Задание №8, Комбинаторика

Теория

itertools - стандартная библиотека Python, содержащая распространённые шаблоны итераторов. Бесконечные счётчики, сочетания и размещения, итераторы среза и многое другое.

product(p,repeat=n) - для этого задания: всевозможные перестановки из набора **p** с повторениями, где **n** длина комбинаций. Перебирает все комбинации, где порядок важен и **разрешены повторы букв из набора**.

permutations(p, n) - всевозможные перестановки из набора **p** без повторений (если только не повторяются символы в p, тогда надо брать **set(list(permutations(p)))**)

combinations(p,r) - всевозможные сочетания (комбинации без повторяющихся элементов и учёта порядка) из набора **p** длины **r**

Функция	Порядок важен?	Повторы?	Пример для 'ABC', 2 буквы
`product`	✅ Да	✅ Да	AA, AB, AC, BA, BB, BC, CA, CB, CC
`permutations`	✅ Да	❌ Нет	AB, AC, BA, BC, CA, CB
`combinations`	❌ Нет	❌ Нет	AB, AC, BC

💡 Часто встречающиеся в этом задании функции и методы Python:

s.count(x) - количество элементов **x** в списке **s** или количество подстрок **x** в строке **s**

“”.join(s) - слияние элементов списка **s** в строку без разделителя (разделитель - пустая строка “”)

len(s) - длина строки/списка **s** (= количество элементов)

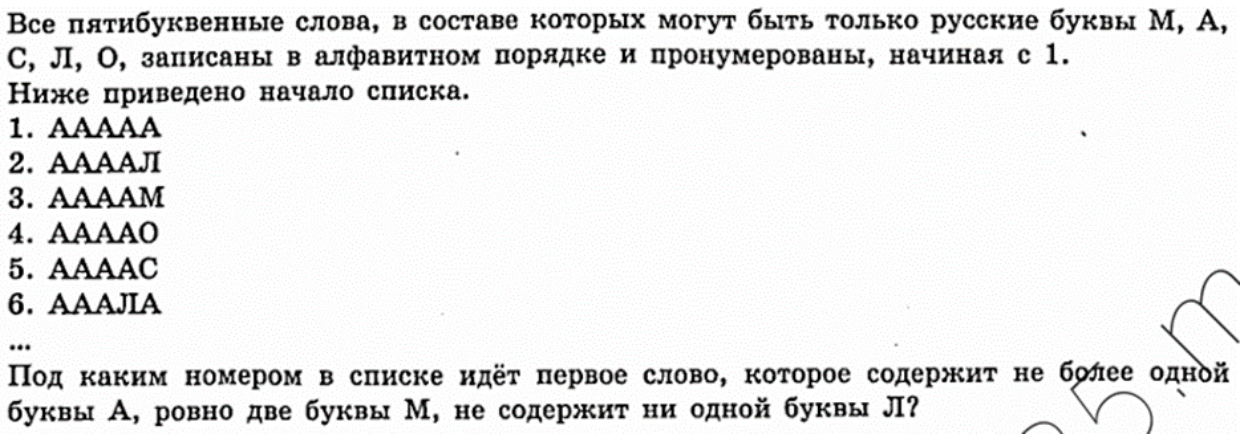
set(s) - оставляет только уникальные элементы в списке/строке **s**

s.find(x) - индекс первого нашедшегося элемента **x** в списке **s** или вхождения подстроки **x** в строку **s**

План решения

Номер слова

Крылов, 15 вариант



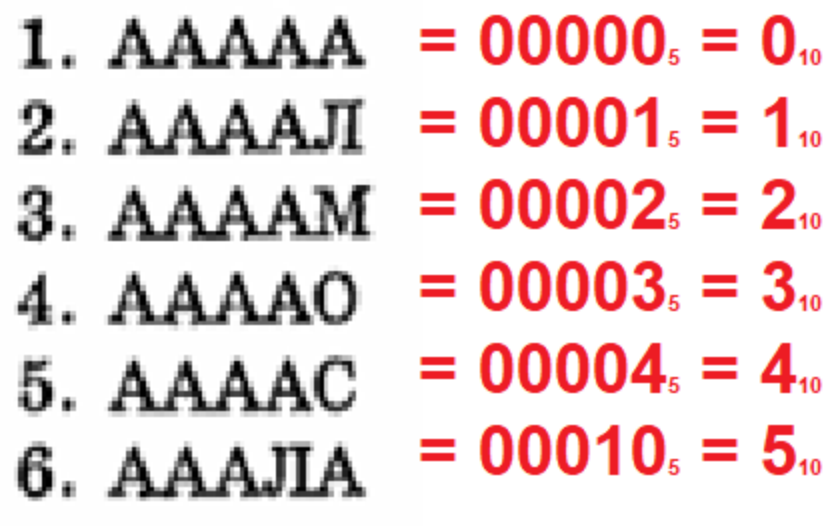
Все пятибуквенные слова, в составе которых могут быть только русские буквы М, А, С, Л, О, записаны в алфавитном порядке и пронумерованы, начиная с 1.
Ниже приведено начало списка.
1. ААААА
2. ААААЛ
3. ААААМ
4. ААААО
5. ААААС
6. АААЛА
...
Под каким номером в списке идёт первое слово, которое содержит не более одной буквы А, ровно две буквы М, не содержит ни одной буквы Л?

Руками

В заданиях на нахождение номера слова можно представлять слова как число в системе счисления, базис которой равен мощности набора используемых букв, цифры которого равны номеру буквы в порядке, установленном примером в условии (как правило алфавитный порядок). Тогда, если список в условии начинается с 1, то и порядковый номер слова

будет равен соотнесённому числу, переведенному в десятичный вид, увеличенному на 1.

1. Т.к. в словах используются 5 букв, то можно построить соответствие слов числам в пятеричной системе счисления. Тогда, т.к. слова записаны в алфавитном порядке (смотрим на приведенное начало списка), то **А** будет соответствовать цифре **0**, **Л - 1, М - 2, О - 3, С - 4.**
2. Заметим, что при переводе этого пятеричного числа в десятичный вид получается число, которое на 1 меньше порядкового номера.



3. Преобразуем условие для числового вида: не более одной **А ⇒ не более одного 0, ровно две М ⇒ ровно две 2, ни одной Л ⇒ ни одной 1.**
4. Т.к. для ответа нам нужно найти первое число, то оно должно быть наименьшим. Подбираем такое число, учитывая условие. Очевидно, что для наименьшего числа в самые левые разряды стоит ставить наименьшие цифры, это будет **0.** Мы не можем ставить его более 1 раза, поэтому дальше используем следующие по наименьшести. 1 мы по условию использовать не можем, поэтому дальше идёт 2, которых у нас должно быть ровно две штуки. Оставшиеся разряды заполняем 3. Получаем число **02233**, которое соответствует слову **АММОО**.
5. Переводим это пятеричное число в десятичный вид и получаем 318. Порядковый номер будет на 1 больше. Получаем ответ **319**.

Кодом

1. Импортируем из библиотеки **itertools** функцию **product** (в 99.9% задач используется именно она, **permutations** изредка используется для перестановки набора букв без повторений).

```
from itertools import product
```

2. Вводим переменную порядкового номера.

```
n = 1
```

3. Создаём цикл перебора всех возможных перестановок из данного в условии набора букв (с повторениями букв из набора), указав вторым аргументом **repeat=** длине слова. Сам набор букв должен идти в том порядке, который указан в условии в качестве начала списка.

```
for x in product('АЛМОС', repeat=5):
```

4. Для удобства можем перевести **x** из кортежа символов в обычную строку через **‘’.join(x)**.

```
    x = ''.join(x)
```

5. Записываем заданное условие.

```
if x.count('А') <= 1 and  x.count('М') == 2 and x.count('Л') == 0:
```

6. Если условие выполняется, то выводим порядковый номер и само слово. Нам нужно только первое подошедшее, поэтому можем выйти из цикла через break.

```
        print(n, x)
        break
```

7. В конце итерации цикла увеличиваем переменную номера числа на 1.

```
    n += 1
```

8. Получаем вывод в консоль «**319 АММО**». Весь код:

```
from itertools import product

n = 1
for x in product('АЛМОС', repeat=5):
    x = ''.join(x)
    if x.count('А') <= 1 and  x.count('М') == 2 and x.count('Л') == 0:
        print(n, x)
        break
    n += 1
```

Подсчёт количества

Демовариант 2025

Определите количество 12-ричных пятизначных чисел, в записи которых ровно одна цифра 7 и не более трёх цифр с числовым значением, превышающим 8.

Кодом

Перебор всевозможных чисел аналогичен перебору всевозможных слов из алфавита мощности, равной базе системы счисления числа. **Т.е. количество слов длины 5, которые можно составить из 4-х букв (буквы могут повторяться сколько угодно раз), равно количеству чисел длины 5 в четверичной системе счисления. Только важно проверять, чтобы первый левый разряд числа не был равен 0.**

1. Вводим переменную подсчёта подходящих чисел.

```
cnt = 0
```

2. Создаём цикл перебора всех возможных 12-ричных пятизначных чисел.

```
for x in product('0123456789AB', repeat=5):
```

3. Если у числа первый левый разряд будет равен 0, то оно не будет пятизначным, поэтому их учитывать не будем.

```
    if x[0] != '0':
```

4. Записываем условие из задачи. Если число удовлетворяет ему, то увеличиваем переменную подсчёта.

```
    if x.count('7') == 1 and (x.count('9') + x.count('A') + x.count('B')) <= 3:
        cnt +=1
```

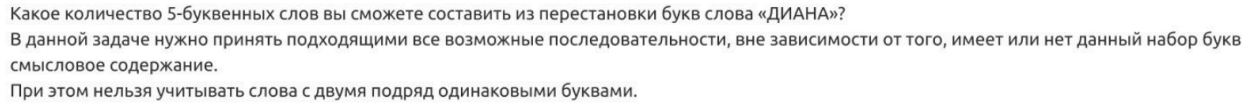
5. Вне цикла (после окончания подсчёта) выводим счётчик. Получаем ответ **67476**. Весь код:

```
from itertools import product

cnt = 0
for x in product('0123456789AB', repeat=5):
    if x[0] != '0':
        if x.count('7') == 1 and (x.count('9') + x.count('A') + x.count('B')) <= 3:
            cnt +=1
print(cnt)
```

Дополнительно: permutations (могло быть на ЕГЭ до 21 года, но сейчас нет ни в банке заданий ЕГЭ, ни у Крылова, поэтому шанс его встретить стремится к 0)

Когда идёт речь про количество слов, которое можно составить из букв определенного слова без уточнения, что буквы могут использоваться сколько угодно раз, речь идёт о перестановках без повторений, функции **permutations**. Т.е. буквы можно использовать ровно столько раз, сколько они встречаются в слове. Обязательно использовать **set**, чтобы оставить только уникальные наборы.


Какое количество 5-буквенных слов вы сможете составить из перестановки букв слова «ДИАНА»?
В данной задаче нужно принять подходящими все возможные последовательности, вне зависимости от того, имеет или нет данный набор букв смысловое содержание.
При этом нельзя учитывать слова с двумя подряд одинаковыми буквами.

1. Вводим переменную подсчёта подходящих чисел.

```
cnt = 0
```

2. Т.к. мы ищем количество слов, которые можно составить из перестановок букв слова «ДИАНА» и, следовательно, в этих словах должно быть столько же букв, сколько и в заданном слове, то используем функцию **permutations(”ДИАНА”, 5)**, где 5 - длина слов, которые нам нужно найти. Важно заметить, что permutations также будет содержать идентичные слова из-за 2-х неразличимых **А (т.е. все слова будут встречаться дважды).** Поэтому обязательно циклом перебираем именно уникальные наборы, применив функцию **set**.

```
for x in set(permutations('ДИАНА', 5)):
```

3. Для удобства проверки на наличие двух букв подряд, будем приведить кортежи значений **x** к строке через **‘’.join(x).**

```
    x = ''.join(x)
```

4. Универсальный метод проверки на две идущие подряд буквы: для каждого элемента, кроме последнего (чтобы не выйти за границы), проверяем, что следующий элемент не равен ему. Если они всё-таки оказываются равны, то выходим из цикла через ***break***, тогда **else** для

этого цикла не будет срабатывать и слово не будет считаться в **cnt**.

```
for i in range(len(x)-1):
    if x[i] == x[i+1]:
        break
else:
    cnt += 1
```

Либо можем просто заметить, что повторяться может только буква А, т.к. только она повторяется в слове «ДИАНА». Тогда просто проверяем, что **x** не содержит сочетания «АА»

```
if 'AA' not in x:
    cnt += 1
```

5. Выводим полученное количество через print вне цикла. Получаем ответ **36**. Весь код:

```
cnt = 0
for x in set(permutations('ДИАНА', 5)):
    x = ''.join(x)
    if 'AA' not in x:
        cnt += 1
print(cnt)
```