent.

**User:** lindseybocatto  
Explain this code

**GitHub Copilot:**  
The code above is a Java class called `Book` which is part of a package called `com.github.demo.model`. The purpose of this class is to model a book, and it has three instance variables: `title`, `author`, and `cover`. The `title` and `author` variables are of type `String`, while the `cover` variable is also of type `String`, but it represents the path to the book's cover image.

The class has three constructors: a default constructor, a constructor that takes an author and a title as parameters, and a constructor that takes an author, a title, and a cover as parameters. The default constructor does not take any parameters and does not initialize any instance variables. The second constructor takes an author and a title as parameters and initializes the `author` and `title` instance variables. The third constructor takes an author, a title, and a cover as

The background shows a code editor with the following Java code:

```
src > main > java > com > github >
1 package com.github.de
2
3 /**
4  * Model class for bo
5  */
6 public class Book {
7
8     private String t
9
10    private String au
11
12    private String co
13
14    public Book() {
15
16    }
17
18    public Book(Strin
19        this.author =
20        this.title =
21    }
22
23    public Book(Strin
24        this.author =
25        this.title =
```

# プロンプト 作成での 4原則はChat にも適用

文脈

意思

鮮明

具体的



# スラッシュコマンド

**/explain**

選択したコードの  
仕組みを解説

**/tests**

選択したコードに  
対する単体テスト  
を生成

**/fix**

選択したコード内  
の問題に対する修  
正案を提示

**/workspace**

現ワークスペースに  
間する質問を  
訊く

**/help**

Copilot Chatの  
使い方を解説

CHAT: GITHUB COPILOT



**lindseybocatto**

`/tests`



**GitHub Copilot**

Sure, here's an example set of unit tests for the BookService

```
package com.github.demo.service;

import com.github.demo.model.Book;
import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import java.util.List;

import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertNotEquals;

/**
 * Unit test for BookService
 */
public class BookServiceTest {

    private BookService bookService;
```

# Chatで新しいプロジェクトを始める

- `/new` や `/createNotebook` で新しいワークスペースやノートブックを作成
- プロンプト作成の4原則に従って新しい機能の実装方法をChatから提案をもらう
- 繰り返して試す! Chatから望ましい結果を得られない場合、Chatがどのように提案を修正できるかをフィードバックとして与える
- `/tests` を使って単体テストを生成
- Chatを使って READMEファイルやドキュメント生成を行う

# 既存の コードベースで Chatを使う

- わからないコードがあれば `/explain` でCopilotに解説をお願いする
- Chatを使って **脆弱性の修復** や **バグ修正** に関するアドバイスをもらう
- **関連ファイル**をIDE内で開き、無関係のファイルを閉じる
- **テストの提案**をChatに訊いて単体テストのカバー範囲を強化する
- Chatを使って**コードのリファクタリング**や別の開発言語への**変換**を行う



Thank you