Python文法IV

松尾・岩澤研究室
Matsuo-Iwasawa Lab UTokyo

2024/03/19

目次

- 文法I
  - 演算
  - 変数
  - データ型
- 文法II
  - コレクション
  - List
  - Tuple
  - Dict
- 文法III
  - 条件分岐、繰り返し
  - 条件分岐(if文)
  - 繰り返し(for文)
- 文法IV
  - 関数
  - モジュール
  - クラス

関数とは

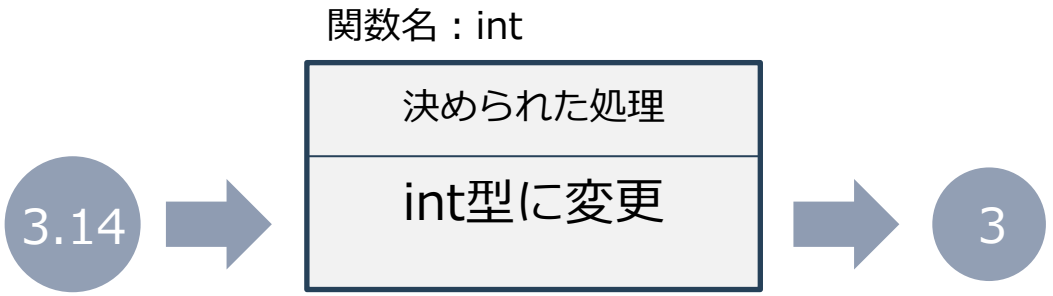
関数は決められた処理を実行するコードをひとくくりに管理するものです
Pythonには事前に定義されている関数が多くあります

これまでに使用したprint()やtype()も関数です

Pythonで事前に作られている関数の例

関数	処理内容
print()	値を表示
type()	データ型を出力
int()	整数のint型に変更
str()	文字列のstr型に変更
sum()	合計値を出力
len()	要素数を出力

（例）int関数のイメージ



関数の定義方法と使い方

開発者はオリジナルの関数を作成できます
何度も同じような処理を書かずに済み、メンテナンスも容易です

関数の定義と使い方

def 関数名():
インデント 決められた処理を行うコード

関数名()と書くと定義した処理が実行さます

（オリジナル関数例）

関数名：greet

決められた処理
“こんにちは”を出力

コード例

```
In  [1]: def greet():
             print(“こんにちは”)

         greet()
```

1回 ”こんにちは” を出力

```
Out [1]: こんにちは

In  [2]: greet()
         greet()

Out [2]: こんにちは
         こんにちは
```

2回 ”こんにちは” を出力

defは、define（定義する）の略称です。定義のかっこ、カンマ、インデント（スペース4つ）は省略できないので注意

関数の引数

関数に変数の値を引き渡したい場合に**引数**を定義します

引数のある関数の定義と使い方

def 関数名(引数):
インデント 決められた処理を行うコード

関数名(引数)と書くと定義した処理が実行さます

（オリジナル関数例）

コード例

```
In [1]: def greet(name):
            print(name+“さん、こんにちは”)

        greet(‘松尾’)
```

Out [1]： 松尾さん、こんにちは

```
In [2]: greet(‘岩澤’)
```

Out [2]： 岩澤さん、こんにちは

引数は関数定義のかっこ内にカンマを挟んで複数定義できます。例 ➤ def(引数1, 引数2, …):

関数の戻り値

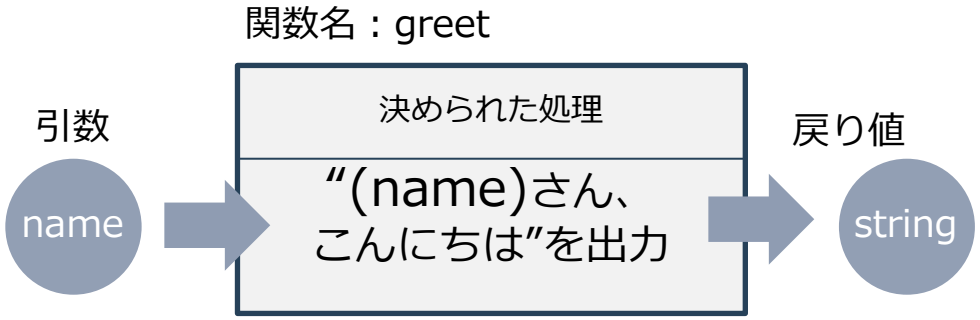
関数から計算結果を受け取りたい場合に**戻り値**を定義します

戻り値のある関数の定義と使い方

def 関数名(引数):
インデント 決められた処理を行うコード
インデント return 戻り値

関数名(引数)で呼び出し、戻り値を変数に代入できます

（オリジナル関数例）



コード例

```
In [1]: def greet(name):
            string = name+“さん、こんにちは”
            return string

        string = greet(‘松尾’)
        print(string)
```

Out [1]： 松尾さん、こんにちは

モジュール

モジュールとは、プログラムを再利用するための機能のこと

- 大きなプログラムを複数の小さなプログラムに分割して管理できる
- 他人が開発したプログラムを簡単に利用できる

importの書式：
Import モジュール名

※モジュール名はファイル名から.pyを除いたファイル名

Importされた関数の呼び出し方法：
モジュール名.関数名()

functions.py

```
In [1]: def Nibai(x):
            return 2 * x
        def Sanbai(x):
            return 3 * x
```

再利用

program.py

```
In [1]: import functions

        f = functions.Nibai(2)
        g = functions.Sanbai(2)
        h = f + g
        print(h)
Out [1]: 10
```

ライブラリ

ライブラリとは、再利用可能な形でまとめられたモジュール群

- Pythonはデータ分析・機械学習のために開発されているライブラリが多数ある
- 仕様を知るだけで便利な道具が使い放題になる！
- この講座では様々なライブラリの使い方を学習する
  (Numpy, Pandas, Matplotlib, Scikit-learn, ……)

Python文法Ⅳ

ライブラリのas

ライブラリの名前が長い場合はasを付けることで短い名前で扱う

importの書式：
Import モジュール名 as 別名

※モジュールを別名で扱うことができます

コード例

```
In  [1]: import numpy
         import pandas

         x = numpy.array([1, 2])
         df = pandas.DataFrame({})
```

コード例

```
In  [1]: import numpy as np
         import pandas as pd

         x = np.array([1, 2])
         df = pd.DataFrame({})
```

as の後に続く単語は自由に設定することができます
しかし、NumpyやPandasのような外部ライブラリは
コミュニティの常識として、npやpdと設定するように決まっています

ライブラリのfrom

fromを使うとモジュール(ライブラリ)で定義された関数・変数・クラスなどを指定してインポートできます

importの書式：
from モジュール名 import 識別子(関数名、クラス名など)

コード例 **fromを使わない例**

```
In  [1] : import math

          print(math.pi)

Out [1] : 3.141592653589793
```

コード例 **fromを使った例**

```
In  [1] : from math import pi

          print(pi)

Out [1] : 3.141592653589793
```

※クラスはこの後説明します

クラス

クラスとは**データと処理をまとめたもの**です

クラス＝データ＋処理(計算・分析など)

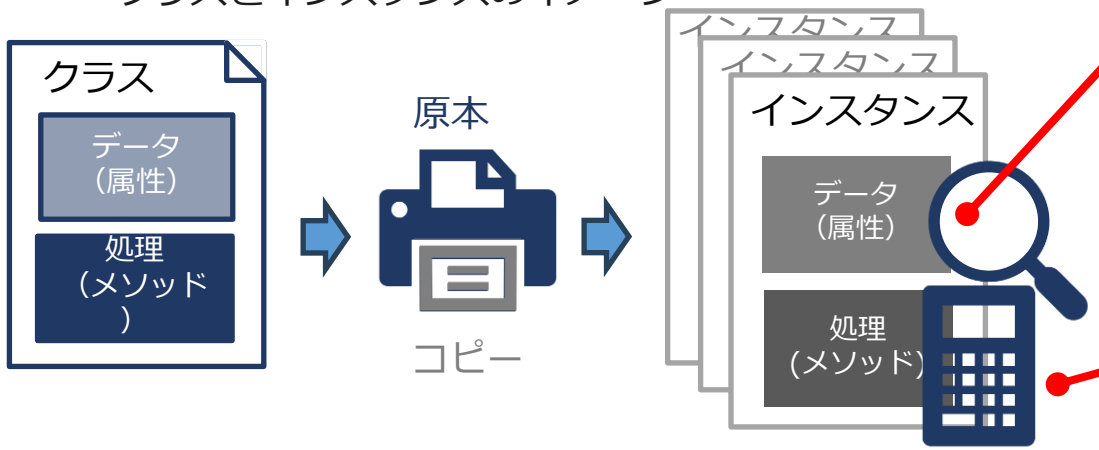
Pythonにはさまざまなクラスがあります

- **表データを管理するクラス**
- **データを可視化するクラス**
- **機械学習モデルのクラス**

クラスの利用方法

クラスのコピーのことをインスタンスと言います
インスタンスは、変数へ代入することで利用できます

クラスとインスタンスのイメージ



属性の参照方法

クラスを代入した変数名**.属性名**

通常の変数の参照と異なり、
クラスを代入した変数名と.（ドット）が必要です

クラスに所属する変数

メソッドの使用方法

クラスを代入した変数名**.メソッド名()**

通常の関数の使用方法と異なり、
クラスを代入した変数名と.（ドット）が必要です

クラスに所属する関数

インスタンスの生成方法

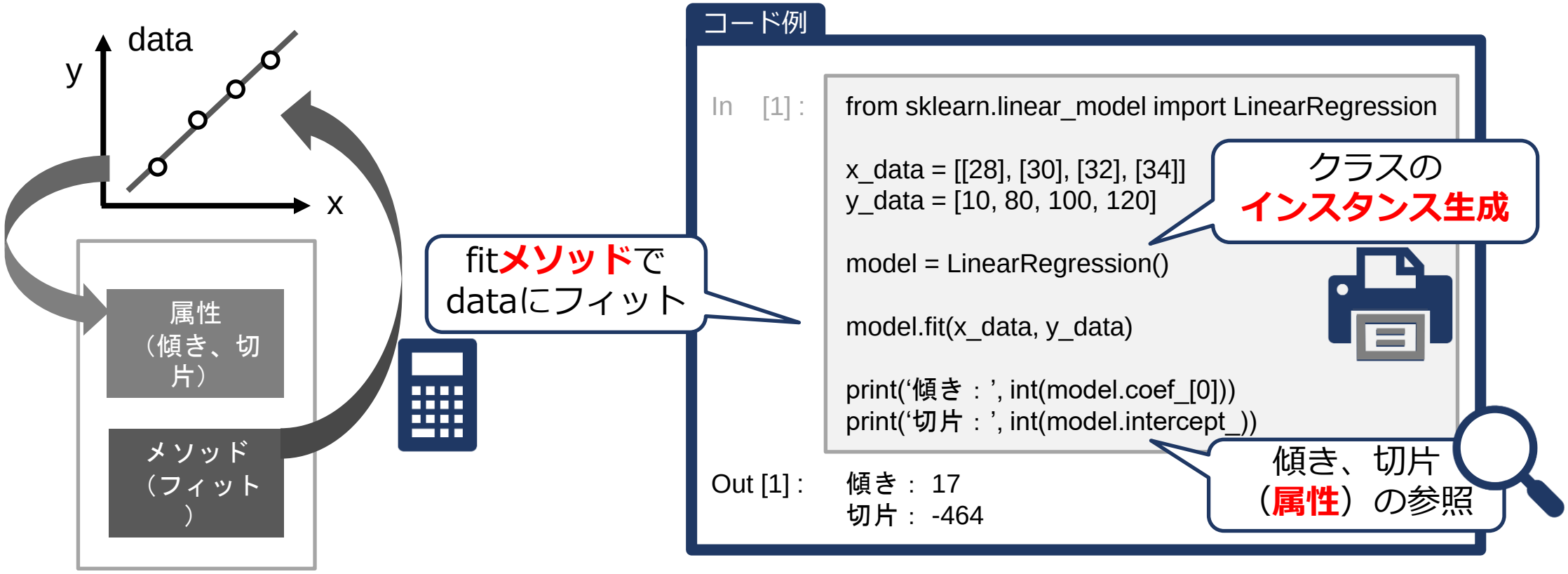
変数名 **= クラス名()**

関数を使用するときのようにかっこは必須です
かっこ内に値を入れる必要がある場合もあります

(発展事項) 機械学習におけるクラスの利用例(Week4にて説明)

線形回帰のクラスでは傾き・切片(属性)を求めるメソッド(Fit)を活用します

（例）線形回帰モデル「y = ax + b」の例 (a:傾き、b:切片)



松尾研究室

MATSUO LAB THE UNIVERSITY OF TOKYO