

「配列」(以下、「リスト」と呼ぶ)とは？

コンピュータを使って大量のデータを使いたいとき、これまでのように変数を1つずつ用意しているのは、非効率的です。

そこで、「配列」と呼ばれるデータ構造を使うことで、大量のデータを容易に扱えるようにします。

Python では、「リスト」というデータ型で配列に相当する機能を利用できます。

(例1) 次のプログラムを入力し、実行してみよう

```
1 a = [1,2,3,4,5,6,7,8,9,10]
2 print(a)
3 print(type(a))
```

Python のリストに入っているデータにアクセスするためには、「変数[添字番号]」で指定します。

(例2) 次のプログラムを入力し、実行してみよう

```
1 a = [1,2,3,4,5,6,7,8,9,10]
2 print(a[7])
3 print(type(a[7]))
```

Python のリストは型の違うデータを入れることもできます。

(例3) 次のプログラムを入力し、実行してみよう

```
1 lst = [1, "name", 3.14, True]
2 print(lst)
3 print(type(lst))
```

リストの初期化として、(例1)や(例2)のようにあらかじめデータを入れることもできますが、まだどんなデータをいれるかわからないが、変数をリストとして定義する場合は次のように何も書かない宣言をします。

(例4) 次のプログラムを入力し、実行してみよう

```
1 lst = []
2 print(lst)
3 print(type(lst))
```

※type で型がリストになっていることが確認できます

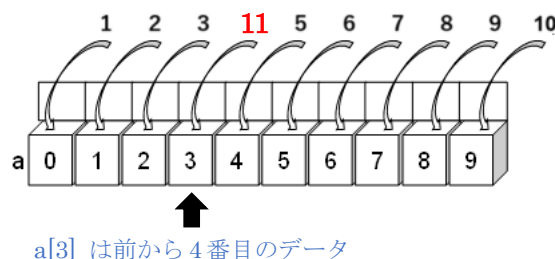
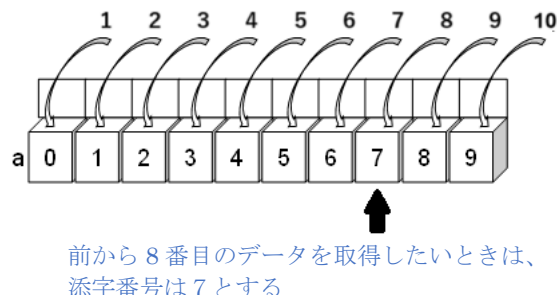
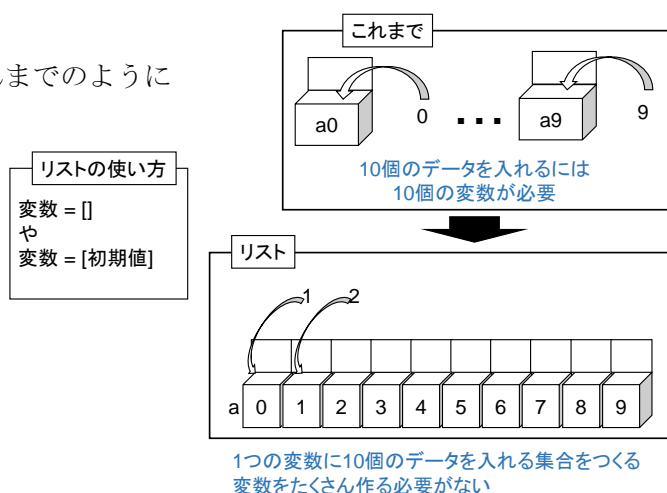
(問1) (例1)～(例4)を参考に、10～19の数字が大きい順に格納されているリストを作成し、4番目のデータを、print()を使って表示させてみよう。

リストの操作 (要素内のデータ変更)

リストの中のデータを変更するためには、変更したい箇所を「変数[添字番号]」で指定し、変更後の値を代入する

(例5) 次のプログラムを入力し、実行してみよう

```
1 a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
2 print(a)
3 a[3] = 11
4 print(a)
```



リストの操作（データの交換）

リストの要素同士を交換するためには、交換したリストの添え字をそれぞれ指定します。

交換方法は、変数の入れ替えと同じです。（プリント No.2 参照）

（問 2） 次のリストの 2 番目と 5 番目のデータを入れ替えたプログラムを作ってみよう。

```
lst = ["a", "b", "c", "d", "e", "f"]
```

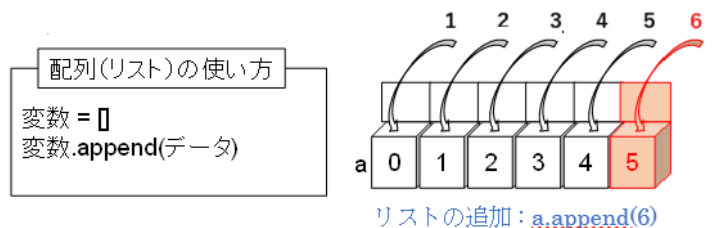
リストの操作（新しいデータの追加）

リストにデータの追加をするときや空っぽのリストにデータを入れるときには、「append」を使います。

append を使った場合、データはリストの最後尾に追加されます。

（例 6） 次のプログラムを入力し、実行してみよう

```
1 a = [1, 2, 3, 4, 5]
2 print(a)
3 a.append(6)
4 print(a)
```

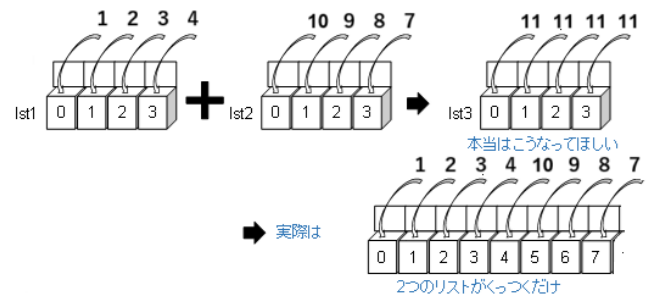


リストの演算（リスト内の要素同士の計算）

リストの演算について考えます。まずは加算処理ですが、リスト同士の加算処理をしてみてください。

（例 7） 次のプログラムを入力し、実行してみよう

```
1 lst1 = [1, 2, 3, 4]
2 lst2 = [10, 9, 8, 7]
3 lst3 = lst1 + lst2
4 print(lst3)
```



結果としては、2 つのリストが結合した状態になりました。

本来は、2 つのそれぞれの場所の加算がしたいわけです。リストの要素同士を計算したい場合は、添え字を指定して、1 つずつ計算します。この時、結果を入れる lst3 のリストは空っぽですので、（例 8）のように、append を使って計算した結果を入れます。

（例 8） 次のプログラムを入力し、実行してみよう

```
1 lst1 = [1, 2, 3, 4]
2 lst2 = [10, 9, 8, 7]
3 lst3 = []
4 lst3.append(lst1[0]+lst2[0])
5 lst3.append(lst1[1]+lst2[1])
6 lst3.append(lst1[2]+lst2[2])
7 lst3.append(lst1[3]+lst2[3])
8 print(lst3)
```

☆ 別解 (lst3 の全要素を暫定的に 0 とする)

```
1 lst1 = [1, 2, 3, 4]
2 lst2 = [10, 9, 8, 7]
3 lst3 = [0, 0, 0, 0]
4 lst3[0] = lst1[0] + lst2[0]
5 lst3[1] = lst1[1] + lst2[1]
6 lst3[2] = lst1[2] + lst2[2]
7 lst3[3] = lst1[3] + lst2[3]
8 print(lst3)
```

発展 （例 8）を参考にして、次に示す lst1 と list2 の各添え字どうしの減算、積算、除算、剰余を求めてみましょう。

```
lst1 = [10, 5, 3, 8]    lst2 = [5, 2, 3, 4]
```