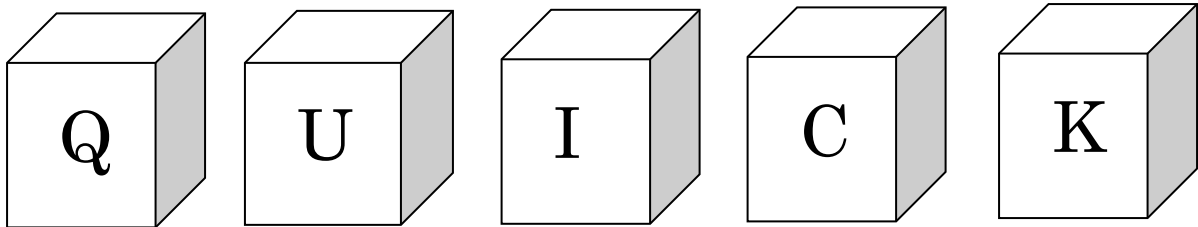


## Quick Check

### ▶ Grammer & Usage



>> Python で学び直すアルゴリズムの基礎①  
～フローチャートと代表的なソートを理解しよう

近年、初等・中等教育課程でプログラミングの授業が行われるようになったこともあり、「どのように問題を解いていくか？」という処理手順（＝アルゴリズム）を考える視点が注目されています。世の中のニーズに合わせて、Python などのプログラミングを通してモノづくりに興味を持つ生徒・学生も増えることでしょう。






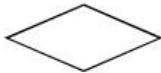



今回はアルゴリズムについての取っ掛かりとして、ちょっとした頭の体操をしていければと考えています。具体的には、基礎となるフローチャートの書き方と代表的な整列（sort）アルゴリズムについて考えていきます。基礎知識を学ぶことで、開発に対する素養を身に付けていきましょう。

それでは、アルゴリズムの基本的な知識を確認していきたいと思います。

## ○ アルゴリズムとは？

アルゴリズムとは、「問題を解くための処理手順を定式化したもの」

アルゴリズムの記述法としては、フローチャート（流れ図）が有名です。以下の記号を用いて、処理内容を理解しやすいように簡潔に記述します。

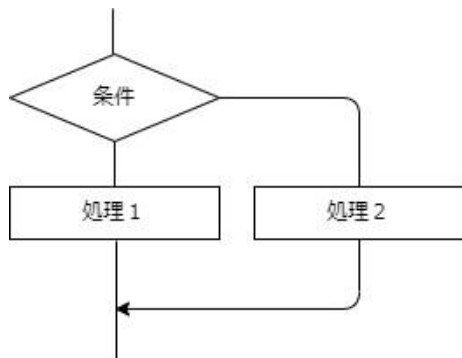
 記号	 名称	 説明
	端子	フローチャートの開始と終端を表す記号
	処理	実行する処理を表す記号
	判断	条件によって次に行う処理を選択するという処理を表す記号
	線	処理の流れを表す
	ループ始端	繰り返しの始まりを表す記号
	ループ終端	繰り返しの終わりを表す記号

アルゴリズムは一般的に上から下、左から右に処理が流れていき、構造としては、「順次構造」「選択構造」「繰り返し構造」があります。

順次構造：各計算や操作が直線的につながっている構造



選択構造：条件により処理内容が分かれる構造



繰り返し構造：終了条件を満たすまで、一連の処理を繰り返すという構造



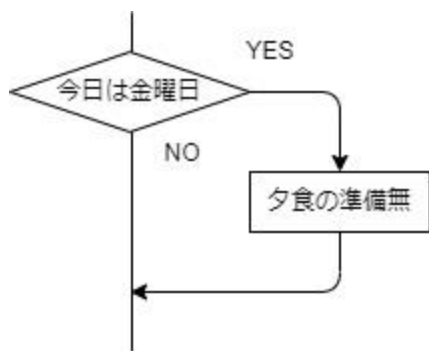
## 問題

以下のアルゴリズム（問題を解くための処理手順）をフローチャートで表してください。

- 1) 「金曜日は夕食の準備をしない。」
- 2) 「同じ文字列を 5 回表示する。」

### 1 に関する解答例

条件（金曜日であるか）によって処理を分岐させます。「選択構造の記号」を用いて、フローチャートを記載していきます。



### 2 に関する解答例

5 回文字列を表示するという処理を繰り返す構造を、フローチャートで記載していきます。



⇒ ここで、添付資料「フローチャート作成」シートにある演習問題を解いてみましょう

参考 URL :

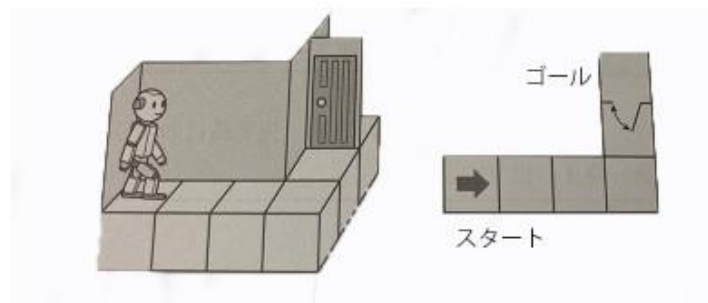
<http://open.shonan.bunkyo.ac.jp/~ohtan/jugyo/old/old2007aki/C-flowchart-2007.pdf>

## 演習

文研出版『教科書ガイド 実教出版版 情報Ⅰ』より抜粋

下図のような迷路がある。次の4つのルールに従って、ゴールに達するまでのアルゴリズムをフローチャートで表しなさい。

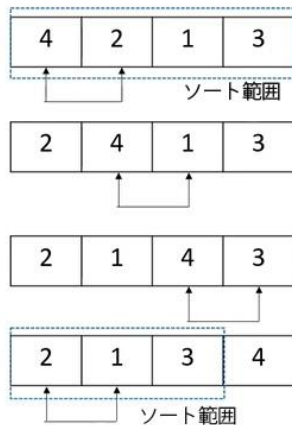
- ・使用できる処理は、前に進む(1マス進む)、右に90°向く、左に90°向く、ドアを開けるの4つである。
- ・ゴールにはドアが付いている。また閉まっている場合は、ドアを開けてから入る必要がある。ドアの前に行くまで、ドアが開いているか閉まっているかはわからない。
- ・ドアが開いている状態で、「前に進む」と到着となる。
- ・3つの基本制御構造をすべて使用する。



次は、代表的な整列アルゴリズムについて、Python で考えていきましょう。

### ・バブルソート（基本交換法）

バブルソートは、データの端から順番に隣り合ったデータ同士を比較し、順番が逆ならば入れ替える操作を繰り返すことによって整列をします。



## 問題

以下のリストについてバブルソートを用いて、昇順に並び替えてください。

```
target_list = [6,3,4,8,7,4,1,9,2]
```

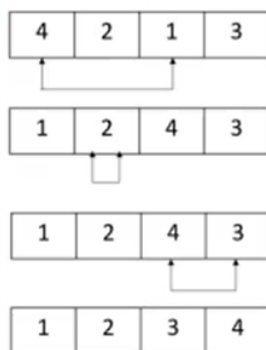
```
1  # バブルソート
2  target_list = [6,3,4,8,7,4,1,9,2]
3
4  # i:0=> target_listの長さを超えない範囲でループ（インデックス）
5  for i in range(len(target_list)):
6      # j:1回のソートで配列の何番目まで行くか
7      # 1回目は右端、ソートする範囲を狭める
8      for j in range(0, len(target_list) - i - 1):
9          if target_list[j] > target_list[j+1]:
10             target_list[j],target_list[j+1] = target_list[j+1],target_list[j]
11
12  print(target_list)
13
14
15
```

⇒ ここで、添付資料「バブルソート」シートにある演習問題を解いてみましょう

### ・選択ソート（単純選択法）

選択ソートは、データの中で最小値（または最大値）を探して端から順に格納することで並び替えを行う整列方法です。次のようなアルゴリズムで未整列の配列  $A[j]$  ( $j=1,2,3 \dots, n$ ) を整列します。

- ・  $A[1] \sim A[n]$  の中から最小の要素を探し、それを  $A[1]$  とする。
- ・  $A[2] \sim A[n]$  の中から最小の要素を探し、それを  $A[2]$  とする。
- ・ 同様に、範囲を狭めながら処理を繰り返す。



## 問題

以下のリストについて選択ソートを用いて、昇順に並び替えてください。

```
target_list = [6,3,4,8,7,4,1,9,2]
```

```
1  #選択ソート
2
3  target_list = [6,3,4,8,7,4,1,9,2]
4
5  # i: 0 => target_listの長さを超えない範囲でループ（インデックス）
6  for i in range(len(target_list)):
7
8      #idx_min :i以上の最小値の入っている配列の添え字（インデックス）
9      idx_min = i
10     # j: i+1 => target_listの長さを超えない範囲でループ
11     for j in range(i+1, len(target_list)):
12         if target_list[idx_min] > target_list[j]:
13             idx_min = j
14
15     #target_listのiとidx_minの箇所を入れ替え
16     target_list[i], target_list[idx_min] = target_list[idx_min], target_list[i]
17
18     print(target_list)
19
20
```

⇒ ここで、添付資料「選択ソート」シートにある演習問題を解いてみましょう

=====

業務上、問題解決の手順としてアルゴリズムを意識する場面がある方にとっては、身近な話題でも、はじめてプログラミングを学ぶ方にとっては、今回のような知識を積み上げられる機会があると取っ掛かりやすく感じるのではないのでしょうか。自身の PC に環境がある方は、簡単に動かすことができますので、この機会にアルゴリズムにも触れてみようかなと思っていただければ幸いです。

以上となります。

参考文献：

・文研出版編集部『教科書ガイド 実教出版版 情報Ⅰ』