Разберём вариант задания №17 без файлов

Пример задания

Дан отрезок целых чисел [15785, 16425]. Определите количество и сумму чисел, в которых сумма цифр равна 32.

В качестве ответа запишите сначала количество, а потом сумму без пробелов и разделителей. Например, если количество чисел 99, а сумма 1234, то ответ 991234.

Для выполнения задания рекомендуется написать программу или воспользоваться редактором электронных таблиц.

Для упрощения решения и чтобы не запутаться с отступами, разобьём задачу на две:

1) Дано число, посчитать сумму цифр в нём
2) Перебрать все числа на диапазоне и посчитать сумму и количество тех, которые нас устраивают (сумма цифр = 32)

Решение задачи №1

Чтобы посчитать сумму цифр в числе, необходимо разбить число по цифрам. Стандартный алгоритм заключается в том, что мы каждый круг цикла выделяем из числа последнюю цифру (x % 10), а затем удаляем из числа последнюю цифру (x // 10). Тем самым, каждый раз мы выделяем новую цифру, а наше число уменьшается, пока совсем не закончится:

```
while x > 0:
    a = x % 10
    x = x // 10
```

Где x – первоначальное число, от которого мы «отрезаем» цифры
a – переменная, в которой каждый круг оказываются все цифры числа по очереди.

Теперь необходимо посчитать сумму цифр. Для этого заведём переменную sum_digit и будем добавлять к ней каждый круг новую цифру числа. Не забываем, что начальное значение суммы необходимо задать нулём!

```
sum_digit = 0
while x > 0:
    a = x % 10
    x = x // 10
    sum_digit += a
```

Важно!

Итоговое значение суммы получается только после того, как цикл полностью завершится:

```
sum_digit = 0
while x > 0:
    a = x % 10
    x = x // 10
    sum_digit += a
print(sum_digit)
#Выводим итоговую
#правильную сумму
```

```
sum_digit = 0
while x > 0:
    a = x % 10
    x = x // 10
    sum_digit += a
    print(sum_digit)
#Выводим кучу неверных
#промежуточных сумм
```

Также важно понимать, что этот алгоритм преобразует переменную x, в которой хранилось начальное число в 0, т.к. число «разбирается по цифрам».

Решение задачи №2

Чтобы обработать все числа в диапазоне, используем цикл for:

```
for x in range(15785, 16425 + 1):
    
```

+1 необходим, поскольку мы обрабатываем отрезок [15785, 16425], то есть и левая и правая границы должны быть включены.
В случае с функцией range правая граница «выколота»: мы идём до неё не включительно. Если написать без «+1», 16425 обрабатываться не будет. Можно написать и просто до 16426, разницы нет.

Чтобы посчитать сумму и количество чисел (переменные Sum и kol), подходящих условию, используем стандартные фрагменты алгоритмов (условие пока что не прописано). Обратите внимание, что к сумме нужно прибавлять первоначальное значение переменной x, а в алгоритме выше оно теряется. Для этого мы заводим переменную-копию:

```
Sum = 0
kol = 0
for x in range(15785, 16425 + 1):
    copy_x = x
    "считаем количество цифр в числе"
    if "количество цифр в числе 32":
        Sum = Sum + copy_x
        kol = kol + 1
print(Sum)
print(kol)
```

Важно!

Вывод итоговых значений Sum и kol должен происходит после цикла for, поскольку мы получаем верное значение, только после того, как все числа будут перебраны.

Остаётся объединить обе программы вместе:

```
Sum = 0
kol = 0
for x in range(15785, 16425 + 1):
    copy_x = x
    sum_digit = 0
    while x > 0:
        a = x % 10
        x = x // 10
        sum_digit += a
    if sum_digit = 32:
        Sum = Sum + copy_x
        kol = kol + 1
print(Sum)
print(kol)
```

Важно!

1) Начальное значение sum_digit = 0 задаётся внутри цикла for, поскольку мы его считаем отдельно для каждого нового числа x. И каждый раз заново (с нуля)
2) Условие if sum_digit == 32 должно находится после цикла while, потому что итоговый sum_digit мы получим только после обработки всех цифр числа x (после завершения цикла while).

Современный вариант задания №17 содержит файлы

Открытие текстового файла

Для открытия файла необходимо использовать конструкцию:

```
f = open('name_file.txt', 'r')
```

где
f – название переменной, которая будет ассоциирована с этим файлом
‘r’ – режим файла, о нём чуть позже.
‘name_file.txt’ – название файла, который вы хотите открыть. Расширение указывать обязательно.
Если ваш файл находится в одной папке с файлом кода, можно указать только название файла.
Если файл находится в другом месте, необходимо прописать полный путь, например:

```
f = open('C:\\Users\\pavel\\Downloads\\name_file.txt', 'r')
```

Важно везде использовать именно двойной обратный слэш \\, поскольку одиночный включает/отключает «особые функции» следующего символа. Например, вы уже знаете, что \n – это перенос строки (Энтер). \\n как раз отключит эту функцию.

Закрытие текстового файла

После того, как работа с файлом окончена, его необходимо закрыть:

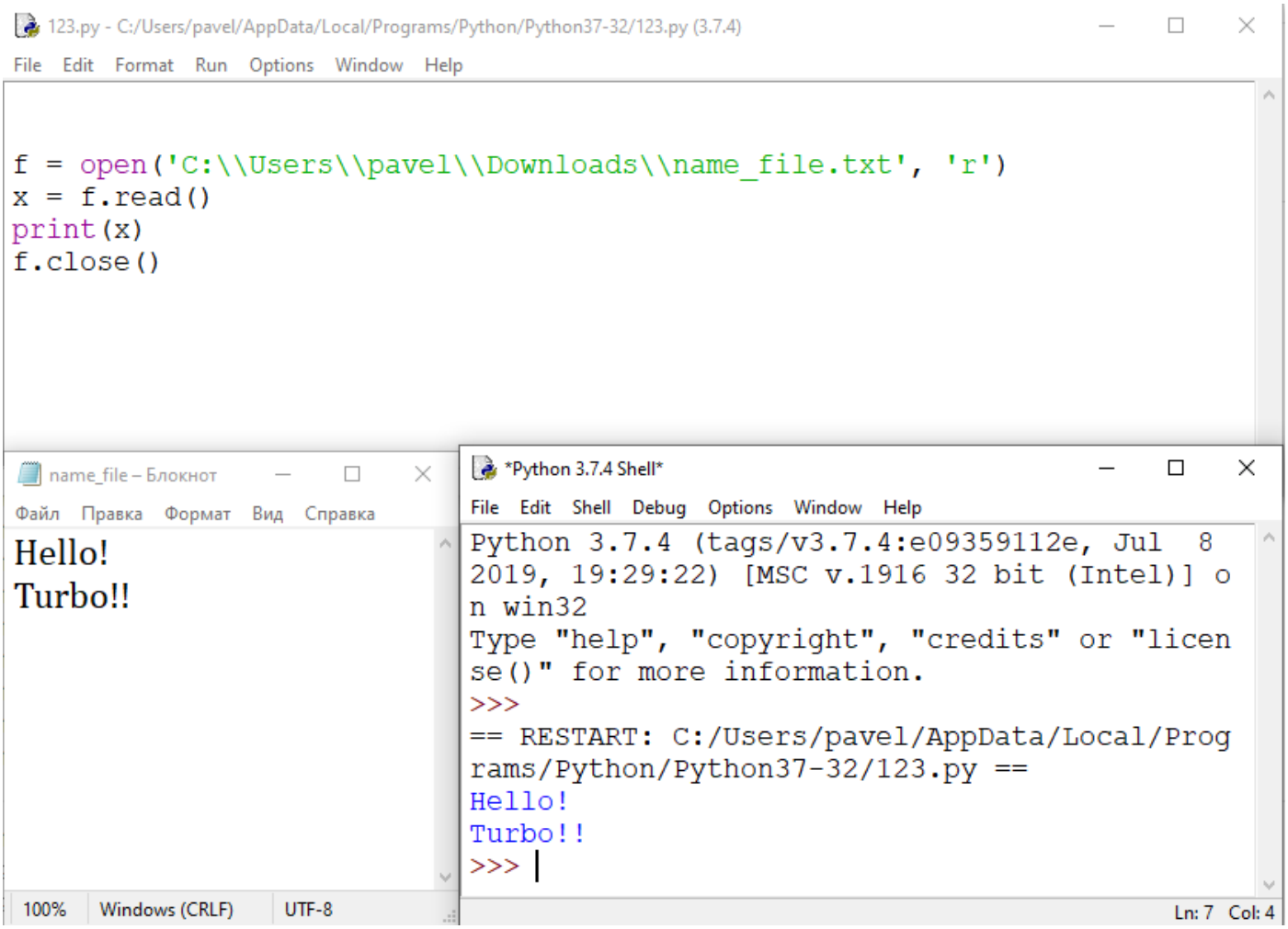
```
f.close()
```

Чтение информации из файла

Способ №1

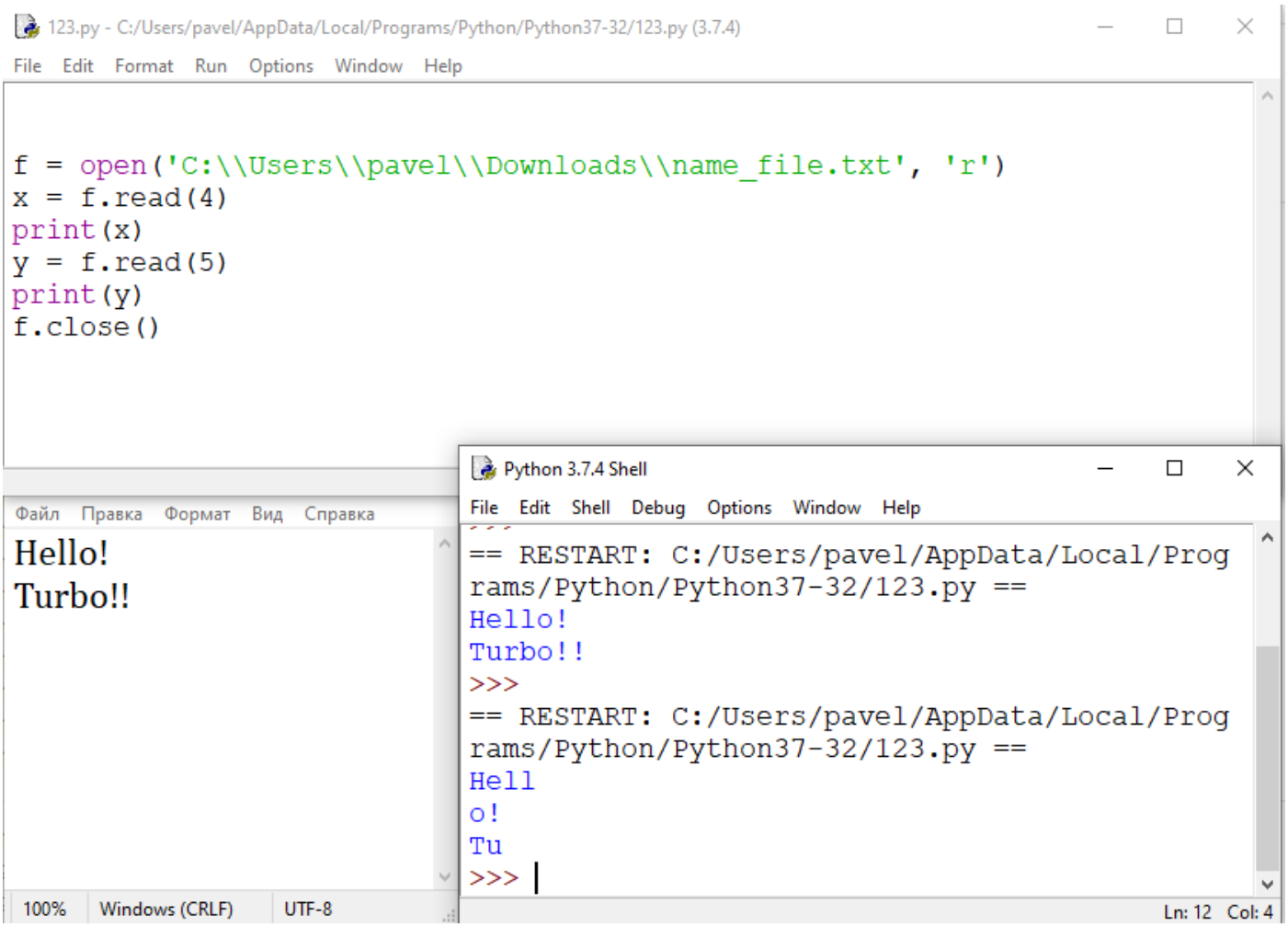
Можно прочитать сразу всю информацию из файла

a = f.read() – в переменную a запишется весь текст из файла (тип str).



Способ №2

Можно читать информацию из файла постепенно по несколько символов. Для этого нужно указать количество символов в скобках для функции read()



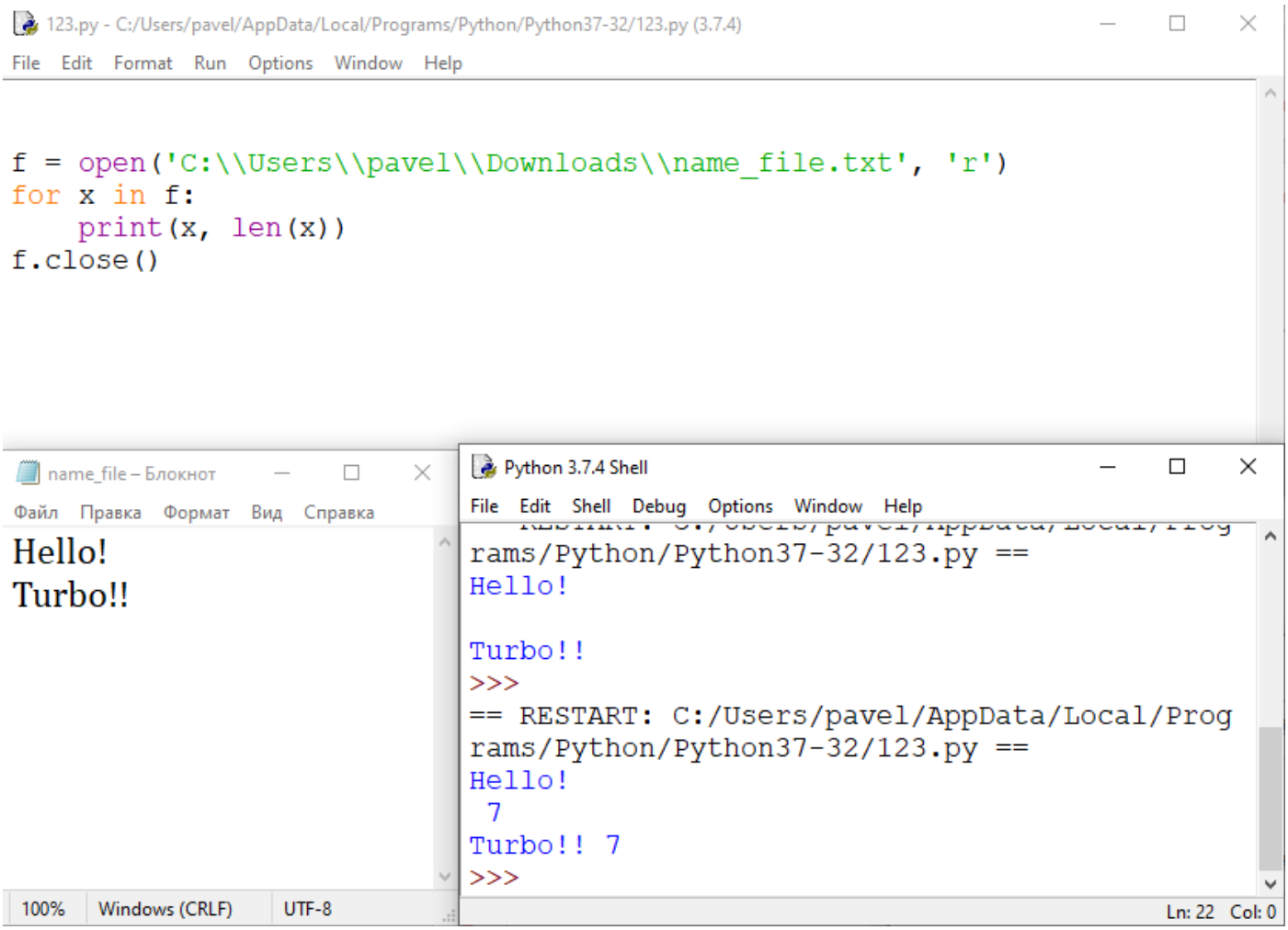
Важно:

В первый раз было прочитано 4 символа (‘H’, ‘e’, ‘l’, ‘l’), после чего курсор остался между символами ‘l’ и ‘o’. Следующее чтение начнётся с символа ‘o’.

Во втором чтении читается 5 символов (‘o’, ‘!’, ‘энтер’, ‘T’, ‘u’). Переход на следующую строку (энтер) – тоже считается символом.

Способ №3

Построчное чтение (похоже на input()) осуществляется через цикл for x in f:



Важно:
Строка читается целиком вместе с энтером, если он есть в конце строки.
В первой строке 7 символов: ‘H’, ‘e’, ‘l’, ‘l’, ‘o’, ‘!’, ‘enter’.
Во второй строке тоже 7 символов: ‘T’, ‘u’, ‘r’, ‘b’, ‘o’, ‘!’, ‘!’. В конце второй строки в файле не был нажат энтер, там перехода нет.

Любое чтение происходит в типе str.

Способ №4 (наиболее частый в ЕГЭ)

Работает один-в-один как input(): считывает строку текста из файла

x = f.readline()

В файле хранится сначала количество чисел (20), а затем 20 чисел:

```
=========
349
855
775
152
906
534
8
991
448
783
671
974
529
911
389
328
316
275
75
999
```

```
f = open("file.txt")
n = int(f.readline())
for i in range(n):
    x = int(f.readline())
    print(x)
f.close()
```

Разбор современного типа №17

Пример задания

В файле содержится последовательность целых чисел. Элементы последовательности могут принимать целые значения от -10 000 до 10 000 включительно. Определите и запишите в ответе сначала количество пар элементов последовательности, в которых хотя бы одно число делится на 3, затем максимальную из сумм элементов таких пар. В данной задаче под парой подразумевается два идущих подряд элемента последовательности. Например, для последовательности из 5 элементов: 7; 2; 9; -3; 6 – ответ: 4 11

Решение задачи

Организуем чтение информации из файла:

```
f = open("file.txt")
for x in f:
    x = int(x)
    'обработка числа x'
f.close()
```

Обработка числа и его предыдущего соседа:

```
f = open("file.txt")
pred = 15000 #начальное значение, заранее невозможное
for x in f:
    x = int(x)
    if pred != 15000: #проверяем, что это не первая итерация
        'обработка пары текущего и предыдущего элемента'
    pred = x
f.close()
```

Добавляем условия из задачи:

```
f = open("file.txt")
pred = 15000
kol = 0
maxx = -10001 #числа могут быть отрицательными
for x in f:
    x = int(x)
    if pred != 15000:
        if x % 3 == 0 or pred % 3 == 0: # хотя бы одно кратно 3
            kol += 1 #считаем кол-во
            if x+pred > maxx: #считаем макс сумму
                maxx = x+pred
    pred = x
print(kol, maxx) #вывод ответа
f.close()
```