

2024年9月最新版

教員のための ChatGPT 実践

-教育現場での活用 -



畠山 大有 | Daiyu Hatakeyama

佐賀大学 非常勤講師

Architect && Software Engineer && Applied Data Scientist (目指している)@Microsoft Japan

-  /dahatake
-  @dahatake
-  /in/dahatake
-  /dahatake
-  /dahatake
-  /dahatake

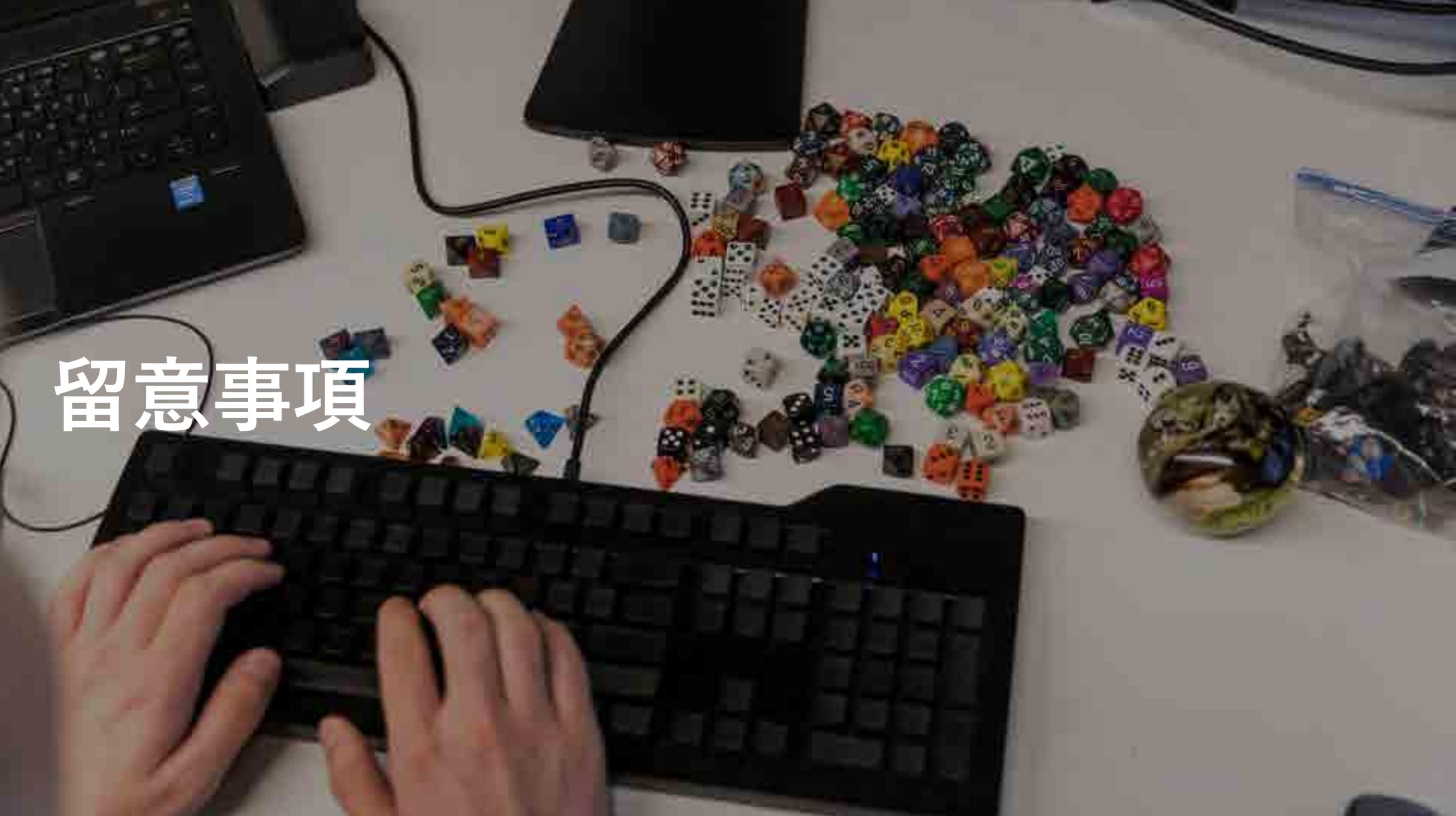


毎日

Prompt

していますか？

留意事項



留意事項

- 個人的な見解 😊 -

- **機密**情報の入力

- サービスによっては、学習に使われる可能性がある



汎化・抽象化

- 出力文字列の**妥当性**

- 何の根拠もなく正しいと判断すること
- 知識としての情報を表示しているわけではない



1次情報の確認
| 学び続ける

- **知財**の扱い

- サービスによって利用許諾や出力データの知財などが異なる



自社の法務担当に
相談

Microsoft のAIの保護

- お客様のデータは お客様のもの
- お客様のデータはAIモデルのファインチューニングには利用されません
- お客様のデータとAIモデルは全ての段階で保護されます
- 新たな Copilot Copyright Commitment

生徒に、以下も提出してもらおう

- レポート作成に使用したプロンプト
- 生成AIに作成させたレポート
- 生成AIに作成させたレポートを学生自身が添削した修正レポート
- どのような理由・観点・論点で添削したのかのレポート

AIサブフィールドPBLでのレポート依頼例

The screenshot shows a web interface titled "AIサブフィールドPBL". It contains a form with several sections:

- 1. 目的(何のために?)**: A text input field with the placeholder "目的を入力してください".
- 2. 作成した理由・視点・論点(Prompt内容)**: A text input field with the placeholder "理由・視点・論点を入力してください".
- 3. 作成した理由・視点・論点(Prompt内容)を説明する**: A text input field with the placeholder "理由・視点・論点を説明してください".

At the bottom of the form, there is a blue button labeled "実行".

レポート作成に使用した
Prompt

どのような理由・観点・
論点で添削したのか

教員自身の学びにもなる。
「教えて頂く機会」という謙虚さも忘れずに



Prompt を業務で使うための
コツ

私のお勧めの Prompt の歩き方

[ポイント] そのタスクで、何をどのように
お願いしたいのかを話してみる

Step.3
都度、自分に必要なPromptを作る

その場で試行錯誤しながら
Prompt を作成できる

[ポイント] ネットで検索する。あるいは、ChatGPTに
「xxxのタスクを行うプロンプトを作って」と依頼してみる

Step.2
特定業務シナリオの
Promptを試す

[ポイント] 自動化したい場合。Promptじゃなくて、
プログラムを書いた方がいい場合もある

自分の業務に活かす道を知る

[ポイント] Microsoft 365 Copilot には
Copilot Lab があって、超絶便利

Step.1
用意された
Promptをそのまま試す

テンプレート

Prompt で出来る事を理解する

Microsoft 365 Copilot Lab

<https://copilot.cloud.microsoft/ja-JP/prompts>

- Copilot for Microsoft 365 の使い方を知る
- Prompt のテンプレート集



Microsoft 365 Copilot Lab

<https://copilot.cloud.microsoft/ja-JP/prompts>

The image displays a grid of 24 prompt cards from the Microsoft 365 Copilot Lab. Each card contains a specific prompt in Japanese, a small icon, and a 'Copilot' logo. The prompts are organized into four columns and six rows. The first column on the left is partially obscured by a sidebar with additional prompts.

新機能 さんからの最新情報メール、チャット、ファイルを表示する。	要点を整理する この要点を整理してください	バケットの作成 特定の目的のためのバケットを作成します	どのようなオブジェクト 識別されているオブジェクトと場所をまとめて	目標を編集する この目標を編集してください	会議の要約 会議の内容を要約する	アイデアのリスト作成 ディスカッションされた5つのアイデアのリストを作成する。	効果的な書き出しにする この文章の興味を引く導入の段落を書いてください
ToDo リストを作成する Fabrikam クライアントのToDo リストを作成します	ミッションステートメントを作成する 新しいマーケティングチームのミッションステートメントを作成する。	手順の追加 サブタスクをステップに追加して、チームに共有してください。	どのようなチーム チームは決定したセンサスを選択します	ヒントとコツ このヒントでヒントをいくつか提供してください。	旅程を作成する 3日間の旅程を作成してください。	重要なものを追跡する 過去48時間の上記のメッセージを要約してください。	バケットの管理 実際のタスクをエン지니어リングバケットに移動する
議題を作成する (World Wide Import)の会議メモに基づいて追加する。	目標を作る ビジネスを成長させるためのいくつかの目標を作ります	アイデアを生成する 特定の目標を達成するためのアイデアを生成する方法について、新製品を開発する方法をスライドに盛り込んだプレゼンテーションを作成する。	作業項目を作成する この作業項目を作成してください	考えを整理する このアイデアを整理します	誰が誰か このプロジェクトに関する連絡先は誰ですか	明確さを高める さまざまな意見のリストを作成し、それぞれについて明確にするための質問をどのように提案する。	パンフレットを作成する 特定のサービスに関するパンフレットを作成する
	メールに返信する このメールに返信してください	アイデアを比較する ディスカッションされた各アイデアについて、長所と短所を特定し、次のステップの優先順位を決定するためのアイデア、長所、短所、実行	リストを作成する このリストを作成してください	データの強調表示 このデータの重要な部分を強調表示してください	急いで最新状況を確認する 受信トレイの最新状況を確認してください	ぜひお試しください 今回自分がメンションされた Teams のメッセージを印刷します。	優先順位を整理する このプランで最も優先度の高い作業は何ですか

困ったときの Microsoft Copilot での鉄板 Prompt

<https://copilot.microsoft.com/>



製品の使い方:

Power BIのPower Queryでのデータ取得の処理が凄く遅いです。
考えられる原因の詳細な説明と、その測定方法と、
詳細な解決策をリストアップしてください。

データを見た後での推測の補助:

海外から佐賀県を訪れる観光客がとっても少ないです。
考えられる原因の詳細な説明と、その測定方法と、
詳細な解決策をリストアップしてください。

ネット上の情報が必ず正しいとは限らないため、
可能な限り**1次情報** (元となったWebサイト) を確認する事

そのまま意外といろいろと出来る

ChatGPT

ChatGPT 4o

日本の教員が仕事をしていく中で、大変だと思っていることは何ですか？その理由と具体的な理由をリストアップしてください。

日本の教員が仕事をしていく中で大変だと思っていることには、以下のような理由があります。それぞれの具体的な理由も併せて示します。

1. 長時間労働
 - 理由: 授業準備、部活動の指導、保護者対応、学校行事の準備などで、定時以降の実務が多くなるため。
2. 授業準備の負担
 - 理由: 授業の内容を計画し、教材を作成するのに多くの時間がかかる。また、新しい教育カリキュラムへの対応も必要。
3. 生徒指導
 - 理由: 問題行動を起こす生徒への対応が難しく、個別の支援が必要な場合があるため。また、いじめや不登校などの問題にも対応しなければならない。
4. 部活動の指導
 - 理由: 平日放課後や週末にも指導が必要で、プライベートな時間が削られる。また、専門的な知識や技術が求められることも多い。
5. 保護者対応
 - 理由: 保護者からの問い合わせや相談に対応する必要があり、中には過度な要求をする保護者もいるため。

6. 書類作業
 - 理由: 進路指導計画や成績評価表、授業記録など、多くの書類を作成しなければならず、その量が膨大であるため。
7. 教育改革の対応
 - 理由: 新しい教育方針やカリキュラムに対応するための研修や準備が必要で、その都度、既存の業務に加えて新しい業務が増えるため。
8. 同僚との連携
 - 理由: 学年主任や教科主任との連携が必要で、意思疎通の不足や意見の相違がストレスになることがあるため。
9. ICTの活用
 - 理由: デジタル教材やオンライン授業の準備、実施に慣れていない場合、技術的な問題やトラブル対応が負担になるため。
10. 精神的ストレス
 - 理由: 生徒や保護者、同僚との人間関係の問題や、高い成果を求められるプレッシャーが精神的な負担となるため。

これらの問題は、教育の質を保ちつつ教員の働きやすさを向上させるための対策が求められています。

普段から、この作業をしている方は、出力結果の妥当性はわかりますよね? 😊

そのまま意外といろいろと出来る

Microsoft Copilot (Bing Chat)

自分
日本の教員が仕事を休んでいる中で、大変だと感じていることは何ですか？その項目と具体的な理由をリストアップしてください。

Copilot
✓ 回答をお読みしています。
日本の教員が仕事を大変だと感じていることには以下のような理由があります

1. **長時間労働** 日本の教員はOECD加盟国の中で最も長い勤務時間を有しており、平均して週に52.9時間を勤務しています。これは加盟国平均の28.3時間を大きく上回っています。
2. **課外活動の指導** 日本の教員は課外活動に参加国平均の3倍以上の時間を費やしており、これが大きな負担となっています。
3. **一般事務業務** 学校運営に関わる一般事務業務にも多くの時間を割いており、これが教育活動から時間を奪っている一因となっています。
4. **保護者や地域との関係** 保護者や地域からの希望や苦情への対応も教員にとっては大きなストレス源です。
5. **教育現場の改善** 教員自身の自己効力感の低下や、教育現場の改善が求められている点も、教員が悩む主要な理由です。

これらの項目は、教員が子どもたちの教育に集中することを妨げ、また教員自身のワークライフバランスに悪影響を与えています。教育現場の改善には、これらの課題に対処することが不可欠です。

自分
具体的な対策について、最近どんな具体的な対策が講じられて、実行されてきましたか？そのうち多くはいつと前より多くはいつなかった事例と、それらの感想をリストアップしてください。

Copilot
✓ 回答をお読みしています。
日本の教員の長時間労働に対する対策として、以下のような取り組みが行われてきました。

うまくいった事例

1. **短時間の改正** 2019年に公立学校教員の残業時間に対し「原則月48時間、年560時間」という上限が設けられました。これにより、過剰な残業が抑制し、家内や私生活が減少しました。
2. **課外活動の削減** 放課後の習字や読書などが削減され、地域の人材に委託する「課外活動の地域移行」が実施されており、これにより教員の負担が軽減されることが見られています。

うまくいかなかった事例

1. **持ち帰り仕事の増加** 学校の残業削減は減少したものの、持ち帰り仕事は増加しており、教員の勤務時間削減にはつながっていません。
2. **教職調整費** 後進の代わりにお支払われる給与月給の削減分の「教職調整費」は、実際の残業時間に経過して給与ではなく、教員の退職金削減の根本的な解決には繋がっていないとされています。

これらの事例から、教員の長時間労働に対する対策は、過剰な残業の削減や地域化などの措置による負担の軽減が一定の効果を出しているものの、家内の労働時間の削減や、教員の負担の軽減にはまだ課題が残っていることがわかります。教員の働き方改革は、引き続き多角的なアプローチと持続的な取り組みが求められています。

普段から、この作業をしている方は、出力結果の妥当性はわかりますよね? 😊

実は...

2023年だとイマイチだったものもあります 😊

一度動かなかったからと言って
結論付けてしまうのは、勿体ないかも

仕事で ChatGPT を使うためのポイント

変換

抽出

参照データ

その場 (In-Context) で学習の例

日本の内閣総理大臣は誰ですか?



申し訳ありませんが、私の知識は2021年9月までで更新されておらず、その後の具体的な人物や事象については情報を提供できません。最新の情報を知るためには、信頼できるニュースソースやウェブサイトをご確認ください。

その場 (In-Context) で学習の例

日本の内閣総理大臣は誰ですか? 以下の情報を基にして教えてください。

情報 ###

内閣総理大臣（ないかくそうりだいじん、英: Prime Minister[1]）は、日本の内閣の首長たる国務大臣[2]。文民である国会議員が就任し、その地位及び権限は日本国憲法や内閣法などに規定されている[3]。

現任は、第101代岸田文雄（在任: 2021年〈令和3年〉11月10日 - ）。歴代の内閣総理大臣は内閣総理大臣の一覧を参照。

###

instruction

context

Instruction と context は
や "" を使って分離

あなたが提供した情報に基づくと、日本の内閣総理大臣は岸田文雄です。彼は2021年11月10日からその職にあります。ただし、これは私の最新の知識更新が2021年9月までであること、そしてあなたが提供した情報が現在（2023年6月22日）までに更新されていない可能性があることに注意してください。そのため、最新の情報を確認するためには信頼性の高いニュースソースや公式ウェブサイトをチェックすることをお勧めします。

その場 (In-Context) で学習の例

日本の内閣総理大臣は誰ですか? 以下の情報を基にして教えてください。

情報:###

内閣総理大臣（ないかくそうりだいじん、英: Prime Minister[1]）は、日本の内閣の首長たる国務大臣[2]。文民である国会議員が就任し、その地位及び権限は日本国憲法や内閣法などに規定されている[3]。

どこから、**データ**を取ってきて

埋めてくれればいい!

現任は、第101代岸田文雄（在任: 2021年11月10日 - ）。歴代の内閣総理大臣は内閣総理大臣の一覧を参照。

###

instruction

context

あなたが提供した情報に基づくと、日本の内閣総理大臣は岸田文雄です。彼は2021年11月10日からその職にあります。ただし、これは私の最新の知識更新が2021年9月までであること、そしてあなたが提供した情報が現在（2023年6月22日）までに更新されていない可能性があることに注意してください。そのため、最新の情報を確認するためには信頼性の高いニュースソースや公式ウェブサイトをチェックすることをお勧めします。

仕事で ChatGPT を使うためのポイント

変換

抽出

参照データ

丁寧な謝罪文章の添削

GPT-4

以下の文章を内容は変更せずに、最大限考えうる謝罪のメールの文章にしてください。
何を添削したのかの解説を(改善点)にてリストアップして説明をしてください。

文章 ##

島山さん、先日は飛行機の遅延があったとはいえ、15分も会議に遅れてしまいました。今後は気を付けます
###

改訂後のメール文面:

件名: 遅刻に関するお詫びと今後の対応について

島山様

お忙しい中をおいてのご迷惑、誠にありがとうございます。先日の会議におきまして、私の到着が予定時刻を15分も過ぎてしまったことに関しましては、心よりお詫び申し上げます。

飛行機の遅延という不可抗力ではございましたが、それに断ることなく、より早めのフライトを選択するなど、さまざまなリスクに備えた行動が求められたことを痛感しております。私の遅刻が島山様をはじめとする会議参加者の皆様のお忙しい時間を奪い、また会議の流れに影響を及ぼしたことは、許される行為ではございません。

今後はこのような事態を未然に防ぐため、時間管理を徹底し、余裕をもって行動することを誓います。また、何らかの不安の事態が発生した場合には、可能な限り速やかにご連絡し、適切な対応を取らせていただく所存でございます。

この度は本当に申し訳ございませんでした。島山様のご理解とご寛容を賜れますよう、お願い申し上げます。

敬具

(改善点)

- 1 敬称の追加: 「島山さん」を「島山様」に変更して、より丁寧な印象を与えました。
- 2 件名の設定: メールに明確な件名を設定することで、メールの目的と重要性を受け取り手に即座に伝えます。
- 3 お詫びの深化: 単に「すみませんでした」という軽い表現から、「心よりお詫び申し上げます」と改めて、謝罪の诚意を増しました。
- 4 責任の認識: 不可抗力に言及しつつも、それを言い訳にしない姿勢を示しました。
- 5 影響の認識: 遅刻が他者に及ぼす影響を明確にし、遅刻が不適切な行為であることを認めます。
- 6 今後の改善策: 単に「気を付けます」という曖昧な表現ではなく、具体的な行動の改善策を提示しました。
- 7 万一の事態に対する対応策: 今後同様の状況が発生した際の対応方法も述べ、信頼回復に努めています。
- 8 結びの言葉: 改めて謝罪を表明し、相手の理解と寛容を求める丁寧な表現を用いました。
- 9 署名の形式: 最後に「敬具」という形式を用いて、書式を整えました。

英文の向上!

GPT-4

指示

次の{文章}の内容を変更せずに、CEFRのC1レベルに添削してください。
添削した文章は英語で作成してください。なぜ、その改善をしたのかの理由は日本語で作成してください。

{文章}: ###

I found the technical issue at our cloud services. The almost of behavior of issue looks like come from storage layer. I guess disk io is near peak. How can I check this?

###



Revised Text:

I have identified a technical issue with our cloud services. It appears that the majority of this issue's behaviors originate from the storage layer. I suspect the disk I/O is nearing its peak. How might I verify this?

Revision Reasons (in Japanese):

1. 「I found」を「I have identified」に変更: 「I have identified」はC1レベルに適しており、専門的なコンテキストでの発見をより正確に表現します。
2. 「The almost of behavior」を「the majority of this issue's behaviors」に変更: 「The majority of this issue's behaviors」はより自然で、文脈に適した表現です。「この問題のふるまいの大部分」という意味で、より明確です。
3. 「looks like come from」を「originate from」に変更: 「Originate from」は「～から発生する」という意味で、より洗練されており、C1レベルの英語に適しています。
4. 「I guess」を「I suspect」に変更: 専門的な文脈において、「I suspect」という表現は、「I guess」よりも適切で、推測をより確信に近いニュアンスで表現します。
5. 「How can I check this?」を「How might I verify this?」に変更: 「How might I verify this?」は「このことをどのように確認すればいいですか?」という意味で、よりフォーマルで適切な表現です。「verify」は「check」よりも正確性や確実性のニュアンスが強いため、技術的な問題を確認する文脈に適しています。

あらゆる変換を試みしてくれる

カジュアルな日本語

畠山さん。先日は飛行機の遅延があったとはいえ、15分も会議に遅れてすみませんでした。今後は気を付けます

フォーマルに

ビジネスで通用する日本語

畠山様
まず初めに、先日の会議に遅刻したことににつきまして、心より深くお詫び申し上げます。私の不手際が原因で、貴重なお時間を無駄にしてしまい、大変申し訳ございませんでした。
...

英語で

ビジネスで通用する英語

Dear Mr. Hatakeyama,
I am writing to express my deepest apologies for arriving 15 minutes late to our meeting the other day. Despite the fact that my delay was due to an unforeseen and regrettable plane delay, I fully recognize that this is no excuse for disrupting our scheduled meeting time.

HTMLで

Web画面

```
<html lang="ja">
<head>
  <meta charset="UTF-8">
  <title>謝罪のメッセージ</title>
<style> ...
```

ドキュメント=モデル化・言語化=情報共有



ドキュメント



モデル



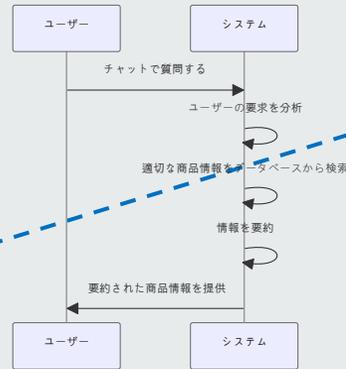
コード

ビジネス上の実現したい事

ユーザーが迷わないように、膨大な商品情報から、その都度のユーザーの知りたい事に合わせた情報を提供したい。

システムでの実現手段

- チャットで、都度ユーザーのやりたい事の言語化を手伝う
- そのやりたい事を、商品情報から要約した文章を作成する



sequenceDiagram
participant U as ユーザー
participant S as システム

- U->>S: チャットで質問する
- S->>S: ユーザーの要求を分析
- S->>S: 適切な商品情報をデータベースから検索
- S->>S: 情報を要約
- S->>U: 要約された商品情報を提供

注) Mermaid 記法

```
def analyze_question(question):  
    """ユーザーの質問を分析してキーワードを抽出する"""  
    # ここでは単純に質問をキーワードに分割する  
    keywords = question.split()  
    return keywords  
  
def search_database(keywords):  
    """キーワードに基づいてデータベースから商品情報を検索する"""  
    # 仮の商品情報データベース  
    product_database = {  
        "スマートフォン": "最新のスマートフォンには、高画質のカメラと長持ちするバッテリーがあります。",  
        "ヘッドホン": "多様なノイズキャンセリング機能を持つヘッドホンが市場にあります。",  
        "コーヒーメーカー": "自宅で手軽に美味しいコーヒーを楽しめるコーヒーメーカーが人
```

Business Engineering

Software Engineering

仕事で ChatGPT を使うためのポイント

変換

抽出

参照データ

メールから案件情報を抽出

あなた
指示: ###
以下の文章の各{項目}についてリストアップしてください。分からない事は「不明」としてください。日本語で答えてください。
###

{項目}: ###
- 概要
- 目的
- スコープ
- 成果物
- スケジュール
- コスト
- リスク
- 要求品質
- リソース
- アクションプラン
###

文章: ###
畠山 大有 <dahatake@hotmail.com>

Daiyu Hatakeyama

畠山さん、早速のご返答ありがとうございます。

はい、PCはお持ち込みでも大丈夫です。
機材確認のため、大変お手数なのですが、セッション登壇の30分前、つまり8:30には会場にお越しください。スタッフがご案内します。

大有

畠山 大有 <dahatake@hotmail.com>

大有さん、ありがとうございます。
一点教えてください。プレゼンテーション用のPCを持ち込みたいのですが、大丈夫でしょうか? デモをする際に自分のPCが必要なんです。

Kind Regards,

Daiyu Hatakeyama

Daiyu Hatakeyama

畠山さん、この度はお話をとお受けいただきありがとうございます。
以下がお願い事項となります。ご不明点などがありましたら仰ってください。

日程: 2024年7月15日
時間: 9:00-10:00
内容: 会議を削減するためのChatGPT利用方法
場所: 日本マイクロソフト品川オフィス
参加者: 一般社員・意思決定者

どうぞ、よろしく申し上げます。

大有
Contso Corporation
###

ChatGPT
以下のリストは、提供された文章に基づいています。

- 概要: 「会議を削減するためのChatGPT利用方法」についてのプレゼンテーション
- 目的: 会議削減のためのChatGPTの利用方法を伝える
- スコープ: 不明
- 成果物: プレゼンテーション (おそらくPowerPointなどの形式)
- スケジュール: 2024年7月15日、9:00-10:00
- コスト: 不明
- リスク: 不明
- 要求品質: 不明 (しかし、畠山さんは自分のPCを使用してデモを行う予定であるため、技術的な品質は重要と考えられる)
- リソース: 畠山さんのPC、日本マイクロソフト品川オフィスの会場
- アクションプラン: 畠山さんがセッション登壇の30分前 (8:30) に会場に到着し、スタッフの案内を受ける

詳細な情報は提供された文章には含まれていないため、いくつかの項目については「不明」となっています。



メールから案件情報を抽出

{項目}: ###

- 概要
 - 目的
 - スコープ
 - 成果物
 - スケジュール
 - コスト
 - リスク
 - 要求品質
 - リソース
 - アクションプラン
- ###

ご自身の会社・組織のアプリケーションの
入力画面の項目ともいえる😊

ChatGPT

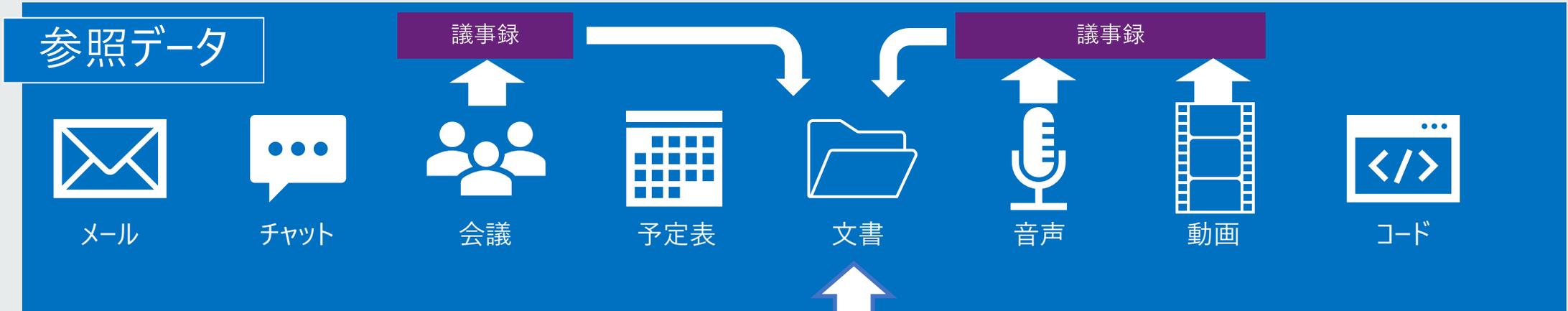
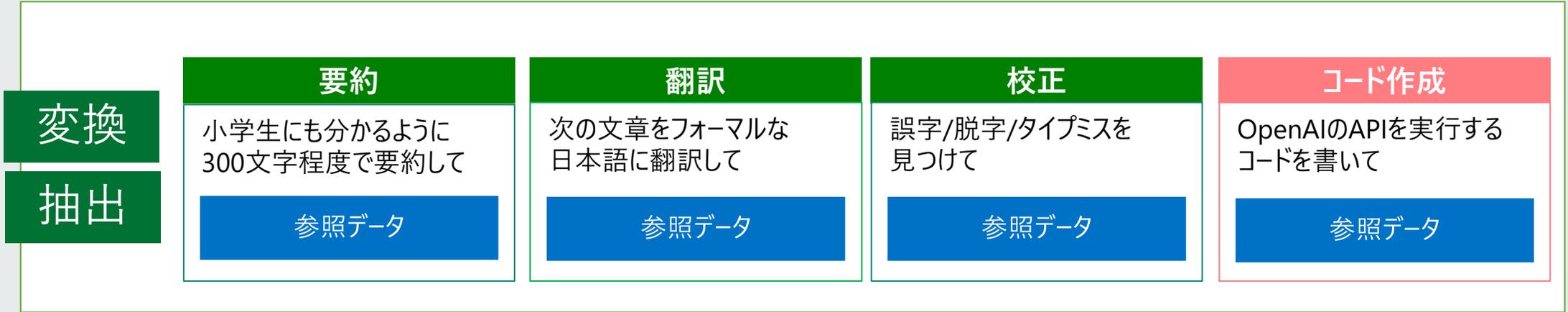
以下のリストは、提供された文章に基づいています。

- 概要: 「会議を削減するためのChatGPT利用方法」についてのプレゼンテーション
- 目的: 会議削減のためのChatGPTの利用方法を伝える
- スコープ: 不明
- 成果物: プレゼンテーション（おそらくPowerPointなどの形式）

データ にしておけば、 ChatGPT 君で何とか出来る



GPT-4o



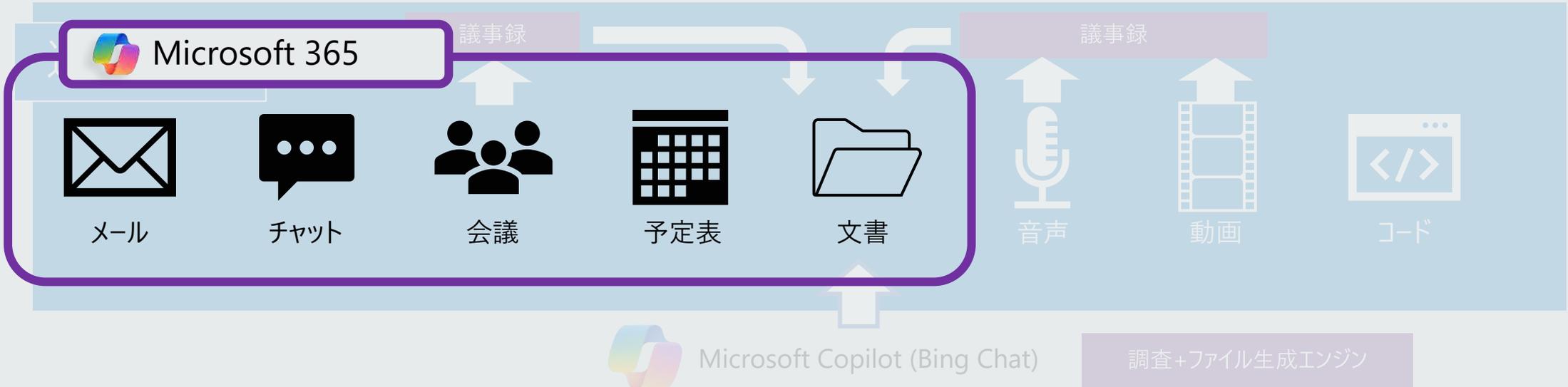
Microsoft Copilot (Bing Chat)

調査+ファイル生成エンジン

データがどこにあるかで どのアプリが使えるかを考えればいい 🧐

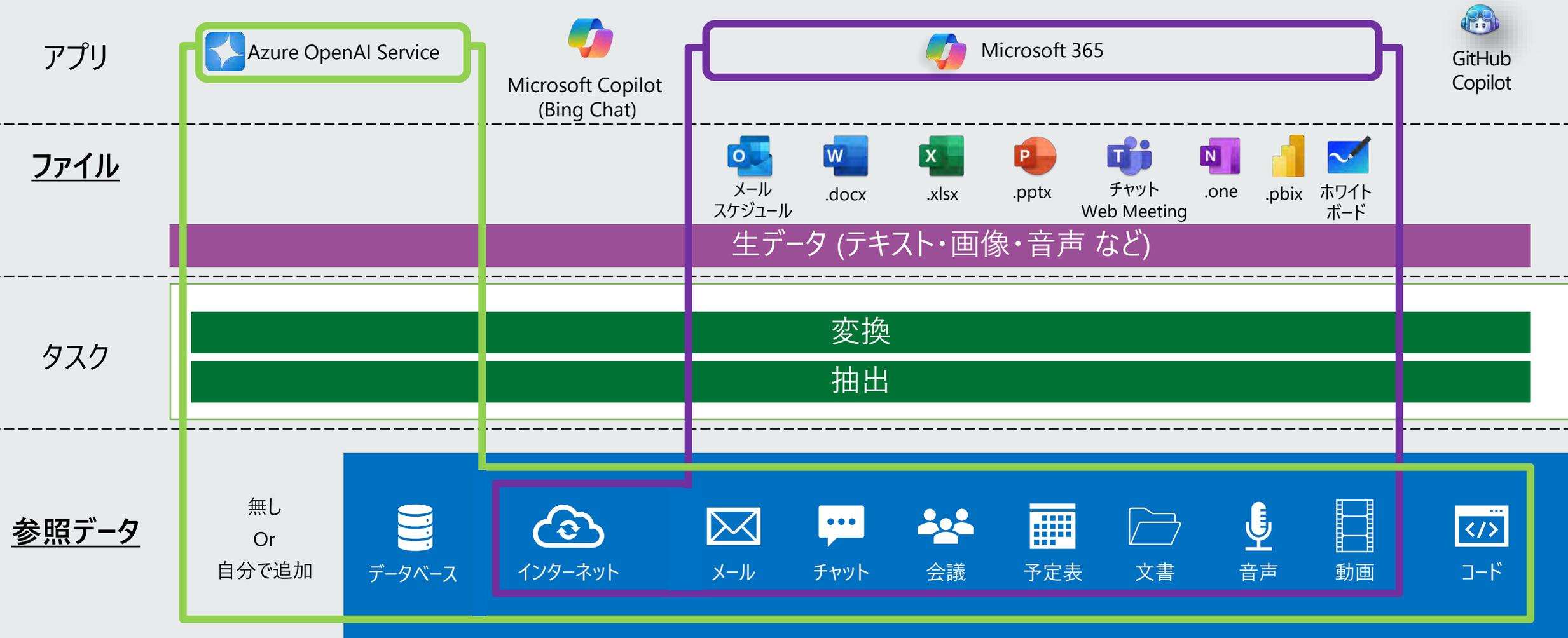


GPT-4o



一般的な Copilot 君の使い分け

最終的に何のファイルを作りたいのか?
どこにあるデータを勝手に探ってきて欲しいのか?





Prompt 作成の **コツ**

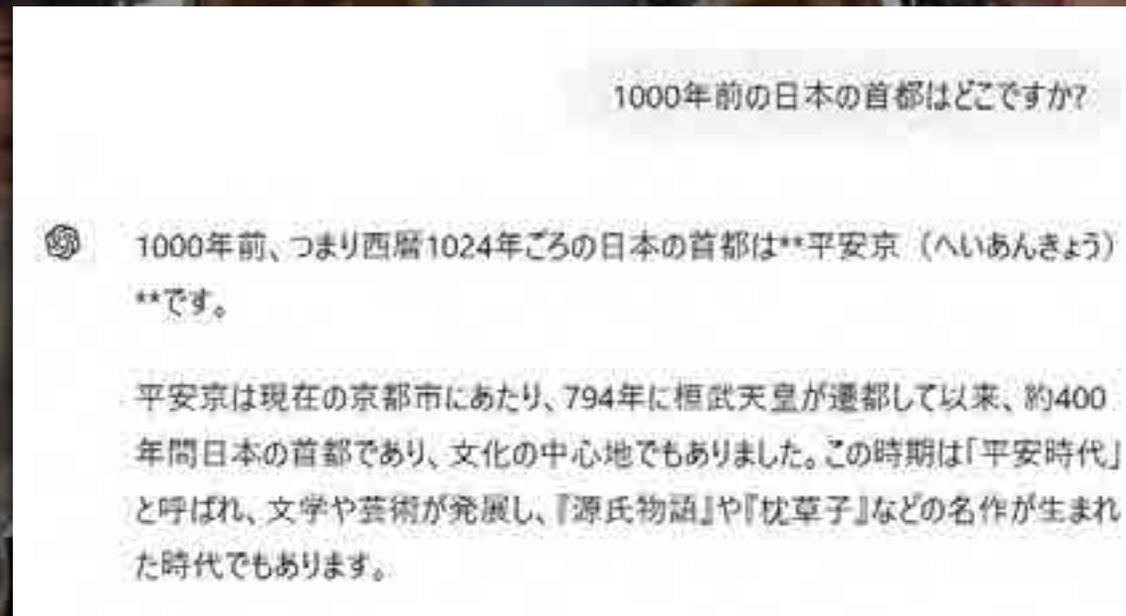
ドラフトを作ってくれる

生成AIに完成品を作らせようとして、Promptをし直すのは止めたほうがいい
Promptの改善は勿論大事 😊

言葉不足になりがち 😊



言葉不足になりがち 😊

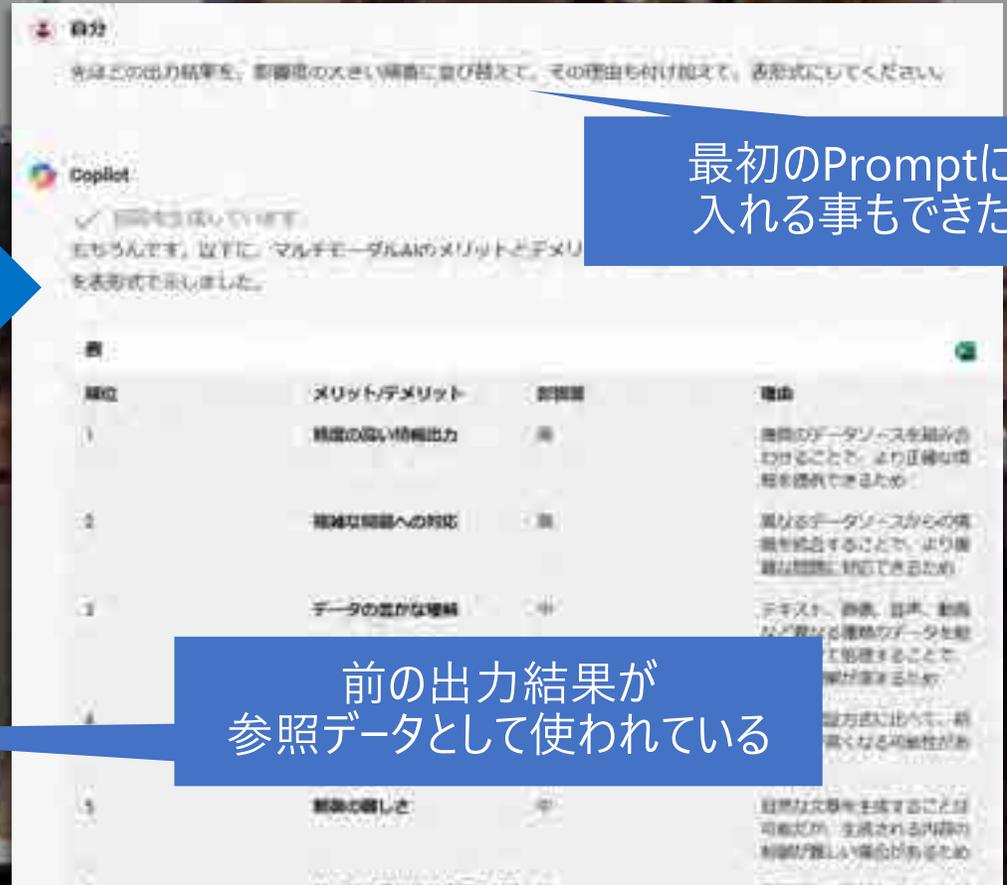


私自身、このノウハウを知っていますが。毎回出来ていません 😊

Chat (会話)は複数のターンでコトを進められる



一度のPromptで、欲しい回答が出てくるとは、まず無い 😊



最初のPromptに入れる事もできた

前の出力結果が参照データとして使われている

タスク遂行に必要な言語化を全て出来る人は **少ない** 😊
結果を見て 気づく事も多いですし

Prompt の構造

Prompt

タスク

この問題を解いてください

問題:

目標

- 各スタッフが指定された勤務時間を満たすようにシフトを割り当てる
- スタッフの希望や制約を考慮する
- 店舗の営業時間中、必要な人数を確保する

条件

- 店舗は毎日8:00から22:00まで営業
- スタッフは1日最大8時間勤務
- スタッフは週に5日勤務
- 各時間帯に最低3人のスタッフが必要
- スタッフA、B、C、D、Eがいる

スタッフの希望・制約

- スタッフAは月曜日と水曜日は18:00以降勤務できない
- スタッフBは火曜日と木曜日は休み
- スタッフCは金曜日は終日勤務可能
- スタッフDは平日しか勤務できない
- スタッフEは週末しか勤務できない

例題

各スタッフの希望・制約を反映して、1週間のシフトを計画してください。

タスク

以下の問題を解いてください。

問題

各スタッフの希望・制約を反映して、1週間のシフトを計画して、表形式でシフト表を作成してください。

目標

- 各スタッフが指定された勤務時間を満たすようにシフトを割り当てる
- スタッフの希望や制約を考慮する
- 店舗の営業時間中、必要な人数を確保する

条件

- 店舗は毎日8:00から22:00まで営業
- スタッフは1日最大8時間勤務
- スタッフは週に5日勤務
- 各時間帯に最低3人のスタッフが必要
- スタッフA、B、C、D、Eがいる

スタッフの希望・制約

- スタッフAは月曜日と水曜日は18:00以降勤務できない
- スタッフBは火曜日と木曜日は休み
- スタッフCは金曜日は終日勤務可能
- スタッフDは平日しか勤務できない
- スタッフEは週末しか勤務できない



各スタッフの希望・制約を考慮し、1週間のシフトを計画し、シフト表を作成しました。

以下がスタッフのシフト表です。

シフト表

時間帯×日	月曜日	火曜日	水曜日	木曜日	金曜日	土曜日	日曜日
8:00-12:00	A, C, D	C, E	C, E				
12:00-16:00	B, C, D	C, E	C, E				
16:00-20:00	B, C, D	B, C, D	B, C	B, C, D	B, C, D	C, E	C, E
20:00-22:00	B, C, D	B, C, D	B, C	B, C, D	B, C, D	C, E	C, E

解説

- スタッフAは月曜日と水曜日の18:00以降は勤務できないため、これらの日の夜のシフトには含まれていません。
- スタッフBは火曜日と木曜日は休みのため、これらの日のシフトには含まれていません。
- スタッフCは金曜日は終日勤務可能であり、全ての時間帯に配置しています。
- スタッフDは平日しか勤務できないため、土曜日と日曜日のシフトには含まれていません。
- スタッフEは週末しか勤務できないため、土曜日と日曜日のシフトにのみ含まれています。

このシフト表により、各時間帯に最低3人のスタッフを確保し、スタッフの希望や制約を考慮しています。

Prompt の構造

Prompt

役割

数理最適化の先生として振舞ってください

タスク

この問題を解いてください

問題:

目標

- 各スタッフが指定された勤務時間を満たすようにシフトを割り当てる
- スタッフの希望や制約を考慮する
- 店舗の営業時間中、必要な人数を確保する

条件

- 店舗は毎日8:00から22:00まで営業
- スタッフは1日最大8時間勤務
- スタッフは週に5日勤務
- 各時間帯に最低3人のスタッフが必要
- スタッフA、B、C、D、Eがいる

スタッフの希望・制約

- スタッフAは月曜日と水曜日は18:00以降勤務できない
- スタッフBは火曜日と木曜日は休み
- スタッフCは金曜日は終日勤務可能
- スタッフDは平日しか勤務できない
- スタッフEは週末しか勤務できない

例題

各スタッフの希望・制約を反映して、1週間のシフトを計画してください。

役割

数理最適化の先生として振舞ってください

タスク
以下の問題を解いてください。

問題
各スタッフの希望・制約を反映して、1週間のシフトを計画して、表形式でシフト表を作成してください。

目標
各スタッフが指定された勤務時間を満たすようにシフトを割り当てる
スタッフの希望や制約を考慮する
店舗の営業時間中、必要な人数を確保する

条件
店舗は毎日8:00から22:00まで営業
スタッフは1日最大8時間勤務
スタッフは週に5日勤務
各時間中に最低3人のスタッフが必要
スタッフA、B、C、D、Eがいる

スタッフの希望・制約
スタッフAは月曜日と水曜日は18:00以降勤務できない
スタッフBは火曜日と水曜日は休み
スタッフCは金曜日は終日勤務可能
スタッフDは平日しか勤務できない
スタッフEは週末しか勤務できない

数理最適化問題の設定

1. 問題定式化

この問題はソフト割り当て問題です。

2. 変数の設定

x_{ij} : スタッフ i が曜日 j に勤務する人数

3. パラメータの設定

$i \in \{A, B, C, D, E\}$
 $j \in \{0, 1, 2, 3, 4, 5, 6\}$
 $k \in \{1, 2, 3\}$: 時間帯

4. 制約条件

1. 各時間中に最低3人のスタッフが勤務する

$$\sum_i x_{ij} \geq 3, \forall j, k$$

2. スタッフは1日最大8時間勤務する

$$\sum_k x_{ij} \leq 8, \forall i, j$$

3. スタッフは週に5日勤務する

$$\sum_j \sum_k x_{ij} \leq 40, \forall i$$

4. 各スタッフの希望・制約:

• スタッフAは月曜日と水曜日は18:00以降勤務できない

$$x_{A,0,3} = 0, x_{A,1,3} = 0$$

• スタッフBは火曜日と水曜日は休み

$$x_{B,1,0} = 0, x_{B,1,1} = 0$$

• スタッフCは金曜日は終日勤務可能

$$x_{C,5,k} = 1, \forall k$$

• スタッフDは平日しか勤務できない

$$x_{D,6,k} = 0, x_{D,0,k} = 0$$

• スタッフEは週末しか勤務できない

$$x_{E,0,k} = 0, x_{E,1,k} = 0$$

5. 目標

- 各スタッフが指定された勤務時間を満たすようにシフトを割り当てる
- スタッフの希望や制約を考慮する
- 店舗の営業時間中、必要な人数を確保する

6. 数理最適化モジュールの解法

この問題をPythonの"pulp"ライブラリを用いて解きます。

```
import pulp

# 問題定式化
prob = pulp.LpProblem("Shift Scheduling", pulp.LpMinimize)

# スタッフの希望・制約
staffs = ["A", "B", "C", "D", "E"]
days = range(7)
shifts = range(3)

# 変数の宣言
x = pulp.LpVariable.dicts("x", (staffs, days, shifts))

# 各時間中に最低3人のスタッフが勤務
for j in days:
    for k in shifts:
        prob += pulp.lpsum(x[i][j][k]) >= 3

# スタッフは1日最大8時間勤務
for i in staffs:
    for j in days:
        prob += pulp.lpsum(x[i][j][k]) <= 8

# スタッフは週に5日勤務
for i in staffs:
    for j in days:
        prob += pulp.lpsum(x[i][j][k]) <= 40

# スタッフAは月曜日と水曜日は18:00以降勤務できない
for j in days:
    for k in shifts:
        if j == 0 and k >= 1:
            prob += x["A"][j][k] == 0
        if j == 1 and k >= 1:
            prob += x["B"][j][k] == 0

# スタッフBは火曜日と水曜日は休み
for j in days:
    for k in shifts:
        if j == 1 and k >= 0:
            prob += x["B"][j][k] == 0
        if j == 1 and k >= 1:
            prob += x["B"][j][k] == 0

# スタッフCは金曜日は終日勤務可能
for j in days:
    for k in shifts:
        if j == 5:
            prob += x["C"][j][k] == 1

# スタッフDは平日しか勤務できない
for j in days:
    for k in shifts:
        if j == 6:
            prob += x["D"][j][k] == 0
        if j == 0:
            prob += x["D"][j][k] == 0

# スタッフEは週末しか勤務できない
for j in days:
    for k in shifts:
        if j == 0:
            prob += x["E"][j][k] == 0
        if j == 1:
            prob += x["E"][j][k] == 0
```

現在お環境では"pulp"ライブラリがインストールされていないため、数理最適化問題を解くことができません。

代わりに、制約条件と希望を考慮したシフト表の例を手動で作成してみます。

シフト表 (例)

スタッフ	月	月	月	火	火	火	水	水	水	木	木
日	8:00-14:00	14:00-20:00	20:00-22:00	8:00-14:00	14:00-20:00	20:00-22:00	8:00-14:00	14:00-20:00	20:00-22:00	8:00-14:00	14:00-20:00
A	○	○	×	○	○	○	○	○	×	○	○
B											
C											
D	○	○	○	×	×	×	○	○	○	×	×
E											
F	○	○	○	○	○	○	○	○	○	○	○
G											
H											
I											
J											
K											
L											
M											
N											
O											
P											
Q											
R											
S											
T											
U											
V											
W											
X											
Y											
Z											

このシフト表は、各スタッフの希望と制約を考慮して作成されました。必要な人数が確保されていることも確認されて

Prompt の構造

Prompt

役割
数理最適化の先生として振舞ってください

タスク
この問題を解いてください

タスクの進め方の指示
以下の手順で進めてください。

タスクの進め方の指示
ステップバイステップで考えてください

タスクの進め方の指示
深呼吸をして、一步一步取り組んでください。

タスクの進め方の指示
(SELF DISCOVERのテクニック...)

タスクの進め方の指示
タスクを完了できない場合は、不足している情報を、質問してください。

問題:
目標
•各スタッフが指定された勤務時間を満たすようにシフトを割り当てる
•スタッフの希望や制約を考慮する
•店舗の営業時間中、必要な人数を確保する
条件
•店舗は毎日8:00から22:00まで営業
•スタッフは1日最大8時間勤務
•スタッフは週に5日勤務
•各時間帯に最低3人のスタッフが必要
•スタッフA、B、C、D、Eがいる
スタッフの希望・制約
•スタッフAは月曜日と水曜日は18:00以降勤務できない
•スタッフBは火曜日と木曜日は休み
•スタッフCは金曜日は終日勤務可能
•スタッフDは平日しか勤務できない
•スタッフEは週末しか勤務できない
例題
各スタッフの希望・制約を反映して、1週間のシフトを計画してください。

進め方を
明示する。
より近道

もしくは

進め方を
考えてもらう

まず、各スタッフの希望と制約を整理し、それに基づいてシフト表を作成します。

スタッフの希望と制約の整理

- **スタッフA**
 - 月曜日と水曜日は18:00以降勤務できない
 - 週に5日勤務
- **スタッフB**
 - 火曜日と木曜日は休み
 - 週に5日勤務
- **スタッフC**
 - 金曜日は終日勤務可能
 - 週に5日勤務
- **スタッフD**
 - 平日しか勤務できない
 - 週に5日勤務
- **スタッフE**
 - 週末しか勤務できない
 - 週に5日勤務

シフトの設計

1. 週5日勤務のための勤務日決定

- A: 月、水、金、土、日
- B: 月、水、金、土、日
- C: 月、火、水、木、金
- D: 月、火、水、木、金

役割

数値最適化の先生として指導してください

タスク

以下の問題を解いてください。

深呼吸をして、一歩ずつ取り組んでください。

問題

各スタッフの希望・制約を反映して、1週間のシフトを計画して、表形式でシフト表を作成してください。

目標

- 各スタッフが指定された勤務時間を満たすようにシフトを割り当てる
- スタッフの希望や制約を考慮する
- 店舗の営業時間中、必要な人数を確保する

条件

- 店舗は毎日8:00から22:00まで営業
- スタッフは1日最大8時間勤務
- スタッフは週に5日勤務
- 各時間帯に最低3人のスタッフが必要
- スタッフA、B、C、D、Eがいる

スタッフの希望・制約

- スタッフAは月曜日と水曜日は18:00以降勤務できない
- スタッフBは火曜日と木曜日は休み
- スタッフCは金曜日は終日勤務可能
- スタッフDは平日しか勤務できない
- スタッフEは週末しか勤務できない

2. 時間帯の割り当て

- 店舗の営業時間は8:00から22:00
- シフトは8時間
- シフトは8:00-16:00と14:00-22:00の2つの時間帯に分割

シフト表の作成

時間帯	月曜日	火曜日	水曜日	木曜日	金曜日	土曜日	日曜日
8-16	A, B, D	C, D, E	A, B, D	C, D, E	C, D, E	A, B, E	A, B, E
14-22	C, E	C, D, E	C, E	C, D, E	C, E	A, B, E	A, B, E

スタッフのシフト配置の説明

- 月曜日: A, B, Dが8:00-16:00, C, Eが14:00-22:00
- 火曜日: C, Dが8:00-16:00, C, Eが14:00-22:00
- 水曜日: A, B, Dが8:00-16:00, C, Eが14:00-22:00
- 木曜日: C, Dが8:00-16:00, C, Eが14:00-22:00
- 金曜日: C, Dが8:00-16:00, C, Eが14:00-22:00
- 土曜日: A, B, Eが8:00-16:00, A, B, Eが14:00-22:00
- 日曜日: A, B, Eが8:00-16:00, A, B, Eが14:00-22:00

改善案

各曜日に最低3人のスタッフを確保するために、空いているシフトに追加のスタッフを割り当てます。例えば、火曜日の8:00-16:00のシフトにもう1人を追加し、2名にスタッフの希望や制約を調整します。

具体的なシフト調整例としては、例えばスタッフAの希望を考慮して火曜日の8:00-16:00にAを割り当てるなどが考えられます。また、シフトの偏りをなくすために、シフト間の調整も行います。

これで1週間のシフト表の初稿が完成です。必要に応じて、各スタッフの勤務日数や希望に応じて微調整を行います。

ChatGPT 自身に見直しをしてもらおう 😊

作成された回答は本意に反しているか? 精度に見直しをしてください。もし、正しくない場合は、どうすれば正しい回答を得られるかを考えてください。その考えを Prompt に反映して、再度実行してください。

🗣️ この指図の通り、シフト表の作成には正確な計算と考慮が必要です。最適化の観点から再度見直し、修正していきます。

条件の再確認

- 店舗は毎日8:00から22:00まで営業。
- 各時間帯に最低3人のスタッフが必要。
- スタッフは1日最大8時間勤務、週に5日勤務。
- スタッフの希望や制約を考慮。

スタッフの希望・制約の再整理

- スタッフA: 月-金は18:00以降勤務不可。
- スタッフB: 火-金は休み。
- スタッフC: 全曜日は終日勤務可能。
- スタッフD: 平日勤務のみ。
- スタッフE: 週末勤務のみ。

1日のシフト分割

- シフト1: 8:00-16:00
- シフト2: 14:00-22:00

シフト表の作成

月曜日

- 8:00-16:00: A, B, D
- 14:00-22:00: C, D, E

水曜日

- 8:00-16:00: A, B, D
- 14:00-22:00: C, D, E

木曜日

- 8:00-16:00: C, D, E
- 14:00-22:00: C, D, E

金曜日

- 8:00-16:00: C, D, E
- 14:00-22:00: C, D, E

土曜日

- 8:00-16:00: A, B, E
- 14:00-22:00: A, B, E

日曜日

- 8:00-16:00: A, B, E
- 14:00-22:00: A, B, E

これにより、各シフトに最低3人のスタッフを確保し、スタッフの希望と制約を考慮したシフト表が作成されました。

最終シフト表

時間帯	月曜日	火曜日	水曜日	木曜日	金曜日	土曜日	日曜日
8-16	A, B, D	C, D, E	A, B, D	C, D, E	C, D, E	A, B, E	A, B, E
14-22	C, D, E	A, B, E	A, B, E				

このシフト表では、各時間帯に必要な3人のスタッフを確保し、スタッフの希望や制約を考慮したものととなります。

“Simple is the Best”

長文のPromptのリスク

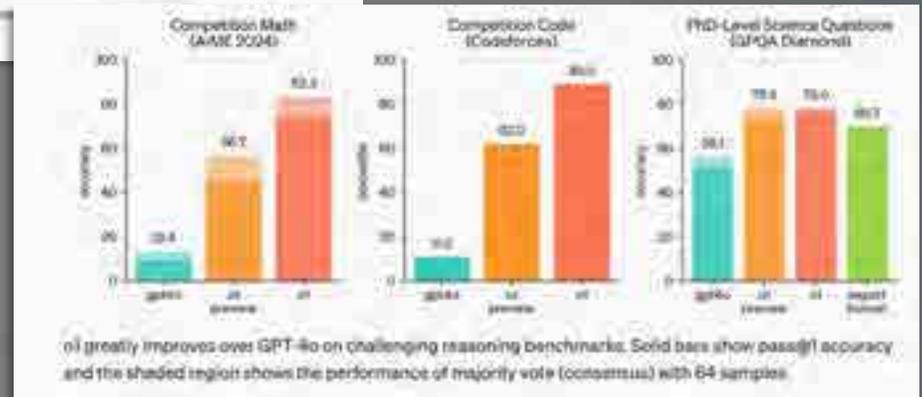
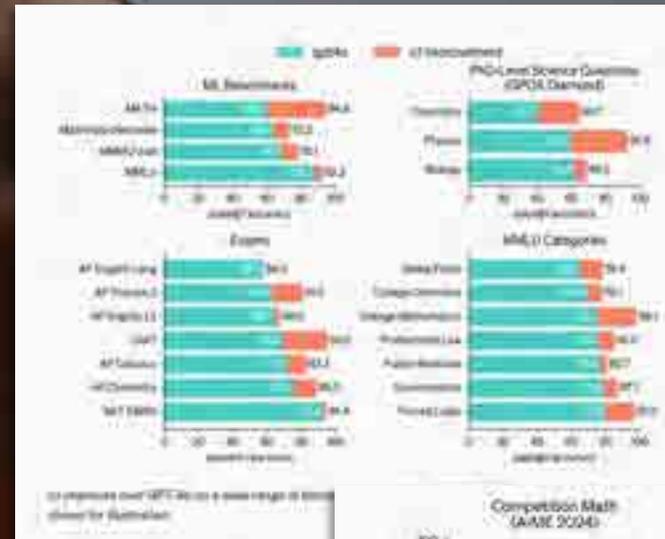
- 応答時間が遅くなる
- ノイズ増加により出力データの精度が落ちる
- Chat回数が減る。入力トークン数には上限がある
- サービスによっては使用料金が增える

o1 Preview 登場! (9/13です) 出始めた!

- 推論時に強化学習を使って、複雑な問題を解く実行計画を自ら作成
- GPTシリーズとは別のモデル
- 正規版は今後登場予定



The o1-preview and o1-mini models are now accessible in Azure AI Studio and GitHub Models to a select group of Azure customers to collaboratively explore and identify the unique strengths of each model. The o1 series of advanced reasoning models excel at complex and nuanced problem spaces like these:



Prompt の本質に集中できる!

Prompt

役割
数理最適化の先生として振舞ってください

タスク
この問題を解いてください

タスクの進め方の指示
以下の手順で進めてください。

タスクの進め方の指示
ステップバイステップで考えてください

タスクの進め方の指示
深呼吸をして、一步一步取り組んでください。

タスクの進め方の指示
(SELF DISCOVERのテクニック...)

タスクの進め方の指示
タスクを完了できない場合は、不足している情報を、質問してください。

問題:
目標
•各スタッフが指定された勤務時間を満たすようにシフトを割り当てる
•スタッフの希望や制約を考慮する
•店舗の営業時間中、必要な人数を確保する
条件
•店舗は毎日8:00から22:00まで営業
•スタッフは1日最大8時間勤務
•スタッフは週に5日勤務
•各時間帯に最低3人のスタッフが必要
•スタッフA、B、C、D、Eがいる
スタッフの希望・制約
•スタッフAは月曜日と水曜日は18:00以降勤務できない
•スタッフBは火曜日と木曜日は休み
•スタッフCは金曜日は終日勤務可能
•スタッフDは平日しか勤務できない
•スタッフEは週末しか勤務できない
例題
各スタッフの希望・制約を反映して、1週間のシフトを計画してください。

進め方を
明示する。
より近道

もしくは

進め方を
考えてもらう

統計学

を学び始める

[Master] 統計学を学ぶ (chatgpt.com)

統計学の初心者です。これから統計学について学びます。体系立てて理解を進めたいのですが、どんな手順で学ぶのが効果的ですか？

⑧ 統計学を体系立てて学ぶための効果的な手順は以下の通りです。初心者向けに、各ステップを簡単に説明します。

1. 基本的な概念の理解

- **データとは何か:** データの種類（定性的データ、定量的データ）、データの収集方法（観察、実験、調査）。
- **記述統計:** データの要約と整理（平均、中央値、最頻値、分散、標準偏差など）。

2. データの視覚化

- **グラフの作成と解釈:** 棒グラフ、ヒストグラム、散布図、箱ひげ図などを使ってデータを視覚化し、パターンや傾向を理解する。

3. 確率の基礎

- **確率の概念:** 確率の基本定義、確率変数、確率分布（正規分布、二項分布など）。
- **確率規則:** 足し算のルール、掛け算のルール、条件付き確率、独立性。

4. 推測統計

- **標本と母集団:** 標本抽出方法、標本統計量と母集団パラメータの違い。
- **推定と仮説検定:** 点推定と区間推定、仮説検定の基本的な考え方（帰無仮説、対立仮説、有意水準、p値など）。

5. 回帰分析

- **単回帰分析:** 1つの独立変数を使った回帰モデル、回帰直線の意味、決定係数。
- **重回帰分析:** 複数の独立変数を使った回帰モデル、共線性の問題、モデルの適合性の評価。

6. 実践的なデータ分析

- **データ収集と前処理:** データのクレンジング、欠損値の処理、データの標準化。
- **ソフトウェアの使用:** Excel、R、Pythonなどの統計ソフトウェアを使ったデータ分析の基本操作。

7. 応用統計

- **多変量解析:** 主成分分析、因子分析、クラスター分析など。
- **時間系列解析:** 時系列データの特徴、トレンド、季節性、移動平均法、ARIMAモデルなど。

8. 統計学の倫理と実務

- **データ倫理:** データのプライバシー、データのバイアス、統計の誤用のリスク。
- **レポート作成と結果の解釈:** 分析結果の正しい解釈と報告方法。

学習の進め方のポイント

- **小さなステップで進める:** 各テーマを小さな部分に分けて学習し、理解を深める。
- **実践を重視する:** 実際にデータを扱って分析する経験を積むことで理解が深まります。
- **リソースの活用:** 書籍、オンラインコース、動画チュートリアルなど多様な学習リソースを活用する。
- **継続的な学習:** 定期的に学習と実践を繰り返し、理解を深める。

この学習手順を参考に、着実に統計学の知識とスキルを身につけていきましょう。

「統計の基本用語」について深く知りたいです。ユースケースの説明と、その実行した詳細な具体例、計算がある場合には計算の詳細なステップバイステップの手順とともに、統計用語の全てについて可能な限り詳細な解説してください。

深呼吸をして、一歩ずつ取り組んでください。

- 統計の各分野について、ユースケースと具体例、計算の概要を含めて説明します。以下は各分野を詳しく解説します。

1. データ (Data)

定義: 観察や測定によって得られる事象や値。

ユースケース: 市場調査での顧客の年齢、性別、購入履歴など。

具体例

- 顧客の年齢データ: 25, 34, 28, 40, 30

2. 母集団 (Population) と標本 (Sample)

定義

- 母集団: 調査対象となる全体的なデータ。
- 標本: 母集団の一部を取り出したもの。

ユースケース: ある製品の全出荷量 (母集団) からサンプルサイズ100入の抽出 (標本)。

具体例

- 母集団: 50,000人の出荷量
- 標本: 100人の出荷量

3. 記述統計 (Descriptive Statistics)

定義: データを要約し、特徴を記述する統計手法。

ユースケース: 社内の従業員の高身データを要約し、給与の傾向を理解する。

具体例

- 平均 (Mean)
- 中央値 (Median)
- 最頻値 (Mode)
- 分散 (Variance)
- 標準偏差 (Standard Deviation)

計算の詳細

- 平均: データの合計をデータ数で割る
 - 例: 25, 34, 28, 40, 50 の平均 = $(25 + 34 + 28 + 40 + 50) / 5 = 35.4$
- 中央値: データを小さい順に並べて中央の値
 - 例: 25, 28, 34, 40, 50 の中央値 = 34
- 最頻値: 最も頻りに出現する値
 - 例: 25, 34, 34, 40, 50 の最頻値 = 34
- 分散: データ各値と平均との差の2乗の平均
 - 例: $(25 - 35.4)^2 + (34 - 35.4)^2 + (28 - 35.4)^2 + (40 - 35.4)^2 + (50 - 35.4)^2 / 5 = 94.24$
- 標準偏差: 分散の平方根
 - 例: $\sqrt{94.24} = 9.71$

4. 確率 (Probability)

定義: ある事象が起こる可能性の度合いを示す数値。

ユースケース: サイコロを振ったときに「3」が出る確率を求める。

具体例

- サイコロの1つの面が出る確率 = $1/6 = 0.167$

5. 確率分布 (Probability Distribution)

定義: 確率変数を取り得る値とその値に対する確率を表したものを。

ユースケース: 試験の点数が正規分布に従うと仮定し、平均点と標準偏差を基に成績を評価する。

具体例

- 正規分布 (Normal Distribution)
 - 平均 $\mu = 70$ 、標準偏差 $\sigma = 10$ の場合の分布
 - 例: 試験の点数が60~80の範囲に収まる確率を求める

6. 標準正規分布 (Standard Normal Distribution)

定義: 平均0、標準偏差1の正規分布。

ユースケース: 標準化されたテストスコアの比較。

具体例

- 標準正規分布の値をzスコアで表す
 - 例: $z = (x - \mu) / \sigma$
 - ある点数 $x = 85$ が、平均70、標準偏差10の分布でのzスコア
 - $z = (85 - 70) / 10 = 1.5$

7. 仮説検定 (Hypothesis Testing)

定義: 仮説を統計的に検証する方法。

ユースケース: 新しい薬が従来の薬より効果があるかを検定する。

具体例:

- 帰無仮説 (H0): 新しい薬の効果は従来の薬と同じ
- 対立仮説 (H1): 新しい薬の効果は従来の薬より高い

計算の詳細:

- p値 (p-value) の計算
 - サンプルデータから統計量を計算し、対応するp値を求める
 - p値が有意水準 (通常0.05) 以下なら、帰無仮説を棄却

8. 回帰分析 (Regression Analysis)

定義: ある変数が他の変数にどのように影響されるかを分析する方法。

ユースケース: 広告費用が売上に与える影響を調査する。

具体例:

- 単回帰分析
 - 売上 = $a + b \times$ 広告費用
 - 例: 広告費用が100万円のときの売上を予測する
 - $a = 50, b = 2$ の場合
 - 売上 = $50 + 2 \times 100 = 250$ 万円

9. 信頼区間 (Confidence Interval)

定義: 母集団のパラメータが特定の範囲内にある確率を示す区間。

ユースケース: 製品の平均寿命の推定範囲を示す。

具体例:

- 95%信頼区間
 - 例: 平均寿命が1000時間、標準誤差が50時間の場合
 - 信頼区間 = 平均 \pm (1.96 \times 標準誤差)
 - $1000 \pm (1.96 \times 50) = 1000 \pm 98 = [902, 1098]$

10. p値 (p-value)

定義: 仮説検定において、観測されたデータが帰無仮説のもとで起こる確率。

ユースケース: 新製品が既存製品より優れているかを検証する。

具体例:

- 帰無仮説が真である場合に観測データ以上の結果が得られる確率が0.03 (3%) なら、 p 値 = 0.03
- p 値が0.05未満なら、帰無仮説を棄却する。

総まとめ

これらの基本用語を理解することで、統計学の基礎が身につきます。各ステップを丁寧に学び、実際のデータを使って計算や分析を行うことで、より深い理解が得られるでしょう。

ありがとうございます、良く理解できました。

ただ、数式について理解できないものがあります。「分散」の計算式について、子どもでも分かるようなかみ砕いた表現で、詳細な手順と共に教えてください。

深呼吸をして、一歩ずつ取り組んでください。

一人一人異なる理解度への対応力

分散の計算を子どもでも分かるように、簡単な表現と詳細な手順で説明します。

分散とは？

分散は、データがどれだけ平均からばらついているかを表す数字です。簡単に言うと、データの「散らばり具合」を測るものです。

分散の計算手順

例として、次の5つの数値を使います

データ: 4, 8, 6, 5, 3

手順1: 平均を求める

平均は、すべての数値を足して、その数値の個数で割ることで求められます。

1. すべての数値を足します。

$$4 + 8 + 6 + 5 + 3 = 26$$

2. 数値の個数で割ります。

$$26 \div 5 = 5.2$$

なので、このデータの平均は5.2です。

手順2: 各数値と平均の差を求める

次に、各数値が平均からどれだけ離れているかを計算します。

1. $4 - 5.2 = -1.2$
2. $8 - 5.2 = 2.8$
3. $6 - 5.2 = 0.8$
4. $5 - 5.2 = -0.2$
5. $3 - 5.2 = -2.2$

手順3: 差の2乗を求める

それぞれの差を2乗（自分自身に掛ける）します。

1. $(-1.2)^2 = 1.44$
2. $(2.8)^2 = 7.84$
3. $(0.8)^2 = 0.64$
4. $(-0.2)^2 = 0.04$
5. $(-2.2)^2 = 4.84$

手順4: 2乗した結果を足し合わせる

今求めた2乗の数値を全部足します。

$$1.44 + 7.84 + 0.64 + 0.04 + 4.84 = 14.8$$

手順5: 合計をデータの個数で割る

最後に、この合計をデータの個数で割ります。

$$14.8 \div 5 = 2.96$$

これが分散です。このデータの分散は2.96です。

分散の計算をまとめると

1. データの平均を計算する
2. 各数値と平均の差を求める
3. 差の2乗を計算する
4. 2乗した結果を全部足す
5. 合計をデータの個数で割る

分散は、データが平均からどれだけ散らばっているかを教えてくれる重要な指標です。

確率の基礎について、学校の教科書でも使えるような具体性で、可能な限り詳細に説明をしてください。

📌 統計学を体系立てて学ぶための効果的な手順は以下の通りです。初心者向けに、各ステップを簡単に説明します。

1. 基本的な概念の理解

- **データとは何か:** データの種類（定性的データ、定量的データ）、データの収集方法（観察、実験、調査）。
- **記述統計:** データの要約と整理（平均、中央値、最頻値、分散、標準偏差など）。

2. データの視覚化

- **グラフの作成と解釈:** 棒グラフ、ヒストグラム、散布図、箱ひげ図などを使ってデータを視覚化し、パターンや傾向を理解する。

3. 確率の基礎

- **確率の概念:** 確率の基本定義、確率変数、確率分布（正規分布、二項分布など）。
- **確率規則:** 足し算のルール、掛け算のルール、条件付き確率、独立性。

4. 推測統計

- **標本と母集団:** 標本抽出方法、標本統計量と母集団パラメータの違い。
- **推定と仮説検定:** 点推定と区間推定、仮説検定の基本的な考え方（帰無仮説、対立仮説、有意水準、p値など）。

5. 回帰分析

- **単回帰分析:** 1つの独立変数を使った回帰モデル、回帰直線の意味、決定係数。
- **重回帰分析:** 複数の独立変数を使った回帰モデル、共線性の問題、モデルの適合性の評価。

6. 実践的なデータ分析

- **データ収集と前処理:** データのクレンジング、欠損値の処理、データの標準化。
- **ソフトウェアの使用:** Excel、R、Pythonなどの統計ソフトウェアを使ったデータ分析の基本操作。

確率の基礎について、学校の教科書でも使えるように、具体例を交えて詳細に説明します。

確率の基礎

確率とは？

確率は、ある事象（出来事）が起こる可能性の度合いを示す数値です。0から1の間の数値で表され、0は「絶対に起こらない」、1は「絶対に起こる」を意味します。

確率の基本用語

1. 試行 (Trial) : 確率を計算するための実験や操作のこと。例：コインを投げる、サイコロを振るなど。
2. 事象 (Event) : 試行の結果のうち、関心のある特定の出来事。例：コインで表が出る、サイコロで6が出るなど。
3. 標本空間 (Sample Space) : すべての可能な結果の集合。例：コインを投げたときの標本空間は「表」と「裏」。
4. 確率 (Probability) : 事象が起こる可能性を数値で表したものを。

確率の計算

基本的な確率の公式

確率 P は次の公式で計算されます：

$$P(A) = \frac{\text{事象Aが起こる場合の数}}{\text{すべての可能な場合の数}}$$

具体例1: コイン投げ

- 試行: コインを1回投げる
- 標本空間: (表, 裏)
- 事象A: 表が出る

コインを投げると、表が出る場合は1つ、裏が出る場合も1つです。つまり、すべての可能な場合の数は2つです。

$$P(\text{表}) = \frac{1}{2} = 0.5$$

コインの裏が出る確率も同じく0.5です。

具体例2: サイコロ

- 試行: 6面のサイコロを1回振る
- 標本空間: (1, 2, 3, 4, 5, 6)
- 事象B: 3が出る

サイコロを振ると、1から6のいずれかの目が出ます。すべての可能な場合の数は6つです。

$$P(3) = \frac{1}{6} \approx 0.167$$

他の目が出る確率もそれぞれ1/6です。

確率のルール

加法定理 (The Addition Rule)

- 独立した事象: 事象Aと事象Bが同時に起こることがない場合、どちらかが起こる確率は、個別の確率の和になります。

$$P(A \cup B) = P(A) + P(B)$$

具体例: サイコロで偶数 (2, 4, 6) または奇数 (1, 3, 5) が出る確率。

$$P(\text{偶数}) = \frac{3}{6} = 0.5, \quad P(\text{奇数}) = \frac{3}{6} = 0.5$$

$$P(\text{偶数} \cup \text{奇数}) = 0.5 + 0.5 = 1$$

乗法定理 (The Multiplication Rule)

- 独立した事象: 事象Aと事象Bが互いに影響を与えない場合、両方が同時に起こる確率は、それぞれの確率の積になります。

$$P(A \cap B) = P(A) \times P(B)$$

具体例: コインを2回投げて、両方とも表が出る確率。

$$P(\text{1回目に表}) = \frac{1}{2} = 0.5, \quad P(\text{2回目に表}) = \frac{1}{2} = 0.5$$

$$P(\text{両方とも表}) = 0.5 \times 0.5 = 0.25$$

条件付き確率 (Conditional Probability)

ある事象が起こった場合に、別の事象が起こる確率を条件付き確率と言います。

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

具体例: サイコロを振って、偶数が出た後に、さらに4が出る確率。

$$P(\text{偶数}) = \frac{3}{6} = 0.5, \quad P(4|\text{偶数}) = \frac{1}{3} = 0.333$$

$$P(4 \cap \text{偶数}) = P(4|\text{偶数}) \times P(\text{偶数}) = 0.333 \times 0.5 \approx 0.167$$

ベイズの定理 (Bayes' Theorem)

事象Aと事象Bに関連する確率を逆に求めるための公式です。

$$P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$$

具体例: 風邪を引いている人がある症状 (例えば、咳) が出る確率が0.8、全体の人が咳をする確率が0.3、風邪を引く確率が0.1の場合、咳をしている人が風邪を引いている確率を求める。

$$P(\text{風邪}|咳) = \frac{P(咳|\text{風邪}) \times P(\text{風邪})}{P(咳)} = \frac{0.8 \times 0.1}{0.3} \approx 0.267$$

まとめ

確率は、日常の多くの状況で役立つ概念です。コインを投げたり、サイコロを振ったりする簡単な例から始めて、確率の基本をしっかりと理解しましょう。こうした基礎があれば、もっと複雑な統計や確率の問題にも取り組めるようになります。

「推測統計」について具体的なユースケースをあげてください。

具体例が、理解を助けてくれる場合は多い

- ④ 推測統計は、データの一部（標本）から全体（母集団）の特性を推測するための統計手法です。以下に具体的なユースケースを挙げて説明します。

ユースケース1: 新製品の市場調査

目的: 新製品の発売前に、消費者の意見を知るために推測統計を使います。

手順:

1. **標本抽出:** 新製品に関するアンケートを無作為に選んだ1000人の消費者に実施します。
2. **記述統計:** アンケートの結果から、各回答の平均、中央値、最頻値を計算します。
3. **推定**
 - **点推定:** 標本から得られた平均スコアを母集団の平均スコアの推定値とします。
 - **区間推定:** 信頼区間を計算して、母集団の平均スコアがある範囲を推定します。
4. **仮説検定**
 - 新製品の購買意欲が高いと仮定して、その仮説を検定します。
 - p値を計算し、購買意欲が高いかどうかを統計的に検証します。

ユースケース2: 医療試験

目的: 新しい治療法が効果的かどうかを評価するために推測統計を使います。

手順:

1. **標本抽出:** 新しい治療法を50人の患者に実施し、従来の治療法を50人の患者に実施します。
2. **記述統計:** 各グループの治療結果の平均、標準偏差を計算します。
3. **推定**
 - **点推定:** 各グループの治療効果の平均を推定します。
 - **区間推定:** 各グループの治療効果の信頼区間を計算します。
4. **仮説検定**
 - **帰無仮説 (H0):** 新しい治療法と従来の治療法に差がない。
 - **対立仮説 (H1):** 新しい治療法の方が効果がある。
 - t検定を実施し、p値を計算して、帰無仮説を棄却するかどうかを判断します。

ユースケース3: 工場の品質管理

目的: 生産ラインの製品品質を監視し、標準からの逸脱を検出するために推測統計を使います。

手順:

1. **標本抽出:** 毎時間、ランダムに選んだ50個の製品を検査します。
2. **記述統計:** 各標本の平均品質スコア、標準偏差を計算します。
3. **推定**
 - **点推定:** 品質スコアの平均を推定します。
 - **区間推定:** 品質スコアの信頼区間を計算します。
4. **仮説検定**
 - **帰無仮説 (H0):** 生産ラインは安定している。
 - **対立仮説 (H1):** 生産ラインは標準から逸脱している。
 - コントロールチャートを作成し、品質スコアが管理限界を超えているかどうかを判断します。

推測統計の具体的な手順

1. 標本抽出

- ランダムサンプリング: 母集団から無作為に標本を選ぶ方法。
- 例: 1000人の消費者から無作為に100人を選ぶ。

2. 記述統計の計算

- 平均: データの中心傾向を示す。

$$\text{平均} = \frac{\sum X_i}{n}$$

- 標準偏差: データのばらつきを示す。

$$\text{標準偏差} = \sqrt{\frac{\sum (X_i - \mu)^2}{n}}$$

3. 推定

- 点推定: 母集団のパラメータの推定値。
- 区間推定: パラメータが含まれる範囲を推定。

$$\text{信頼区間} = \text{点推定値} \pm (z \times \text{標準誤差})$$

4. 仮説検定

- 帰無仮説 (H0): 現状の仮定。
- 対立仮説 (H1): H0に対する仮定。
- 検定統計量: 仮説の検証に用いる値。
- p値: 仮説が正しいときに観測された結果が得られる確率。
- 有意水準: 通常0.05。p値がこれより小さい場合、帰無仮説を棄却。

推測統計は、標本データから母集団全体について結論を導くための強力な手法です。これらのユースケースを通じて、実際にどのように推測統計が使われるかを理解することで、より実践的な統計分析のスキルを身につけることができます。

- ◎ 推測統計に関連する興味深いエピソードはいくつもあります。ここでは、その中の一つとして「ジェリコ・セレブニックのギネス記録」に関するエピソードを紹介します。このエピソードは、推測統計の重要性とその実践的な応用を示す良い例です。

エピソード: ジェリコ・セレブニックのギネス記録

背景

ジェリコ・セレブニックは、1990年代にギネス記録に挑戦した統計学者です。彼は「世界一のポップコーン作り機」という記録を達成するために、統計的手法を駆使しました。この挑戦は、製品開発や品質管理に推測統計を適用する実例として有名です。

記録挑戦

セレブニックの目標は、連続して最も多くのポップコーンを破裂させる機械を作ることでした。彼は、製品の効率性と品質を最大化するために、徹底的なデータ収集と分析を行いました。

推測統計の応用

1. **データ収集:** セレブニックは、さまざまな温度、圧力、時間などの条件下でポップコーンを破裂させる実験を何百回も行いました。各試行ごとに、破裂したポップコーンの数と、未破裂の数を記録しました。
2. **記述統計の計算:** 各条件下での平均破裂数、標準偏差などの記述統計を計算し、データのばらつきを理解しました。
3. **推定:** 収集したデータから、最適なポップコーン破裂条件（温度、圧力、時間）を推定しました。
4. **仮説検定:** 最適条件を確認するために、仮説検定を行いました。例えば、「温度が一定以上であれば、破裂率が95%を超える」という仮説を検証しました。
5. **反復改善:** 推測統計の結果を基に、ポップコーン作り機の設計を何度も改良しました。

結果

セレブニックは、最適な条件を見つけ出し、その条件下で機械を動かした結果、見事にギネス記録を達成しました。彼の機械は、ほぼすべてのポップコーンを完璧に破裂させることができました。

一人一人の学ぶ方への寄り添う存在として

- **事実確認**は必須
 - それは、書籍・ネットの記事とて同様
- 作成**時間**が圧倒的に速い
 - 話を聞くよりも、文字・記号などを読むほうが圧倒的に速い
- **個々**の知りたい事、**学び方**への対応が出来る
 - 「尋ね方」を学ぶ必要はある。第一歩は「尋ねられるんだ」という能力を知る事
- **網羅性**は高い
- 適当な**事例**・**ユースケース**の作成力が非常に高い

A top-down view of a desk. In the upper left, a portion of a black laptop is visible. Below it, a black keyboard is being used by a person whose hands are visible at the bottom of the frame. To the right of the keyboard, a large, colorful pile of dice is scattered on the white desk surface. The dice are in various colors (red, blue, green, yellow, purple, black) and some have different faces or patterns. A black mouse is partially visible near the dice. On the far right, there is a clear plastic bag containing more dice and a small glass bowl. The overall scene suggests a workspace where a complex task, possibly related to optimization, is being performed, with the dice representing a stochastic or combinatorial problem.

数理最適化への取り組み

戦略

- 各種 数理最適化の手法は、比較的**一般論**が多い
- 実際の**計算処理**は複雑なものが多く、プログラムコードを実行すると長時間要するものも出てくる
 - プログラムの実行が、ChatGPTの中で可能であれば、それを利用
 - 出来ない場合は、自分のPCやクラウド環境などを適時利用

真実

- この思考レベルになると、Prompt Engineering のテクニックが必須
- それでも、正解に辿り着かない事は多々ある



“巡回セールスマン問題”

[Math] 巡回セールスマン問題 (chatgpt.com)

「問題」😊

問題:

次の5点をちょうど1度ずつ訪問する順回路のうち最短距離のものとその距離を考えてください。

(1,0), (2,2), (2,1), (0,0), (2,0)

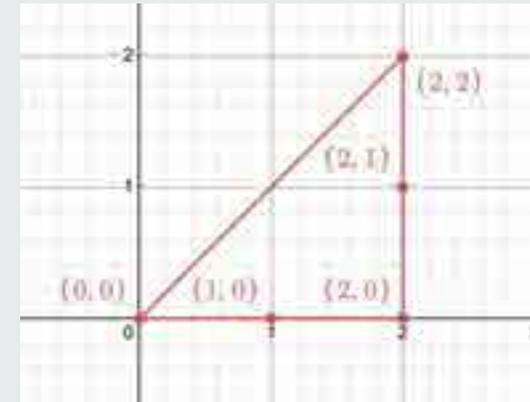
進め方:

段階的に計画を作成してください。計画が問題の解決に役に立つのかを詳細に分析して、改善点があればリストアップをしてください。

改善点を基に計画を見直して、再度問題の解決に取り組んでください。

深呼吸をして、一歩ずつ取り組んでください。

解答



最短距離の順回路: (1,0) → (0,0) → (2,2) → (2,1) → (2,0) → (1,0)

総距離: $4 + 2\sqrt{2} \doteq 6.83$

◎ 計画

段階 1: 問題の理解

まず、問題の要件と目標を理解します。

- 問題の要件:
 - 5つの点: (1,0), (2,2), (2,1), (0,0), (2,0)
 - 各点を1度だけ訪問する順回路 (巡回セールスマン問題)
 - 最短距離の経路を求める

段階 2: 各点間の距離計算

各点間の距離をユークリッド距離で計算します。

距離 d は次のように計算されます。

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

段階 3: 距離行列の作成

各点間の距離を行列形式で整理します。

段階 4: 全ての順回路の生成

全ての可能な順回路を生成します。点の数が5なので、順回路の総数は $5! = 120$ 通りです。

段階 5: 各順回路の総距離計算

各順回路の総距離を計算し、最短距離の順回路を特定します。

段階 6: 最短距離の順回路とその距離の報告

最短距離の順回路とその距離を出力します。

段階 2: 各点間の距離計算

まず、各点間の距離を計算します。

点の座標:

- A (1,0)
- B (2,2)
- C (2,1)
- D (0,0)
- E (2,0)

各点間の距離:

- AB: $\sqrt{(2-1)^2 + (2-0)^2} = \sqrt{1+4} = \sqrt{5}$
- AC: $\sqrt{(2-1)^2 + (1-0)^2} = \sqrt{1+1} = \sqrt{2}$
- AD: $\sqrt{(0-1)^2 + (0-0)^2} = \sqrt{1} = 1$
- AE: $\sqrt{(2-1)^2 + (0-0)^2} = \sqrt{1} = 1$
- BC: $\sqrt{(2-2)^2 + (1-2)^2} = \sqrt{1} = 1$
- BD: $\sqrt{(0-2)^2 + (0-2)^2} = \sqrt{4+4} = \sqrt{8}$
- BE: $\sqrt{(2-2)^2 + (0-2)^2} = \sqrt{4} = 2$
- CD: $\sqrt{(0-2)^2 + (0-1)^2} = \sqrt{4+1} = \sqrt{5}$
- CE: $\sqrt{(2-2)^2 + (0-1)^2} = \sqrt{1} = 1$
- DE: $\sqrt{(2-0)^2 + (0-0)^2} = \sqrt{4} = 2$

これを基に距離行列を作成します。

距離行列

	A	B	C	D	E
A	0	$\sqrt{5}$	$\sqrt{2}$	1	1
B	$\sqrt{5}$	0	1	$\sqrt{8}$	2
C	$\sqrt{2}$	1	0	$\sqrt{5}$	1
D	1	$\sqrt{8}$	$\sqrt{5}$	0	2
E	1	2	1	2	0

段階 3: 全ての順回路の生成と最短距離の計算

次に、全ての可能な順回路を生成し、その総距離を計算します。この計算はPythonなどのプログラムを使用するのが現実的です。

Pythonを使用して順回路を生成し、最短距離のものを特定します。

プログラムコードの方が最適と判断

Pythonのコード生成

Pythonコード

```
python
import itertools
import math

# 点の座標
points = {
    'A': (0, 0),
    'B': (1, 0),
    'C': (0, 1),
    'D': (1, 1),
    'E': (0, 1)
}

# ユークリッド距離を計算する関数
def euclidean_distance(p1, p2):
    return math.sqrt((p2[0] - p1[0])**2 + (p2[1] - p1[1])**2)

# 全ての順回路を生成
permutations = list(itertools.permutations(points.keys()))

# 最短距離の順回路を特定
min_distance = float('inf')
best_route = None

for perm in permutations:
    # 総距離を計算
    total_distance = 0
    for i in range(len(perm)):
        total_distance += euclidean_distance(points[perm[i]], points[perm[(i + 1) % len(perm)]])
    if total_distance < min_distance:
        min_distance = total_distance
        best_route = perm

print("最短距離の順回路:", best_route)
print("総距離:", min_distance)
```

このプログラムは全ての可能な順回路を生成し、それぞれの総距離を計算して最短距離の順回路を特定します。

段階 4: 結果の報告

最短距離の順回路とその距離を報告します。

この計画に基づいて解決案を提供します。もし問題があれば、段階ごとに見直し調整を行います。

🔊 📄 📄 📄

作業を続けてください。

分析が完了しました

```
Python
Python 3.10.12 (tags/v3.10.12:10.12.2022)

import itertools
import math

# 点の座標
points = {
    'A': (1, 0),
    'B': (2, 2),
    'C': (2, 1),
    'D': (0, 0),
    'E': (2, 0)
}

# ユークリッド距離を計算する関数
def euclidean_distance(p1, p2):
    return math.sqrt((p2[0] - p1[0])**2 + (p2[1] - p1[1])**2)

# 全ての順回路を生成
permutations = list(itertools.permutations(points.keys()))

# 最短距離の順回路を見つける
min_distance = float('inf')
best_route = None

for perm in permutations:
    # 総距離を計算
    total_distance = 0
    for i in range(len(perm)):
        total_distance += euclidean_distance(points[perm[i]], points[perm[(i + 1) % len(perm)]])
    if total_distance < min_distance:
        min_distance = total_distance
        best_route = perm
```

```
for perm in permutations:
    # 総距離を計算
    total_distance = 0
    for i in range(len(perm)):
        total_distance += euclidean_distance(points[perm[i]], points[perm[(i + 1) % len(perm)]])
    if total_distance < min_distance:
        min_distance = total_distance
        best_route = perm

best_route, min_distance
```

結果:
(('A', 'D', 'B', 'C', 'E'), 6.82842712474619)

最短距離の順回路とその距離は以下の通りです。

- 最短距離の順回路: $A \rightarrow D \rightarrow B \rightarrow C \rightarrow E$
- その距離: 約 6.83

この順回路では、各点を1度だけ訪問し、全ての点を巡回して元の点に戻る最短距離を達成しています。[-]

🔊 📄 📄 📄

{最短距離の順回路}を、問題の中にあつたデータで置き換えてください。

< 2/2

最短距離の順回路を問題の中にあつたデータで置き換えると、以下のようになります。

- 最短距離の順回路: $(1,0) \rightarrow (0,0) \rightarrow (2,2) \rightarrow (2,1) \rightarrow (2,0)$
- その距離: 約 6.83

この順回路では、各点を1度だけ訪問し、全ての点を巡回して元の点に戻る最短距離を達成しています。

Pythonのコード **実行**

Code Interpreterによる

正解

“シフトスケジューリング問題”

[Math] シフトスケジューリング問題

問題も作ってもらいました

ChimGPT 44

4 役割
あなたは数値最適化が専門の、数学の大学の教授です。

4 作業
シフトスケジューリング問題について、学生に課す課題を作成してください。

シフトスケジューリング問題 - 課題

課題の概要

シフトスケジューリング問題は、労働者の勤務シフトを効率的に割り当てるための数理最適化問題です。この課題では、あなたは特定の制約条件の下で労働者にシフトを割り当てるモデルを構築し、最適解を求めることが求められます。

課題の背景

ある企業では、以下のような労働者のシフトスケジュールを作成する必要があります。

- 労働者は8人います (A, B, C, D, E, F, G, H)。
- シフトは3つの時間帯に分かれています (朝 (6:00 - 14:00)、昼 (14:00 - 22:00)、夜 (22:00 - 6:00))。
- それぞれのシフトには少なくとも2人の労働者が必要です。

制約条件

- 各労働者は1日に1つのシフトしか担当できません。
- 各労働者は連続する2日間のシフトが与えてはなりません。
- 各労働者は1週間にかぎらずとも1日は休む必要があります。
- 各シフトには少なくとも2人の労働者が必要です。
- 各労働者のシフト希望は以下の通りです (希望するシフトには空欄に割り当ててください)。

労働者	希望シフト
A	朝シフト
B	昼シフト
C	夜シフト
D	朝シフト、夜シフト
E	昼シフト、夜シフト
F	夜シフト
G	朝シフト
H	夜シフト

課題

- 数学モデルの構築
 - 変数、目的関数、制約条件を含む数学モデルを構築してください。
- 実装
 - 構築した数学モデルを用いて、Pythonの数値最適化ライブラリ (例: PyLP, Gurobi) を使って解を求めてください。
- 結果の分析
 - 得られた解が制約を満たしているかを確認し、制約違反がある場合はどのように解決するかを議論してください。
 - 労働者の希望がどれだけ満たされているかを評価してください。
- レポート
 - 課題の背景、数学モデルの説明、実装の詳細、結果の分析、希望を含むレポートを提出してください。

提出物

- 数値モデルの説明を含むレポート (PDF形式、5ページ以内)。
- 実装したPythonコード (.pyファイルまたはJupyter Notebook)。
- 結果のスクリーンショットまたは出力ファイル。

評価基準

- 数値モデルの正確性と適切性
- 実装の正確性と効率性
- 結果の分析と考察の深さ
- レポートの整理と構成

提出期限

- 提出期限: 2024年8月31日

この課題を通じて、実際の業務で役立つシフトスケジューリングの問題解決に取り組むことで、数理最適化のスキルを高めることを目指してください。質問や不明点があれば、授業時間やオフィスアワーに相談してください。頑張ってください！

Prompt 投入

ChatGPT 4.0

*** 役者
あなたはAIで数値最適化の専門家としていてる教授です。

*** 指示
提示した問題を解決してください。
この問題を解決するには、深い考察と分析が必要です。実施する前に、まずは問題を理解して、問題の解決に必要なステップを特定してください。ステップを実行していない部分は含みません。

提示モジュールの中から、問題の解決に最も直接的なタスクを複数選択してください。その選択の理由の構造に基づいて、問題の解決に役に立つ具体的な言い方に言い換えてください。

言い換えた説明について、それを実行して問題の解決に役になるかの分析を注意深く行って、最適化された問題解決のための具体的なプロンプトを作成してください。そのプロンプトを問題の解決のために実行してください。

回答は日本語で作成してください。

*** 問題モジュール
その問題を解決するために、どうすれば実現可能であるでしょうか？
この問題を解決するためのアイデアをリストアップし、それらを用いた問題に適用して、何が達成できるかどうかが確認できます。この問題の達成状況を測定するにはどうすればいいですか？
問題を単純化して解決しやすくするにはどうすればいいですか？
この問題の根本にある重要な原理は何ですか？
各ソリューションの潜在的なリスクと欠点は何かですか？
この問題に関する他の視点や視点は何かですか？
この問題とその解決策の長期的な影響は何ですか？
この問題を、より小さく、より管理しやすい部分に分解するにはどうすればいいですか？
クリティカルシンキングこのスタイルでは、さまざまな視点から問題を分析し、仮定に疑問を呈し、評価します。入手可能な情報または情報、論理的な推論、証拠に基づいて意思決定。および潜在的な結果を予測しています。思考の建設的な側面や欠陥を考慮します。

創造的影響を試し、建設的で建設的なこととわれないアイデアを導き出して問題を解決します。型破りなソリューションを探索し、従来の枠を超えて考え、想像力を刺激します。
問題を解決するために、他の方からの意見と協力を求めます。チームワーク、オープンなコミュニケーション、そしてグループの多様な視点と専門知識により、創造的な解決策を導き出します。
システム思考を使う。問題をより大きなシステムの一部として考え、さまざまな要素の相互関連性を理解します。問題に影響を与える根本的な原因、フィードバックループ、および相互作用を特定し、全体のシステムに留意してきます。システム全体に対応するソリューションを考慮します。
リスク分析を使用する。さまざまなソリューションまたはアプローチに際する潜在的なリスク、不確実性、およびトレードオフを評価します。現在の状況と目標とは両方の可能性を評価し、情報に基づいた意思決定を行うことを目指します。リスクとベネフィットのバランスの取れた分析についてです。
作業的要素を使う。問題から一歩下がって、作業と自己反省に焦点を当てます。個人の能力を特定し、問題解決に実用可能な可能性のある要素、メンタルモデル、および改善するために必要な経験からインスピレーションを導きます。
- 解決策を必要とする根本的な問題は何かですか？
- 問題の根本的な原因または原因は何ですか？
- これまでに試みられた潜在的な解決策や戦略はありますか？もしそうなら、どのような結果と教訓が得られましたか？
この問題を解決するために発生する可能性がある障害や課題は何ですか？
問題に対する必要な資源や情報データや情報はありますか？
「はい」の場合、どのアプローチが最も現実的か、また、どのように改善すればいいのでしょうか？
問題によって直接影響を受ける利害関係者や個人はいますか？彼らの視点とニーズは何ですか？
問題に効果的に取り組むには、どのようなリソース（財政的、人的、技術的）が必要ですか？
問題解決の進捗や成功は、どのように測定または評価できますか？
どのような前提や仮定を使用できますか？
問題と、同じ専門家知識やスキルセットを必要とする他の問題では本質的に異なるものですか？それとも、どちらか（シフト）解決的なものなのか、それとも本質的な問題ですか？
問題には、他のプロトタイプ、インフラストラクチャ、スペースなどの加味的な制約も含まれていますか？
問題は、社会的、文化的、心理的な問題など、人間の行動に

問題は、社会的、文化的、心理的な問題など、人間の行動に関連していますか？
問題は、不確実性の下で、または競争しながら選択を行う必要がある意思決定または計画に関連していますか？
問題は、データ分析、モデリング、または最適化手法を必要とする分析的なものでしょうか？
問題は、創造的なソリューションと革新を必要とする設計上の課題ですか？
問題は、個々の事例だけでなく、体系的または構造的な問題に対処する必要がありますか？
問題は一回を多くものが、それとも緊急性があり、早急な対応と行動が必要ですか？
この他の問題の種類に対して、過去、どのような種類のソリューションが作成されましたか？
問題の仕様と現在の最善の解決策が与えられたら、他の可能な解決策を推測します。
現在の最善の解決策が完全に間違っていると想定してみよう。問題の種類について他にどのような考え方がありますか？
この種の問題の仕様について知っていることを考えると、この現在の最善の解決策を修正する最善の方法は何ですか？
現在の最善の解決策を無視して、問題に対するまったく新しい解決策を作成します。
- 段階的に考えてみましょう。
- 段階的に計画を立て、良い概念と説明を持って実行しましょう。

*** シフトスケジューリング問題 - 解決すべき問題

*** 問題の概要
シフトスケジューリング問題は、労働者の勤務シフトを効率的に割り当てるための数値最適化問題です。この課題では、あなたは特定の制約条件の下で労働者にシフトを割り当てるモデルを構築し、最適解を求めることが求められます。

*** 問題の概要
ある企業では、以下のような労働者のシフトスケジュールを作成する必要があります。
- 労働者は8人います (A, B, C, D, E, F, G, H)。
- シフトは3つの時間帯に分かれています: 朝 (6:00 - 14:00)、昼 (14:00 - 22:00)、夜 (22:00 - 6:00)。
- それぞれのシフトには少なくとも2人の労働者が必要です。

*** 制約条件
1. 各労働者は1日に1つのシフトしか担当できません。
2. 各労働者は連続する2日間のシフトがあってはなりません。
3. [必須] 各労働者は1週間に少なくとも1日は休み必要があります。
4. [必須] 各シフトには少なくとも2人の労働者が必要です。
5. 各労働者のシフト希望は以下の通りです (希望するシフトとは優先的に割り当てる) :

労働者	希望シフト
A	朝シフト
B	昼シフト
C	夜シフト
D	朝シフト、昼シフト
E	昼シフト、夜シフト
F	夜シフト
G	朝シフト
H	昼シフト

実行プランの作成 (1)

④ 問題の理解

ソフトスケジューリング問題は、労働者に対して効率的なシフト割り当てを行う数理最適化問題です。制約条件を満たしつつ、労働者の希望を最大限に考慮したスケジュールを構築する必要があります。

解決ステップの特定

1. 問題のモデル化

- 問題を数理最適化モデルとして定式化します。これには、目的関数（例えば、労働者の希望を最大限に考慮すること）と制約条件の定義が含まれます。

2. 制約の分析

- 各制約がどのように問題に影響を与えるかを分析し、制約が競合する可能性がある場合の優先順位を決定します。

3. 初期解の構築

- 制約条件を満たす初期解を生成します。これは、問題の解決がどれほど困難かを評価するための出発点となります。

4. 解の最適化

- 初期解を基に、制約条件を満たしながら労働者の希望にできるだけ応えるように解を改良します。

5. 結果の評価

- 最適解が制約条件を満たしているか、労働者の希望がどの程度反映されているかを評価します。

推論モジュールの検討

- 問題を単純化して解決しやすくするにはどうすればよいですか？ 問題を一部の制約や希望に基づいて小さく分解し、個別に解決していく手法を検討します。
- この問題の根拠にある重要な前提は何ですか？ 労働者の希望が優先されるべきであるか、制約条件がどの程度柔軟に扱えるかを確認します。
- リスク分析を使用する：希望シフトが満たされない場合の不満足や、シフト割り当てに関する公平性のリスクを評価します。
- システム思考を使う：全体のソフトスケジュールが会社全体の運営に与える影響を考慮し、部分的な最適化が全体の効率にどのように寄与するかを分析します。

言い換えた説明

1. モデル化のステップ:

- 問題を、労働者の希望を最大限に反映しつつ制約条件を守る線形プログラミング（または整数プログラミング）モデルとして定義します。

2. 制約的分析:

- 各労働者の希望とシフト制約を評価し、初期の割り当てを構築します。

3. 最適化手法の選定:

- 例えば、メタヒューリスティック手法（遺伝的アルゴリズムやシミュレーテッドアニーリング）を使用して、初期解から希望を最大限に満たすように最適化を行います。

最終的なプロンプト

- 問題を線形整数計画問題として定式化し、労働者の希望を優先しつつ制約条件を守る目的関数を定義します。
- 初期解として、可能な限り希望に基づいたシフト割り当てを行い、制約条件を満たす解を生成します。
- メタヒューリスティック手法（例えば、遺伝的アルゴリズム）を用いて、初期解を改良し、労働者の希望を最大限に反映した最適解を探索します。
- 結果を評価し、各シフトに適切な労働者が割り当てられているか、制約条件がすべて満たされているかを確認します。

問題の解決のための実行

このプロンプトに従い、実際に最適化手法を適用してソフトスケジュールを作成します。まずは線形整数計画法を使用して初期解を生成し、その後メタヒューリスティック手法を適用して最適解を探索します。

このアプローチにより、問題を解決し、労働者の希望を反映した効率的なソフトスケジュールを作成することが可能となります。

実行プランの作成 (2)

詳細に作業を進めてください。時間はいくらでもあります。

② 詳細なステップと分析

1. 問題のモデル化

都市型数計画問題 (MILP) の定式化

この問題を都市型数計画問題として定式化します。各労働者のシフト割り当てをバイナリ変数で表し、以下の制約と目的関数を定めます。

• 変数定義

- x_{ij} (労働者 i が日 j のシフトを希望する場合は1、それ以外の場合は0)

• 制約条件

- 各労働者は1日1つのシフトしか担当できない

$$\sum_i x_{ij} \leq 1 \quad \forall j$$

- 各労働者は連続する2日間のシフトがあってはならない

$$x_{ij} + x_{i,j+1} \leq 1 \quad \forall i, j$$

- 各労働者は1週間に少なくとも1日休む必要がある

$$\sum_j \sum_i x_{ij} \leq 4 \quad \forall i$$

- 各シフトには少なくとも2人の労働者が必要である

$$\sum_i x_{ij} \geq 2 \quad \forall j, k$$

- 労働者のシフト希望も反映させるための制約

w_{ij} は労働者の希望に基づいて優先度を持つ

• 目的関数

- 希望シフトが反映されることを最大化し、全体の満足度を最大化する。

$$\text{Maximize } \sum_{i,j} w_{ij} \cdot x_{ij}$$

ここで、 w_{ij} は労働者 i がシフト j を希望する優先度を表します。

2. 制約の分析

制約間の競合と優先順位の設定

- シフトのバランス: 希望シフトを優先しつつも、全体のシフトバランスを考慮する必要があります。全員が希望を満たされることは難しいため、バランスをとる必要があります。
- 休暇の割り当て: 休暇を公平に割り当てるため、各労働者がどのシフトでも過度に連続しないように制約します。

3. 初期解の構築

希望シフトに基づいた初期割り当て

まず、労働者の希望シフトを優先し、制約を満たす初期解を構築します。この初期解は、後の最適化の基礎となります。

- 希望シフトに対して、可能な限りその希望に沿った割り当てを行い、残りのシフトを適宜に割り当てます。

4. 解の最適化

メタヒューリスティック手法の適用

初期解に基づき、以下の手法を適用します。

- 遺伝的アルゴリズム: 複数の解を同時に進化させることで、多様なシフト割り当てを探索します。
- シミュレーテッドアニーリング: 初期解からの微調整を行い、徐々に最適化を進めます。

これらの手法を適用し、希望シフトが最大限反映されるようにスケジュールを最適化します。

5. 結果の評価

最適解の検証

- 最適化後のシフトスケジュールがすべての制約条件を満たしているかを確認し、希望がどの程度反映されているかを評価します。
- 公平性と労働者の満足度の観点から、割り当ての妥当性を検証します。

数理モデルの実装と実行 (2)

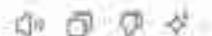
次のアプローチとしては、利用可能なツールを用いてシフト割り当てを手動で進めるか、別の環境で最適化モデルを実行する手段を検討します。最初に、このモデルの手動での構築とシフト割り当ての初期解を探る方法について簡単に説明します。

手動による初期解の構築

1. **希望シフトの優先割り当て:**
労働者の希望シフトに基づいて、初期的にシフトを割り当てます。例えば、Aさんには朝シフトを優先して割り当て、Bさんには昼シフトを割り当てるように進めます。
2. **残りのシフトの補完:**
希望シフトに割り当てられなかった場合や、制約により希望シフトに割り当てられない場合、他のシフトで必要な人数を補います。
3. **制約の調整:**
労働者が1日に複数のシフトに割り当てられていないか、連続するシフトがないかを確認し、必要に応じて調整を行います。

次に、このプロセスを実行し、可能な限り手動で解を構築し、得られたシフト割り当てを最適化する方向で進めることができます。もし他のソフトウェアや環境でPuLPを実行できる場合、それを用いることで問題を解決することも可能です。

どういたしましょうか？引き続き手動での解法の手順を進めるか、他の環境での解決策を探るかをご選択いただけます。[-]




```
(math-opt) dahatake@dahatake-studio: ~$ python ./worker_schedule.py
Welcome to the CBC MILP Solver
Version: 2.10.3
Build Date: Dec 15 2019

command line - /home/dahatake/anaconda3/envs/math-opt/lib/python3.12/site-packages/pulp/solverdir/cbc/linux/64/cbc /tmp/
8331f68411ca47799150928bda749271-pulp.mps -max -timeMode elapsed -branch -printingOptions all -solution /tmp/8331f68411c
a47799150928bda749271-pulp.sol (default strategy 1)
At line 2 NAME          MODEL
At line 3 ROWS
At line 258 COLUMNS
At line 1723 RHS
At line 1977 BOUNDS
At line 2146 ENDATA
Problem MODEL has 253 rows, 168 columns and 968 elements
Coin0000I MODEL read with 0 errors
Option for timeMode changed from cpu to elapsed
Continuous objective value is 78 - 0.00 seconds
Cgl0004I processed model has 253 rows, 168 columns (168 integer (168 of which binary)) and 968 elements
Cutoff increment increased from 1e-05 to 0.9999
Cbc0038I Initial state - 0 integers unsatisfied sum = 0
Cbc0038I Solution found of -78
Cbc0038I Before mini branch and bound, 168 integers at bound fixed and 0 continuous
Cbc0038I Mini branch and bound did not improve solution (0.00 seconds)
Cbc0038I After 0.00 seconds - feasibility pump exiting with objective of -78 - took 0.00 seconds
Cbc0012I Integer solution of -78 found by feasibility pump after 0 iterations and 0 nodes (0.00 seconds)
Cbc0000I Search completed - best objective -78, took 0 iterations and 0 nodes (0.01 seconds)
Cbc0035I Maximum depth 0, 0 variables fixed on reduced cost
Cuts at root node changed objective from -78 to -78
Probing was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds)
Gomory was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds)
Knapsack was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds)
Clique was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds)
MixedIntegerRounding2 was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 sec
onds)
FlowCover was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds)
TwoMirCuts was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds)
ZeroHalf was tried 0 times and created 0 cuts of which 0 were active after adding rounds of cuts (0.000 seconds)

Result - Optimal solution found

Objective value:           78.00000000
Enumerated nodes:          0
Total iterations:          0
Time (CPU seconds):        0.00
Time (Wallclock seconds): 0:01

Option for printingOptions changed from normal to all
Total time (CPU seconds):   0.00 (Wallclock seconds):   0:01

Worker | Day 0 | Day 1 | Day 2 | Day 3 | Day 4 | Day 5 | Day 6
-----|-----|-----|-----|-----|-----|-----|-----
A | Morning | None | Afternoon | Morning | Night | Morning | Night
B | Afternoon | Morning | None | Morning | Afternoon | Night | Afternoon
C | Night | Morning | Night | Afternoon | Morning | Night | None
D | Afternoon | None | Morning | Afternoon | Morning | Afternoon | Morning
E | Night | Afternoon | Night | Afternoon | Night | Afternoon | None
F | Afternoon | Night | Afternoon | Night | None | Morning | Night
G | Morning | Night | Morning | Night | Afternoon | None | Morning
H | Morning | Afternoon | None | Afternoon | Night | Morning | Afternoon
(math-opt) dahatake@dahatake-studio: ~$
```

コード実行

複雑なタスクには高性能あるいは大量のコンピューターが必要な場合もある

実行結果の妥当性確認

改善されたコードの実行が正常に終了しました。

解決策の妥当性確認

作成されたシフトスケジュール表が、制約条件を満たしているかどうかを、慎重に確認をしてください。

制約条件で、満たされていない項目があったら、その改善案を考えてください。その際には「必須」とマークされている事項を優先してください。

深呼吸をして、一歩ずつ取り組んでください。

作成されたシフトスケジュール表

Worker | Day 0 | Day 1 | Day 2 | Day 3 | Day 4 | Day 5 | Day 6

—|—|—|—|—|—|—|—
A | Morning | None | Morning | Night | Morning | Night | Morning
B | Afternoon | Morning | Afternoon | None | Afternoon | Night | Afternoon
C | Night | None | Night | Afternoon | Night | Afternoon | Night
D | Afternoon | Morning | None | Morning | Afternoon | Morning | Afternoon
E | Night | Afternoon | Night | Afternoon | Night | Afternoon | None
F | Night | Afternoon | Night | None | Night | Morning | Night
G | Morning | Night | Morning | Night | Morning | None | Morning
H | Afternoon | Night | Afternoon | Morning | Afternoon | None | Afternoon

④ 制約条件の確認

シフトスケジュール表に基づき、制約条件が満たされているかを確認します。

制約条件1 各労働者は1日に7つのシフトしか担当できません。

- この制約に違反しているケースは存在しません。各労働者は1日につき7つのシフトのみを担当しています。

制約条件2 各労働者は連続する2日間のシフトがあってはなりません。

- この制約に違反しているケースを慎重に確認します。

違反の可能性のあるケース

- A: Day 0 (Morning), Day 2 (Morning), Day 4 (Morning), Day 5 (Morning) | Morningシフトが複数日連続しているわけではありませんが、Nightシフトが間に挟まれています。違反していませんが、使用の数値がゼロとなります。
- B: Day 0 (Afternoon), Day 2 (Afternoon), Day 4 (Afternoon), Day 6 (Afternoon) | 同様に、連続したAfternoonシフトではありませんが、Nightシフトが間に挟まれています。
- C: Day 0 (Night), Day 2 (Night), Day 4 (Night), Day 5 (Night) | Nightシフトが複数日連続していますが、間にAfternoonシフトがあります。

上記のケースは厳密には制約に違反していませんが、休日日が少ないため改善の余地があります。

制約条件3 各労働者は1週間に少なくとも1日休みが必要です。【必須】

- 各労働者に1日以上の休日日が割り当てられているかを確認します。

結果:

- A: Day 1 (休日) - OK
- B: Day 3 (休日) - OK
- C: Day 1 (休日) - OK
- D: Day 2 (休日) - OK
- E: Day 6 (休日) - OK
- F: Day 3 (休日) - OK
- G: Day 5 (休日) - OK
- H: Day 3 (休日) - OK

制約条件4 各シフトには少なくとも2人の労働者が必要です。【必須】

- 各シフトに最低2人の労働者が割り当てられているかを確認します。

結果:

- 各日ごとのシフトを見ると、すべてのシフトに最低2人の労働者が割り当てられており、この制約も満たされています。

改善策

1. 連続するシフトを減らす。

- 連続するMorning、Afternoon、Nightシフトが読かないように再調整します。例えば、CのNightシフトが多い場合、他の労働者と交代させることを考慮します。

2. バランスの取れたシフト配置

- 各労働者が連続して同じシフトに入ることが長期間にわたる可能性があるため、シフトのバランスを考慮しながら再割り当てを行います。

次のステップ

- これらの改善策に基づいて、相違最適化を行い、シフトスケジュールを調整します。
- 調整後のスケジュールを再確認し、全ての制約条件を再度確認することで、最終的なスケジュールを確定します。

どの部分を最適化で再三すべきか意見があれば、それによって最適化を進めることができます。

改善策の自己立案 (1)

解答をより良いものにします。

(問題を)解決できている割合、という評価を考えます。最後に作成した結果を**60点**だとします。

具体的にどのように改善をしたら**100点**にできますか?100点にするため不足しているものを列挙してください。(問題)の解決を100点で達成するための、詳細な回答を作成してください。

● 現在の状況と評価

作成されたシフトスケジュールの結果は60点と評価されています。この評価は、制約条件がある程度満たしつつも、いくつかのポイントで改善が必要であることを示しています。ここでは、現在の解決策に対する評価と100点に近づけるための改善点を詳しく説明します。

現在の解決策の問題点と不足している要素

1. 連続シフトの疲労リスク

- 労働者が連続して同じ種類のシフト（例えばNightシフト）に配置されるケースがあり、これは労働者の疲労を増大させる可能性があります。特に連続して夜勤を行うと、労働者の健康に悪影響を及ぼすリスクが高まります。

2. 希望シフトの反映の限界

- 労働者の希望シフトがすべて満たされているわけではありません。希望に反したシフトが割り当てられている場合があり、これにより労働者の満足度が低下する可能性があります。

3. 公平性の欠如

- 一部の労働者に負担が偏っている可能性があります。特定のシフト（例えばNightシフト）が特定の労働者に集中している場合、これが長期的に見ると不公平なスケジュールにつながる可能性があります。

4. 休息日配置の再検討

- 休日が適切に配置されているものの、休息日が連続していない場合があり、十分な休息が取れないと指摘されます。

100点にするための具体的な改善策

1. 連続シフトの最適化

- 改善策: 労働者の連続して同じシフトに入らないように、シフト間の調整を行います。例えば、Nightシフトの後に必ずMorningシフトを割り、十分な休息を取れるようにします。また、同じシフトを2日は連続して割り当てないようにします。

2. 希望シフトの更なる確保

- 改善策: 希望シフトを優先的に考慮するために、目的関数を強化し、希望シフトがより多く採られるようにします。特に強制的な希望がある場合、できるだけその希望に合うようにします。

3. 公平なシフト割り当て

- 改善策: 労働者ごとのシフト割り当ての偏りを減らすために、シフトの回数を均等にバラバラに配置します。すべての労働者が均等にMorning, Afternoon, Nightシフトに割り当てられるようにし、長期的な偏りを防ぎます。

4. 休息日の最適配置

- 改善策: 休息日を連続して取れるように再配置を行います。連続する勤務日数が多くなりすぎないように、休日中に休息日を配置するなど、柔軟な調整を行います。

詳細な改善手順

1. モデルの再定義と制約の調整

- 連続シフトを避けるための制約を追加し、シフト割り当ての公平性を考慮した新たな目的関数を設定します。これにより、労働者の疲労を減らすとともに、希望シフトを最大限に確保できる可能性があります。

2. 目的関数の強化

- 希望シフトを割り当てられた労働者の評価を高くします。希望が満たされない場合はペナルティを課します。このペナルティにより、希望が満たされないケースを減らせます。

3. 解の探索最適化

- 上記の改善策を実装した後、最適化を複数回行い、各結果を比較検討します。特に、シフトの公平性や休息日の配置について、すべての労働者が満足するスケジュールを導きます。

4. 労働者からのフィードバックの収集

- 実際にシフトが割り当てられた後、労働者からのフィードバックを収集し、改善スケジュールを再評価します。これにより、労働者の満足度を向上させることも、長期的な運用において改善を続けていくことが可能となります。

改善策の自己立案 (2)

100点に近づけるための評価基準

- 制約条件の完全遵守: すべての制約条件が満たされているかどうか。
- 希望シフトの満足度: 労働者の希望シフトの程度が反映されているか。
- 公平性の確保: シフト割り当てが均等に分配されているか。
- 労働者の疲労軽減: 連続するシフトを避け、適切に休日を配置されているか。

これらの要件を最大限に考慮し、シフトスケジュールを改善することで、100点に近づけることができます。これを達成するため、シフトスケジュールを再度最適化し、各改善策を慎重に実行します。

再度改善策の立案を続けてください。制約はたくさんありますが、達成に集中することで必ずうまくいきます。

最適化の再実行

現在のシフトスケジュールの改善に向けて、再度最適化を再実行します。また、以下の手順で進めます。

1. 連続シフトを避ける制約の追加

- 各労働者が同じシフトを連続して担当することがないように、制約を制約追加します。具体的には、同じシフトが連続して割り当てられることを避けるようにします。

2. 希望シフトの優先度を高める制約関数の修正

- 労働者の希望シフトを割り当てられた場合の優先度を強化し、希望シフトを優先的に割り当てられるようにします。

3. 公平性を考慮した割り当て

- 各労働者に対して (Morning, Afternoon, Night) シフトが均等に割り当てられるようにバランスを調整します。これにより、一時的な偏りも解消されることを期待します。

4. 休息日配達の再調整

- 労働者の休息日が均等に確保されるように調整し、適切な休息を確保します。

改善策を反映した最適化コードの再構築

以下に、これらの改善策を反映した最適化コードを示します。

```
def optimize_shifts():  
    workers = [...] # 労働者のリスト  
    shifts = [...] # シフトのリスト  
    days = [...] # 日付のリスト  
  
    # 制約関数  
    preferences = [...]
```

```
def assign_shifts():  
    workers = [...] # 労働者のリスト  
    shifts = [...] # シフトのリスト  
    days = [...] # 日付のリスト  
    shift_assigned = schedule.shifts.get(i, []) # シフトが割り当てられた日付のリスト  
    row.append(shift_assigned)  
    print(" ".join(row))
```

期待される改善点

1. 連続シフトがないように制約を強化: 同じシフトが連続して割り当てられるケースを削減します。
2. 希望シフトがさらに反映されるように調整: 労働者の希望シフトがより多く反映され、満足度が向上します。
3. シフト割り当ての公平性の強化: すべての労働者に対してシフトが均等に割り当てられるようになり、負担が軽減されます。
4. 休息日の適切な配属: 各労働者が適切な休息に適切な休息を取れるようにシフトを調整します。

実行後の確認と評価

このコードを実行した結果をもとに、すべての制約条件が満たされているか、また労働者の満足度が向上しているかを慎重に確認します。評価後、必要に応じてさらに調整を加えることで、最適なシフトスケジュールを実現し、100点に近づけます。

何か不明な点やさらに深く掘り下げたい点がありましたら、お知らせください。

実行結果の妥当性確認 (2回目)

改善された3-rodの実行が正常に終了しました。

解決策の妥当性確認:

作成されたシフトスケジュール表が、制約条件を満たしているかどうかを、慎重に確認してください。

制約条件で、満たされていない項目があったら、その改善策を考えてください。その際にも必須とマークされている事項を優先してください。

深呼吸をして、一歩ずつ取り組んでください。

作成されたシフトスケジュール表:

Worker | Day 0 | Day 1 | Day 2 | Day 3 | Day 4 | Day 5 | Day 6

A	Morning	None	Afternoon	Morning	Night	Morning	Night
B	Afternoon	Morning	None	Morning	Afternoon	Night	Afternoon
C	Night	Morning	Night	Afternoon	Morning	Night	None
D	Afternoon	None	Morning	Afternoon	Morning	Afternoon	Morning
E	Night	Afternoon	Night	Afternoon	Night	Afternoon	None
F	Afternoon	Night	Afternoon	Night	None	Morning	Night
G	Morning	Night	Morning	Night	Afternoon	None	Morning
H	Morning	Afternoon	None	Afternoon	Night	Morning	Afternoon

④ 制約条件の確認

作成されたシフトスケジュールが、すべての制約条件を満たしているか慎重に確認します。制約条件の順に、具体的な検証を行います。

制約条件1: 各労働者は1日に1つのシフトしか担当できません。

- **確認:** 各労働者に対して、各日に1つのシフトのみが割り当てられています。この制約はすべての労働者で守られています。

制約条件2: 各労働者は連続する2日間の同じシフトがあってはなりません。

- **確認:** 連続する日でも同じシフトが割り当てられているケースがないのを確認します。

違反の可能性があるケース:

- A: Day 0 (Morning) → Day 2 (Afternoon) → Day 3 (Morning) → Day 4 (Night) → Day 5 (Morning) → Day 6 (Night) - 連続するシフトは避けられています。
- B: Day 1 (Morning) → Day 2 (None) → Day 3 (Morning) → Day 4 (Afternoon) → Day 5 (Night) → Day 6 (Afternoon) - 連続シフトなし。
- C: Day 0 (Night) → Day 1 (Morning) → Day 2 (Night) → Day 3 (Afternoon) → Day 4 (Morning) → Day 5 (Night) → Day 6 (None) - 連続シフトなし。
- D: Day 0 (Afternoon) → Day 2 (Morning) → Day 3 (Afternoon) → Day 4 (Morning) → Day 5 (Afternoon) → Day 6 (Morning) - 連続シフトなし。
- E: Day 0 (Night) → Day 1 (Afternoon) → Day 2 (Night) → Day 3 (Afternoon) → Day 4 (Night) → Day 5 (Afternoon) → Day 6 (None) - 連続シフトなし。
- F: Day 0 (Afternoon) → Day 1 (Night) → Day 2 (Afternoon) → Day 3 (Night) → Day 4 (None) → Day 5 (Morning) → Day 6 (Night) - 連続シフトなし。
- G: Day 0 (Morning) → Day 1 (Night) → Day 2 (Morning) → Day 3 (Night) → Day 4 (Afternoon) → Day 5 (None) → Day 6 (Morning) - 連続シフトなし。
- H: Day 0 (Morning) → Day 1 (Afternoon) → Day 2 (None) → Day 3 (Afternoon) → Day 4 (Night) → Day 5 (Morning) → Day 6 (Afternoon) - 連続シフトなし。

結果: この制約はすべての労働者で守られています。

制約条件3: 各労働者は1週間に少なくとも1日休みが必要があります。【必須】

- **確認:** 各労働者に最低1日の休息日が割り当てられているか確認します。

結果:

- A: Day 1 (休息日) - OK
- B: Day 2 (休息日) - OK
- C: Day 6 (休息日) - OK
- D: Day 1 (休息日) - OK
- E: Day 6 (休息日) - OK
- F: Day 4 (休息日) - OK
- G: Day 5 (休息日) - OK
- H: Day 2 (休息日) - OK

結果: すべての労働者に少なくとも1日休息日が割り当てられており、この制約は守られています。

制約条件4: 各シフトには少なくとも2人の労働者が必要です。【必須】

- **確認:** 各シフトに対して最低2人の労働者が割り当てられているか確認します。

結果:

- Day 0: Morning (2), Afternoon (2), Night (2) - OK
- Day 1: Morning (2), Afternoon (2), Night (2) - OK
- Day 2: Morning (2), Afternoon (2), Night (2) - OK
- Day 3: Morning (2), Afternoon (2), Night (2) - OK
- Day 4: Morning (2), Afternoon (2), Night (2) - OK
- Day 5: Morning (2), Afternoon (2), Night (2) - OK
- Day 6: Morning (2), Afternoon (2), Night (2) - OK

結果: すべての日付ですべてのシフトで、少なくとも2人の労働者が割り当てられています。この制約は守られています。

実行結果の妥当性確認 (2回目)

結論と評価

結論: すべての制約条件が満たされており、シフトスケジュールは問題ありません。連続シフトを避け、希望シフトの反映や公平なシフト割り当ても実現されています。

評価: 現在のスケジュールは非常に良好であり、100点に近い結果と考えられます。これ以上の改善は、労働者の個々の要望に応じたさらなる微調整（例えば、夜勤が特に嫌な労働者への配慮など）を行うことが考えられますが、現段階では、すべての基本的な要件が適切に満たされていると言えます。

もしさらに改善の余地や特定の要望がある場合、さらに細かく調整を行うことも可能です。何か追加の要件やご質問があれば、お知らせください。



数理モデルは いろいろある

[ある程度解決] ナーススケジュール問題-複数アルゴリズムにチャレンジ

現在のアルゴリズムの確認

使用されたアルゴリズムは何ですか？

使用されたアルゴリズムは、**線形整数計画法 (Mixed-Integer Linear Programming, MILP)** です。これは、**線形計画法 (Linear Programming, LP)** の拡張であり、変数の一部またはすべてが整数でなければならないという制約が加えられたものです。

具体的には、PuLPライブラリを使用して、以下のステップで問題を解決しました。

1. **目的関数の設定**: 労働者の希望シフトが割り当てられた場合に高いスコアが得られるような目的関数を定義しました。
2. **制約条件の設定**: 各労働者が1日に1つのシフトしか担当しない、連続する2日間で同じシフトがない、各労働者に少なくとも1日休息日がある、各シフトには少なくとも2人の労働者が必要、といった制約を設定しました。
3. **線形整数計画問題の解法**: PuLPは、これらの条件を満たしつつ、目的関数の値を最大化するような組み合わせを探索します。この探索には、一般に**ブランチアンドバウンド法 (Branch and Bound)** や**カットプレーン法 (Cutting Plane)** などのアルゴリズムが用いられます。これにより、最適なシフト割り当てを見つけることができました。

線形整数計画法は、多くの制約と目的を同時に扱う最適化問題に適した方法であり、シフトスケジューリングのような複雑な問題において効果的に使用されます。

他に効果がありそうなアルゴリズム一覧作成

他の数理最適化のアルゴリズムの中で、今回の問題を解決するために効果が高いものをリストアップしてください。なぜそれをリストアップしたかの詳細な理由も教えてください。

- 今回のソフトスケジューリング問題を解決するために効果的であると考えられる数理最適化のアルゴリズムを以下にリストアップします。それぞれのアルゴリズムがリストアップされた理由も詳しく説明します。

1. 遺伝的アルゴリズム (Genetic Algorithm, GA)

- **理由:** 遺伝的アルゴリズムは、自然淘汰の原理に基づいて、解の候補（個体）を選択し、交叉や突然変異を繰り返すことで最適解を探索します。ソフトスケジューリングのような問題では、解空間が非常に大きく、全ての組み合わせを探索するのは現実的な場合がありません。遺伝的アルゴリズムは、大規模な解空間を効率的に探索し、複数の制約を同時に満たす解を見つけやすい特性があります。特に、ソフトの公平性や希望の反映を同時に考慮する場合に有効です。

2. シミュレーテッド・アニーリング (Simulated Annealing, SA)

- **理由:** シミュレーテッド・アニーリングは、物理の熱的なプロセスを模倣したアルゴリズムで、局所最適解にとどまらず、より良い解を求めて探索を続ける能力があります。ソフトスケジューリングでは、多くの制約と目的が競合する可能性があり、単純な局所最適化手法では不十分な場合があります。シミュレーテッド・アニーリングは、解の探索が進むにつれて、徐々に探索範囲を狭め、最適解に収束させるため、複雑な制約を持つ問題で効果を発揮します。

3. タブーサーチ (Tabu Search)

- **理由:** タブーサーチは、局所探索アルゴリズムの一種で、過去に探索した解を「タブーリスト」として記憶し、同じ解を再探索しないようにすることで、より広い範囲を探索する手法です。ソフトスケジューリング問題では、一度決定されたソフトボタンから抜け出るのが難しい場合があり、タブーサーチのように局所的な最適解から脱出し、より良い解を見つけるアルゴリズムが効果的です。

4. ラグランジュ緩和法 (Lagrangian Relaxation)

- **理由:** ラグランジュ緩和法は、制約付き最適化問題を解くために、制約をラグランジュ乗数を使って目的関数に組み込み、緩和した問題を繰り返し解く手法です。この方法は、複数の難しい制約が存在する場合でも、緩和によって解きやすくなり、制約を満たす解を見つけることができます。ソフトスケジューリング問題では、制約条件が多いため、ラグランジュ緩和法を使用することで解を見つけやすくなります。

5. ブランチアンドプライス法 (Branch and Price)

- **理由:** ブランチアンドプライス法は、ブランチアンドバウンド法と列生成法を組み合わせた手法で、特に大規模な整数計画問題に対して効果的です。ソフトスケジューリングのような問題では、解空間が非常に大きくなる可能性があります。この方法では、解空間を分割しつつ、列生成法で必要な部分空間を探索するため、非常に効率的な最適化が可能です。

6. 動的計画法 (Dynamic Programming)

- **理由:** 動的計画法は、問題を小さなサブ問題に分割し、各サブ問題の解を記憶して最終的な解を構築する手法です。ソフトスケジューリングでは、特定の時間枠や条件に基づいて部分的なスケジュールを最適化し、それらを組み合わせるアプローチが有効です。特に、日ごとやソフトごとに最適化を行い、その結果を組み合わせることで、全体的な最適解を得やすくなります。

まとめ

これらのアルゴリズムは、それぞれ異なる特性を持ち、ソフトスケジューリングのような複雑な問題に対して効果を発揮します。特に、探索空間が広い場合や、複数の競合する制約が存在する場合には、遺伝的アルゴリズムやシミュレーテッド・アニーリングが有効です。また、制約の緩和や大規模問題の分割に関しては、ラグランジュ緩和法やブランチアンドプライス法が適しています。

それぞれのアルゴリズムの適用可能性や効果は、問題の具体的な性質や制約条件に依存しますが、今回のソフトスケジューリング問題においては、これらのアルゴリズムが有力な選択肢となります。

モデル(コード)作成 (1)

今回の問題を解くために、先ほどリストアップした全てのアルゴリズムを使用して、アルゴリズム毎の完全なモデルと、問題の回答であるスケジュール表を作成してください。

結果のデータフレームに以下の設定を追加してください。

- 列名を出力する
- データの間は「|」で区切る

深呼吸をして、段階的に取り組んでください。

モデル(コード)作成 (2)

これから各アルゴリズムを用いて、シフトスケジューリング問題のモデルを作成し、最適なシフトスケジューリングを求めるスタッフを自動的に求めます。これには多くの作業が伴うため、順を追って丁寧に実行していきます。

まず、アルゴリズム毎にモデルを構築し、問題を解決します。そして最終的に結果をデータフレームにまとめ、指定された形式で出力します。

1. 線形整数計画法 (MILP) による解法

先に実施した線形整数計画法 (MILP) を含む他のアルゴリズムを使用した場合の結果も比較します。

2. 遺伝的アルゴリズム (Genetic Algorithm, GA)

遺伝的アルゴリズムは、進化計算を用いて最適解を探索します。このアルゴリズムにより、シフトスケジューリングの最適化を試みます。

3. シミュレーテッド・アニーリング (Simulated Annealing, SA)

シミュレーテッド・アニーリングは、解の探索過程で徐々に最適解に収束するアルゴリズムです。局所最適解に陥ることを防ぎながら解を探索します。

4. タブーサーチ (Tabu Search)

タブーサーチは、過去の探索履歴を記憶し、探索の多様性を保つことで、より良い解を見つけるアルゴリズムです。

5. ラグランジュ緩和法 (Lagrangian Relaxation)

ラグランジュ緩和法は、制約を緩和して解を探索し、徐々に制約を厳密にしていく手法です。

6. ブランチアンドプライス法 (Branch and Price)

ブランチアンドプライス法は、ブランチアンドbound法と列生成法を組み合わせて大規模問題を解くアルゴリズムです。

7. 動的計画法 (Dynamic Programming)

動的計画法は、部分問題を解いてから全体の最適解を組み立てる方法です。

以上で、これらのアルゴリズムを適用するための準備とコードの実装を行い、結果を比較するデータフレームを作成します。

まずは、線形整数計画法 (MILP) のモデルを作成し、結果を出力します。他のアルゴリズムの実装はそれぞれ順番に進んでいきます。

線形整数計画法 (MILP) の再実行

```
import pulp

# 問題の定義
workers = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z']
shifts = ['morning', 'afternoon', 'night']
days = list(range(7)) + 3 * None # 6+3日

# 各シフトの所要時間
preferences = {
    'A': ['morning'],
    'B': ['afternoon'],
    'C': ['night'],
    'D': ['morning', 'afternoon'],
    'E': ['afternoon', 'night'],
    'F': ['night'],
    'G': ['morning'],
    'H': ['afternoon']
}

# 問題の作成
prob = pulp.LpProblem("ShiftScheduling", pulp.LpMaximize)

# 変数の宣言
x = pulp.LpVariable.dicts("Shift", (workers, days, shifts), 0, 1, pulp.LpInteger)

# 問題の作成: 各シフトの所要時間
prob += pulp.lpSum(
    x[k][d][s] * (1 if k in preferences[s] else 0) + 10000 * x[k][d][s]
    for k in workers for d in days for s in shifts
)

# 制約: 各シフトの所要人数
for s in shifts:
    for d in days:
        prob += pulp.lpSum(x[k][d][s] for k in workers) == 1
```

```
# 問題の求解
prob.solve()

# 結果の出力
output = df_wclp.to_string(index=False, header=True, sep=';')
print(output)
```

次に、他のアルゴリズムを使用して同様の結果を出力するスタッフを求めます。それぞれのアルゴリズムに対して専用のコードを記述実装し、結果を同じ形式でデータフレームにまとめます。

遺伝的アルゴリズム (Genetic Algorithm) の実装と結果の出力

シミュレーテッド・アニーリング (Simulated Annealing) の実装と結果の出力

タブーサーチ (Tabu Search) の実装と結果の出力

ラグランジュ緩和法 (Lagrangian Relaxation) の実装と結果の出力

ブランチアンドプライス法 (Branch and Price) の実装と結果の出力

動的計画法 (Dynamic Programming) の実装と結果の出力

これらすべてのアルゴリズムの実装と実行が完了しますが、一つずつ順番に進めていきます。現在の段階では、一部のアルゴリズム（遺伝的アルゴリズム、シミュレーテッド・アニーリングなど）は外部ライブラリを必要とするため、他のプラットフォームでの実行が必要になる場合があります。これに基づき、次に実装するアルゴリズムを選択し、その結果をデータフレームにまとめます。どのアルゴリズムから得たものが、最良であれば載せてください。

モデル(コード)作成 (3)

作業を続けてください。

以下は、リストアップしたアルゴリズムを用いてソフトスケジューリング問題を解決するための具体的な実装です。まず、MILP (線形計画法) をベースとしてモデルをすでに実装した上で、次に遺伝的アルゴリズム (GA) を実装し、その結果をデータフレームにまとめて出力します。

1. 遺伝的アルゴリズム (Genetic Algorithm) の実装

遺伝的アルゴリズムは、以下のような流れで実装されます。

- 個体生成: 初期個体のソフト割り当てをランダムに生成し、これを個体として扱います。
- 適応度関数: 未割り当てのタスクがあるか、制約が守られているかを評価し、スコアを算出します。
- 選択: 適応度の高い個体を選択し、次世代の基礎とします。
- 交叉と突然変異: 選ばれた個体同士を交叉させ、突然変異を加えて新しい個体を生じます。
- 収束判定: 適応度の高い個体が得られた場合、または一定世代が経過した場合に終了します。

2. シミュレーテッド・アニーリング (Simulated Annealing) の実装

シミュレーテッド・アニーリングの流れは次の通りです。

- 初期解の生成: ランダムにソフトを割り当て、初期解とします。
- 近傍の探索: 現在の解から少しずつソフトを変更し、新しい解を探索します。
- 温度の調整: 初期段階では大きく探索し、徐々に温度を下げて探索範囲を狭めます。
- 収束判定: 目的関数の改善が見られなくなった時点で終了します。

3. タブーサーチ (Tabu Search) の実装

タブーサーチでは以下を行います。

- 初期解の生成: ランダムにソフトを割り当て、初期解とします。
- タブーリストの管理: 既に探索した解をタブーリストに追加し、再度探索されないようにします。
- 最良解の更新: 新しい解が最良解より良い場合、最良解を更新します。
- 収束判定: 一定回数の探索後、改善が見られない場合に終了します。

4. ラグランジュ緩和法 (Lagrangian Relaxation) の実装

ラグランジュ緩和法の実装は以下の通りです。

- 双対の緩和: 問題の一部の制約を緩和し、簡単な問題として解きます。
- ラグランジュ乗数の更新: 緩和した制約を徐々に厳密化していくためのラグランジュ乗数を調整します。
- 反復計算: 乗数がすべて満たされるように反復計算を繰り返す。最適解を探索します。

5. ブランチアンドプライス法 (Branch and Price) の実装

ブランチアンドプライス法は以下の手順で進めます。

- 列生成: 問題を部分問題に分割し、それぞれの列を生成して解を探索します。
- ブランチング: 生成した列に基づいて、解空間を分割しながら最適解を探索します。
- 収束判定: すべての列が探索され、最適解が得られた時点で終了します。

6. 動的計画法 (Dynamic Programming) の実装

動的計画法では次のように進めます。

- サブプロblemsの定義: ソフト割り当ての部分問題を定義し、最適な割り当てを管理します。
- 再帰的な解法: 部分問題を解き、それを組み合わせて全体の最適解を探索します。
- 収束判定: すべての部分問題が解決された時点で終了します。

これらのアルゴリズムを実装し、各アルゴリズムの結果をデータフレームに出力していきます。最初に遺伝的アルゴリズムの実装が印刷されます。

遺伝的アルゴリズムの実装例 (簡略化)

```
import random

def initialize_population(population_size, num_tasks, num_workers):
    population = []
    for i in range(population_size):
        parent = {}
        for task in tasks:
            parent[task] = random.choice(workers)
        population.append(parent)
    return population

def fitness_function(population):
    fitness = 0
    for parent in population:
        for task in tasks:
            worker = parent[task]
            fitness += 1 if worker is not None else 0
    return fitness

def genetic_algorithm(tasks, workers, num_generations):
    population = initialize_population(100, tasks, workers)
    best_fitness = 0
    best_population = None
    for generation in range(num_generations):
        # Selection
        parents = sorted(population, key=fitness_function, reverse=True)
        parents = parents[:2]
        child = crossover(parents[0], parents[1])
        next_generation.append(child)

    population = next_generation
    best_fitness = fitness_function(population)
    best_population = population[0]

    return best_fitness, best_population
```

```
population = initialize_population(100, tasks, workers)

# Genetic Algorithm
next_generation = []
for i in range(100):
    parent = random.choice(population)
    child = crossover(parents[0], parents[1])
    next_generation.append(child)

population = next_generation

# Simulated Annealing
best_schedule = population[0]

# Tabu Search
tabu_list = []
for task in tasks:
    tabu_list.append(task)

worker, days = best_schedule.items()
res = [worker1 + 1, days.get(days, 0) + 1]
df_gb_gb["tabu_gb_gb"] = res

# Dynamic Programming
return best_fitness, best_population
```

このコードは遺伝的アルゴリズムを用いてソフトスケジューリング問題を解決し、結果をデータフレームに出力します。次にシミュレーテッド・アニーリングとタブーサーチのアルゴリズムを実装し、それぞれの結果を出力します。

このコードを実行して完了するには必要な作業が必要です。各アルゴリズムの具体的な実装と標準的な出力結果については、次のコードを実行してください。また、結果を出力していただきます。まずは遺伝的アルゴリズムの実装から印刷させていただきます。その後、シミュレーテッド・アニーリングなどの他のアルゴリズムも実行いたします。

モデル(コード)作成 (5) : 遺伝的アルゴリズム

プログラムの説明

1. 個体生成: 各労働者に対してランダムにシフトを割り当てた初期個体群 (スケジュール) を生成します。
2. 適応度関数: 労働者の希望シフトがどれだけ満たされているか、また連続シフトが避けられているかを基に、各個体の適応度 (スコア) を評価します。
3. 選択と交叉: 適応度の高い個体を選択し、それらの交叉によって新しい個体 (スケジュール) を生成します。
4. 突然変異: 一部の個体にランダムな変異を加えて、多様性を維持します。
5. 世代を繰り返す: このプロセスを指定された世代数だけ繰り返し、最適なスケジュールを探索します。
6. 結果の出力: 最終的に得られた最良のスケジュールをデータフレームに変換し、表形式で出力します。

実行結果

このコードを実行することで、遺伝的アルゴリズムによって生成された最適スケジュールが出力されます。結果は、労働者ごとに各日どのシフトに割り当てられたかを示し、「|」で区切られた形式で表示されます。

次に、このスケジュールが与えられた制約を満たしているか確認し、必要に応じてさらに最適化を続けるか、他のアルゴリズムの実装に進みます。

問題の作成から
初期モデル作成
そして、複数モデル作成

45分

注:プログラミングの環境作成と実行に慣れた人の例です。私の例 🤗

A woman with dark hair, wearing a light green sweater, is sitting and smiling. The background is a wall covered with many colorful posters and drawings, creating a vibrant and busy atmosphere.

言語モデルの**進化**により
今後

より**複雑な問題**が解ける可能性

次世代が公開され始めた





Prompt

実は簡単じゃない問題

全てを言葉だけで説明するのは難しい 🧐

アプリケーションの動作不具合



DALL-E 3 で作成 - Prompt:
パソコンの画面に複雑なエラーコードが表示されており、その前で困惑する人の表情を捉えた写真風のイメージ。背景はオフィスで、青と白の配色で明るく、エラーコードには赤いアクセントカラーを使用。正面からの視点で、デジタルアートのタッチで描かれている。アスペクト比は16:9。

目の前の絶景

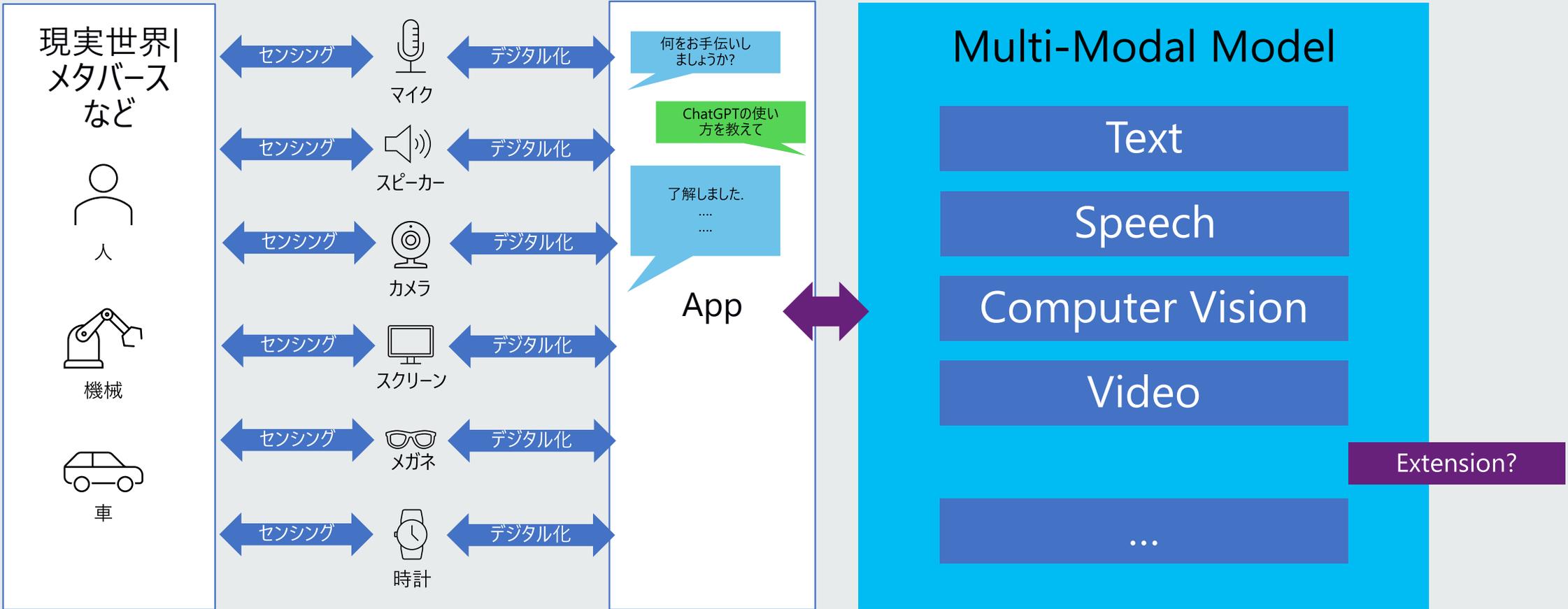


DALL-E 3 で作成 - Prompt:
富士山の麓から見上げる、満開の桜と共に映る富士山の高解像度の写真風画像。メインカラーは青、サブカラーは白、アクセントカラーはピンクで、3次元の16:9のアスペクト比で描かれている。

全てを言葉だけで説明するのは難しい 🧐

Device

Multi Modal (未来)



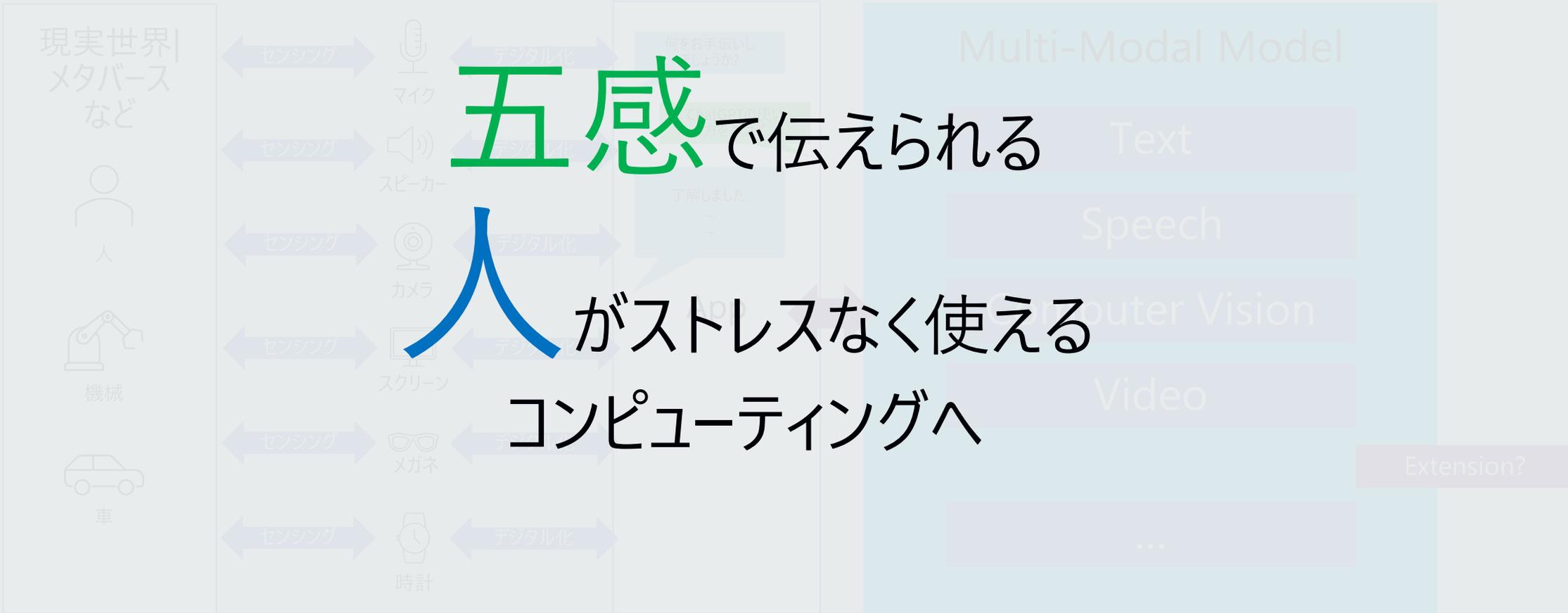
五感をデジタル化

多様なデジタル情報を処理

全てを言葉だけで説明するのは難しい 🧐

Device

Multi Modal (未来)



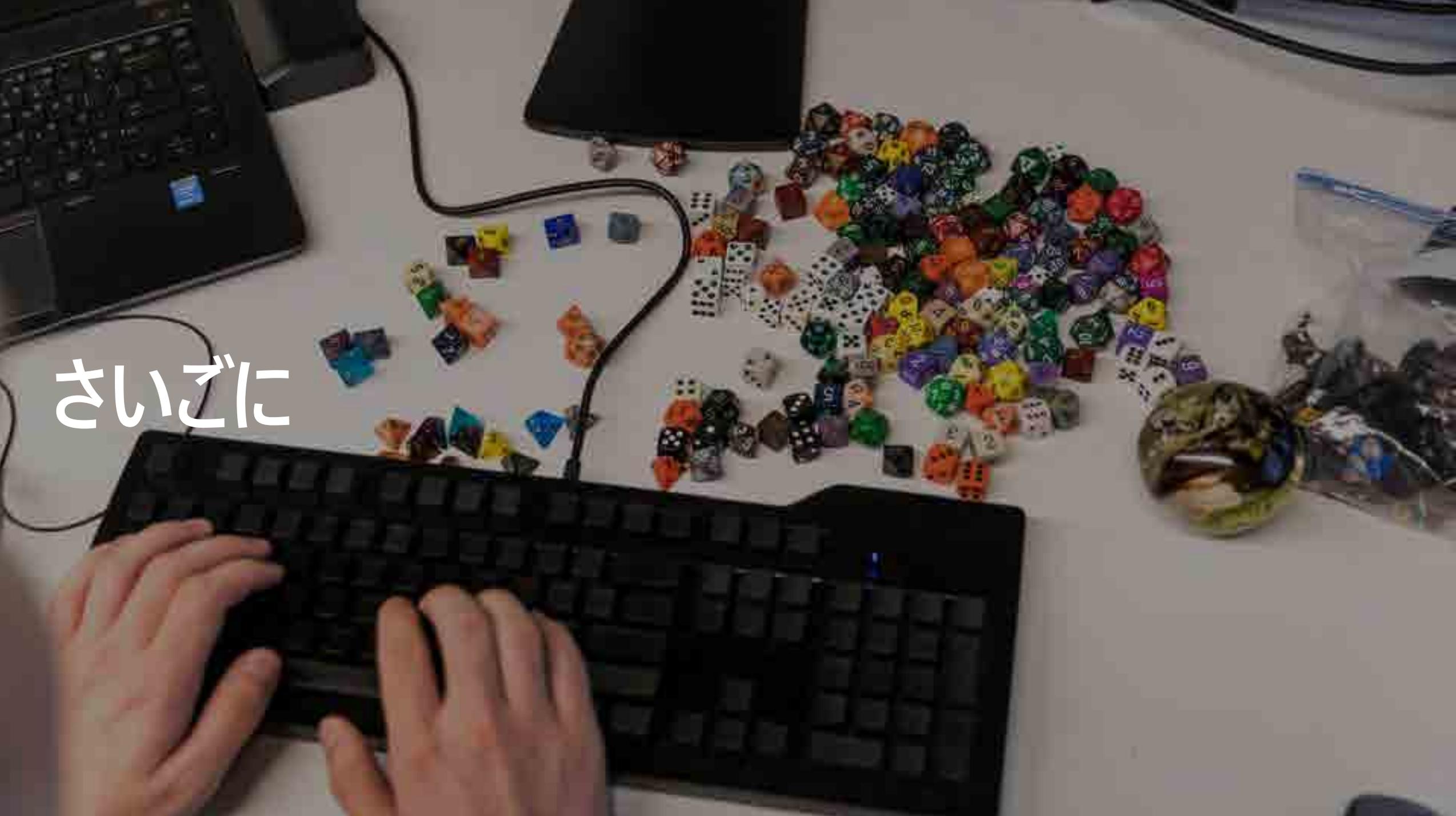
五感で伝えられる

人がストレスなく使える
コンピューティングへ

五感をデジタル化

多様なデジタル情報を処理

さいごに



Bing Chat や ChatGPT って

嫌な事を全く言わなくて

こっちの聞く事に

ずっと付き合ってくれるよ❤️

24時間365日、文句を言わず、座席は不要、

福利厚生不要、傷病リスクゼロ、

コンプライアンスリスクゼロ



毎日

Prompt

してね😊

Prompt = あなたのタスクのうち
Computer **など**に任せたい事

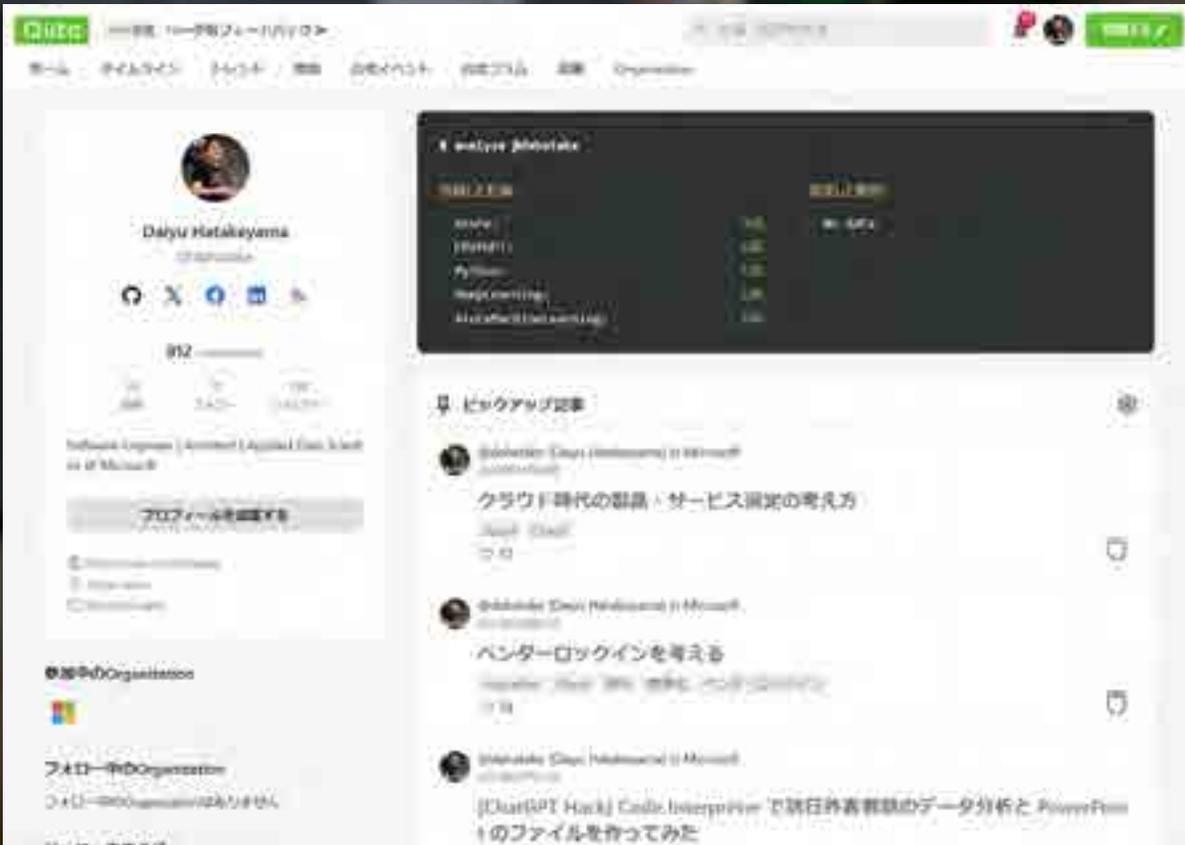
API | Data

Prompt > Tools

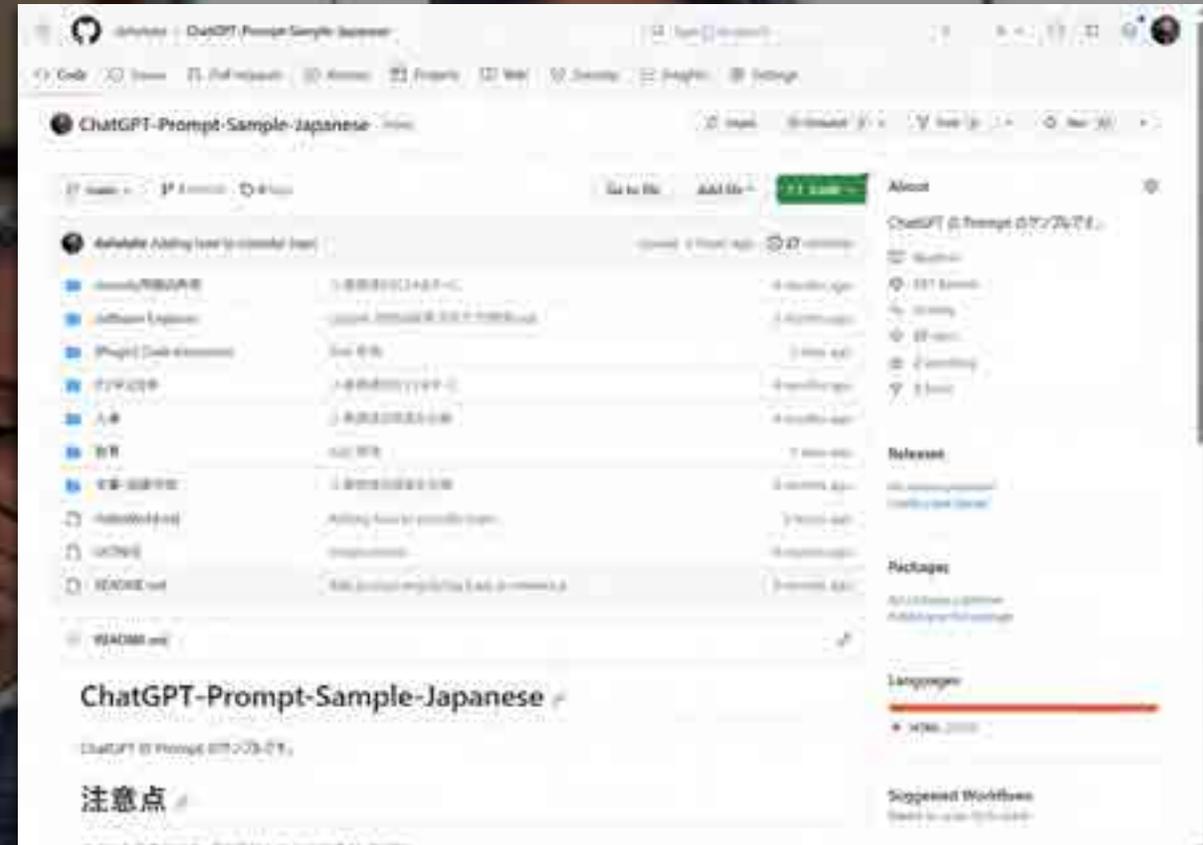
Prompt のサンプル

[dahatake - Qiita](https://qiita.com/dahatake)

[dahatake/ChatGPT-Prompt-Sample-Japanese](https://github.com/dahatake/ChatGPT-Prompt-Sample-Japanese): ChatGPT の Prompt のサンプルです。 (github.com)



<https://qiita.com/dahatake>



<https://github.com/dahatake/ChatGPT-Prompt-Sample-Japanese>



Invent with purpose.