10 Python基礎

複数の値をまとめて扱う仕組み→コレクション

- コレクションには主に4つある

特徴	リスト	辞書	タプル	集合
格納方法	複数の値	複数のキーと値	複数の値	複数の値
定義方法	[値,値,...]	{キー:値,キー:値,...}	(値,値,...)	{値,値,...}
個別の要素参照	変数名[添字]	変数名[キー]	変数名[添字]	None
要素の追加	変数名.append(値)	変数名[キー]=値	None	変数名.add(値)
要素の変更	変数名[添字]=値	変数名[キー]=値	None	None
要素の削除	変数名.remove(値)	del 変数名[キー]	None	変数名.remove(値)
要素の順序関係	あり	なし	あり	なし
重複値の格納	可	キーは不可	可	不可
集合演算	不可	不可	不可	可
`len( )` 要素数	可	可	可	可
`sum( )` 合計値	可	不可	可	可
スライスの利用	可	不可	可	不可
+での連結	可	不可	可	不可

- コレクションの基本的な使用方法について、再度確認しておいて欲しい

基本的なデータの取り出し方（まとめ）

リスト

- リストから、データを取り出す
  【loop_list_1.py】

```
lst = [1, 3, 5, 7, 2, 4, 6, 8]

for n in lst:
    print(n)
```

- 要素の位置とデータを取り出す
  【loop_list_2.py】

```
lst = [1, 3, 5, 7, 2, 4, 6, 8]

for i, n in enumerate(lst):
    print(i, "番目の要素は", n)
```

- 新しいリストにコピーする
  【loop_list_3.py】

```
lst = [1, 3, 5, 7, 2, 4, 6, 8]

new_lst = []
for n in lst:
    new_lst.append(n)

print(new_lst)
```

【loop_list_4.py】

```
lst = [1, 3, 5, 7, 2, 4, 6, 8]

new_lst = [n for n in lst]

print(new_lst)
```

【loop_list_5.py】データによっては若干異なる場合がある

> https://atmarkit.itmedia.co.jp/ait/articles/1906/04/news009_4.html

```
lst = [1, 3, 5, 7, 2, 4, 6, 8]

new_lst = lst.copy()

print(new_lst)
```

  - deepcopyの意味と使い方を理解すること

辞書

- 辞書から添字（キー）を取り出す
  【loop_dic_1.py】

```
dic = {1: "a", 2: "b", 5: "c", 9: "d"}

for key in dic:
    print(key)
```

【loop_dic_2.py】

```
dic = {1: "a", 2: "b", 5: "c", 9: "d"}

dic_kyes = dic.keys()

print(dic_kyes)
```

【loop_dic_2_2.py】値だけを取り出す

```
dic = {1: "a", 2: "b", 5: "c", 9: "d"}

dic_values = dic.values()

print(dic_values)
```

- キーと値を取り出す
  【loop_dic_3.py】

```
dic = {1: "a", 2: "b", 5: "c", 9: "d"}

for key, value in dic.items():
    print(f"添字：{key} 値：{value}")
```

【loop_dic_4.py】

```
dic = {1: "a", 2: "b", 5: "c", 9: "d"}

for key in dic:
    print(f"添字：{key} 値：{dic[key]}")
```

- 新しい辞書にコピーする
  【loop_dic_5.py】

```
dic = {1: "a", 2: "b", 5: "c", 9: "d"}

new_dic = {}
for key in dic:
    new_dic[key] = dic[key]

print(new_dic)
```

【loop_dic_6.py】

```
dic = {1: "a", 2: "b", 5: "c", 9: "d"}

new_dic = {k: v for k, v in dic.items()}

print(new_dic)
```

【loop_dic_7.py】

```
dic = {1: "a", 2: "b", 5: "c", 9: "d"}

new_dic = dic.copy()

print(new_dic)
```

練習1

- タプル、集合に関しても、同様な処理ができるようなサンプルをそれぞれ作成してください。

オブジェクトの確認

- Pythonでは、値、関数など扱う全てはオブジェクトとして管理されている。
    - オブジェクトの管理番号(id)は、以下の命令で取り出すことができる。

```
obj_id = id(object)
```

- 確認してみよう
【obj_id.py】

```
a = 1
print(f"a.id = {id(a)} {a=}")

a = 2
print(f"a.id = {id(a)} {a=}")
```

```
a = [1, 2, 3]
print(f"a.id = {id(a)} {a=}")
print(f"a[1].id = {id(a[1])} {a=}")

a[1] = 10
print(f"a.id = {id(a)} {a=}")
print(f"a[1].id = {id(a[1])} {a=}")

a = [4, 5]
print(f"a.id = {id(a)} {a=}")
```

- 次のプログラム結果がわかるだろうか？
  【obj_id_2.py】

```
a = [1, 2, 3]
b = [1, 2, 3]

if a == b:
    print("同じ")
else:
    print("異なる")

if a is b:
    print("同じ")
else:
    print("異なる")

c = a
if a == c:
    print("同じ")
else:
    print("異なる")

if a is c:
    print("同じ")
else:
    print("異なる")
```

練習2

- 以下のプログラムを作成してください。（ren2_1.py）
  - 税抜き金額を渡すと、税込金額を返す関数`tax_include()`を作成する
    - 税込金額は、小数点以下切り捨てとする

- 消費税は、デフォルト引数として10%を設定する
  - 不正な税率の場合はNoneを返すものとする
- mainは、以下のような感じで正常に動作するかをテストできるようにする。

```
# main
price = 5000
# tax = 10 # デフォルト値を使用
print(f'{price}の税込金額は{tax_include(price)}円')

tax = 8
print(f'{price},消費税{tax}%の税込金額は{tax_include(price,tax)}円')

tax = -5
print(f'{price},消費税{tax}%の税込金額は{tax_include(price,tax)}円')
```

【実行結果】

```
5000の税込金額は5500円
5000,消費税8%の税込金額は5400円
5000,消費税-5%の税込金額はNone円
```

【関数例】

```
def tax_include( 省略;各自記述 ) -> Union[int , None]:
        """税込金額を計算する関数

        Args:
            price (int): 税抜き金額
            tax_rate (float, optional): 税率（%)。デフォルトは10.0

        Returns:
            int | None: 税込金額（小数点以下切り捨て)。不正な税率の場合はNone
        """
```

  - 型ヒントに関しては、バージョンにより書き方が異なることに注意
    - Python 3.9 `-> Union[int, None]:`
    - Python 3.10以降 `-> int | None:`

> https://gihyo.jp/article/2022/09/monthly-python-2209

- 以下のプログラムを作成してください。（ren2_2.py）

- 辞書に、商品名とリスト（税抜き金額と税率）の一覧が格納されている。
- 辞書を受け取り、各商品ごとのリストに税込金額を追加して返す関数`calc_tax()`を作成する
  - 税込金額は、ren2_1.pyの`tax_include()`を使用する

【実行結果】

```
商品名：クッキー,価格：200,消費税：8,税込価格：216
商品名：少年ジャンプ,価格：264,消費税：10,税込価格：290
商品名：NotePC,価格：13400,消費税：10,税込価格：14740
```

【初期データ】

```
# main
products = {
    "クッキー": [200, 8],
    "少年ジャンプ": [264, 10],
    "NotePC": [13400, 10]}
```

練習3

- 以下のプログラムを作成してください。(ren3.py)
  - 商品名と金額を登録するための、チェック関数`check_shopping()`を作成する
  - すべての入力が終わったらチェックする。
    - 金額は100円未満の場合エラーを表示。すべて100円以上なら問題なし。
  - 商品数は何点あるか分からない。

【実行結果1】OKな場合

```
> python func_kwd2.py
商品名を入力してください（0:終了）：abc
金額を入力してください（0:終了）：100
商品名を入力してください（0:終了）：zzzz
金額を入力してください（0:終了）：150
商品名を入力してください（0:終了）：0

全てのデータは問題ありませんでした
abc : 100
zzzz : 150
```

【実行結果2】NGな場合

```
> python func_kwd2.py
商品名を入力してください（0:終了）：aaaaa
金額を入力してください（0:終了）：80
商品名を入力してください（0:終了）：bbbb
金額を入力してください（0:終了）：150
商品名を入力してください（0:終了）：ccccc
金額を入力してください（0:終了）：110
商品名を入力してください（0:終了）：0

最低価格を下回った商品があります。
aaaaa : 80
bbbb : 150
ccccc : 110
```

【ren3.py】（例）

```
def check_shopping(order_dic: dict[str, int]) -> bool:
    ⋮
    ⋮


# main
order_dic = {}
 ⋮
 ⋮

if check_shopping(order_dic):
    print('\n全てのデータは問題ありませんでした')
else:
    print('\n最低価格を下回った商品があります。')

for key, value in order_dic.items():
    print(f'{key} : {value}')
```

練習4（提出）

- 以下のプログラムを作成してください。
    - 割り勘計算機【10_warikan.py】
        - 支払合計金額と参加者人数を入力するものとする。
        - 一人当たりの支払額は、支払合計金額を参加者人数で割った金額とする。
        - 支払の単位は100円とする。100円未満は切り上げる。
        - 支払金額合計を払って残った分は、幹事がもらえることとする。（端数分だけ幹事が安くなる）
    - 関数化して、見やすく分かりやすいプログラムを作成しよう。

- input_int(メッセージ)
  - メッセージを出力し、入力した値をintに変換して返す
- calc_payment(支払合計金額,参加者人数)
  - 1人当たりの支払額、幹事の支払額を計算する。
  - リストもしくは辞書で返す
- output_payment(1人当たりの支払額,幹事の支払額,参加者人数)
  - 引数を分かりやすく出力する
- `import math` は使用してもOK
  - math（数学）で良く使用される関数が利用可能（要調査）

【実行結果　例1】

```
支払合計金額を入力してください：10000
参加者人数を入力してください：3
支払金額：3,400円 (2人)
幹事金額：3,200円
```

【実行結果　例2】

```
支払合計金額を入力してください：10000
参加者人数を入力してください：6
支払金額：1,700円 (5人)
幹事金額：1,500円
```