AI（人工知能）概論【Ⅱ】（Ⅱ）
～　教員のための実践的データサイエンス入門　～

第９講　「データサイエンスにおけるプログラミング基礎」

小松尚登(滋賀大学データサイエンス・
AIイノベーション研究推進センター)

第９講「データサイエンスにおけるプログラミング基礎」

【目　的】

プログラミングは、データの収集、前処理、分析、可視化といった一連の作業を自動化し、効率的に行うためのツールです。第９講では、データサイエンスの基礎を理解するために必要なプログラミングの基礎知識を扱います。特に、構文のシンプルさとライブラリの豊富さからデータサイエンス分野において頻繁に用いられている言語であるPythonを重点的に学びます。

【学修到達目標】

①　プログラミングの基本的な概念と構文を理解し、データ処理や分析に必要な基本操作を実行できる。

②　代表的なプログラミング言語（例：Python）の基本的な文法とライブラリの使い方を習得し、簡単なデータ分析プログラムを作成できる。

目次

1. プログラミングの基本概念
2. Pythonと基本的な文法
   - 変数の代入と基本操作
   - 条件分岐と繰り返し
   - 関数の定義と呼び出し
3. Pythonのライブラリ
4. 大規模言語モデル(LLM)の利用

１．プログラミングの基本概念

プログラミング：あらかじめ定めた計算やファイル操作などの手続きを計算機に処理させるための命令（プログラム）を作ること

プログラミング言語：プログラミングのために開発された言語
…C、Java、Python、Rなど

変数：具体的な値を変化させることができる存在

型：値の種類
- 整数型(int型)：1, 4, 0, -2など
- 浮動小数点型(float型)：0.5, 3.14, -1.0など
- 文字列型(str型)："apple", "banana", "Hello World!"など

プログラミングは変数に対する各種演算、条件分岐、繰り返し、関数の適用などを組み合わせることで、様々な操作をコンピュータに行わせている。

Pythonにおいて、整数型の1と浮動小数点型の1.0は異なる値であることに注意

2. Pythonと基本的な文法

ダウンロード：公式サイト(https://www.python.org/downloads/)から
- ✓ インストーラの指示には極力従うこと
- ✓ 「Add python.exe to PATH」というチェックボックスはチェックを入れることを推奨

実行
- ソースコードをファイルとして用意し、コンソール上でファイル名を指定して実行
  - 拡張子は「.py」にすること。
  - 編集自体は通常のテキストエディタでも可能
- 対話環境にて直接命令を書き込む
  - 毎回命令を入力する必要があるので、関数などの挙動を確認するために使う

対話環境の実行例

2. Pythonと基本的な文法

- 変数の代入：「=」を使用

```
x = 1
fruit = "apple"
```

✓ 変数の名前は「x」のように一文字でも、「fruit」のように複数の文字でも構わない。
✓ 文字列は引用符（' '）または二重引用符（" "）で囲む。

2. Pythonと基本的な文法

- 四則演算：int型やfloat型などの変数間に定義されている

演算子	意味
a + b	和
a - b	差
a * b	積
a / b	(実数としての)商
a // b	(a / b)を超えない最大の整数
a % b	a - b * (a // b)

（実行例）

```
a = 5
b = 2

print(a + b, a - b, a * b, a / b, a // b, a % b)
```

（実行結果）

```
7 3 10 2.5 2 1
```

2. Pythonと基本的な文法

- 条件分岐：ifを使う
（実行例）：aに代入した数(57)が3の倍数かどうか判定するプログラム

```
a = 57

if a % 3 == 0:
    print("a is a multiple of 3")
else:
    print("a is not a multiple of 3")
```

（実行結果）

```
a is a multiple of 3
```

✓ 条件を満たす場合の命令は、インデントを下げて記述する。
✓ 条件を満たさない場合に別の処理をしたい場合、「else:」を使用。
✓ 条件Aを満たさない場合に、別の条件Bを満たしているか判別してから命令をするなら、「else:」ではなく「elif 条件B:」とする。

2. Pythonと基本的な文法

- 比較演算子：変数の大きさを比較したり、等しいかどうかを判別したりする演算子

演算子	意味
`a == b`	aとbが等しい
`a != b`	aとbが等しくない
`a > b`	aがbより大きい
`a >= b`	aがb以上である
`a < b`	aがbより小さい
`a <= b`	aがb以下である

- bool型：「`a == b`」などのように、TrueかFalseの値を持つ

- 論理演算子：bool型の値AとBの間に定義される

演算子	意味
`not A`	Aではない（否定）
`A and B`	AかつB（論理積）
`A or B`	AまたはB（論理和）

2. Pythonと基本的な文法

（論理演算子の使用例）

```
x = True
y = False

print(x and y, x or y)
```

（実行結果）

```
False True
```

✓ ifの後に記述する条件はbool型の値であればよい。そのため、「if a % 3 == 0:」のように条件となる式を明記する場合だけでなく、あるbool型変数xを用いて「if x:」と記述する場合も正しい記法となる。

2. Pythonと基本的な文法

- 繰り返し：forまたはwhileを使う
（実行例）：0から9までの整数を足すプログラム

```
total = 0
for n in range(10):
    total += n

print(total)
```

0以上10未満の整数値を順に取り出す

total = total + n

（実行結果）

```
45
```

✓ 繰り返し行う処理は、インデントを下げて記述する。
✓ 「for (変数名) in (変数を順に動かす範囲を指定するもの):」という構文で繰り返しの際に値を変える変数とその範囲などを指定。
✓ 「変数を順に動かす範囲を指定するもの」をイテラブルという。イテラブルとしては、range()という特殊な関数の他にlist型の値なども使える。

2. Pythonと基本的な文法

- 繰り返し：forまたはwhileを使う

（実行例）：0から9までの整数を足すプログラム

```
total = 0
n = 0
while n < 10:
    total += n
    n += 1

print(total)
```

（実行結果）

```
45
```

✓ 繰り返し行う処理は、インデントを下げて記述する。
✓ 「while (条件):」で「条件を満たす間、指定の処理を繰り返す」という意味
✓ 条件が満たされ続けているとプログラムが止まらないことに注意

※条件分岐でも繰り返しでも、複数行の処理の場合インデントを揃えること。

2. Pythonと基本的な文法

- 条件分岐や繰り返しの中で別の条件分岐や繰り返しを使う
（実行例）：0から9までの整数を列挙し、偶数か奇数かを判定するプログラム

```
for n in range(10):
    if n % 2 == 0:
        print(n, "is an even number.")
    else:
        print(n, "is an odd number.")
```

✓ 条件分岐や繰り返しの中に別の条件分岐や繰り返しを記述する際は、インデントを更に下げる必要あり

2. Pythonと基本的な文法

- 関数：print()などの、特定の処理をまとめて行うもの

…様々な種類が定義されている他、独自の関数を定義して使うことも可能

（実行例）：変数xを入力すると、2*(x + 1)という値を返す関数f(x)を定義し、その関数を実際に使用してみるプログラム

```
def f(x):
    y = x + 1
    return 2 * y


print(f(1))
```

（実行結果）

```
4
```

✓ 関数が行う処理も、インデントを下げて記述する。

3. Pythonのライブラリ

- ライブラリ：プログラム開発者向けに便利なプログラムをまとめたもの
  - ➢ Pythonのメリットの一つは、ライブラリが充実していること

- Pythonライブラリの例
  - NumPy : 配列の操作、特に線形代数に関する演算を高速で行う。
  - pandas : データフレームの作成や欠損値処理など、データ管理において有用な機能が充実している。
  - scikit-learn : 各種の機械学習アルゴリズムを実行する。
  - Matplotlib, seaborn : データをグラフとして可視化する。
  - この他、様々な目的に合わせた多数のライブラリが開発されている。

3. Pythonのライブラリ

- ライブラリのインストール：コンソール上で
「pip install (ライブラリ名)」

と入力
（実行例）

```
>>pip install numpy
```

- ライブラリの使用：ソースコード上で「import (ライブラリ名)」と入力
（実行例）

```
import numpy as np

a = np.array([1.0, 5.0, 3.0])

print(np.sum(a)) # 配列の要素の和
print(np.mean(a)) # 配列の要素の平均
print(np.median(a)) # 配列の要素の中央値
```

4. 大規模言語モデル(LLM)の利用

Pythonを使いこなすには、 Python本体だけでなく、各種ライブラリの機能を熟知する必要あり。

- 従来は教科書やWeb上の資料を熟読する必要があった
- しかし、現在はChat GPTなどの大規模言語モデル(LLM)が精度の高いコードを出力してくれるようになってきている。

（注意点）

- 長く複雑なコードになると齟齬が発生する確率も上がるので、そうした場合は想定している操作全体を幾つかの小さい部分に分け、別々にコードを作ってもらうなどの対処をする方がよい。

- 利用者人口が少ない言語やライブラリに関しては、学習データ不足でコードの精度が落ちる可能性がある。

※逆にPython本体、およびNumPyやpandasなどの主要ライブラリは利用者人口が多く、LLMが参考にできるような学習データが多いので、比較的高確率で正確なコードを書けるようになっている。

4. 大規模言語モデル(LLM)の利用

実行例： ChatGPT 5に「 Matplotlibで関数 y = x**2 のグラフを描くソースコードを作ってください。」と指示
…以下が実際に作成してもらったコード

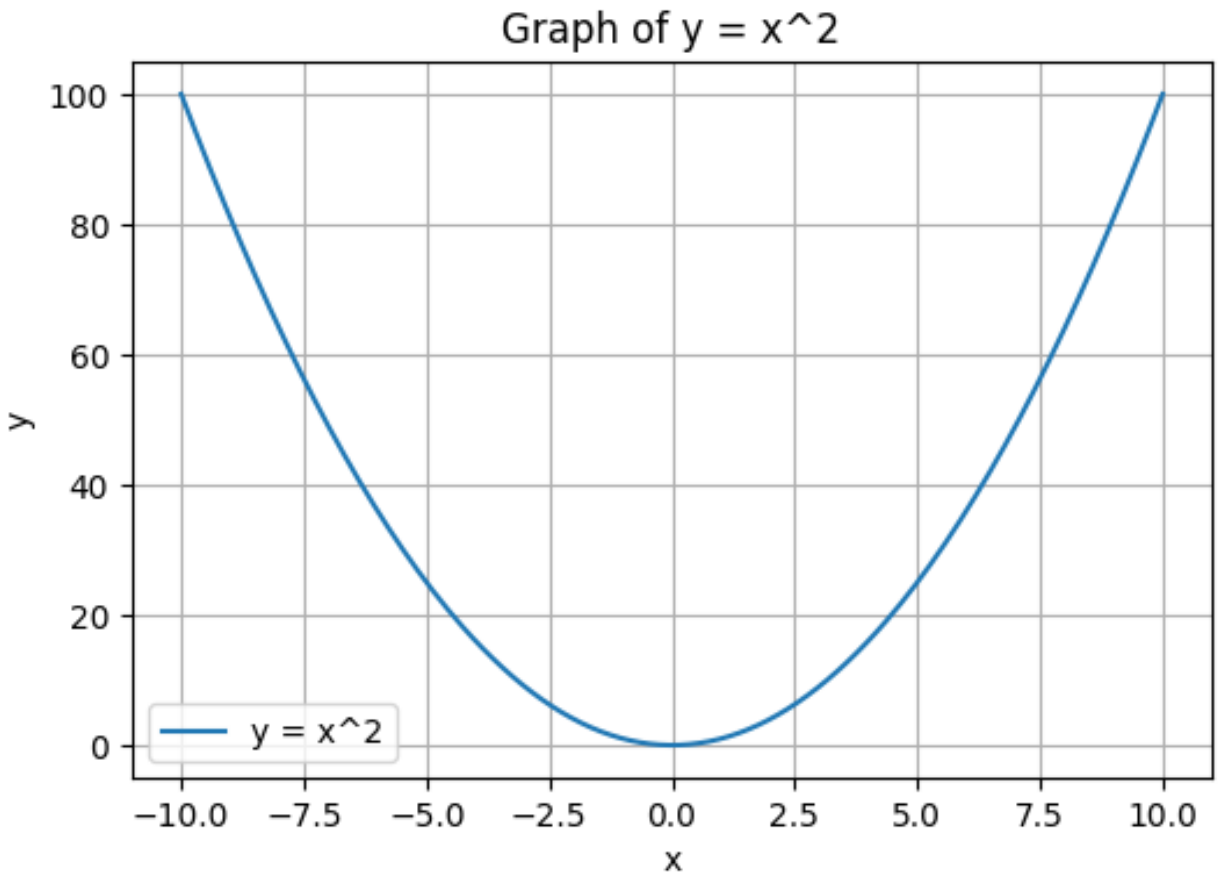
```
import numpy as np
import matplotlib.pyplot as plt

# xの範囲を設定
x = np.linspace(-10, 10, 200)  # -10から10まで200点
y = x ** 2  # 関数 y = x^2

# グラフの描画
plt.figure(figsize=(6, 4))
plt.plot(x, y, label="y = x^2")
plt.xlabel("x")
plt.ylabel("y")
plt.title("Graph of y = x^2")
plt.grid(True)
plt.legend()
plt.show()
```

4. 大規模言語モデル(LLM)の利用

書いてもらったソースコードの実行結果：



課　題

① Pythonを用いて、リストに格納された数値データの平均値と中央値を計算するプログラムを作成してください。

② pandasライブラリを使って、CSVファイルからデータを読み込み、特定の列の欠損値を平均値で埋める処理を行うコードを書いてください。

③ matplotlibやseabornを用いて、データの散布図とヒストグラムを作成し、データの分布や関係性を視覚的に表現してください。