Pythonで学ぶ
データサイエンス入門
(Python編)

もくじ

1.　プログラミングの基礎知識

1.1　アルゴリズムとフローチャート

（①　　）・・・計算や情報処理の手順を定式化したもの

（②　　）・・・アルゴリズムを分かりやすく図式化したもの

表１　主なフローチャート記号（JIS　X　0121）

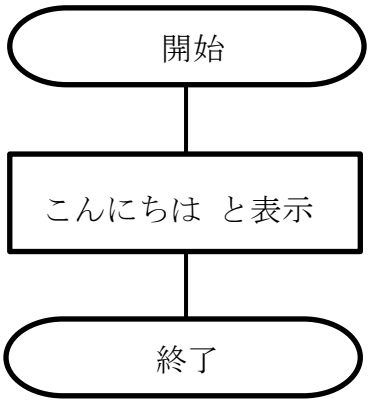
名称	記号	意味
端子		開始と終了
データ		データ入出力
処理		演算などの処理
判断		条件による分岐
ループ始端		繰り返しの始まり
ループ終端		繰り返しの終わり
定義済み処理		別な場所で定義された処理
線		処理の流れ

主なフローチャート記号として表１のようなものがある。アルゴリズムは（③　　）構造、（④　　）構造、（⑤　　）の３つの基本制御構造で表現することができる。

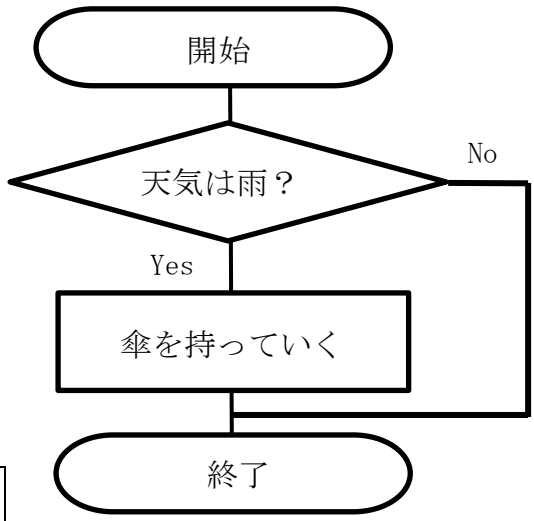
例題1-1

次の（１）～（２）の処理結果はどのようになるか。（１）は出力される文字列を、（２）は正しい方に〇をつけなさい。

（１）



（２）条件：天気は雨



（１）	
（２）	傘を持って（ いく　・　いかない ）

1.2 高水準言語と機械語

一般的にコンピュータは（①　　）と（②　　）で文字や画像を表現する。そのためコンピュータへ指示を出すときも（①）と（②）だけで表現する必要がある。（①）と（②）だけで表現されたプログラムを（③　　）という。当然ながら人間には理解が困難であるため英語に似た（④　　）を使用する。（③）には様々なものがあり種類によって特徴が異なる。

一般的に目にするプログラミング言語は（④）である。

コンピュータでは（④）を理解することができないため、（④）を（③）に翻訳する必要がある。この作業のことを（⑤　　）という。

機械語
10100110101010101010
10101110100010101110
10010011001110010101

高水準言語
print(1+2)
print(‘Hello’)

1.3 プログラミングにおける演算子

プログラミングで演算を実行する場合は少し特殊な記号を使用する場合がある。このときに使用する記号のことを演算子という。演算子には算術演算子の他にも論理演算に使用するものもある。以下の表はPythonにおける算術演算子である。**プログラミング言語によって算術演算子が異なる場合があるので注意。**

表２　Pythonにおける算術演算子

記号	意味
+	足し算
-	引き算
*	かけ算
/	割り算
//	割り算の商を求める（小数点以下切り捨て）
%	割り算の余りを求める
**	べき乗

2. PythonとGoogle Colaboratory

2.1 Pythonの特徴

Pythonは文法がシンプルで分かりやすいという特徴がある。Pythonにはプログラムを作成するための便利な道具（ライブラリ）が用意されていて、様々なサービスでPythonが使われている。例えばYouTubeやGoogle、Instagramなどがある。

以下のソースコードは「Hello World」と表示するプログラムをC言語とPythonで比較したものである。比べてみるとPythonのコードのシンプルさが伝わる。

C言語

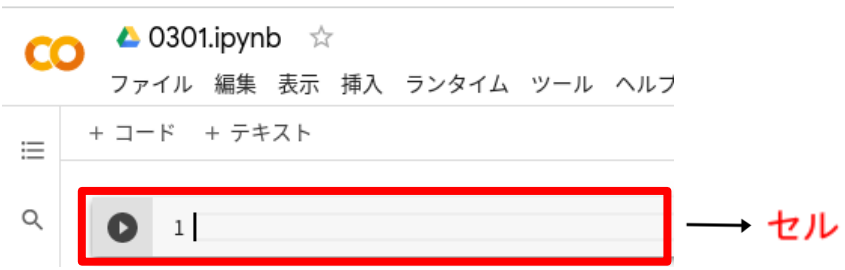
```
#include <stdio.h>
  int main() {
      printf("Hello World"¥n);
      return 0;
  }
```

Python

```
print("Hello World")
```

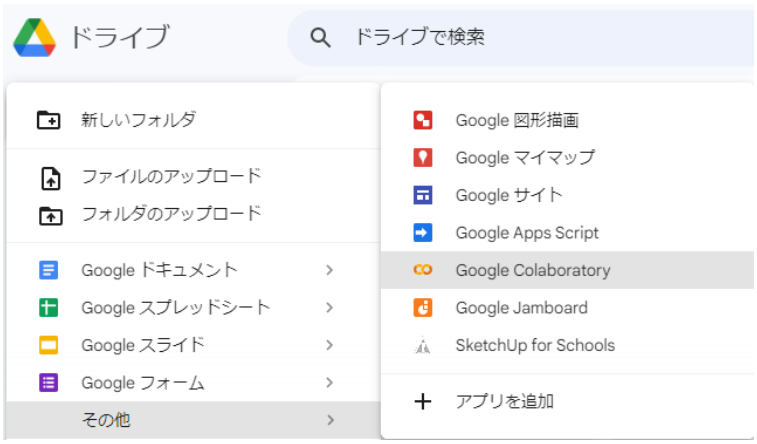
2.2 Pythonの開発環境を準備する

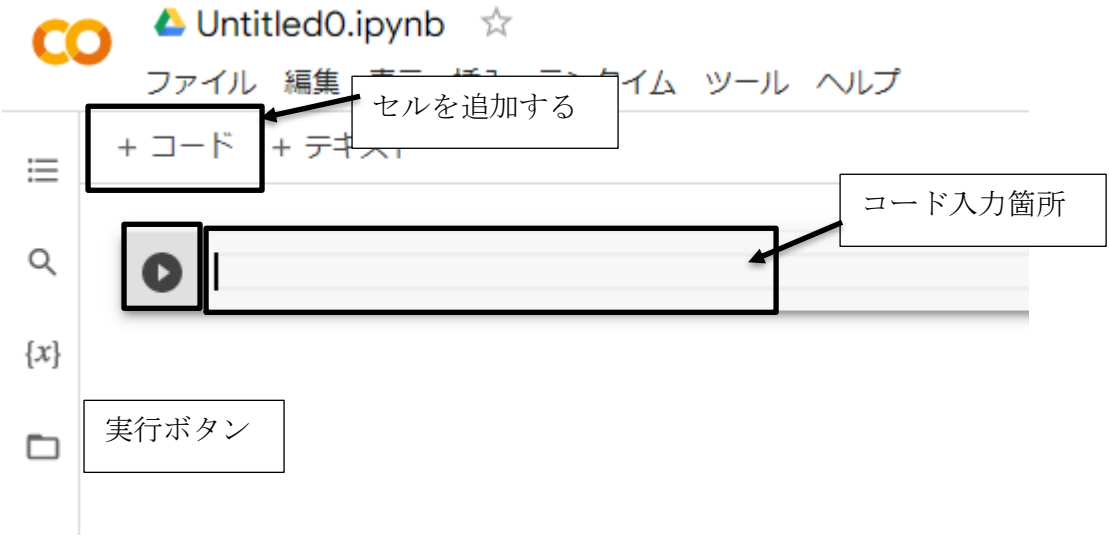
通常であればPython公式サイトからパッケージのインストール（パソコンの中に入れること）をして、ソースコードを記述するためのテキストエディタをインストールしなければならない。しかしこれは面倒なので、Googleのプログラミングができるサービス（Google Colab)を利用する。Google ColabとはGoogleが提供している、機械学習の教育、研究を目的として開発したツール。無料で使用でき、プログラミングの最初の難関ポイントである環境構築が不要。セルと呼ばれる部分にコードを記述し、再生ボタンをクリックするとプログラムを実行することができる。



2.3 Google Colabを開く

Googleドライブを開き、「新規」ボタンをクリックする。その他→Google Colaboratoryと選択する。Googleドライブにフォルダを作成し、Google Colabファイルをまとめておくとよい。





3. Pythonの基礎

3.1 関数

ある定型的な処理をまとめたものを（①　　）という。（①）にはPythonに元から用意されている（②　　）と、自分で作成する（③　　）とがある。

3.2 print関数

print関数は（　）内の変数や文字列、計算結果などを（①　　）してくれる関数。文字列とは文字が１個以上連なったものであり、こんにちはやHelloなどがある。関数や数値、空白はすべて（②　　）で入力する。

例題3-1

(1)　1+2の結果を表示するプログラムを作成しなさい。

(2)　Hello world!と表示するプログラムを作成しなさい。

解説：print(式)で式を表示してくれる。数値はそのまま入力し、文字列はシングルクォーテーション（'）もしくはダブルクォーテーション（"）で囲む。

(2)　print(‘Hello World’)　のように開始と終了の記号は一致させる必要がある。print(“Hello World’)のように開始と終了の記号が一致していない場合エラーとなる。

演習3-1

(1)　10-2×4の結果を表示するプログラムを作成しなさい。

(2)　こんにちはと表示するプログラムを作成しなさい。

3.3 読みやすいコードの作成

プログラミングを行っていく中でコードが複雑になったり、自分以外の人と共同で制作したりする場面がある。そのときに読みやすいコードを作成することが大切である。Pythonでは**演算子の前後で空白や改行**を入れたり、**コメントを入力**したりすることでコードを読みやすくする。空白や改行のルールやPythonにおけるコメントの形式は以下の通り。

空白改行のルール：

Ⅰ　（①　　）の前後は空白OK

Ⅱ　（②　　）の空白、改行NG

Ⅲ　（③　　）の空白NG ※必要な場合もある

Ⅳ　（④　　）の改行はOK、（⑤　　）の改行はNG

Pythonにおけるコメント：

Ⅰ（①　　）以降、（②　　）までがコメントになる

Ⅱ（③　　）もしくは（④　　）で囲まれた部分がコメントになる

　コメントはコードを書く上でとても重要な役割をもつ。他人が書いたコードはもちろん自分が書いたコードでさえ分からなくなるときがよくある。明日の自分は他人だと思い、いつ誰が見ても分かるように、コードの説明をコードの塊ごとに書いておく。また、プログラムを実行したときに意図しない動作になる場合がある。この場合に一部をコメント化することで、プログラムのどこに問題があるのかを特定しやすくすることができる。このように一時的にコメント化し無効化することを**コメントアウト**という。

4. 変数と演算

4.1 データの型

プログラムで扱うデータはすべて型をもつ。とりあえず最初に覚えておきたい型は、（①　　）(int型)、（②　　）(str型)、（③　　）(float型）の３つ。

例題4-1

(1)　3 + ”Hello”を表示するプログラムを作成しなさい。
　　結果：

(2)　”Hello” + ”World”を表示するプログラムを作成しなさい。
　　結果：

(3)　”Hello World” - ”Hello”を表示するプログラムを作成しなさい。
　　結果：

(4)　数値型 1 + 2を文字列型に変換するプログラムを作成しなさい。
　　結果：

(5)　”1 + 2”の型を確認するプログラムを作成しなさい。
　　結果：

解説：

(1)　数値と文字列の足し算は実行不可
(2)　文字列同士の足し算（厳密には結合）は実行可
(3)　文字列同士の引き算は実行不可
(4)　データの型を変換する場合は　データの型(式)　で変換可能
　　例：3.14を整数型へ変換　→int(3.14)
　　3.14を文字列型へ変換 →str(3.14)
　　3を浮動小数点型へ変換→float(3)
(5)　データの型を確認したい場合は　type(式)　で確認できる

演習4-1

(1)　3 * ”Hello”を表示するプログラムを作成し、その結果を答えなさい。
　　結果：

(2)　”3” - 2 を計算するプログラムを作成し、その結果を答えなさい。
　　結果：

(3)　3.14を文字列に変換するプログラムを作成しなさい。

4.2 変数

コードが長く複雑になってくると途中の値を一時的に保存したり、保存したものを複数回使用したりすることがある。そのときに使用するのが変数と呼ばれるものである。数学における変数とは少し異なる部分があり、プログラミングにおける変数は値を入れておくための箱というイメージを持っておくとよい。この変数には区別するために変数名を付けることができる。変数名の付け方には以下のようなルールがある。

Ⅰ　使用できるのは（①　　）、（②　　）、（③　　）
Ⅱ　先頭に（④　　）を使用することはできない
Ⅲ Pythonで役割が与えられたキーワード（予約語）でないこと

Pythonの予約語

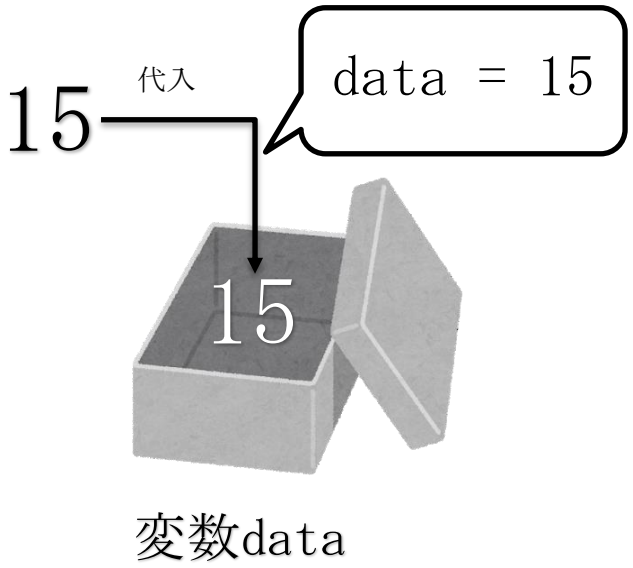
False	None	True	and	as	assert
break	class	continue	def	del	elif
else	except	finally	for	from	global
if	import	in	is	lambda	nonlocal
not	or	pass	raise	return	try
while	with	yield			

また、使用できるが読みやすいコードであることのために以下の点にも注意する。

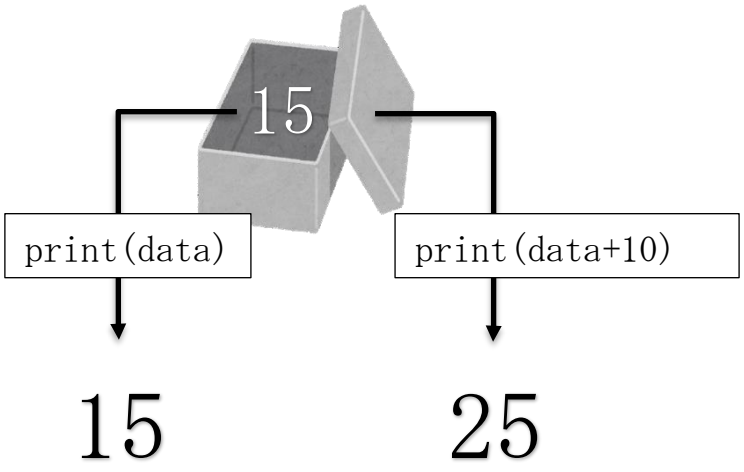
Ⅰ　　組み込み関数(print関数など)でないこと
Ⅱ　　値の意味を類推しやすいこと
Ⅲ　　長すぎず、短すぎないこと
Ⅳ　　一般的には英単語で、単語の間はアンダーバーでつなぐようにする

例題4-2
(1)　dataという変数に15を代入し、dataの値を表示するプログラムを作成しなさい。
(2)　dataに代入されている値に10を足した値を表示するプログラムを作成しなさい。

解説：
(1)　数学におけるイコールとは異なり、Pythonにおけるイコールは、左辺に定義した変数へ、右辺の値を代入するという意味になる。プログラミング言語によっては、最初に変数をデータの型とともに定義したうえで使用するものもあるが、**Pythonは数値と文字列を同じ変数へ代入することが可能。**言語によっては変数の型を最初に宣言しなければ使用できず、宣言した型以外を代入するとエラーになるものもある。また、Pythonでは「a = 1; b = 2」と表記することで、１行で変数a, bを宣言することができる。



(2)　変数を計算した結果を表示することも可能



演習4-2

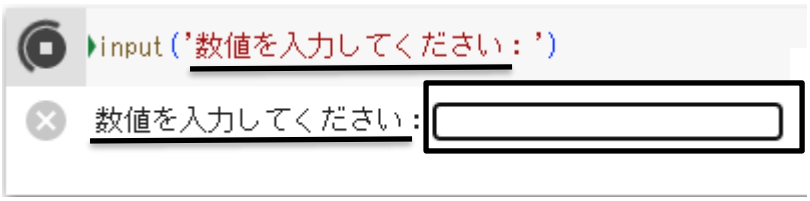
a という変数に 25 を代入し表示するプログラムと a の 4 倍を表示するプログラムを作成しなさい。

4.3 input関数

input関数はキーボードから入力された（①　　）を返す関数。関数には処理で必要な材料を渡すことができる。この材料のことを（②　　）という。そして関数は処理を行った結果を返すことができる。この結果を（③　　）という。関数には引数のみある関数や、引数も戻り値もある関数が存在する。

注意が必要なのが、input関数の戻り値は文字列型であるため、数値の計算にそのままでは使用できないという点である。

input関数の入力について



数値等を入力する場所

例題4-3

(1)　入力した数値を表示するプログラムを作成しなさい。ただし、以下のような結果になるように入力すること。

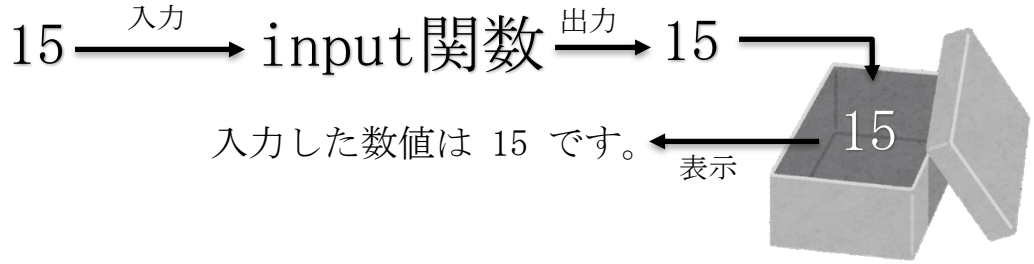
```
数値を入力してください：15
入力した数値は 15 です。
```

(2)　入力した数値を２倍にして表示するプログラムを作成しなさい。ただし、以下のような結果になるように入力すること。

```
数値を入力してください：15
入力した数値の２倍は 30.0 です。
```

解説：変数を使用せずプログラムをかくことも可能

（１）処理のイメージ



print(‘文字列’,変数)のように文字列と変数を(　)の中に記述するときには、文字列と変数の間をカンマ(,)で区切る必要がある。

（２）input関数の戻り値は文字列型。そのまま 2 倍しようとすると、文字列型と整数型の掛け算となり、1515 という出力結果になる。そのため、input関数の戻り値を浮動小数点型に変換する必要がある。

演習4-3

身長と体重をキーボードから入力し、BMIを表示するプログラムを作成しなさい。ただし、以下のような結果になるように入力すること。　※BMI = (体重(kg))÷(身長(m))^2

体重(kg)を入力してください：53.5
身長(m)を入力してください：1.65
結果：19.65105601469238

5. 分岐

5.1 比較演算子

比較演算子は、比較結果を（①　　）として返す。（①）は（②　　）、（③　　）のいずれかの値を表す。（④　　）や（⑤　　）とも呼ばれ、コンピュータ内部での数値は（②）が（⑥　　）、（③）が（⑦　　）で表現される。この比較演算子は数値だけでなく、文字列などの比較も可能。比較演算子には以下のようなものがある。

演算子	判定する内容
==	左辺と右辺が等しいか
!=	左辺と右辺が等しくないか
<	左辺が右辺より小さいか
<=	左辺が右辺以下か
>	左辺が右辺より大きいか
>=	左辺が右辺以上か
in	左辺が右辺に含まれているか

例題5-1

(1) 10と10が等しいかどうか確認するプログラムを作成しなさい。

(2) ’あいう’という文字列と’えおか’という文字列が等しいか比較するプログラムを作成しなさい。

(3) ’abc’ という文字列が ’xyz’ という文字列より大きいか比較するプログラムを作成しなさい。

(4) ’あいうえお’ という文字列に’いう’という文字列が含まれているか確認するプログラムを作成しなさい。

解説

(3) 文字列の大小比較は辞書式で比較する。例えば ’a’ < ’b’ であり、’abc’ < ’abd’ である。文字数を比較しているわけではないので注意。

演習5-1

(1) 5と10が等しくないかどうか確認するプログラムを作成しなさい。

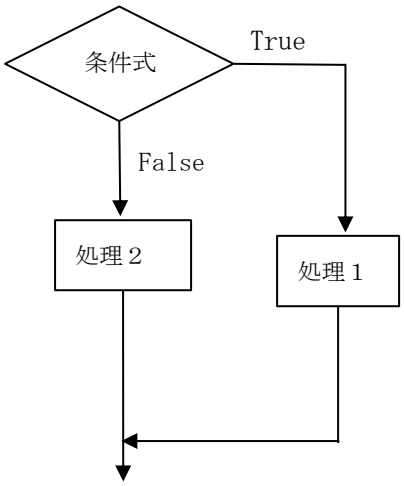
(2) ’abcd’ と ‘acbd’ の文字列の大小比較をした場合、True になるようなプログラムを作成しなさい。

(3) ’あう’ という文字列が ’あいうえお’ に含まれているか確認するプログラムを作成しなさい。

5.2　if文

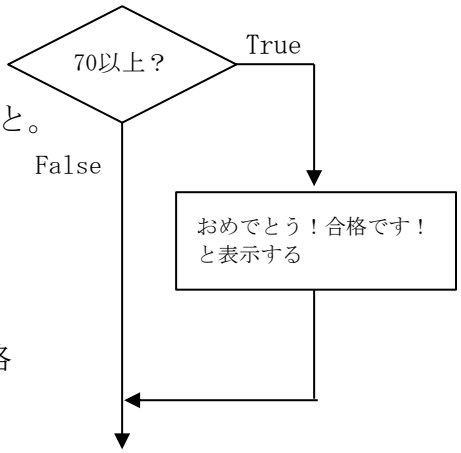
if文は以下の構文で表され、条件式が真であれば処理１を実行し、条件式が偽であれば処理２を実行する。プログラミングでは命令文の塊を（①　　）といい、言語によって（①）をどのように表すか異なる。例えばC言語では {} がひとつの（①）として表される。Pythonでは（②　　）によって（①）を表している。そのため必要なところで（②）がされていなければエラーとなる。（②）にはスペースを利用するものとタブを利用するものがある。基本的にどちらを利用しても構わないが、タブを利用する場合はタブの幅の設定や別の環境にコピーした場合のエラーに注意する必要があるため、スペースで空白を設定するほうが無難である。また、スペースの幅は半角スペース４つが一般的である。

```
if 条件式:
    処理1
else:
    処理2
```



例題5-2

(1)　input関数を用いて数値を入力し、70以上であれば
おめでとう！合格です！ と表示するプログラムを
作成しなさい。ただし、以下の結果になるように入力すること。



あなたの点数は？ 75
おめでとう！合格です！

(2)　(1)のプログラムにおいて70未満であれば 残念、不合格
と表示するプログラムを追加しなさい。

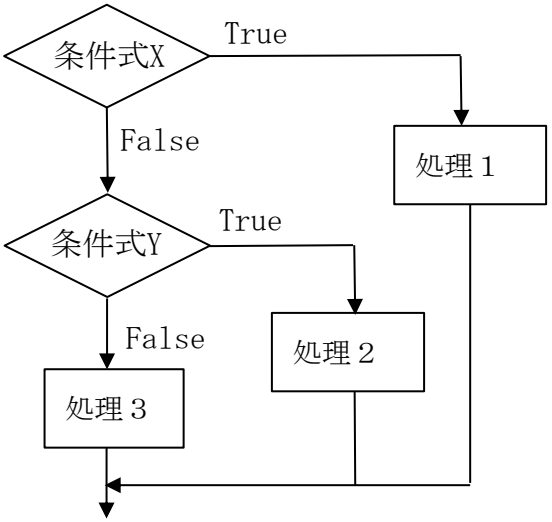
演習5-2

input関数を用いて数値を入力し、1以下であれば 当たり と、1より大きければ はずれ と表示するプログラムを作成しなさい。

5.3　複雑な分岐処理

if … else 文では「もし○○○なら□□□しなさい、そうでなければΔΔΔしなさい。」というように２つに分岐させることができた。しかし60以上、40以上60未満、40未満という３つに分岐したい場合もある。このときは（①　　）を用いて複数に分岐させることができる。※if…elseを組み合わせてもできる。

```
if 条件式X:
    処理１
elif 条件式Y:
    処理２
else:
    処理３
```



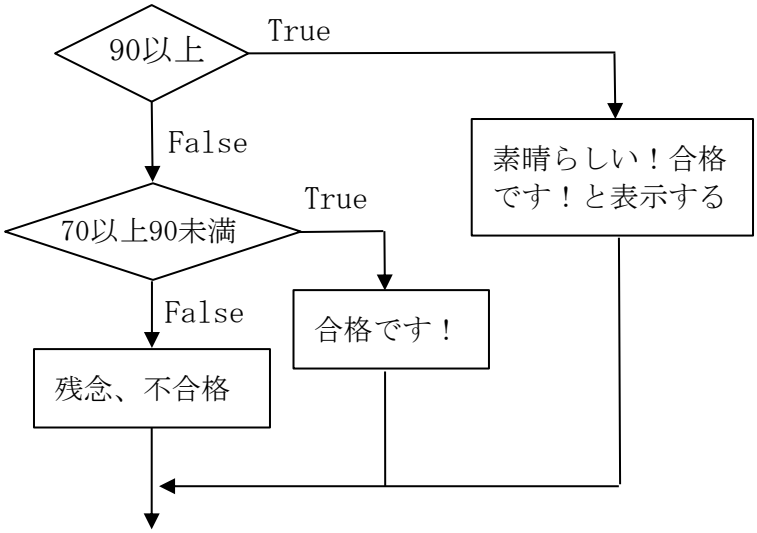
例題5-3

キーボードから入力した数値が90以上、70以上90未満、70未満で場合分けをし、以下のように出力するプログラムを作成しなさい。3回実行して正しく動くことを確認すること。

```
あなたの点数は？　95
素晴らしい！合格です！

あなたの点数は？　75
おめでとう！合格です！

あなたの点数は？　50
残念、不合格
```



演習5-3

キーボードから入力した数値が30未満なら1、30以上40未満なら２、40以上65未満なら3、65以上79未満なら4、80以上なら5を出力するプログラムを作成しなさい。

6.　繰り返し

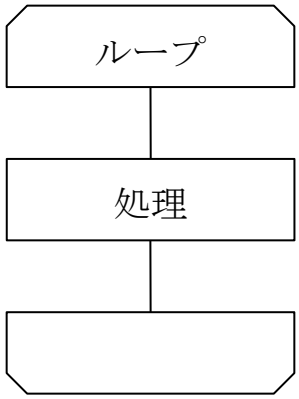
6.1　条件を満たしている間繰り返す

プログラムでは同じことを繰り返したい場合がある。例えば、「あいうえお」という文字列を１０回表示したいとする。このときに

```
print(‘あいうえお’)
print(‘あいうえお’)
      :
print(‘あいうえお’)
```

と書くとコードが長くなり複雑になる。そこで繰り返し（ループ）を用いる。条件を満たしている間繰り返す処理には **while文**を用いる。

```
while　条件式:
    処理
```



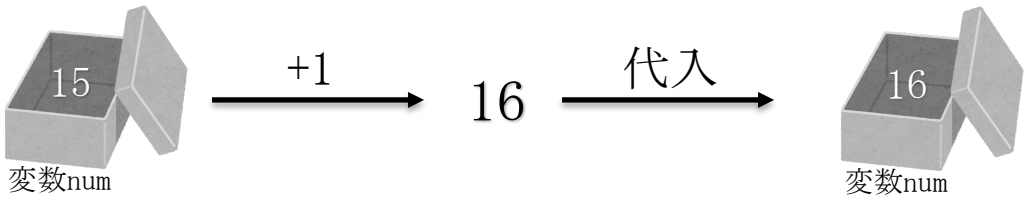
ループ処理の流れ

例：

```
num = 1
while num < 4:
    print(num)
    num += 1
```

①num = 1でありnum < 4を満たす
②print(num)を実行
③num を1から2へ増やす
④num = 2でありnum < 4を満たす
⑤print(num)を実行
⑥num を2から3へ増やす
⑦num = 3でありnum < 4を満たす
⑧print(num)を実行
⑨numを3から4へ増やす
⑩num = 4でありnum < 4を満たさない
⑪ループを抜ける

+= のように代入とその他の演算を組み合わせた演算子のことを複合代入演算子という。例えば「num += 1」は「num = num + 1」と同じ意味であり、num という変数に 1 を加えて、num という変数に代入するという意味になる。

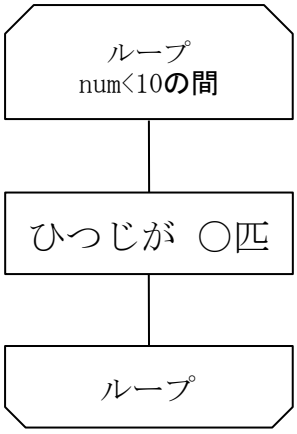


実際にプログラムをかいたり、問題を解いたりするときには、上図のようにひとつひとつ処理を追っかけていくことが大切。

例題6-1

numという変数を定義し、numが10未満のとき「ひつじが○匹…」と出力されるプログラムを作成しなさい。

出力結果：ひつじが１匹…
ひつじが２匹…
：
ひつじが８匹…
ひつじが９匹…



演習6-1

(1)かけ算の７の段を出力するプログラムを作成しなさい。ただし、出力は以下のようにすること。

出力結果：7 ＊ 1 = 7
7 ＊ 2 = 14
7 ＊ 3 = 21
7 ＊ 4 = 28
7 ＊ 5 = 35
7 ＊ 6 = 42
7 ＊ 7 = 49
7 ＊ 8 = 56
7 ＊ 9 = 63

(2)入力された数値と 1 ～ 9 の数値を掛けた結果を表示するプログラムを作成しなさい。ただし、出力は以下のようにすること。

出力結果：整数を入力してください：6
6 ＊ 1 = 6
6 ＊ 2 = 12
6 ＊ 3 = 18
6 ＊ 4 = 24
6 ＊ 5 = 30
6 ＊ 6 = 36
6 ＊ 7 = 42
6 ＊ 8 = 48
6 ＊ 9 = 54

6.2　指定された回数だけ処理を繰り返す

6.1ではwhile文を用いてループ処理を行ったが、指定された回数だけ処理を行うときにはfor 文を用いる。Pythonのfor文では、新しくrange関数を用いる。

```
for 変数名 in range():
    処理
```

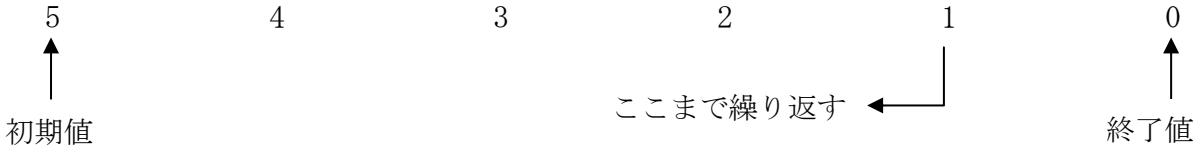
range()には初期値、終了値、増分値を入力する。初期値はデフォルトで0が設定されているので注意が必要。増分値を省略すると１ずつ増やすことができる。

```
range(初期値, 終了値, 増分値)
```

range関数の使用例：
range(10)　　→　10回（0,1,2,3,4,5,6,7,8,9）繰り返す
range(5, 10)　→　5から9まで繰り返す
range(0, 10, 3)　→　0, 3, 6, 9 と繰り返す
range(10, 0, -1)　→　10, 9, 8, 7, 6, 5, 4, 3, 2, 1 と繰り返す

例題6-2
（１）例題6-1のプログラムをfor文で記述しなさい。
（２）例題6-1の1匹～9匹であった部分を、5匹～9匹になるようにしなさい。
（３）以下のように出力されるプログラムを作成しなさい。
　　ひつじが 5 匹…
　　ひつじが 4 匹…
　　ひつじが 3 匹…
　　ひつじが 2 匹…
　　ひつじが 1 匹…

解説：
（１）range(10)とすると0～9まで繰り返す。1～9まで繰り返すように初期値を追加すればよい。Pythonのrange関数の初期値は0であることに注意。
（３）増分値を負の数にすることで数字を減らしながら繰り返すことができる。



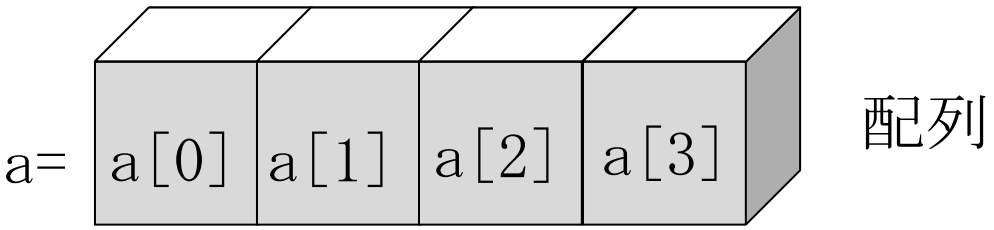
初期値から終了値まで数字を並べたときに、終了値のひとつ手前の数字まで処理を繰り返すことになる。

演習6-2
1 ～ 100 までの合計を求めるプログラムを、for文を用いて作成しなさい。
※5050 になれば正解です。

7. 配列

同じ型の変数をいくつも集めて一つの名前を付けたものを（①　　）という。この一つ一つの変数を（②　　）という。配列の（②）は配列名を番号で表す。この番号を（③　　）という。**プログラミング言語によってこの（③）は、０から始まるものと１から始まるものがあるので注意が必要。**

Pythonにおける配列の役割を持っているのがリストである。リストは配列と異なり、違う型の変数を入れることができるなど、リストと配列は異なるものである。以下、配列という単語はPythonにおけるリストとして考える。



複数の変数名を用意することなく、複数の変数を扱うことができるのが配列のメリットである。配列を定義するときには a = [変数0, 変数1, 変数2, …] のように変数同士をカンマで区切って記述する。

例題7-1

以下の配列を作成し、以下の問いに答えなさい。

r = [2, 4, 6, 8, 10]

（１）r[3] の値を表示するプログラムを作成しなさい。

（２）r[0] + r[4] の値を表示するプログラムを作成しなさい。

（３）配列rの要素数を表示するプログラムを作成しなさい。

※len(a)で配列aの要素数を返すことができる

解説：結果の表示にはprint関数を用いる。配列の要素を取り出すときには、r[3]のように[]内に添え字を入力する。

演習7-1

以下の配列の作成し、以下の問いに答えなさい。

a = [1, 3, 5, 7, 9, 11]

（１）a[0] の値を表示するプログラムを作成しなさい。

（２）a[1] × a[4] の値を表示するプログラムを作成しなさい。

（３）配列aの要素数を表示するプログラムを作成しなさい。

8.　探索プログラム

8.1　計算量

コンピュータの計算速度は人間と比べると速いが、コンピュータの計算速度にも限界がある。1秒間に10億回（10^9回）計算できるコンピュータで、計算回数が1京回（10^{16}回）必要なプログラムを実行すると、10^7秒（約115日）かかることになる。とても現実的な数字ではないので、プログラムを書くときに効率の良いプログラムを書くことが大切である。

ここで計算回数とは「答えが出るまでに行う演算の回数」のことである。しかし現実で計算回数をきっちり算出することはほとんど不可能なので、計算回数がざっくり何回かを考える。これを**計算量**といい、**O（オーダ）**で表す。計算量はざっくりした計算回数を考えるので、2N+1の値を求める計算は2Nの計算と+1の計算の２回行っているが、計算量としては1回も2回も変わらないと考えることができ、計算量は１回であると言ってよい。このときの計算量は**O(1)**と表す。

8.2　線形探索

たくさんのデータの中から目的のデータを探し出すことを（①　　）という。最も単純なアルゴリズムは、最初のデータから順番に見ていくものである。これを（②　　）という。イメージは以下の通り。

探したいデータ：3

A[0]	A[1]	A[2]	A[3]	A[4]	A[5]	
5	2	3	4	6	1	A[0]と3が等しいかチェック　5≠3
5	2	3	4	6	1	A[1]と3が等しいかチェック　2≠3
5	2	3	4	6	1	A[2]と3が等しいかチェック　3＝3

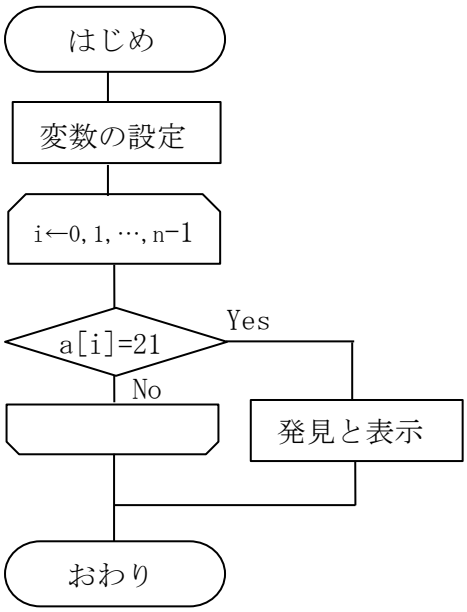
目的のデータはA[2]に存在した。

（②）の探索アルゴリズムは単純であり、データ量が多くなれば探索に時間がかかる。データ数 N=6 であれば、最大6回の比較で終わるが、N=5100 であると最大5100回の比較が必要になる。このように線形探索の計算量はデータ数Nに比例するので、計算量は**O(N)**である。

例題8-1

配列a=[4, 10, 54, 21, 48, 36] の中から21を線形探索で探し出すプログラムを作成しなさい。また、目的の数値が見つかったときは「発見」と表示させなさい。

解説：




```
プログラムの解答例
a = [4, 10, 54, 21, 48, 36]
n = len(a)
s = 21
for i in range(0, n):
    if a[i] == m:
        print(‘発見’)
        break
```

※while文を用いたプログラムを記述することも可能

プログラムの１～３行目で配列a、配列aの要素数、探し出す数字(21)の変数を定義する。for文のiは配列aの添え字を表し、このiを変化させることでa[0]～a[5]までを探索する。配列の添え字は0からなので、range関数の初期値は0を指定し、配列の最後の添え字5まで繰り返すので、終了値はnまでを指定する。５行目のif文は配列の要素とsが一致するかを確かめ、一致すれば発見と表示させループを抜け、一致しなければループが続くようにする。**break を記述することで、ループを抜けることが可能。**具体的な処理の流れは以下の通り。

①配列a、n、sを定義する。このとき、len(a)=6であるのでn = 6
②for i in range(0, n)とあるので、i を0からn-1=5まで繰り返す。まずはi = 0 として処理を行う。
③i = 0 であるので、a[0] = 4 と s = 21 が等しいかチェックする。
④4≠21よりi = 1 としてループ継続。
⑤i = 1 であるので、a[1] = 10 と s = 21 が等しいかチェックする。
⑥10≠21よりi = 2 としてループ継続。
⑦i = 2 であるので、a[2] = 54 と s = 21 が等しいかチェックする。
⑧54≠21よりi = 3 としてループ継続。
⑨i = 3 であるので、a[3] = 21 と s = 21 が等しいかチェックする。
⑩21＝21であるので発見と表示しループを抜ける。

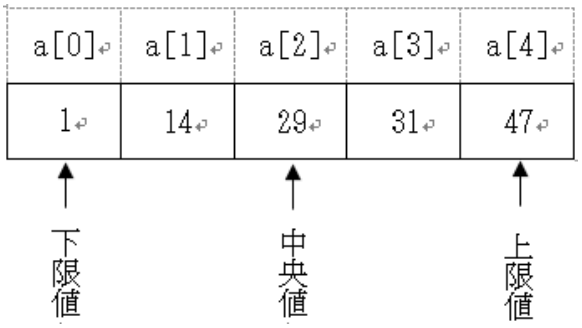
演習8-1

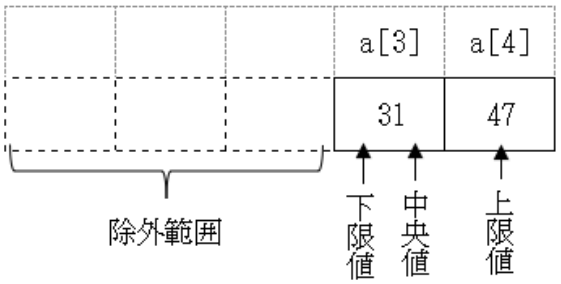
例題8-1で探索する数値をキーボードから入力し、入力した数値を探索するプログラムに書き換えなさい。

8.3　二分探索

データを二分しながら探索を繰り返して探索値を絞り込むアルゴリズムを（①　　）という。二分探索の手順は、探索範囲の中央値と探索値の大小比較を繰り返して探索範囲を絞り込んでいく。二分探索を行う場合は、**データが昇順（小さい順）または降順（大きい順）に整列されている必要がある。**

例：配列a=[1, 14, 29, 31, 47] の中から探索値31を探し出す手順
①配列の添え字の中央値は(0+4)÷2=2であるので、中央値はa[2]=29。
②中央値である29と探索値31の大小を比較する。
③29<31であるので、探索値は右側に存在することがわかる。よって探索範囲をa[3]からa[4]に絞り込むことができた。
④(3+4)÷2=3.5であるので、中央値はa[3]=31。
※整数の除算であるので、**小数点以下は切り捨てる**処理を行う
⑤31=31であるから目的の数値はa[3]に存在することが分かった。
この例を図で説明すると以下のようになる。





この例ではデータ数N=5に対して、探索回数は２回であった。また、最小の探索回数は１回、最大の探索回数は３回となる。二分探索ではデータ数Nに対して、最大の探索回数が $\left\lfloor \log_2 N \right\rfloor + 1$ であることが分かっている。二分探索の計算量は$\mathbf{O}(\log_2 N)$である。

補足
$\log_2 N$ は2を何乗すればNになるかを表すものである。（数学Ⅱ対数関数で学習）
例えば$\log_2 8 = 3$ である。$\lfloor x \rfloor$ は x を超えない最大の整数を表す。例：$\lfloor 3.5 \rfloor = 3$

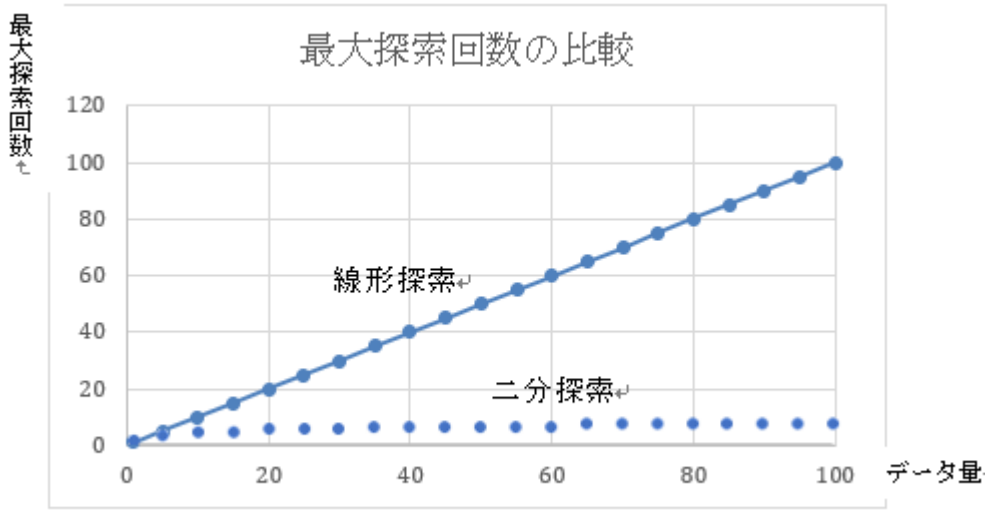


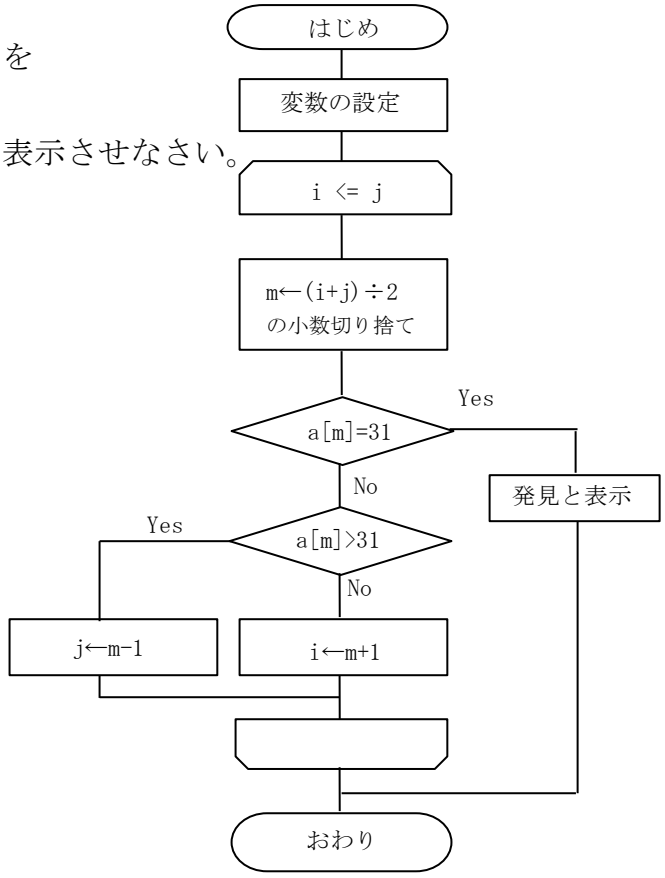
図 1:データ量に対する最大探索回数の比較

	1	2	3	4	5	…	98	99	100
線形探索	1	2	3	4	5	…	98	99	100
二分探索	1	2	2	3	3	…	7	7	7

表 1:線形探索と二分探索のデータ量に対する最大探索回数の比較

例題8-2

配列a=[1, 14, 29, 31, 47]の中から探索値31を
二分探索で探し出すプログラムを作成しなさい。
また、目的の数値が見つかったときは「発見」と表示させなさい。



解説：

```
プログラムの解答例

a = [4, 10, 54, 21, 48, 36]
n = len(a)
s = 31
i = 0    #下限値の添え字
j = n-1  #上限値の添え字

while i <= j:
    m = int((i + j)/2)  #中央値
    if a[m] == s:
        print('発見')
        break
    if a[m] > s:
        j = m - 1
    else:
        i = m + 1
```

演習8-2

例題8-2で探索する数値をキーボードから入力し、入力した数値を探索するプログラムに書き換えなさい。

9.　整列プログラム（交換法）

　ある規則に従って配列の中の隣り合うデータの大小を比較し交換を行うことで整列（ソート）させるアルゴリズムをバブルソート（交換法）という。

例：a=[29, 31, 5, 6, 58]　を昇順に並べ替える(n=len(a)=5)

i=3
- ①a[0]=29とa[1]=31の大小を比較する
  a[0]<a[1]よりそのまま
- ②a[1]=31とa[2]=5の大小を比較する
  a[1]>a[2]であるのでa[1]とa[2]を入れ替える
  ここで、a[1]=5, a[2]=31となる
- ③a[2]=31とa[3]=6の大小を比較する
  a[2]>a[3]よりa[2]とa[3]を入れ替える
  ここで、a[2]=6, a[3]=31となる
- ④a[3]=31とa[4]=58の大小を比較する
  a[3]<a[4]よりそのまま
  a[4]=58が確定

a[0]	a[1]	a[2]	a[3]	a[4]
29	31	5	6	58
29	31	5	6	58
29	5	31	6	58
29	5	6	31	58

i=2
- ①a[0]=29とa[1]=5の大小を比較する
  a[0]>a[1]よりa[0]とa[1]を入れ替える
  ここで、a[0]=5, a[1]=29となる
- ②a[1]=29とa[2]の大小を比較する
  a[1]>a[2]よりa[1]とa[2]を入れ替える
- ③a[2]=29とa[3]の大小を比較する
  a[2]<a[3]よりそのまま
  a[3]=31が確定

a[0]	a[1]	a[2]	a[3]	a[4]
29	5	6	31	58
5	29	6	31	58
5	6	29	31	58

i=1
- ①a[0]=5とa[1]=6の大小を比較する
  a[0]<a[1]よりそのまま
- ②a[1]=6とa[2]=29の大小を比較する
  a[1]<a[2]よりそのまま
  a[2]=29が確定

a[0]	a[1]	a[2]	a[3]	a[4]
5	6	29	31	58
5	6	29	31	58

i=0
- ①a[0]=5とa[1]=6の大小を比較する
  a[0]<a[1]よりそのまま
  a[0]=5, a[1]=6が確定

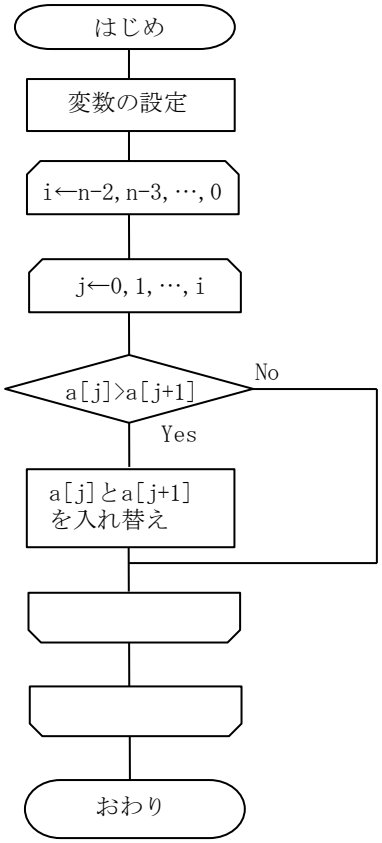
a[0]	a[1]	a[2]	a[3]	a[4]
5	6	29	31	58

整列完了

データの件数が n の場合、比較回数は$(n-1)+(n-2)+\cdots 2+1=\frac{n(n-1)}{2}$となる。
また、計算量は$\boldsymbol{O(n^2)}$である。
交換法をPythonのプログラムで記述すると次のページのようになる。

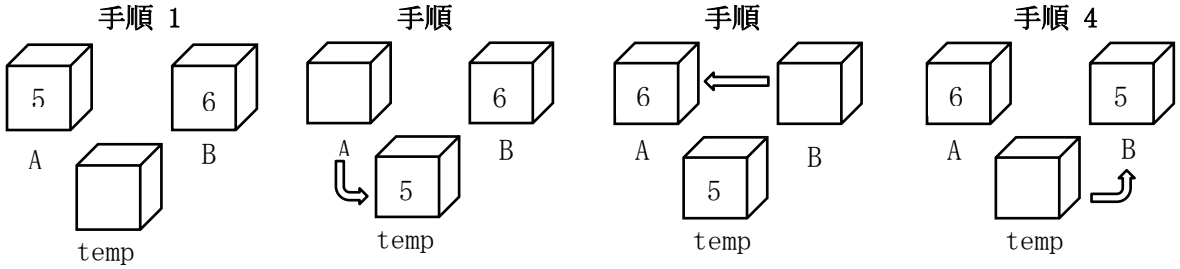
プログラムの解答例

```
a = [29, 31, 5, 6, 58]
n = len(a)
temp = 0
for i in range(n-2, 0, -1):
    for j in range(0, i):
        if a[j] > a[j+1]:
            temp = a[j]
            a[j] = a[j+1]
            a[j+1] = temp
print(a)
```



１～３行目は変数の定義。aは並べ替えを行う配列。
nはデータの個数。tempはデータの入れ替えを行うために一時的にデータを逃がす変数。

例：AとBのデータを入れ替える場合



※Pythonは逃がす変数を定義せずに入れ替え可能