AIとはじめるプログラミング
― 言葉がコードに変わる

金子邦彦

AIが拓くプログラミングの新時代

対話型AIの登場により、

プログラミングは

「構文を暗記して書く作業」から

「**実現したいことを言葉で伝え、生成されたコードを評価・改善する創造的な活動**」へと

変化している

本授業で学ぶ内容

- **対話型AIに日本語で指示を出してプログラムを作成**する方法
- 生成されたコードを理解し、評価し、修正する技術

この授業の意義

- 「**何を作りたいか**」を考え「**正しく動くか**」を**確認**する能力を習得
- **数分でプロトタイプを作成**し、**対話で改善**する開発スタイルを身につける

デモンストレーション→演習→メリットと注意点→基礎知識→自己学習方法

本授業の実行環境

- Copilot, Trinket の Python 2 を使用

1. 対話型AIでプログラムを作る（デモンストレーション）

人間の言葉からコードを生成

対話型AI（人間の言葉を理解し対話するAI）が、人間の言葉（指示）をプログラムコードに変換

利点

- 開発を容易に開始
- 「**実現したいこと**」を日本語で指示すればコードが生成される

例：「**2つの数を足し算するプログラム**」という指示から生成されるコード

```
a = 3
b = 5
print(a + b)
```

実行すると8と表示される

プログラムの基本構造

- プログラムは入力、処理、出力の3要素で構成される

例：

```
data = [1, 2, 3, 4, 5]     # 入力
total = sum(data)          # 処理
print(total)               # 出力
```

dataが入力、sumによる計算が処理、printが出力。
実行すると15と表示される

対話型AIと会話形式でプログラムを作成・修正

エラーが発生しても**対話を通じて修正できる**

例：構文エラーがある場合

```
total = 0
for i in range(1, 10)
    total += i
print(total)
```

```
SyntaxError: bad input on line 2 in main.py
```

実行結果

- 対話型AIに「**SyntaxErrorが出ました**」と**伝える**と、**行末にコロンを追加する修正案**が得られる

この手法の利点

- 一度で完璧なコードを作成する必要が**ない**
- **段階的に改善**しながら完成

Trinket の Python 2 の使い方

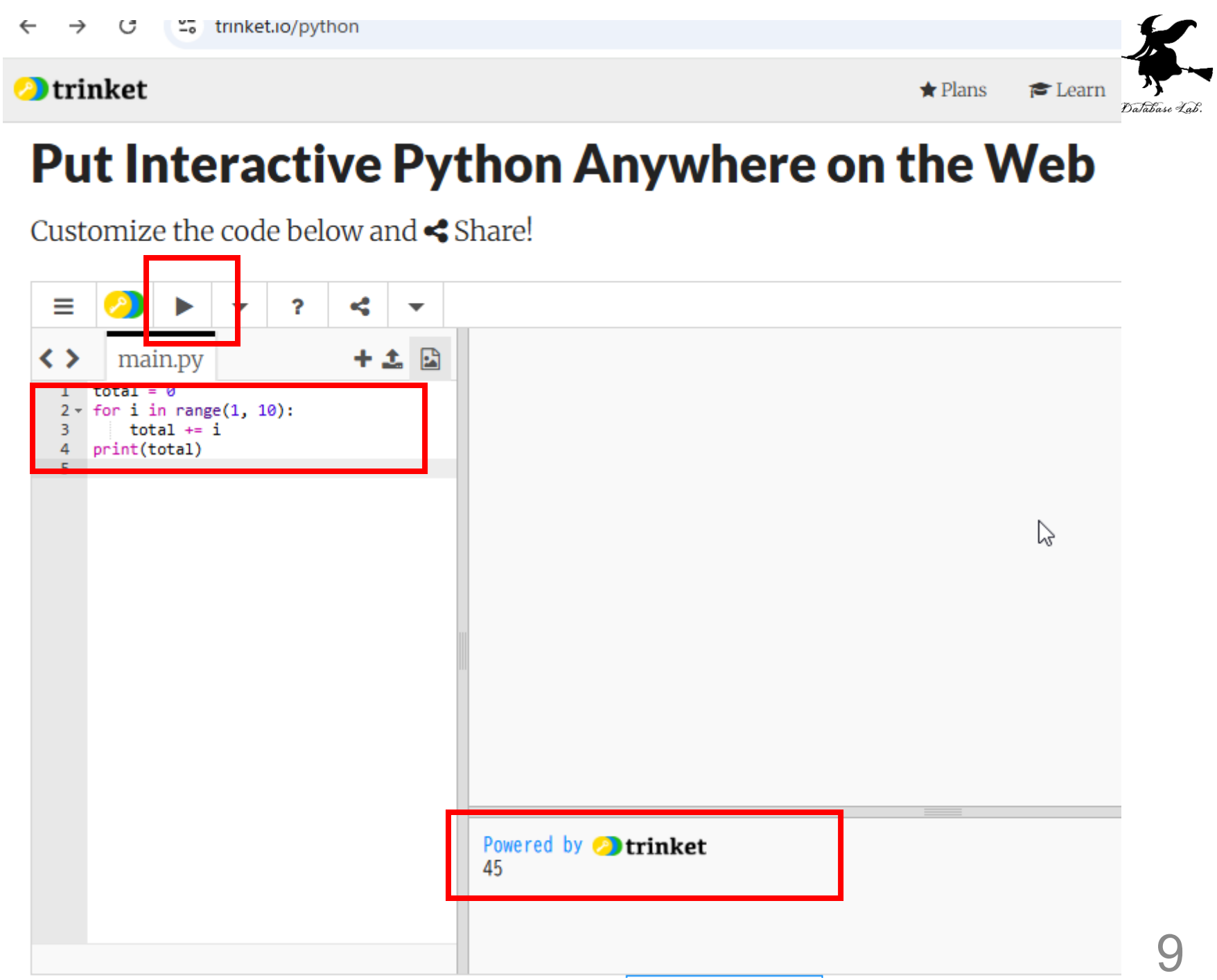
Trinketとは、ブラウザ上でプログラムを作成・実行できる無料のWebサービスである

利用手順

- ブラウザで Trinketにアクセス
  https://trinket.io/python
- 画面**左側のエディタ**にコードを入力
- 画面上部の実行ボタン（**三角形**）をクリック
- 画面右側に実行結果が表示される

注意事項

- アカウント作成は不要である
- 本授業ではPython 2を使用する


trinket.io/python
trinket
Plans
Learn
Put Interactive Python Anywhere on the Web
Customize the code below and Share!
main.py
total = 0
for i in range(1, 10):
total += i
print(total)
Powered by trinket
45

デモンストレーション

例題：1から100までの合計を計算するプログラム

手順1：対話型AIに日本語で指示を入力する

Copilot を使用

「1から100までの整数の合計を計算するPythonプログラムを作成してください。Trinket の Python 2 で動作するコードでお願いします」

手順2：生成されたコードをTrinket の Python 2 で実行し、動作を確認する　**https://trinket.io/python**

```
total = 0
for i in range(1, 101):
    total += i
print(total)
```

結果：5050

2. 対話型AIへの指示を自分で書いて実行する

対話型AIへの指示を自分で書いて実行する

要件定義は、「**何をしたいか**」を**明確に言語化する作業**である。対話型AI活用において特に重要。

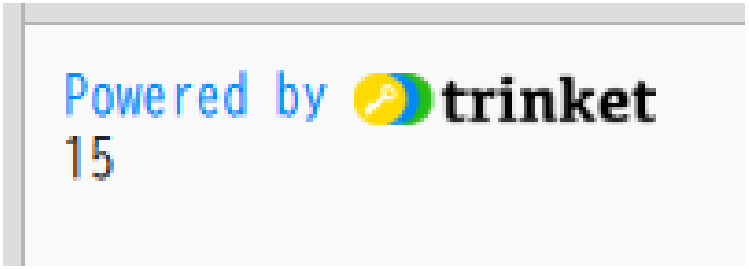
具体例

曖昧な指示：「数字を足すプログラムを作って」 NG

明確な指示：「リスト[1, 2, 3, 4, 5]の合計を計算して表示するプログラムを作って」

生成されるコード例

```
numbers = [1, 2, 3, 4, 5]
total = sum(numbers)
print(total)
```



生成されたコードが要求を満たしているか確認する（検証）

検証の具体例：リストの合計を求めるコード

```
numbers = [1, 2, 3, 4, 5]
total = sum(numbers)
print(total)
```

・期待する結果を準備しておく

1+2+3+4+5=15。実行結果が15であれば正しい

検証を行う意義

- バグの少ないプログラムを作成できる
- 対話型AIの誤りに気づくことができる

演習1. リストの最大値を見つける

課題：リスト[10, 25, 8, 42, 17]の中から最大値を見つけて表示するプログラムを作成する

入力は何か → 数値のリスト[10, 25, 8, 42, 17]

出力は何か → 最大値（42）

・手順

1. ブラウザで Microsoft Copilotにアクセス
2. **以下のプロンプトを入力欄にコピー＆ペーストし、送信する**
   リスト[10, 25, 8, 42, 17]の中から最大値を見つけて表示するPythonプログラムを作成してください。Trinket の Python 2 で動作するコードでお願いします。
3. Microsoft Copilotが生成したコードをコピー
4. 別のタブでTrinket（https://trinket.io/python）にアクセス
5. 画面左側のエディタに、コピーしたコードを貼り付ける
6. 画面上部の実行ボタンをクリック
7. 画面右側に実行結果が表示される

・生成されるコードの例

```
numbers = [10, 25, 8, 42, 17]
print(max(numbers))
```

・**実行すると42と表示される**

3. 対話型AI活用のメリットと従来のプログラミングとの違い

役割の変化

人間は「コードを書く実装者」から「**問題を定義し評価する設計者**」へ

設計者の役割

- **何を作るか**を決める
- 生成されたコードが**正しいか確認**する
- **改善点を指示**する

設計者の視点で確認

```
scores = [80, 90, 75, 85, 70]
average = sum(scores) / len(scores)
print(average)
```

・**実行すると80と表示される。**
(80+90+75+85+70)/5=80なので**正しい**

本質的な変化

従来のプログラミング

- 構文規則を学習し、一から記述する
- エラー解決に時間がかかる

対話型AI活用

- **自然言語で指示する**
- **数分でプロトタイプが完成する**
- **対話で段階的に改善する**

求められる能力の変化

- 従来：文法や関数名を暗記し正確に書く能力
- 現在：「**何を作りたいか**」を明確にし「**正しいか**」を判断する能力

4. 対話型AI利用時の注意点と失敗を避ける方法

AIの不確実性

対話型AIは誤りや幻覚（hallucination：**存在しない情報**を**事実のように生成する現象**）が発生することがある

誤りの例：存在しないメソッドを使用したコード

```
numbers = [1, 2, 3, 4, 5]
result = numbers.average()  # このメソッドは存在しない
```

正しいコード

```
numbers = [1, 2, 3, 4, 5]
result = sum(numbers) / len(numbers)
print(result)
```

⇒　実行すると3と表示される

対話型AIは100%正確ではない。エラーが出たら**対話型AIに質問**すれば解決のヒントが得られることが多い

検証責任と批判的思考

生成された**コードの正当性を保証**するのは**人間**である

批判的思考の具体的な方法

- コードを一行ずつ読み、何をしているか確認する
- 異なる入力値で実行し、結果が正しいか確認する
- 「なぜこの書き方なのか」を対話型AIに質問する

検証の例

```
scores = [80, 90, 70]
average = sum(scores) / len(scores)
print(average)
```

・実行すると80と表示される。**(80+90+70)/3=80なので正しい**

生成コードの限界

- 正確性の問題：構文エラーや論理エラーを含む可能性がある
- 品質の問題：非効率なコードを含む可能性がある

```
# 冗長な書き方
numbers = [1, 2, 3, 4, 5] total = 0
for i in range(len(numbers)):
    total = total + numbers[i]
print(total)
```

非効率なコード

```
# 簡潔な書き方
numbers = [1, 2, 3, 4, 5]
print(sum(numbers))
```

効率の良いコード

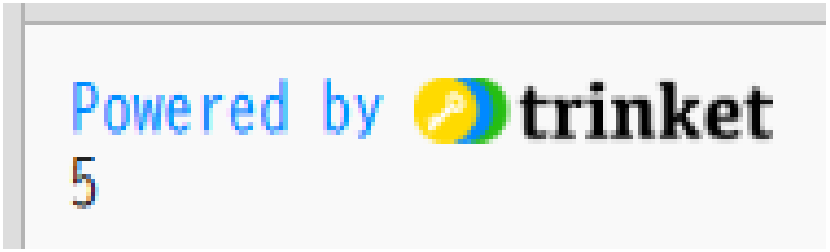
どちらも15と表示されるが、簡潔な書き方は読みやすい。より良い書き方があるか対話型AIに**相談**できる

学習上のポイント

- コピー＆ペーストだけでなく、コードを読む習慣も重要
- **基礎概念に慣れているとスムーズになる**

<u>コードの例</u>

```
count = 0
for i in range(1, 11):
    if i % 2 == 0:
        count += 1
print(count)
```

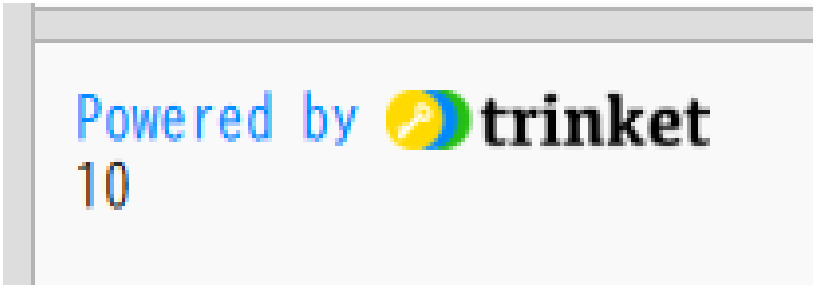


- このコードは1から10までの偶数の個数を数えている。
- countは変数、forは繰り返し、ifは条件分岐、%は余りを求める演算子。
- 実行すると5と表示される。わからない点は対話型AIに質問できる

効果的な学習方法

- 生成されたコードを一行ずつ読む
- なぜそのアルゴリズム（問題を解決するための手順）が選ばれたかを確認

```
total = 0
for i in range(5):
    total += i
print(total)
```



- 1行目：totalを0で初期化。
- 2行目：iを0から4まで変化させながら繰り返す。
- 3行目：totalにiを加える。
- 4行目：合計を表示。計算過程は0+1+2+3+4=10で、実行すると10と表示される

AIに説明を求めることも可能

5. Pythonプログラミングの必須基礎知識

変数（状態）

変数：値を保持する仕組み

例（Trinket の Python 2 で実行可能）

```
total = 0
name = "Python"
is_valid = True
print(total, name, is_valid)
```

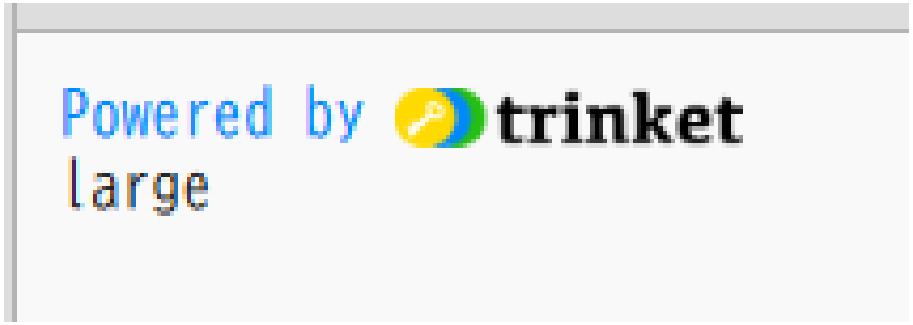
totalには数値0、nameには文字列"Python"、is_validには真偽値Trueが保持されている。

実行すると(0, 'Python', True)と表示される

制御構造（条件分岐・繰り返し）

if文：条件分岐

```
x = 15
if x > 10:
    print("large")
else:
    print("small")
```



xが10より大きければ"large"、そうでなければ"small"を表示する。実行するとlargeと表示される

for文：繰り返し

```
for i in range(5):
    print(i)
```

range(5)は0から4までの数を生成。実行すると0, 1, 2, 3, 4が各行に表示される

関数

関数：**処理をまとめる仕組み**

例（Trinket の Python 2 で実行可能）

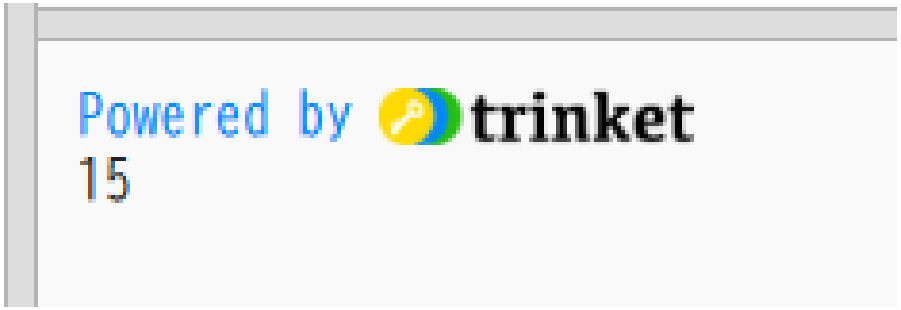
```
def add(a, b):
    result = a + b
    return result


print(add(3, 5))
print(add(10, 20))
```

addは2つの数を受け取り合計を返す関数である。add(3, 5)は8を、add(10, 20)は30を返す

プログラムの意味を理解する

各行について「**この行は何をしているか**」を**自分で説明**してみる

```
total = 0
for i in range(1, 6):
    total += i
print(total)
```



1行目：totalを0で初期化。2行目：iを1から5まで変化させる。3行目：totalにiを加える。4行目：totalを表示。

計算過程は1+2+3+4+5=15で、実行すると15と表示される。AIに説明を求めることも可能

演習2. 条件分岐を使ったプログラム

課題：**点数に応じて「合格」または「不合格」を表示するプログラムを作成する（60点以上で合格）**

・手順

1. ブラウザで Microsoft Copilotにアクセス
2. 以下のプロンプトを入力欄にコピー＆ペーストし、送信する
   点数を変数scoreに代入し、60点以上なら「pass」、60点未満なら「fail」と表示するPythonプログラムを作成してください。scoreには75を代入してください。Trinket の Python 2 で動作するコードでお願いします。
3. Microsoft Copilotが生成したコードをコピーする
4. 別のタブでTrinket（https://trinket.io/python）にアクセス
5. 画面左側のエディタに、コピーしたコードを貼り付ける
6. 画面上部の「Run」ボタンをクリックする
7. 画面右側に実行結果が表示される

・生成されるコードの例

```
score = 75
if score >= 60:
    print("pass")
else:
    print("fail")
```

・実行するとpassと表示される。scoreを55に変更するとfailと表示される

6.発展的演習による自己学習

メタ認知と問題分解

- メタ認知とは、**自分の理解度**を**客観的に評価する能力**である
  - 「**このコードを他の人に説明できるか**」と自問する
  - **説明できない部分**は**対話型AIに質問できる**
- **問題分解**とは、**大きな問題**を**小さな問題に分ける**手法である

  例：「**じゃんけんゲームを作る**」→**①手を選ぶ②勝敗を判定③結果を表示**

段階的学習

簡単なものから**徐々に難しく**する進め方

```
# レベル1：数値を表示する
print(42)
# レベル2：変数を使う
x = 42
print(x)
# レベル3：計算を行う
print(40 + 2)
# レベル4：繰り返しを使う
for i in range(1, 6):
    print(i)
```

レベル1から4まで順に42、42、42、1から5が表示される

学習の進め方

- 簡単な課題から始める
  - 自分のレベルに適した課題を選択
- 生成されたコードを理解してから次へ進む
  - 各行が何をしているか説明してみる
- コードを改造して挙動を観察する
  - range(5)をrange(10)に変更して結果の違いを確認する
- 時々は自分で書く練習をする
  - 対話型AIに頼りすぎず、自分でも挑戦してみる

演習3. リストの平均値を計算する

課題：リスト[10, 20, 30, 40, 50]の平均値を計算して表示するプログラムを作成する

手順

1. ブラウザで Microsoft Copilotにアクセス
2. 以下のプロンプトを入力欄にコピー＆ペーストし、送信する
   リスト[10, 20, 30, 40, 50]の平均値を計算して表示するPythonプログラムを作成してください。Trinket の Python 2 で動作するコードでお願いします。
3. Microsoft Copilotが生成したコードをコピーする
4. 別のタブでTrinket（https://trinket.io/python）にアクセス
5. 画面左側のエディタに、コピーしたコードを貼り付ける
6. 画面上部の「Run」ボタンをクリックする
7. 画面右側に実行結果が表示される

生成されるコードの例

```
data = [10, 20, 30, 40, 50]
average = sum(data) / len(data)
print(average)
```

実行すると30と表示される。(10+20+30+40+50)/5=30なので正しい

演習4. じゃんけんゲーム（コンピュータ同士の対戦）

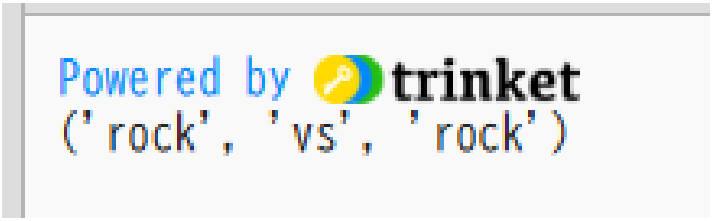
課題：2台のコンピュータがランダムに手を選び、対戦結果を表示するプログラムを作成する

手順

1. ブラウザで Microsoft Copilotにアクセス
2. 以下のプロンプトを入力欄にコピー＆ペーストし、送信する
   じゃんけんゲームを作成してください。2台のコンピュータがそれぞれランダムに手（rock, paper, scissors）を選び、両者の手を表示するPythonプログラムを作成してください。Trinket の Python 2 で動作するコードでお願いします。
3. Microsoft Copilotが生成したコードをコピーする
4. 別のタブでTrinket（https://trinket.io/python）にアクセス
5. 画面左側のエディタに、コピーしたコードを貼り付ける
6. 画面上部の「Run」ボタンをクリックする
7. 画面右側に実行結果が表示される

生成されるコードの例

```
import random
hands = ["rock", "paper", "scissors"]
p1 = random.choice(hands)
p2 = random.choice(hands)
print(p1, "vs", p2)
```



実行するとランダムな結果が表示される。実行するたびに結果が変わる

演習５. テキストベースの棒グラフ

課題：リスト[3, 5, 2, 7, 4]のデータをアスタリスク（*）で棒グラフ表示する

手順

1. ブラウザで Microsoft Copilotにアクセス
2. 以下のプロンプトを入力欄にコピー＆ペーストし、送信する
   リスト[3, 5, 2, 7, 4]のデータをテキストベースの棒グラフで表示するPythonプログラムを作成してください。各データの値の数だけアスタリスク（*）を横に並べて表示してください。Trinket の Python 2 で動作するコードでお願いします。
3. Microsoft Copilotが生成したコードをコピーする
4. 別のタブでTrinket（https://trinket.io/python）にアクセス
5. 画面左側のエディタに、コピーしたコードを貼り付ける
6. 画面上部の「Run」ボタンをクリックする
7. 画面右側に実行結果が表示される

生成されるコードの例

```
data = [3, 5, 2, 7, 4]
for value in data:
    print("*" * value)
```

実行すると***、***、、*******、****が各行に表示される

演習6. エラー修正の体験

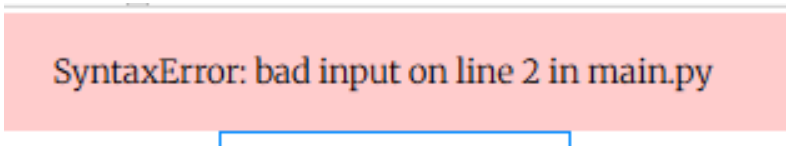
課題：エラーを含むコードを実行し、対話型AIに修正を依頼する

手順

１．Trinket（https://trinket.io/python）にアクセス

２．以下のエラーを含むコードを貼り付ける

```
total = 0
for i in range(1, 11)
total += i
    print(total)
```



３．「Run」をクリックし、SyntaxErrorが表示されることを確認

４．Microsoft Copilotにアクセス

５．以下のプロンプトを送信する

以下のPythonコードを実行したところ、SyntaxError: invalid syntaxというエラーが出ました。原因と修正方法を教えてください。

```
total = 0
for i in range(1, 11)
total += i
    print(total)
```

６．回答を読み、for文の行末にコロンがないことを理解する

７．修正後のコードを実行し、55と表示されることを確認

全体まとめ

- **対話型AIで人間の言葉からプログラム**を生成できる
- **生成されたコード**を**理解・評価・修正する能力**が重要である
- 基礎知識に慣れていると**対話型AIをより効果的に活用**できる
- わからないことがあれば**対話型AIに質問しながら学べる**

復習用コード

```
total = 0
for i in range(1, 6):
    total += i
print(total)
```

変数totalを0で初期化し、iを1から5まで加算し表示する。実行すると15と表示される

次のステップ：演習課題に取り組み、段階的に学習を進める