# Строки. Срезы

1. [Работа со строками (повторение)](https://lyceum.yandex.ru/courses/468/groups/3752/lessons/2485/materials/6130#1)
2. [Срезы строк](https://lyceum.yandex.ru/courses/468/groups/3752/lessons/2485/materials/6130#2)

## Аннотация

*На этом занятии мы продолжим отрабатывать навыки работы со строкой и познакомимся с новым методом извлечения подстроки — срезами.*

## Работа со строками (повторение)

Рассмотрим еще одну задачу. Билет называют счастливым по-питерски, если сумма цифр его номера, стоящих на четных местах, равна сумме цифр, стоящих на нечетных местах. Нам необходимо написать программу, которая определяет, является ли билет счастливым по-питерски.

Если рассматривать номер билета как строку из цифр, задача сводится к подсчету суммы цифр, стоящих на позициях 0, 2, 4..., и суммы цифр, стоящих на позициях 1, 3, 5... Чтобы перебрать элементы, мы можем воспользоваться конструкцией for i in range(...), указав шаг 2. Тогда соответствующий фрагмент программы может выглядеть следующим образом:

number = input()

odd = even = 0

for i in range(0, len(number), 2):

odd += int(number[i])

for i in range(1, len(number), 2):

even += int(number[i])

if odd == even:

print('Счастливый по-питерски!')

Подумайте, как можно решить данную задачу за один цикл.

## Срезы строк

На примере разобранной задачи мы увидели, что перебор элементов строки с помощью конструкции for i in range(...) является достаточно гибким: можно перебрать не все индексы, можно идти с шагом, скажем, 2 или даже −1, то есть в обратном порядке. Но существует способ без всякого цикла преобразовать строку нужным образом: взять отдельный ее кусок, символы с нечетными номерами и т. д. Этот способ — **срез (slice)**.

**Срез строки**

В самом простом варианте срез строки — ее кусок от одного индекса включительно и до другого не включительно (как для range). То есть это новая, более короткая строка.

Срез записывается с помощью квадратных скобок, в которых указывается начальный и конечный индекс, разделенные двоеточием.

text = 'Hello, world!'

print(text[0:5])

print(text[7:12])

Если не указан **начальный индекс**, срез берется от начала (от 0). Если не указан **конечный индекс**, срез берется до конца строки. Попробуйте предположить, что будет выведено на экран, если в предыдущей программе записать срезы следующим образом:

text = 'Hello, world!'

print(text[:5])

print(text[7:])

**Разрешены отрицательные индексы** для отсчета с конца списка. В следующем примере из строки, содержащей фамилию, имя и отчество, будет извлекаться фамилия.

full\_name = 'Иванов И. И.'

surname = full\_name[:-6]

Как и для range, в параметры среза можно добавить третье число — **шаг обхода**. Этот параметр не является обязательным и записывается через второе двоеточие. Вот как может выглядеть программа «Счастливый билет», если решать ее с помощью срезов:

number = input()

odd = even = 0

# срез будет от начала строки до конца с шагом два: 0, 2, 4,...

for n in number[::2]:

odd += int(n)

# срез от второго элемента строки до конца с шагом два: 1, 3, 5,...

for n in number[1::2]:

even += int(n)

if odd == even:

print('Счастливый по-питерски!')

Интересное отличие среза от обращения по индексу к отдельному элементу состоит в том, что мы не получим ошибку при указании границ среза за пределами строки. В срез в таком случае попадут только те элементы, которые находятся по валидным индексам среза:

a = 'Python'

print(a[2:10000]) # thon

print(a[999:]) # пустая строка

**Шаг может быть и отрицательным** — для прохода по строке в обратном порядке. Если в этом случае не указать начальный и конечный индекс среза, ими станут последний и первый индексы строки соответственно (а не наоборот, как при положительном шаге):

text = 'СЕЛ В ОЗЕРЕ БЕРЕЗОВ ЛЕС'

text\_reversed = text[::-1]

print(text == text\_reversed)

Итак, с помощью квадратных скобок можно получить доступ как к одному символу строки, так и к некоторой последовательности символов, причем совсем необязательно идущих подряд!