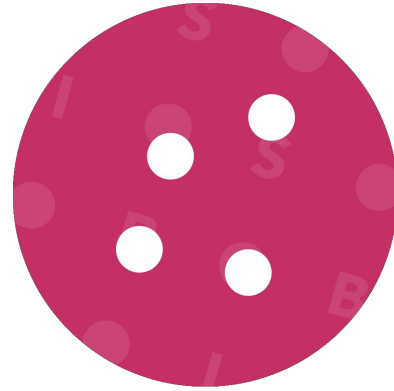
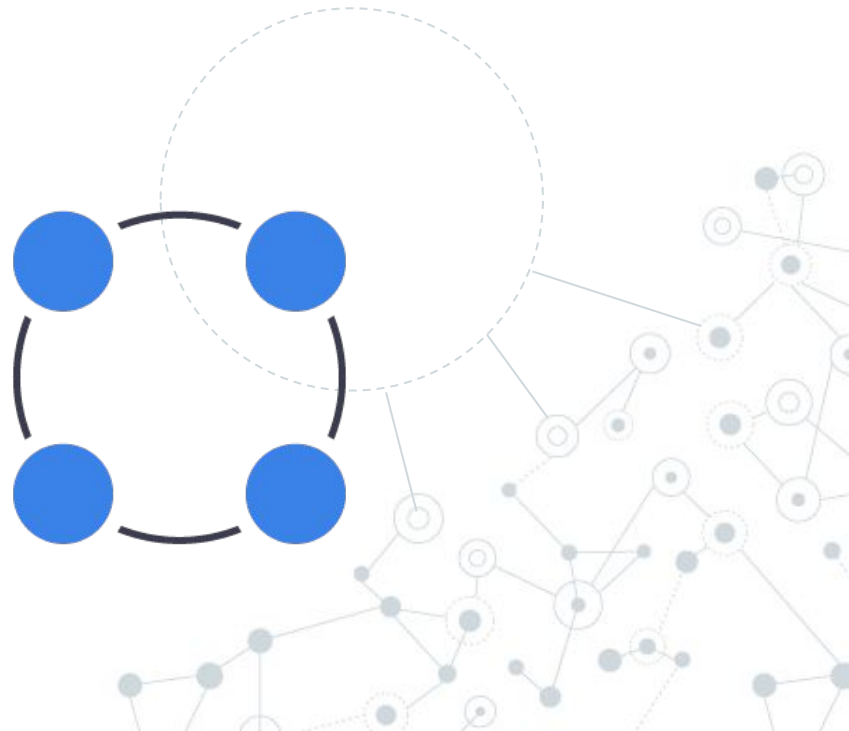


Python



Тимофей Проданов

timofey.prodanov@gmail.com



Кратчайшая история

- ◎ Создан в 1991
- ◎ Назван в честь Monty Python
- ◎ В 2008 разделился на
python 2
python 3
- ◎ Python 2 уже почти умер

Философия

- ◎ Beautiful is better than ugly
- ◎ Explicit is better than implicit
- ◎ Simple is better than complex
- ◎ Complex is better than complicated
- ◎ Readability counts

R

- ⊙ Красивые графики
- ⊙ Очень медленный
- ⊙ Плохая архитектура языка
- ⊙ Много хороших библиотек

Python

- ⊙ Много удобных встроенных функций
- ⊙ Очень интуитивный
- ⊙ Приятно писать
- ⊙ Много, но меньше хороших библиотек

Зачем использовать

- ⊙ Рисовать графики
- ⊙ Использовать только библиотечные функции
- ⊙ Что-то писать самому

Запустить Python

`python3`

Иногда сойдёт

`python`

`jupyter notebook`



Математические выражения

Математические выражения

- ⊙ 5
- ⊙ 10
- ⊙ 2.3
- ⊙ -4
- ⊙ $5 - 4$
- ⊙ $8 * 6.6$
- ⊙ $(5 + 6) * 4$

Операции

- ⊙ **+** сложение
- ⊙ **-** вычитание
- ⊙ ***** умножение
- ⊙ **/** деление
- ⊙ ****** возведение в степень
- ⊙ **//** целочисленное деление
- ⊙ **%** остаток от деления

Переменные

Нет особого смысла использовать питон как калькулятор

Результаты можно сохранять в переменные

Переменные - хранилища данных

```
a = 5
```

```
In [3]: a = 5
```

```
In [4]: print(a)
```

5

```
In [6]: b = 13 + 14
```

```
In [7]: print(b)
```

27

In [11]: a, b

Out[11]: (5, 27)

In [12]: c = a + b + 10

In [13]: c

Out[13]: 42

Можно перезаписывать значение

```
In [14]: a
```

```
Out[14]: 5
```

```
In [15]: a = 10
```

```
In [16]: a
```

```
Out[16]: 10
```

Ввести число

```
In [*]: b = int(input())
```


Ввёл 111

```
In [18]: b = int(input())
```

```
111
```

```
In [19]: b
```

```
Out[19]: 111
```



Условные выражения

Условие

```
In [22]: if a > 0:  
         print("Positive number :)")
```

```
Positive number :)
```


Чуть более сложное условие

```
In [23]: if a > 20:  
         print("Hot")  
else:  
         print("Cold")
```

Cold

Самое сложное

```
In [24]: if a > 0:  
        print("Positive")  
elif a == 0:  
        print("WOW")  
        print("a is ZERO")  
        print("ZER0000")  
else:  
        print("Negative")
```

Positive

Написать самим

- ◎ Дан возраст, написать, какое учебное заведение подходит:
- a. "Too young"
 - b. "School"
 - c. "University"
 - d. "Too old"

Можно писать

$0 < a \leq 10$

Булевы выражения

⊙ <, <=

⊙ >, >=

⊙ ==, !=

⊙ **and**

⊙ **or**

⊙ **not**

```
a
```



```
10
```

```
5 < a < 15
```

```
True
```

```
a > 5 and a < 15
```

```
True
```

- 
- 
- ◎ **and** - оба условия должны быть правдой
 - ◎ **or** - хотя бы одно из условий должно быть правдой
 - ◎ **not** - условие должно быть неправдой



Циклы

Цикл for

Начинается с нуля, до 5 не доходит

```
for i in range(5):  
    print(i)
```

0
1
2
3
4

Сумма чисел от 0 до 9

```
s = 0
for i in range(10):
    s = s + i
print(s)
```

45

Сумма чисел от 1 до 10

```
s = 0
for i in range(1, 11):
    s = s + i
print(s)
```

55

Сокращение

Одно и то же:

```
S = S + i
```

```
S += i
```

```
S = S + i * 10 + 213768
```

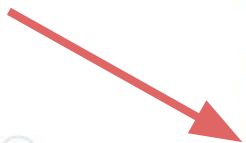
```
S += i * 10 + 213768
```

Работает не только с плюсом

if внутри for

```
for i in range(7):  
    if i != 4:  
        print(i)
```

0
1
2
3
5
6



Сделать самим

С помощью одного цикла и if посчитать отдельно

- a. сумму всех **чётных** чисел
- b. сумму всех **нечётных** чисел

на промежутке от 0 до 100

A decorative network diagram in the top-left corner, consisting of various sized nodes (some solid grey, some hollow white) connected by thin grey lines, forming a complex web structure.

Строки

A decorative network diagram in the bottom-right corner, similar to the one in the top-left, with nodes and connecting lines.

Уже знаем три типа данных

Числа:

- ◎ 0
- ◎ 10
- ◎ -5.5

Булены (True, False)

Строки:

- ◎ "Hello"
- ◎ "a"
- ◎ ""
- ◎ "sajdbshdb asdhsadsaldjaskdh"

```
print("Hi")
```

Hi

```
s = "Summer"
```

s

'Summer'

Разница между '' и ""

Никакой. Разве что

```
print('aaa " aaa')
```

aaa " aaa

```
print("aaa ' aaa")
```

aaa ' aaa

```
print('aaa \' " aaa')
```

aaa ' " aaa

Обратите
внимание

Экранированные символы

```
print( 'aaa\nbbb' )
```

aaa

bbb

```
print( 'aaa\tbbb' )
```

aaa

bbb

Сам слэш тоже надо экранировать

```
print('Common slash: /')
```

```
this is ok: /
```

```
print('Backslash: \\')
```

```
Backslash: \
```

Ещё бывает

```
s = '''This  
is  
multiline  
string'''
```

```
print(s)
```

```
This  
is  
multiline  
string
```

Считать строку

```
In [*]: s2 = input()
```

Я ввёл приветствие

```
In [45]: s2 = input()
```

Heeeeeey

```
In [46]: s2
```

```
Out[46]: 'Heeeeeey'
```

Индексация

Можно получить любую букву (только нумерация **с нуля**):

```
s = 'Summer'
```

```
s[0]
```

```
'S'
```

```
s[4]
```

```
'e'
```

Индексация с конца

Длина строки

```
len(s)
```

6

Последняя
буква


```
s[len(s) - 1]
```

'r'

Тоже
последняя
буква

```
s[-1]
```

'r'



Hello

0 1 2 3 4

-5 -4 -3 -2 -1

Цикл по строке

```
s = "index"
```

```
for i in range(len(s)):  
    print(i, s[i])
```

```
0 i  
1 n  
2 d  
3 e  
4 x
```


Немного про print


```
print('aaa')  
print('bbb')
```

aaa
bbb

```
print('aaa', end='')  
print('bbb')
```

По умолчанию `end='\n'`

aaabbb



Сделать самим

Примечание: для этого поможет предыдущий слайд (end = ‘’)

- ⊙ Вывести комплементарную строку

ACCCGAA → TGGGCTT

- ⊙ Вывести реверс комплементарную строку

ACCCGAA → TTCGGGT

- ⊙ Посчитать GC контент

ACAGT → $\frac{2}{5} = 0.4$



Ещё немного, пожалуйста

Проверить строку на палиндром
(Слева направо строка такая же, как и
справа налево)

Палиндром например

ACGATAGCA

ACGATAGCA → True

ACGGTAGCA → False

