

# Программирование на Python

Глеб Пехов

Бэкенд-разработчик на Python



# Проверка связи





## Если у вас нет звука:

- убедитесь, что на вашем устройстве и на колонках включён звук
- обновите страницу вебинара или закройте страницу и заново присоединитесь к вебинару
- откройте вебинар в другом браузере
- перезагрузите компьютер (ноутбук) и заново попытайтесь зайти



## Поставьте в чат:

-  если меня видно и слышно
-  если нет

# Знакомство с дисциплиной



# Глеб Пехов

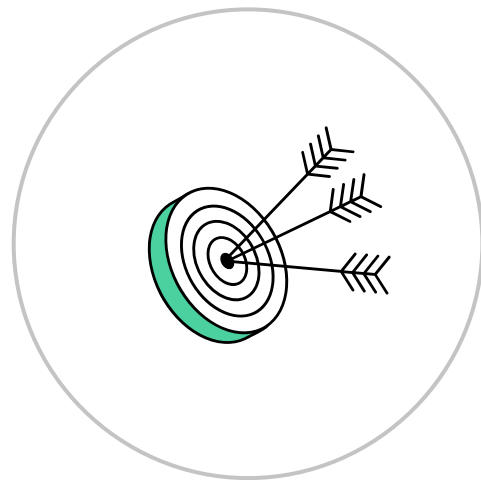
О спикере:

- бэкенд-разработчик на Python
- опыт разработки 4 года
- преподаватель в Нетологии



# Цель курса

Освоить компетенции, которые нужны для развития навыков и знаний в области программирования и кибербезопасности с использованием Python



# Планируемые результаты обучения

- **Знать** основные конструкции языка, принципы объектно ориентированного программирования, специализированные сервисы, библиотеки и модули Python, которые используют в кибербезопасности
- **Уметь** разрабатывать программы на Python для решения практических задач, создавать и использовать функции для структурирования кода, использовать классы в разработке программного кода
- **Владеть** навыками использования Python, чтобы автоматизировать задачи кибербезопасности



# Общая информация о курсе

## Длительность

- 2 модуля
- 1-й курс: 2-й и 3-й модуль

## Объём

- 6 зачётных единиц\* — по 3 зачётные единицы в каждом модуле
- 228 академических часов\*\* — по 114 в каждом модуле

## Аттестация

- Аттестация по дисциплине — экзамен (в 3-м модуле)

\* 1 ЗЕ (зачётная единица) = 38 академических часов

\*\* 1 академический час = 40 минут

# Логика прохождения курса

Программа дисциплины состоит из 15 тем. Каждая тема включает в себя материал для самостоятельного изучения на платформе и вебинары с преподавателем.

## Материалы на платформе

1. Видеолекции
2. Презентации и дополнительные материалы
3. Домашние задания

## Вебинары

Весь материал преподаватель передаёт на занятии или перед ним.

Записи вебинаров и презентации можно будет посмотреть на следующий день после занятия



# Структура курса

## Темы 2-го модуля

№	Название темы	Формат учебного контента
1	Введение в программирование	Видеолекция, вебинар, домашнее задание с самопроверкой
2	Управляющие конструкции и типы данных	Видеолекция, вебинар, домашнее задание с самопроверкой
3	Функции и области видимости	Видеолекция, вебинар, домашнее задание с проверкой преподавателем
4	Работа с файлами и пакетами	Видеолекция, вебинар, домашнее задание с самопроверкой
5	Время и даты в Python	Видеолекция, вебинар, домашнее задание с самопроверкой
6	Классы и объекты (ООП)	Видеолекция, вебинар, домашнее задание с проверкой преподавателем
7	Работа с API	Видеолекция, вебинар, домашнее задание с самопроверкой

# Структура курса

## Темы 3-го модуля

№	Название темы	Формат учебного контента
8	Git — система контроля версий	Видеолекция, домашнее задание с самопроверкой
9	Введение в практическую безопасность	Вебинар
10	Взаимодействие в WWW	Видеолекция, вебинар, домашнее задание с самопроверкой
11	Python для аналитиков ИБ: API и анализ данных	Видеолекция, вебинар, домашнее задание с самопроверкой
12	Python для аналитиков ИБ: эксплойты	Видеолекция, вебинар, домашнее задание с самопроверкой
13	Python для аналитиков ИБ: поиск негативных событий	Вебинар, домашнее задание с самопроверкой
14	Python для аналитиков ИБ: форензика	Вебинар, итоговая работа с проверкой преподавателем
15	Управление системами и ресурсами защиты информации с помощью Python	Вебинар, домашнее задание с самопроверкой

# Аттестация по дисциплине

- 1 Промежуточная аттестация включает:
  - домашние задания, которые проверяет преподаватель
  - домашние задания с самопроверкой
  - тест по темам 2-го модуля
  - итоговую работу по 3-му модулю
- 2 Оценка за дисциплину формируется как сумма оценок с учётом коэффициентов каждого элемента контроля
- 3 Формула расчёта итоговой оценки за курс (2–3-й модуль):

$0,3 \times \text{ДЗ с проверкой} + 0,1 \text{ итоговый тест по 2 модулю} + 0,2 \times \text{ДЗ с самопроверкой} + 0,4 \times \text{Итоговая работа по 3-му модулю} = 1$

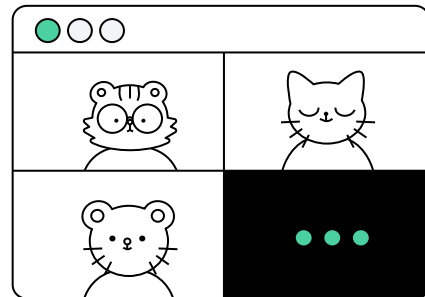
## Распределение по элементам:

№	Элемент контроля	Весовой коэффициент
1	Домашние задания с проверкой преподавателем	0,3
2	Итоговый тест по 2-му модулю	0,1
3	Домашние задания с самопроверкой	0,2
4	Итоговая работа по 3-му модулю	0,4

# Давайте знакомиться

Напишите в чате информацию о себе:

- Как вас зовут, откуда вы?
- Какие у вас ожидания от этой дисциплины?
- Что для вас будет наилучшим результатом обучения?



# Введение в программирование

Глеб Пехов  
Бэкенд-разработчик на Python



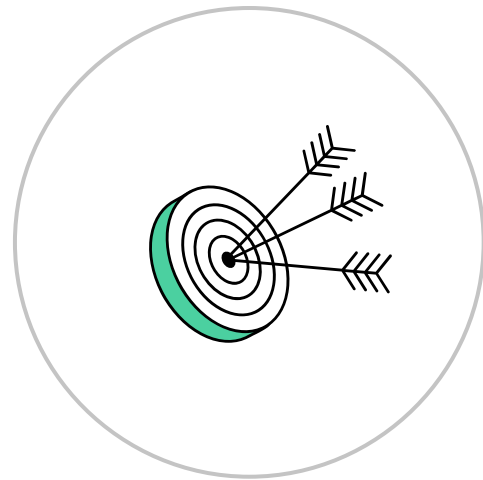
# Правила участия

- 1 Приготовьте блокнот и ручку, чтобы записывать важные мысли и идеи
- 2 Продолжительность вебинара — 80 минут
- 3 Вы можете писать свои вопросы в чате
- 4 Запись вебинара будет доступна в личном кабинете
- 5 Обсуждение можно продолжить в Telegram



# Цели занятия

- Рассмотреть разные языки программирования, разобраться, чем они отличаются и почему существует такое количество языков программирования
- Познакомиться с языком программирования Python
- Узнать различные виды архитектуры приложений и разобраться, чем обусловлен выбор той или иной архитектуры



# План занятия

- 1 Языки программирования
- 2 Язык программирования Python. Преимущества и недостатки
- 3 Современная архитектура приложений



\* Нажмите на нужный раздел для перехода





**Есть ли у вас опыт  
программирования?**

# Языки программирования



1

# Язык программирования

«Программа = алгоритмы + структуры данных»

*Никлаус Вирт*

Язык программирования — средство для реализации программ



Схема алгоритма

# Виды языков программирования

1

Компилируемые

Преобразование  
в машинный код

Примеры: C, C++, Go

2

Транслируемые

Преобразование  
в промежуточный код

Примеры: Java, C#

3

Интерпретируемые

Преобразование в машинные  
команды в момент  
выполнения программы

Примеры: Python,  
Ruby

# Низкоуровневые и высокоуровневые языки программирования

1

## Низкоуровневые

Низкий уровень абстракции.  
Язык ближе к компьютеру,  
чем к человеку

Примеры: Ассемблер,  
C

2

## Высокоуровневые

Высокий уровень абстракции.  
Язык ближе к человеку,  
чем к компьютеру

Примеры: Python,  
Go

# Пример кода на низкоуровневом языке

ooo

```
SECTION .text  
org 0x100
```

```
mov ah, 0x9  
mov dx, hello  
int 0x21
```

```
mov ax, 0x4c00  
int 0x21
```

```
SECTION .data  
hello: db "Hello, world!", 0xD, 0xA,  
'$'
```

# Пример кода на высокоуровневом языке

ooo

```
print("Hello world")
```



Ваши вопросы



# Язык программирования Python

Преимущества и недостатки



2

# Python

**Python** — высокоуровневый язык программирования с динамической строгой типизацией и автоматическим управлением памятью.

**Появился 20 февраля 1991 года**



# Python

## Преимущества Python

- Ориентация на повышение производительности разработчика
- Легкочитаемый код
- Высокое качество кода
- Переносимость написанных на Python программ

# Python

## Преимущества Python

- Ориентация на повышение производительности разработчика
- Легкочитаемый код
- Высокое качество кода
- Переносимость написанных на Python программ

## Недостатки Python

- Более низкая скорость работы
- Более высокое потребление памяти (по сравнению с компилируемыми языками C или C++)



Ваши вопросы

# Современная архитектура приложений



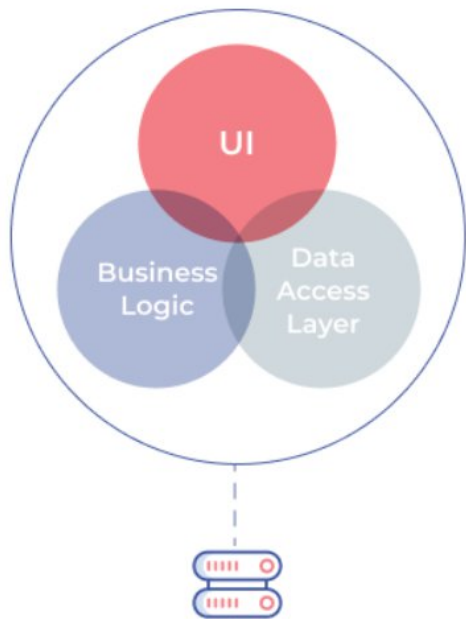
3

# Виды архитектур приложений

- Монолитная архитектура
- Сервис-ориентированная архитектура (SOA)
- Микросервисная архитектура (MSA) — тип сервисно-ориентированной архитектуры (SOA)

# Монолитная архитектура

Это единый общий модуль, в котором все компоненты приложения связаны между собой



**Если убрать хотя бы один элемент  
— всё сломается**



# Сервис-ориентированная архитектура

Сервис-ориентированная архитектура (SOA) — стиль архитектуры ПО, который предполагает модульное приложение, состоящее из дискретных и слабосвязанных программных агентов, выполняющих конкретные функции.

Сервис-ориентированный подход предполагает выделение групп связности.

В дальнейшем это упрощает подход к поддержке решения, но не решает проблему полного отключения модуля

# Микросервисная архитектура

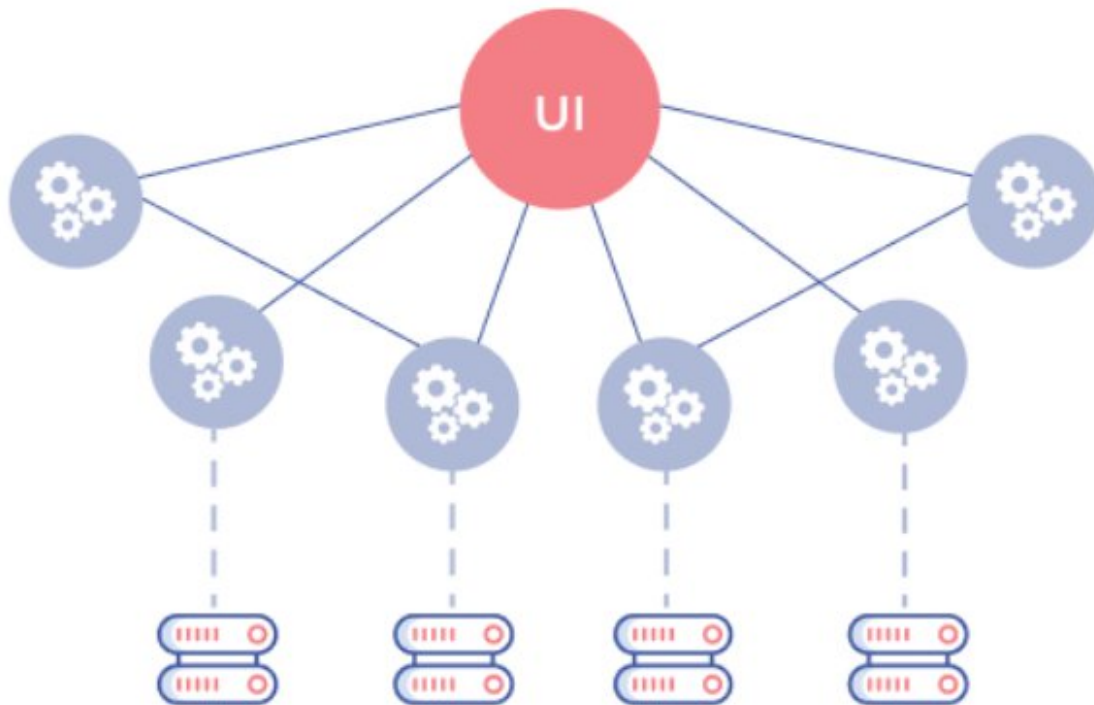
**Микросервисная архитектура (MSA)** — тип SAO (сервис-ориентированных архитектур), направленный на создание ряда автономных компонентов, составляющих приложение

# Микросервисная архитектура

**Микросервисная архитектура (MSA)** — тип SAO (сервис-ориентированных архитектур), направленный на создание ряда автономных компонентов, составляющих приложение.

- **Отличие от монолитных приложений**, созданных как единое целое: микросервисные приложения состоят из нескольких независимых компонентов, которые склеены вместе с помощью API
- **Отличие от сервис-ориентированных архитектур**: за счёт слабой связности допускается точечное отключение модулей при сохранении работоспособности программы

# Микросервисная архитектура





Ваши вопросы

# Итоги

В этой теме мы:

- 1 Рассмотрели разные языки программирования
- 2 Познакомились с Python
- 3 Научились различать разные виды архитектуры приложений



# Анонс следующего занятия

## Вебинар 2. Введение в синтаксис Python



# Введение в программирование

Глеб Пехов  
Бэкенд-разработчик на Python

