cs-14. AI活用リテラシーとIT応用

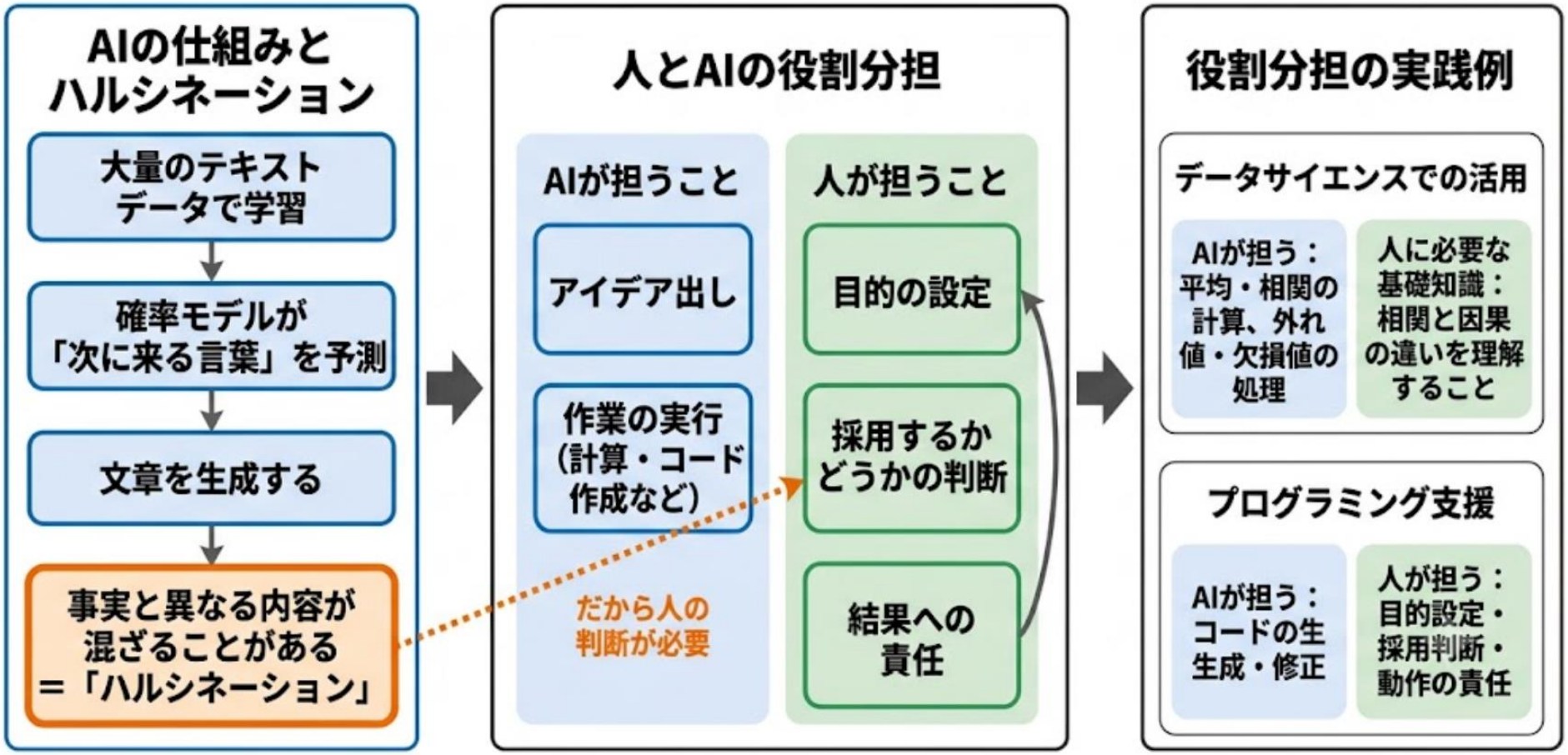
（コンピューターサイエンス）

金子邦彦

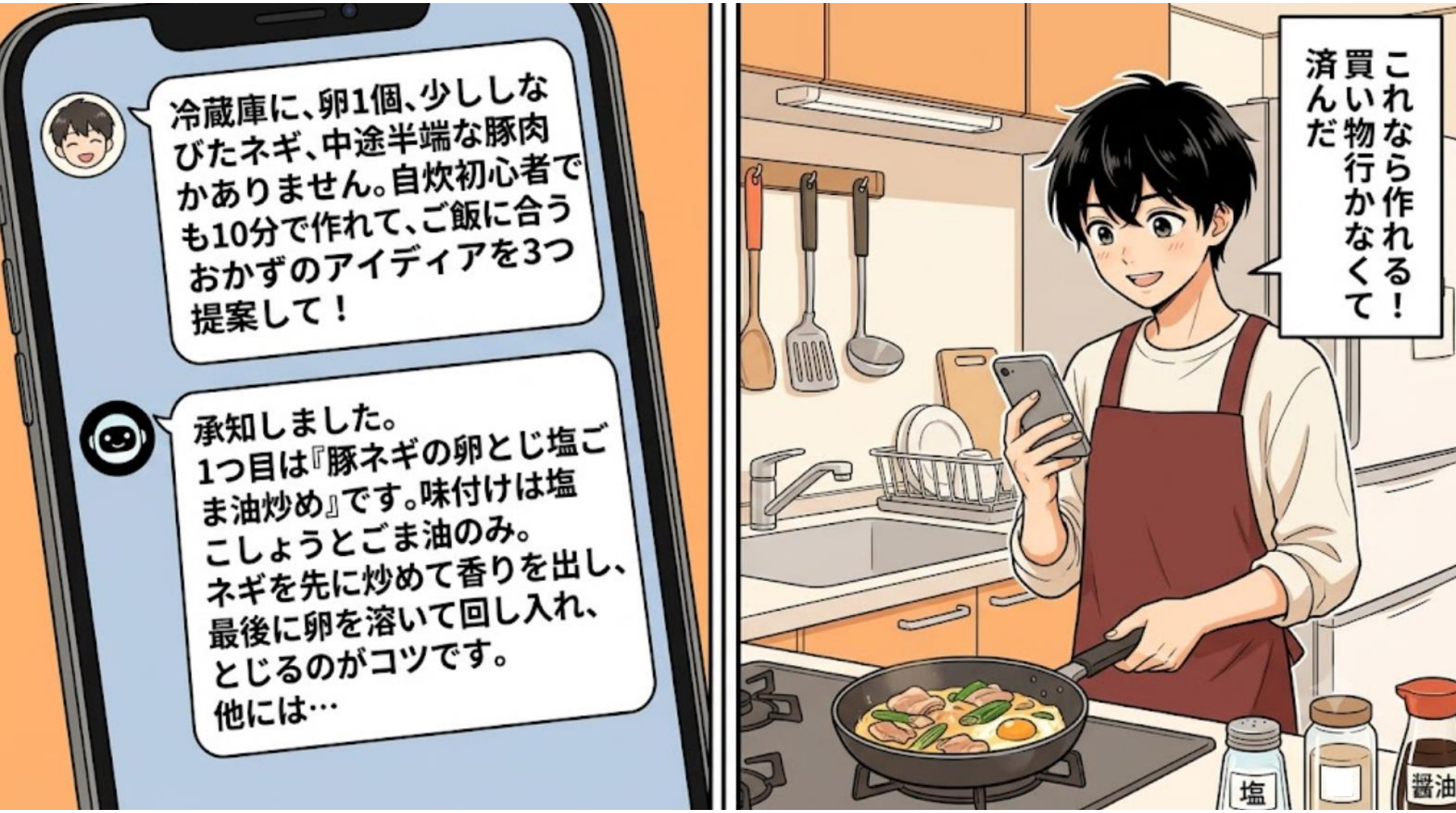
資料中の図などは Google Nano Banana 2 を用いて作成

「コンピューターサイエンス」第14回の内容

対話型AIは『次の言葉』を予測して文章を生成する仕組み。人とAIで役割を分けて使うことが活用の鍵となる



14-1. 対話型AI


冷蔵庫に、卵1個、少ししなびたネギ、中途半端な豚肉かありません。自炊初心者でも10分で作れて、ご飯に合うおかずのアイディアを3つ提案して！
承知しました。
1つ目は『豚ネギの卵とじ塩ごま油炒め』です。味付けは塩こしょうとごま油のみ。
ネギを先に炒めて香りを出し、最後に卵を溶いて回し入れ、とじるのがコツです。
他には…
これなら作れる！買い物行かなくて済んだ
塩
醤油

身近になった対話型 AI

Copilotのようなチャット型AIは、質問に対して自然な文章で応答を返す。この応答は、内部で『大規模言語モデル（LLM）』と呼ばれる仕組みによって生成されている。

この“自然な応答”は、どのような仕組みで作られているのか？
— 次のページで、その仕組みの中心にある『LLM』と『トークン』を見ていく


よし、AIが言うなら
これやってみよう！
え、それ…
まだ早くない？
先週始めたばか
りだよね？
フィトネスAI
高強度トレーニング
30分×3セット
があります！
OK
4月
筋トレ
開始！
CAMPUS
LIFE

AI の回答．こんな経験はありませんか？

聞き方を変えたら、答えの質が変わった

自信を持って答えてくれたが、
本当に正しいのだろうか

なぜ、このようなことが起こるのだろうか？

この資料では、AIの“仕組み”を理解することで、こうした“振る舞い”の理由を読み解いていく

大規模言語モデルとトークンとは

大規模言語モデル（LLM: Large Language Model）

直前の文脈から、次に来やすいトークンを確率的に予測し、それを繰り返して文章を生成するモデル。
テキストデータで学習した言語モデルである。

文脈→予測→追加→繰り返し

トークンとは

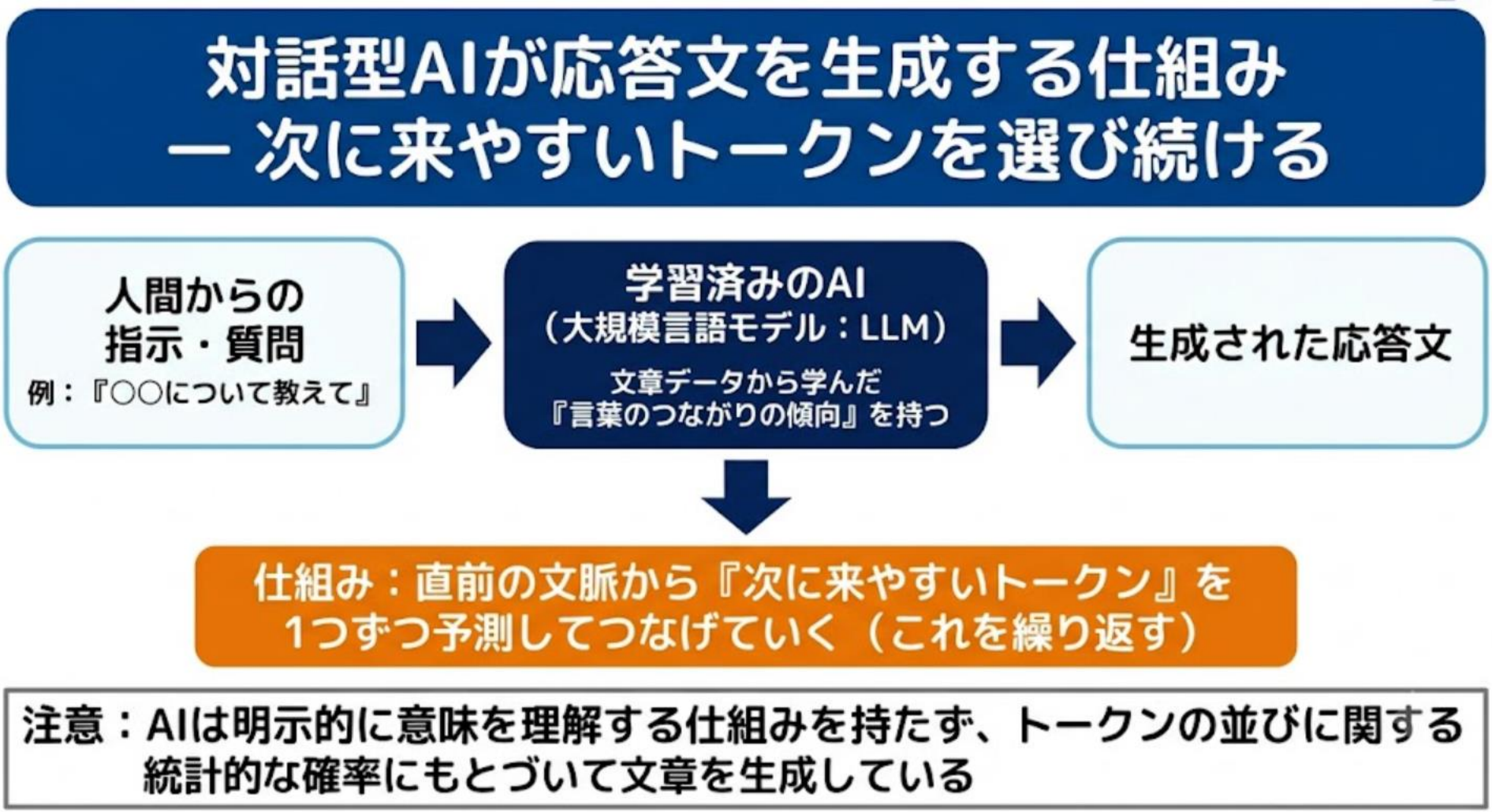
LLMが予測・処理する単位であり、単語そのものではなく、単語または単語の一部に相当する。

対話型AI → 対話 型 AI

※トークンの分割方法はモデルにより異なる

※英語では tokenization → token／ization のように"単語の一部"に分かれることもある

対話型 AI の対話の仕組み



プロンプトと出力の関係 ー なぜ聞き方で答えが変わるのか

AIの予測は直前の文脈全体を手がかりに行われる ー 人間が入力するプロンプトは、その文脈の一部である

入力例：『AIについて教えて』	入力例：『生成AIの仕組みを3つの観点で説明して』
文脈に含まれる情報が少ない（対象や範囲が曖昧）	文脈に含まれる情報が多い（対象や範囲が明確）
次に来やすいトークンの候補が幅広く分散する	次に来やすいトークンの候補が、関連性の高いものに絞られる
出力：一般的・曖昧な内容になりやすい	出力：意図に沿った具体的な内容になりやすい

注意：プロンプトに「正確に回答せよ」と書いても、正確性が保証されるわけではない。プロンプトを具体的・詳細にすることで回答は変化するが、「正しさ」は別のことである

機械学習

機械学習とは、ルールを人が教えるのではなく、データから
コンピューター自身にパターンやルールを見つけさせる仕組みである

これまでのやり方

人がルールを考える
コンピューターはルールに
従って動作するだけ

人がルールを与える

機械学習

データを与える
コンピューターがデータから
パターンやルールを抽出する

分類結果：猫

人はデータを与える

機械学習で人が与えるのはデータであり、ルールそのものではない

対話型AIと機械学習

Database Lab.

対話型AIの仕組み ― 大量のデータから"言葉のつながり"を学ぶ

① 大量の文章（本・Web記事・会話など）を教材として集める

↓

② AIが「どのトークンの次に、どのトークンが来やすいか」の規則性を学ぶ

↓

③ 学んだ傾向は膨大な"数値の設定"としてAIの中に蓄積される

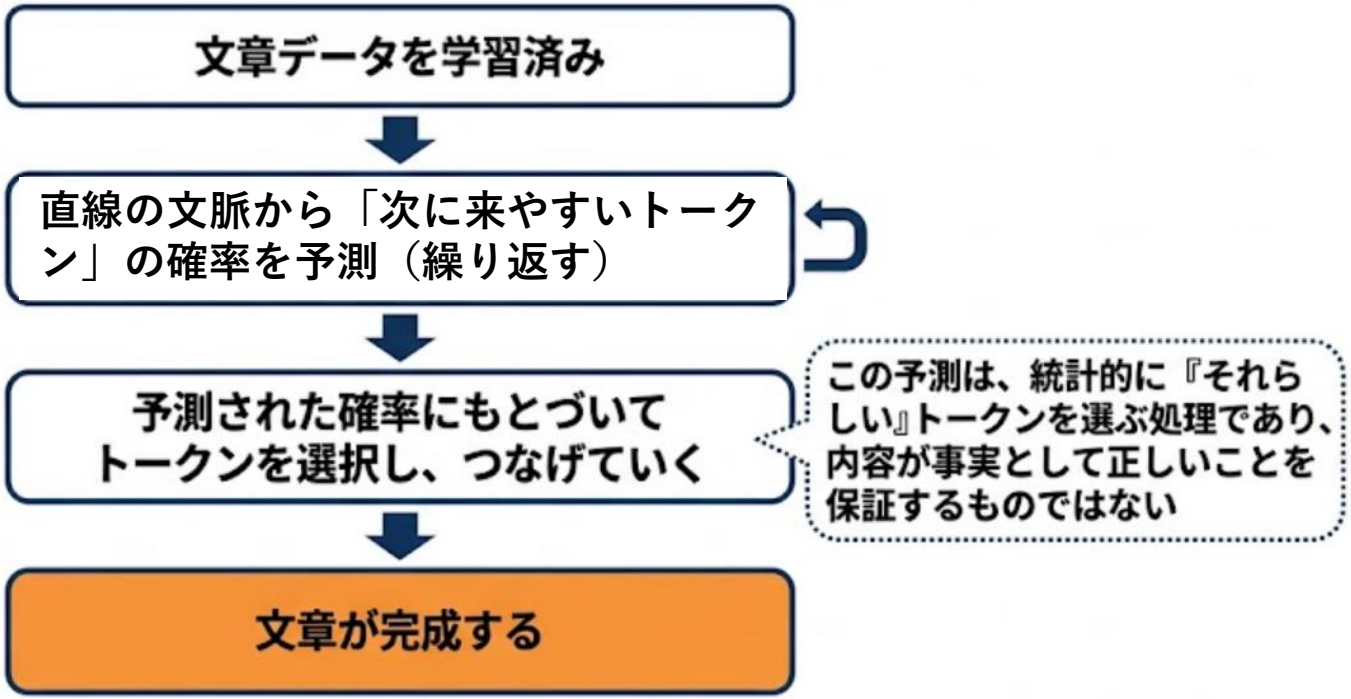
↓

④ 質問には、その傾向をもとに"次に来やすいトークン"を一つずつ選んで文章を作る

注意：AIは人間のように"意味"を理解しているわけではなく、
言葉のつながりの確率をもとに文章を作っています

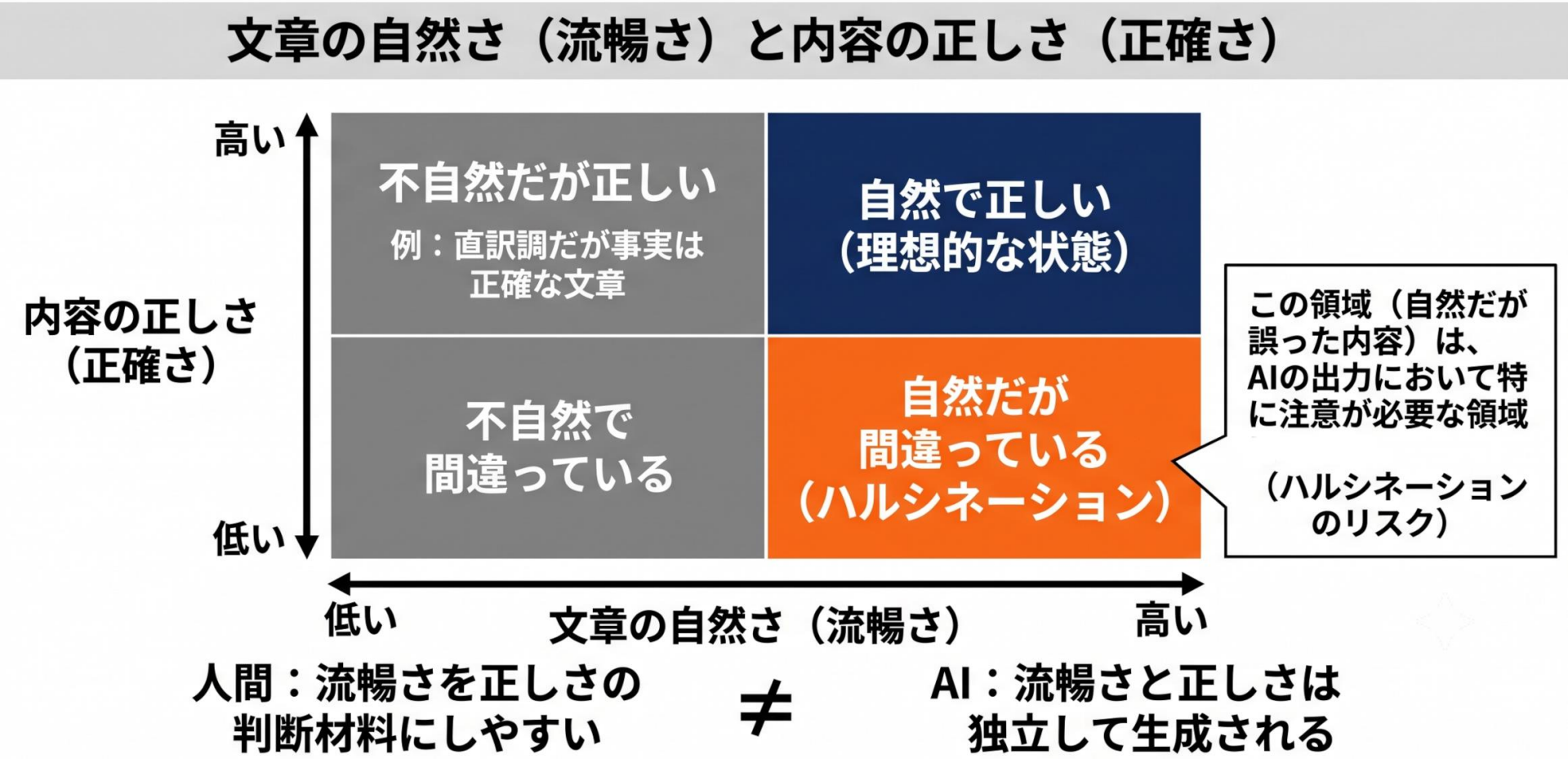
対話型AIの仕組みと限界

AIはどうやって文章を作るのか ― 『次に来るトークン』を予測し続ける仕組み



結論：AIの文章生成は文章の自然さ（流暢さ）を実現する仕組みであり、内容の正しさを保証する仕組みではない

対話型AIの特質：「流暢さ」と「正しさ」は別の軸で動く



ハルシネーションとは、AIが誤った内容を、自信を持って正しいかのように述べてしまう現象を指す

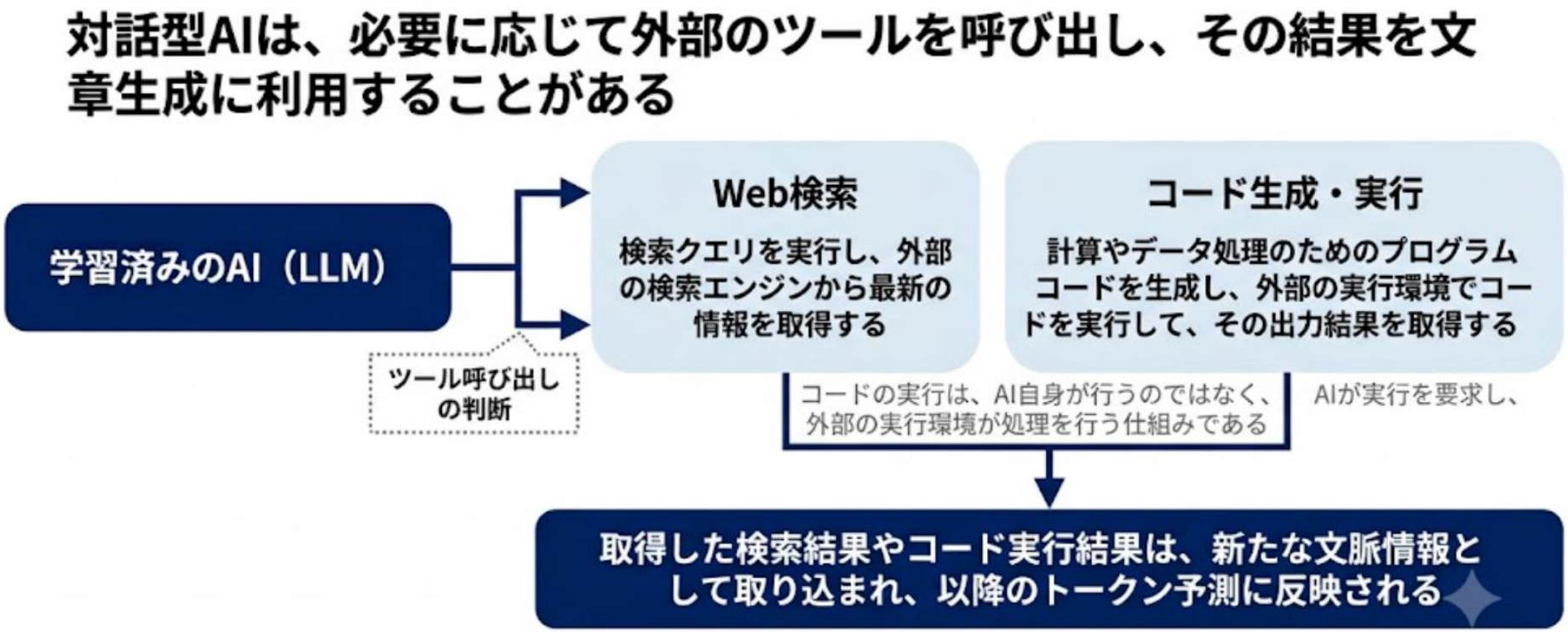
イメージ：AIによるWeb検索

AIは自身が学習した情報だけでなく、必要に応じてWeb検索を行い、
その結果を回答に反映することがある

イメージ：AIによるプログラム生成と実行

AIはプログラムコードを生成し、外部の実行環境でそのコードを動かして結果を得ることがある。コードの実行はAI自身の内部ではなく、外部の環境で行われる

AI による外部ツール（Web検索、プログラム生成と実行）の活用



注意：ツールを利用しても、最終的な文章はトークンの確率的予測によって生成されるため、内容の正しさが保証されるわけではない

14-2. 対話型AIの活用


ここ行ってみよう！
AIのおすすめだって
JR 大学前駅
お食事処 きらり
お食事処
テナント募集
おきえ
AI
え…
そこもう潰れて、
今は別のお店に
なってるよ？
駅前でおすすめの
『トラットリア○○』
です！

AI の回答は次の 3 点で注意が必要

Database Lab.

指示（プロンプト）の出し方で答えは変わり、時には自信満々だが事実や実際の内容と異なる回答が起こることもある。

プロンプトの工夫で結果が変わる

『AIについて教えて』
（ざっくりとした指示）

↓

曖昧で漠然とした回答

『生成AIのしくみを
3つのポイントで説明して』
（具体的な指示）

↓

整理された明確な回答

**指示の出し方で
答えの質が大きく変わる**

AIは自信満々に間違うこともある（ハルシネーション）

※以下はハルシネーションを理解するための、わかりやすくした簡略化例です

富士山の高さは？

5000メートルです

実際の正しい高さ：3776メートル

↓

**もっともらしく自信満々に答えるが、
事実とは異なることがある
＝ハルシネーション**

※AIは事実を検索しているのではなく、学習データのパターンから次に来る言葉（トークン）を統計的に予測して文章を作っている。そのため誤りにも気づかず自信満々に答えてしまう

実際の内容とズレた提案をしてしまうこともある

『Python入門の授業
レポートを考えて』

↓

『tryは信頼性向上に役立つ』
など、授業でまだ習っていない
内容を含めてしまう

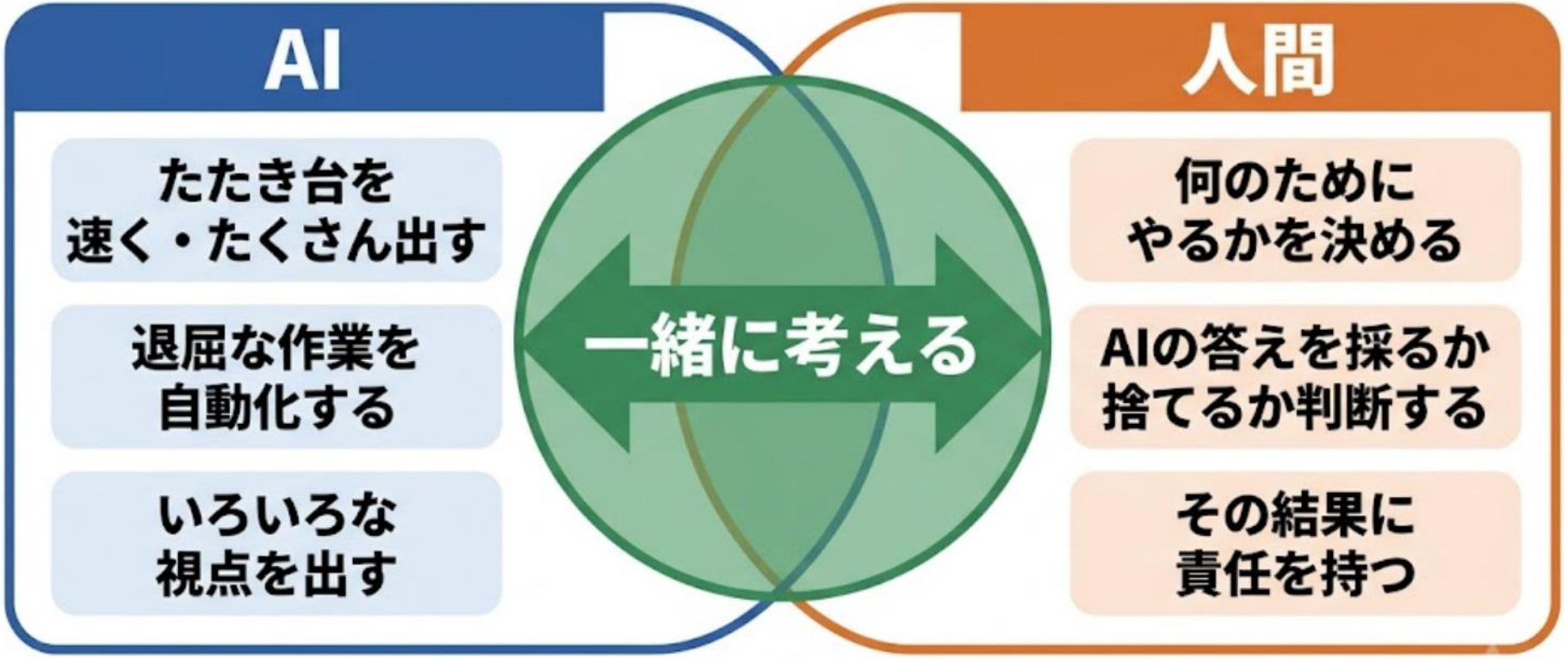
実際の授業で習った内容について、
自分で考えて作成することで
実力が伸びるもの

↓

**AIはあくまで相談相手であり、
正解をそのまま出す存在ではない。
内容は自分で確認・調整する**

Database Lab.

AIの得意と人間の役割を分けて、一緒に考える



AIに考えさせるのではなく、AIと一緒に考える

AIを使うときは、情報の『入れ方』と『答えの使い方』に注意が必要

AIに入れてはいけない情報

- 自分や他人の個人情報
- 他人の秘密
- 公開されていない機密情報

AIの答えを使うときに守ること

- AIの回答をそのまま信じない
- 最終的判断は人間の役割

AIの結論を『たたき台』から『検証済みの結論』にする

AIの結論（たたき台）

例：富士山の高さは5000メートルです

検証の3つの視点

1 根拠は一次資料までたどれるか

2 自分が知っている事実や他の資料と一致するか
（例：富士山の正しい高さは3776　メートル。食い違うためハルシネーションの可能性がある）

3 その情報はいつの時点のものか

検証済みの結論

根拠を確認し、採用すると人間が判断した結論

根拠を確認するまで、AIの結論は『たたき台』のまま

14-3. データサイエンスのための AI


そうか、
雨の日の
せいか！
9:41
ラップタイム(秒)
(秒)
14
12
10
8
6
4
2
0
外れ値
(雨の日)
1日目
2日目
3日目
4日目
6日目
7日目
8日目
10日目
AIで返信
平均：8.9秒/本
遅い
中央値：いつもと近い
メッセージ

データの基礎知識があるほど、AIとより的確に対話できる

データの基礎知識

平均／中央値／外れ値／欠損値

相関／因果／標本

AIへの的確な指示ができる

例1：外れ値を除いた場合の傾向も教えて

例2：相関だけでなく因果関係の可能性も検討して

AIの結果を鵜呑みにせず見極める力になる

例1	相関関係を因果関係と混同する
例2	少数の外れ値に平均値が引きずられる

AIは自信を持って誤った結論を提示することがある

平均値と中央値

65　70　72　75　80

平均値：すべての値を足して個数で割った値

(65+70+72+75+80)÷5＝72.4

データ全体の合計をならした値。すべての数字が計算に使われる

中央値：小さい順に並べたとき、ちょうど真ん中に来る値

65　70　【72】　75　80

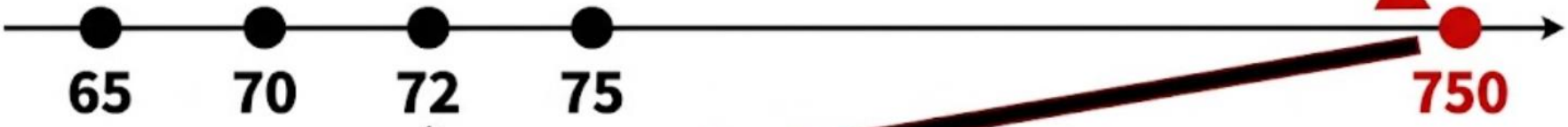
大きさの順番だけに注目した値。真ん中の位置にある数字がそのまま答えになる

平均値と中央値は、必ずしも同じ値になるわけではない

平均値と中央値：外れ値による影響の違い

外れ値が1つ混ざるだけで、平均値は大きく引きずられるが、中央値はほとんど変わらない

外れ値：他の値から極端に離れた値
入力ミスや特殊事情で生じる (75の入力を750と誤入力)



平均値：すべての値を足して個数で割った値

(65+70+72+75+750)÷5＝206.4

外れ値に大きく引きずられて、実際の点数からかけ離れた値になる

中央値：小さい順に並べたとき、ちょうど真ん中に来る値

65　70　[72]　75　750

外れ値（750）があっても、真ん中の位置は変わらないためほとんど影響を受けない

欠損値

Database Lab.

表の中の空欄（欠損値）は、対処が必要

空値（欠損値）とは？

氏名	身長	睡眠時間	好きな食べ物	学食の満足度
リク	175	（空欄）	ハンバーガー	3
ナナ	152	6.5	（空欄）	4
モモ	（空欄）	（空欄）	パスタ	2
トモヤ	176	7.0	（空欄）	4

空値（欠損値）＝回答がなく、データが記録されていない箇所

対処方法を結果が変わる

睡眠時間の平均を計算する

- **方法A：空欄の行を無視する（除外する）**
- **方法B：空欄を平均値で埋める**

欠損値をどう扱うかで分析結果は変わる。無視するか、値で埋めるかを目的に応じて正しく選ぶ必要がある。


相関・因果…
似てるようで
全部違うんだな。
ちゃんと見分けよう
?

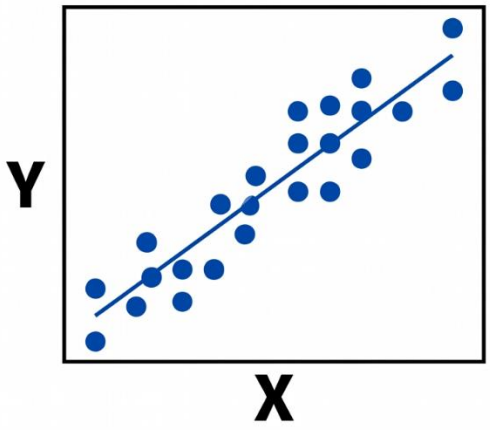
相関の定義と種類（正・負・なし）

2つの変数の関係を分析（数値データ用）

相関がある場合，一方が変化するともう一方も変化する傾向にある

種類

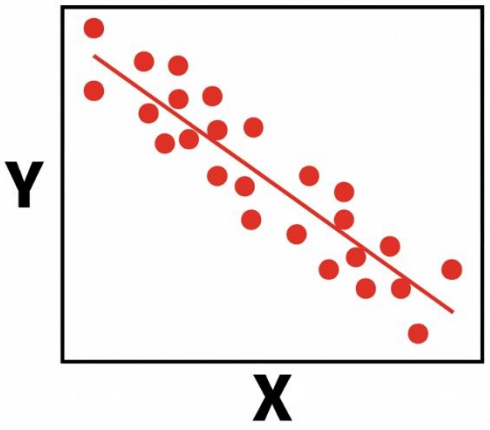
正の相関



Xが増えるとYが増える傾向

例：勉強時間が増えると得点が上がる

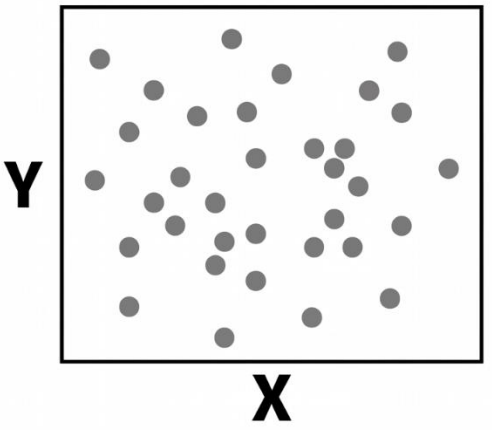
負の相関



Xが増えるとYが減る傾向

例：ガソリン代が上がると車の利用が減る

相関なし



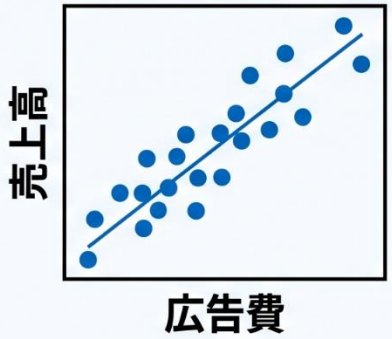
XとYに関係がない

例：足のサイズと勉強時間に関係がない

相関の活用例

2つの量の間の関係性の分析に活用される

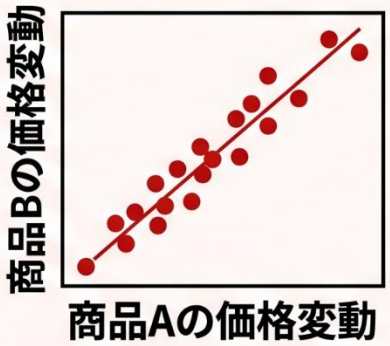
①マーケティング



相関係数 ≒ 0.8

広告を増やすと売上高が増えそうか

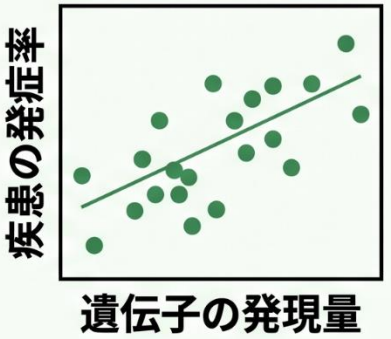
②金融



相関係数 ≒ 0.9

相関が高い複数の金融商品を扱うとリスクが高いか

③医療・生命科学

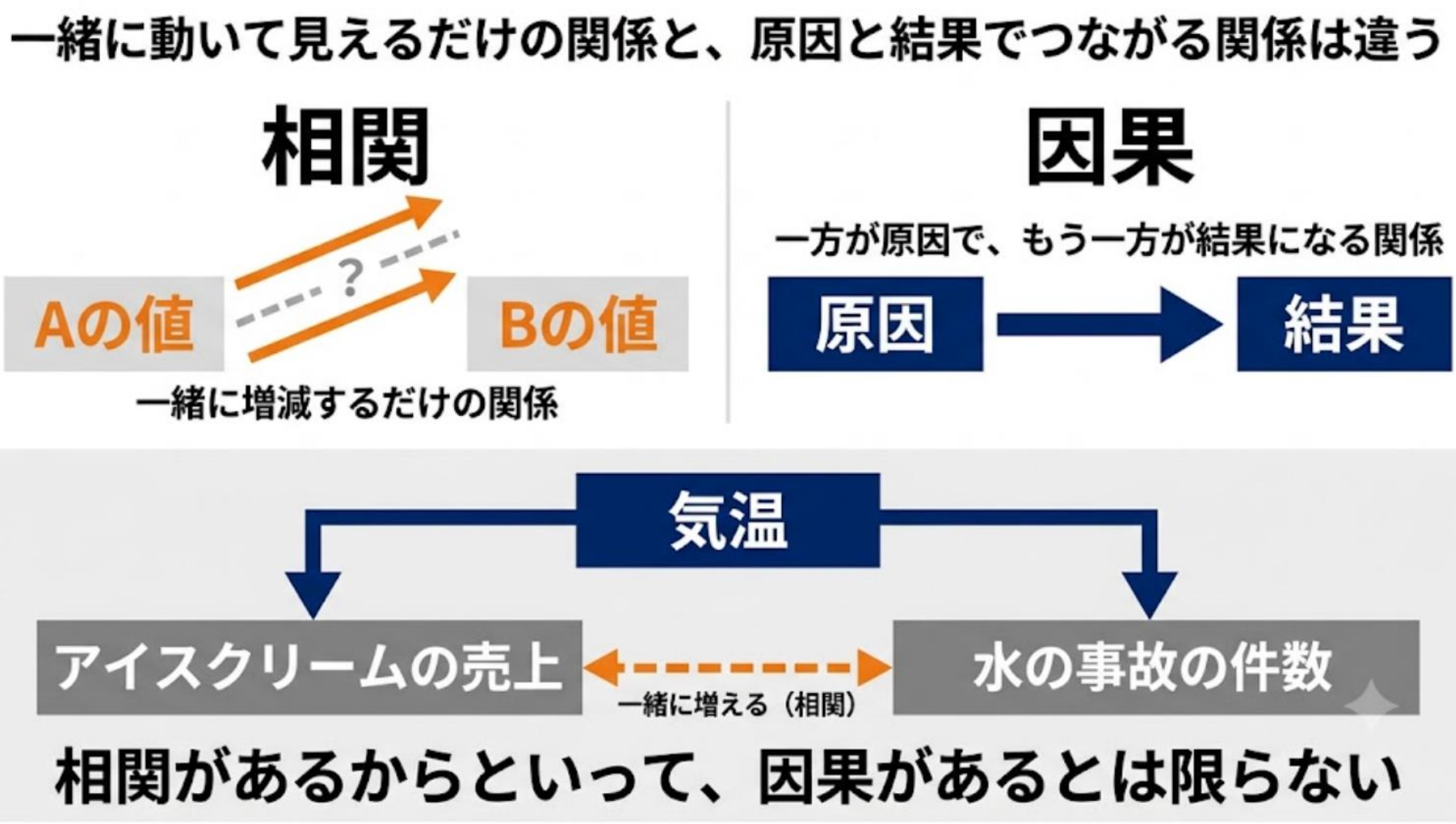


相関係数 ≒ 0.6

遺伝子と疾患に関係がありそうか

いずれも2つの量の間の関係性の分析に活用される

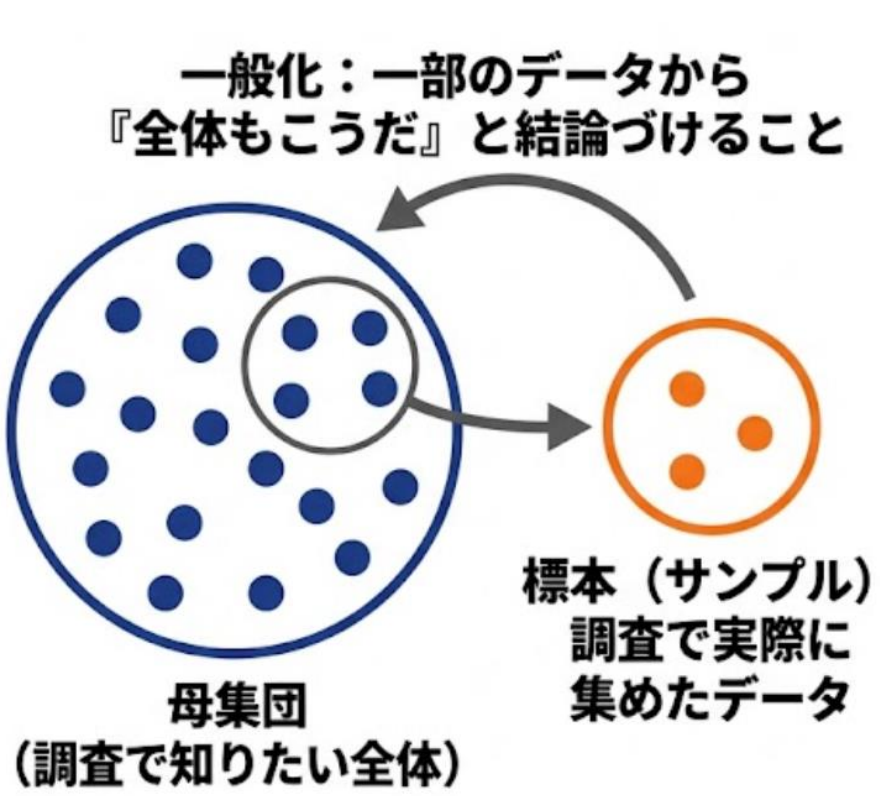
相関（２つのデータの関連）と因果（原因と結果の関係）は違う

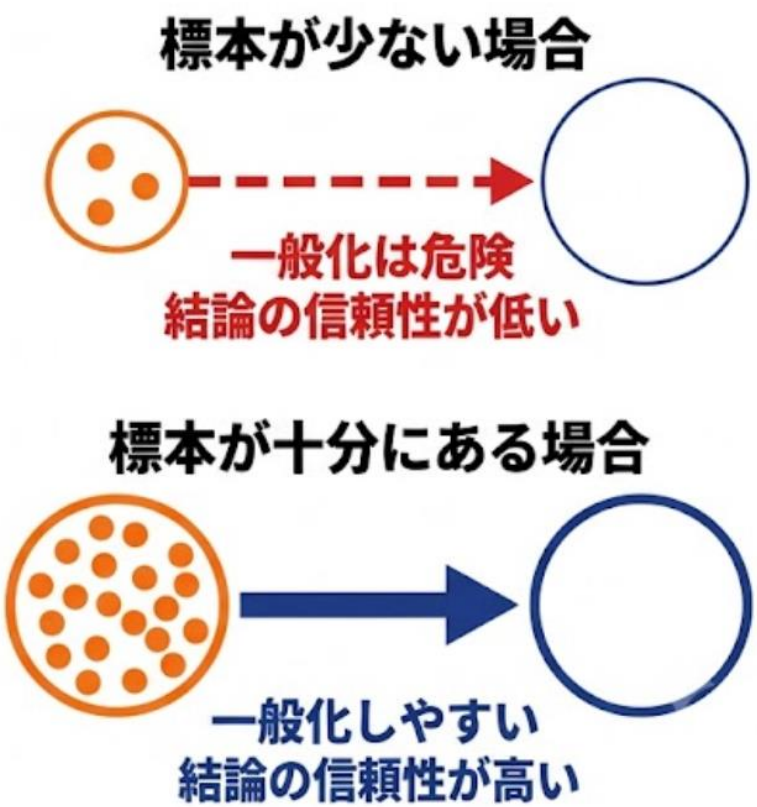


一般化における注意点

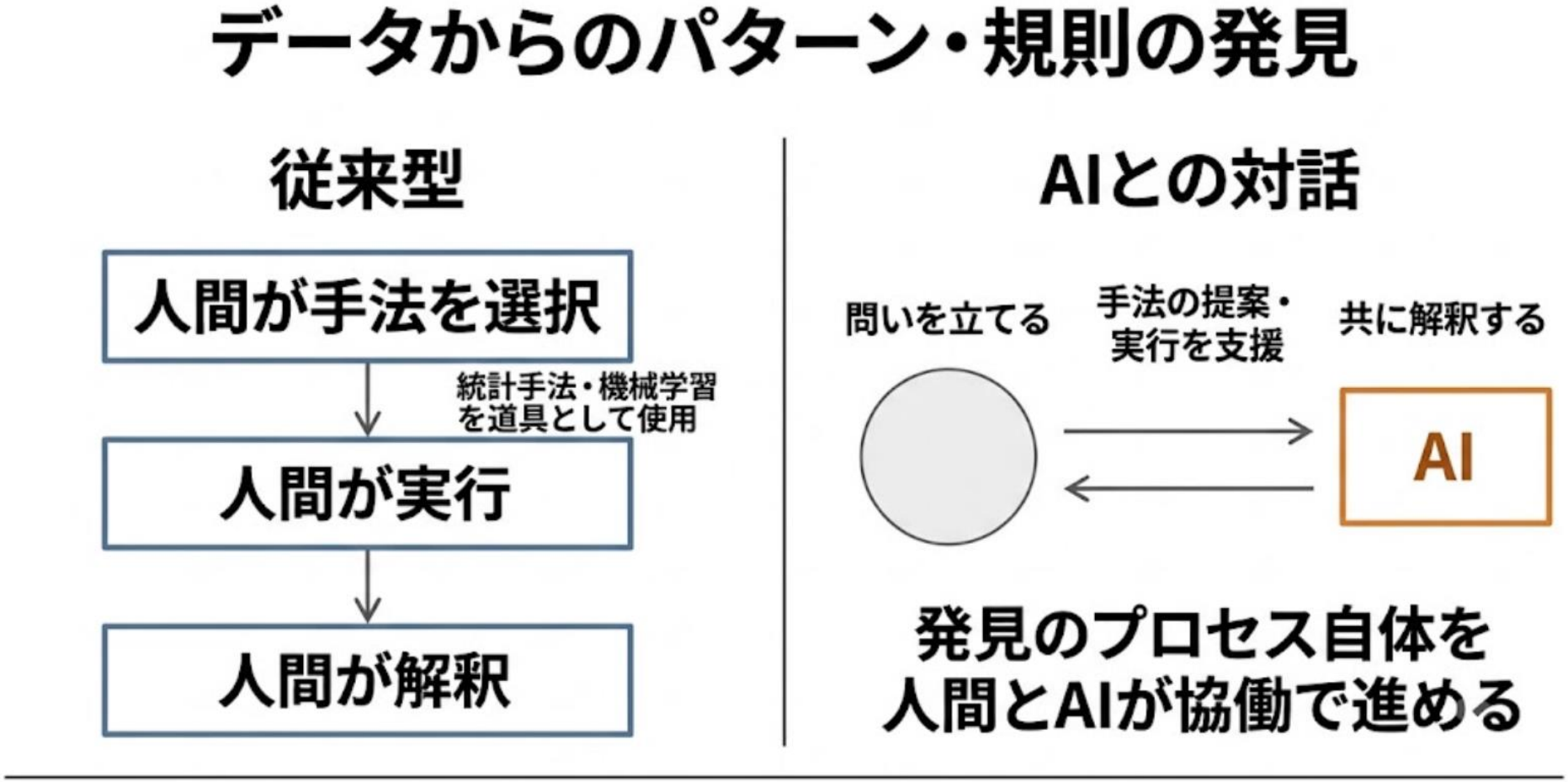
相関や因果を正しく判断するには、十分な量のデータが必要である。標本が少ない場合の注意点を確認する

標本が小さいと、そこから『全体もそうだ』と結論づける一般化は危険になる





対話型AIの登場により、手法選択から解釈までの一連の作業を、人間とAIが対話しながら進められるようになった

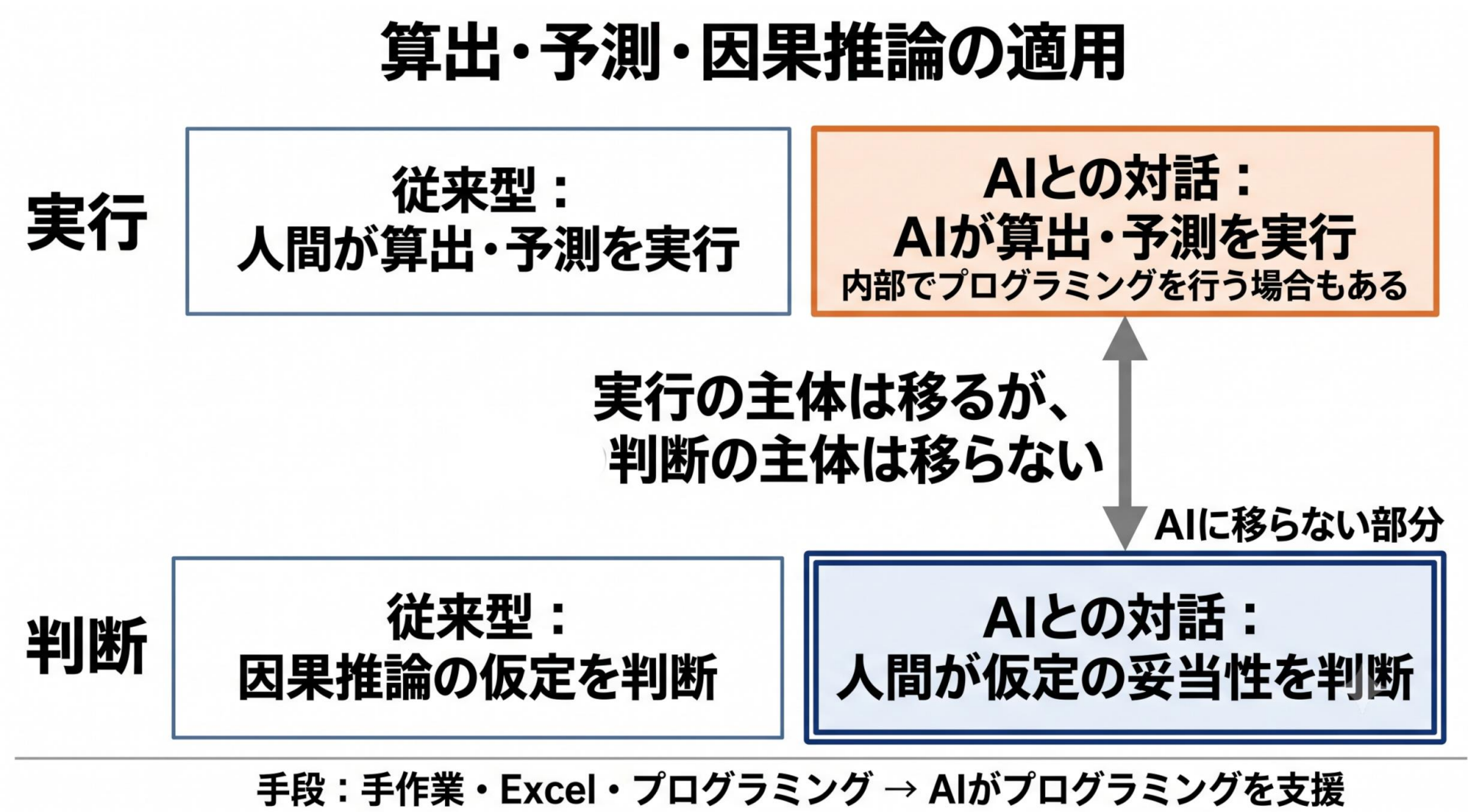


AIの回答をそのまま信じないために、確認が必要。

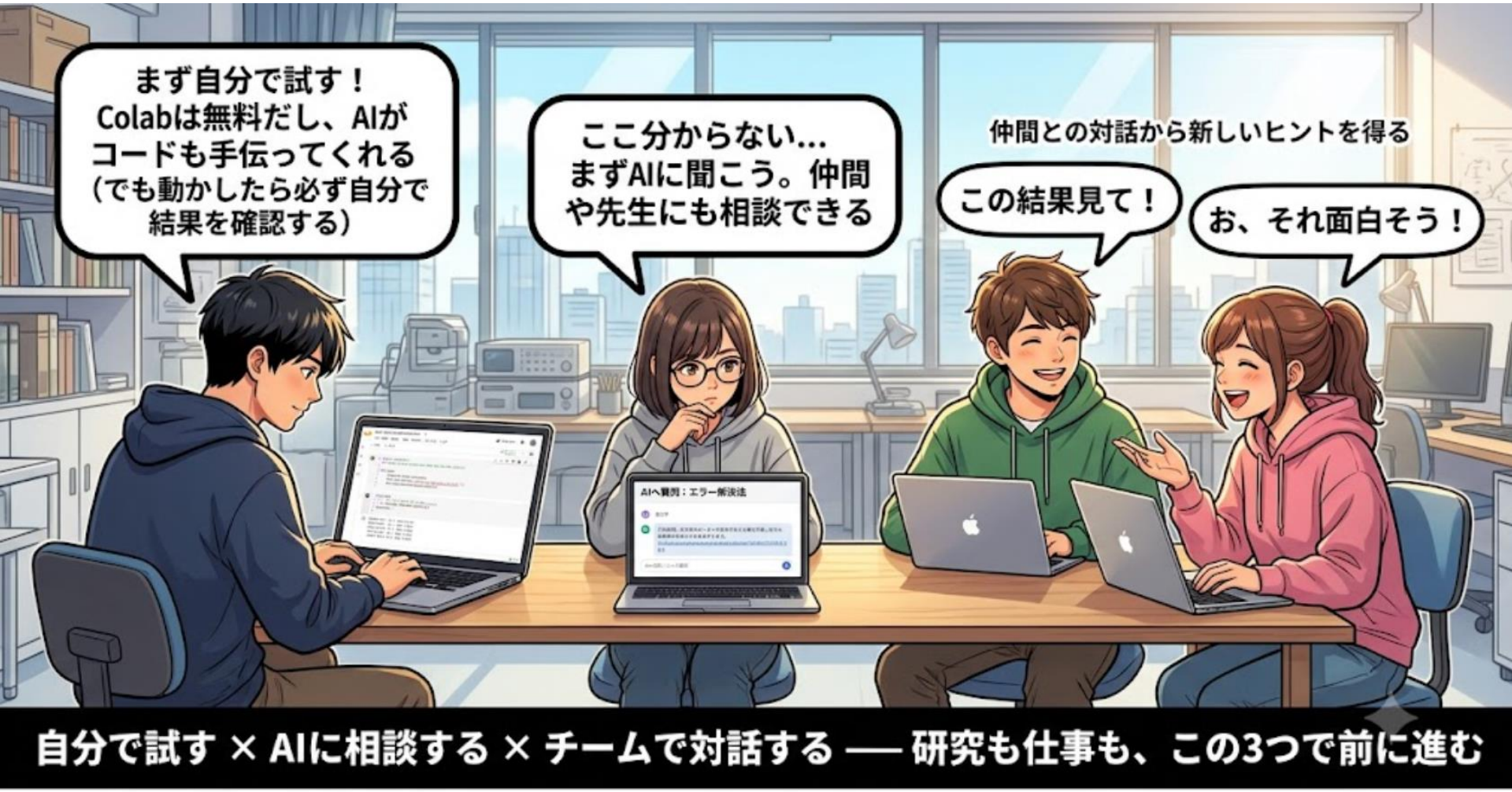
結論の根拠を確認

「AIがそう言ったから」は、理由になりません

AIが実行を担う場面が増えても、目的設定や仮定の妥当性判断など、最終的な判断は引き続き人間が担う



14-4. プログラミングのための AI



自分で試す × AIに相談する × チームで対話する ── 研究も仕事も、この3つで前に進む

この進め方は、将来の仕事でこそ役立つ

AIアシスタントとプログラミング

AIアシスタントの例 — Colabは登録無料

やりたいことを言葉で伝えれば、AIがコードを書くのを手伝ってくれる

コードを書く場所

```
import plotly.graph_objects as go
import numpy as np

np.random.seed(42)
n = 100
x = np.random.randn(n)
y = np.random.randn(n)
z = np.random.randn(n)

fig = go.Figure(data=[go.Scatter3d(x=x,
          y=y, z=z, mode='markers')])
fig.show()
```

相談しながら
コードを完成

AIアシスタント（Gemini）

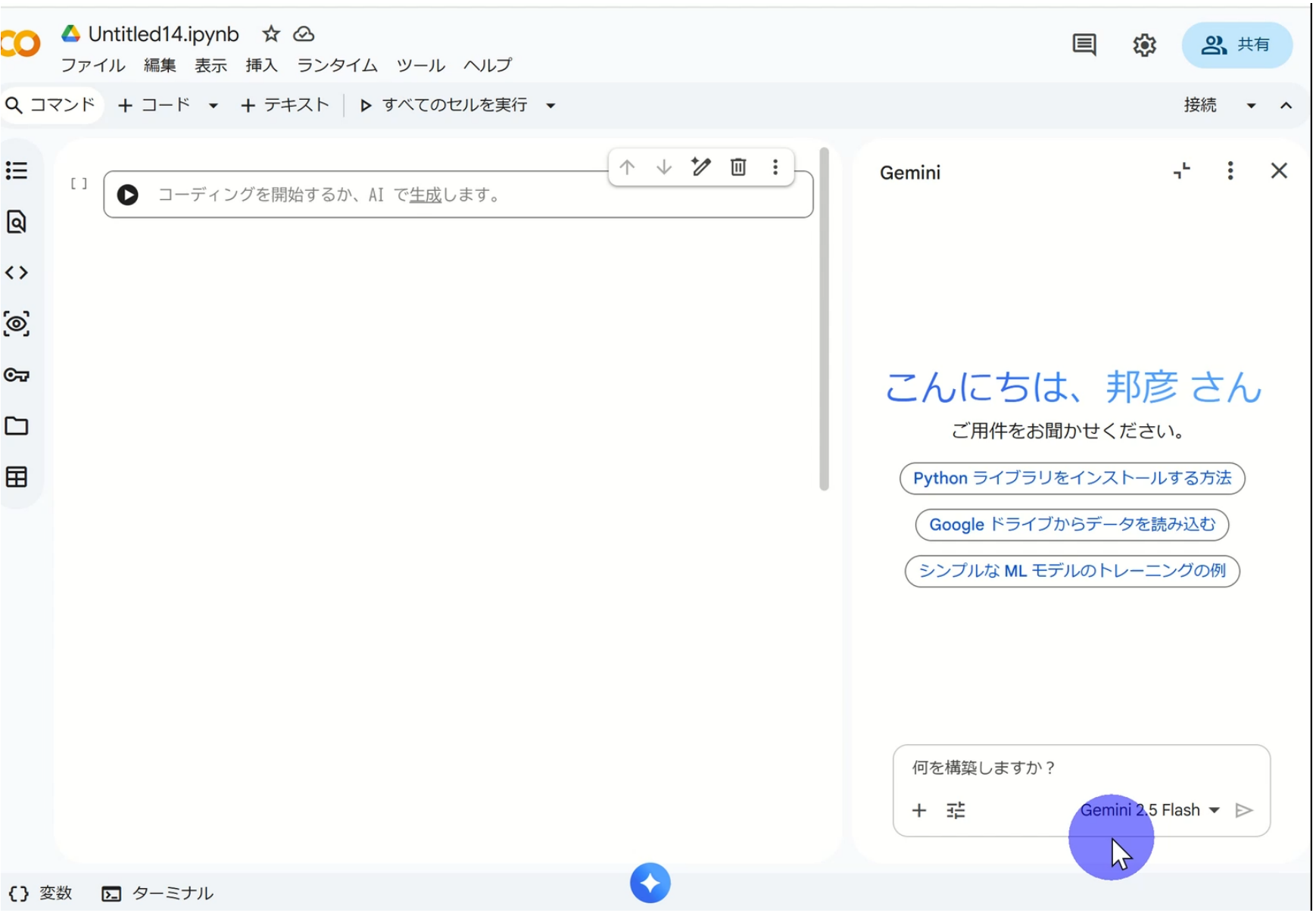
こんにちは。ご用件をお聞かせください

Pythonライブラリのインストール方法

Googleドライブからデータを読み込む

シンプルなMLモデルの例

プログラミングが苦手でも安心。コードは自力で解くものではなく、自分の道具。
『動いた』を一つずつ積み重ねよう


Untitled14.ipynb
ファイル 編集 表示 挿入 ランタイム ツール ヘルプ
共有
コマンド
コード
テキスト
すべてのセルを実行
接続
コーディングを開始するか、AI で生成します。
Gemini
こんにちは、邦彦 さん
ご用件をお聞かせください。
Python ライブラリをインストールする方法
Google ドライブからデータを読み込む
シンプルな ML モデルのトレーニングの例
何を構築しますか？
Gemini 2.5 Flash
変数
ターミナル

コードは動くが、「6〜12歳の子供料金」の区分が抜けている（10歳が通常料金になる）。確認・修正が必要

コンピュータとプログラム

プログラミングは、自分のアイデアをコンピュータ上で形にする創造的な手段

コンピュータの5つの基本動作

入力
（情報をもらう）

出力
（結果を見せる）

順次実行
（書いた順に実行）

条件分岐
（条件で処理を変える）

繰り返し
（決めた回数くり返す）

プログラムの動き

ソースコード → コンピュータが処理 → 結果

人間が読める言葉で書いた命令の集まり
＝ 読める・書ける・直せる

Python の3つの利点

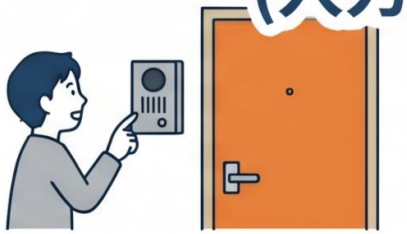
①文法がシンプル
＝ print / if・else / for・while
字下げ（インデント）でブロックを示す

②豊富なライブラリ
＝ データ分析・AI・Web・自動化

③柔軟性 ＝ 小規模～大規模まで同じ言語で

コンピュータはプログラムの指示通りに動く

キーボードなどから
情報をもらう (入力)



順次実行

結果を相手に見せる
(出力)

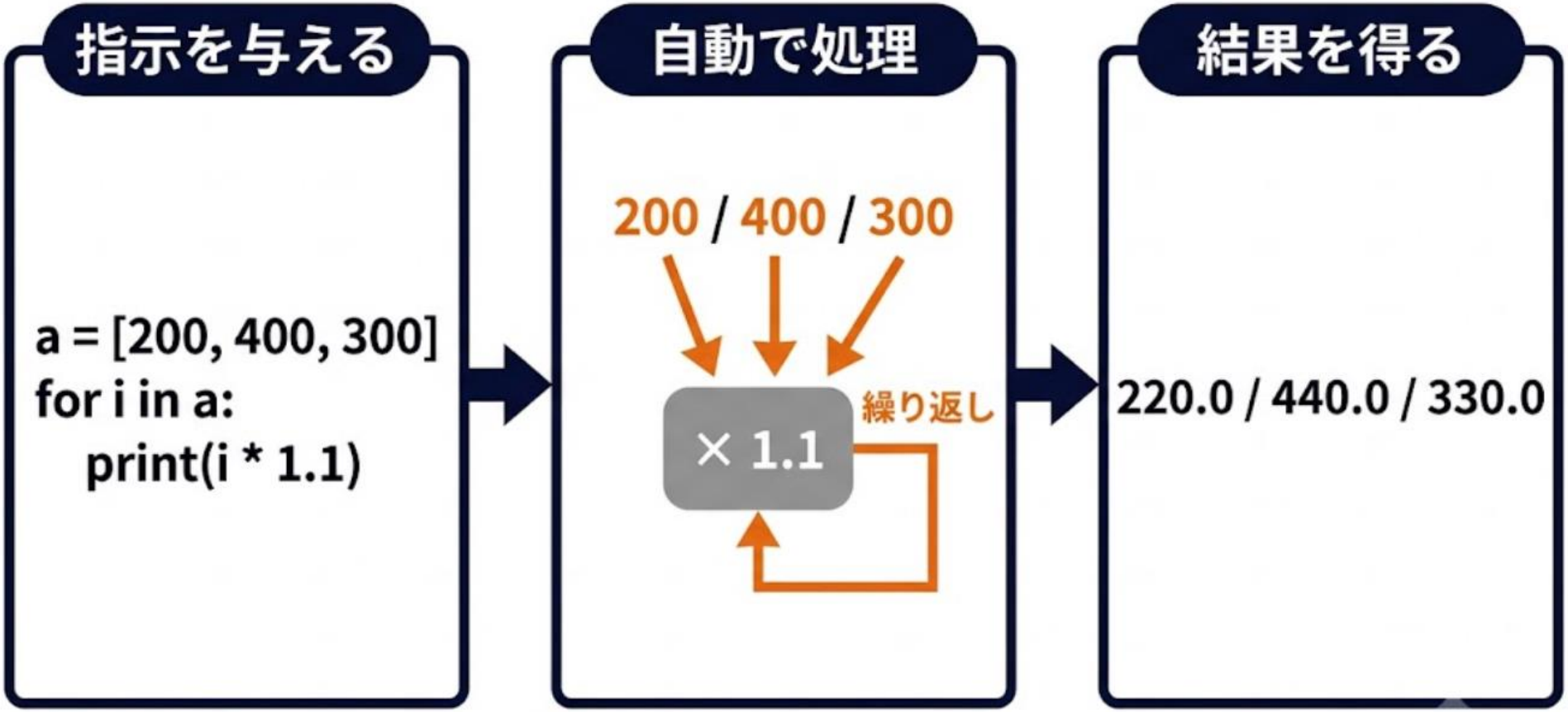
もし雨なら傘を持つ、
晴れなら持たない (条件分岐)

決めた回数だけ
同じ動作をくり返す
(繰り返し)

どの言語も
働きは同じ、
書き方が違う

Pythonは
書き方の見た目
整理されていて
読みやすい

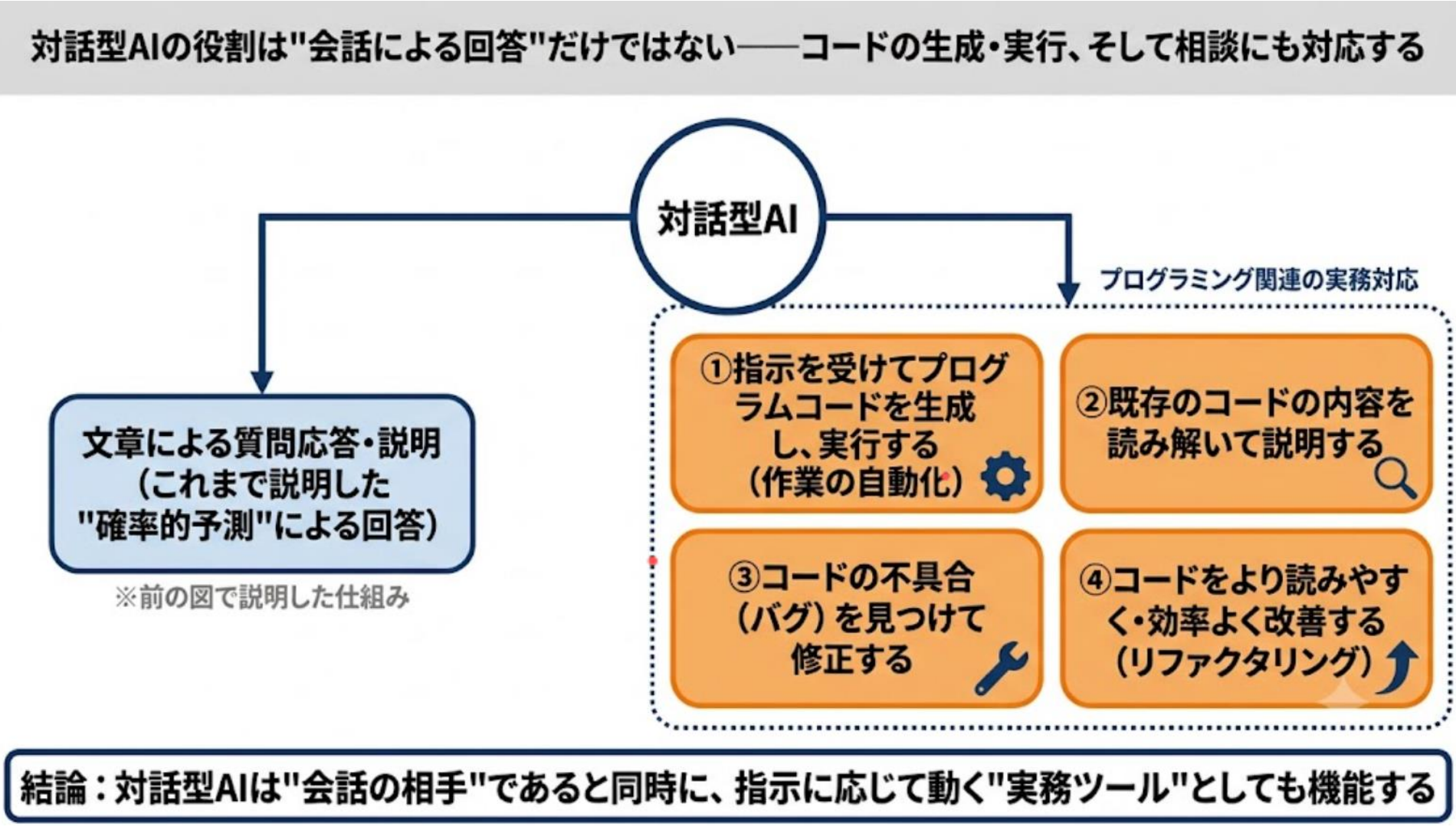
プログラムとは — 命令を書いた手順の集まり



だから複雑な作業も【自動化・効率化】できる

オブジェクト・メソッド・引数

プログラミングでの対話型AIの活用



演習

演習1. Copilotでアンケートを検証する

手順の概要

- CSVデータをCopilotに貼り付け
- 中央値・外れ値・欠損値を質問
- 結論を聞く

得られる結果

- 身長の中央値・外れ値の数値
- 欠損値の扱いに関する回答
- アンケートからの結論（文章）

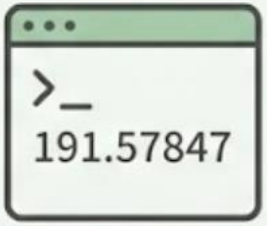
注意点・ヒント・考察ポイント

- 欠損値を0扱いしていないか確認
- 計算した人数が30人か確認
- 断定的な結論は根拠を確認

演習2. コードを実行して数値を照合する

手順の概要

- OneCompilerのAIにコード生成を依頼
- 実行ボタンで動かす
- Copilotが書いたコードとも比較



得られる結果

- 実行された中央値の数値
- 演習1の数値との一致／不一致

注意点・ヒント・考察ポイント

- 欠損値がコード内でどう処理されているか確認
- print文で件数を確認

演習3. AIにゲームを作らせて遊ぶ

手順の概要

- おみくじ等をAIに作成依頼
- 実行して動作確認
- 曖昧な指示を出して反応を見る
- 1か所だけ自分で書き換え

得られる結果

- 動くミニゲーム
- 曖昧な指示に対するAIの解釈の違い

注意点・ヒント

注意点・ヒント・考察ポイント

- OneCompilerとCopilotの作り方の違いを比較
- 書けることと理解することの違いを意識