

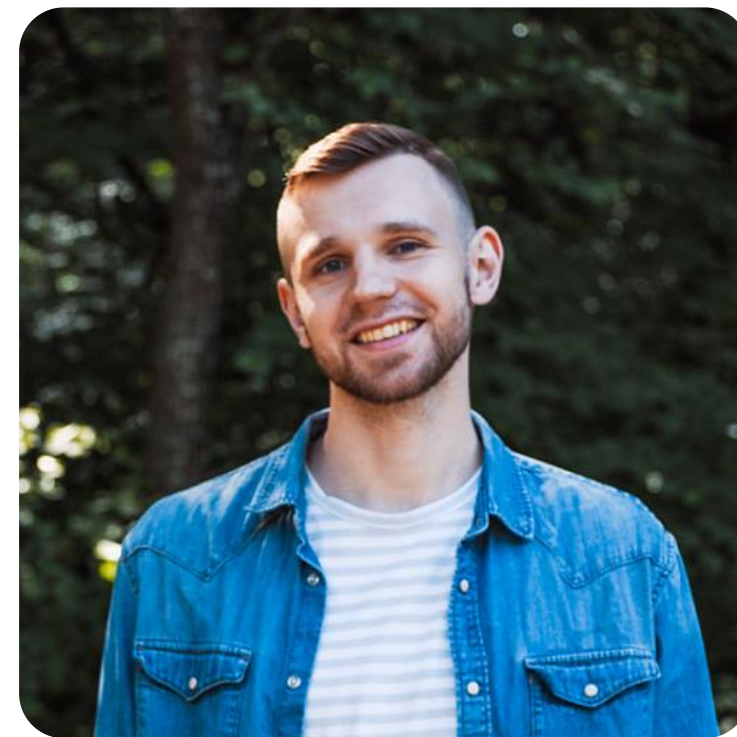
# Когда обновляться?

Что полезного в Python  
после версии 3.7

# 0 спикере

## Алексей Марашов

- Руководитель отдела разработки продуктов Cloud в EdgeЦентр
- 6+ лет в разработке на Python (ex Aviasales, ex Technokratos)
- BCS Software Engineering (СПбГЭТУ «ЛЭТИ», 2016)
- MCS Data Science (Innopolis, 2018)



# 0 компании

EdgeЦентр предоставляет передовые облачные решения от единого вендора



## CDN

Доставка любого контента быстрее, чем за 30 мс



## Облако

Масштабируемая облачная инфраструктура



## Стриминг

Решение любых технических вопросов, связанных с видео



## Хранилище

Хранение данных в облаке близко к пользователям



## Защита

Комплексная защита от DDoS-атак, хакеров и ботов для инфраструктуры, веб-сервисов и API



## Хостинг

Надёжные виртуальные и выделенные серверы для разных задач



## QA-центр

Тестирование игр, сайтов и приложений на всех стадиях разработки и запуска



## DNS

Быстрый DNS-хостинг для ускорения и повышения отказоустойчивости ваших ресурсов



## Управление IT-инфраструктурой

Полный IT-аутсорс без лишних вложений



## Разработка софта

Разработка полного цикла: от дизайна архитектуры до внедрения

# План лекции

1

**Популярность и зарплаты**

2

**Реализации Python**

3

**История развития**

4

**Направления развития Python**












5

**Что нового в последних версиях**

6

**Резюмируя**

# Самый популярный язык программирования

Sep 2023	Sep 2022	Change	Programming Language		Ratings	Change
1	1			Python	14.16%	-1.58%
2	2			C	11.27%	-2.70%
3	4	▲		C++	10.65%	+0.90%
4	3	▼		Java	9.49%	-2.23%
5	5			C#	7.31%	+2.42%
6	7	▲		JavaScript	3.30%	+0.48%
7	6	▼		Visual Basic	2.22%	-2.18%
8	10	▲		PHP	1.55%	-0.13%
9	8	▼		Assembly language	1.53%	-0.96%
10	9	▼		SQL	1.44%	-0.57%
11	15	▲▲		Fortran	1.28%	+0.26%

Источник: Tiobe



# В каких областях популярен

Источник: JetBrains

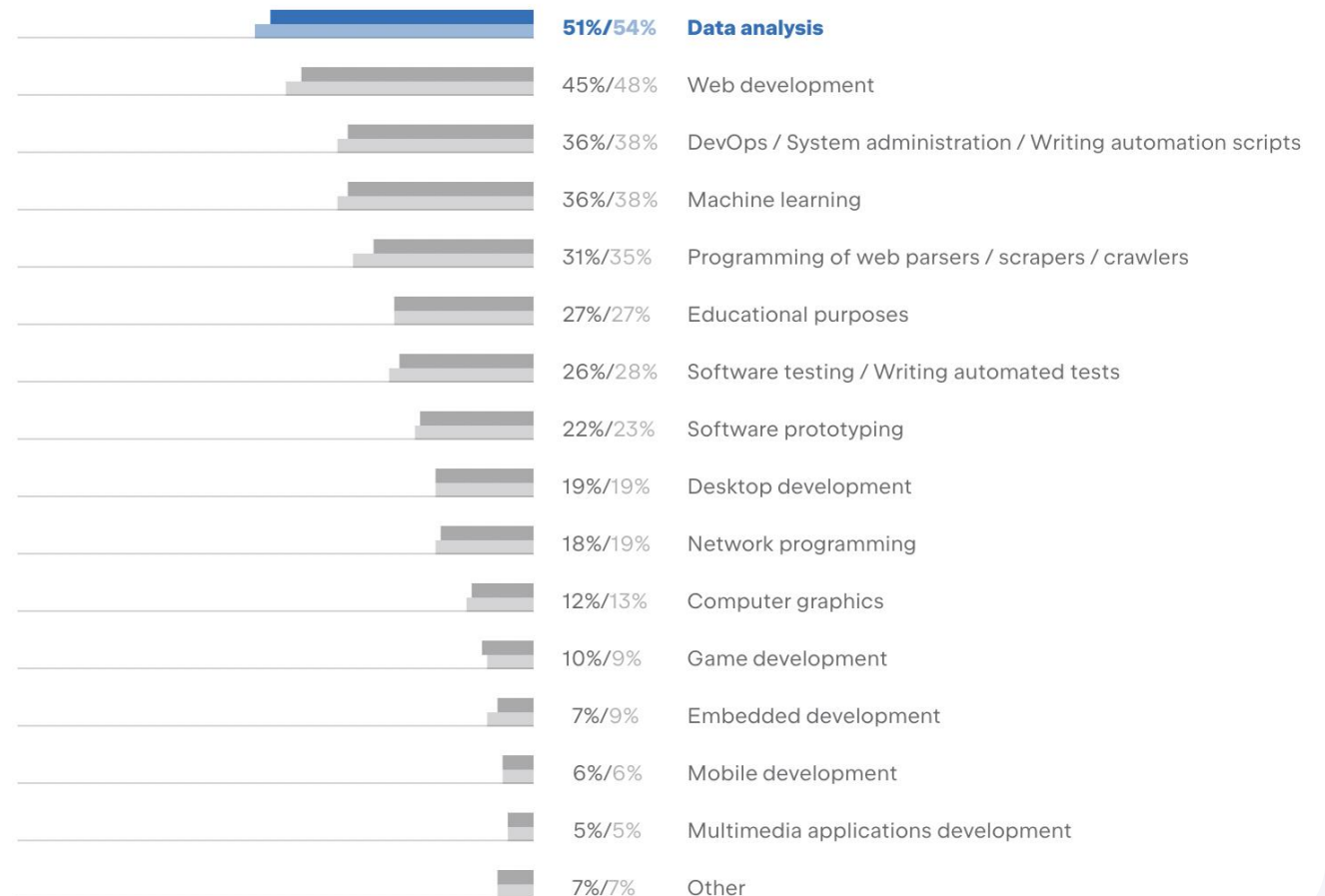


sales@edgecenter.ru 8 800 775 08 54 [edgecenter.ru](https://edgecenter.ru)

## Python usage in 2020 and 2021 100+

● 2021

● 2020





# Зарплаты разработчиков

## Сколько зарабатывают Python-разработчики в российском IT?

Поделиться

### Специальность

Все Java / Scala **Python** C# iOS

Android C / C++ Golang Ruby

PHP JS / Frontend JS / Backend

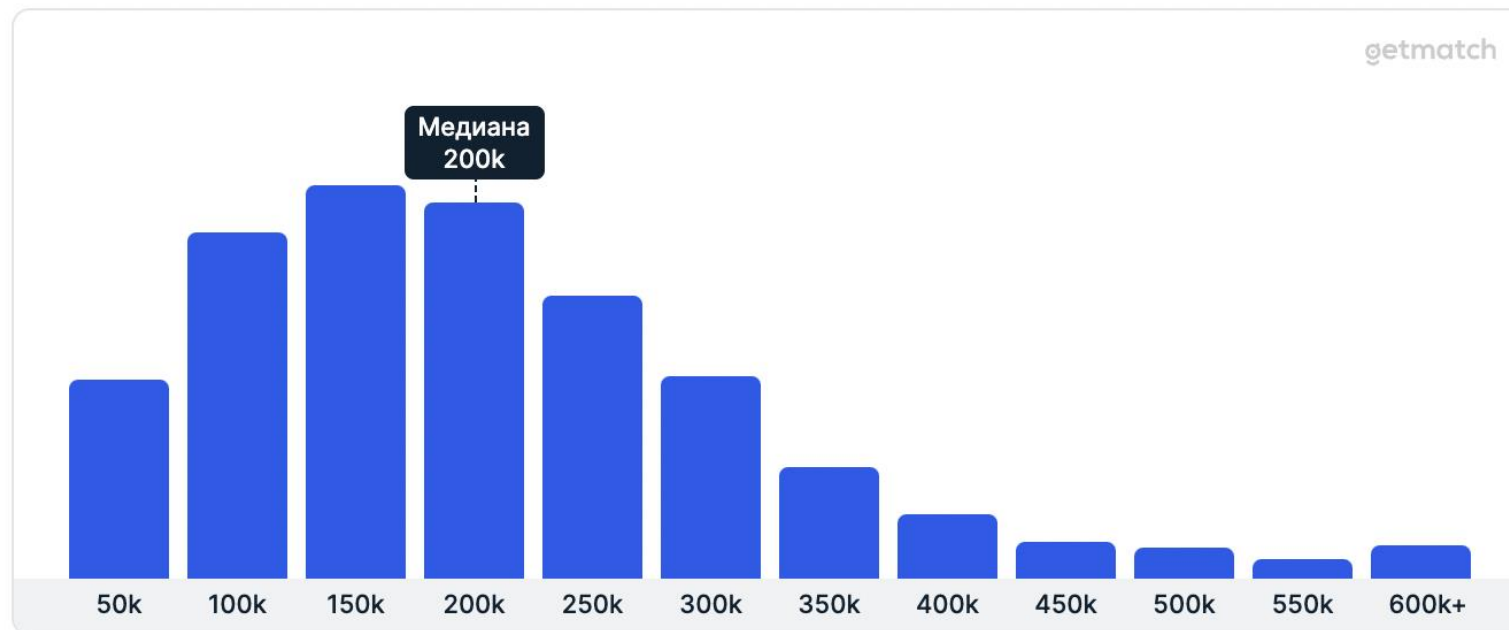
Fullstack Data Science DevOps

Product Management

Project Management Product Analyst

System Analyst Product Design

QA / Manual QA / Auto



Источник: GetMatch



sales@edgecenter.ru 8 800 775 08 54 [edgecenter.ru](https://edgecenter.ru)

# Рынок вакансий

## Работа и вакансии

Получать вакансии на почту

RSS

ВСЕ ВАКАНСИИ

ПОДХОДЯЩИЕ

Поиск

По убыванию зарплаты

Выбранные вами фильтры: Python Бэкенд разработчик

[Сохранить фильтры](#) [Сбросить](#)

Найдено 150 вакансий

Хабр Разработка:

hh

Помощь

python

Найти

Вакансии

Резюме

Компании

11 579 вакансий «python»



# Реализации Python

- [Cpython](#)
- [Jython](#)
- [PyPy](#)
- [Cython](#)



# Интерпретатор CPython: построение AST

```
import ast

tree = ast.parse("x, y = y, x")
print(ast.dump(tree, indent=4))
```

```
Module(
  body=[
    Assign(
      targets=[
        Tuple(
          elts=[
            Name(id='x', ctx=Store()),
            Name(id='y', ctx=Store())],
          ctx=Store())],
      value=Tuple(
        elts=[
          Name(id='y', ctx=Load()),
          Name(id='x', ctx=Load())],
        ctx=Load()))],
  type_ignores=[])
```

# Интерпретатор CPython: компиляция в байткод

```
import dis

def modulus(x, y):
    result = x % y
    return result

dis.dis(modulus)
```

Номер строки	Номер байта	Название инструкции	Индекс аргумента	Значение аргумента
4	0	LOAD_FAST	0	(x)
	2	LOAD_FAST	1	(y)
	4	BINARY_MODULO		
	6	STORE_FAST	2	(result)
5	8	LOAD_FAST	2	(result)
	10	RETURN_VALUE		

# Компоненты CPython

- 1 Runtime
- 2 Interpreter
- 3 Thread
- 4 Frame
- 5 Evaluation loop

Рекомендую прочитать: Устройство CPython. Доклад Яндекса ([pvsm.ru](https://pvsm.ru))

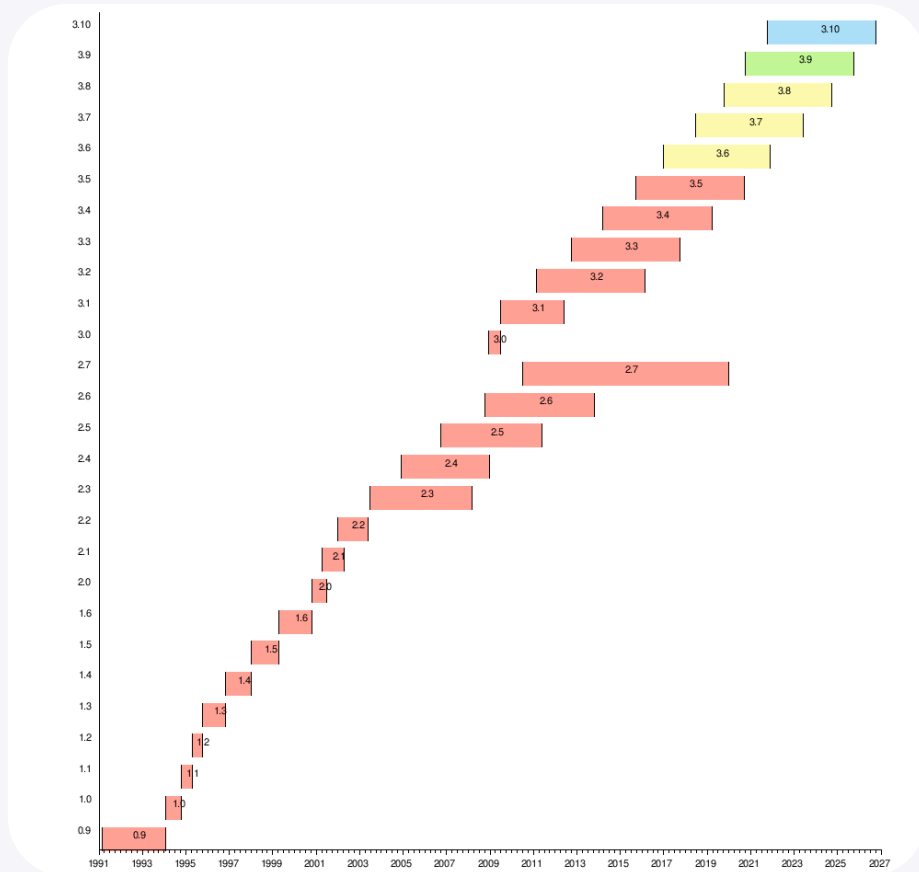
# Инструменты разработчика

- Debugging and Profiling
  - Audit events table
  - bdb — Debugger framework
  - faulthandler — Dump the Python traceback
  - pdb — The Python Debugger
  - The Python Profilers
  - timeit — Measure execution time of small code snippets
  - trace — Trace or track Python statement execution
  - tracemalloc — Trace memory allocations

Источник: The Python Standard Library — Python 3.11.5 documentation

# История разработки

- Версия 0.9.0 (февраль 1991)
- Версия 1.0 (январь 1994)
- Версия 2.0 (октябрь 2000)
- Версия 3.0 (декабрь 2008)



# Python Enhancement Proposal (PEP)

## Contents

- Abstract
- Rationale and Goals
- Terminology
- Backwards compatibility
- Implementation
  - Parameters to generics are available at runtime
  - Forward compatibility
- Reference implementation
- Rejected alternatives
  - Do nothing
  - Generics erasure
  - Disallowing instantiation of parameterized types
  - Making `isinstance(obj, list[str])` perform a check ignoring generics
  - Making `isinstance(obj, list[str])` perform a runtime type check
  - Naming the type `GenericType` instead of `GenericAlias`
- Note on the initial draft
- Acknowledgments
- Copyright

Источник: PEP 0 – Index of Python Enhancement Proposals (PEPs) | [peps.python.org](https://peps.python.org)



# Версии Python

1

## Версия 0.9.0

Модули, классы с наследованием, основные типы данных, обработка исключений.

3

## Версия 2.X

Сборщик мусора с поддержкой циклических ссылок, генераторы, элементы ООП.

2

## Версия 1.0

Элементы функционального программирования.

4

## Версия 3.0

Unicode для строк, аннотация типов, функция print, новый синтаксис.

# Python 3.6

## f-strings (PEP 498)

```
import math

radius = 1.2
length = 2 * math.pi * radius
f"Length of the circle with the radius {radius} = {length:.2f}"
```

## Typing module

```
primes: List[int] = []
captain: str # Note: no initial value!
class Starship:
    stats: Dict[str, int] = {}
```

# Python 3.6

## Path module

```
from pathlib import Path
path = str(Path("/") / Path("tmp") / Path("demo.tmp"))
with open(path, "w") as f:
    f.write("Hello, Path!")
```

# Python 3.7

## Dataclasses (PEP 557)

```
from dataclasses import dataclass

@dataclass(order=True)
class User:
    name: str
    age: int
```

# Python 3.8

## Assignment expression “the walrus operator” (PEP 572)

```
# Loop over fixed length blocks  
while (block := f.read(256)) != '':  
    process(block)
```

```
if (n := len(a)) > 10:  
    print(f"List is too long ({n} elements, expected <= 10)")
```

# Python 3.8

## Positional only parameters

```
def strlen(obj: str, /):  
    c = 0  
    for _ in obj:  
        c += 1  
    return c  
  
strlen("hello")  
strlen(obj='hello') # The "obj" keyword argument impairs readability
```

# Python 3.8

## An "=" specifier in f-strings

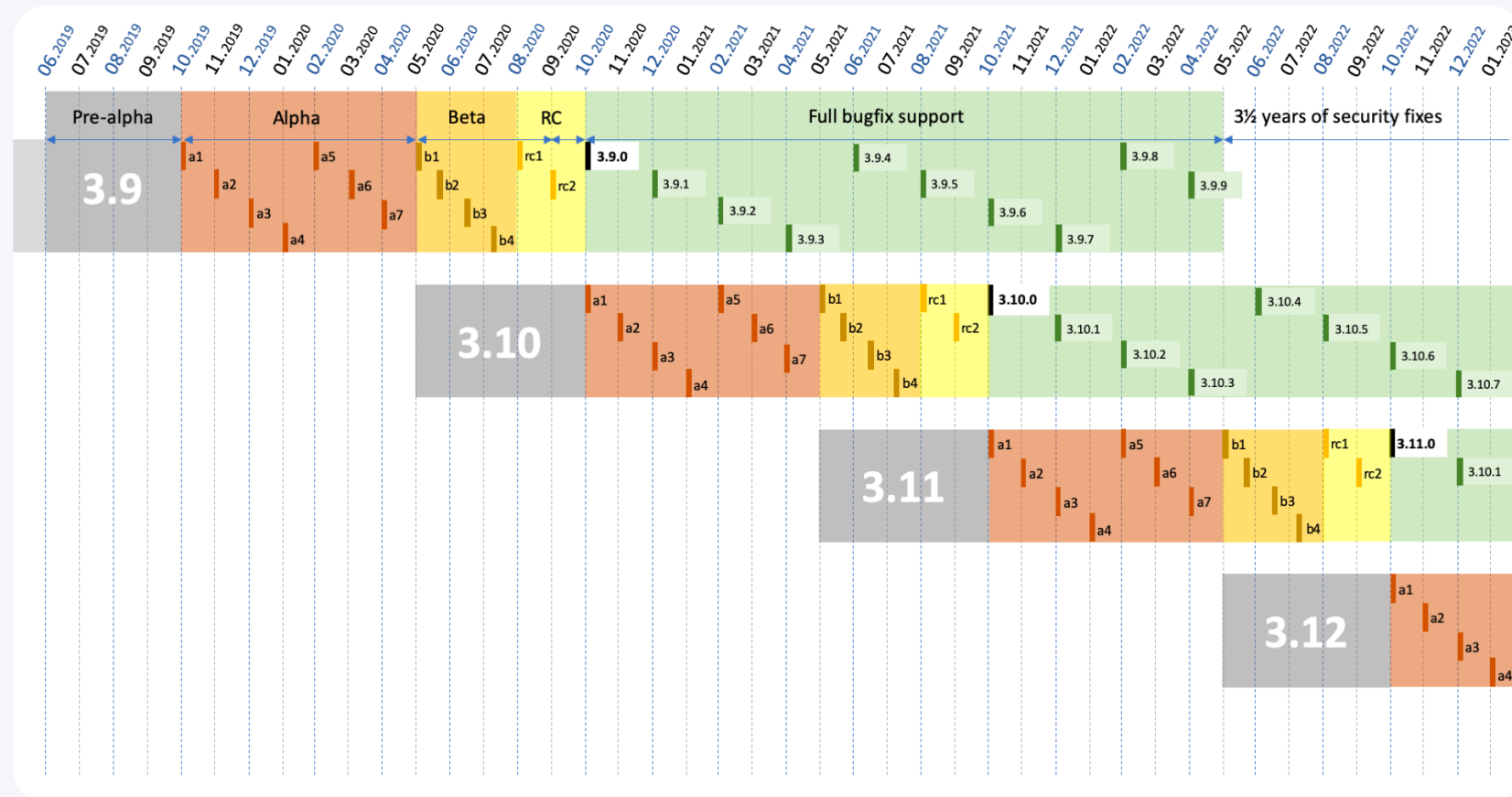
```
import math

radius = 1.2
length = 2 * math.pi * radius
f"A circle with the {radius=} has a length {length=:.2f}"
```



# Python 3.9

## Регулярный график обновления версий (PEP 602)



# Python 3.9

## Dictionary Merge operator (PEP 584)

```
dict1 = {"x": "1", "y": "1"}  
dict2 = {"y": "2", "z": "2"}  
  
print(dict1 | dict2)      # {'x': '1', 'y': '2', 'z': '2'}  
print(dict2 | dict1)      # {'y': '1', 'z': '2', 'x': '1'}
```

# Python 3.9

## Type hint generics from standard collection (PEP 585)

```
def greet_all(names: list[str]) -> None:  
    for name in names:  
        print("Hello", name)
```

# Python 3.10

## Structural pattern matching (PEP 634, 635, 636)

```
def http_error(status):  
    match status:  
        case 400:  
            return "Bad request"  
        case 404:  
            return "Not found"  
        case 418:  
            return "I'm a teapot"  
        case _:  
            return "Something's wrong with the internet"
```

```
case 401 | 403 | 404:  
    return "Not allowed"
```

# Python 3.10

## Writing Union types as X | Y

```
def add_5(value: int | float) -> int | float:  
    return value + 5
```

# Python 3.10

## New Context manager syntax

```
from contextlib import redirect_stdout

with (open("file.txt", "w") as file, redirect_stdout(file)):
    ...
```

```
with (open("file1.txt", "r") as file1, open("file2.txt", "w") as file2):
    ...
```

# Python 3.11

## ExceptionGroup and a new exception syntax

```
try:
    raise ExceptionGroup("Data validations", (ValueError("problem 1"), ValueError("problem 2")))
except * (ValueError, ValueError) as exception_group_1:
    print("validations failed")
    raise ValueError from exception_group_1
```



# Python 3.12

## Type Parameter Syntax (PEP 695)

```
class Loadable(typing.Protocol):
    def load(self) -> None:
        ...

class Stack[T:Explainable]:
    ...

class Element:
    def load(self) -> str:
        ...

stack = Stack[Element]()
stack.push(Element())
```

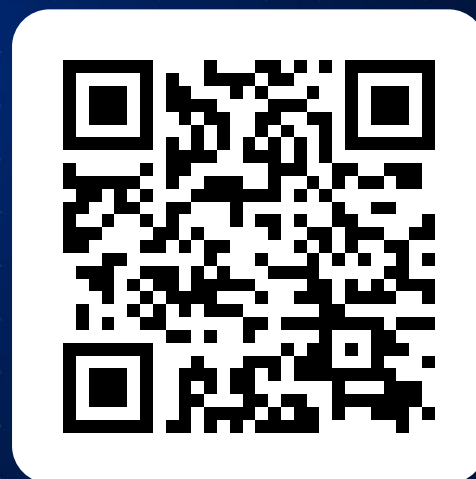
# Резюмируя

- 1 Разработчики уделяют все больше внимания оптимизации Cpython
- 2 Новые версии CPython в несколько раз быстрее и эффективнее
- 3 Обновления удобно планировать благодаря регулярному графику выхода новых версий
- 4 Обновления закрывают уязвимости
- 5 Появляется удобный и актуальный синтаксис, востребованный сообществом

# Спасибо за внимание!



**Репозиторий с  
материалами лекции**



**Вакансии  
EdgeЦентр**