

Погружение в Python



Погружение в Python

Урок 1
Основы Python





Знакомство и содержание курса





Алексей Петренко

Python developer

- 🌟 Python-developer, Team Leader
- 🌟 Опыт программирования более 20 лет
- 🌟 Разрабатывал IT-решения для Министерства обороны РФ
- 🌟 Преподаватель GeekBrains с 2018 года



План курса



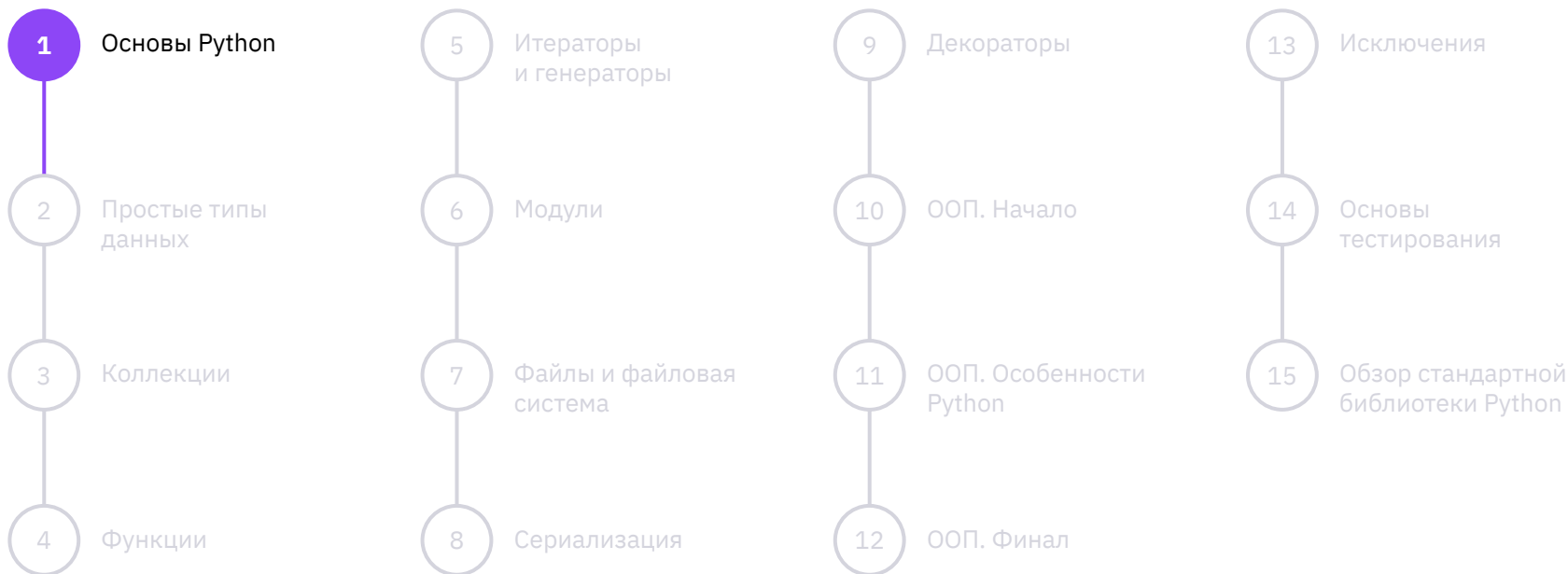


Содержание урока





План курса





Что будет на уроке сегодня

- 📌 Установка и настройка Python
- 📌 Повторим основы синтаксиса языка Python
- 📌 Ветвление в Python
- 📌 Циклы в Python





Установка и настройка Python



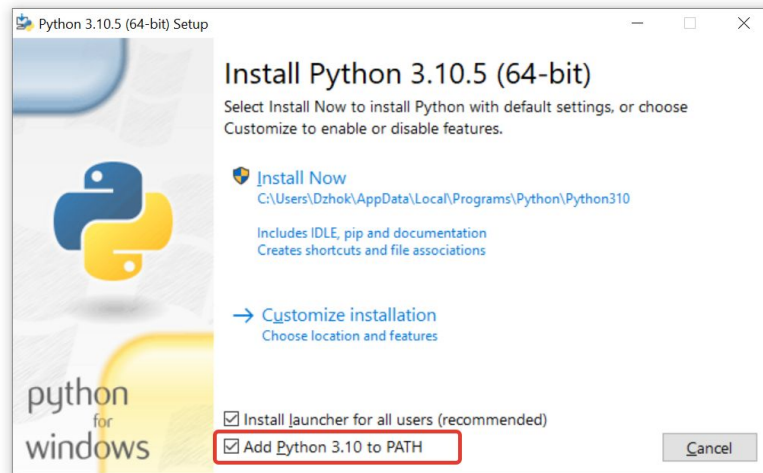


Установка Python

Для работы нам понадобится актуальная версия Python

- Переходим на официальный сайт Python в раздел «Загрузки» <https://www.python.org/downloads/>
- Выбираем вашу версию операционной системы
- Скачиваем установочный дистрибутив
- Запускаем установку, отвечаем на вопросы и ждём завершения
- Проверяем установку командой:

```
python --version  
python3 --version
```





Создание виртуального окружения



Подготовка к созданию окружения

```
mkdir new_project  
cd new_project
```



Создание окружения

```
python -m venv venv — для Windows;  
python3 -m venv venv — для Linux и MacOS.
```



Активация окружения

```
venv\Scripts\activate — для Windows;  
source venv/bin/activate — для Linux и MacOS.
```



Выход из окружения

```
deactivate
```





Работа с pip

Package Installer for Python — система управления пакетами, которая используется для установки и управления программными пакетами, написанными на Python.

- `pip install requests`
- `pip freeze`
- `pip freeze > requirements.txt`
 - `more requirements.txt`
- `pip install -r requirements.txt`

```
Командная строка

(venv) C:\Users\Dzhok\new_project>pip install requests
Collecting requests
  Downloading requests-2.28.1-py3-none-any.whl (62 kB)
----- 62.8/62.8 KB 835.0 KB/s eta 0:00:00
Collecting urllib3<1.27,>=1.21.1
  Downloading urllib3-1.26.10-py2.py3-none-any.whl (139 kB)
----- 139.2/139.2 KB 2.1 MB/s eta 0:00:00
Collecting charset-normalizer<3,>=2
  Downloading charset-normalizer-2.1.0-py3-none-any.whl (39 kB)
Collecting certifi==2017.4.17
  Downloading certifi-2022.6.15-py3-none-any.whl (160 kB)
----- 160.2/160.2 KB 2.4 MB/s eta 0:00:00
Collecting idna<4,>=2.5
  Downloading idna-3.3-py3-none-any.whl (61 kB)
----- 61.2/61.2 KB 3.2 MB/s eta 0:00:00
Installing collected packages: urllib3, idna, charset-normalizer, certifi, requests
Successfully installed certifi-2022.6.15 charset-normalizer-2.1.0 idna-3.3 requests-2.28.1 urllib3-1.26.10
WARNING: You are using pip version 22.0.4; however, version 22.2 is available.
You should consider upgrading via the 'C:\Users\Dzhok\new_project\venv\Scripts\python.exe -m pip install --upgrade pip'
command.

(venv) C:\Users\Dzhok\new_project>pip freeze
certifi==2022.6.15
charset-normalizer==2.1.0
idna==3.3
requests==2.28.1
urllib3==1.26.10

(venv) C:\Users\Dzhok\new_project>pip freeze > requirements.txt

(venv) C:\Users\Dzhok\new_project>
```



Работа в режиме интерпретатора

Пришло время написать свою первую программу.

И писать её мы будем в режиме интерпретатора. Чтобы активировать интерпретатор выполним команду:

- `python`
- `python3`

```
dzhoker@DESKTOP-D3TAQF: /mnt/c/Users/Dzhok/other_project
(venv) dzhoker@DESKTOP-D3TAQF: /mnt/c/Users/Dzhok/other_project$ python
Python 3.8.10 (default, Mar 15 2022, 12:22:08)
[GCC 9.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> 2+2
4
>>> 8*6
48
>>> 4+3*8-(2+4)
22
>>> _ + 2
24
>>>
```



Арифметические операторы в Python

Основные математические операторы представлены в таблице.

Оператор	Описание	Примеры
+	Сложение	$398 + 20 = 418$
-	Вычитание	$200 - 50 = 150$
*	Умножение	$34 * 7 = 238$
/	Деление	$36 / 6 = 6.0$ $36 / 5 = 7.2$
//	Целочисленное деление	$36 // 6 = 6$ $9 // 4 = 2$ $-9 // 4 = -3$ $15 // -2 = -3$
%	Остаток от деления	$36 \% 6 = 0$ $36 \% 5 = 1$
**	Возведение в степень	$2 ** 16 = 65536$



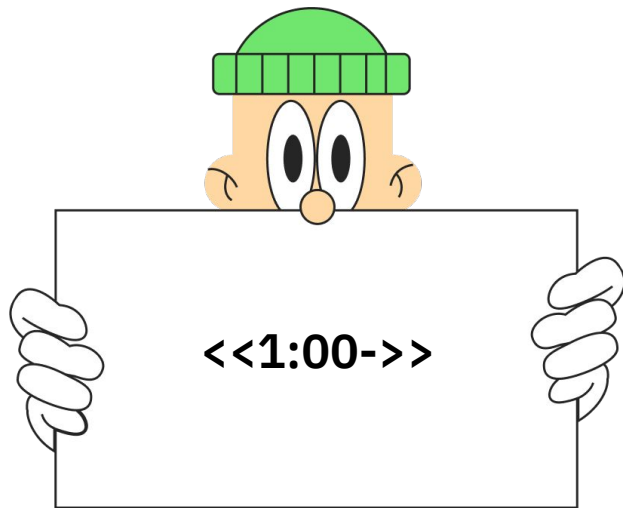
Проверим усвоение материала.
Для ответа на каждый вопрос вам даётся
одна минута. Ответы напишите в чат.





Установка и настройка Python

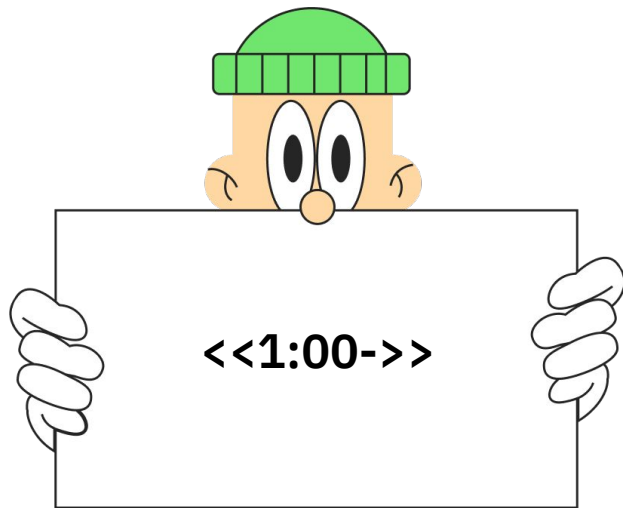
Какая версия Python у вас установлена? Если не установлена, какую версию будете устанавливать?





Установка и настройка Python

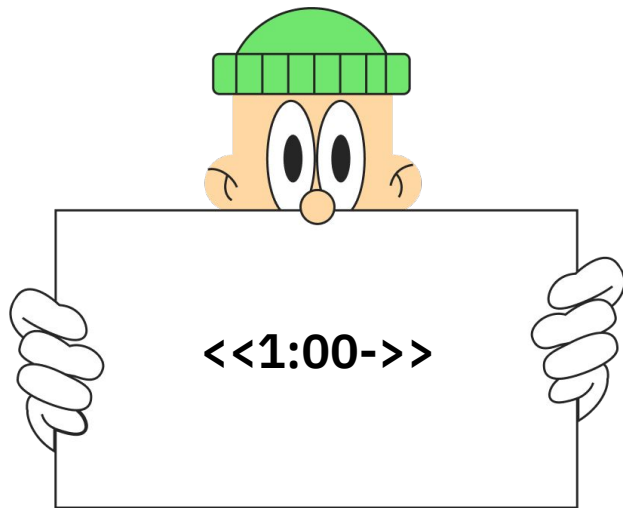
Запомнили ли вы имя файла, в котором сохраняется перечень всех необходимых дополнений для вашего проекта. Как он называется?





Установка и настройка Python

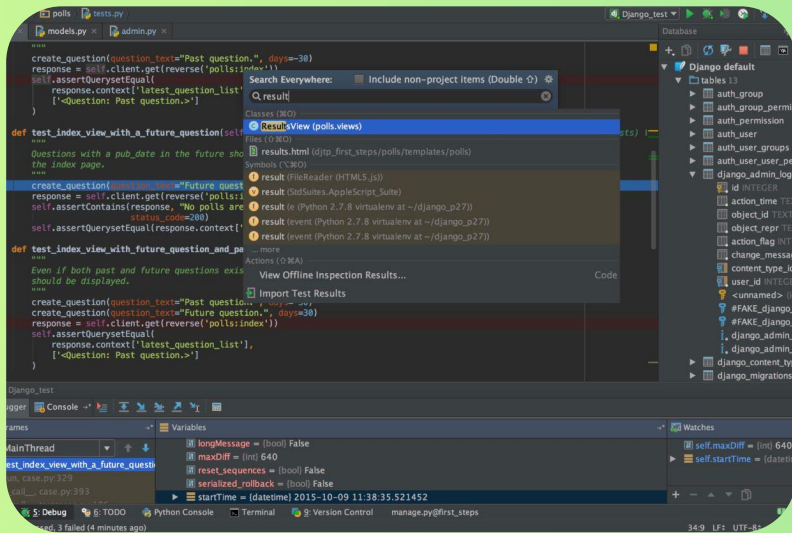
Как получить доступ к результату последней операции в режиме интерпретатора Python?





Повторяем основы Python







Python Style



PEP-8

руководство по стилю



PEP-257

оформление документации и комментариев



"Превед каг тваи дила"





Переменные и требования к именам

Откроем IDE для живого примера, а потом вернёмся к перечню ниже



Верно

```
first_name
```

```
user_1, request
```

```
_tmp_name
```

```
min_step_shift
```



Не верно

```
1_name
```

```
User_1
```

```
0request
```

```
tmpName
```

```
minStep_shift
```



```
ИМЯ
```



Константы

Дополнительных команд для создания констант в языке Python нет!



Создаваемые

```
MAX_COUNT = 1000
```

```
ZERO = 0
```

```
DATA_AFTER_DELETE = 'No data'
```

```
DAY = 60 * 60 * 24
```



Встроенные

True — истина

False — ложь

None — ничего



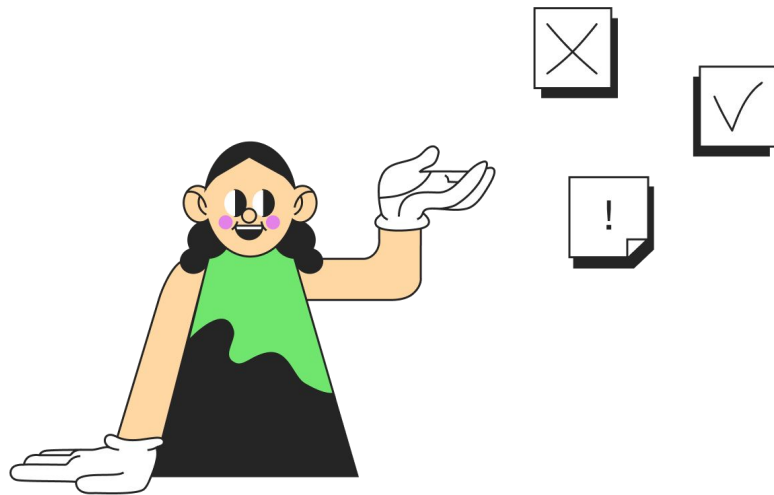
Функция `id()`

Функция `id()` возвращает адрес объекта в оперативной памяти вашего компьютера

```
a = 5  
print(id(a))
```

```
a = "hello world"  
print(id(a))
```

```
a = 42.0 * 3.141592 / 2.71828  
print(id(a))
```



Зарезервированные слова

Базовый синтаксис языка Python образуют нижеперечисленные зарезервированные слова:

```
False, None, True, and, as, assert, async,  
await, break, class, continue, def, del, elif,  
else, except, finally, for, from, global, if,  
import, in, is, lambda, nonlocal, not, or,  
pass, raise, return, try, while, with, yield.
```

А также `case` и `match` начиная с версии Python 3.10.





Ввод и вывод данных



Вывод, функция print()

```
print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
```



Ввод, функция input()

```
result = input([prompt])
```

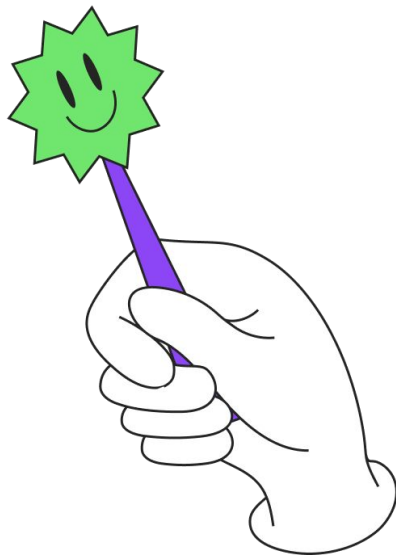
Готовьтесь, будет много живого кода



Антипаттерн «магические числа»

Магическое число — это оперирование явно указанными в коде коэффициентами (как правило целочисленными), значение и смысл которых знает только автор программы. Считается плохим стилем с 1960-х годов!

```
age = float(input('Ваш возраст: '))  
how_old = age - 18  
print(how_old, "лет назад ты стал  
совершеннолетним")
```





Проверим усвоение материала.
Для ответа на каждый вопрос вам даётся
две минуты. Ответы напишите в чат.

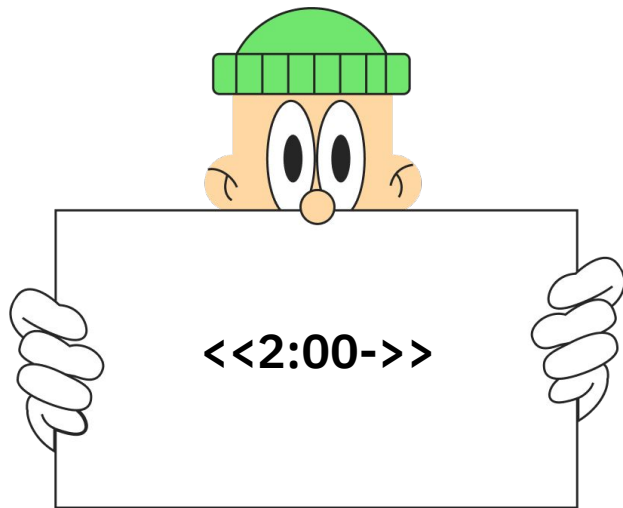




Повторяем основы Python

Какое из слов лишнее и что означают остальные слова?

- `True`
- `NaN`
- `False`
- `None`

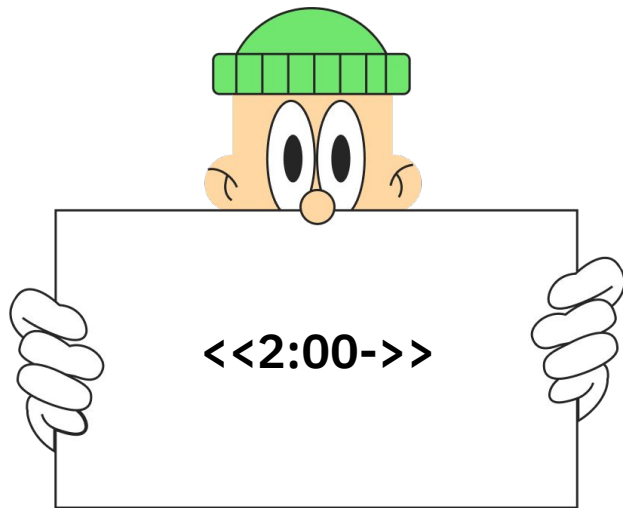




Повторяем основы Python

Какие имена переменных плохие,
а какие правильные и почему?

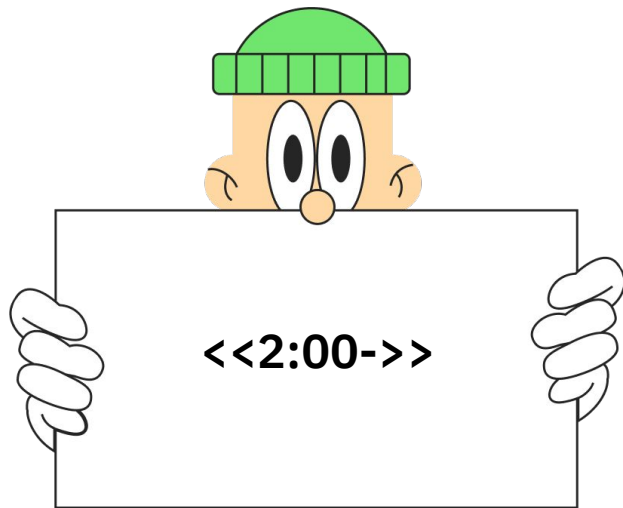
- `name`
- `1_year`
- `_hello`
- `id`
- `MAX_DATE`
- `zdorove`





Повторяем основы Python

Что вы можете рассказать про магические числа?





Ветвления в Python





Операции сравнения

В Python доступны шесть операций сравнения:

- «==» — равно
- «!=» — не равно
- «>» — больше
- «<=» — меньше или равно
- «<» — меньше
- «>=» — больше или равно





Зарезервированные слова

Разберём на примерах

- Если, `if`
- Иначе, `else`
- Ещё если, `elif`
- Выбор из вариантов, `match` и `case`





Логические конструкции

В Python доступны три логических оператора:

- `and` — логическое умножение «И»;
- `or` — логическое сложение «ИЛИ»;
- `not` — логическое отрицание «НЕ».

first	second	first and second	first or second	not first
True	True	True	True	False
False	True	False	True	True
True	False	False	True	-
False	False	False	False	-



Три дополнения о ветвлении в Python



Ленивый if

Проверяем дальше, только
если в этом есть смысл



Проверка на вхождение

Ищем вхождение элемента
в коллекции используя
зарезервированное слово `in`



Тернарный оператор

Сокращённая запись `if-else`
в одну строку

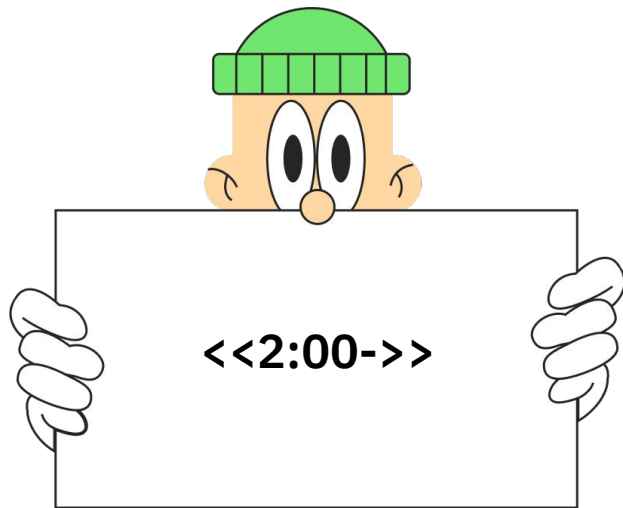


Попробуйте прочитать код и выбрать
верный вариант вывода на печать без его запуска.
У вас две минуты на каждую пару значений.



Ветвления в Python

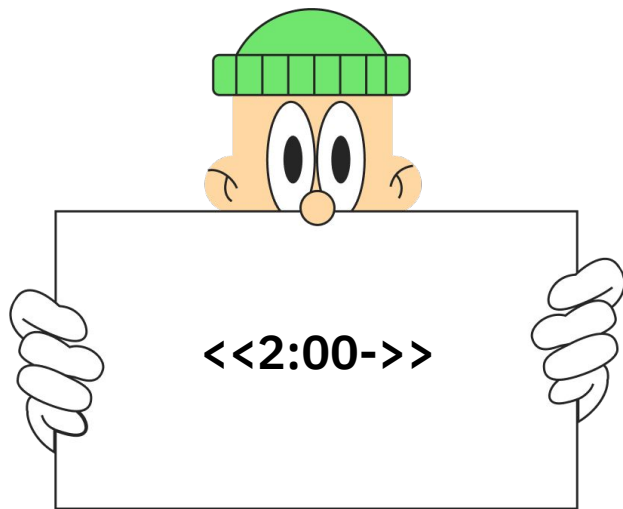
```
num = 42
name = 'Bob'
if num > 30:
    if num < 50:
        print('Вариант 1')
    elif name > 'Markus':
        print('Вариант 2')
    else:
        print('Вариант 3')
elif name < 'Markus':
    print('Вариант 4')
elif num != 42:
    print('Вариант 5')
else:
    print('Вариант 6')
```





Ветвления в Python

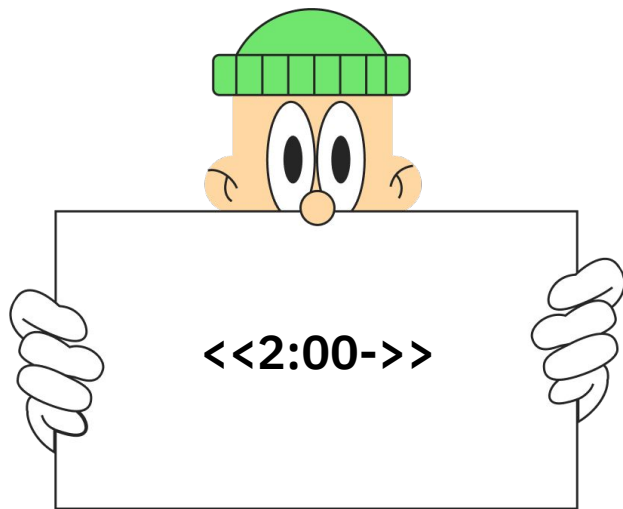
```
num = 64
name = 'Bob'
if num > 30:
    if num < 50:
        print('Вариант 1')
    elif name > 'Markus':
        print('Вариант 2')
    else:
        print('Вариант 3')
elif name < 'Markus':
    print('Вариант 4')
elif num != 42:
    print('Вариант 5')
else:
    print('Вариант 6')
```





Ветвления в Python

```
num = 7
name = 'Neo'
if num > 30:
    if num < 50:
        print('Вариант 1')
    elif name > 'Markus':
        print('Вариант 2')
    else:
        print('Вариант 3')
elif name < 'Markus':
    print('Вариант 4')
elif num != 42:
    print('Вариант 5')
else:
    print('Вариант 6')
```





Циклы в Python



Логический цикл `while`

Разберём на примерах

- Начало цикла, `while`
- Возврат в начало цикла, `continue`
- Досрочное завершение цикла, `break`
- Действие после цикла, `else`





Цикл итератор `for in`

Разберём на примерах

- Начало цикла, `for in`
- Цикл по целым числам, он же арифметический цикл, функция `range()`
- Цикл с нумерацией элементов, функция `enumerate()`





Чуть больше об арифметическом цикле



Варианты функции range()

`range(stop)` — перебираем значения от нуля до `stop` исключительно с шагом один

`range(start, stop)` — перебираем значения от `start` включительно до `stop` исключительно с шагом один

`range(start, stop, step)` — перебираем значения от `start` включительно до `stop` исключительно с шагом `step`.



Имена переменных в цикле

```
count = 10
for i in range(count):
    for j in range(count):
        for k in
range(count):
            print(i, j, k)
```



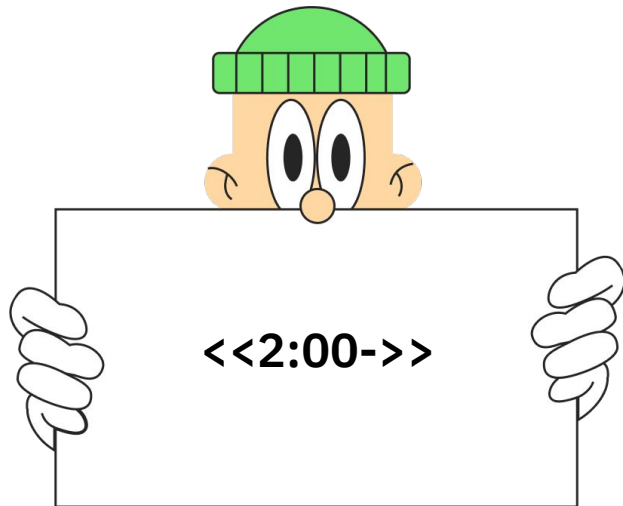
Попробуйте мысленно пройти по коду
и напишите в чат финальное значение переменной
data. У вас две минуты на каждую пару значений.





Циклы в Python

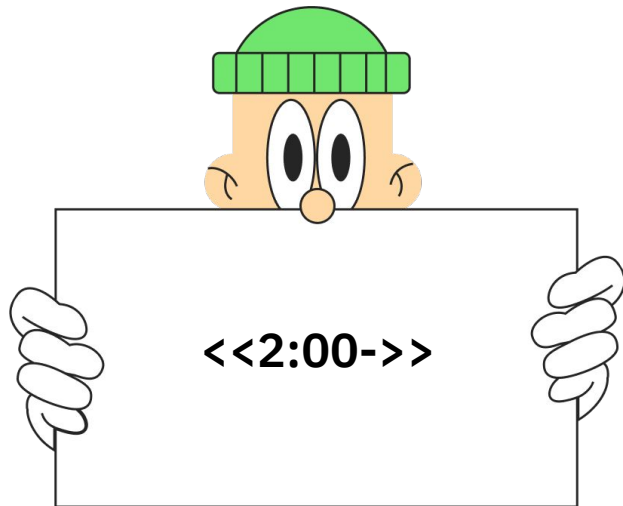
```
data = 0
while data < 100:
    data += 2
    if data % 40 == 0:
        break
print(data)
```





Циклы в Python

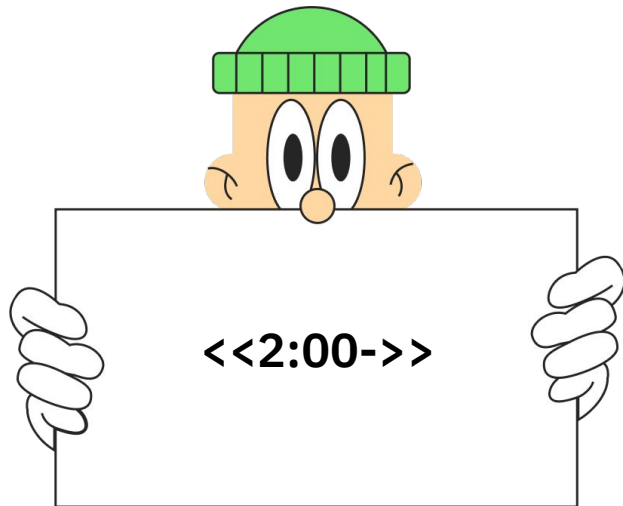
```
data = 0
while data < 100:
    data += 3
    if data % 40 == 0:
        break
else:
    data += 5
print(data)
```





Циклы в Python

```
data = 0
while data < 100:
    data += 3
    if data % 19 == 0:
        continue
    data += 1
    if data % 40 == 0:
        break
else:
    data += 5
print(data)
```









Итоги занятия



На этой лекции мы

-  Разобрали установку и настройку Python. Изучили правила создания виртуального окружения и работу с `pip`.
-  Повторили основы синтаксиса языка Python. Познакомились с рекомендациями по оформлению кода.
-  Изучили способы создания ветвящихся алгоритмов на Python.
-  Разобрались с разными вариантами реализации циклов.

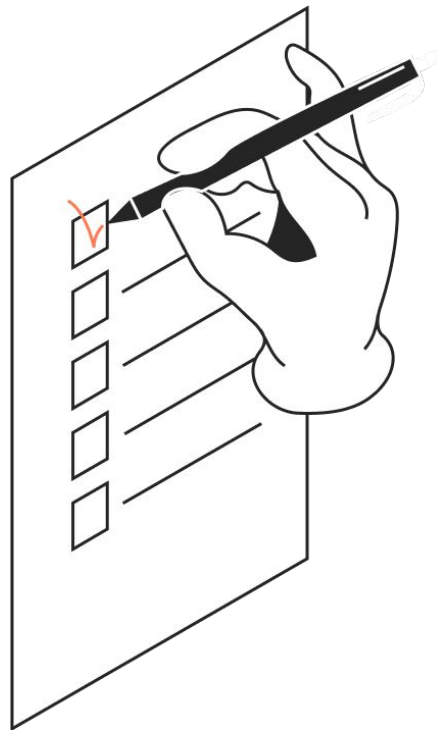




Задание

Для подготовки к семинару вам необходимо установить Python и подготовить вашу любимую IDE для практической работы

- Установите Python на ваш ПК. Убедитесь, что можете работать как в режиме интерпретатора, так и запускать код в файлах `.py`.
- Настройте IDE, в которой планируете выполнять задания на семинарах и практические работы. Проверьте возможность запуска программ средствами IDE.





Спасибо за внимание