

Лекция 3

Списки.
Генераторы списков.

Списки

- **Список (list)** — основная структура для работы с последовательностью данных в Python
- Можно задать перечислением элементов в `[]` или с помощью конструктора / преобразования типа
- Элементом списка может быть любой объект!

Списки

- **Список (list)** — основная структура для работы с последовательностью данных в Python
- Можно задать перечислением элементов в `[]` или с помощью конструктора / преобразования типа
- Элементом списка может быть любой объект!

```
days = ['Mon', 'Tue', 'Wed',  
        'Thu', 'Fri', 'Sat', 'Sun']  
  
day_numbers = [1, 2, 3, 4, 5,  
               6, 7]  
  
weird_list = [42, None,  
              'sunny', ['loan']]  
  
empty_list = list()  
  
print(days[1],  
       day_numbers[1])
```

Списки

- Преобразование типов: *range* можно привести к типу *list* !
- Как перебрать все элементы списка в цикле?

```
l = [0, 1, 2, 3, 4]  
l = list(range(5))
```

```
colors = ['red', 'green',  
          'blue', 'white', 'black']
```

Списки

- Преобразование типов: *range* можно привести к типу *list* !
- Как перебрать все элементы списка в цикле?
- ***for i in range(len(lst))***
— обход по индексам (как в любом языке программирования)
- ***for elem in lst***
— обход по элементам (pythonic way)

```
l = [0, 1, 2, 3, 4]
l = list(range(5))
```

```
colors = ['red', 'green',
          'blue', 'white', 'black']
```

```
for i in range(len(colors)):
    print(colors[i], end=' ')
print()
```

```
for color in colors:
    print(color, end=' ')
```

Списки и строки

- Можно обращаться по индексам и измерять длину
- Можно использовать срезы!
- Можно использовать вместо *range* для цикла *for*
- Можно конвертировать строку в список символов

Списки и строки

- Можно обращаться по индексам и измерять длину
- Можно использовать срезы!
- Можно использовать вместо *range* для цикла *for*
- Можно конвертировать строку в СПИСОК СИМВОЛОВ

```
s = 'abc'  
l = [1, 2, 3]  
  
len(s) == len(l)  
print(s[::-1], l[::-1])  
  
for c in s:  
    print(c)  
  
print(list(s))
```



Списки и строки

- Можно конвертировать строку в СПИСОК СИМВОЛОВ
- А наоборот?

```
print(list('abc'))
```

```
print(list(str(list('abc'))))
```


Списки и строки

- Можно конвертировать строку в СПИСОК СИМВОЛОВ
- А наоборот?
- *s.split(sep)* разбивает строку *s*, используя *sep* как разделитель. По умолчанию — пробел + табуляция + `'\n'`
- *sep.join(lst)* — объединяет элементы списка *lst* в строку, разделяя их *sep*. Все элементы списка должны быть *str*!

```
sent = 'fall leaves as soon as  
leaves fall'
```

```
words = sent.split()  
print(words)
```

```
words = words[::-1]  
sent = ' '.join(words)  
print(sent)
```

```
nums = '1,2,3,4'  
print(' \n'.join(nums.split(',')  
))
```

Списки vs строки

- Строки нельзя изменять. Однако можно делать повторное присваивание.
- Списки изменять можно: и отдельные элементы, и срезы!
- *str* — неизменяемый (immutable) тип, *list* — изменяемый (mutable).

```
s = 'misppell'
l = [1, 1, 2, 3, 5, 9]

#s[3] = 's'
s = s[:3] + 's' + s[4:]

l[-1] = 8
l[-1:] = [8, 13, 21]
print(l)
```

Списки: методы

Метод	Назначение
<i>len(l)</i>	Возвращает длину списка
<i>l1 + l2</i>	Конкатенация списков l1, l2
<i>l * n</i>	Повторение списка n раз
<i>l.append(elem)</i>	Вставляет элемент в конец списка
<i>l.extend(lst)</i>	Вставляет все элементы из lst в конец списка l
<i>l.pop()</i>	Удаляет последний элемент из списка и возвращает его
<i>l.sort([key=None])</i>	Сортировка списка (можно указать ключ, по которому нужно сортировать)
<i>l.count(elem)</i>	Подсчет числа вхождений

Списки: замечания

Быстрые операции (работают за $O(1)$):

- добавить в конец (*append*)
- убрать с конца (*pop*)
- обратиться/изменить по индексу (*[]*)
- узнать длину (*len*)

Медленные операции (работают за $O(n)$):

- добавить в произвольное место (*insert*)
- удалить по значению (*remove*)
- проверить принадлежность (*in*)

Медленные методы, как правило, не используют даже при работе с небольшими данными: для эффективного выполнения этих операций есть другие структуры.

Запятая

- Упрощение присваивания
- Легко менять местами переменные
- Перечисление элементов списка
- Можно использовать отдельные элементы в *for*

```
a = 1
b = 2
a, b = 1, 2

# simplest swap ever
a, b = b, a

a, b = [1, 2]

for direction in 'east',
    'west', 'south', 'north':
    print(direction)
```

Кортеж (tuple)

- Кортеж — список, который нельзя изменять
- Можно конвертировать из *list* в *tuple* и наоборот
- Кортежи нужны, например, для словарей - подробнее на следующих лекциях.

```
a = (1, 2, 3)
b = tuple() # b = ()
c = (3,) # comma is required

l = [0, 1, 2, 6, 4, 5]
t = tuple(l)
print(t[0], t[-2:])
l[3] = 3
# t[3] = 3
```

Генераторы списков

Генератор списка (*list comprehension*) — удобный способ создания списка по формуле

[expr(i) for i in sequence]

Генераторы списков

Генератор списка (*list comprehension*) — удобный способ создания списка по формуле

[expr(i) for i in sequence]

- Заполнение константами или простыми выражениями
- Генерация списка для join
- Чтение списка чисел

```
zeroes = [0, 0, 0, 0, 0]
zeroes = [0] * 5
zeroes = [0 for i in range(5)]

squares = [i * i for i in range(5)]

s = ' '.join([str(i) for i in
squares])
print(s)

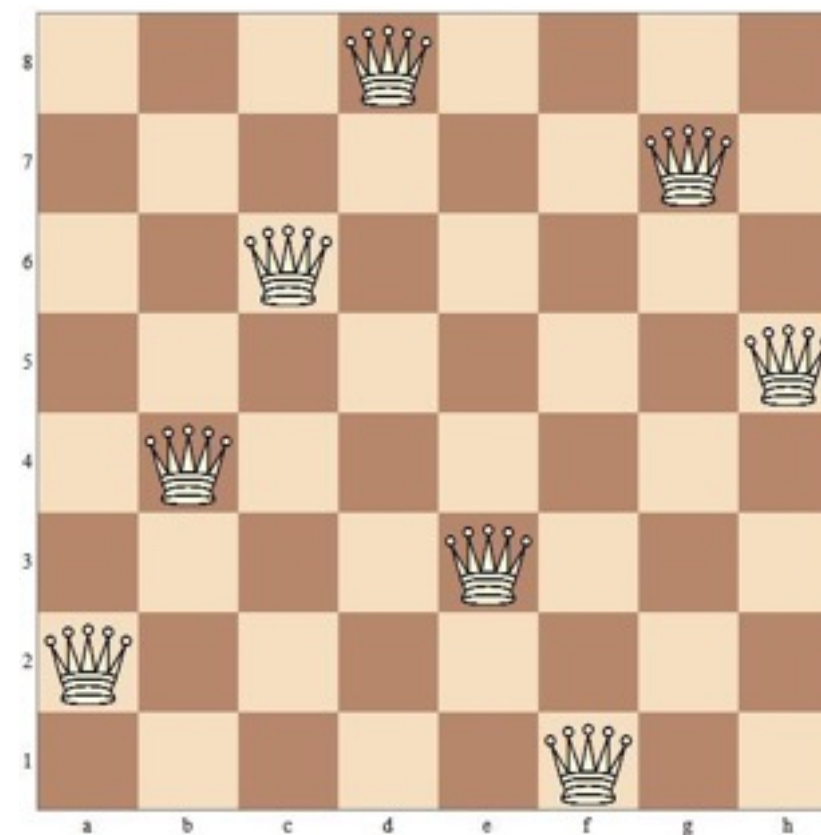
ints = [int(i) for i in
input().split()]
print(sum(ints))
```

Подробнее про генераторы и функциональное программирование — на следующих лекциях.

Вложенные списки

Задача: дана шахматная доска, на которой стоит несколько ферзей. Нужно определить, бьют ли они друг друга.

Для чтения данных мы хотим создать таблицу из нулей размера 8x8: как это можно сделать?



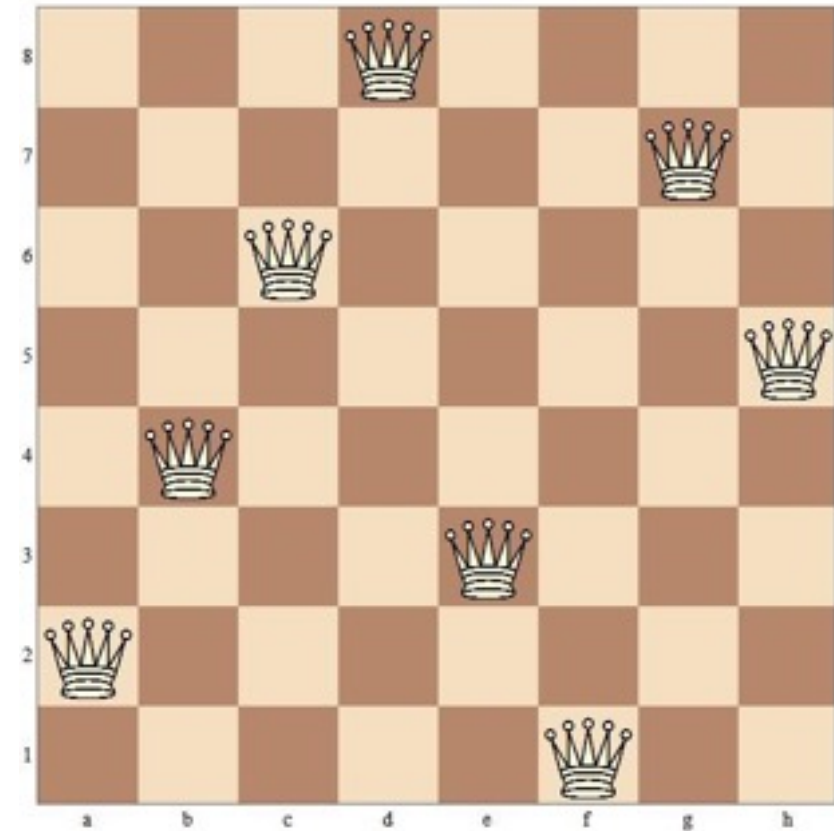
0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0

Вложенные списки

Задача: дана шахматная доска, на которой стоит несколько ферзей. Нужно определить, бьют ли они друг друга.

Для чтения данных мы хотим создать таблицу из нулей размера 8x8: как это можно сделать?

- `[[0] * 8] * 8` — создает 8 ссылок на один и тот же список, так делать нельзя
- `[[0] * 8 for i in range(8)]` — ОК
- `[[0 for i in range(8)] for j in range(8)]` — тоже ОК



0	0	0	1	0	0	0	0
0	0	0	0	0	0	1	0
0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0
0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	1	0	0

Вложенные списки

- Обращение к элементу таблицы: $a[i][j]$
- Обход всех элементов таблицы (для вывода на печать, поиска суммы и т.д.) — двумя вложенными циклами или вложенным генератором списков.
- Можно создавать списки размерности 3 и больше — the sky is the limit.

Чтение текста

```
# Read multiple lines
n = int(input())
lines = []
for i in range(n):
    lines.append(input())
text = ' '.join(lines)
words = text.split()
print(words)
```

```
>3
>An old silent pond...
>A frog jumps into the pond.
>Splash! Silence again.
['An', 'old', 'silent', 'pond...', 'A', 'frog', 'jumps',
'into', 'the', 'pond.', 'Splash!', 'Silence', 'again.']
```

Задачи

1. (Odd indices)

Вывести все элементы списка *lst* с нечетными индексами.

2. (Unique elements)

На вход подается последовательность из n чисел. Нужно построить список, из которого удалены все повторяющиеся элементы. Порядок можно не сохранять.

Например,

1, 2, 1, 1, 3, 4, 2 -> 2, 1, 3, 4