Конспект №2

На прошлом занятии разбирали основы перебора различных комбинаций, задачи с простыми условиями. На этом занятии узнаем о модуле `itertools`, который упрощает способ решения и дает новые возможности. `itertools` позволяет организовывать перебор комбинаций проще, короче и удобнее.

Задача №1 (212)

Сколько слов длины 6, начинающихся и заканчивающихся с согласной буквой, можно составить из букв ГОД? Каждая буква может входить в слово несколько раз. Слова не обязательно должны быть осмысленными словами русского языка.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241179?t=1m25s

Решение

Для примера сначала напишем код, как писали в прошлый раз. В начале создаем счетчик для подсчета подходящих слов и присваиваем ему значение 0. Создаем шесть вложенных циклов, которые перебирает все варианты букв.

```
k = 0
for a1 in 'ГД':
  for a2 in 'ГОД':
    for a3 in 'ГОД':
      for a4 in 'ГОД':
        for a5 in 'ГОД':
          for a6 in 'ГД':
            k = k + 1
print(k)
```

Ответ: 324.

С помощью модуля itertools, мы можем вместо шести циклов написать один, который сразу получит все такие слова.

Модуль itertools – это набор вспомогательных, уже готовых, написанных функций, которые идут вместе с самим языком, как дополнение.

Нам нужны три из них: product, permutations и combinations.

product

Функция, которая имитирует работу вложенных циклов for. Она организует то, что называется декартовым произведением, то есть перебор всевозможных комбинаций.

С помощью этой функции можно вместо шести вложенных циклов писать один.

permutations - перебор перестановок.

combinations - перебор различных комбинаций неповторяющихся элементов.

Перепишем ранее написанный код, используя product.

Первое, что надо сделать – подключить этот модуль. Для этого мы пишем команду

```
from itertools import *
```

после слова import можно указать конкретные функции, например, product, или можно ставить звездочку, которая означает импорт всех функций в программу.

Запишем

```
for x in product('ГД','ГОД','ГОД','ГОД','ГОД','ГД')
```

Наборы букв: 'ГД' – это возможные значения для первой буквы, потом 'ГОД' значения для второй. Таким же образом для третьей, четвертой, пятой и для шестой. То есть тем самым указываем, что получаются всевозможные комбинации из шести элементов. Таким образом, шесть циклов заменяются одним.

Заводим счетчик подходящих слов и увеличиваем его на каждой итерации.

```
from itertools import *

k = 0
for x in product('ГД','ГОД','ГОД','ГОД','ГОД','ГД'):
    k = k + 1

print(k)
```

Запустим программу и убедимся, что ответ получится такой же.

Посмотрим, что в этой программе перебирается. Для чего выведем эти комбинации на экран. Это конструкция называется кортеж, т.е. неизменяемый список.

```
>>>
...
('Д', 'Д', 'О', 'О', 'Д', 'Д')
('Д', 'Д', 'О', 'Д', 'Г', 'Г')
('Д', 'Д', 'О', 'Д', 'Г', 'Д')
('Д', 'Д', 'О', 'Д', 'О', 'Г')
...
```

Можно сократить эту запись и не писать для каждой буквы набор символов. Для этого есть специальный параметр, который называется repeat. Для упрощения выражения проверки на согласные заменим все согласные в слове ГОД на Г.

После такой замены необходим проверить, что первая и последняя буква – согласные.

```
k = 0
for a1,a2,a3,a4,a5,a6 in product('ГОГ', repeat = 6):
    if a1 == 'Г' and a6 == 'Г':
        k = k + 1
print(k)
```

Ответ: 324

Задание №2 (1910)

Из букв слова Р У С Т А М составляются 6-буквенные последовательности. Сколько можно составить различных последовательностей, если известно, что в каждой из них содержится не менее 3 согласных?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241179?t=17m55s

Решение

Вначале также подключаем модуль, иначе нужные функции не появится. Потом описываем перебор вариантов. Чтобы не писать набор букв 6 раз подряд, пишем repeat=6, то есть повтори 6 раз.

```
from itertools import *
for a1,a2,a3,a4,a5,a6 in product('РУСТАМ',repeat=6):
```

Работать удобнее со строкой, а не с кортежем.

```
s = a1+a2+a3+a4+a5+a6
```

Можно сразу заменять все согласные буквы на одну, например, на 'Р' в цикле.

Так можно делать, если общее количество комбинаций не изменится и это не мешает проверке прочих условий.

```
from itertools import *

for a1,a2,a3,a4,a5,a6 in product('РУРРАР',repeat=6):
    s = a1+a2+a3+a4+a5+a6
    # если в итоговой строке 3 и более букв Р,
    # то в строке без замен было бы три согласных
    if s.count('Р')>=3:
        k = k + 1
print(k)
```

Ответ: 41984.

Задание №3 (1911)

Василий составляет 4-буквенные коды из букв В, А, Я, Ю, Щ, И, Й. Каждую букву можно использовать любое количество раз, при этом код не может начинаться с буквы Й и должен содержать хотя бы одну гласную. Сколько различных кодов может составить Василий?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241179?t=32m45s

Решение

Для удобства подсчета гласных заменяем все гласные на букву А. Нужно будет проверять в цикле, что Й не может быть первой буквой, и чтобы хотя бы одна

гласная (А) была в слове.

```
from itertools import *

k = 0
for a1,a2,a3,a4 in product('ВАААЩАЙ',repeat=4):
    s = a1+a2+a3+a4
    if a1 != 'Й' and s.count('А')>0:
        k += 1 #k = k + 1
print(k)
```

Ответ: 2004

Задание №4 (1930)

Вася составляет 6-буквенные коды из букв П, А, Н, Е, Л, Ь. Каждую букву нужно использовать ровно 1 раз, при этом код не может начинаться с буквы ‘Ь’ и не может содержать сочетания ‘ЕЬ’. Сколько различных кодов может составить Вася?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241179?t=40m30s

Решение

Эта задача отличается от рассмотренных ранее задач, так как есть ограничение, что каждую букву нужно использовать ровно один раз. Это перестановка букв, они просто меняются местами. Использование вложенных циклов здесь нецелесообразно, потому что вложенными циклами нельзя сделать так, чтобы в слове были буквы из набора ПАНЕЛЬ и все они встречались ровно по 1 разу.

Для решения такого типа задач, когда все буквы в слове встречаются ровно 1 раз, в модуле *itertools* существует специальная функция *permutations*, которая организует нам перебор перестановок.

В предыдущих задачах некоторых букв могло и не быть в слове или они могли встречаться несколько раз. Поэтому мы использовали функцию product.

В этой задаче количество встречающихся букв в слове ограничено, каждая буква встречается ровно один раз, значит это перестановка. Поэтому будем использовать функцию *permutations*.

Все перестановки опишем с помощью permutations(‘ПАНЕЛЬ’)

Для удобства проверки вхождения комбинаций «ЕЬ» склеим полученный кортеж символов в одну строку.

```
for x in permutations('ПАНЕЛЬ'):
    s = ''.join(x)
```

Такая запись означает, что мы берем пустой разделитель (‘’) и через него перечисляем все строки из x.

Если, например, вместо пустой строки взять значение ‘2’, то получим такой результат (для одной из строк).

```
s = '2'.join(x) # П2А2Н2Е2Л2Ь2
```

Теперь нужно проверить условие, чтобы слово не начиналось с мягкого знака.

```
s[0] != 'Ь'
```

А также в слове не должна встречаться комбинация букв ‘ЕЬ’, поэтому дописываем в условие

```
'ЕЬ' not in s
```

Оператор in в питоне это проверка вхождения в строку. Если написать оператор not in, то он проверит, что комбинации ‘ЕЬ’ в строке нет.

```
from itertools import *

k = 0
for x in permutations('ПАНЕЛЬ'):
    s = ''.join(x)
    if s[0] != 'Ь' and 'ЕЬ' not in s:
        k = k + 1
print(k)
```

Ответ 480

Задание №5 (1955)

Алексей составляет 5-буквенные слова из букв М, А, Г, И, С, Т, Р. Каждую букву можно использовать не более одного раза, при этом в слове нельзя использовать более одной гласной. Сколько различных кодов может составить Алексей?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241179?t=54m40s

Решение

Если набор букв строго ограничен, то это перестановка.

Заметим, что перестановка пятибуквенная, а букв можно использовать 7. То есть получается букв больше, а длина слов меньше. С такой задачей permutation тоже справляется.

Для удобства проверки на количество гласных заменим И на А (так как для всех функций модуля itertools одинаковые буквы на разных позициях считаются разными).

После такой замены проверить, что в слове не более 1 гласной можно с помощью условия

```
s.count('A')<=1
```

Полный код программы

```
from itertools import *

k = 0
for x in permutations('МАГАСТР',5):
    s = ''.join(x)
    if s.count('A')<=1:
        k += 1
print(k)
```

Ответ: 1320

Задание №6 (1961)

Петя составляет семибуквенные слова перестановкой букв слова ТРАТАТА. Сколько всего различных слов может составить Петя?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241179?t=1h

Решение

Так как речь в задаче про перестановки, то будем использовать функцию permutations.

Попробуем решить как обычно

```
from itertools import *

k = 0
for x in permutations('ТРАТАТА'):
    k += 1
print(k)
```

Получим ответ 5040. И это будет неправильный ответ.

Давайте разберемся почему. Для этого выведем первые 200 перестановок и проанализируем, есть ли среди них повторяющиеся.

Например, найдем комбинации (‘T’,’P’,’A’,’T’,’A’,’T’,’A’)

Заметим, что их несколько. Но по условию задания должна быть только одна такая комбинация.

Чтобы убрать повторяющиеся слова используем структуру set. set — это множество, то есть, другими словами, набор различных уникальных элементов. Множество содержит только различные элементы. Если в него попробовать положить какие-то одинаковые, например, цифры, будет положена только одна, а остальные имеющие то же самое значение просто пропадут. В множестве элементы не упорядочены.

Поэтому, если перестановки с повторяющимися значениями перевести в множество, то из всех перестановок остаются только уникальные, то есть те, которые отличаются друг от друга.

```
from itertools import *

k = 0
for x in set(permutations('TPATATA')):
    k += 1
print(k)
```

Получаем ответ 140.

В этой задаче нельзя заменять буквы, например, букву **‘P’** поменять на букву **‘T’**, как это делали во всех прошлых задачах, потому что при этой замене изменится общее количество комбинаций. После замены буквы, изменяется количество уникальных символов, следовательно, и различных перестановок станет меньше.

Задание №7 (4232)

Лера составляет 5-буквенные слова из букв Л, О, Г, А, Р, И, Ф, М, причём никакие две гласные или две согласные не должны стоять рядом. Буквы в слове не должны повторяться. Сколько различных слов может составить Лера?

Ссылка на видео-разбор с таймингами: https://vk.com/video-205546952_456241179?t=1h10m55s

Решение

Набор букв жестко ограничен, буквы можно брать по одной. И из них, составляются пятибуквенные слова. Значит это перестановка.

Так как в слове все буквы разные, то можно заменить в слове ЛОГАРИФМ все согласные на одну из них (например, Л), гласные – на одну из гласных (например, О). Такая замена позволит удобнее проверять наличие двух подряд идущих гласных или двух подряд идущих согласных (ОО и ЛЛ нет в строке).

```
from itertools import *

k = 0
for x in permutations('ЛОЛОЛОЛЛ',5):
    s = ''.join(x)
    if 'ЛЛ' not in s and 'ОО' not in s:
        k = k + 1
print(k)
```

Ответ: 480

Проверить есть в слове комбинации ‘ЛЛ’ или ‘ОО’ или нет можно и через count:

```
if s.count('ЛЛ') == 0 and s.count('ОО') == 0:
```

Задача №8 (4235)

Лена составляет 5-буквенные слова из букв Я, С, Н, О, В, И, Д, Е, Ц, причём слово должно начинаться с согласной и заканчиваться гласной. Первая и последняя буквы слова встречаются в нем только один раз; остальные буквы могут повторяться. Сколько различных слов может составить Лена?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241179?t=1h16m30s

Решение

Встает вопрос, что использовать product или permutations. Так как вторая и последующие буквы могут повторяться, используем product.

Начало программы стандартное

```
from itertools import *

k = 0
```

```
for a1,a2,a3,a4,a5 in product('ЯСНОВИДЕЦ',repeat=5):
    s = a1+a2+a3+a4+a5
```

Первое условие: первая буква согласная и последняя – гласная будет выглядеть так:

```
a1 in 'СНВДЦ' and a5 in 'ЯОИЕ'
```

Второе условие: первая и последняя буквы слова встречаются в нем только один раз

```
s.count(a1)==1 and s.count(a5)==1
```

Использовать замену нельзя, потому что нужно посчитать, что первая буква ровно один раз и последняя ровно один раз. Это значит, что нужно сохранить оригинальные буквы, чтобы посчитать количество этих букв.

Объединяем условия в одно и считаем количество подходящих слов.

```
from itertools import *

k = 0

for a1,a2,a3,a4,a5 in product('ЯСНОВИДЕЦ',repeat=5):
    s = a1+a2+a3+a4+a5
    if a1 in 'СНВДЦ' and a5 in 'ЯОИЕ' and s.count(a1)==1 and s.count(a5)==1:
        k += 1
print(k)
```

Ответ: 6860.

Задание №8 (6342)

Добрыня составляет коды из букв, входящих в слово ДОБРЫНЯ. Код должен состоять из 6 букв, буквы в коде не должны повторяться, согласных в коде должно быть больше, чем гласных, две гласные буквы нельзя ставить рядом. Сколько различных кодов может составить Добрыня?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241179?t=1h23m20s

Решение

Так как буквы не должны повторяться, нужно использовать permutations. Так как все буквы в слове разные, удобно применить замену. Заменим все согласные на Д, а гласные на О.

```
for x in permutations('ДОДДОДО',6):
```

Количество перестановок от этого не поменяется, оно будет таким же.

Первое условие: согласных должно быть больше чем гласных.

```
s.count('Д')>s.count('О')
```

Второе условие: две гласные не ставятся рядом

```
'ОО' not in s
```

Объединяем все в одну программу

```
from itertools import *

k = 0
for x in permutations('ДОДДОДО',6):
    s = ''.join(x)
    if s.count('Д')>s.count('О') and 'ОО' not in s:
        k += 1
print(k)
```

Ответ: 1440

Задание №9 (6817)

Сколько существует шестнадцатеричных трёхзначных чисел, в которых все цифры различны и никакие две чётные или две нечётные цифры не стоят рядом?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241179?t=1h26m55s

Решение

По условию задачи все цифры различны. Работаем с перестановками, значит применим permutations.

Нужно помнить о том, что ноль не может стоять в начале числа.

```
for x in permutations('0123456789ABCDEF', 3):
    s = ''.join(x)
```

Так как в данном задании нам необходимо отличать только четные/нечетные и 0, можно в строке оставить только три различных символа. Например, 0, 1 и 2.

```
for x in permutations('0121212121212121', 3):
    s = ''.join(x)
```

Первое что нужно проверить это то, что s[0], то есть первая цифра в числе не равна 0

Теперь заменим 0 на 2 (или наоборот, тогда будем проверять пары 00) и проверим отсутствие пар 11 и 22, чтобы проконтролировать отсутствие двух четных или двух нечетных цифр, стоящих подряд.

```
'22' not in s and '11' not in s
```

Полный код программы

```
from itertools import *

k = 0

for x in permutations('0121212121212121', 3):
    s = ''.join(x)
    if s[0]!='0':
        s = s.replace('0','2')
        if '22' not in s and '11' not in s:
            k += 1
print(k)
```

Ответ: 840.

Задание №10 (7171)

Все 4-буквенные слова, составленные из букв Б, Ю, У, О, Ф, Ц, Ж, записаны в алфавитном порядке и пронумерованы. Вот начало списка:

1. ББББ
2. БББЖ
3. БББО
4. БББУ
5. БББФ
6. БББЦ

Сколько существует слов в списке с чётными номерами, которые начинаются на буквы ЖО?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241179?t=1h34m25s

Решение

Заметим, что букв 7, а начало списка приведено для первых 6 слов. То есть в последнем столбце можно найти не все отсортированные буквы.

Так как в слове нет Ё, то для перебора слов в алфавитном порядке можно использовать комбинацию product + sorted.

Заведем переменную k для хранения номера слова.

```
from itertools import *

k = 0
for a1,a2,a3,a4 in product(sorted('БЮУОФЦЖ'),repeat=4):
    k = k + 1
```

Проверка четности номера

```
k%2==0
```

Проверка начала на ЖО

```
a1+a2=='ЖО'
```

Для подсчета подходящих слов заведем еще один счетчик k2.

```
from itertools import *

k = 0
k2 = 0

for a1,a2,a3,a4 in product(sorted('БЮУОФЦЖ'),repeat=4):
    k = k + 1
    if k%2==0 and a1+a2=='ЖО':
        k2 = k2 + 1
print(k2)
```

Ответе: 25.

Задача №11 (7167)

Все 6-буквенные слова, составленные из букв Ж, Ю, Я, У, З, Ч, Д, О, Ф, записаны в алфавитном порядке и пронумерованы. Вот начало списка:

1. ДДДДДД
2. ДДДДДЖ
3. ДДДДДЗ
4. ДДДДДО
5. ДДДДДУ

6. ДДДДДФ

Под каким номером в списке стоит первое слово с нечётным номером, которое не начинается и не заканчивается буквой У, при этом содержит две буквы Ю, стоящие рядом?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241179?t=1h39m15s

Решение

Для поддержания алфавитного порядка и нумерации слов воспользуемся подходом из решения предыдущего задания.

```
from itertools import *

k = 0
for a1,a2,a3,a4,a5,a6 in product(sorted('ЖЮЯУЗЧДОФ'),repeat=6):
    s = a1+a2+a3+a4+a5+a6
    k = k + 1
```

Проверка «нечетный номер слова»

```
k%2!=0
```

Проверка «не начинается и не заканчивается буквой У»

```
a1!='У' and a6!='У'
```

Проверка «содержит две Ю подряд»

```
'ЮЮ' in s
```

После нахождения первого совпадения прервем цикл, так как будет найдено искомое слово

Полный код программы

```
from itertools import *

k = 0
for a1,a2,a3,a4,a5,a6 in product(sorted('ЖЮЯУЗЧДОФ'),repeat=6):
    s = a1+a2+a3+a4+a5+a6
    k = k + 1
    if k%2!=0 and a1!='У' and a6!='У' and 'ЮЮ' in s:
```

```
        print(k,s)
        break
```

Ответ: 71