```
a.up()
a.up()
a.right()
a.down()
print(a.pos())
```

出力：(3, 4)

ターミナルで実行するときには、Main.py と point.py を同じディレクトリに置く。.py の拡張子のファイルはモジュールとして認識されて、Main.py の import point で point.py がモジュールとして読み込まれる。a = point.Point(2,3)では、後で説明されるように point モジュールの Point クラスが読み込まれる。

point.py すなわち point モジュールの Point クラスは、二次元座標平面上の点(x,y)をあらわすクラスである。class Point: で、Point という名称のクラスを定義することを宣言する。この文をヘッダとして、インデントによってクラスの中身が書かれている。そして、クラスの中には def ではじまるメソッドの定義が 6 個ほど書かれている。

最初のメソッドは def __init__(self, x, y): である。__init__ という名称のメソッドは、クラスからオブジェクトが生成されるときに最初に実行される。このメソッドには 3 つの引数があり、その 1 つ目は self である。self は「自分自身」つまり「そのオブジェクト自身」である。クラスのメソッドには必ず最初の引数に self を入れる。そして、(self, x, y) のように、self の次に x と y という引数があり、これはオブジェクトを生成するときにユーザーが明示的に指定をする。Main.py で

```
a = point.Point(2,3)
```

が実行されると、point モジュールの Point クラスから (2,3) を引数としてオブジェクトが生成され、この __init__(self, x, y) が呼び出されることにより、x=2、y=3 として処理が実行される。そして、self.x = x では、その自分自身 (self) の x という属性 (インスタンス変数) が x、つまり 2 であると設定される。この self.x というのは、たった今作成しているオブジェクト、つまり Main.py では a という変数の属性なので、Main.py からは a.x として呼び出すことができる。次に、self.y = y によって、y=2 として得られた引数が、self.y に代入される。このようにして、a という変数の座標平面上の位置を表現している。

次に、def up(self): では、up という名前のメソッドが定義されている。このメソッドは、座標平面上を上に 1 つ進む、つまり y 座標の値を 1 増やす、という操作をするメソッドである。y 座標の値は self.y に格納されているため、その値に 1 を足すことで実現している。同様に、down という名前のメソッドは下に進む、つまり y 座標の値を 1 減らしている。また、left という名前のメソッドは左に進む、つまり x 座標の値を 1 減らし、right という名前のメソッドは右に進む、つまり x 座標の値を 1 増やしている。

```
a.up()
a.up()
a.right()
a.down()
```

の箇所では、先ほど(2,3)の位置に設定された a という変数が、上に進んで、上に進んで、右に進んで、下に進んでいる。そして、最後の

```
print(a.pos())
```

では、a.pos() の値を表示している。ここでは、a は Point クラスのインスタンスなので、def pos(self): で定義されているメソッドが呼ばれて、x 座標と y 座標の組（タプル）が返され、それが表示される。

【★課題】
上のプログラムの Main.py を修正して、

(1) 座標（7,5）に設定して
(2) 左に移動して
(3) 上に移動して
(4) 左に移動して
(5) 上に移動して
(6) 下に移動して
(7) 座標を表示する

という動きをするプログラムに書き換えて、ToyoNet-ACE の「レポート」から Main.py **だけ**を提出してください。point.py は編集も提出もしないこと。

【発展：スタイルガイドの命名規約】
Python のコードを読みやすくするためのスタイルガイドとして PEP8 がある[1]。PEP は Python Enhancement Proposal の略で、Python の機能拡張のためにコミュニティが話し合ってとりまとめた文書がまとめられたもので、その 8 番目である。そこには、たとえばインデントのそろえ方、1 行の長さなどが書かれている。命名規約として書かれているものとして、いくつか抜粋する。

(1) 小文字のエル、大文字のオー、大文字のアイなどを単一の文字で変数名にしない。
(2) ASCII の文字を使う。
(3) モジュールの名前は小文字のみの短い名前にする。
(4) クラスの名前は単語の初めの文字だけを大文字にする CapWords 方式を使う。
(5) 関数や変数の名前は小文字のみにして、必要に応じて単語をアンダースコアで区切る。
　メソッドとインスタンス変数も同様にする。

今回の課題では、モジュールの名前を point クラスの名前を Point としたのは、この命名規約に従った。

スタイル以前の問題として重要なことは、キーワード（予約語）は関数や変数などの名前として使えない、ということである。キーワードとは、たとえば if, True, return のように Python が特別な意味を持つ単語として解釈するものであり、キーワードの一覧を次のコマンドで確認できる。

```
import keyword
print(keyword.kwlist)
```

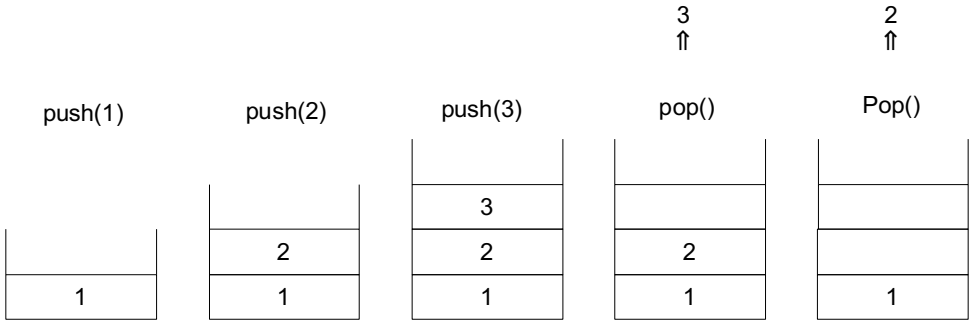
[1] PEP8 の原文 https://www.python.org/dev/peps/pep-0008/ と和訳 https://pep8-ja.readthedocs.io/ja/latest/

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】

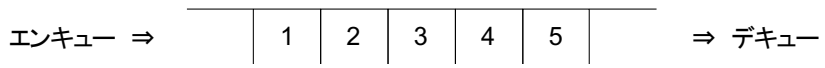
スタック(Stack；物を積み上げた山)は、「後入れ先出し（LIFO: Last In, First Out)」型のデータ構造である。つまり、最後に格納したデータが最初に取り出されるデータ構造である。データを格納する操作をプッシュ、データを取り出す操作をポップと言う。スタックの動作イメージを以下に示す。



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