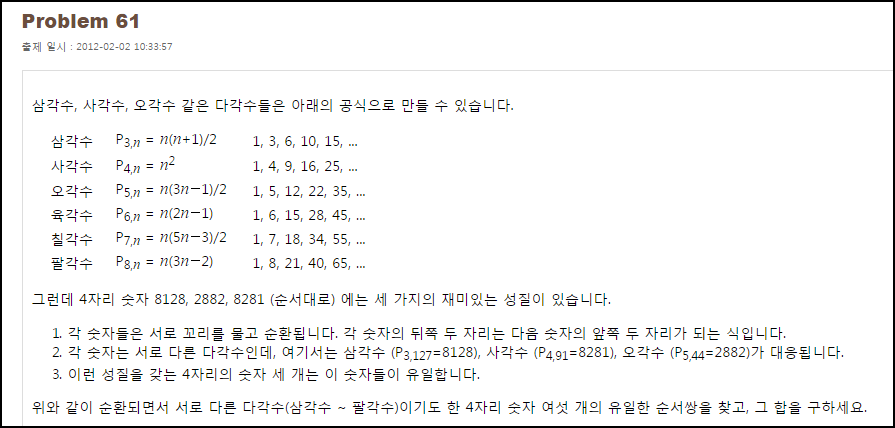
**Exercise**

삼각수~팔각수 인 4자리 숫자 중에서 앞 뒤 2글자씩 6개의 숫자가 순환되는 순서쌍을 찾는 문제입니다.



**Solution\_code**

code 가 상당히.. 지저분한 느낌입니다.. 재귀를 쓰려다 머리로 생각이 잘 안되서, 하드 코딩을 하긴 했는데,, 나름 빨리 나오기는 한 것 같습니다. 처음엔, 해당 범위의 각 다각수 집합을 구한 뒤 교집합, 합집합, 차집합을 이용해 풀어보려고 했으나 생각보다 비효율 적이라 for 문을 도는 다각수의 순서쌍을 구한 뒤 for 문에서 각 경우의 수 별로 다각수를 돌며 순환되는 숫자의 쌍을 찾는 방식입니다. 3각수~8각수의 리스트 명을 가지고 6개 뽑는 순열을 구한 뒤, 해당 순열 요소 별로 for 문에 넣고 각 요소 값과 dictionary value 를 이용해서 해당 set 을 매치 시켰습니다.

|  |  |  |
| --- | --- | --- |
| **1**  **2**  **3**  **4**  **5**  **6**  **7**  **8**  **9**  **10**  **11**  **12**  **13**  **14**  **15**  **16**  **17**  **18**  **19**  **20**  **21**  **22**  **23**  **24**  **25**  **26**  **27**  **28**  **29**  **30**  **31**  **32**  **33**  **34**  **35**  **36**  **37**  **38**  **39**  **40**  **4** | **from itertools import \***  **import time**  **st\_time = time.time()**    **n = 1**  **p3\_s = set()**  **p4\_s = set()**  **p5\_s = set()**  **p6\_s = set()**  **p7\_s = set()**  **p8\_s = set()**  **p\_total = []**    **for n in range(1,150):**    **P3 = n\*(n+1)/2**  **P4 = n\*\*2**  **P5 = n\*(3\*n-1)/2**  **P6 = n\*(2\*n-1)**  **P7 = n\*(5\*n-3)/2**  **P8 = n\*(3\*n-2)**    **if P3 < 10000 and P3 > 1000:**  **p3\_s.add(P3)**  **p\_total.append(P3)**    **if P4 < 10000 and P4 > 1000:**  **p4\_s.add(P4)**  **p\_total.append(P4)**    **if P5 < 10000 and P5 > 1000:**  **p5\_s.add(P5)**  **p\_total.append(P5)**    **if P6 < 10000 and P6 > 1000:**  **p6\_s.add(P6)**  **p\_total.append(P6)**    **if P7 < 10000 and P7 > 1000:**  **p7\_s.add(P7)**  **p\_total.append(P7)**    **if P8 < 10000 and P8 > 1000:**  **p8\_s.add(P8)**  **p\_total.append(P8)**    **def Slice\_front(n):**  **return str(n)[:2]**  **def Slice\_back(n):**  **return str(n)[2:]**      **p\_list = ['p3\_s','p4\_s','p5\_s','p6\_s','p7\_s','p8\_s']**  **dic = {'p3\_s':p3\_s,'p4\_s':p4\_s,'p5\_s':p5\_s,'p6\_s':p6\_s,'p7\_s':p7\_s,'p8\_s':p8\_s}**  **pp\_list = list(permutations(p\_list,6))**    **for z in pp\_list:**  **# print z**  **for i in dic[z[0]]:#p3\_s:**  **for j in dic[z[1]]:#p5\_s:**  **if Slice\_back(i) == Slice\_front(j):**  **for k in dic[z[2]]:#p4\_s:**  **if Slice\_back(j) == Slice\_front(k):**  **for l in dic[z[3]]:**  **if Slice\_back(k) == Slice\_front(l):**  **for o in dic[z[4]]:**  **if Slice\_back(l) == Slice\_front(o):**  **for p in dic[z[5]]:**  **if Slice\_back(o) == Slice\_front(p):**  **if Slice\_back(p) == Slice\_front(i):**  **print '[+]',i,j,k,l,o,p, z[0],z[1],z[2],z[3],z[4],z[5], i+j+k+l+o+p**  **print '[\*]',time.time()-st\_time**  ***[Colored by Color Scripter](http://colorscripter.com/info#e)*** | [cs](http://colorscripter.com/info#e) |

**Result**

만족하는 6개의 숫자의 순서쌍이 하나 나오는 것을 확인 할 수 있었습니다. 시간은 만족스럽긴한데,.. 나중에 재귀를 이용하거나, 좀 더 효율적인 알고리즘으로 해결해볼 예정!

