B1001

この課題ではex10.pdfの内容を理解している前提で、 list, tuple, set, dict に関する命令(del 文) や関数、それぞれの型に用意されているメソッドの動作確認をしてもらいます。動作確認はコマンドプロンプトでpythonの対話モードに入って、提示されているコマンドを実行してください。ファイルの最後に簡単な課題がありますので、その課題のプログラムb1001.py とその出力結果 b1001.txt をB1001の課題として提出してください。

1. del 文と副作用

1.1 del文

del 文はpython の組み込み命令の一つです。関数やメソッドと書式が異なることに注意してください。

globals()関数は（大域）変数の一覧をdict型のコンテナとして返します。大域変数という言葉については、関数を学ぶときにまた解説しますので、ここでは単に変数と読んでください。

以下の命令は、int型のオブジェクトを生成し、xという名前をつけ、大域変数の一覧を確認した後、xという名前を消し、大域変数の一覧からxが亡くなったことを確認するコードです。

```
x=1
print(globals())
del x
print(globals())
```

`del x` の前後で `'x': 1` の部分に変化があるはずです。

1.2 for文でコンテナを壊す

次のコードは `x=2` の処理が行われていないように見えます。`del A[1]` を `x=1` のときに実行した結果、`A=[0,1,2,3]` から `A=[0,,2,3]` となり、抜けている部分を埋めるため `A=[0,2,3]` と添字をずらす操作が行われます。`A=[0,2,3]` として index が `2` のケース、すなわち `x=3` の処理が始まります。これはループの回数が1回減っていますので、4回ループを回すことを意図している場合には副作用となってしまいます。このようにループに使用しているコンテナを del 文で壊してしまうと意図しない動作の原因になります。

```
A=[0,1,2,3]
for x in A:
    print(x)
    if x==1:
      del A[x]
    print(A)
```

このコードの副作用を無くすには、for文のコンテナとして与えるのは `A` のコピーオブジェクト `A.copy()` を与えます。

```
A=[0,1,2,3]
for x in A.copy():
  print(x)
  if x==1:
    del A[x]
  print(A)
```

2. in 演算子

str型と同様に特定の要素がlist,tuple,set,dict にあるかどうかを確認するための演算子です。dict型の場合はkeyにデータがあるかを調べます。もし valueの方にデータがあるかを調べたい場合は `values()` メソッドを使ってください。

```
data={'perfect':'human','radio':'fish','nakata':'atsushi'}
'm' in data['perfect'] # True
'e' in data['perfect'] # False
1 in [8,8,3,1] # True
0 in [8,8,3,1] # False
'radio' in data # True
'fish' in data # False
'radio' in data.values() # False
'fish' in data.values()
```

3. dict ⇔keys, values

3.1 dict ⇒keys, values by keys(), values()

dictのkeyだけのリスト, valueだけのリストを抜き出す方法です。

```
data={'perfect':'human','radio':'fish','nakata':'atsushi'}
data_keys=list(data.keys())
data_values=list(data.values())
print(data_keys,data_values)
```

3.2 keys, values ⇒ dict by zip()

先と逆の操作として、2つのlistやtupleのデータが与えられているときに、それらを対にして dict を作成できると楽にdictを作れるようになります。

```
x=[0,1,2,3,4]
y=[3,2,4,1,5]
a=dict(zip(x,y))
b=dict(zip(y,x))
print(a)
print(b)
```

4. 要素調査系メソッド count(), index()

リストやタプルは要素の重複が可能です。 `count()` メソッドは検索語に該当する要素の個数を数えます。

```
print([1,2,2,3,3,3,4,4,4,4,5,5,5,5,5].count(4))
```

`index()`メソッドで検索語のある要素のindexを調べることができます。複数ある場合は最初に見つかった要素のindexを返します。

```
print([1,2,2,3,3,3,4,4,4,4,5,5,5,5,5].index(4))
```

5. 要素追加系メソッド append(),insert(),extend()

`append()` はリストの最後に要素を追加します。

```
a=[1,2,3]
a.append('dog')
print(a)
a.append('cat')
print(a)
```

`insert()` は指定したindexの位置に要素を追加します。それより右にある要素はindexを一つずつずらします。要素数が多いと処理時間がかかります。

```
a=[1,2,3]
a.insert(0,'dog')
print(a)
a.insert(0,'cat')
print(a)
```

extend() はリストの連結ができます。+ でも連結ができます。extend()メソッドを利用した場合はオブジェクトを直接操作するため a のid は変化しません。元のaの情報を破壊して変化させているの破壊的操作としてみなせます。一方+演算子による連結は非破壊的操作です。id(a)の値に注目すると+演算してaで名前をつけ直している前後で異なるidになっていることが観測できます。

```
a=[1,2,3]
a.extend([4,5])
print(a)
print(id(a))
a=a+[6,7,8]
print(a)
print(id(a))
```

6. 要素削除系メソッド clear(), pop(), remove()

`clear()`は要素を全て削除して空のリストにしてしまいます。

```
a=[1,2,3]
a.clear()
print(a)
```

`pop()`は特定した位置の要素を削除します。またメソッドの戻り値は削除した要素です。

```
a=[1,2,3,4,5]
x=a.pop(0)
print(x,a)
x=a.pop()
print(x,a)
```

`remove()` は指定した値を削除します。指定した値が複数ある場合は最初に見つかった要素 1 つだけを削除します。

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

7. sort()メソッドとsorted() 関数

要素が順番に並んでいない場合に要素を順番に並べたいことがあります。その場合はソートをします。`sort()` メソッドはリストに対して使えます。このメソッドはリストオブジェクトの中身を書き換えてしまいますので、破壊的操作です。

```
a=[4,3,5,2,4,1]
a.sort()
print(a) # a そのものが書き換えられた。
```

一方で aの順番の情報を残しておきたい場合があります。`copy()` メソッドで複製を作ってから `sort()` メソッドでソートするか、もしくはsorted()関数で元のデータは残したままソートした後のオブジェクトを作成することができます。このように元の情報を残す操作を非破壊的操作といいます。タプルはもともと書き換え不可能ですから、sorted()関数を使って別のオブジェクトを生成するしかできません。

```
a=[4,3,5,2,4,1]
b=a.copy()
b.sort()
print(a) # オリジナル a はそのまま
print(b) # ソートしたもの

a=(4,3,5,2,4,1)
b=sorted(a)
print(a)
print(b)
```

8. unpacking

リストやタプルの要素を受け取るには参照を使う以外にも便利な書式があります。

```
a=['takada','hiroyuki']
sei,mei=a # 変数の個数とコンテナの要素数は一致していないといけない
print(sei,mei)

x=('toshi','hide','pata','taiji','yoshiki')
vocal,guitar1,guitar2,base,drum=x
print(drum)
```

9. 内包表記 list comprehension

数学では集合の要素を列挙する外延的記法と要素の満たす性質を記述する内包的記法とがあります。

例えば1~40までのナベアツ数集合の外延的記法は、

$$\{3, 6, 9, 12, 13, 15, 18, 21, 23, 24, 27, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39\}$$

です。一方で、内包的記法は、

$$\{x | x \text{は整数}, 1 \leq x \leq 40, x \text{は} 3 \text{で割り切れるかまたは} 3 \text{のつく数である。}\}$$

となります。

問(i) プログラミング概論の資料(ex10.pdfの6/9ページ参照)の内包表記を利用して、１～９９の整数のうち以下の条件を満たす整数からなるリストを作成し、表示するコードを作成してください。

- 2の倍数からなるリスト
- 3の倍数からなるリスト
- 3のつく数からなるリスト
- ナベアツ数のリスト（３の倍数または３のつく数）
