

# Хорошо слышно и видно?

Ставьте +, если все хорошо  
Пишите, если есть проблемы

Вебинар Основы Python



Y<sub>—</sub>LAB  
UNIVERSITY

# Преподаватель

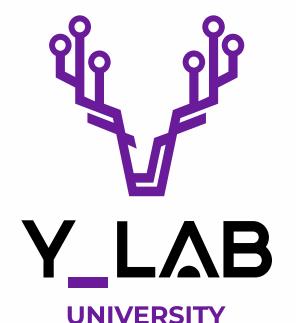
---



## Михаил Вотинов

- Более 8 лет работаю в ИТ
- Занимаюсь развитием отдела Python разработки в качестве тимлида
- Интересуюсь разработкой и реализацией архитектуры высоконагруженных веб-сервисов

# Цели вебинара



## После занятия вы сможете:

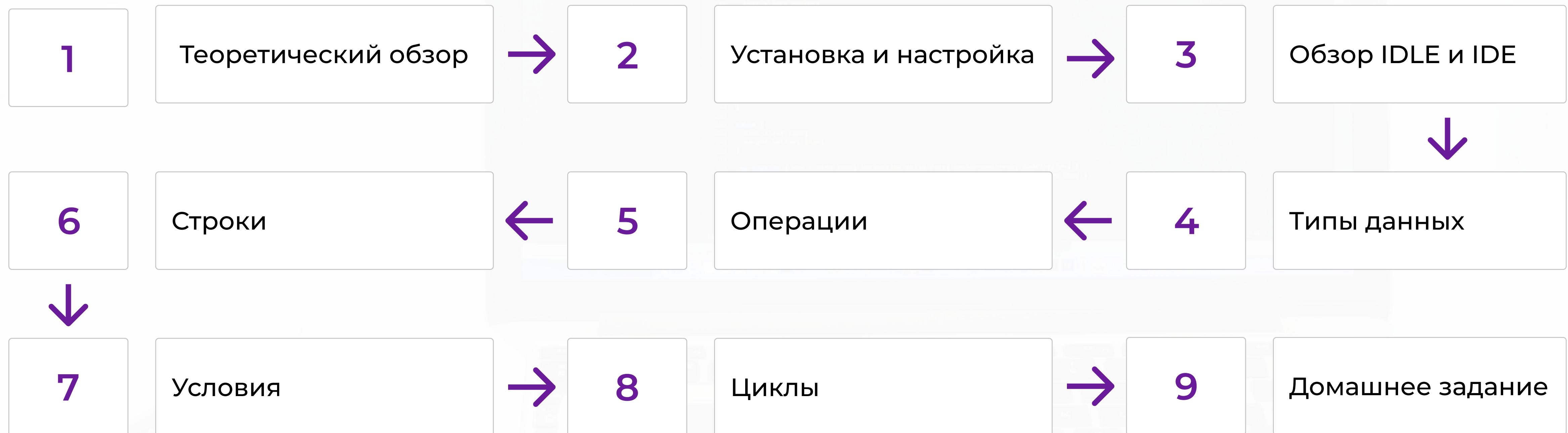
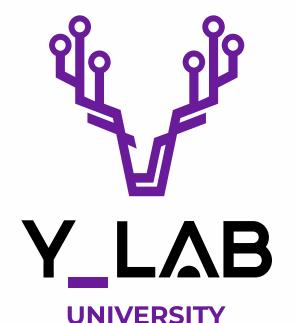
1 Установить и настроить Python на своем компьютере

2 Ориентироваться в типах и структурах данных

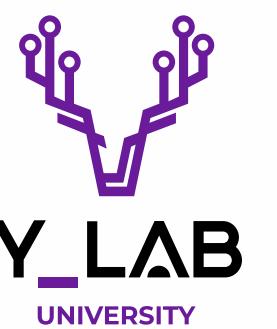
3 Работать со строками

4 Выполнять операции, разрабатывать условия и циклы

# Маршрут вебинара



# Правила вебинара



- Активно участвуйте
- Задавайте вопросы
- Оффтоп вопросы можем обсуждать после занятия в общем чате

# Python

## Теоретический обзор



- Высокоуровневый
- Мультипарагдимальный
- Общего назначения
- Интерпретируемый
- Динамическая строгая типизация
- Автоматическое управление памятью
- Лаconичный синтаксис
- Разработан на С (CPython)

# Python

## Применение



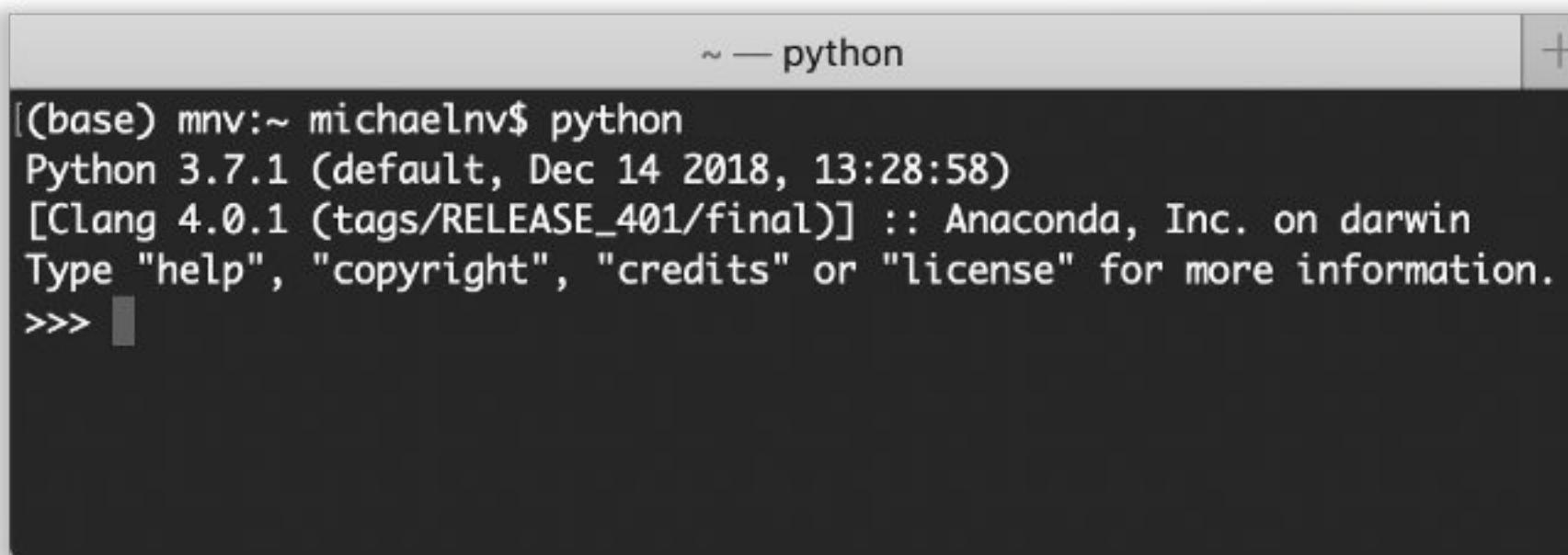
- Data Science
- Web-разработка
- Десктопные приложения
- Автоматизация
- Разработка игр
- Научные вычисления

# Python

## Установка и настройка

### macOS / Linux

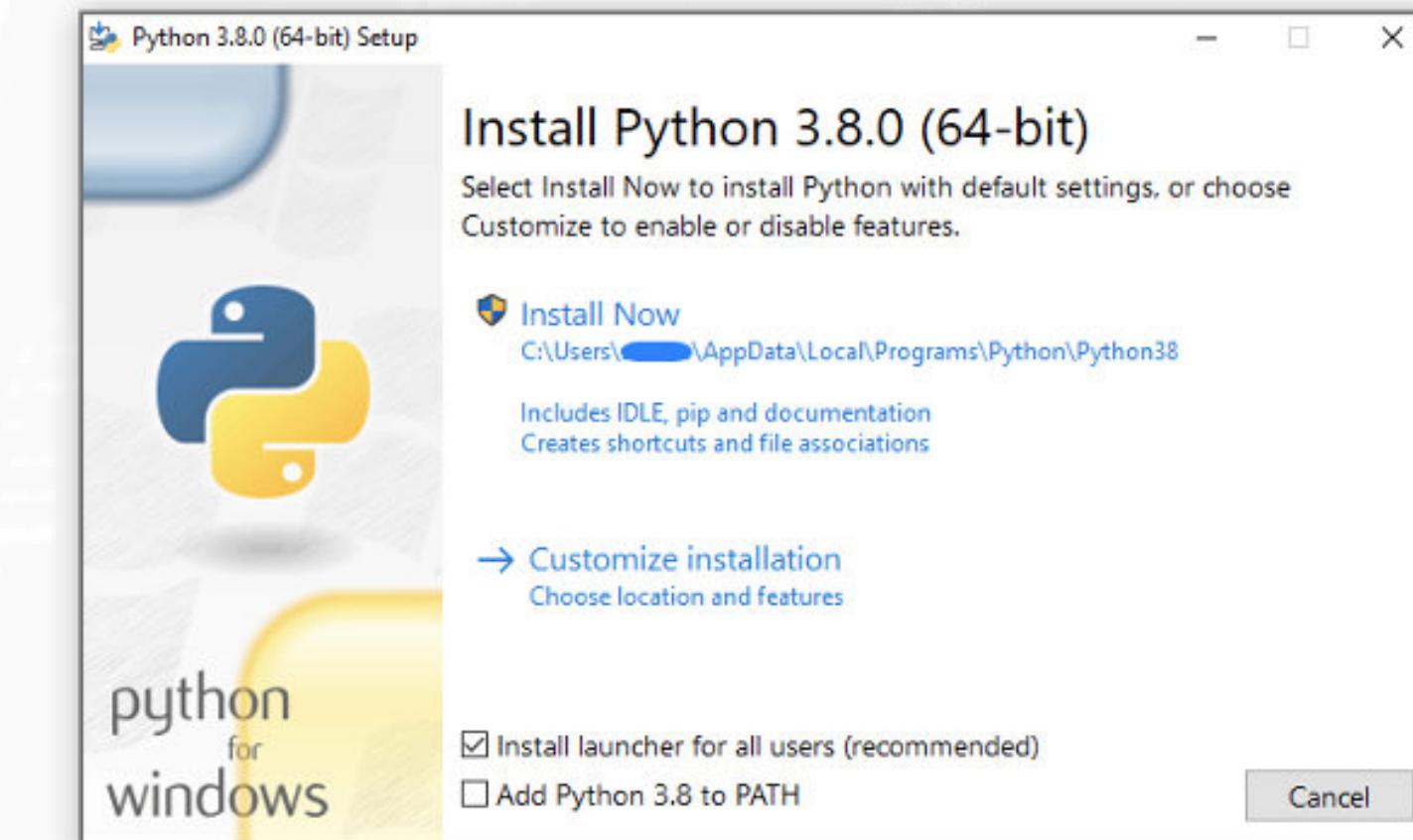
Поставляется вместе с ОС



```
~ — python
(base) mnv:~ michaelnv$ python
Python 3.7.1 (default, Dec 14 2018, 13:28:58)
[Clang 4.0.1 (tags/RELEASE_401/final)] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

### Windows

Требуется установка



<https://www.python.org/downloads>

# Python

## Обзор IDLE и IDE



**IDLE** (Integrated Development and Learning Environment)

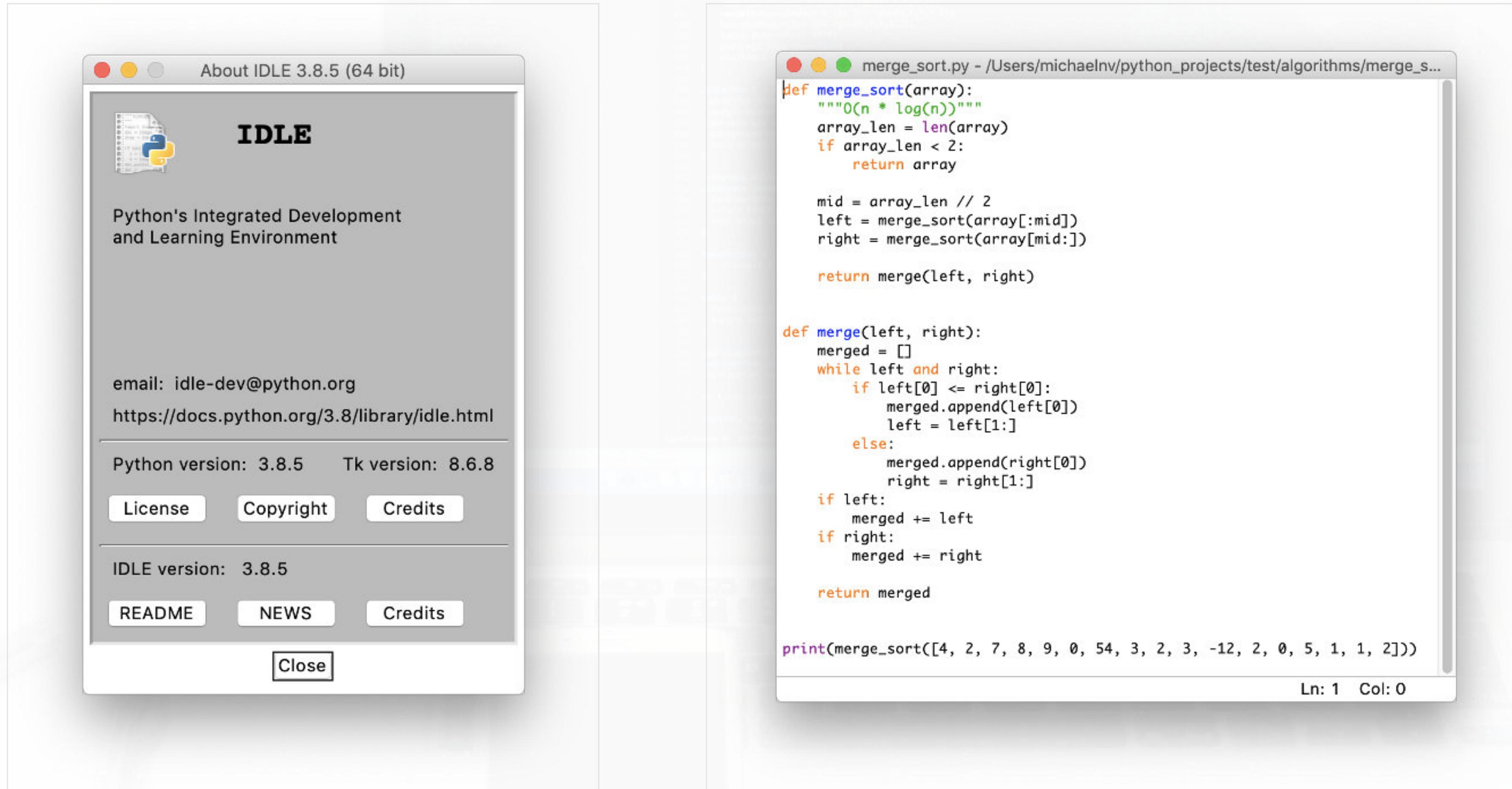
- Обучение
- Быстрый запуск скриптов
- Поставляется вместе с Python
- Подсветка синтаксиса

**IDE** (Integrated Development Environment)

- Комплекс программ
- Промышленная разработка ПО
- Работа с виртуальными окружениями и СУБД
- Средства отладки

# Python

## Обзор IDLE



<https://docs.python.org/3/library/idle.html>

# Python

## Обзор IDE



- Eclipse+PyDev

- Visual Studio

- PyCharm

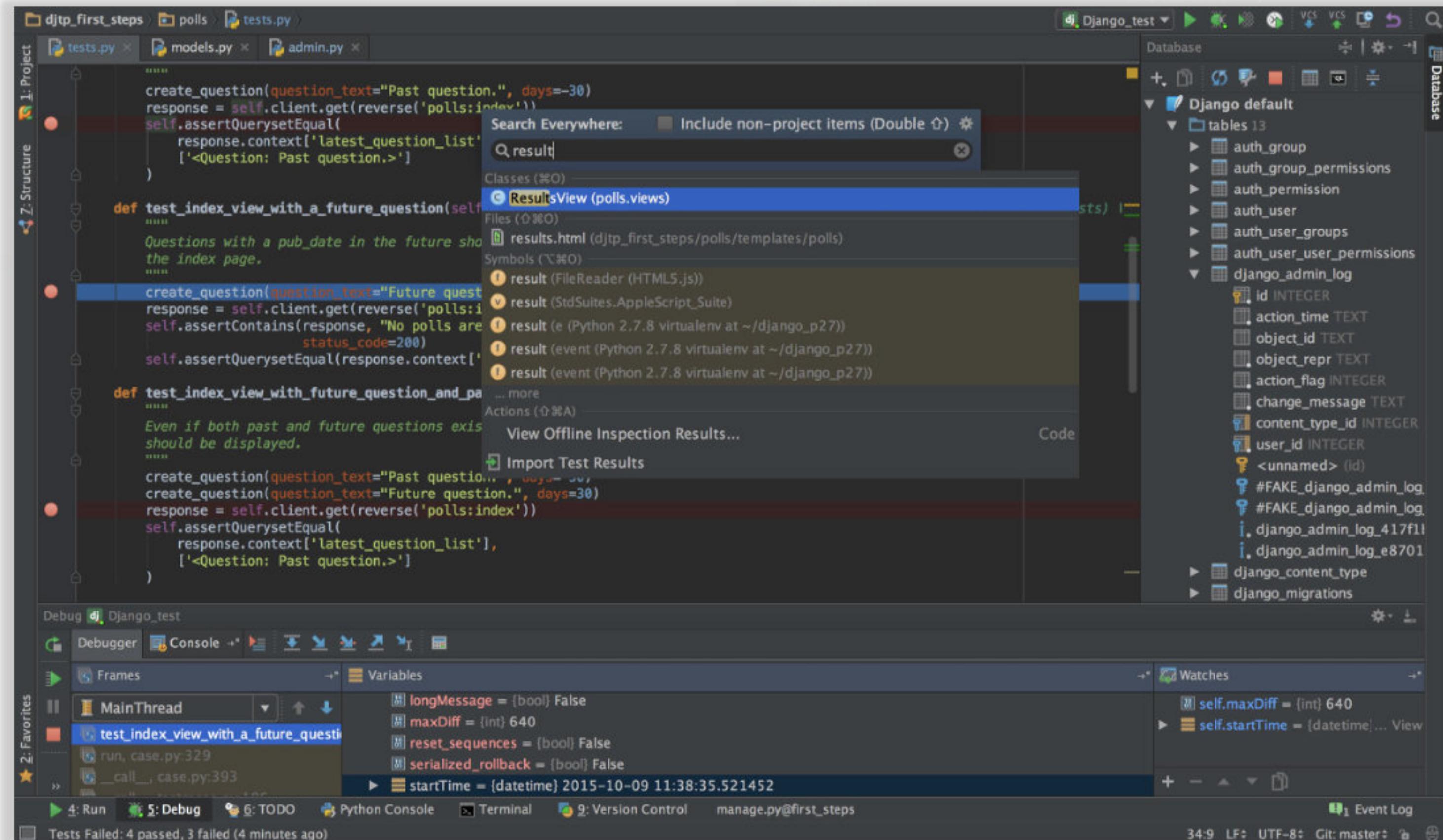
- Atom

- Sublime Text

- vim

# Python

## Обзор IDE PyCharm



- Плагины для фреймворков

- Автодополнение кода

- Проверка ошибок

- Поддержка систем контроля версий

- Работа с базами данных

- Встроенный терминал

<https://www.jetbrains.com/ru-ru/pycharm>

# Python

Интерактивная оболочка



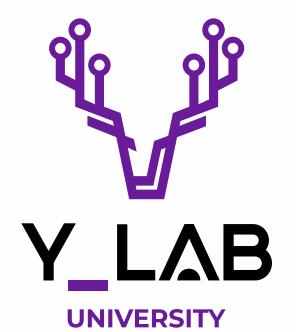
## Jupyter Notebook

A screenshot of a Jupyter Notebook interface. The title bar says "jupyter Untitled (unsaved changes)". The toolbar includes File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, Python 3, and memory status (176.4 MB / 2 GB). Below the toolbar are download, GitHub, and Binder buttons. The main area shows a cell with the code "In [1]: print('Hello, World!')". The output of the cell is "Hello, World!". A new cell is being typed with "In [ ]:".

<https://jupyter.org/try>

# Python

## Типы данных и работа с памятью



### Логические переменные

bool

### Числа

int

float / decimal

### Списки

list

tuple

### Строки

str

### Множества

set

### Словари

dict

### Всё является объектом:

**ID**

**Значение**

**Тип**

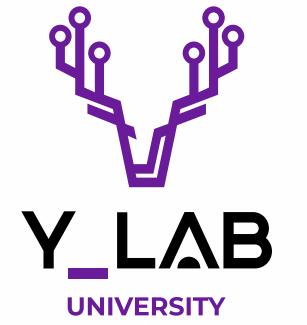
**145488**

**Привет**

**str**

# Python

## Типы данных и работа с памятью. Изменяемость типов



### Неизменяемые типы

Логический (bool)

Числа (int, float)

Строки (str)

Кортежи (tuple)

Неизменяемые множества (frozenset)

Отсутствие значения (None)

Созданный объект больше не изменится

Значения переменных можно менять

### Изменяемые типы

Списки (list)

Множества (set)

Словари (dict)

**Значение объекта можно изменить**

**ID и тип не изменяются**

**ID    Значение    Тип**

### Имя переменной = Значение

Начинается с буквы или символа подчеркивания

Не должно совпадать с ключевыми словами (False, None, and, as, assert, break, class, ...)

```
my_variable = "Текстовое значение"
```

```
my_variable = 5
```

# Python

## Типы данных и работа с памятью. Примеры кода



```
my_str = "Привет"
```

```
my_int = 5
```

```
my_float = 5.789
```

```
my_bool = True
```

ID  
Значение  
Тип

```
my_tuple = ("Привет", 5, True)
```

```
my_list = ["Привет", 5, True]
```

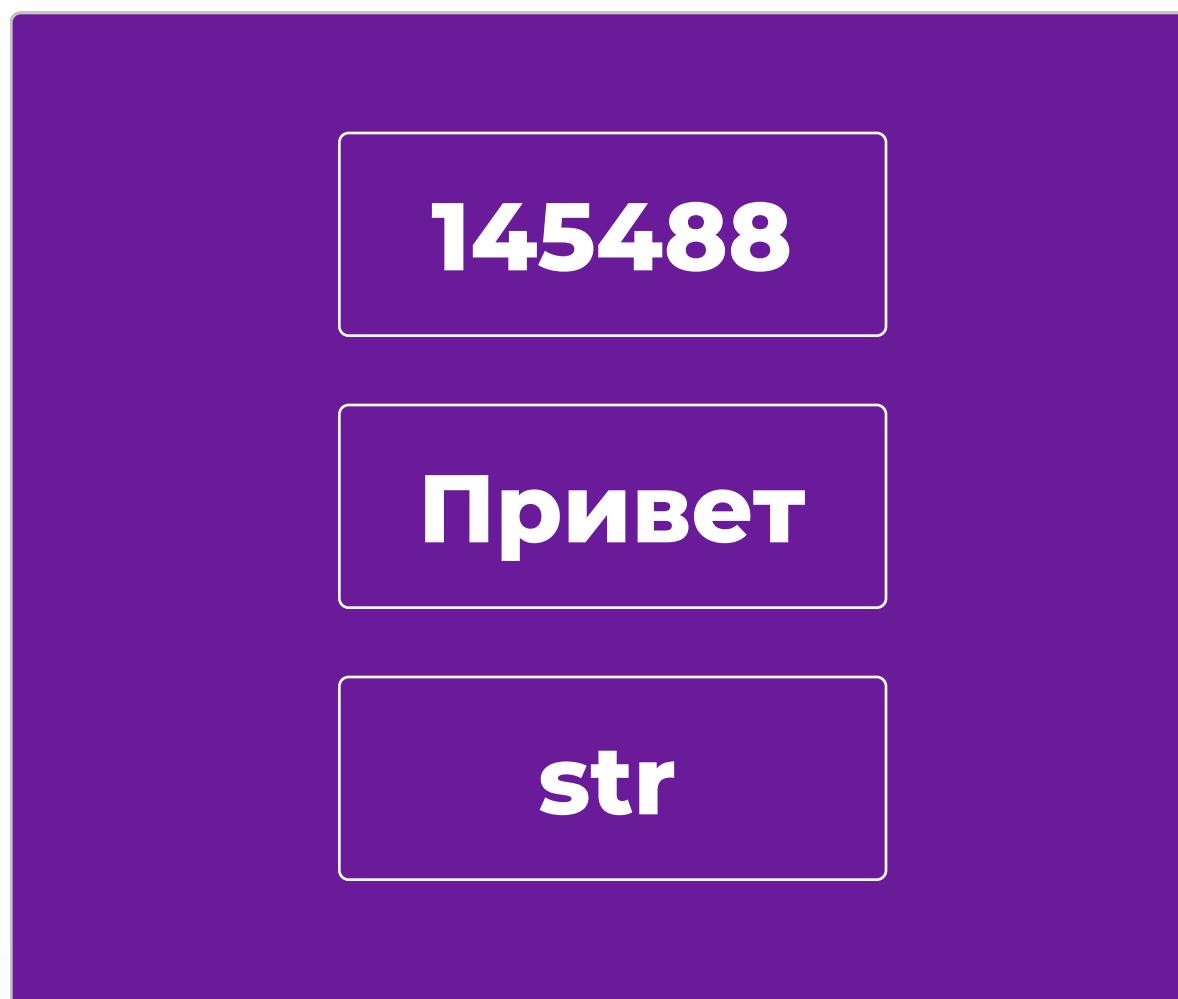
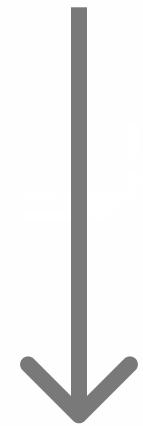
```
my_set = {"Привет", 5, True}
```

```
my_dict = {  
    "ключ_1": "Привет",  
    "ключ_2": 5,  
    "ключ_3": True,  
}
```

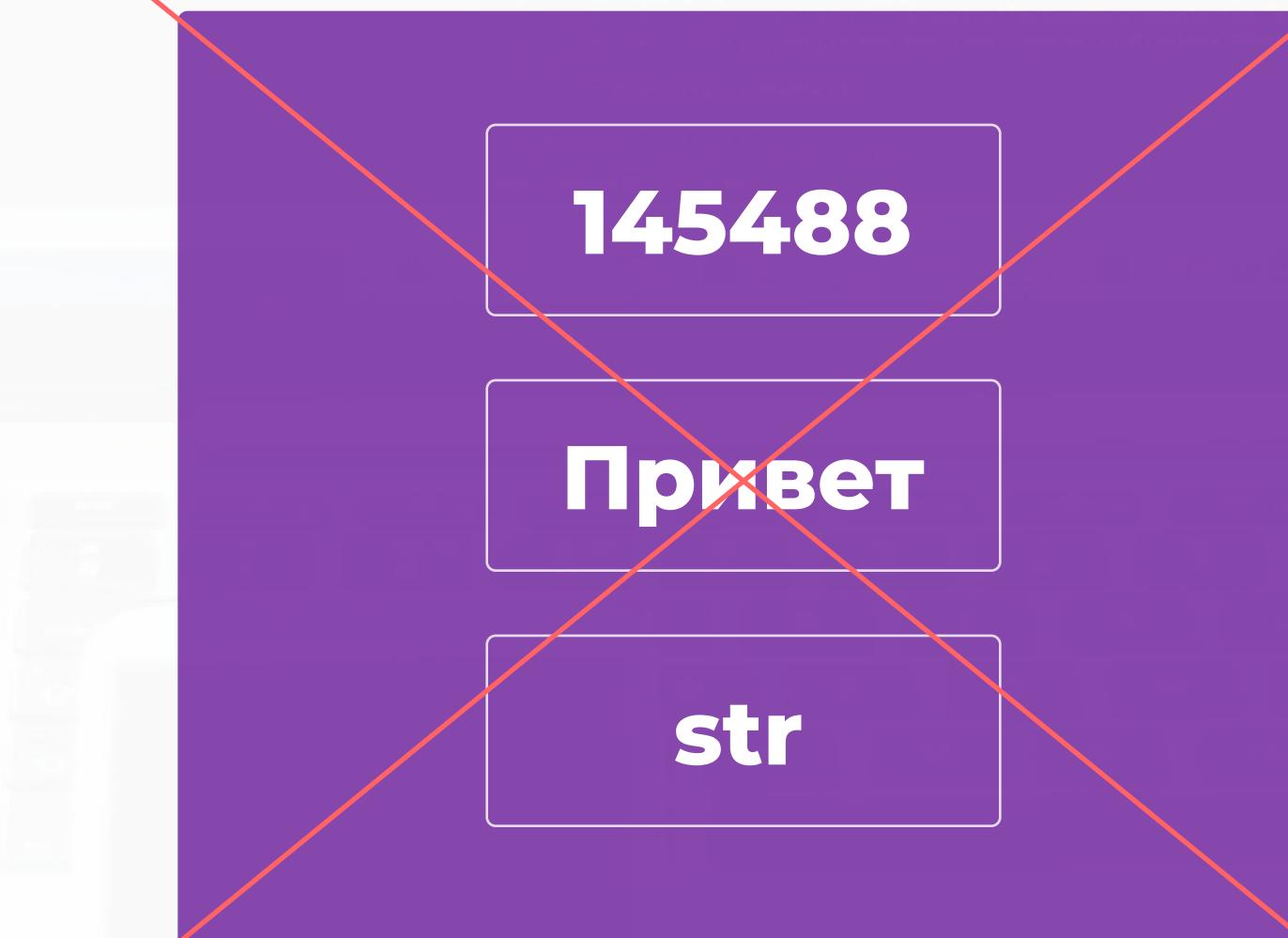
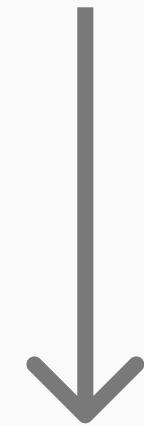
# Python

## Типы данных и работа с памятью. Неизменяемый объект

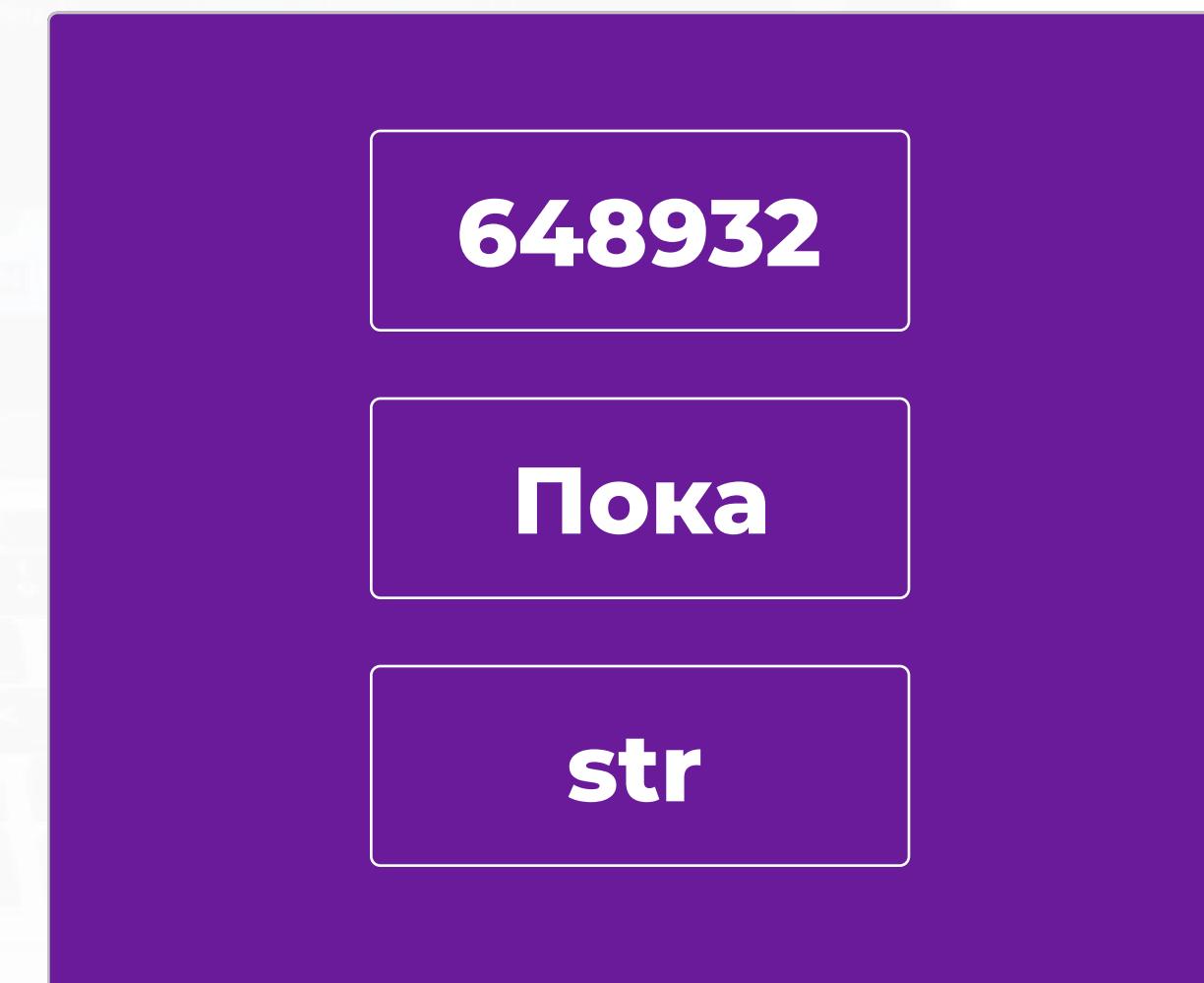
```
my_str = "Привет"
```



```
my_str = "Пока"
```



**Новый объект**

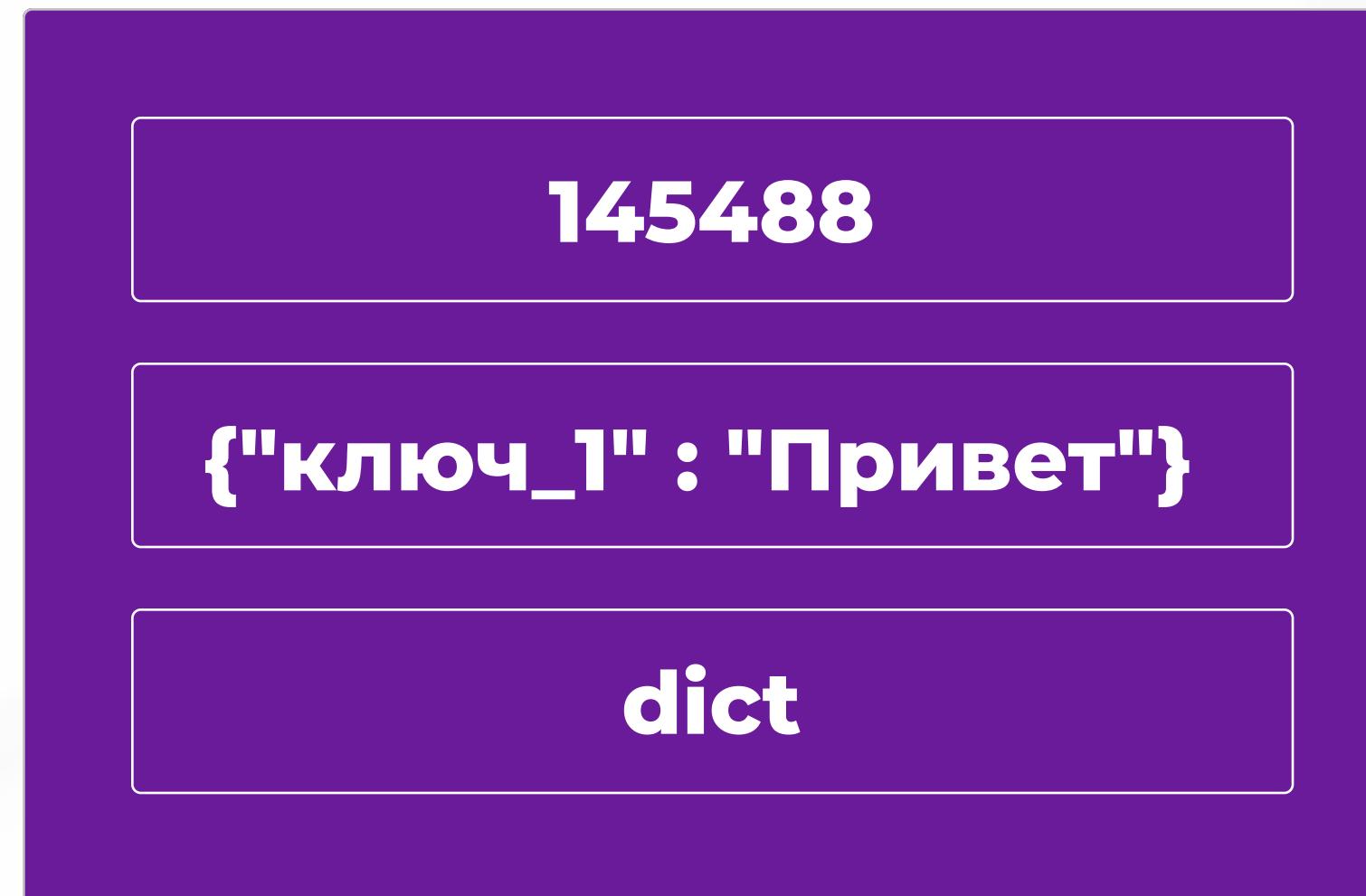
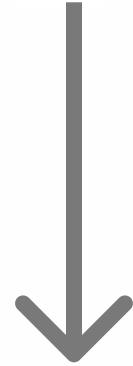


# Python

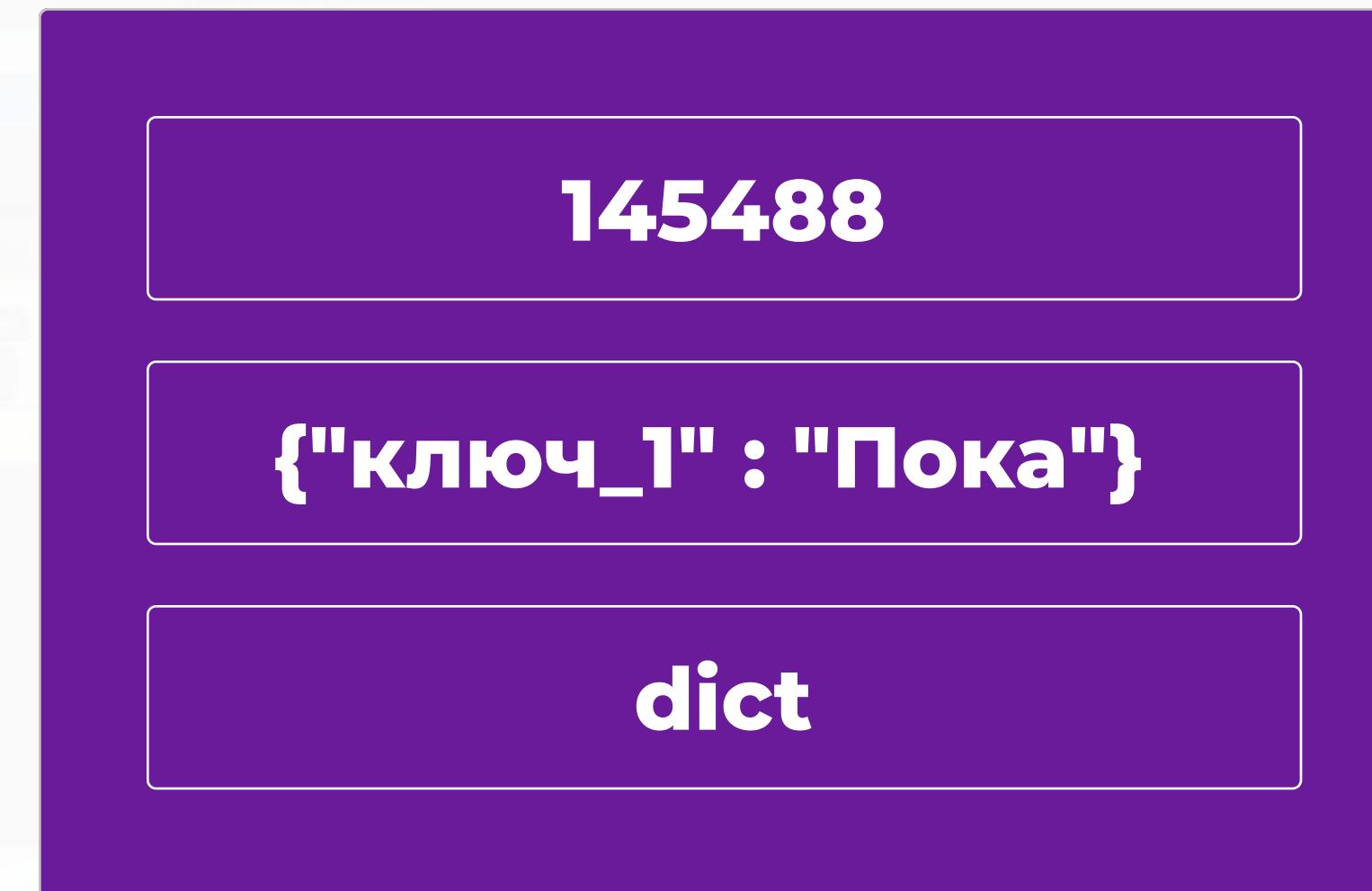
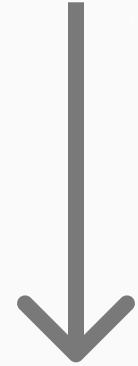
## Типы данных и работа с памятью. Изменяемый объект



```
my_dict = {  
    "ключ_1" : "Привет"  
}
```



```
my_dict = {  
    "ключ_1" : "Пока"  
}
```

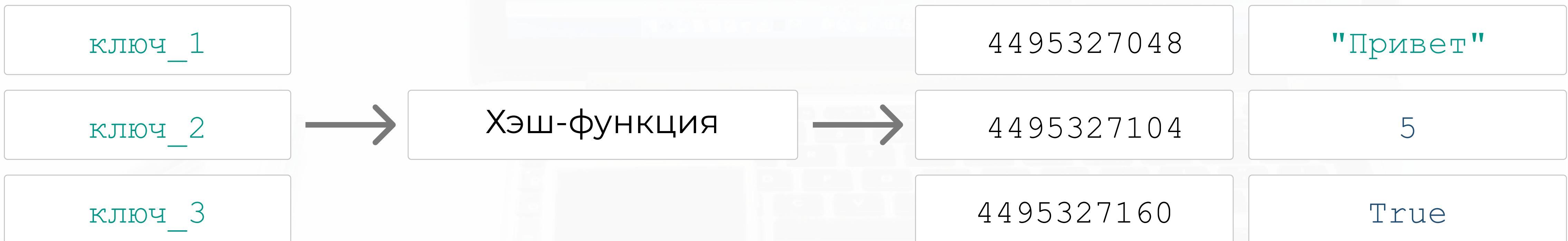


```
my_dict = {  
    "ключ_1": "Привет",  
    "ключ_2": 5,  
    "ключ_3": True,  
}
```

Нотация Big O

Как **количество операций** выполняемого алгоритма  
зависит от **количество входных данных?**

$O(1)$  - константная сложность



<https://habr.com/ru/company/otus/blog/448350/>

Сложение

```
account = account + money
```

Вычитание

```
apples = apples - 5
```

Умножение

```
apples = apples * 5
```

Деление

```
pie = pie / 5
```

Возведение в степень

```
interest = interest ** 1.12
```

# Python

## Операции. Короткая запись



Сложение

```
account += money
```

Вычитание

```
account -= money
```

Умножение

```
apples *= 5
```

Деление

```
pie /= 4
```

Возведение в степень

```
interest **= 1.12
```

# Python

## Операции. Деление



С остатком

Целочисленное (без остатка)

По модулю (остаток от деления)

`7 / 2 # будет 3.5`

`7 // 2 # будет 3`

`7 % 2 # будет 1`

# Python

## Операции. Деление с остатком



Всё в памяти представлено в бинарном виде

```
10010.0100001011010000111  
# 18.261
```

Некоторые выражения могут быть  
представлены в виде бесконечной дроби

```
1 / 3  
# 0.333333333333...
```

Как их хранить в памяти?

При переводе из двоичного формата  
в десятичный накапливается погрешность  
округления

```
print(0.1 + 0.1 + 0.1)  
# 0.3000000000000004 😱
```

Как работать с делением?

**decimal** – десятичная арифметика с фиксированной и плавающей запятой

**fractions** – рациональные числа

**Пакеты:** ScyPi, NumPy, Pandas, ...

```
getcontext().prec = 6
print(
    Decimal(0.1) +
    Decimal(0.1) +
    Decimal(0.1)
)
# 0.300000
```

```
getcontext().prec = 6
print(
    Fraction(1, 10) +
    Fraction(1, 10) +
    Fraction(1, 10)
)
# 3/10
```

<https://docs.python.org/3/tutorial/floatingpoint.html>

Строгое

равенство

`== # a == b`

неравенство

`!= # a != b`

больше

`> # a > b`

меньше

`< # a < b`

Нестрогое

больше или равно

`>= # a >= b`

меньше или равно

`<= # a <= b`

Тип переменной не объявляется – не нужно указывать объем выделяемой памяти.

Python работает с целыми числами любого размера! 😮 # пока хватает памяти 😊

Информация о числе хранится в виде расширяемого массива.

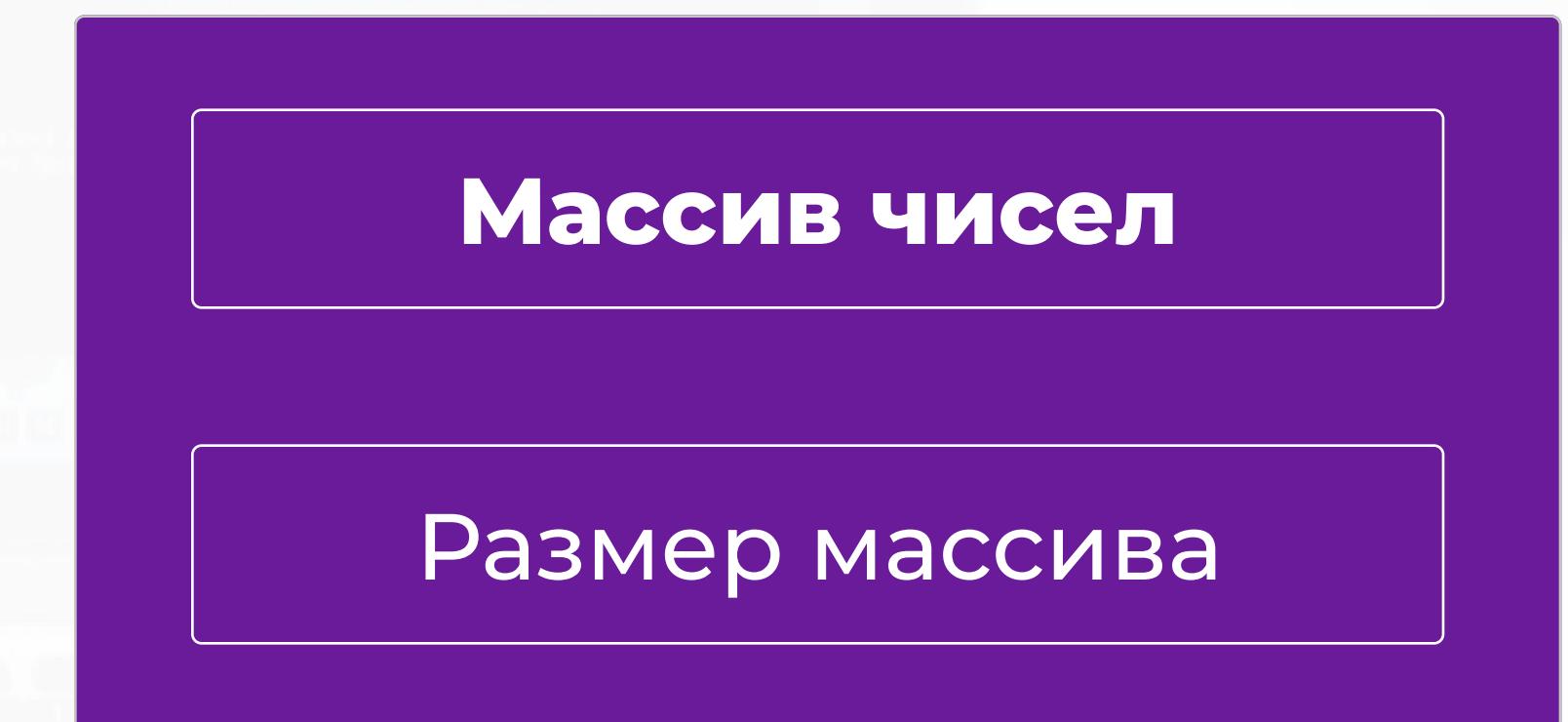
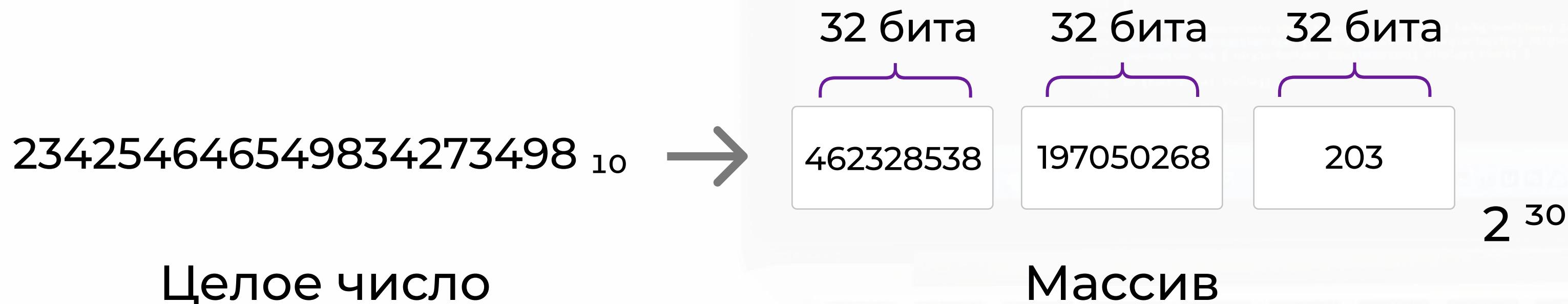
Сложение и вычитание реализовано по принципам школьной математики (с переносом).

Умножение основано на алгоритме Карацубы А. А. (советский математик).

```
print(5 ** 100000) # 5100000
# 100099890379869416681626471319330624849934750830578004920283338007...
```

Целое число имеет произвольную точность, так как хранится в расширяемом массиве

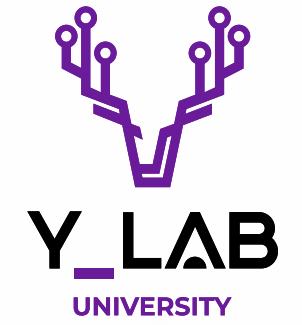
Значения хранятся в массиве с основанием 2 в степени 30



$$462328538 \times (2^{30})^0 + 197050268 \times (2^{30})^1 + 203 \times (2^{30})^2 = 234254646549834273498$$

# Python

## Операции. Форматирование строк – метод "format"



### Именованные метки

```
"Привет, {friend}! Это  
{me}.".format(  
    friend="Петя",  
    me="Вася"  
)  
# Привет, Петя! Это Вася.
```

```
"Привет, {1}! Это {0}.".format(  
    "Вася",  
    "Петя"  
)  
# Привет, Петя! Это Вася.
```

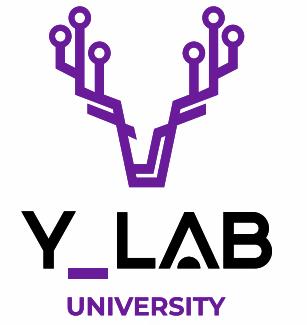
### Позиционные метки

```
"Привет, {}! Это {}".format(  
    "Петя",  
    "Вася"  
)  
# Привет, Петя! Это Вася.
```

### Без меток

# Python

## Операции. Форматирование строк – "f-строки"



Подстановка переменных

```
friend = "Петя"  
me = "Вася"  
  
f"Привет, {friend}! Это  
{me}."  
# Привет, Петя! Это Вася.
```

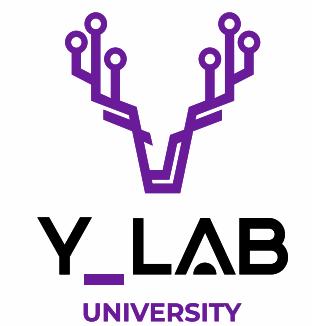
Форматирование

```
from datetime import datetime  
  
print(f"Текущее время {datetime.now():%d.%m.%Y %H:%M:  
%S}")  
# Текущее время 20.07.2021 19:25:45  
  
a = 5  
b = 10  
  
print(f"a + b = {a + b}")  
# a + b = 10
```

Выполнение операций

# Python

## Операции. Многострочное форматирование строк



```
print(  
    """  
Привет, Петя! Это Вася.  
Как у тебя дела?  
"""  
)  
  
#  
#    Привет, Петя! Это Вася.  
#    Как у тебя дела?  
#
```

```
friend = "Петя"  
me = "Вася"  
  
print(  
    f"""  
Привет, {friend}! Это {me}.  
Как у тебя дела?  
"""  
)  
  
#  
#    Привет, Петя! Это Вася.  
#    Как у тебя дела?  
#
```

### Объединение строк

```
friend = "Петя"  
me = "Вася"  
  
print("Привет, " + friend + "! Это " + me + ".")  
# Привет, Петя! Это Вася.
```

```
-----  
my_str = "Привет, Петя! Это Вася. " \  
      "Как у тебя дела?"  
print(my_str)  
# Привет, Петя! Это Вася.
```

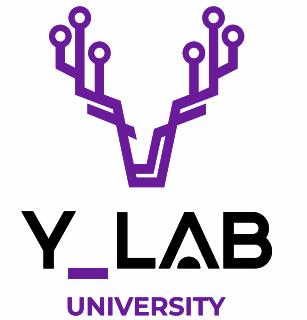
```
-----  
my_str = (  
    "Привет, Петя! Это Вася. "  
    "Как у тебя дела?"  
)  
print(my_str)  
# Привет, Петя! Это Вася.
```

- Объявление переменных
- Математические операции
- Форматирование строк

[Ссылка на практику](#)

# Python

## Условия



**Если** <выполняется условие>,  
**то** <выполнение действий>,  
**иначе** (если условие не выполняется)  
<выполнение других действий>

**if** условие: } обязательно  
двоеточие

команды

4 пробела

отсутствие символов  
окончания строки

**else:**

команды

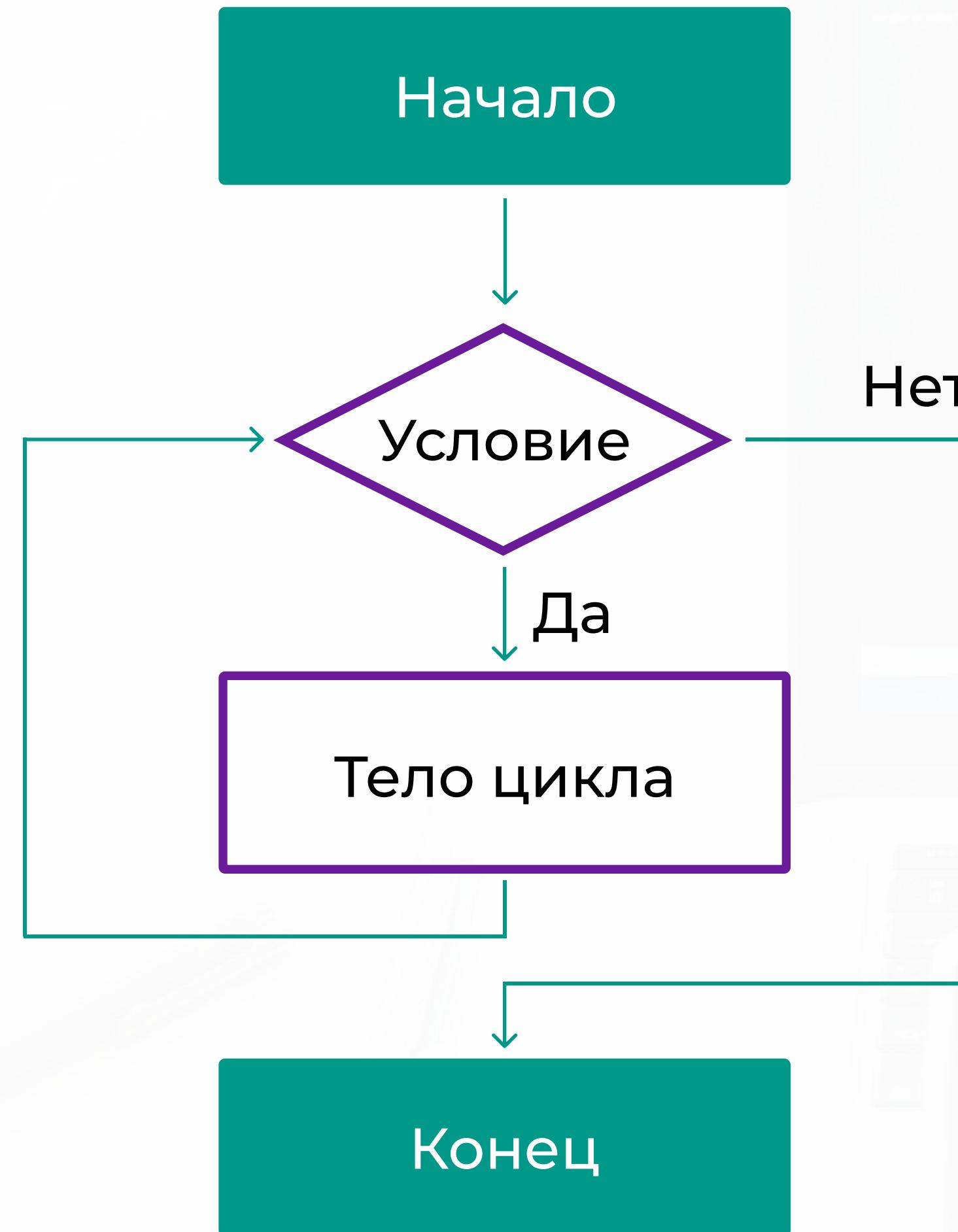
```
if username == 'Миша':  
    print('Привет!')  
  
else:  
    print('Приятно познакомиться.')  
  
  
if username == 'Миша':  
    print('Привет!')  
  
elif username == 'Jack':  
    print('Hello!')  
  
else:  
    print('Приятно познакомиться.')
```



```
for number in range(10):
    print(number, end=' ')
# 0 1 2 3 4 5 6 7 8 9

my_friends = ['Катя', 'Ваня', 'Петя']

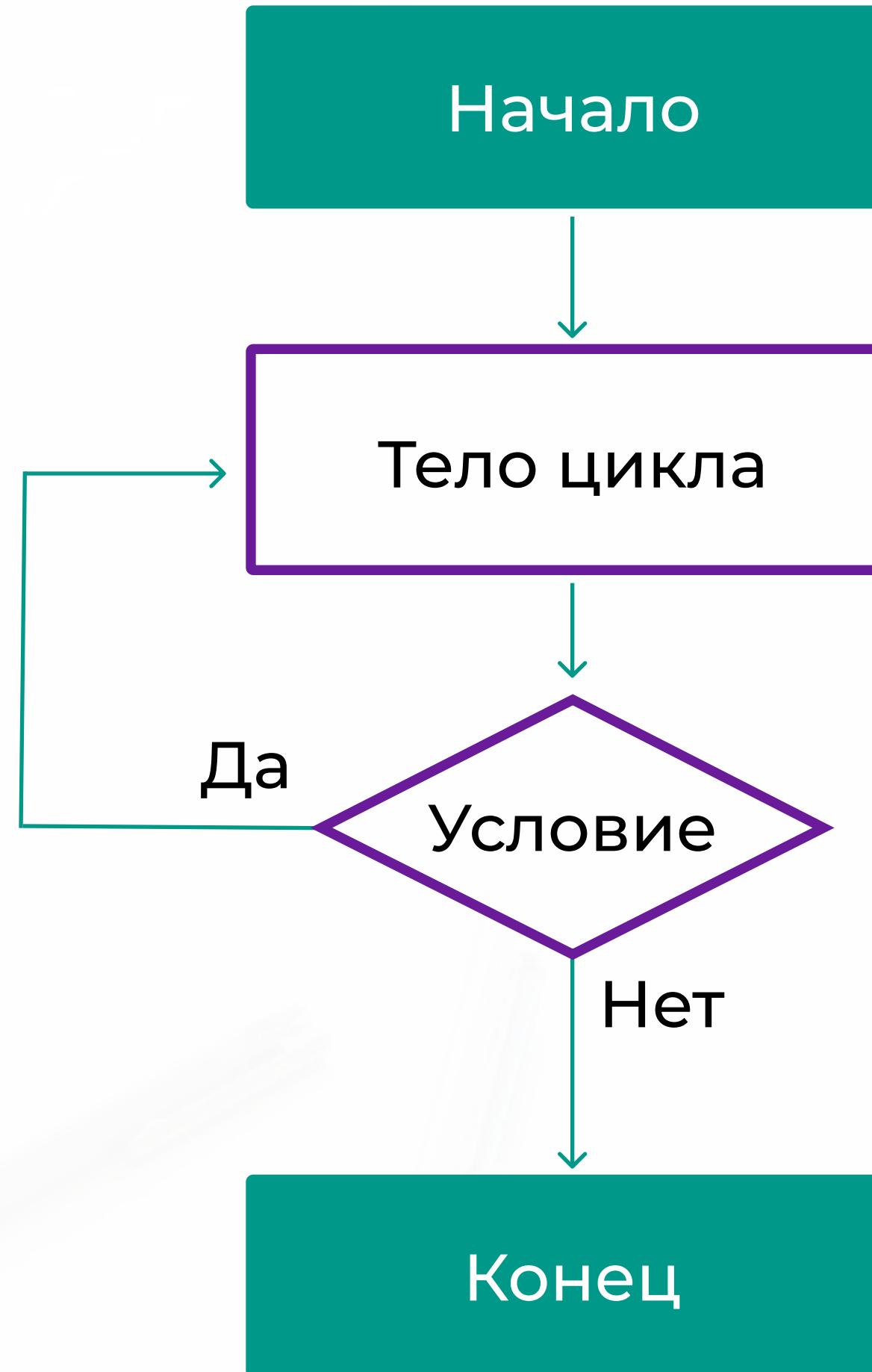
for friend in my_friends:
    print(friend, end=' ')
# Катя Ваня Петя
```



```
money = 1000

while money:
    money -= 100
    print(f'Осталось {money} руб.')

# Осталось 900 руб.
# Осталось 800 руб.
# Осталось 700 руб.
# Осталось 600 руб.
# Осталось 500 руб.
# Осталось 400 руб.
# Осталось 300 руб.
# Осталось 200 руб.
# Осталось 100 руб.
# Осталось 0 руб.
```



```
credit = 1000

while True:
    credit -= 200
    print(f'Баланс {credit} руб.')

    if credit <= 0:
        break

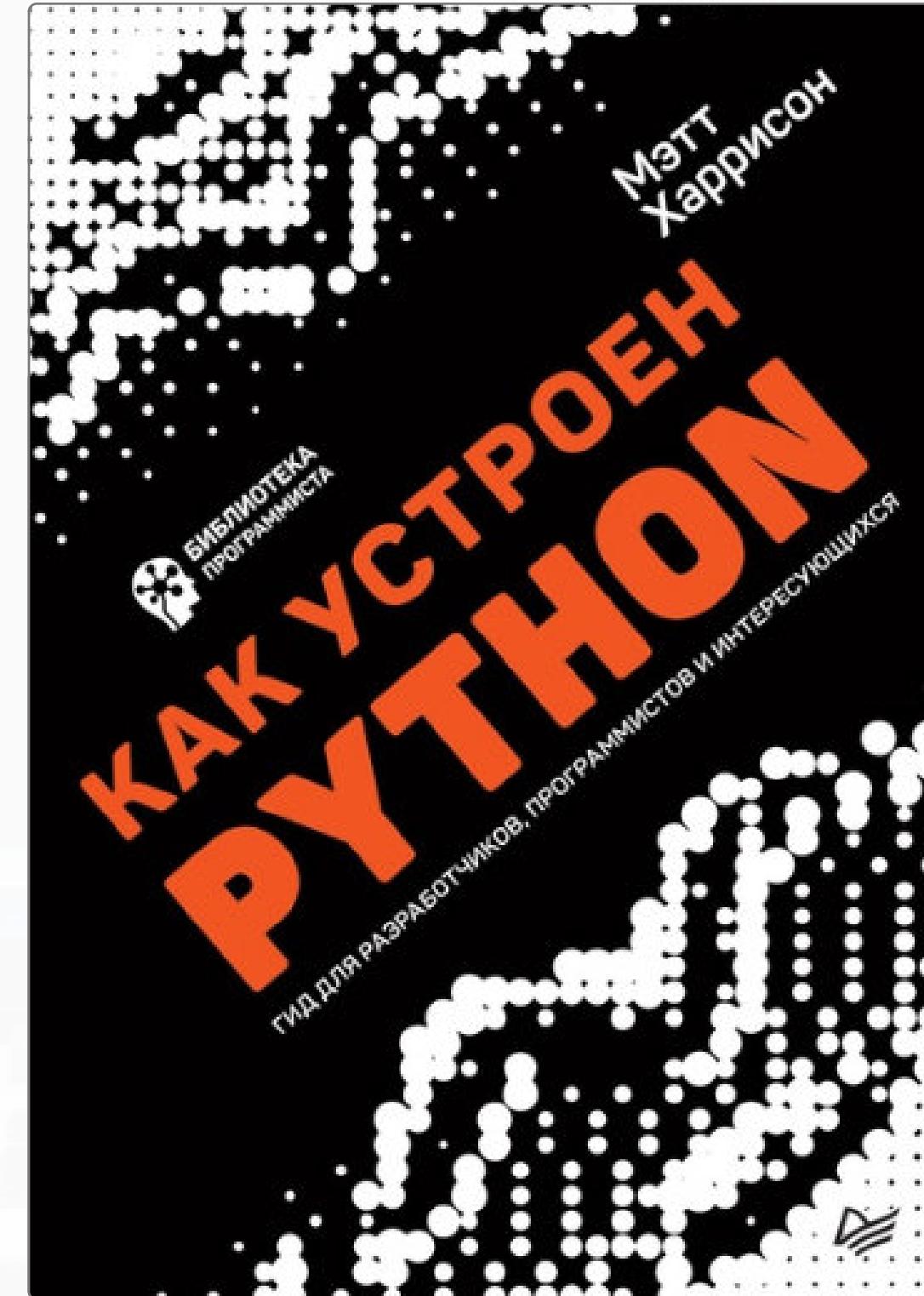
# Баланс 800 руб.
# Баланс 600 руб.
# Баланс 400 руб.
# Баланс 200 руб.
# Баланс 0 руб.
```

- Условия
- Арифметический цикл
- Цикл с предусловием
- Цикл с постусловием

[Ссылка на практику](#)

# Python

## Список материалов для изучения



**Мэтт Харрисон: Как устроен Python**

Гид для разработчиков, программистов и интересующихся

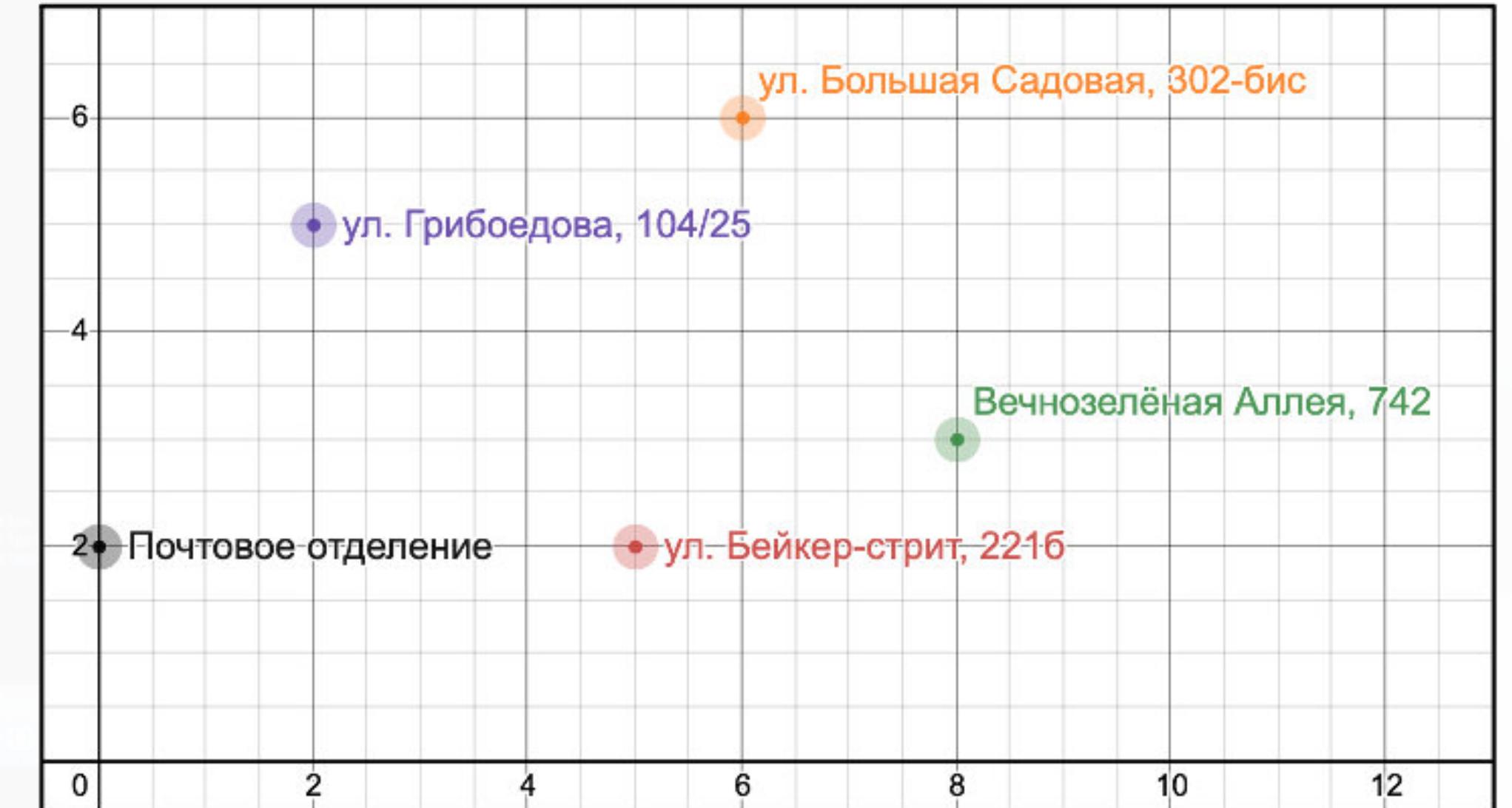
# Python

## Домашнее задание

Разработать программу для вычисления кратчайшего пути для почтальона.

Почтальон выходит из почтового отделения, обезжает всех адресатов для вручения посылки и возвращается обратно в почтовое отделение.

Необходимо найти кратчайший маршрут для почтальона.



Количество операций =  $(n - 1)!$

$(5 - 1)! = 4! = 1 * 2 * 3 * 4 = 24$  (маршрута)

<https://github.com/mnv/python-basics>

Достили ли целей вебинара?

Что запомнилось / понравилось?

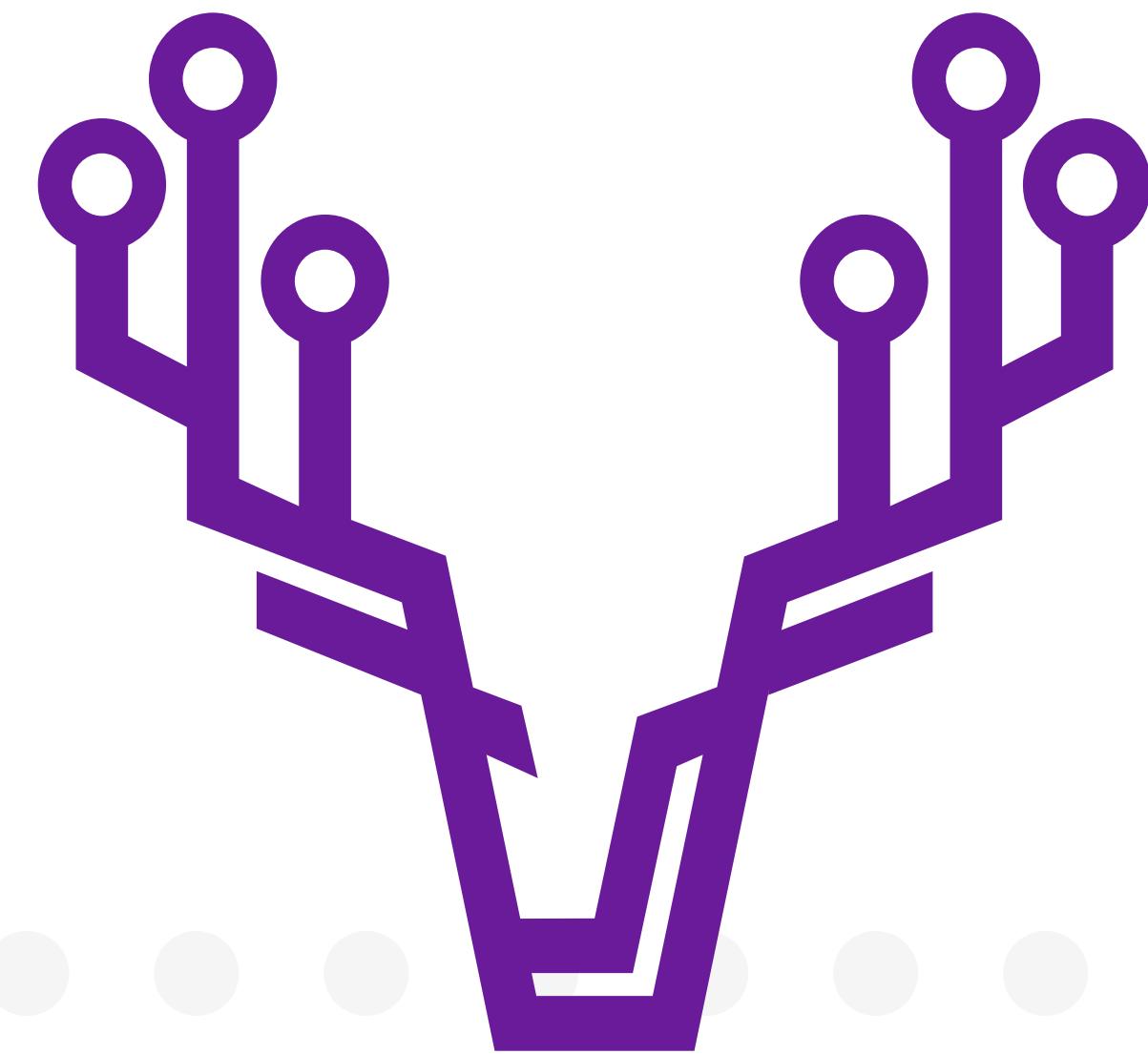
# Обратная связь

Пожалуйста, пройдите опрос о занятии.  
Нам очень важно ваше мнение!

Опрос о занятии



Спасибо за  
внимание!



Y-LAB  
UNIVERSITY