



リスト

初めてのプログラミング

2020年度

只木進一（理工学部）

データ構造とは Data Structure

- ▶ データの集まりに構造を与えたもの
 - ▶ 効率的にデータを扱うために導入
- ▶ 例えば、1000個の整数の和を考える
 - ▶ 1000個にすべて変数名を付け、加算？
 - ▶ 1000行以上のプログラム？
 - ▶ 一つにまとめて、番号で区別する？

リストとは

	0	1	2	3	4	5	6	7	8
d	60	78	95	78	85	98	100	60	70

一次元（鎖状）のデータの列

$$D = [d_0, d_1, d_2, d_3, d_4, d_5, d_6, d_7, d_8]$$

- 一つの名前を付けて、順番で管理する
- 0番から始まることに注意する

リスト(list)

- ▶ 複数の値を一次元的にまとめたもの
 - ▶ 各要素には番号（インデクス）が付く
 - ▶ 0番から始まる
- ▶ 要素は同じ型が基本だが
 - ▶ pythonでは、別の型を混ぜることも可能
- ▶ 高次元のリストも使える

サンプルプログラムの取得

- ▶ GitHubRepositoryを指定
 - ▶ <https://github.com/first-programming-saga/dataStructure>

リストを作る

➡ 要素を指定

```
numbers = [10,20,0,10,5,7,-7]
```

```
colors = ['red','blue','green']
```

➡ 同じ要素を複数含むものを生成

```
zeros = [0]*5
```

```
xyz=['x','y','z']*3
```

```
dataStructure/listSamples.ipynb
```

- ▶ list()関数を使って
 - ▶ 引数には、順序のあるデータ構造 (iterable)を指定する

```
evens = list(range(0,10,2))  
chars = list("saga")  
nullList=list()
```

条件付き文のリスト内容表記

- 条件文を書いて、リストを定義する

```
negatives = [x for x in numbers if x<0]  
positives = list(x for x in numbers if x>0)
```


要素の参照

- ➡ 文字列と同様に番号で要素を指定
 - ➡ 先頭から0, 1, 2...
 - ➡ 終端から-1, -2, -3 ...

```
chars = list('saga')  
print(chars[0])  
print(chars[1])  
print(chars[-1])  
print(chars[-2])
```

```
numbers = [10,20,0,-10,5,7,-7]  
numbers[0]=15
```

要素の参照：リストを順に辿る

- ▶ for文を使って辿る
 - ▶ 要素を取り出す
 - ▶ インデクスを指定する

```
s = 0
for x in numbers:
    s += x
    print(f'sに{x}を加算, s={s}')
print(s)
```

```
sqrs = list(numbers)
#リストのインデクスを用いて値を参照
for i in range(len(sqrs)):
    x = sqrs[i]
    sqrs[i] = x * x
print(sqrs)
```

リストの操作

- 要素の追加 : `append()`
 - `+` も使える
- 要素の挿入 : `insert()`
- 要素の取り出し : `pop()`
- 要素の削除 : `remove()`, `del()`

`dataStructure/modifyList.ipynb`

リストに対する判定

```
listA = ['red','green','blue']  
if 'red' in listA:  
    print("listA contains 'red'")
```

リストの変数名

- リストの変数名は、リストのデータが保存されている領域を指示している
 - 「参照」 (reference) と言う
- 代入は、リストに新たな参照を付けることに注意
 - 二つは同じもの

```
a=1  
b=a  
a+=1  
print(a)  
print(b)
```

リストの比較

▶ 比較は二種類あることに注意

▶ 内容が等しい: `==`

▶ 同じオブジェクト: `is`

▶ リストの複写: `copy()`

```
listA = ['red','green','blue']  
listB = ['red','green','blue']  
#listAに別名listCをつける  
listC = listA  
print(listA == listB)  
print(listC == listB)  
print(listA is listB)  
print(listA is listC)
```

リストの複写

- ▶ `list.copy()`で複写する
 - ▶ 内容は同じだが、別のリストができる

```
listD = listA.copy()
```

- ▶ `list()`の引数で指定しても良い

```
listD = list(listA)
```

リストの並び替え

```
data = ['Kim','Bob','Mary','Tom','Sam','Beth','Ann']  
data.sort()  
print(data)  
data.reverse()  
print(data)  
import random  
random.shuffle(data)  
print(data)
```

整列

逆順に整列

でたらめな順序

元のリストが変化していることに注意

```
['Beth', 'Bob', 'Kim', 'Mary', 'Sam', 'Tom']  
['Tom', 'Sam', 'Mary', 'Kim', 'Bob', 'Beth']  
['Kim', 'Sam', 'Mary', 'Bob', 'Beth', 'Tom']
```


文字列とリスト

```
text = 'may god bless you'  
wordList = text.split(' ')  
print(wordList)
```

```
joiner = ','  
text2 = joiner.join(data)  
print(text2)
```

多次元リスト

- リストの要素としてリストを用いることで多次元のリストを生成

```
colors = [  
    ['red',255,0,0],  
    ['breen',0,255,0],  
    ['blue',0,0,255],  
    ['yellow',255,255,0]  
    ]  
for c in colors:  
    print(c)  
print(colors[0])  
print('redの要素')  
redComponent = colors[0][1:]  
print(redComponent)
```

次回

- 7章「タプル」
- 8章「セット（集合）」
- 9章「辞書」