

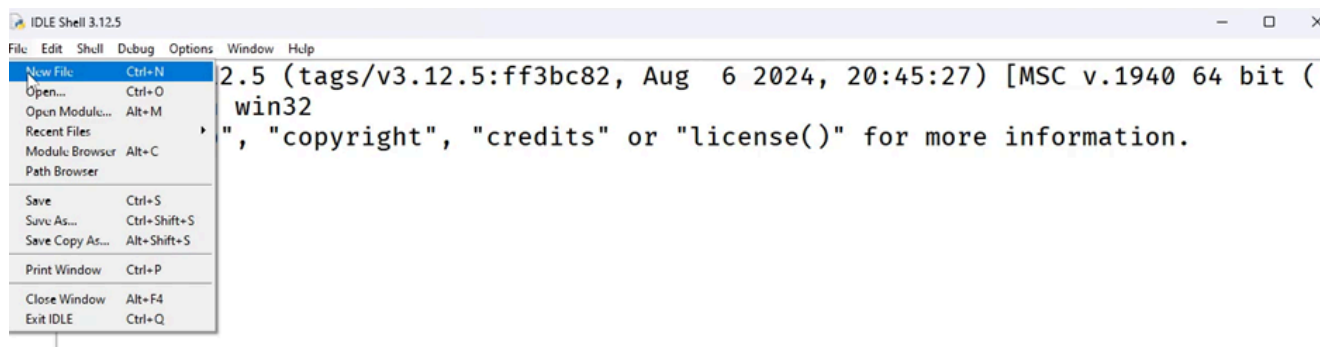
На этом занятии разберемся:

- как организовать перебор комбинаций
- как организовать подсчет подходящих комбинаций
- с основными командами (вывод на экран, проверка условий, простые логические выражения)

Будем использовать IDLE, которым мы уже пользовались, используя терминал для вычислений, когда разбирали 11, 4 задания.

Однако в терминале категорически не рекомендуется писать полноценные многострочные программы, потому что терминал создан для мгновенных однострочных действий. Поэтому программы мы будем писать немножечко по-другому.

Итак, нам нужно создать новый файл. Заходим в меню, «Файл», выбираем пункт «Новый файл».



Мы видим, что у нас открылся редактор кода. Именно в нем мы будем писать программы, которые потом будем запускать.

Первое, с чего мы начнем, это с команды вывода в консоль. Если мы напишем в терминале, то сразу получим ответ. В программе так не работает. Нам нужно использовать специальную команду для того, чтобы выводить значение на экран. И это команда `print`.

Команда `print` выводит что-то на экран. Это могут быть числа, выражения арифметические, строки, всё, что угодно. `Print` означает «выведи на экран», «выведи в консоль».

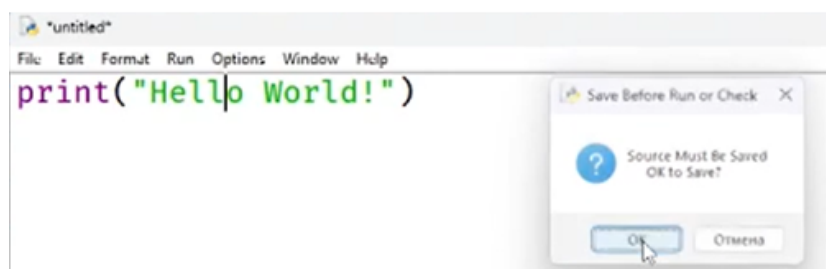
Давайте, например, выведем надпись «Hello, world!». Стоит отметить, что «Hello, world!» — это строка, записанная в кавычках. В данном случае, это

строка с текстом. Кавычки могут быть одинарные и двойные. Важно, чтобы они были одинаковые.

```
print("Hello, world!")
```

Чтобы запустить программу мы либо используем меню Run -> Run Module, либо просто нажимаем кнопку F5 (это кнопка запуска программы).

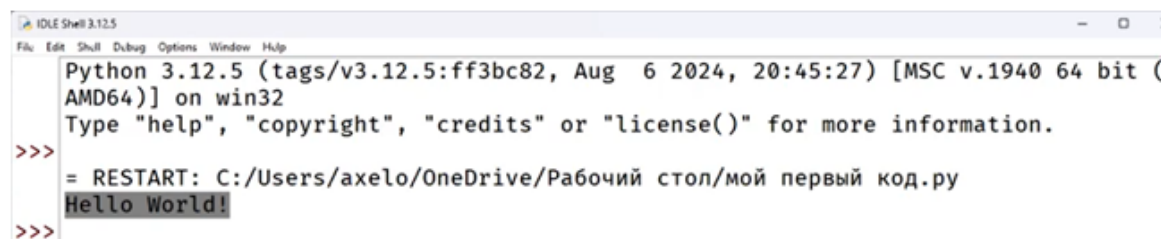
Нажав на кнопку, среда предлагает сохранить файл.



Мы нажимаем ОК и сохраняем ее куда-нибудь в понятное, удобное нам место. Советую сделать отдельную папочку, в которой вы будете сохранять написанный вами код, чтобы он не потерялся и, чтобы можно было потом его посмотреть. Я сохраню файл на рабочий стол. И дам ему имя «Мой первый код».

На экзамене я советую все программки сохранять именно на рабочий стол, потому что это самый быстрый доступ.

Нажимаем «сохранить», и, видим, что появилась надпись, Hello World.



Можно выводить не только надпись, но и, например, выражение. Запишем его: $2+2*2$ и запустим.

```
print(2+2*2)
```

Получаем результат выражения: 6.

Итак, команда **print** - это команда вывода на экран. Мы будем ее использовать, как для вывода ответа, так и для вывода промежуточных значений.

Для решения 8 задания нам также нужно разобраться с тем, как нужно организовывать перебор всевозможных слов. Для этого нам нужна алгоритмическая конструкция - **цикл for**. Это специальная конструкция, которая перебирает по очереди различные элементы последовательности.

Давайте разберемся на примере: *for x in 0,1,2,3,4,5,6,7:*

Эта запись означает, что мы перебираем по очереди элементы из последовательности 0, 1, 2, 3, 4, 5, 6, 7. Эти элементы по очереди записываются в переменную *x*.

Переменные - это место, ячейка в памяти, в которой хранятся какие-то ваши данные, которые программа использует в данный момент.

Переменные можно называть так, как вам удобно: буквами, цифрами, полноценными английскими словами. Главное, не начинать с цифры. Можно использовать нижнее подчеркивание.

В данном случае этой переменной выступает *x*, которая будет хранить по очереди элементы последовательности. Сначала 0, потом 1, потом 2, потом 3, потом 4, потом 5 и так далее. Последовательно, по очереди перебирая, мы можем с ними что-то делать: выводить на экран, преобразовывать, делать какие-то дополнительные действия или вообще ничего не делать.

Примечание Джобса: в Python реализована модель наименования объектов и понятия «переменная» в классическом смысле нет. Однако для удобства на начальных этапах можно считать, что переменная – это ячейка памяти (пока не дошли до изменения множеств, списков и словарей).

Если мы используем цикл *for*, в конце обязательно ставим двоеточие. Двоеточие означает, что у этой команды есть какие-то вложенные действия, которые будут выполняться у нее внутри. Потому что мы в *x* не просто перебираем значения, мы хотим с ними что-то делать. И именно эти действия мы записываем после двоеточия.

```
for x in 0, 1, 2, 3, 4, 5, 6, 7:
```

Важно: Python, в отличие от других языков, вложенность команд (когда что-то находится внутри чего-то) обозначается с помощью отступов. Если после двоеточия нажатии Enter, появится новый уровень вложенности – по умолчанию отступ в четыре пробела.

Если вы сделали цикл, перешли на новую строку и заметили, что у вас нет отступа, значит, вы забыли поставить двоеточие.

Если вы переходите на новую строку, и видите большой отступ (больше 4 пробелов), это тоже говорит об ошибке. Обычно это обозначает, что вы не закрыли скобки в выражении. Однако, об этом позже.

Итак, отступ – это сигнал о том, что сейчас я пишу команды, которые будут выполняться внутри этого цикла for. То есть действия, которые будут написаны в этой строке, будут выполняться над переменной x для каждого нового значения.

Например, напомним print(x). Это значит, что будут выводиться по очереди значения x на экран.

```
for x in 0, 1, 2, 3, 4, 5, 6, 7:  
    print(x)
```

Запустив программу, мы получим по очереди значения 0, 1, 2, 3, 4, 5, 6, 7.

```
>>>  
0  
1  
2  
3  
4  
5  
6  
7
```

С x можно делать какие-то действия. Например, выводить степени двойки.

```
for x in 0, 1, 2, 3, 4, 5, 6, 7:  
    print(2**x)
```

Запустив программу, мы видим, что значение x подставляется в выражение и вычисляется степень двойки. Результат выводится на экран.

```
>>>  
1
```

2
4
8
16
32
64
128

Действия, которые нужно выполнить для перебираемых значений, записываются с одинаковым отступом. Отступ показывает, что команда находится внутри цикла, то есть именно эта команда (или несколько команд) будут повторяться.

Цикл можно применять при переборе слов.

Задание 1. (211)

Сколько различных слов длины 4, начинающихся с согласной буквы и заканчивающихся гласной буквой, можно составить из букв М, Е, Т, Р, О? 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_456241177?t=12m40s

Решение.

Нужно организовать перебор слов из четырёх букв. Для этого нужно организовать по отдельности перебор первой, второй, третьей, четвёртой буквы вместе.

Мы возьмём переменную для первой буквы, назовем ее «a». В кавычках указываем не числа, а буквы, которые нужно перебрать. По условию задачи нужно, чтобы слово начиналось с согласной буквы. Соответственно, это будут буквы МТР. В этом задании нужно писать именно русские буквы. Иначе будет ошибка.

```
for a in 'МТР':  
    print(a)
```

Этот цикл будет перебирать по очереди символы строки МТР.

Давайте выведем каждую букву по очереди: `print(a)`. И запустим наш код, нажав F5.

```
>>>  
М
```

Т
О

Чтобы перебрать все варианты слов длиной 2, нам нужно для каждой первой буквы «а» написать второй цикл, который будет перебирать уже вторую букву. Получается цикл в цикле.

«а» - это переменная, это название первой буквы (так сказать, место ящика, в котором хранится первая буква).

Ниже мы пишем второй цикл for b. «b» - это вторая переменная для второй буквы.

Это называется цикл в цикле или вложенный цикл. Есть первый цикл, который перебирает первую буковку. Для каждого значения первой буковки второй цикл перебирает значение второй буквы.

Выводим на экран, склеивая между собой «а» и «b» с помощью плюса.

```
for a in 'МТР':  
    for b in 'МЕТРО':  
        print(a+b)
```

Запускаем и получаем все возможные варианты из двух букв.

```
>>>  
ММ  
МЕ  
МЬ  
МР  
МО  
ТМ  
ТЕ  
ТЬ  
ТР  
ТО  
РМ  
РЕ  
РТ  
РР  
РО
```

Записываем следующий цикл, в котором переменной назначаем букву «с».

Делаем внутри второго цикла третий цикл. И перебираем все буквы: МЕТРО

Теперь мы имеем тройной цикл. Перебираются всевозможные связки первой, второй, третьей буквы.

Выводим на экран a+b+c. То есть выводим на экран тройки букв.

```
for a in 'МТР':  
    for b in 'МЕТРО':
```

```
for c in 'МЕТРО':  
    print(a+b+c)
```

Вариантов становится все больше и больше. Мы получаем всевозможные подходящие тройки.

```
>>>  
= RESTART:  
MMM  
MMEI  
MMT  
MMP  
MMO  
MEM  
MEE  
MET  
MFD
```

Теперь последняя четвертая буква. Она заканчивается гласной, по условию задачи берем только буквы ЕО.

Для четвертого цикла напишем переменную «d». И получим четверной цикл, который перебирает все четырехбуквенные слова.

```
for a in 'МТР':  
    for b in 'МЕТРО':  
        for c in 'МЕТРО':  
            for d in 'ЕО':  
                print(a+b+c+d)
```

Выводим и получаем, слова, которые подходят под условия задачи, так как все они начинаются с согласной и заканчиваются на гласную букву.

Теперь нужно посчитать количество этих слов. Создаем в самом начале программы (до перебора) переменную k, которая будет считать количество подходящих слов. Изначально эта переменная равна 0.

Каждый раз, когда мы находим одно подходящее слово, счетчик k увеличиваем на единицу.

Чтобы вывести количество подошедших слов после перебора всех комбинаций, нужно написать команду print на уровне первого цикла (без отступов).

```
k = 0  
for a in 'МТР':  
    for b in 'МЕТРО':  
        for c in 'МЕТРО':  
            for d in 'ЕО':  
                k = k + 1
```

```
print(k)
```

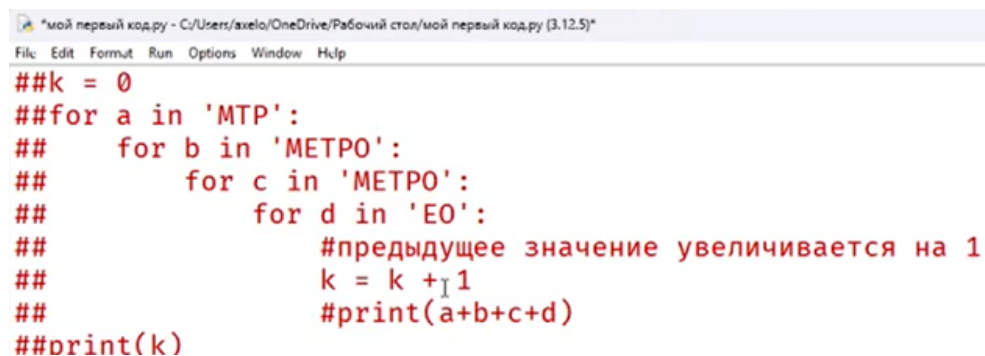
Ответ: 150

Работа с комментариями

Одиночные комментарии – строки, которые не будут выполняться – определяются с помощью знака `#`, записываемого в начале строки. В таких комментариях мы можем писать пояснения, аннотации или временно убирать какой-то код. Также мы можем комментировать сразу большое количество строк. Для этого нужно использовать комбинацию `Alt+3`.

Допустим, вы написали код, он закоментируется выделенные строки.

Чтобы раскомментировать, нажимаем комбинацию `Alt+4`.



```
*мой первый код.py - C:/Users/axelo/OneDrive/Рабочий стол/мой первый код.py (3,12.5)*
File Edit Format Run Options Window Help
##k = 0
##for a in 'MTP':
##    for b in 'МЕТРО':
##        for c in 'МЕТРО':
##            for d in 'ЕО':
##                #предыдущее значение увеличивается на 1
##                k = k + 1
##                #print(a+b+c+d)
##print(k)
```

Почему $k + 1$?

Потому что значение предыдущее, которое было, увеличивается на единицу.

Напишу это пояснение в комментарий в код:

```
k = 0
for a in 'MTP':
    for b in 'МЕТРО':
        for c in 'МЕТРО':
            for d in 'ЕО':
                #предыдущее значение увеличивается на 1
                k = k + 1

print(k)
```

Другими словами, я загибаю пальчик: было 0 слов, стало 1, потом 2 слова, потом 3, 4, 5 слов и так далее. Тем самым мы считаем общее количество подходящих слов, загибая пальчик.

«А мы могли написать `for a in range(8)?`»

«range» перебирает числа. То есть, 0, 1, 2, 3, 4, 5, 6, 7. Если речь про самый первый цикл из лекции, то могли (первый цикл на картинке).

```
for x in 0, 1, 2, 3, 4, 5, 6, 7:  
    print(x)
```

Тоже самое

```
for x in range(8):  
    print(x)
```

Задание 2. (1886)

Игорь составляет 8-буквенные коды из букв И, Г, О, Р, Ъ. Буквы О и Ъ должны встречаться в коде ровно по одному разу, при этом буква Ъ не может стоять на первом месте. Остальные допустимые буквы могут встречаться произвольное количество раз или не встречаться совсем. Сколько различных кодов может составить Игорь?

Ссылка на видео-разбор с тайм-кодом: https://vk.com/video-205546952_456241177?t=26m5s

Решение.

Нам нужно перебрать восьмибуквенные слова из букв И, Г, О, Р, Ъ и разобраться с тем, что буквы О и Ъ встречаются ровно по одному разу.

С помощью вложенных циклов, мы переберем все возможные буквенные комбинации.

Введу переменную «a1», которая обозначает первую букву. Перебираем буквы «ИГОР», ведь Ъ, по условию, не может быть на 1 месте.

```
for a1 in 'ИГОР':
```

Обязательно поставим двоеточие.

Перебираем вторую букву. И даем переменную «a2». Теперь перебираем все 5 букв.

```
for a1 in 'ИГОР':  
    for a2 in 'ИГОРЬ':
```

Здесь можно схитрить, скопировать – вставить цикл, поменяв только имена переменных – a3, a4 и так далее.

```
for a1 in 'ИГОР':
    for a2 in 'ИГОРЬ':
        for a3 in 'ИГОРЬ':
            for a4 in 'ИГОРЬ':
                for a5 in 'ИГОРЬ':
                    for a6 in 'ИГОРЬ':
                        for a7 in 'ИГОРЬ':
                            for a8 in 'ИГОРЬ':
```

Получается 8 вложенных друг к другу циклов, которые по очереди перебирают все возможные буквенные комбинации длины 8.

Однако, не все буквенные комбинации нам подходят, ведь буквы О и Ъ должны встречаться в слове только по одному разу. Значит, мы вводим дополнительные условия.

Мы можем отсеять те слова, которые нам подходят, используя условный оператор. Это вторая важная алгоритмическая конструкция.

Для удобства соберем все буквы в одну строку.

```
for a1 in 'ИГОР':
    for a2 in 'ИГОРЬ':
        for a3 in 'ИГОРЬ':
            for a4 in 'ИГОРЬ':
                for a5 in 'ИГОРЬ':
                    for a6 in 'ИГОРЬ':
                        for a7 in 'ИГОРЬ':
                            for a8 in 'ИГОРЬ':
                                s = a1+a2+a3+a4+a5+a6+a7+a8
```

Мы можем вывести эти комбинации на экран и увидеть, что их много.

```
for a1 in 'ИГОР':
    for a2 in 'ИГОРЬ':
        for a3 in 'ИГОРЬ':
            for a4 in 'ИГОРЬ':
                for a5 in 'ИГОРЬ':
                    for a6 in 'ИГОРЬ':
                        for a7 in 'ИГОРЬ':
                            for a8 in 'ИГОРЬ':
                                s = a1+a2+a3+a4+a5+a6+a7+a8
                                print(s)
```

Чтобы не ждать, когда завершится, нажимаем Ctrl+C. Программа говорит, что зависла. Лучше было просто закрыть окно. Чтобы вернуться к уже написанному

коду, используем в меню файл - «open» и открываем наш код там, где мы его сохраняли.

Теперь мы знаем, что выводить «s» на экран — это плохая идея, потому что вариантов очень много.

Слова, в которых ровно одна буква О и ровно один Ь. Такие слова мы можем отобрать с помощью условного оператора. Начинается он с кодового слова «if» (if - это «если» по-английски).

Здесь мы прописываем условие, что должно быть для того, чтобы это слово мы засчитали.

Для этого действия, для подсчета количества, мы можем использовать **метод COUNT**.

Когда мы применяем метод COUNT к какой-то строке S, то он возвращает нам количество букв или непересекающихся буквосочетаний. Например, когда мы напишем `s.count('O')`, то возвратится количество букв 'O', которое содержится в данной строке.

Итак, если количество букв О равно единице (`s.count('O') == 1`) и количество букв Ь равно единице (`s.count('B')==1`), то слово нам подходит. Данная операция называется операцией сравнения (проверки равенства) и обозначается двумя знаками равно. В то время как один знак равно записывает значения (*прим.Джобса*: устанавливает имя для объекта).

Чтобы последовательно вывести слова на экран (и чтобы программа не зависла) будем использовать **команду input**. Это команда ввода, но ее можно использовать для паузы.

```
for a1 in 'ИГОР':
    for a2 in 'ИГОРЬ':
        for a3 in 'ИГОРЬ':
            for a4 in 'ИГОРЬ':
                for a5 in 'ИГОРЬ':
                    for a6 in 'ИГОРЬ':
                        for a7 in 'ИГОРЬ':
                            for a8 in 'ИГОРЬ':
                                s = a1+a2+a3+a4+a5+a6+a7+a8
                                if s.count('O')==1 and s.count('B')==1:
                                    print(s)
                                    input()
```

Запустив, у нас появится первое слово, программа остановится и будет ждать до тех пор, пока мы не нажмем Enter. После нажатия появляется второе слово. Нажимаем – появляется третье и так далее.

Во всех словах, что вывелись мы видим одну букву О и один Ь.

ИИИИИИОЬ

ИИИИИИЬОІ

ИИИИИГОЬ

ИИИИИГЬО

ИИИИИОИЬ

ИИИИИОГЬ

ИИИИИОРЬ

Нам нужно посчитать, сколько слов восьмибуквенных, в которых ровно одна буква О и ровно один Ь. В самом начале, еще перед перебором, создаем переменную k, равную нулю. Внутри условия, для каждого подходящего слова, увеличиваем k на единицу.

После выполнения всех команд и условий (без отступов) мы выводим ответ на задачу.

```
k = 0
for a1 in 'ИГОР':
    for a2 in 'ИГОРЬ':
        for a3 in 'ИГОРЬ':
            for a4 in 'ИГОРЬ':
                for a5 in 'ИГОРЬ':
                    for a6 in 'ИГОРЬ':
                        for a7 in 'ИГОРЬ':
                            for a8 in 'ИГОРЬ':
                                s = a1+a2+a3+a4+a5+a6+a7+a8
                                if s.count('О')==1 and s.count('Ь')==1:
                                    k = k + 1
print(k)
```

Ответ: 35721

Задание 3. (1892)

Ваня составляет четырехбуквенные слова из букв О, Б, Ъ, Е, М, причём в каждом слове буква О встречается ровно один раз, а буква Ъ не может стоять на первом месте и не может стоять на последнем месте. Все остальные буквы, могут встречаться в слове любое количество раз или не встречаться совсем.

Словом считается любая допустимая последовательность букв, не обязательно осмысленная. Сколько существует таких слов, которые может написать Ваня?

Ссылка на видео-разбор с тайм-кодом: https://vk.com/video-205546952_456241177?t=38m5s

Решение

Начнем перебор с первой буквы и назовем переменную «petya». Также обратим внимание на то, что Ъ не может стоять на первом месте.

```
for petya in 'ОБЕМ':
```

Перебираем вторую букву, которая будет храниться в переменной под названием «vanya». Третью переменную назовем «lena». Четвертую – «dasha».

Использовать кириллические переменные нельзя, так как в IDLE будет выдаваться ошибка.

```
for petya in 'ОБЕМ':  
    for vanya in 'ОБЪЕМ':  
        for lena in 'ОБЪЕМ':  
            for dasha in 'ОБЕМ':
```

Склеим эти буквы в супер-трансформера. Ведь переменные вы можете называть так, как вам удобно. Буквы, которые мы кодируем, в данном случае О,Б,Ъ,Е,М можно писать строчными (маленькими) буквами. Главное потом и в условии писать тоже строчными.

```
for petya in 'ОБЕМ':  
    for vanya in 'ОБЪЕМ':  
        for lena in 'ОБЪЕМ':  
            for dasha in 'ОБЕМ':  
                transformer = petya + vanya + lena + dasha
```

Создадим счетчик и назовем его count. Когда мы находим слово, в котором ровно одна буква О, мы count увеличиваем на единицу.

Вспоминаем условие задачи: в каждом слове буква «О» встречается ровно один раз. Записываем операцию сравнения: Если в строке transformer одна буква О (transformer.count('O') == 1), тогда увеличиваем количество подходящих слов (count = count + 1).

После перебора всех комбинаций, после выполнения всех команд выводим на экран. Обязательно убираем все отступы.

```
count = 0
for petya in 'ОБЕМ':
    for vanya in 'ОБЪЕМ':
        for lena in 'ОБЪЕМ':
            for dasha in 'ОБЕМ':
                transformer = petya + vanya + lena + dasha
                if transformer.count('О')==1:
                    count = count + 1
print(count)
```

Получаем ответ: 168

Задание 4. (1916).

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

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241177?t=46m35s

Решение

Перебираем все четырёхбуквенные слова. Возьмем первую переменную a1. И обратим внимание на то, что буква Ё не должна стоять на первом месте, значит, перебираем буквы ГАФНИ.

Вторая переменная будет называться a2, в ней перебираем буквы ГАФНИЙ. По аналогии кодируем третью и четвертую буквы. После чего склеим их в одно слово.

```
for a1 in 'ГАФНИ':
    for a2 in 'ГАФНИЙ':
        for a3 in 'ГАФНИЙ':
            for a4 in 'ГАФНИЙ':
                s = a1+a2+a3+a4
```

Код должен содержать хотя бы одну гласную. Мы имеем 2 гласные - А и И. Значит, в слове должна быть либо одна буква А, либо одна буква И. То есть их суммарное количество должно быть не равно нулю - больше нуля.

Записываем в код:

- Если количество букв А плюс количество букв И больше нуля ($s.count('A') + s.count('И') > 0$).
- То с каждым подходящим словом счетчик увеличивается на 1.

Также это условие можно было записать, как не равно нулю ($!= 0$):

```
if s.count('A') + s.count('И') != 0:
```

или

Больше или равно 1:

```
if s.count('A') + s.count('И') >= 1:
```

Перед всем кодом обязательно добавляем переменную К для подсчета количества. После выполнения всех действий мы выводим на экран значение k.

```
k = 0
for a1 in 'ГАФНИ':
    for a2 in 'ГАФНИЙ':
        for a3 in 'ГАФНИЙ':
            for a4 in 'ГАФНИЙ':
                s = a1+a2+a3+a4
                if s.count('A') + s.count('И') > 0:
                    k = k + 1
print(k)
```

Ответ: 888

Задание 5. (7545)

(ЕГЭ-2024) Определите количество 15-ричных пятизначных чисел, в записи которых ровно одна цифра 8 и не менее двух цифр с числовым значением, превышающим 9.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241177?t=53m20s

Решение

Используются 15 цифр: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E. Пятизначные числа – это числа из пяти цифр, старшая (левая) из которых не равна 0.

Сразу создадим счетчик $k = 0$.

Первая цифра будет перебираться в переменной a1. Мы помним, что число не может начинаться с нуля. Поэтому переберем значения 1, 2, 3, 4, 5, 6, 7, 8, 9, а, b, c, d, e.

Вторая, третья, четвертая и пятая цифры уже могут быть нулем, значит, мы перебираем: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E.

```
k = 0
for a1 in '123456789abcde':
    for a2 in '0123456789abcde':
        for a3 in '0123456789abcde':
            for a4 in '0123456789abcde':
                for a5 in '0123456789abcde':
                    s = a1+a2+a3+a4+a5
```

Если в этой строке количество цифр 8 равно единице ($s.count('8') == 1$), и сумма количества цифр в числе A, B, C, D, E больше или равно 2 ($s.count('a')+s.count('b')+s.count('c')+s.count('d')+s.count('e') \geq 2$), то это число нам подходит, и мы ставим счетчик $k + 1$ после перебора всех чисел.

```
k = 0
for a1 in '123456789abcde':
    for a2 in '0123456789abcde':
        for a3 in '0123456789abcde':
            for a4 in '0123456789abcde':
                for a5 in '0123456789abcde':
                    s = a1+a2+a3+a4+a5
                    if s.count('8')==1 and
s.count('a')+s.count('b')+s.count('c')+s.count('d')+s.count('e')>=2:
                        k = k + 1
print(k)
```

Ответ: 83175

Задание 6 (7462).

(ЕГЭ-2024) Сколько существует чисел, девятеричная запись которых состоит из пяти цифр, содержит ровно один ноль, причём ни одна нечётная цифра не стоит рядом с нулём?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241177?t=1h2m20s

Решение

Ни слева от нуля, ни справа от нуля нет нечетной цифры. Нечетных цифр несколько – 1, 3, 5, 7. Поэтому нужно сделать много проверок – для каждой цифры и слева, и справа. Мы можем количество этих проверок сократить, если предварительно поменять цифры.

Для начала переберем все пятицифровые числа.

Создадим переменную, в которой буду считать количество подходящих чисел, то есть счетчик $k=0$.

Перебираем первую циферку a_1 – с 1 до 8 (с нуля число не может начинаться).

Перебираем вторую, третью, четвертую и пятую цифры уже с 0. После чего соберем число в одну строку.

```
k = 0
for a1 in '12345678':
    for a2 in '012345678':
        for a3 in '012345678':
            for a4 in '012345678':
                for a5 in '012345678':
                    s = a1+a2+a3+a4+a5
```

.

Не должно быть таких комбинаций, где слева или справа с нулем будет стоять нечетная цифра. Такие комбинации: 10, 30, 50, 70 или 01, 03, 05, 07. Мы можем проверить каждую из них – 8 штук. Но можно пойти хитрее.

В этом задании не так важны значения цифр. Важен признак цифры, что она нечетная. Давайте схитрим и цифры 3, 5 и 7 заменим на единицу в строке.

Важно заменить именно на нечетную цифру.

Запись поменяется, но все нечётные циферки будут просто обозначаться единицей. И вместо 8 разных комбинаций надо будет проверить только 2. То есть мы делаем замену для того, чтобы количество проверяемых комбинаций уменьшилось до адекватного количества.

Чтобы сделать замену, мы применяем специальный метод строки **replace**, который заменяет одно на другое.

Допустим, у вас есть строчка 12345. Я вызываю команду $s = s.replace('3', '1')$, которая заменит цифру 3 на цифру 1. Все цифры 3 поменяются на единицы.

```
s = '12345'
s = s.repalce('3', '1')
print(s)
>>> 12145
```

Количество чисел при такой замене не меняется, так как оно зависит от циклов `for in`. В циклах все цифры разные, поэтому перебираются разные числа. Замена делается уже в конечном числе.

Заменим все нечетные цифры на 1

```
s = s.replace('3','1').replace('5','1').replace('7','1')
```

Тройки, пятерки, семерки превратились в единицы. Теперь вместо восьми комбинаций мы проверим, что в числе нет комбинации 01 и 10 с помощью `count`.

```
s.count('10')==0 and s.count('01')==0
```

Также, количество нулей в числе должно быть равно 1 (по условию задачи). Прописываем это условие через `count` в ту же строку.

```
s.count('0')==1
```

Получается условие: Если в числе есть только один ноль, и рядом с ним нет нечетного числа слева и нет нечетного справа, то это число нам подходит, и счетчик `k` увеличивается на 1.

```
k = 0
for a1 in '12345678':
    for a2 in '012345678':
        for a3 in '012345678':
            for a4 in '012345678':
                for a5 in '012345678':
                    s = a1+a2+a3+a4+a5
                    s = s.replace('3','1').replace('5','1').replace('7','1')
                    if s.count('0')==1 and s.count('10')==0 and s.count('01')==0:
                        k = k + 1
print(k)
```

Ответ: 5120

Задание 7. (4234)

(А. Куканова) Василиса составляет 5-значные числа в 6-ичной системе счисления. Цифры в числе могут повторяться, но никакие две четные или две нечетные цифры не должны стоять рядом. Сколько чисел может составить Василиса?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241177?t=1h19m40s

Решение 1

Можно рассмотреть два случая. ЧНЧНЧ и НЧНЧН.

ЧНЧНЧ

Первая цифра: 2, 4, Вторая и четвертая – 1, 3, 5. Третья и пятая – 0, 2, 4.

НЧНЧН

Первая, третья и пятая цифры: 1, 3, 5, Вторая и четвертая – 0, 2, 4.

```
#1 способ
k = 0
for a1 in '24':
    for a2 in '135':
        for a3 in '024':
            for a4 in '135':
                for a5 in '024':
                    k = k + 1

for a1 in '135':
    for a2 in '024':
        for a3 in '135':
            for a4 in '024':
                for a5 in '135':
                    k = k + 1

print(k)
```

Ответ: 405

Решение 2

Переберем все цифры на всех позициях (кроме 0 на первой).

После чего в получившемся числе заменим все четные цифры на 0, все нечетные – на 1.

Тогда проверка, что в числе нет двух подряд идущих разрядов с одинаковой четностью, сведется к тому, что в строке нет комбинаций 00 и 11.

```
k = 0
for a1 in '12345':
    for a2 in '012345':
        for a3 in '012345':
            for a4 in '012345':
```

```
for a5 in '012345':
    s = a1+a2+a3+a4+a5
    s = s.replace('2','0').replace('4','0')
    s = s.replace('3','1').replace('5','1')
    if s.count('00')==0 and s.count('11')==0:
        k = k + 1

print(k)
```

Ответ: 405

Про порядок возрастания/убывания/невозрастания/неубывания

Возрастание – каждый элемент больше предыдущего

$$a1 < a2 < a3 < a4$$

Убывание – каждый элемент меньше предыдущего

$$a1 > a2 > a3 > a4$$

Невозрастание – каждый элемент не больше предыдущего

$$a1 \geq a2 \geq a3 \geq a4$$

Неубывание – каждый элемент не меньше предыдущего

$$a1 \leq a2 \leq a3 \leq a4$$

В Python можно записывать такие математические неравенства в явном виде, например, порядок неубывания будет определен, как

$$a1 \leq a2 \leq a3 \leq a4$$

Задание 8 (4253)

Евгения составляет 4-значные числа в 8-ичной системе счисления. Числа должны начинаться с чётной цифры, и цифры в них располагаются в невозрастающем порядке. Сколько различных чисел может составить Евгения?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241177?t=1h31m45s

Получается, что первая цифра – одна из цифр 2, 4, 6. Вторая, третья и четвертые цифры – цифры от 0 до 7.

Определим порядок неубывания с помощью множественного сравнения. И подсчитаем количество подходящих чисел.

```
k = 0
for a1 in '246':
    for a2 in '01234567':
        for a3 in '01234567':
            for a4 in '01234567':
                if a1 >= a2 >= a3 >= a4:
                    k = k + 1
print(k)
```

Ответ: 129

Почему в условии он str с str, а не int с int сравнивает?

Это самая главная прелесть, что можно и символы сравнивать между собой, потому что символы идут по порядку, то есть от 0 до 9, и нет никакой проблемы сравнивать так. Так можно сравнивать не только цифры, но и другие символы.

Задание 9 (6902)

Коля записывает восьмизначные восьмеричные числа, которые начинаются и заканчиваются чётной цифрой и по крайней мере три нечётные цифры стоят рядом. Сколько таких чисел может записать Коля?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241177?t=1h40m50s

Решение

Три нечетные цифры стоят рядом. Это могут быть любые нечетные цифры, не обязательно одинаковые. Допустим, 135 тоже годится.

Поэтому много комбинаций проверять придется. Или мы сделаем замену, и будет всего одна комбинация. Если проверять не все комбинации из нечетных цифр, а заменить каждую на один и тот же символ, то нужно будет проверить

только наличие трех подряд идущих символов, на которые была произведена замена.

Первая цифра четная – 2, 4 или 6. Последняя цифра четная – 0, 2, 4 или 6. Остальные цифры могут принимать любые допустимые значения (0-7).

```
k = 0
for a1 in '246':
    for a2 in '01234567':
        for a3 in '01234567':
            for a4 in '01234567':
                for a5 in '01234567':
                    for a6 in '01234567':
                        for a7 in '01234567':
                            for a8 in '0246':
                                s = a1+a2+a3+a4+a5+a6+a7+a8
                                # заменяем все нечетные на 1
                                s = s.replace('3','1').replace('5','1').replace('7','1')
                                # если есть три нечетных подряд
                                if s.count('111')>0:
                                    k = k + 1
print(k)
```

Ответ: 983040

Обратите внимание, довольно долго надо ждать. Но мы в конце получаем ответ. Кстати, такой ответ вполне допустим на ЕГЭ.

Задание 10 (4925)

Определите количество семизначных чисел, записанных в семеричной системе счисления, учитывая, что числа не могут начинаться с цифр 3 и 5 и не должны содержать сочетания цифр 22 и 44 одновременно.

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241177?t=1h48m17s

Решение

Запишем вложенные циклы и сформируем строку

```
k = 0
for a1 in '1246':
```

```
for a2 in '0123456':
    for a3 in '0123456':
        for a4 in '0123456':
            for a5 in '0123456':
                for a6 in '0123456':
                    for a7 in '0123456':
                        s = a1+a2+a3+a4+a5+a6+a7
```

Не должно содержать сочетание 22 и 44 одновременно.

Сначала запишем условие, что 22 и 44 встречаются одновременно.

```
s.count('22')>0 and s.count('44')>0
```

Соответственно, НЕ встречаются одновременно будет записано, как

```
not (s.count('22')>0 and s.count('44')>0)
```

Полный код программы

```
k = 0
for a1 in '1246':
    for a2 in '0123456':
        for a3 in '0123456':
            for a4 in '0123456':
                for a5 in '0123456':
                    for a6 in '0123456':
                        for a7 in '0123456':
                            s = a1+a2+a3+a4+a5+a6+a7
                            if not (s.count('22')>0 and s.count('44')>0):
                                k = k + 1
print(k)
```

Ответ: 466456

Задание 11 (215)

Все 5-буквенные слова, составленные из 5 букв А, К, Л, О, Ш, записаны в алфавитном порядке. Вот начало списка:

1. ААААА
2. ААААК
3. ААААЛ
4. ААААО
5. ААААШ
6. АААКА

...

На каком месте от начала списка стоит слово ШКОЛА?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241177?t=1h53m31s

Решение

Переберем все возможные слова, каждая буква в которых перебирается в алфавитном порядке (тогда и сами слова будут перебираться в алфавитном порядке).

```
for a1 in 'АКЛОШ':
    for a2 in 'АКЛОШ':
        for a3 in 'АКЛОШ':
            for a4 in 'АКЛОШ':
                for a5 in 'АКЛОШ':
                    s = a1+a2+a3+a4+a5
```

Чтобы добавить нумерацию создадим перед циклом переменную k со значением 0 и после формирования нового слова будем увеличивать счетчик на 1. Таким образом счетчик увеличится столько раз, сколько раз программа была в самом вложенном цикле.

И, если найдено слово ШКОЛА, выведем номер этого слова.

```
k = 0
for a1 in 'АКЛОШ':
    for a2 in 'АКЛОШ':
        for a3 in 'АКЛОШ':
            for a4 in 'АКЛОШ':
                for a5 in 'АКЛОШ':
                    s = a1+a2+a3+a4+a5
                    k = k + 1
                    if s == 'ШКОЛА':
                        print(k, s)
```

Ответ: 2711

Задание 12 (7517)

Все пятибуквенные слова, составленные из букв Ф, О, К, У, С записаны в алфавитном порядке и пронумерованы. Вот начало списка:

1. KKKKK
2. KKKKO
3. KKKKS
4. KKKKY
5. KKKKF

...

Под каким номером в списке идёт последнее слово, которое не содержит букв Ф и содержит ровно две буквы У?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241177?t=2h4m

Решение

Буквы ФОКУС в алфавитном порядке КОСУФ.

Также перебираем 5 буквенные слова и нумеруем их.

Условие: не содержит букв «Ф» и содержит ровно две буквы «У».

```
s.count('Ф')==0 and s.count('У')==2
```

Полный код программы

```
k = 0
for a1 in 'КОСУФ':
    for a2 in 'КОСУФ':
        for a3 in 'КОСУФ':
            for a4 in 'КОСУФ':
                for a5 in 'КОСУФ':
                    s = a1+a2+a3+a4+a5
                    k = k + 1
                    if s.count('Ф')==0 and s.count('У')==2:
                        print(k,s)
```

>>>

...

2307 УУСОО

2308 УУСОО

2311 УУССК

2312 УУССО

2313 УУССС

>>>

Ответ: 2313

Определение четности/нечетности/кратности

В Python есть операция нахождения остатка от деления. Если остаток от деления на k равен 0, то число кратно k . Соответственно, если остаток от деления на 2 равен 0, то число четное. Если не равен – нечетное.

Четное: $x \% 2 == 0$

Нечетное: $x \% 2 != 0$

Кратное k : $x \% k == 0$

Не кратное k : $x \% k != 0$

Задание 13 (6718)

Все пятибуквенные слова, составленные из букв К, О, М, П, Ъ, Ю, Т, Е, Р, записаны в алфавитном порядке и пронумерованы. Начало списка выглядит так:

1. EEEEE
2. EEE EK
3. EEE EM
4. EEE EO
5. EEE EP
6. EEE ER
7. EEE ET
8. EEE Ъ
9. EEE Ю

...

Под каким номером в списке стоит последнее слово с нечётным номером, которое не начинается с буквы Ъ и содержит ровно две буквы К?

Ссылка на видео-разбор с таймингом: https://vk.com/video-205546952_456241177?t=2h8m41s

Решение

Если в задаче нужно упорядочить большое количество букв (например, 9), можно использовать функцию `sorted`, которая выведет буквы слова в алфавитном порядке.

Примечание Джобса: будьте аккуратны с буквой Ё, если она есть, конечно же.

Воспользуемся использованными ранее приемами и переберем пятибуквенные слова.

```
k = 0
for a1 in sorted('КОМПЬЮТЕР'):
    for a2 in sorted('КОМПЬЮТЕР'):
        for a3 in sorted('КОМПЬЮТЕР'):
            for a4 in sorted('КОМПЬЮТЕР'):
                for a5 in sorted('КОМПЬЮТЕР'):
                    s = a1+a2+a3+a4+a5
                    k = k + 1
```

Нечетный номер будет тогда, когда остаток от деления на 2 этого номера будет равен 1. Так как при делении на 2 существует всего два остатка – 0 и 1. При этом 0 означает, что число делится без остатка на 2, то есть является четным.

$k \% 2 \neq 0$

Слово не начинается на Ъ – значит первая буква в слове (a1) не равна «Ъ».

$a1 \neq 'Ъ'$

Полный код программы

```
k = 0
for a1 in sorted('КОМПЬЮТЕР'):
    for a2 in sorted('КОМПЬЮТЕР'):
        for a3 in sorted('КОМПЬЮТЕР'):
            for a4 in sorted('КОМПЬЮТЕР'):
                for a5 in sorted('КОМПЬЮТЕР'):
                    s = a1+a2+a3+a4+a5
                    k = k + 1
                    if s.count('К')==2 and k%2!=0 and a1!='Ъ':
                        print(k,s)
```

>>>

...

58493 ЮЮМКК

58655 ЮЮПКК

58817 ЮЮТКК

58979 ЮЮЮКК

>>>

Ответ: 58979

Telegram: @fast_ege