情報リテラシー2B レポート課題 1

情報科学科 22140026 谷 知拓

```
1  Ns = list(range(1, 11))  # 定数
2
3  def poper(n, p):
4      n.remove(p) if p in n else n
5      return n
6
7  combs = []
8  for a in Ns:
9      for b in poper(Ns.copy(), a):  # .copy()に注意! python 特有の「参照渡し」に注意!
10         for c in poper(poper(Ns.copy(), a), b):
11             combs.append([a, b, c])
12
13 print(combs)
14 print(len(combs))
15
16 max_comb = combs[0]
17 N = int(input("Please input the value of N: "))
18 for m in combs:
19     x = (m[0]/m[1])**m[2]
20     if (x < N) and (x >= (max_comb[0]/max_comb[1])**max_comb[2]):
21         max_comb = m
22
23 print(max_comb)
24 print((max_comb[0]/max_comb[1])**max_comb[2])
25
```

行	実行内容
1	1 から 10 までの自然数の集合(正確には Python3 では, range 型)を **list 型に変換し**定義する(Ns). このオブジェクトは, **定数として扱い**今後書き換えない.
3~5	poper()関数を定義する. {引数: {n: list, p: int}, 返り値: list} これは, 受け取った n: list から, p: int の**要素を消して返却する関数**である. プログラム内で**複数回使い**, また**再帰的に利用する場面があるため関数化した.** エラー回避のため, n に p が含まれていない場合には, 三項演算子を用いて 4 行目の n.remove(p)を実行しないようにしている.
7	空のリストを定義している. ここに, 生成したコンビネーション(組み合わせ)を追加していく.

8~11	Ns から 1 要素ずつ取り出して, a に格納し，それを繰り変えし処理する. 次に，上のループの内で, **Ns を.copy()によって複製したものから,** 自作の poper()関数を用いて, a の要素を除いた list を取得し，そこから 1 要素ずつ取り出して, b に格納して，それを繰り返し処理する. さらに，上のループの内で, **Ns を.copy()によって複製したものから,** poper()関数で, a の要素を除き，**続いてその返り値から, b の要素を除く動作を再帰的に**行う. 最後に，取り出した a, b, c を list にして, combs: list の一要素として末尾に追加する. **＊Python では，リストのコピーには細心の注意を払わなければならないことに注意!** **→ 詳細は後述.**
13~14	生成したコンビネーション(組み合わせ)たちを出力．また，コンビネーション(組み合わせ)の数も出力.
16	次に「条件 1」および「条件 2」にマッチするコンビネーション(組み合わせ)を探す. 最初に，仮に一番初めに生成されたコンビネーション(組み合わせ)を条件に合致する暫定的なコンビネーション(組み合わせ)として格納する.
17	問題文中の N を input()により取得する.
18~21	生成されたコンビネーション(組み合わせ)を 1 つずつ取り出して, m に格納する. m: list の中には，それぞれの a, b, c が順に index: 0, 1, 2 で含まれているので，これを m[0], m[1], m[2]で取り出して，評価する．もし，暫定的なコンビネーション(組み合わせ)よりもより適当なコンビネーション(組み合わせ)が見つかった場合は，上書きする．これを繰り返す.
23~24	「条件 1」および「条件 2」にマッチするコンビネーション(組み合わせ)を出力する．また，その組み合わせの(a/b)**c の値を出力する.

＊Python では，リストのコピーには細心の注意を払わなければならないことに注意!

→ 以下の 3 パターンが存在する

- 通常の代入 b = a

  通常の代入では，オブジェクト自体が代入されるため，どちらか片方を編集した場合でも，編集内容はもちろん共有される.(オブジェクトが同じため．実体が同じため)

- 「浅いコピー」 b = a.copy()

  浅いコピーでは，リストの値がコピーされた新しいオブジェクトが作られる．被コピー対象とコピー結果とは，互いに独立しているため，編集も相互に影響しあわない.

  ただし，被コピー対象の要素に別のリストオブジェクトが含まれていた場合，その部分は，値のコピーではなく，オブジェクト自体の代入となる.

- 「深いコピー」 b = copy.deepcopy(a)

  浅いコピーと似ているが，深いコピーでは，被コピー対象の要素に別のリストオブジェクトが含まれていた場合でも，その部分についても，値のコピーとして実行される.

  被コピー対象とコピー結果が，最も独立していて，互いに干渉しないコピーの方法.