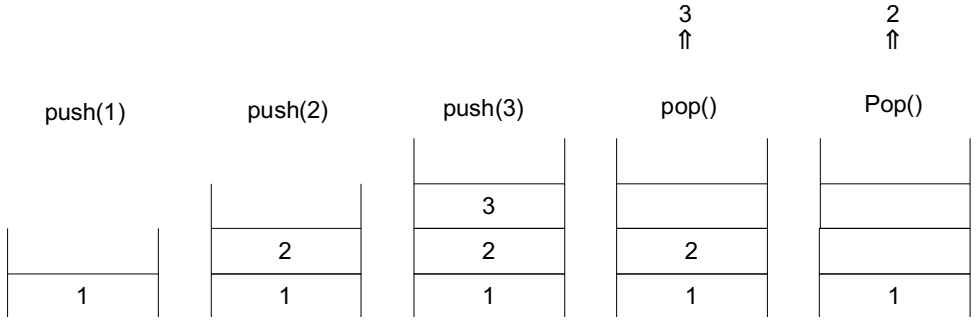
第 13 回　スタックとキューのデータ構造

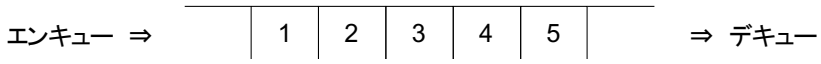
スタックとキューのデータ構造を Python で実現する方法について学ぶ。

【スタック】
スタック(Stack；物を積み上げた山)は、「後入れ先出し（LIFO：Last In, First Out)」型のデータ構造である。つまり、最後に格納したデータが最初に取り出されるデータ構造である。データを格納する操作をプッシュ、データを取り出す操作をポップと言う。スタックの動作イメージを以下に示す。



日常生活の例：食堂の皿の山
プログラムの例：メソッド呼び出し、操作の「元に戻す」「繰り返し」

【キュー】
キュー(Queue；待ち行列)は、「先入れ先出し（FIFO：First In, First Out)」型のデータ構造、すなわち最初に格納したデータから順に取り出されるデータ構造である。データを格納する操作をエンキュー、データを取り出す操作をデキューと言う。キューの動作イメージを以下に示す。



日常生活の例：スーパーのレジ等の待ち行列（先に並んだ客から処理される）
英語では First come, first served (FCFS)
プログラムの例：プリンタの実行待ちジョブのキュー

【両端キュー】
両端キュー（double-ended queue）またはデック（deque）とは、先頭または末尾で要素を追加・削除できるキューである。両端キューを使えば、スタックとキューの操作をいずれも実現できる。Python では、両端キューが collections.deque で実装されている。Python の組み込みコンテナ（list, tuple, dict, set）にはそなわっていないコンテナのデータ型を集めたモジュールに、collections モジュールがある[1]。ドキュメントには次のように書かれている。

> Deque とは、スタックとキューを一般化したものです（この名前は「デック」と発音され、これは「double-ended queue」の省略形です)。Deque はどちらの側

[1] https://docs.python.org/ja/3/library/collections.html

からも append と pop が可能で、スレッドセーフでメモリ効率がよく、どちらの方向からもおよそ O(1) のパフォーマンスで実行できます。
　list オブジェクトでも同様の操作を実現できますが、これは高速な固定長の操作に特化されており、内部のデータ表現形式のサイズと位置を両方変えるような pop(0) や insert(0, v) などの操作ではメモリ移動のために O(n) のコストを必要とします。

たとえば、[1,2,3,4,5]というリストを考える。このリストに 6 というデータを追加すると、[1,2,3,4,5,6]となる。スタックであれば、ここからデータを取り出すときには一番最後に追加した最後のデータ 6 を取り出せば良いが、キューであれば、最初のデータ 1 を取り出すことになる。その場合に、1 を取り出してあいたところに 2 を移動し、さらにあいたところに 3 を移動し、といったデータの移動をするために、時間がかかってしまう。このことが「メモリ移動のために O(n) のコストを必要とします」と表現されている。
Deque のデータ型では、このような問題が解決されて「どちらの方向からもおよそ O(1) のパフォーマンスで実行」できるとされている。つまり、メモリを「詰めていく」という内部処理を必要としないデータ構造となっている。したがって、先頭や途中にデータを追加したり削除したりすることが多い場合（そして大量のデータを取り扱うとき）には、list よりも deque を使う方が良い。
このような計算資源の効率的な使い方は、日常的なプログラミングではほとんど気にする必要がない。なぜならば、コンピュータの処理は高速であり、通常は一瞬で計算が終わるためである。一方、同じような処理を何度も繰り返すような場合や、大量のデータを処理する場合には、処理速度が問題になることがある。たとえば、10 日間かかる計算が 2 秒で終わるのであれば、そのメリットは大きい。アルゴリズムやデータ構造を効率化することで、このようなプログラムの高速化に役立つことがある。
Deque に用意されているメソッドの中から、いくつかピックアップする。

append(x)	x を deque の右側に付け加える。
appendleft(x)	x を deque の左側に付け加える。
index(x)	deque 内の x の位置を返す。
insert(i, x)	x を deque の位置 i に挿入する。
pop()	deque の右側から要素をひとつ削除し、その要素を返す。
popleft()	deque の左側から要素をひとつ削除し、その要素を返す。
remove(value)	value の最初に現れるものを削除する。

★deque を使って次のプログラムを動かしてみよう。

```
import collections
queue = collections.deque(["a", "b", "c"]) # a, b, c という並びのキューを作成
queue.append("d") # d を右側に付け加える
queue.insert(1,"e") # 1番目にeを挿入。「1番目」はどこになるか？
print(queue.pop()) # 右側から削除して、その要素を表示。
print(queue.popleft()) # 左側から削除して、その要素を表示。
print(len(queue)) # キューの長さを表示
queue.remove("b") # b を削除
print(list(queue)) # キューの状態を表示
```

次のように出力される。

```
d
a
3
['e', 'c']
```

【★課題】
　上のプログラムを参考に、次の動作をするプログラムを作成して、ToyoNet-ACE に提出してください。途中の経過がないプログラムは無効です。

(1) a, b, c, d, e という並びのキューを作成
(2) f を右側に付け加える
(3) c の右側に g を付け加える
(4) キューの一番左から削除して、その要素を表示する
(5) キューの状態を表示

出力：

```
a
['b', 'c', 'g', 'd', 'e', 'f']
```

【発展：競技プログラミング】
　これまでの授業で、だいぶプログラミングに慣れてきたでしょうか。自ら、さらに深く勉強しようという人も出てきていることでしょう。そこで「競技プログラミング」について紹介する。競技プログラミングでは、参加者全員に同じ課題が出題され、与えられた課題を正確にかつより早く解決するプログラムを書くことを競う。国内では、AtCoder という競技プログラミング(https://atcoder.jp/)が有名である。コンテストで良い成績を取るとレーティング、ランキングが上がる。
　参考までに、AtCoder Beginner Contest 149 (2019 年 12 月 29)の問題 C (Next Prime)を示す[1]。これまでの授業の知識で十分に解ける問題なので、ぜひ挑戦してみてください。AtCoder のサイトからコンテストで提出された様々なプログラミング言語の様々な回答を見ることができる[2]。

問題文: X 以上の素数のうち、最小のものを求めよ。
注記: 素数とは、2 以上の整数であって、1 と自分自身を除くどの正の整数でも割り切れないようなもののことです。例えば、2,3,5 は素数ですが、4,6 は素数ではありません。
制約: $2 \leq X \leq 10^5$、入力はすべて整数
入力: 入力は以下の形式で標準入力から与えられる。

```
X
```

出力: X 以上の素数のうち、最小のものを出力せよ。

入力例 1: 20
出力例 1: 23
入力例 2: 2
出力例 2: 2
入力例 3: 99992
出力例 3: 100003

[1] https://atcoder.jp/contests/abc149/tasks/abc149_c
[2] 私が作成した回答例は https://paiza.io/projects/YGWJ685Cd2p6s0qEeuvfzg

【発展：ウェブアプリ開発】

ウェブでアプリを実行するための技術には様々なものがある。大きく分けると、サーバー側（サーバーサイド）で動かすプログラムと、ウェブブラウザ（クライアントサイド）で動かすプログラムがある。

Python でサーバーサイドのウェブアプリを開発するときには、Django（ジャンゴ）、Flask（フラスク）などのフレームワークを導入するのが一般的である。Python のフレームワークには、ログイン機能などのウェブアプリで標準的に使われる機能が備わっているため、効率的にアプリを開発することができる。従来は、ウェブアプリケーションを実行するたびに呼び出す CGI がよく使われていたが、CGI は実行するたびにプロセスを起動するため、最近ではサーバ内のプロセスでウェブアプリケーションを実行する方法が主流である。サーバーの開発環境は Docker のコンテナによって構築し、複数のコンテナを Kubernetes によるコンテナオーケストレーションによって運用管理するという手法が標準となっている。私自身は、研究用のソフトウェア SWRC Fit[1]と EC Fit を Python の CGI プログラムとして公開している。

サーバーにデータを保存する必要がなければ、サーバーの環境を選ばないクライアントサイドプログラミングが手軽である。クライアントサイドプログラミングでは JavaScript が使われ、そこに WebAssembly が組み合わされることがある。

JavaScript は言語仕様が ECMAScript として標準化され、ほぼすべてのメジャーなウェブブラウザでサポートされている。サーバーサイドプログラミングでも、フォームに入力するときの入力チェックなど必要に応じて JavaScript が併用されるため、ウェブアプリを開発するのであれば JavaScript は HTML, CSS と並んで必修科目である。JavaScript はウェブブラウザで直接開発できる点も手軽である。たとえば Chrome ではブラウザを右クリックして「検証」を選ぶことで Console が表示され、直接 JavaScript を実行することができる。

WebAssembly は C, Rust 言語などの様々なプログラミング言語がコンパイルしてウェブブラウザの仮想マシンで実行できるコードを生成するように設計されたもので、主要ブラウザでサポートされている[2]。WebAssembly は JavaScript の代替というよりは、JavaScript と組み合わせて使われるものである。Python の WebAssembly への移植である Pyodide[3]を使うことによって、JavaScript プログラムの中で Python を実行することができる[4]。なお、Node.js によって JavaScript をサーバーサイドで実行することも可能である。

私が JavaScript で作成したパズルを 2 つ紹介する。

(1) 15 パズル[5]

ソースコード[6]を GitHub で読むことができる。

(2) ナンプレ[7][8]

Python でナンプレを解き、問題を作成するプログラムを作成し、そのプログラムで自動作成した問題を難易度別問題集として JavaScript プログラムで出題している。ヒント表示機能では、JavaScript から Pyodide によって Python のプログラムを呼び出している。なお、スマホアプリは Flutter により開発した。

---

1 https://seki.webmasters.gr.jp/swrc/?lang=ja

2 https://webassembly.org/roadmap/

3 https://pyodide.org/en/stable/

4 https://sekika.github.io/2022/08/18/Pyodide/

5 https://sekika.github.io/2020/01/17/15Puzzle/

6 https://github.com/sekika/sekika.github.io/blob/master/js/15.js

7 https://sekika.github.io/kaidoku/ja/sudoku

8 ナンプレは世界的には sudoku として有名であるが、「数独」という名称はニコリが商標登録しているため、日本国内ではニコリが関与しないものはナンプレと表記される。