第 7 回　コンテナのデータ構造

データの集まりをまとめて扱うコンテナのデータ構造について学ぶ。

【コンテナのデータ構造】
複数のオブジェクト（データ）をひとまとめに格納するデータ構造をコンテナと呼ぶ。複数のデータの集まりであることからコレクションとも呼ばれる。いずれも様々なデータ構造を含む包括的な名称である。Python では、次のような組み込みのコンテナのデータ型が用意されている。

型の名前	表記例	主な特徴
リスト型 list	[1, 2, 3]	順番に並んでいる。要素を変更できる。
タプル型 tuple	(1, 2, 3)	順番に並んでいる。要素を変更できない。
集合型 set	{1, 2, 3}	順序は関係ない。要素は重複しない。
辞書型 dict	{"one":1,"two":2}	順序は関係ない。キーで要素を呼び出す。

array モジュールや numpy モジュールを読み込むことで配列(array)が使われることもある。一般に、配列ではすべての要素のデータ型が同じである必要があるが、リストやタプルの要素は同じデータ型である必要はない。リストの要素がコンテナであっても良く、リストの中にリストが入ったものを多重リスト（多次元リスト）と呼ぶ。
これまでにも扱ってきた文字列型 str は、文字が順番に並んだデータ型であり、文字列はリストと同じように扱うことができる（ただし、要素の変更はできないのでタプルに近い）。リスト型とタプル型、文字列型、そして前回の授業で扱った range 関数が生成する range 型のようにデータが一列に順番に並んだデータ型を「シーケンス型」と呼ぶ。

【リスト】
この授業では、シーケンス型の中でもリストを中心に扱う。リスト型のデータは、複数のデータを順番に並べたものである。たとえば、5, 2, 9, 7, 1 という 5 個の整数を順番に並べたリストは、[5, 2, 9, 7, 1]のように表記する。リストの要素は同じデータ型である必要はなく、たとえば、[1, 2, “りんご”]のように、整数型と文字列型が混在するリストを作ることも可能である。さらには、[1, 2, [1, 3, 4], (1, 2, 3)]のように、リストの中にリストやタプルを入れることも可能である。
リストに含まれる個々のデータをリストの「要素」、リストに含まれる要素の個数をリストの「長さ」と言う。
リストは「順番に」並べられているので、リストの要素は「何番目の要素か」ということによって指定することができる。これをリストの「インデックス」と言う。インデックスは、リストの何番目にデータが格納されているかを表す数値で、1 からではなく 0 から数える。
リストの要素を参照するには、リストの名前[インデックス] と指定する。list という名称のリストがあった場合、この 3 番目(1 から数えると 4 番目)の要素を参照するには list[3] と指定する。たとえば、

```
list = [5,2,9,7,1]
print(list[3])
```

というプログラムを実行すると、「7」と出力される。「9」ではないことに注意。

インデックスには変数を用いることができる。list の i 番目の要素を参照するには list[i]と指定する。この場合には、i=1 であれば list[1]が、i=3 であれば list[3]が参照される。

★次のプログラムを実行してみましょう。

```
list = [5,2,9,7,1]
print("リストの長さは{0}です。".format(len(list)))
for i in range(len(list)):
    print("{0}番目の要素は{1}です。".format(i,list[i]))
```

これまでの授業を思い出しながら、このプログラムを一行ずつ読んで、どのようにプログラムが実行されているか考えてみましょう。for 文による繰り返し処理で、添字 i を用いてリストの要素を参照していることを確認して下さい。

リストは要素を変更することができるため、たとえば list[2] = 3 とすれば、2 番目の要素（0 から数えるので実際には 3 番目の要素）が 3 になる。

★上のプログラムで、1 行目を「list = "python"」のように好きな文字列に変えて実行してみてください。「文字列はリストと同じように扱える」ということの意味が分かります。

★リストや文字列を for 文の「イテラブルオブジェクト」として使うことができます。次のプログラムを実行してみましょう。

```
for c in "月火水木金土日":
    print(c + "曜日")
```

【リストの要素の最大値を求める】

リストの要素の中から最大値を求めるプログラムを考える（最大値を計算するメソッドを使えば良いが、ここでは自分でそのようなプログラムを作る方法について考える）。第 5 回目の授業で作成したプログラムでは、３つの変数 a,b,c の大きさを順番に比べていた。変数の数が多くなると、大小の比較を１つ１つプログラムに書いていくのは大変である。たとえば、a,b,c,d,e の最大値を比べるプログラムを書くと、どうなるだろうか。

そこで、リストを使うと、プログラムをより効率的に書くことができる。また、リストの要素数が決まっていない場合にも、プログラムを発展させることが可能となる。

★次の最大値を求めるプログラムを実行してみましょう。

```
list = list(map(int,input().split()))
max = list[0]
for i in range(len(list)):
    if list[i] > max:
        max = list[i]
print(max)
```

入力：5 2 9 7 1
出力：9

1 行目では、5 2 9 7 1 のようにスペースで整数を区切って入力されたものが、整数のリスト[5, 2, 9, 7, 1]に変換されて、リスト型変数 list に代入される。その仕組みは若干複雑であるが、以下に説明する。このような処理を簡潔に書けるところは Python の魅力であるが、ここは理解できなくても当面は問題ない。入力として「5 2 9 7 1」のような文字列が与えられたときに、これを input()で受け取って split()でスペースの区切りごとに分けられたリストにする（['5', '2', '9', '7', '1']のようなリストとなる）。そして、map(int,input().split()) によって要素がそれぞれ文字列から整数型に変換される。ここまではこれまでと同じ処理で、これまでは a, b = map(int,input().split()) のような形で変換された整数をそれぞれ順番に変数に格納していたが、ここでは変換されたデータをリスト型の変数として list にそのまま格納するために、map オブジェクトをリスト型に変換するための list 関数を使っている。

【リストの要素を逆順に並べ替える】

リストの要素を逆順に並べ替えるプログラムを考える。長さが 5 のリストの場合、

0 番目の要素と 4 番目の要素を交換する
1 番目の要素と 3 番目の要素を交換する

という処理を実行すれば、配列の要素を逆順に並べ替えられる。

ここで、0 番目の要素と 4 番目の要素を交換する場合に、

```
list[0] = list[4]
list[4] = list[0]
```

とすると、どのような問題が起きるだろうか？

上記の方法では問題が起きるので、交換対象のデータを一時的に保持する変数 tmp を用意し、

```
tmp = list[0];
list[0] = list[4]
list[4] = tmp
```

とするのが、一般的なアルゴリズムである。Python では、このような一時的な変数を使わずに

```
list[4], list[0] = list[4], list[0]
```

とまとめて書くことができる。

【★課題】

次のプログラムは、リストを逆順に書き換えるプログラムを作ろうとして「失敗した」ものです。まずはこの通りに実行して、どのような結果になるのかを確認してください。その後、このプログラムの赤字の箇所を書き換えて、正しく逆順になるようにしてから、ToyoNet-ACE から提出して下さい。赤字の箇所以外は書き換えないこと。

```
list = list(map(int,input().split()))
for i in range(len(list)):
    j = len(list) - 1 - i
    list[i], list[j] = list[j], list[i]
print(list)
```

【課題のヒント】

上記のプログラムは、リストの長さが 5 のときには、次の処理を実行している。

0 番目の要素と 4 番目の要素を交換する
1 番目の要素と 3 番目の要素を交換する

2 番目の要素と 2 番目の要素を交換する
3 番目の要素と 1 番目の要素を交換する
4 番目の要素と 0 番目の要素を交換する

すると、結局元に戻ってしまう。この交換を、2 番目あるいは 3 番目までで「止めて」しまえば、うまくいくはずである。そのためには、赤字の range の中を変えてどこまで繰り返しを進めるかを変えれば良い。len(list)が 5 のときには 2 あるいは 3 となり、len(list)が 10 であれば 5 となるような計算にする。どのような計算（演算）を使えば良いかは、第 4 回の授業の「基本演算」を読み、+, -, *, // の中から選んで使うこと。ここで、range 関数は整数型のみ許容されるので、float 型が返る / を使うとエラーになることに注意すること。

正しく書き換えると、次のような計算結果となる。

入力：5 2 9 7 1
出力：[1, 7, 9, 2, 5]

入力：1 2 3 4 5 6 7 8
出力：[8, 7, 6, 5, 4, 3, 2, 1]

【発展：リスト内包表記】
新しいリストを作るときに「リスト内包表記」を使うと便利である。次のプログラムを実行してみよう。

```
print([10**x for x in range(10)])
```

map 関数と無名関数（ラムダ式）を使って次のように書いても同じである。

```
print(list(map(lambda x: 10**x,range(10))))
```

【発展：Python でデータ解析】
Python でデータ解析をするときによく使われるライブラリが Pandas と Matplotlib である。Pandas は大きな表データを扱うときに便利で、Excel のような表計算ソフトのかわりとして使うことができる。特に、様々な形式のデータを入力として受け取り、細かいデータの加工をして統計的に集計するときには、Pandas が便利である。また、Pandas でデータの基本処理をしてから、数値計算ライブラリの NumPy で線形代数などのさらに高度な数値計算をすることもある。
Matplotlib は、グラフを表示するためのライブラリである。Pandas は Matplotlib と組み合わせて使われることが多く、Pandas でデータを読み込んで計算をして、それを Matplotlib で表示する、というところまでを自動化することで、定型的なデータ集計作業をプログラミングすることが可能である。グラフの表示は様々なタイプがあり、細かく制御できる。Matplotlib のページのギャラリー(https://matplotlib.org/gallery/)に、様々なグラフとそれをどのように描画するかの例が示されている。