# リストと関数

## 配列

　BasicやCでは配列は要素数を決めて宣言し、要素数を後から変更することはできませんでした。それに対してpythonの配列（リスト）は要素数が固定されていません。以下のように使います。

list = ["a", "b", "c"]

print(list) # 配列全体を表示

print(list[0], list[1], list[2]) # 個々の要素を表示

print(len(list)) # 配列の要素数

list2 = []

list2.append(10) # リストの末尾に追加

print(list2)

list2.append(20)

print(list2)

list2.append(30)

print(list2)

print(list2[0], list2[1], list2[2])

　配列を使うときは

list = [要素1, 要素2, 要素3]

のように書きます。この場合listという配列変数は3個の要素を持ちます。最初に空のリストを作成したいときは

list = []

と書きます。配列の末尾に要素を追加する場合、

list.append(要素)

と書きます。配列の要素数は

n = len(list)

で得られます。配列の内容を取り出すには

element = list[2]

のように書きます。[ ] の中の添字は1番目の要素が0です。n番目の要素がn-1です。n個の要素を持つリストのとき、添字の範囲は0 ～ n-1です。

　listという配列変数の3番目の要素を削除し、削除した要素をelementに代入するには

element = list.pop(2)

と書きます。リストの連結は以下のように書きます。

list1 = ["a", "b", 10] # 文字列と数値を混在させてもよい

list2 = ["c", 20, "d"]

list = list1 + list2

print(list) # ["a", "b", 10, "c", 20, "d"] となる

　配列から要素を1個ずつ取り出すには以下のように書きます。

list = ["a", "b", "c"]

for element in list:

 print(element)

　以前学習したfor文はfor i in range(0, 4) のように指定しました。これはfor i in [0, 1, 2, 3] と書くのと同じです。リスト変数から1個ずつ取り出してループを回ります。

　以下のようにBasic風に書くこともできます。

list = ["a", "b", "c"]

n = len(list)

for i in range(0, n):

 print(i, list[i])

　このプログラムはBasicでのプログラミング経験があり、pythonに慣れてない人には分かりやすいかも知れませんが、pythonに慣れた人にとっては冗長な感じがすると思います。お勧めしません。

### 【例題】

　次のリストの平均、最大、最小を求めて下さい。値の範囲は0～100です。

list = [10, 53, 29, 83, 90, 40, 73]

### 【解答】

sum = 0

max = -999

min = 999

for value in list:

 sum = sum + value

 if value > max:

 max = value

 if value < min:

 min = value

average = sum / len(list)

print("average =", average)

print("max =", max)

print("min =", min)

　上の例ではmaxとminを愚直に求めましたが、以下のようにすると簡単に求まります。

max = max(list)

min = min(list)

## アンパック代入

　pythonでは以下のように代入することができます。

a = 10

b = "abc"

c, d = a, b

print(c, d)

　イコールの右辺にある値が左辺の各変数に順番に割り当てられます。ここではcに「aの値」が代入され、dに「bの値」が代入されます。以下のようにリストの値を割り当てることもできます。

list = ["aaa", "bbb"]

a, b = list

print(a, b)

## 関数

　pythonにおける関数はExcel VBAのプロシージャ（Sub）と関数（Function）のどちらとも少し異なります。まずは次の例題を見てください。

def func1(c, d):

 print(c, d)

 c = 30

 d = 40

 print(c, d)

a = 10

b = 20

func1(a, b)

print(a, b)

　defで定義しているのがExcel VBAのSubやFunctionに対応する機能です。この場合、func1という名前の関数を定義しています。defの行の行末には : を付けます。インデントの範囲が関数の範囲です。この場合は、defから2回目のprint(c, d) までが関数func1の範囲です。

　pythonでは「関数」は「関数を呼び出す場所（ここではfunc1(a, b)）より手前にないといけません。また、関数の定義は関数を呼び出す場所より手前であれば、どこに書いてもかまいません。この場合「def ～ 2回目のprint(c, d)」の部分をたとえば「b = 20 と func1(a, b) の間」に書いても問題なく動作します。

　関数の引数は順番に結合されるので、呼び出す側の変数aとbが関数func1において、cとdとして扱われます。ですから、1回目のprint(c, d) の実行時においてc = 10, b = 20です。2回目のprint(c, d) の実行時においてはc = 30, d = 40に更新されています。関数の実行を終了して、呼び出し元に戻ってきてからprint(a, b)を実行しています。このときa = 10, b = 20です。

　Excel VBAとは異なりpythonでは関数側で引数の値を書き換えても、その変更が呼び出し元に影響しません。

　関数側から呼び出し元へ情報を渡すにはreturn文を使います。以下の例を見てください。

def func1(c, d):

 print(c, d)

 sum = c + d # 和

 sub = c - d # 差

 return sum, sub

a = 10

b = 20

e, f = func1(a, b) # 2個の数値をe, fに受け取る

print(e, f)

　returnで2個の数値sumとsubを返し、それをeとfで受け取っています。複数の値を代入するとき、並び順に結合されます。sumの値がeに入り、subの値がfに入ります。

【例題】

　次のinput\_listを入力とするとき、output\_listとして0以上の数だけを取り出して、新しいリストとして返す関数を作りなさい。

input\_list = [10, -20, 3, 0, -5, 50]

output\_list = [10, 3, 0, 50]

【解答】

def delete\_minus\_value(in\_list):

 out\_list = []

 for element in in\_list:

 if element >= 0:

 out\_list.append(element)

 return out\_list

input\_list = [10, -20, 3, 0, -5, 50]

output\_list = delete\_minus\_value(input\_list)

print(output\_list)