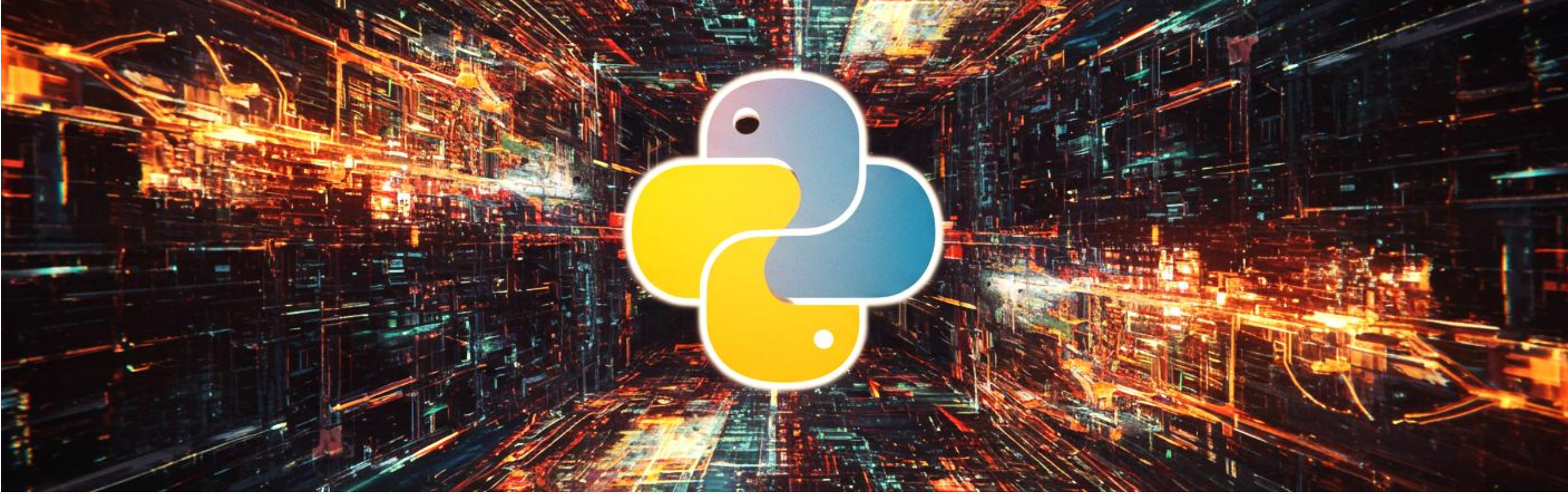


全３回で学ぶPython入門講座
1回目

2025/07/22

Kazuma Sekiguchi

自己紹介

関口和真

株式会社コムセントCTO
Webシステム開発、スマートフォンアプリ制作、
サーバー構築、運用など

スマートフォンを使った動画配信アプリの制作
サーバーサイドシステムの作成
フロントエンド部分の作成
AI周りのいろいろ

目標

- Pythonの基礎が理解できる
- Pythonで外部ライブラリーの活用ができる

今回のAgenda

- Pythonとは何か
- Pythonの動作環境
- Pythonの記述方法
- 変数、文字列連結
- 条件分岐の書き方
- import文の使い方
- 関数の利用
- ユーザー定義関数の作成

Pythonとは

- 本来はプログラミング学習を目的として、作成されたスクリプト言語
  - Pythonはスクリプト言語でもあり、Pythonというスクリプト言語を実行するための実行環境の名前でもある
  - Pythonはスクリプト言語なので、実行にはPythonが必要

Python

- PythonはGoogleが社内の主たるプログラミング言語の1つとして選択したため人気が出た
  - 以前から、比較的簡単に用いることができる点が評価され、統計解析や数学的な分析に用いられてきた
  - 統計解析や数学的なライブラリーが豊富に存在するのも特徴
- 多数のライブラリーが公開されていること、開発事例が多いため、参考となる資料が豊富である
  - 最初に触るプログラミング言語としても適している

Pythonの実行環境

- Pythonの実行環境はいくつか実装系が存在する
  - 他の言語と組み合わせて動かすときに別の言語で実装されたPythonを使うことがある
- C言語で記述された実装
  - 事実上Pythonと単に言った場合はこれを指す
  - 今回利用するのもこれ
  - CPython
- Java言語で記述された実装
  - Jython
- C#言語で記述された実装
  - IronPython

Pythonのバージョン

- 現在主流なのは、Python3系
  - 旧来存在した、Python2系から多少破壊的変更が行なわれているため、旧来のコードは動作しない可能性がある
- マイナーバージョンで変更が行われることがあり、たまに依存するライブラリーが動作しなくなることがある

Pythonの利用場面

- 機械学習の作成（AI）
- ブラウザーの自動化
- 自動化処理
- Excelの自動化
- サーバーの管理ツール
- Webサービスの作成

Pythonの記述

- Pythonは普通のエディタなどで記述が可能
  - 開発環境と呼ばれるソフトウェアを用いた方が、効率的に開発できる
    - 近年では、Pythonの開発には、Visual Studio Codeが用いられるケースが増えてきている
      - Visual Studio Codeには、拡張機能を読み込ませて機能を拡張することが可能
    - Pythonに関係する拡張機能も多く存在するため、導入することで、効率的に間違えの少ないPythonコードを記述することが可能になっている
- Pythonで書かれたプログラムの動作は、基本的にはOSには影響を受けない
  - ファイルの読み込みなどをする場合は、考慮する必要はある
  - Pythonの実行環境自体はさまざまなOS向けに提供されているため、macOSでもWindowsでもLinuxでも記述、動作が可能

Pythonの特徴

- 動的型付け言語
  - 現在は一部型を指定することが可能
- 元々教育的目的のため、比較的容易に記述することが可能
- スクリプト言語なため、確認→修正が容易
- 多数のライブラリーが公開されている
- 開発事例が多いため、参考となる資料が豊富
  - 比較的日本語の資料は少ないとされているが、最近は豊富

Pythonのデメリット

- 型指定が無いため、集団開発時にバグが生じやすい
- スクリプト言語ということもあり、実効速度が少し遅い
  - 少しずつ高速化はしつつある
  - 他の言語で学習しているとインデントが必要な点などで混乱を招く
  - if文などでインデントが必須とされている
- 多くのライブラリが開発されているが、ライブラリの更新頻度が高いため、コードが古くなってしまう

Pythonではできないこと

- スマートフォンアプリの作成
  - Kotlin、Swift、Dartなど
- ブラウザーのフロントエンドの作成
  - JavaScript、TypeScriptなど

Pythonのかなり強力なライブラリ

- Pythonはかなり豊富なライブラリーが用意されている
  - ライブラリーを使うことで、プログラムの開発が容易になる
  - Pythonの標準機能だけでは実現できないものが実現できる
  - ライブラリーは他の言語で作成されているものも多い
- pipコマンドを利用することでライブラリーのインストール、管理が容易に可能
  - pipコマンドはPythonに付属してくる
- 別のファイルに必要なパッケージのリストとバージョンを書いておけば簡単にパッケージ管理を行える
  - pipコマンドで読み込ませれば、必要な記述されているライブラリーをインストールしてくれる

pipの大変な点

- 依存関係がある場合、自動的には解消してくれないので、そこは自分で解消する必要がある
- 一部のライブラリーは組み込めても、他のライブラリーが組み込めず（エラーとなる）使えないことがマレに発生する
  - 比較的解消が大変

Pythonを作成、動作させる

- Pythonを記述する
  - 普通のエディタなどで記述は可能
  - Visual Studio Codeなどでも記述可能
    - 機能拡張を入れると更に便利
  - PyCharmなどの有料IDEも存在

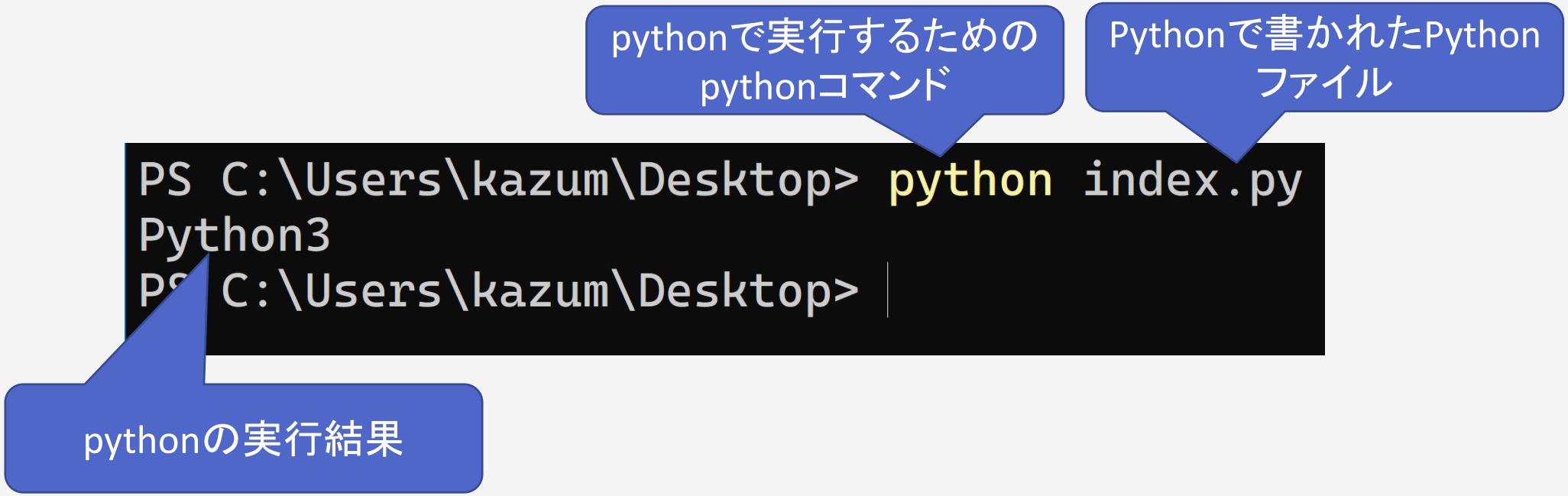
- Pythonの動作はOSには影響を受けない
  - デスクトップアプリケーションのみ影響を受ける
  - Python自体はさまざまなOS向けに提供されている
  - macOSでもWindowsでもLinuxでも記述、動作可能

Pythonを作成、動作させる

- Pythonを実行する環境は必要
  - 最低限Pythonをダウンロードすればok
    - 各OS用のPythonが用意されている
  - 他のツール群と一緒になっているタイプのものも存在
    - オリジナルのPythonと異なる点があるため、個人的にはあまりオススメしない
    - Anaconda
    - miniConda

Pythonを作成、動作させる

- Pythonを記述したらターミナルなどでpythonコマンドと実行したいPythonファイル名を指定すればOK



Pythonの記述

- 変数
- 関数
- 制御
- クラス

の4つから構成される

  - 更に細分化することも可能

書式

- 空の行などは一切意味がない
  - 見やすいように適宜改行を行って良い
- インデント（字下げ）はPythonでは意味を持つので、勝手に行わないようにする
- 1行で一つの動きを書く
  - セミコロンで区切って複数の動きを書くことも可能
- バックスラッシュを文末に付けると次の行と繋がる

```
print('Pythonで文字を表示します')
```

変数

- 変数

```
x = 5
```

  - この中のxが変数。xには５が入る
  - 変数は自由に値を格納できる箱みたいなもの

- 変数名は自由に設定できる
  - 但し、予め別の役割が決められているものは使用不可（予約語）
  - 変数名の最初は英字またはアンダーバーで始まること。
    数字は文頭には使えない

変数（2）

- 変数は宣言せずに使用可能
  - 初期化できる（現在は初期化をすることが推奨）

```
variables = ‘’
```

- 型は一切なし
  - 関数の引数やクラスなどでは指定することもできる
  - txt ＝ ‘’とすれば文字列型
  - txt ＝ 0とすれば数値（Long型）

変数（3）

- 文字はシングルクォーテーションまたはダブルクォーテーションで括る

```
a = ‘text’
```

- 数値はアンダーバーを使って区切りを付けることが可能

```
num = 123_456_789
num = 123456789
```

どちらも同じ意味になる
見やすい方を使うと良い

- 文字列で改行を表すときは¥n（バックスラッシュ＋n）を記述する

```
print(“this is a ¥n Python program”)
```

this is a
Python program
と改行されて表示される

変数（4）

- 特殊な値を格納可能
- 真か偽かを示す値（Bool値）
  - 先頭大文字のTrueまたはFalseを使う
  - フラグとして利用することが多い

```
variables = True
```

- 何も入っていないことを示す値（NULL値）
  - 先頭大文字のNoneを使う

```
variables = None
```

コメント

- スクリプトの実行時には無視される
- プログラムの動きなどを記述しておき、後から見直しやすいようにする
- 先頭に#を付けるとコメント（プログラムの実行には影響しないメモ書き）扱いになる

```
print('Python') #コメントなので影響与えない
```

文字列連結

- 変数同士をくっつける
- 「+」を使う

```
txt_first = “Python”
txt_second = “Program”
moji = txt_first + txt_second
```

mojiにはPythonProgramが入る

- +の記号は算術子（加算）としても使う
  - 変数の値によって役割が変わる

条件（条件分岐）

- 条件に応じて、処理を分けるケースは多い
  - クリックされてもこれ以上右に動かせない→左に動かす
  - いわゆる判断をさせる場合に利用
  - 条件式を記述して判断させる
  - ifのみ必須
    - 他は任意
  - elifは複数回利用可能
  - if文の中ではインデントをする（タブキーまたはスペース4つ）

インデントする

```
if 条件式1:
        条件式1に当てはまるならここが動作
elif 条件式2:
        条件式2に当てはまるならここが動作
else:
        条件式に当てはまらない場合に動作
```

条件式

- 0以外の数、True ，何らかの文字列は全て条件式で当てはまる
  - 0,Falseだと条件式に当てはまらない

- 比較して判断することも
  - if a > 3→aが3よりも大きければtrueになる
  - if a == 3→aが3のときだけtrueになる
  - if a >= 3 and a < 100→aが3以上で、100未満のときだけtrueになる
  - if a >= 100）or a < 10→aが100以上または10未満のときだけtrueになる

条件式の成立

- 成立する場合
  - if True
  - if 1以上の数値
  - if ‘何らかの文字’
  - 条件式が成立しない場合
- 不成立の場合
  - if False
  - if None
  - if 0
  - 条件式が成立しない場合

ループ（繰り返し処理）

- 同じ処理を繰り返す際に利用
  - いくつか方法がある
  - プログラムは楽するために利用する＝同じ処理はできるだけループを使って処理する
  - 条件が成立している限りループする
    - 初期値に0を入れて1回回るごとに1を足していくと10回回った段階で条件を満たさなくなる

```
while n < 10:
    print(n)
    n += 1
    //変数nに格納されている値を表示。0～9まで表示される
```

```
while 条件:
    繰り返す内容
```

インデントする

ループ

- 変数として与えたものを展開し続ける
  - リスト、タプル、辞書などを展開する
- for in文

```
for n in [1, 2, 3]:
    print(n)

for n in range(10):
    print(n)
```

インデントする

range()で()内の数字回数だけ繰り返す

for 変数 in リスト・タプル・辞書など

import文

- Pythonでは一部の機能がライブラリとして提供されている
  - 標準ライブラリ：Pythonで最初から導入されているライブラリ
  - 外部ライブラリ：Pythonには組み込まれていないため、インストールしてから読み込んで利用する
- ライブラリの機能を使うときは、import文を利用する

```
import datetime
dt_now = datetime.datetime.now()
print(dt_now)
```

datetimeライブラリを読み込む

datetimeライブラリの機能を利用する

現在時刻を表示する
例：2023-06-06 20:06:19.156218

関数

- Pythonのデフォルトで利用可能な関数が存在
- 引数を与えることで任意の動作が可能

入力を受付ける関数

```
inputstring = input(“文字を入力してください”)
print(inputstring)
```

型変換

- Pythonは型が明示的には存在しない
  - 暗黙的に存在する
  - 型変換をしないと文字列として処理されてしまうことがある
- 型
  - 文字と数字を区別するために使用される
  - 文字列型、整数型、小数点型などが存在する
  - 文字と数字を区別して利用するために暗黙的に存在する
- input()で受領したデータは文字列型になる
  - 数式などで計算をさせるためには、数字に変換する必要がある

型変換

- int()：文字列や浮動小数点数を整数型（int）に変換する
  - 小数の場合は、小数を切り捨てる
- float()：文字列や整数を浮動小数点数（float）に変換する
- str()：ほぼ任意のオブジェクトを文字列（str）に変換する
  - 整数型や小数点などを文字列に変換する
- bool()：数字などをBoolに変換する
  - 0、0.0、空文字、NoneなどはFalseになる
  - ほかはTrueになる

ユーザ定義関数

- 何らかの処理をまとめたもの
- 同じような処理を繰り返す際に利用する
  - イチイチ同じ処理を何度も書くのは面倒。間違えも増える
- ()内に引数を指定可能
  - 引数はカンマで区切ることで複数指定可能
  - 引数に値を与えて関数を呼び出せば、その引数を利用して処理してくれる
- ユーザ定義関数から更にほかのユーザ定義関数を呼び出すことも可能
- 記述場所は自由
  - 通常まとめて書いておく

ユーザ定義関数

```
def 関数名(引数1,引数2・・・):
    処理の内容
    return 処理の結果
```

インデントする

基本的な記述方法

```
def Calc(a,b):
    c = a + b
    return c
```

作成（Calcという関数を作成）

```
g = Calc(4,6) #gには10が格納される
def Calc(a,b):
    c = a + b
    return c
```

Calcを利用

```
def caution():
    print('注意！！')
```

returnの無いものも可能

Pythonの記述と実行

- 単一のファイルに記述して保存
- 拡張子はpyとする
- ターミナルなどから実行する
  - macからはpython3として実行する

```
python test.py
```

Pythonで実行する
ファイル名

```
1回目> python a07.py
```

フォーマット済み文字リテラル

- 最近のPythonで利用可能な機能で、文字列内に直接変数の値を埋め込んだり、式を評価したりすることができる

```
name = “Python”
type = “Programing”
print(f“{name}は{type}です")
```

PythonはProgramingです
と表示される

- print()でfまたはFを記述し、{}を書くことで変数が展開される
  - 中に式を書くことも可能

```
name = “Python”
print(f“{name}は{len(name)}文字です")
```

len()は文字数を数える関数
Pythonは6文字です
と表示される

フォーマット済み文字リテラル

- 小数点を丸めて表示することも可能

```
pi = 3.141592
print(f“円周率は{pi:.2f}です")
```

円周率は3.14です
と表示される

- 小数第3位を四捨五入して、表示する
- {変数:.2f}などのように記述する

チャレンジ

- 以下のプログラムを作成してみてください
  - 答えの例は格納してあります

- 指定された円の半径から、円周の長さを計算するプログラムを作成してください
  - 円周率πはmathをimportし、math.piで取得できます

チャレンジ

- 以下のプログラムを作成してみてください
  - 答えの例は格納してあります

- 2つの値と演算子（+,-,*,/）を受け取って計算結果を表示するプログラムを作成してください。
  - 0で割り算はできないことに注意
  - inputは連続して使うことが可能です

チャレンジ

- 以下のプログラムを作成してみてください
  - 答えの例は格納してあります

- センチメートルの値を入力すると、インチに変換して結果を表示するプログラムを作成してください。
  - 1インチ＝2.54センチメートル

ありがとうございました。

また次回。