Python文法II

松尾・岩澤研究室
MATSUO-IWASAWA LAB UTOKYO

2024/03/19

目次

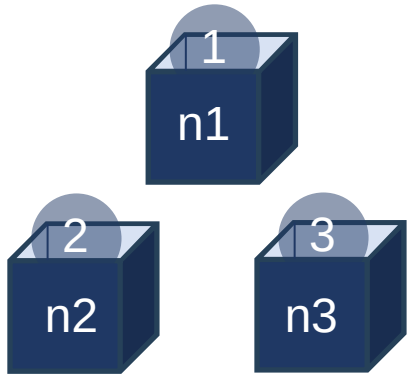
- 文法I
  - 演算
  - 変数
  - データ型
- 文法II
  - コレクション
  - List
  - Tuple
  - Dict
- 文法III
  - 条件分岐、繰り返し
  - 条件分岐(if文)
  - 繰り返し(for文)
- 文法IV
  - 関数
  - モジュール

コレクション

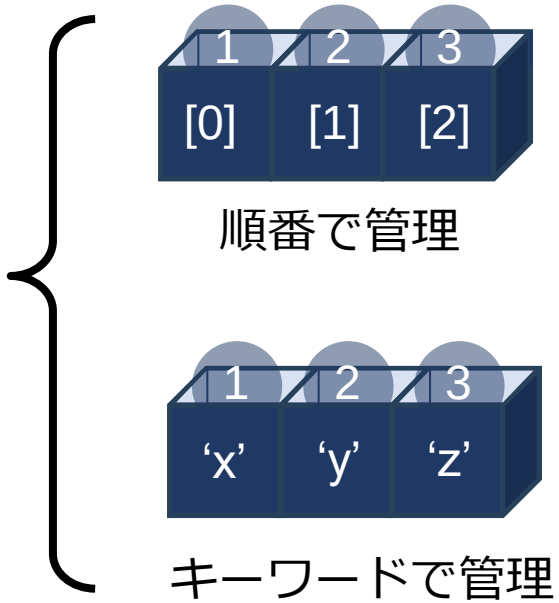
コレクションでは値をまとめて管理できます
リストやタプル、辞書などの種類があります

コレクションに値代入

変数ごとに値代入



まとめる



（例）

・リスト
・タプル

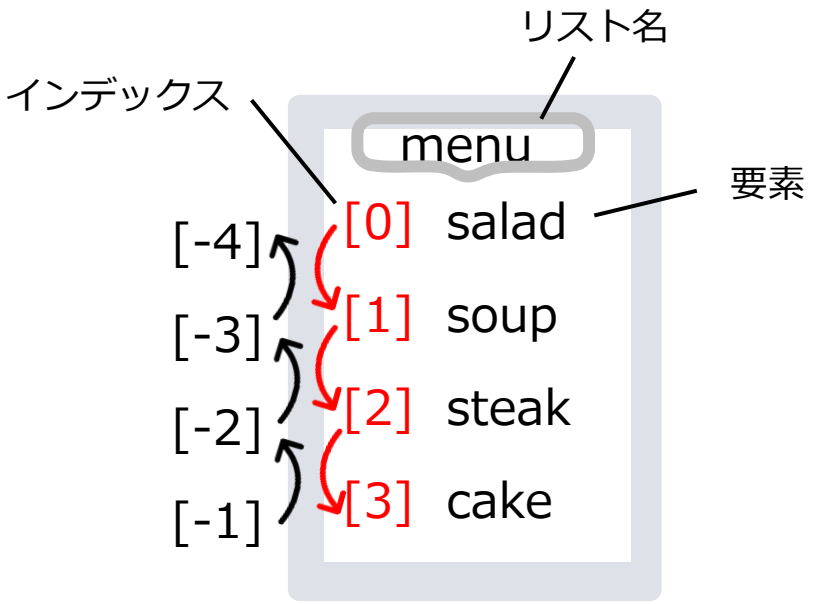
・辞書

コレクションでは変数名は1つになります

リスト

リストは角かっこ[]を使って値（**要素**）を順番（**インデックス**）で管理します
リストからインデックスを指定して要素の抽出・変更(置換)が可能

リストの要素とインデックス



メニューの2番目は？

最後プリンに変更

コード例

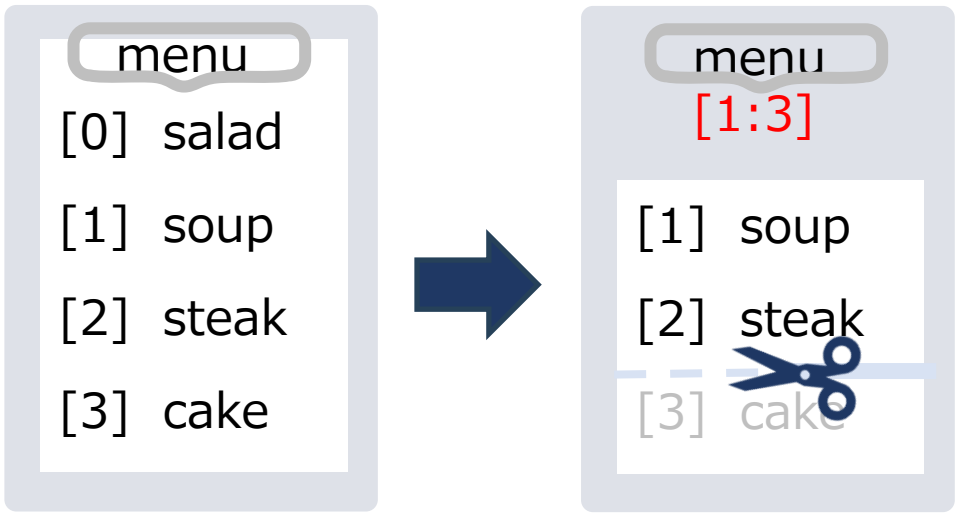
```
In  [1] : menu = ['salad', 'soup', 'steak', 'cake']
          print(menu[1])
Out [1] : soup
In  [2] : menu[-1] = 'pudding'
          print(menu)
Out [2] : ['salad', 'soup', 'steak', 'pudding']
```

インデックスは0から、後ろからは-1から数えます
リストの要素はstr型以外にint型やfloat型などのデータ型も可能

リストの切り出し

リストからインデックスの範囲を指定してリストを切り出せます

リストの切り出し：スライス



1st と 2nd ディッシュ

コード例

```
In  [1] : menu = ['salad', 'soup', 'steak', 'cake']
          print(menu[1:3])
Out [1] : ['soup', 'steak']
In  [2] : menu2 = menu[:-1]
          print(menu2)
Out [2] : ['salad', 'soup', 'steak']
```

ケーキなしメニュー

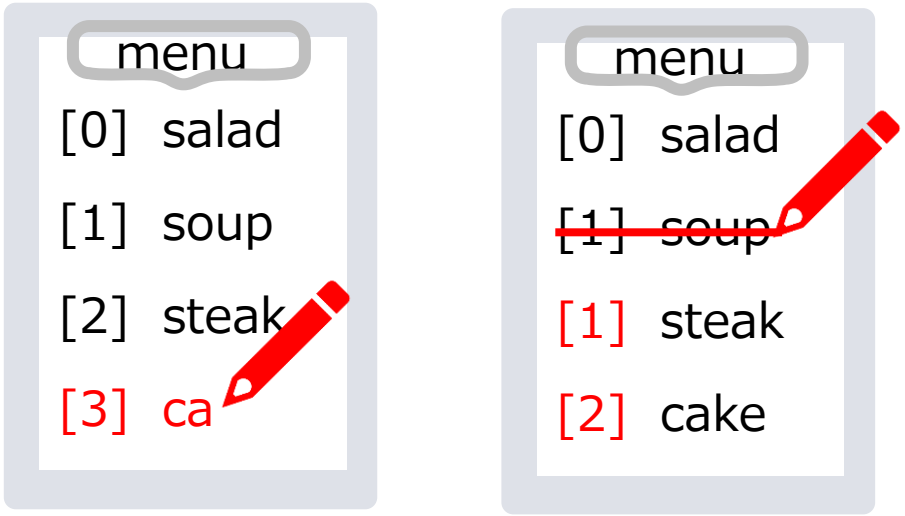
スライス表記の[x:y]はx以上y未満を表し、xやyを省略した場合は残る範囲まで選択されます

リストの追加・削除

リストは**要素の追加・削除**などの関数を持ちます
リストの変数名の後ろに.(ドット）を付けて利用できます

- **要素の追加**
  リスト変数名.append(末尾追加要素)

- **要素の削除**
  リスト変数名.remove(削除要素)



コード例

```
In  [1] : menu = ['salad', 'soup', 'steak']
          menu.append('cake')
          print(menu[-1])
Out [1] : cake
In  [2] : menu.remove('soup')
          print(menu[1:])
Out [2] : ['steak', 'cake']
```

removeで削除される要素がリスト内に重複する場合、該当する最初の要素が削除されます

リストの数値計算に用いる関数

リストの数値計算に便利な関数が用意されています

リストでよく用いる関数

関数	出力
len()	要素数
sum()	合計値
max()	最大値
min()	最小値

例）len(リスト)　➤ リストの要素数
　　max(リスト)　➤ リストの最大値

平均値＝
合計値÷要素数

コード例

```
In  [1] : score = [40, 60, 80]
          print(sum(score) / len(score))
Out [1] : 60
In  [2] : max_score = max(score)
          print(max_score))
Out [2] : 80
```

len関数のlenはlength（長さ）の略。他のデータ型でもよく用います

2次元リスト

2重の角かっこ[[]]を使って2次元の表データも管理できます
行や列番号を指定して要素やリストを抽出できます

2次元のリスト

(例) 2行3列の2次元リスト

	[0]列目	[1]列目	[2]列目
[0]行目	[['24/3' ,	'24/4' ,	'24/5'],
[1]行目	[300,	-400 ,	0]]

24/4

0行1列の要素
[0][1]

コード例

```
In  [1] : table = [['24/3', '24/4', '24/5'], [300, -400, 0]]
          print(table[0][1])
Out [1] : 24/4
In  [2] : print(table[1][2])
Out [2] : 0
```

1行2列の要素
[1][2]

3つ以上の多次元のリストも作成可能です

タプル

リストの中身を変更できないものをタプルと言います
(,)を使い、データを管理します

タプルは一度定義すると中身を変更できないので、特に順番やデータ型が変更されると困る際に使用します

それ以外は基本的にリストと同様です

コード例

```
In  [1] : color = (‘red’, ‘green’, ‘blue’)
          print(color[0])
Out [1] : red
In  [2] : menu = (‘rice’, 200)
          print(menu[1])
Out [2] : 200
```

データ型の違う要素を管理できる

辞書

辞書は{}を使い、キー(key)とそれに対応する値(value)のペアでデータを保持します
要素には順番がなく、各キーを一意(ユニーク)にすることで、ペアの値にアクセスします

辞書は
{キー：値, キー：値, …}の形を取る
例　{‘apple’ : 100, ‘banana’ : 80, …}
　　キー　　値

キーには数値、文字列、タプルなどが入れられる

値の追加、更新関数

辞書変数名[キー名] = 値

キー名のキーが辞書内に存在しない場合は
　新しい値が追加される
すでに存在している場合は
　そのキーに対応する値が更新される

コード例

```
In  [1] : menu = {‘apple’ : 100, ‘banana’ : 80, ‘orange’ : 120}
          print(menu[’apple’])
Out [1] : 100
In  [2] : menu[‘grape’] = 150
          print(menu)
Out [2] : {‘apple’ : 100, ‘banana’ : 80, ‘orange’ : 120, ‘grape’ : 150}
In  [3] : menu[‘banana’] = 60
          print(menu)
Out [3] : {‘apple’ : 100, ‘banana’ : 60, ‘orange’ : 120, ‘grape’ : 150}
```

キーが‘grape’で値が150の要素を追加

キーが’banana’の値を更新

辞書

del 変数名[キー名]で要素を削除でき、複数要素を同時に処理できます

要素の削除関数

del 辞書変数名[キー名] (, [キー名]…)

キーのみ指定

コード例

```
In  [1] : menu = {‘apple’ : 100, ‘banana’ : 80, ‘orange’ : 120}
          del menu[‘apple’]
          print(menu)
Out [1] : {‘banana’ : 80, ‘orange’ : 120}
In  [2] : menu = {‘apple’ : 100, ‘banana’ : 80, ‘orange’ : 120}
          del menu[‘apple’], menu[‘orange’]
          print(menu)
Out [2] : {‘banana’ : 80}
```

複数指定して削除も可能

松尾研究室

MATSUO LAB THE UNIVERSITY OF TOKYO