第 12 回　オブジェクト指向プログラミング

オブジェクト指向プログラミングの考え方について学ぶ。

【オブジェクト指向プログラミング】

Python はオブジェクト指向プログラミングができる。オブジェクト指向プログラミングとは、データとメソッドをひとつにまとめてオブジェクトとして、オブジェクトを中心にプログラミングをすることである。オブジェクト指向プログラミングに特有の用語について、簡単に解説する。

オブジェクトの種類を「クラス」と呼び、あるクラスに基づいて生成された具体的なオブジェクト（物）を「インスタンス」と呼ぶ。たとえば、「ビールとはこういうものである」ということが書かれている酒税法は「ビールクラス」を定めていて、実際に店で売られている１つ１つの缶ビールは、「ビールクラス」に基づいて生成された「ビールインスタンス」（ビールオブジェクト）である、と考えることができる。

Python では、組み込みのクラスとしてたとえば「int クラス」「str クラス」のようなものが定められている。さらに、そのクラスが持つ状態（変数）や、そのクラスがどういう動作をするか（メソッド）が定義されている。そして、あるクラスに基づいて生成されたオブジェクト（インスタンス）は、呼び出されたクラスで定義されているメソッドを使って、操作することができる。

たとえば s = “Hello” という文を実行することで、s という変数に「Hello という文字列が代入される」とこれまで説明をしてきたが、これは str クラスを元に Hello という具体的な文字列の「オブジェクト（インスタンス）」が生成された、ということになる。この Hello という文字列は、str クラスを元に生成されたオブジェクトであるため、str クラスで定義されているメソッドを使うことができる。たとえば、str クラスには replace というメソッドが定義されているので、Hello という文字列に replace というメソッドを適用することができる。なお、replace というメソッドは文字列の置換をするメソッドであり、たとえば次のプログラムの出力は

```
s = "Hello"
print(s.replace("H", "h"))
```

出力：hello

となる。

これまでに使ったメソッドには、str クラスに定義されている split というメソッドがある。これは、文字列をスペースやタブなどの「ホワイトスペース」で区切ったリストを生成するものであり、オプションによって区切り文字を指定することもできるが、詳しくは触れない。これまでのプログラムで使われていた

```
list = list(map(int,input().split()))
```

という文は、input().split() のところで、組み込み関数の input() で得られた文字列が str クラスのオブジェクトとして生成されて、そのオブジェクトに対して、split()というメソッドが適用されている。

このように、str クラスを定義して、そのクラスに「メソッド」を定義することで、文字列に対して replace や split のような「共通する操作」を定義することができる、というところがオブジェクト指向プログラミングの特徴である。

Python で変数に代入される値は、基本的にはどのような値も「オブジェクト」であり、つまりなんらかの「クラス」を元に生成された「インスタンス」である。そして、ある変数がどのようなクラスから生成されたインスタンスであるかは、type を使って確認することができる。

★次のプログラムを実行してみましょう。

```
s = "Hello"
print(type(s))
i = 1
print(type(i))
n = 1.5
print(type(n))
list = [1,2,3]
print(type(list))
b = True
print(type(b))
```

【クラスの作成】

Python では、「数」「文字列」「リスト」といった組み込みのクラスだけではなく、クラスを自作することができる。様々な種類のデータをオブジェクトとしてとらえて、そのオブジェクトの性質を「クラス」として抽象化し、そのオブジェクトにはどのような属性があるか、そしてそのオブジェクトに対してはどのような操作ができるか、ということをプログラミングしていくことが、オブジェクト指向プログラミングの基本的な考え方である。

★次のプログラムを実行してみましょう。ファイルは 2 つあります。コードは
https://paiza.io/projects/3a6G1hHwwmP-ms2k6Prsog

point.py

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def up(self):
        self.y += 1
    def down(self):
        self.y -= 1
    def right(self):
        self.x += 1
    def left(self):
        self.x -= 1
    def pos(self):
        return self.x, self.y
```

Main.py

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

```
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/