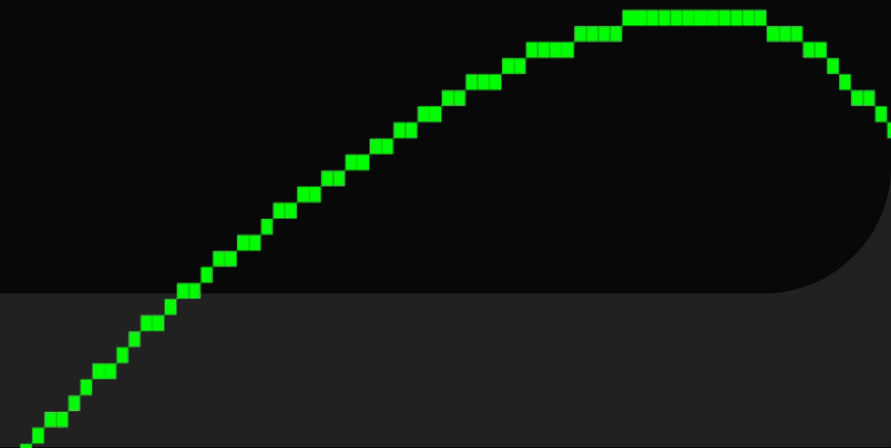
Андрей Шарапов
Контур

PyPy: ускоряем Python с помощью Python

План доклада

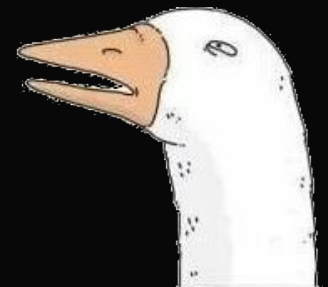
1. Python медленный
2. Находим решение
3. Запускаем тесты
4. Что там с 3.13
5. Выводы

Зачем мы здесь?

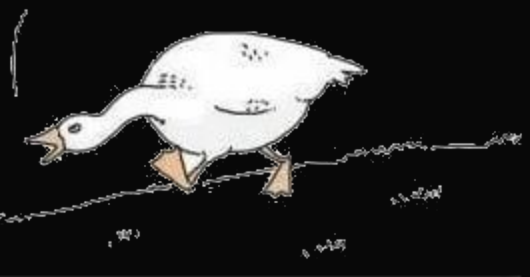


Почему Python медленный?

Почему Python медленный?



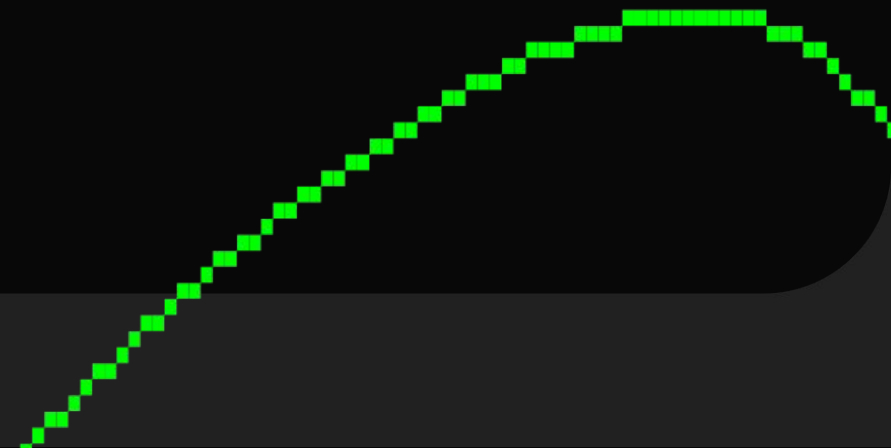
Почему Python медленный?



~~Почему~~ Python медленный?

Когда Python медленный?

Всегда ли проблема в Python?



Всегда ли проблема в Python

- Инфраструктура

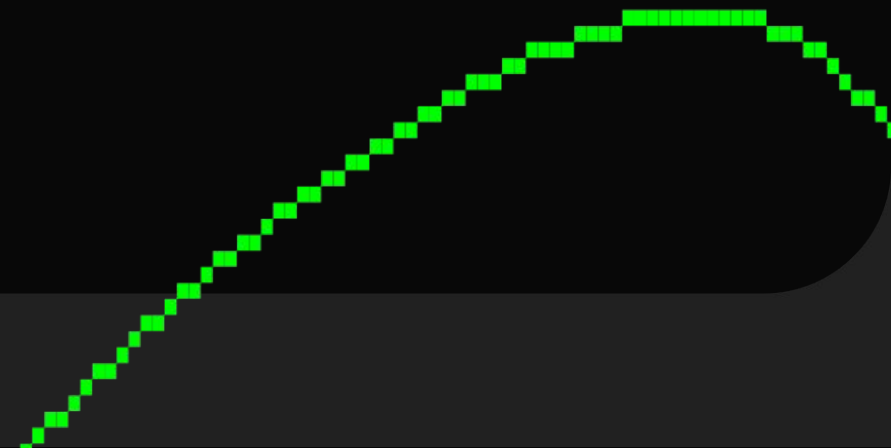
Всегда ли проблема в Python

- Инфраструктура
- Базы данных

Всегда ли проблема в Python

- Инфраструктура
- Базы данных
- Другие сервисы

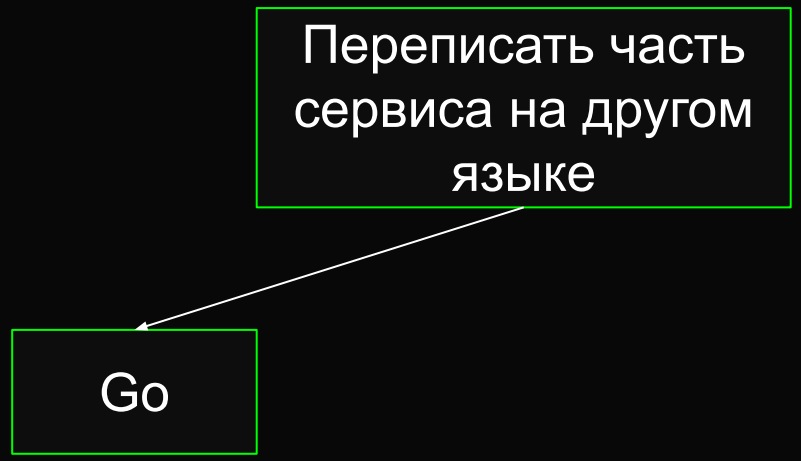
Что с этим делать?



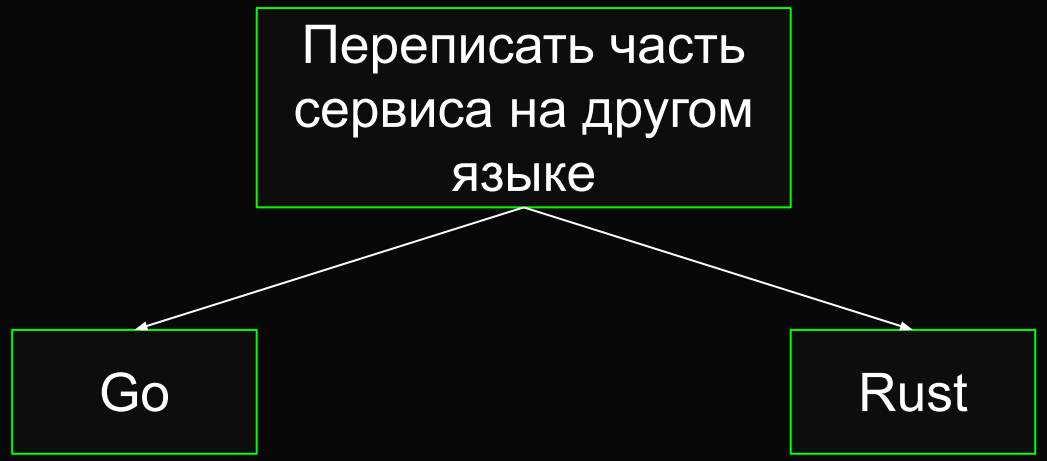
Что с этим делать?

Переписать часть сервиса на другом языке

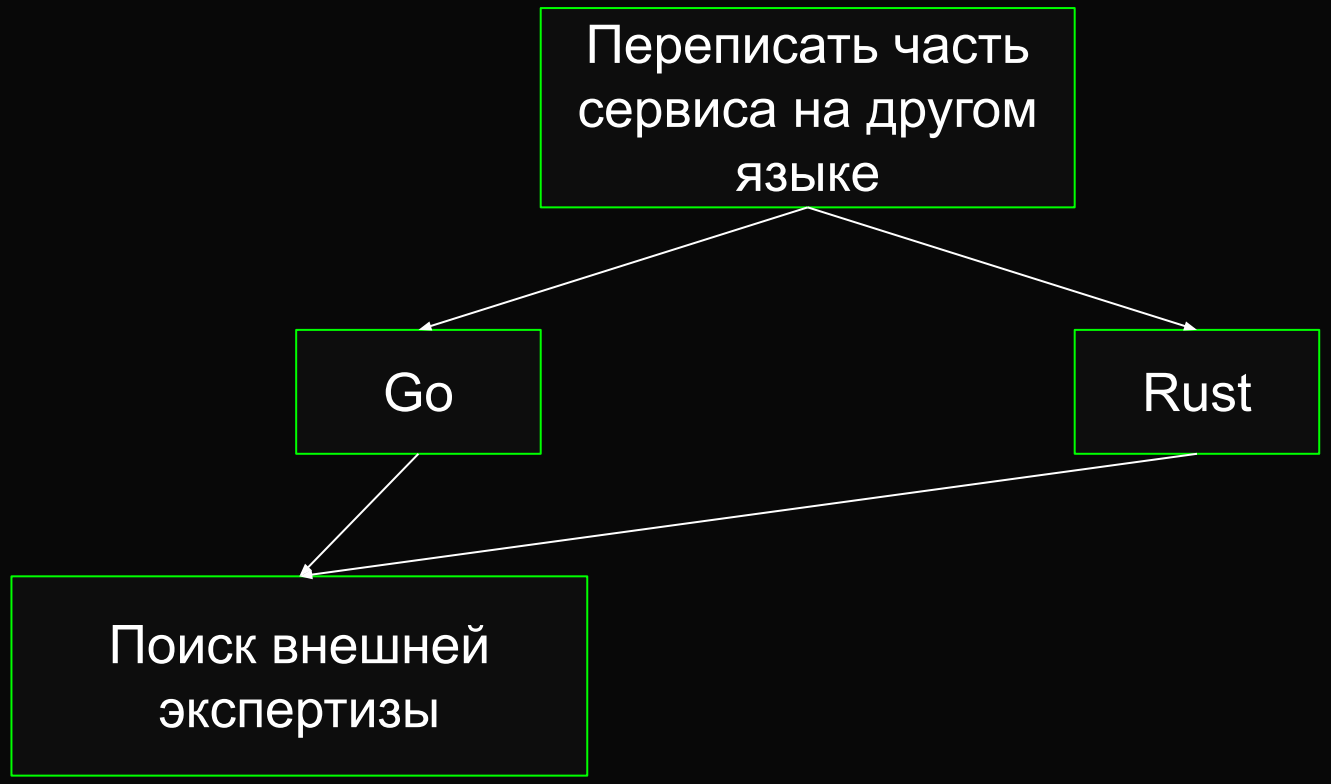
Что с этим делать?



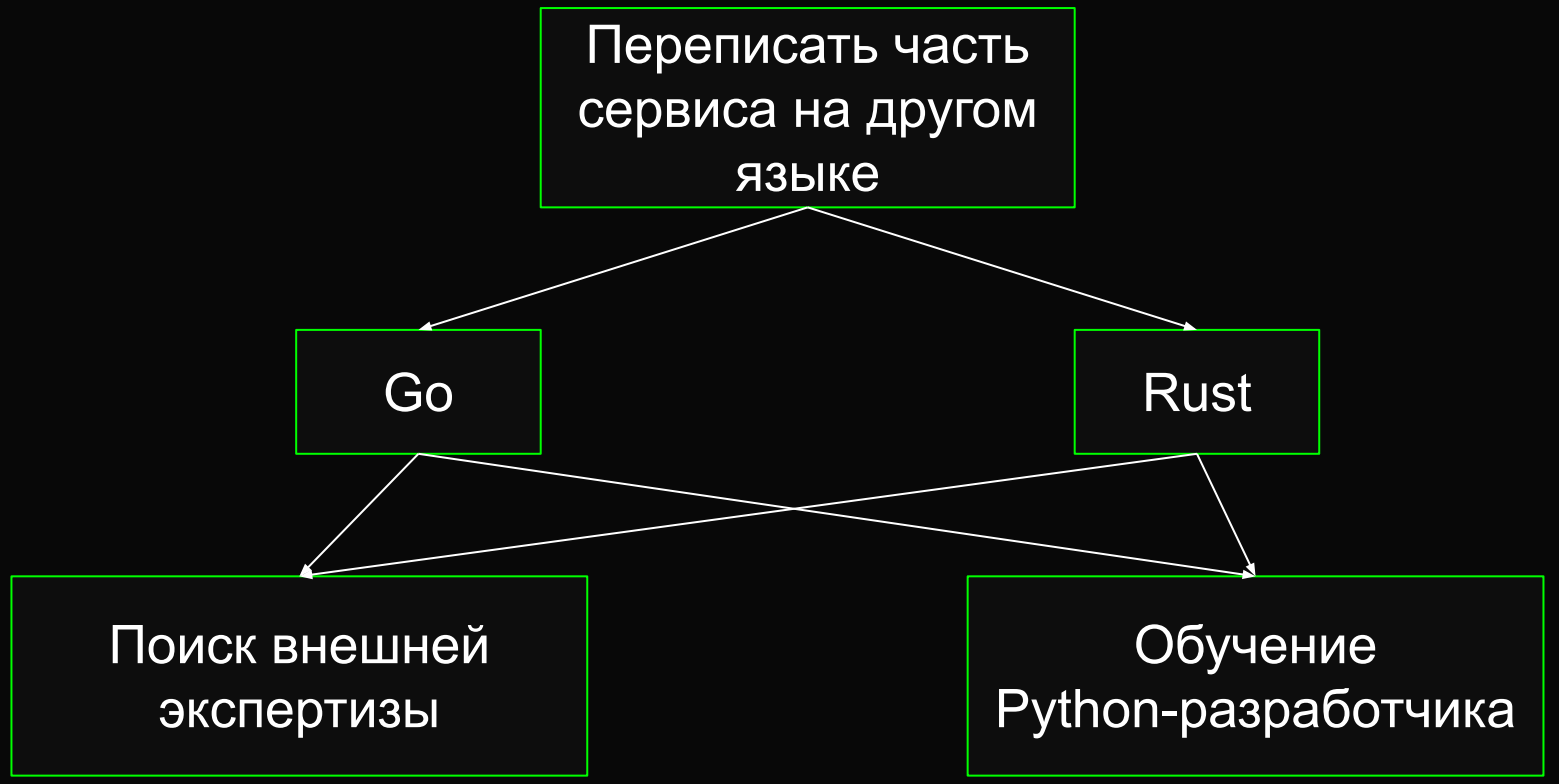
Что с этим делать?



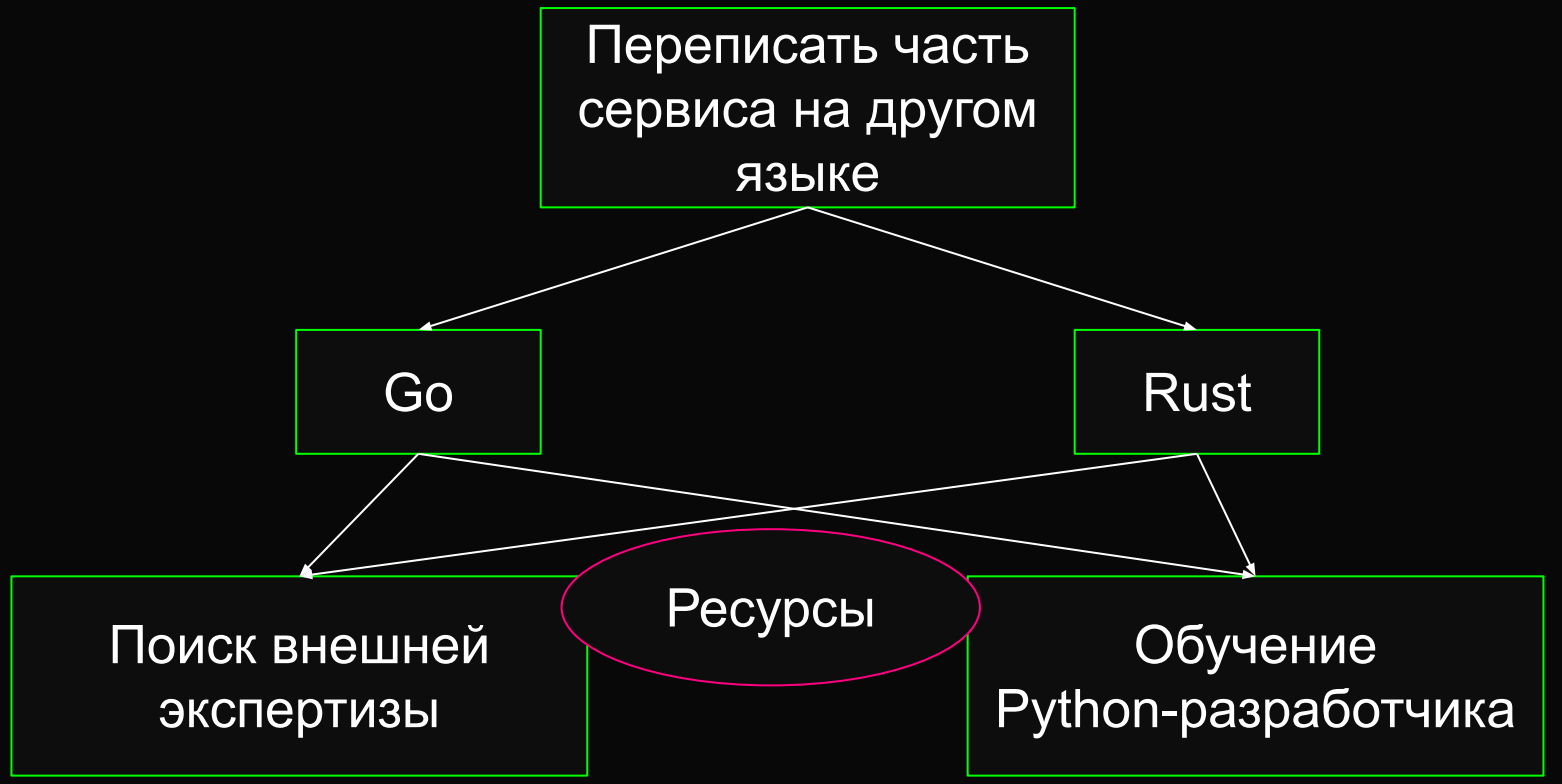
Что с этим делать?



Что с этим делать?



Что с этим делать?

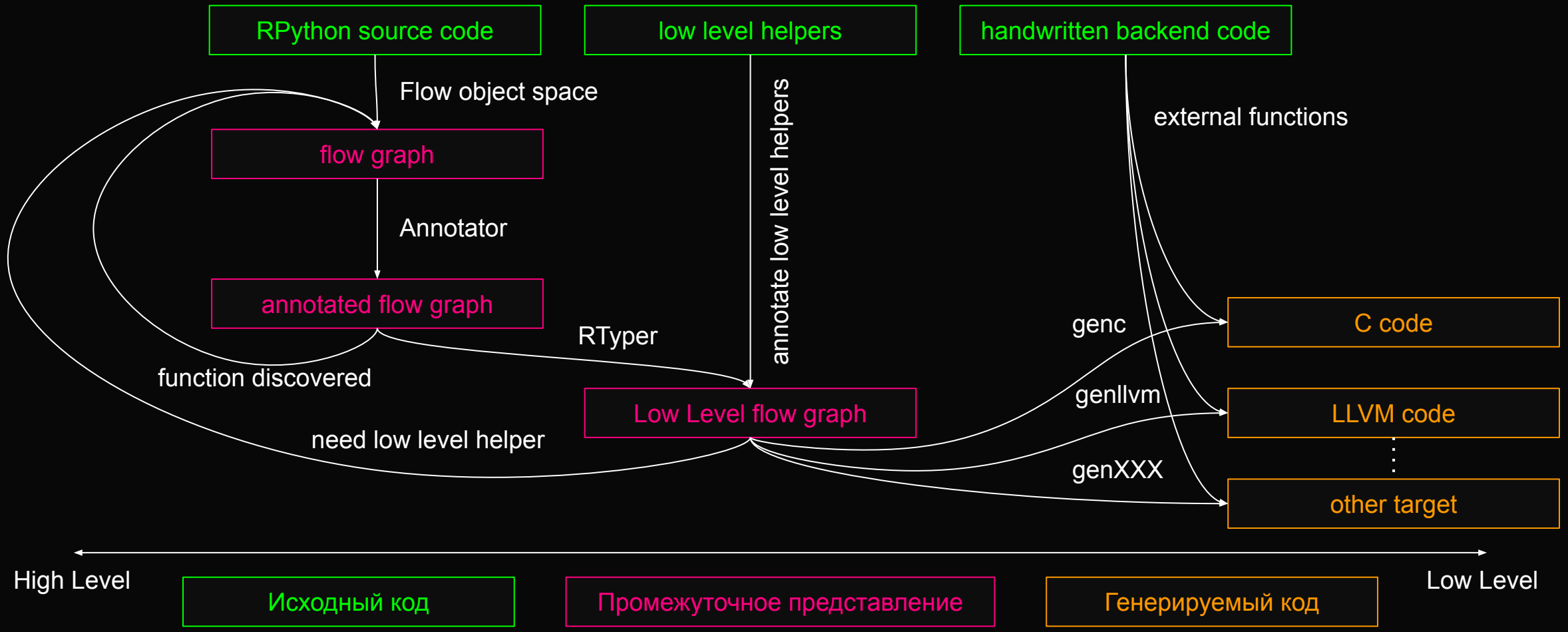


Что с этим делать?

Решаем проблему Python с помощью Python

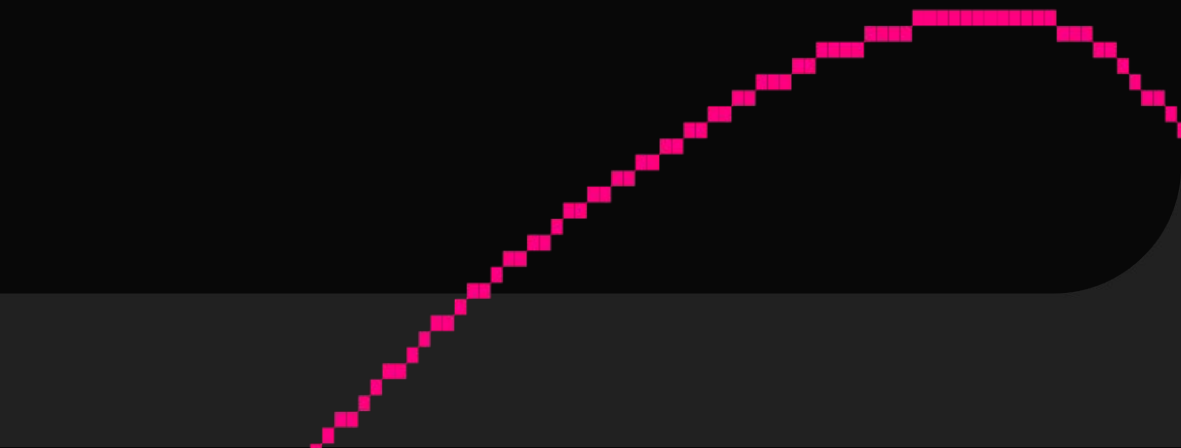
PyPy - интерпретатор Python написанный на Python

PyPy: Translation Process



Армин Риго - один из основателей и ведущий разработчик PyPy

Актуальные версии PyPy



Актуальные версии PyPy

PyPy 2.7

Python 2.7.18

Актуальные версии PyPy

PyPy 2.7

Python 2.7.18

PyPy 3.9

Python 3.9.19

Актуальные версии PyPy

PyPy 2.7

Python 2.7.18

PyPy 3.9

Python 3.9.19

PyPy 3.10

Python 3.10.14

Поддерживаемые платформы

- x86 (IA-32), x86_64

Поддерживаемые платформы

- x86 (IA-32), x86_64
- ARM platforms (ARMv6 or ARMv7, with VFPv3, and Apple Silicon arm64)

Поддерживаемые платформы

- x86 (IA-32), x86_64
- ARM platforms (ARMv6 or ARMv7, with VFPv3, and Apple Silicon arm64)
- PowerPC 64bit both little and big endian, System Z (s390x)

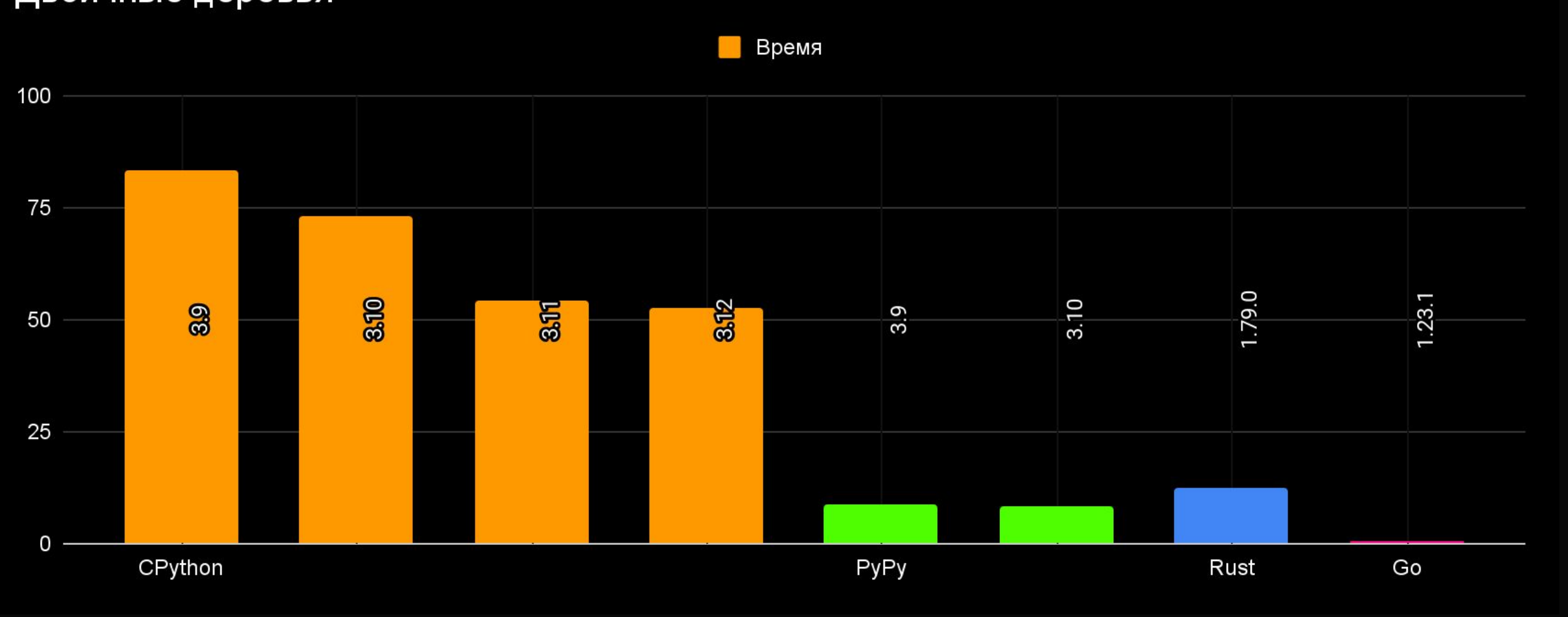
Поддерживаемые платформы

- x86 (IA-32), x86_64
- ARM platforms (ARMv6 or ARMv7, with VFPv3, and Apple Silicon arm64)
- PowerPC 64bit both little and big endian, System Z (s390x)
- RISCV

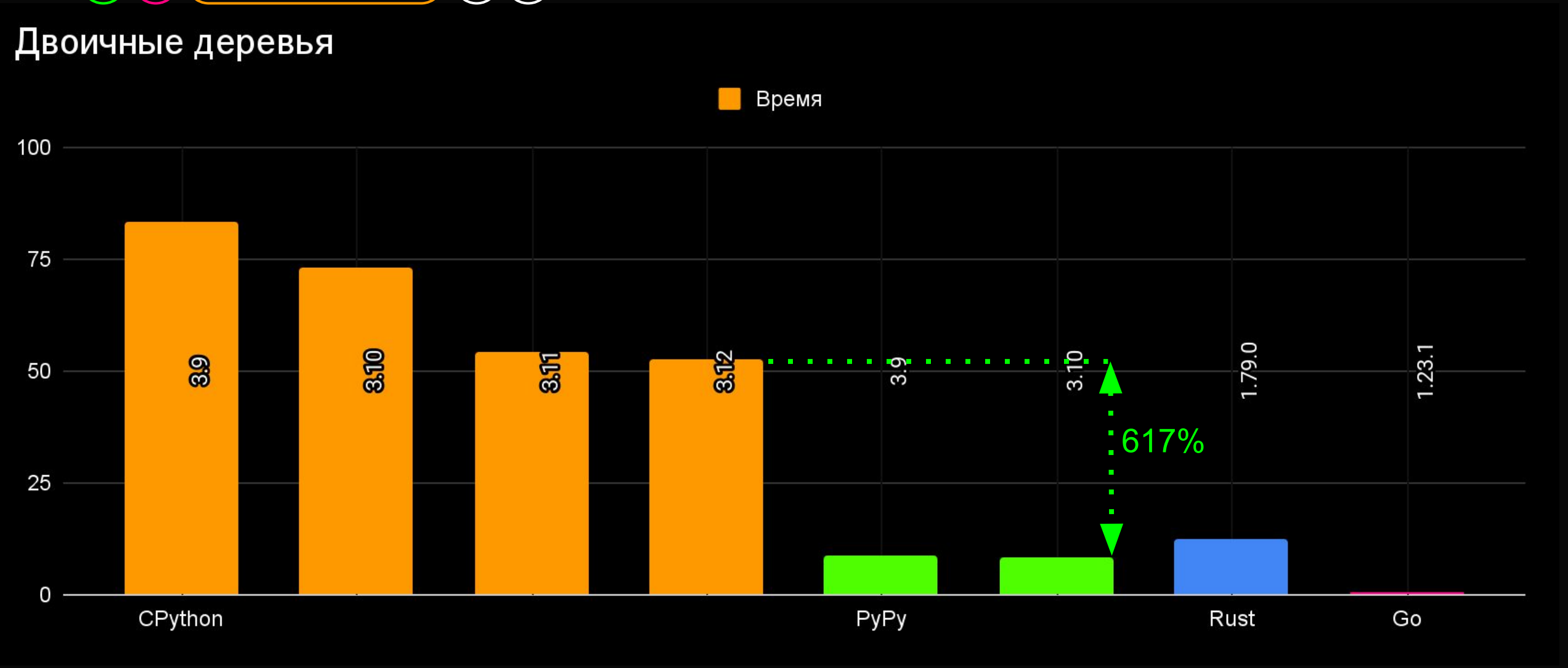
Запускаем тесты

The Computer Language Benchmarks Game

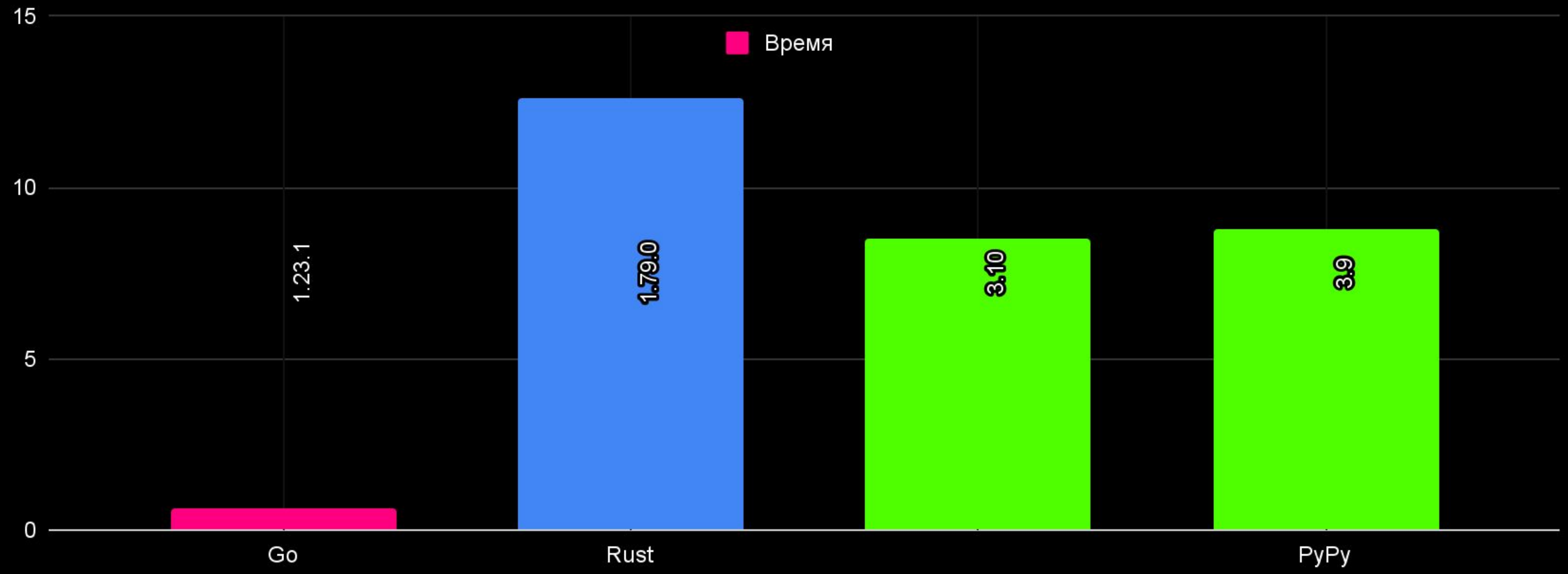
Двоичные деревья



Двоичные деревья

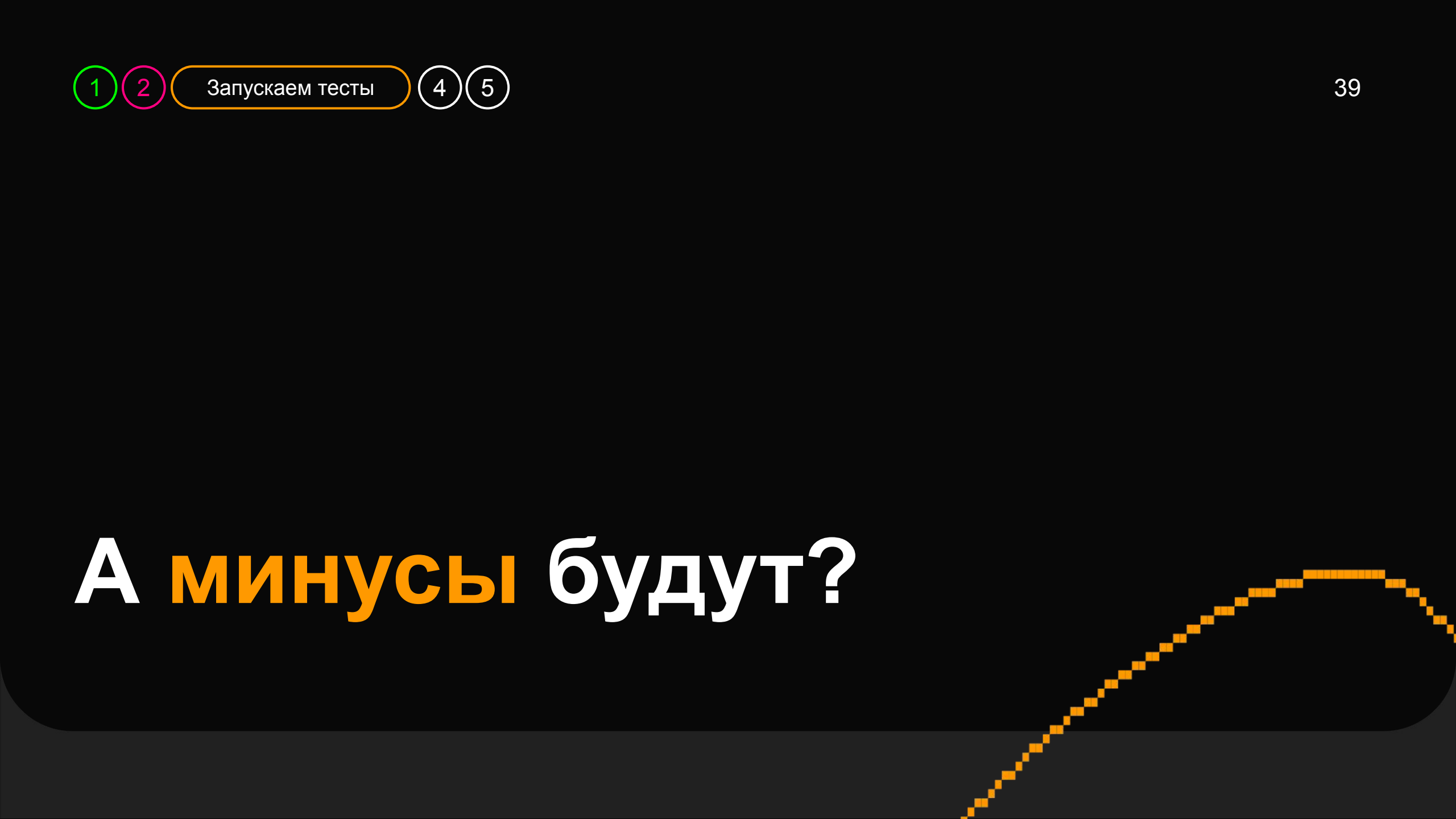


Двоичные деревья

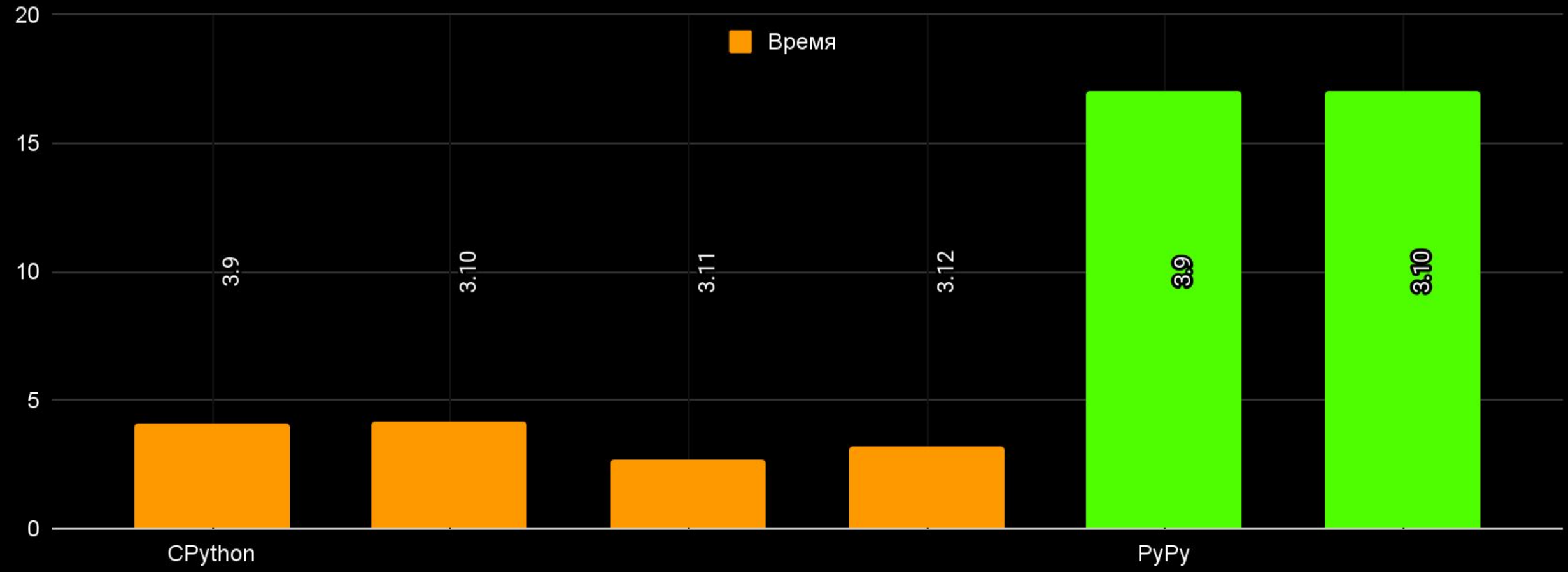


PyPy ускоряет алгоритмы написанные на чистом Python

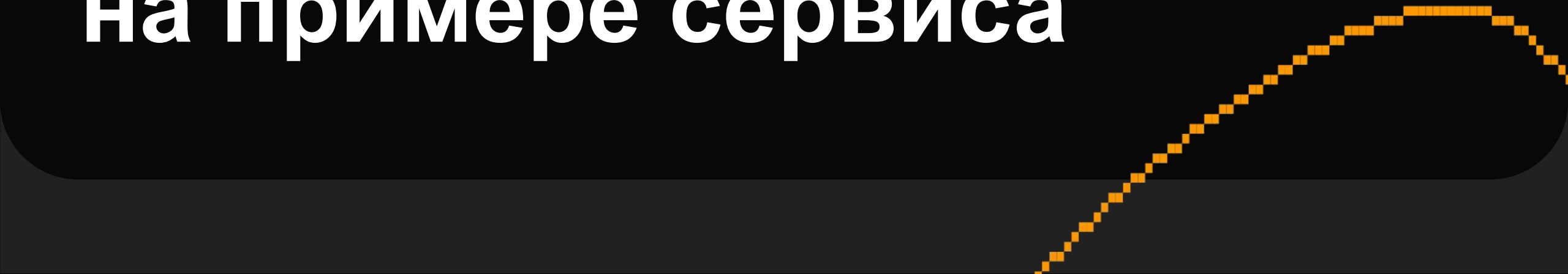
А минусы будут?

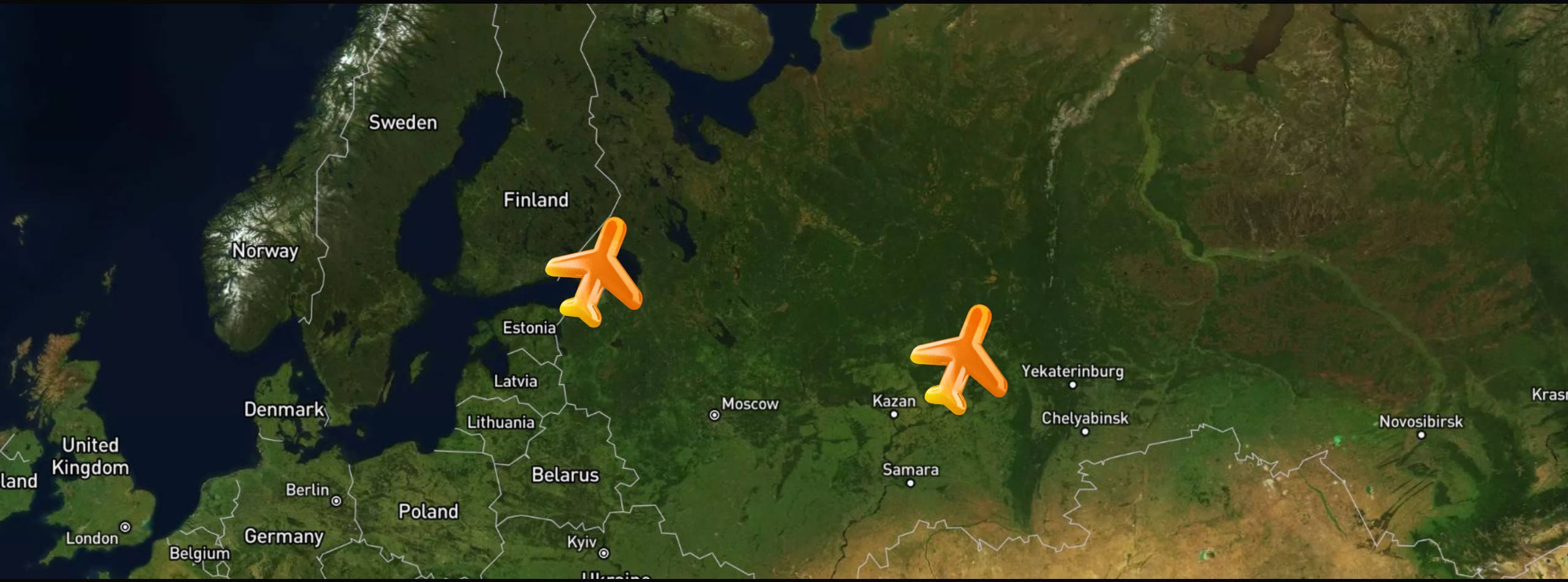


Вызов hexdigest()



Ускорение вычислений на примере сервиса




Sweden
Finland
Norway
Estonia
Latvia
Denmark
Lithuania
Moscow
Belarus
United Kingdom
Berlin
Poland
London
Germany
Belgium
Kyiv
Kazan
Yekaterinburg
Chelyabinsk
Samara
Novosibirsk

```python
from fastapi import FastAPI, Request
from geopy.distance import geodesic

from pydantic import BaseModel


class Distance(BaseModel):
    meters: float


app = FastAPI()
```

```python
@app.get("/distance")
async def distance(
    from_lat: float,
    from_lng: float,
    to_lat: float,
    to_lng: float,
) -> Distance:
    from_point = (from_lat, from_lng)
    to_point = (to_lat, to_lng)

    meters = geodesic(from_point, to_point).meters

    return Distance(meters=meters)
```

ASGI

- Daphne

ASGI

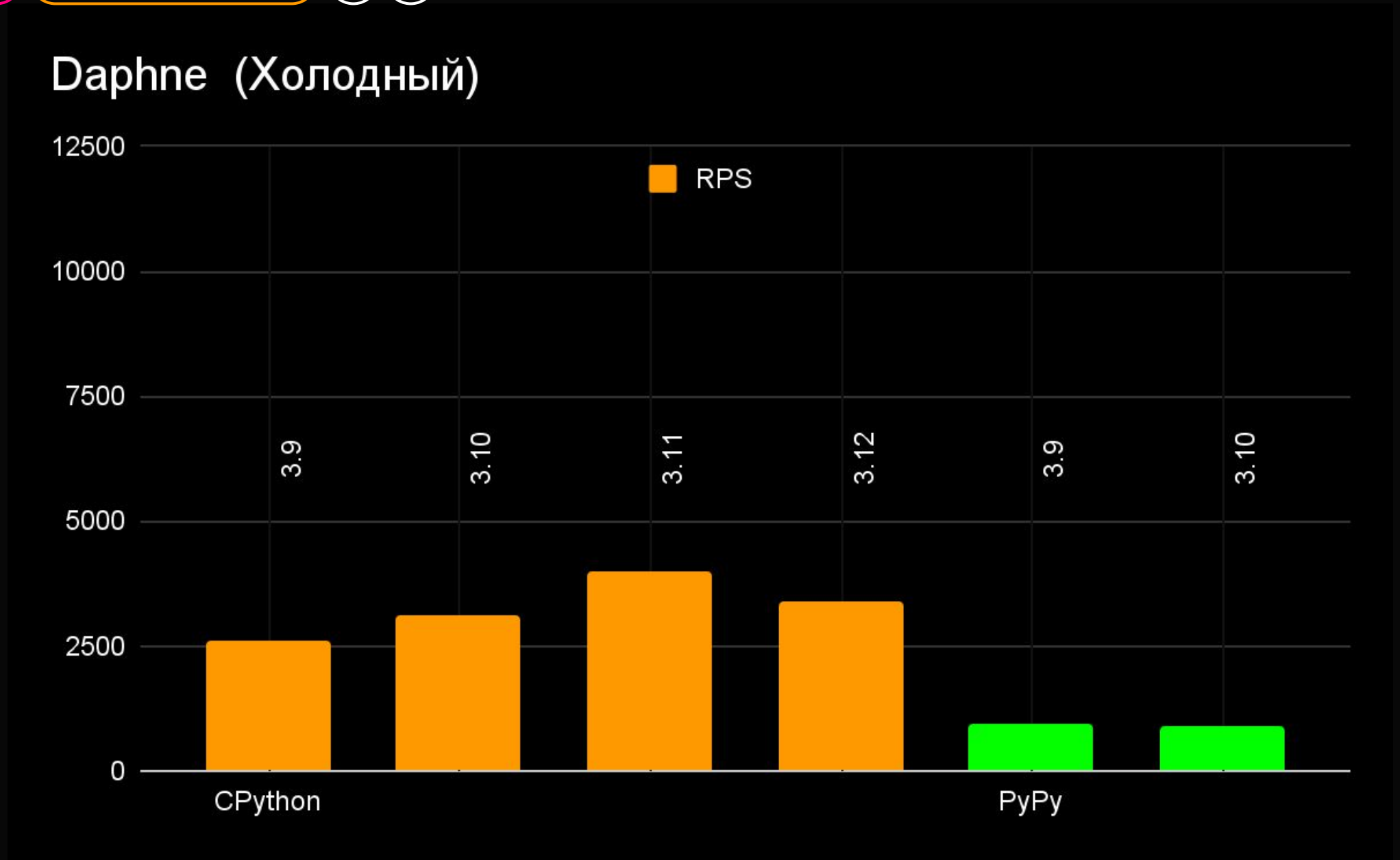
- Daphne
- Hypercorn

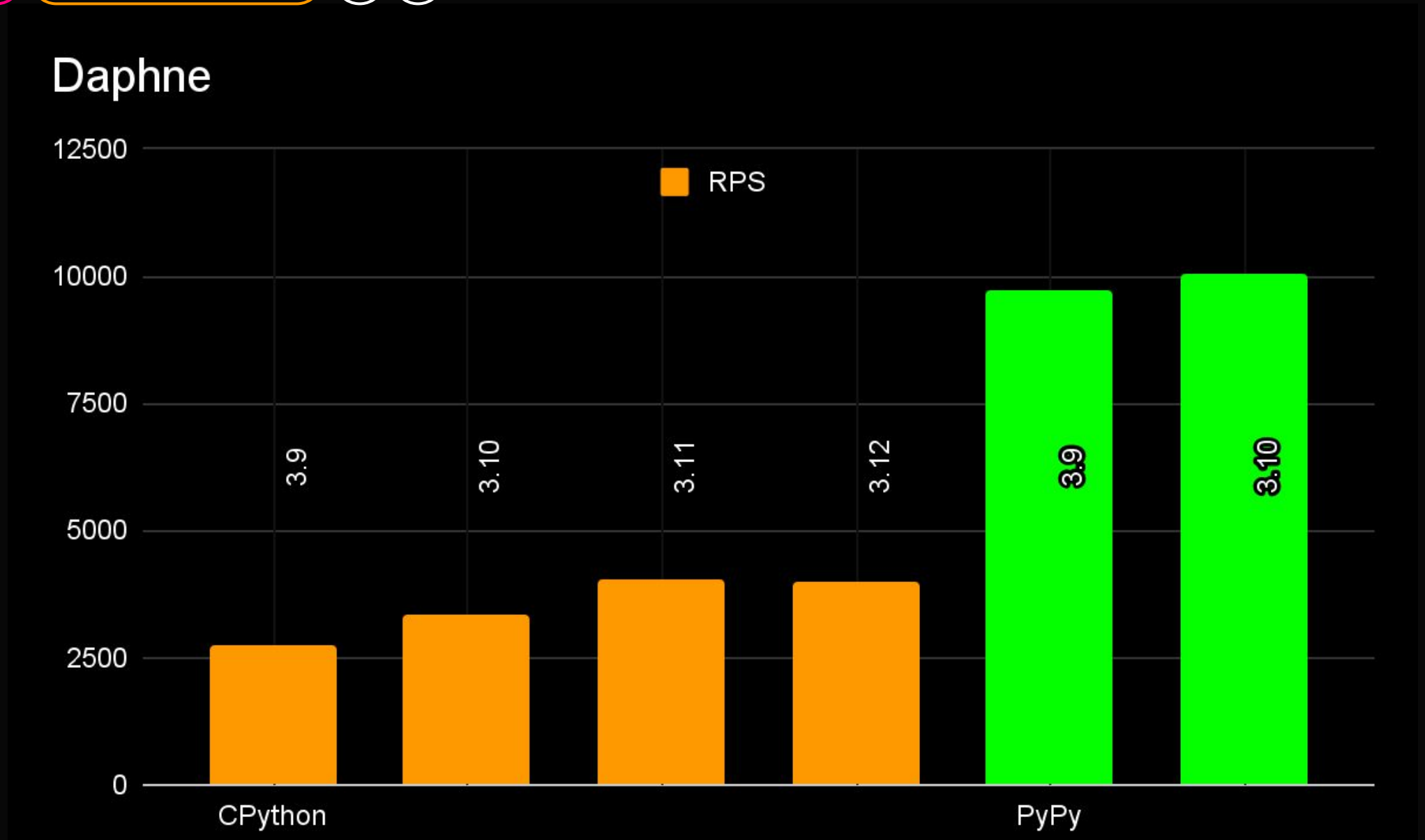
ASGI

- Daphne
- Hypercorn
- Uvicorn

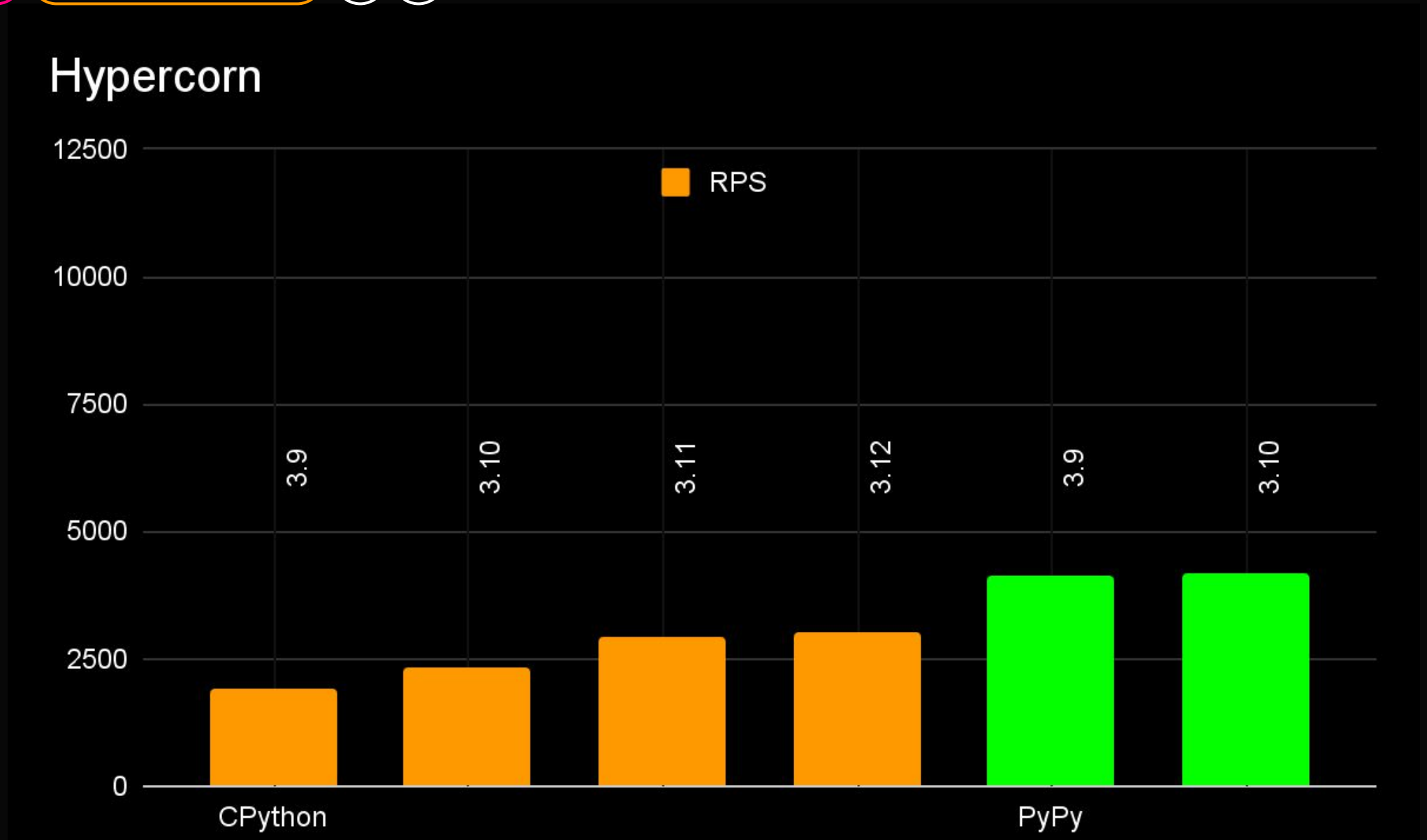
ASGI

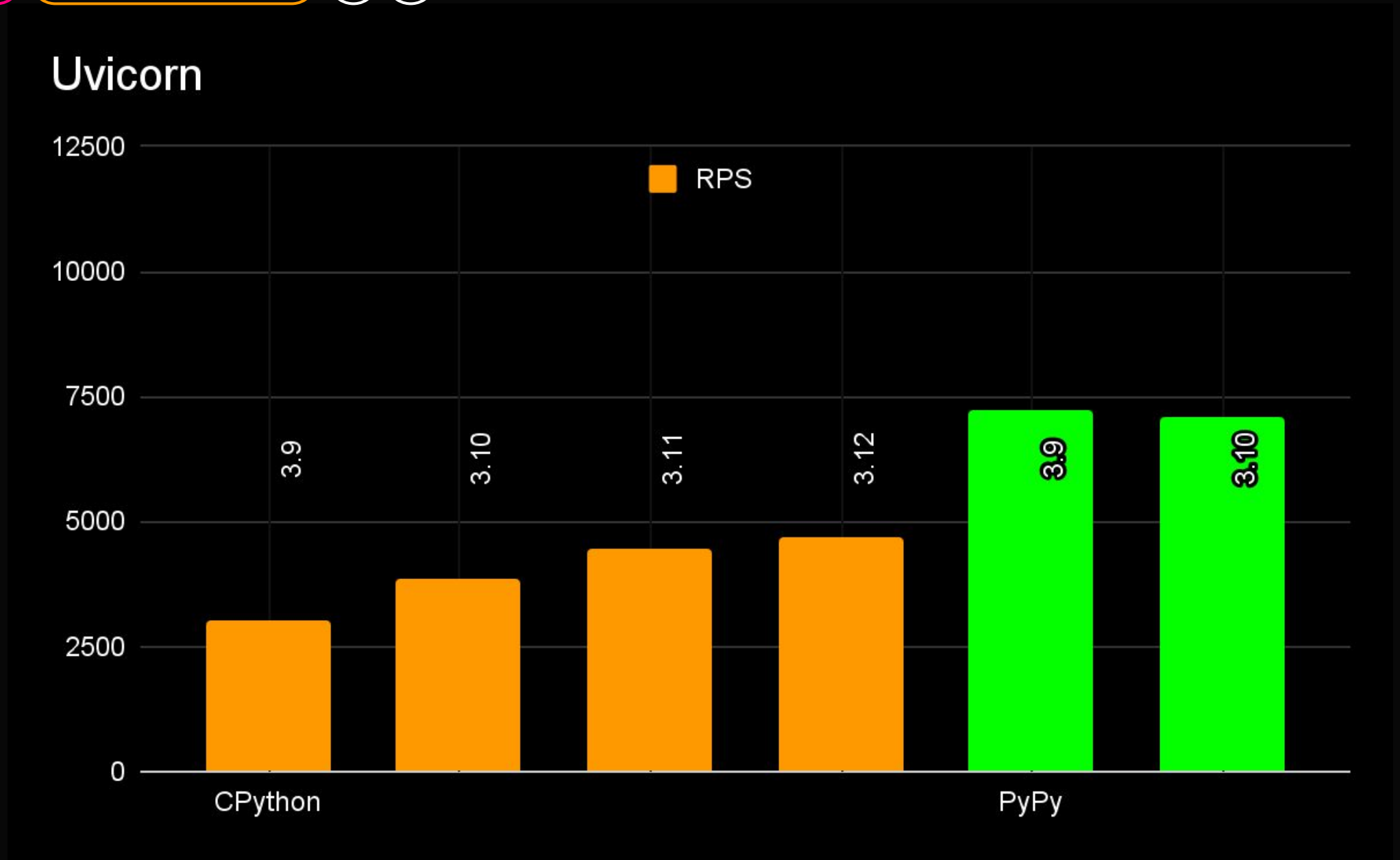
- Daphne
- Hypercorn
- Uvicorn
- Granian

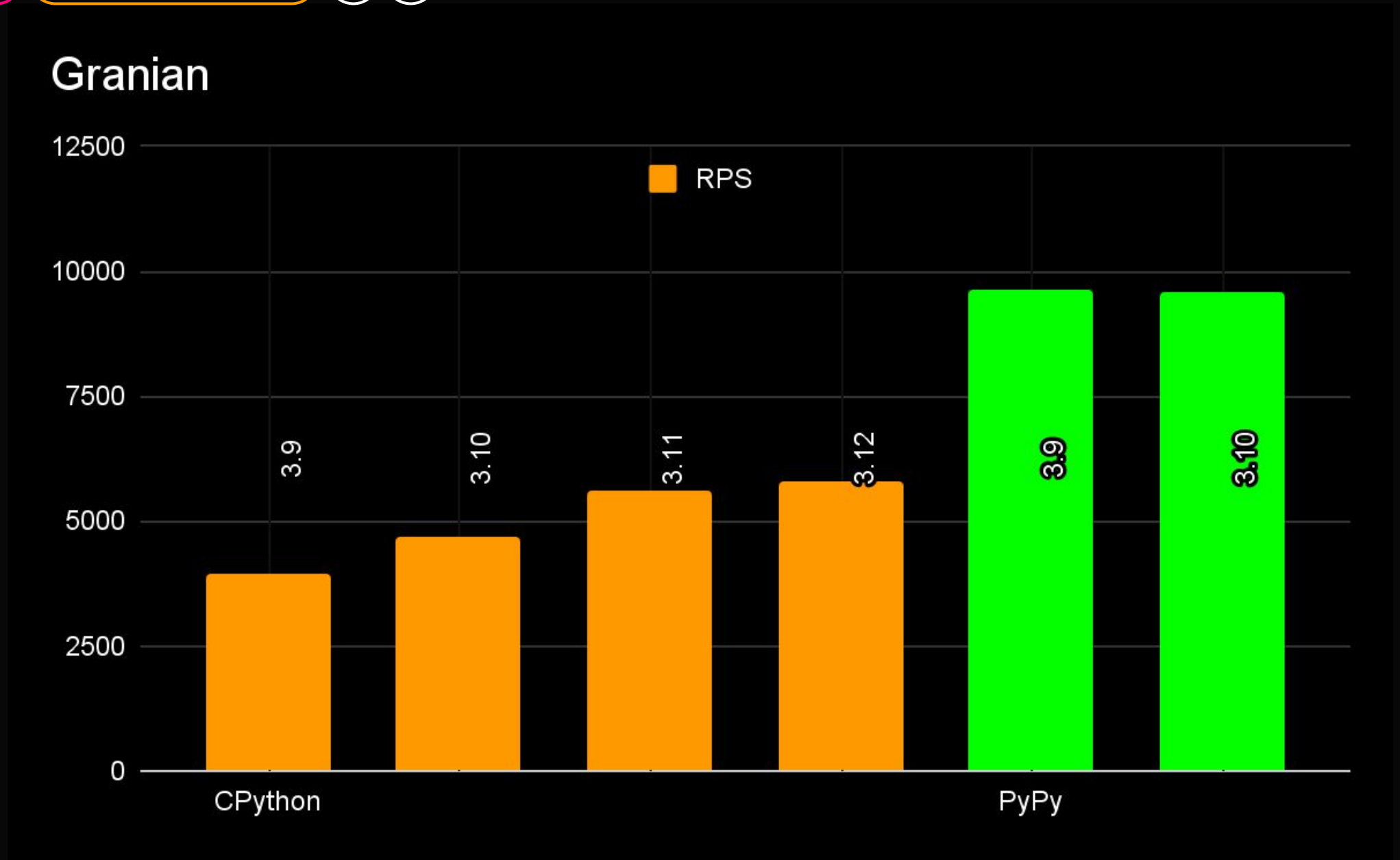

Daphne (Холодный)
12500
10000
7500
5000
2500
0
RPS
3.9
3.10
3.11
3.12
3.9
3.10
CPython
PyPy


Daphne
RPS
12500
10000
7500
5000
2500
0
3.9
3.10
3.11
3.12
3.9
3.10
CPython
PyPy

PyPy быстрее работает
после прогрева


Hypercorn
12500
10000
7500
5000
2500
0
RPS
3.9
3.10
3.11
3.12
3.9
3.10
CPython
PyPy


Uvicorn
12500
10000
7500
5000
2500
0
RPS
3.9
3.10
3.11
3.12
3.9
3.10
CPython
PyPy


Granian
RPS
12500
10000
7500
5000
2500
0
3.9
3.10
3.11
3.12
3.9
3.10
CPython
PyPy

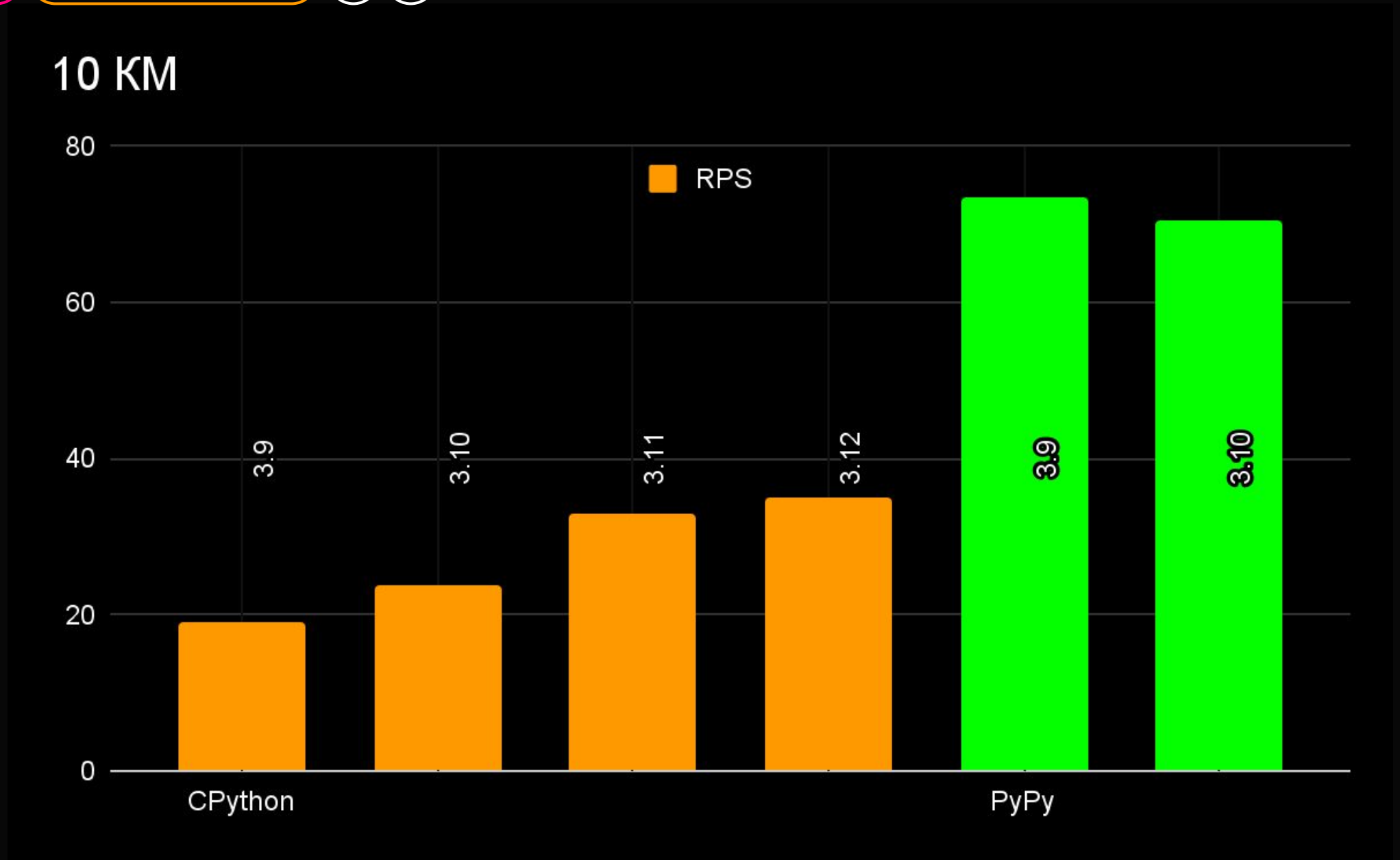
PyPy увеличивает RPS

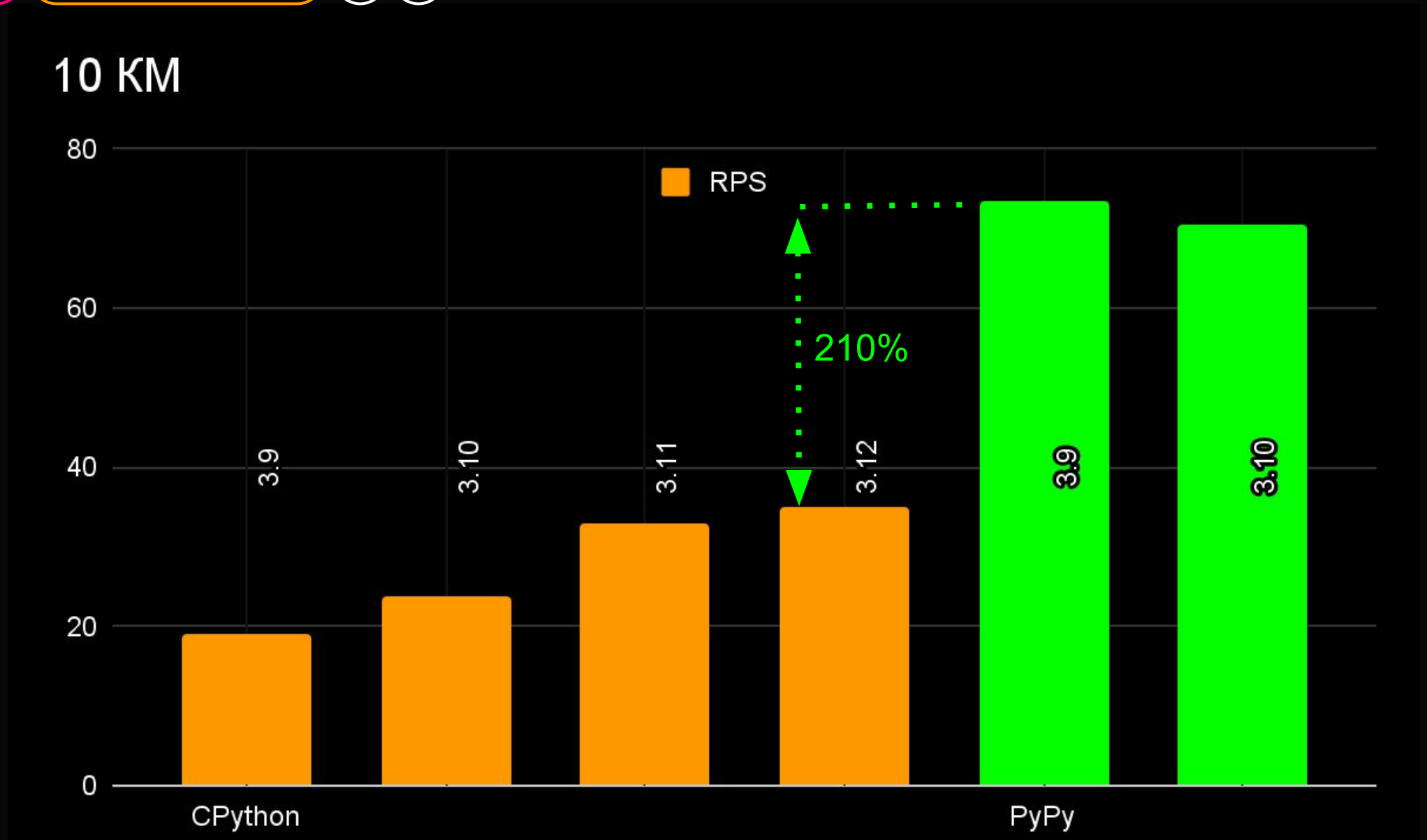
Больше вычислений

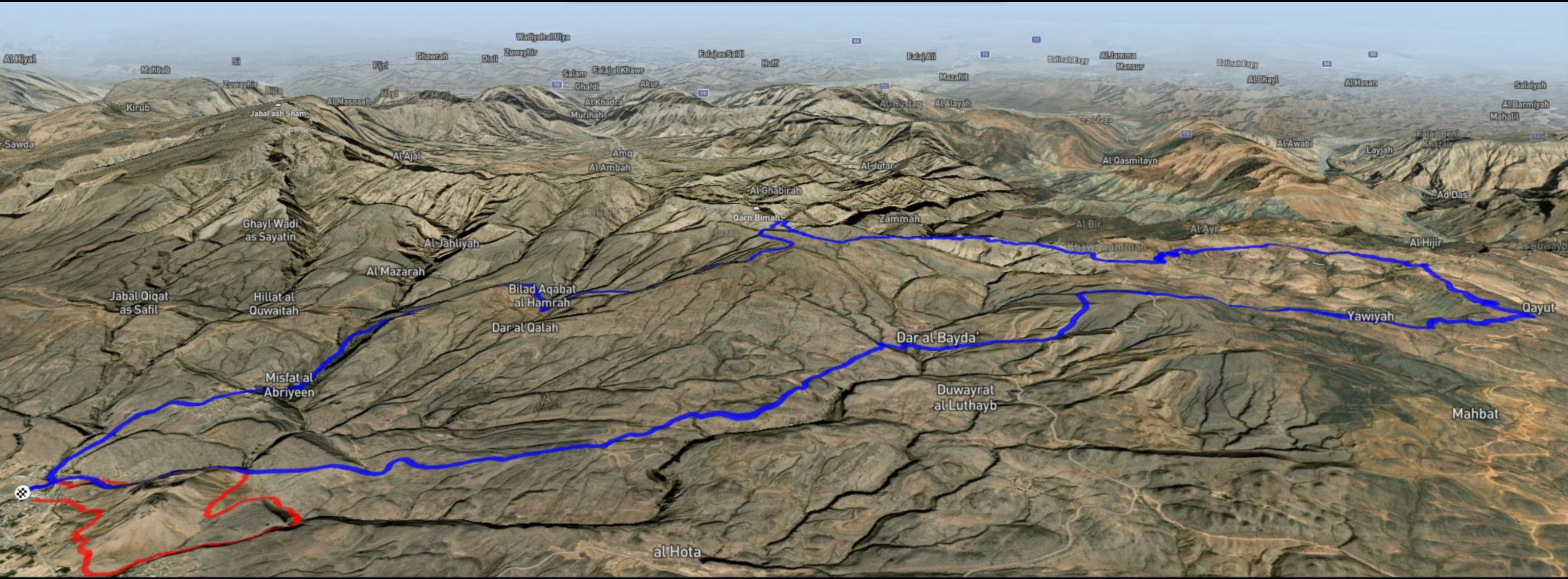
```python
from typing import Iterator, Tuple

from gpxpy import parse
from gpxpy.gpx import GPX


def iter_points(
    gpx: GPX,
) -> Iterator[Tuple[float, float]]:
    for track in gpx.tracks:
        for segment in track.segments:
            for point in segment.points:
                yield point.latitude, point.longitude
```

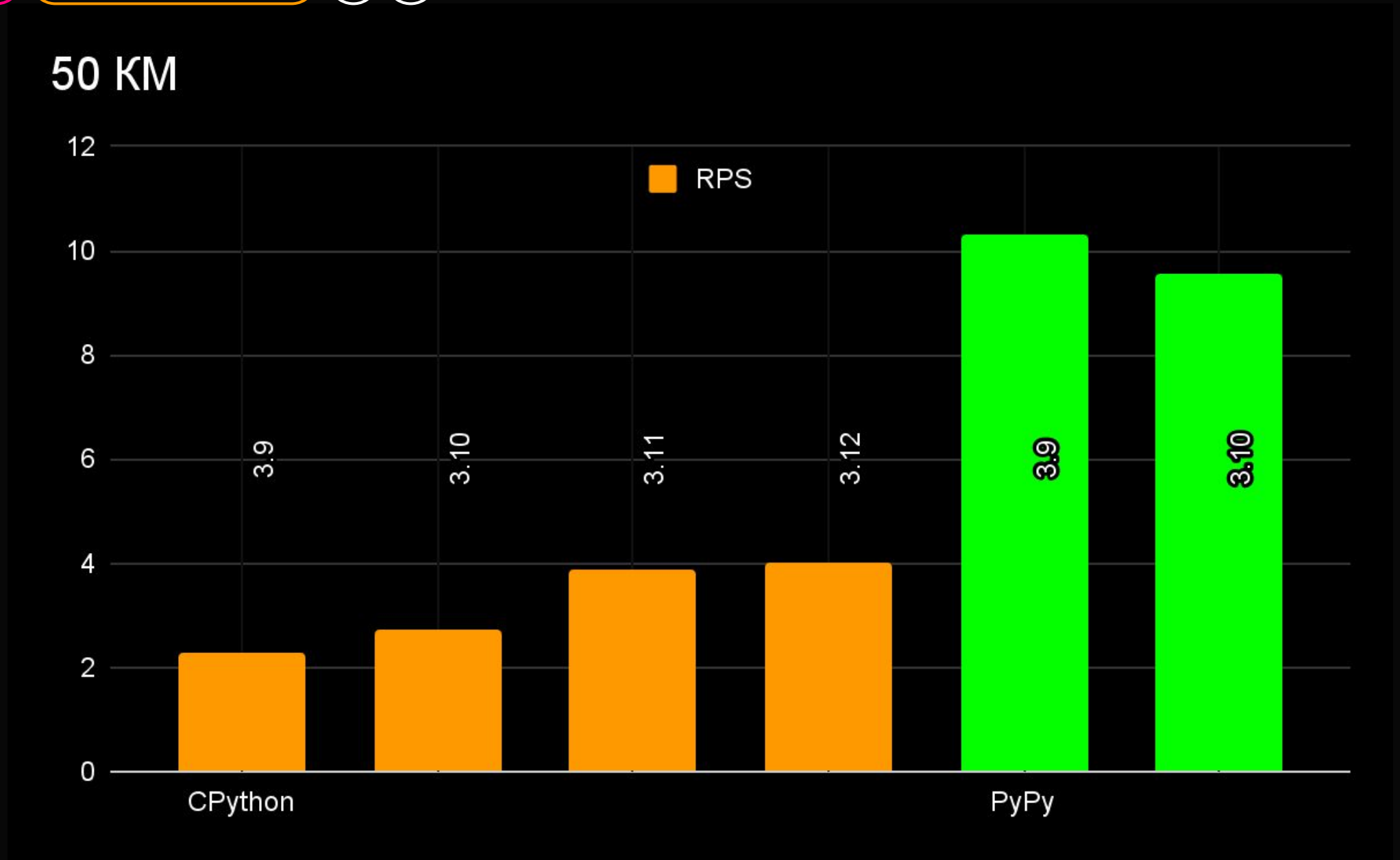
```python
@app.post("/distance")
async def distance(request: Request) -> Distance:
    body = await request.body()
    gpx = parse(body)
    meters = 0.0

    points = iter_points(gpx)
    last_point = next(points)

    for point in points:
        meters += geodesic(last_point, point).meters
        last_point = point

    return Distance(meters=meters)
```

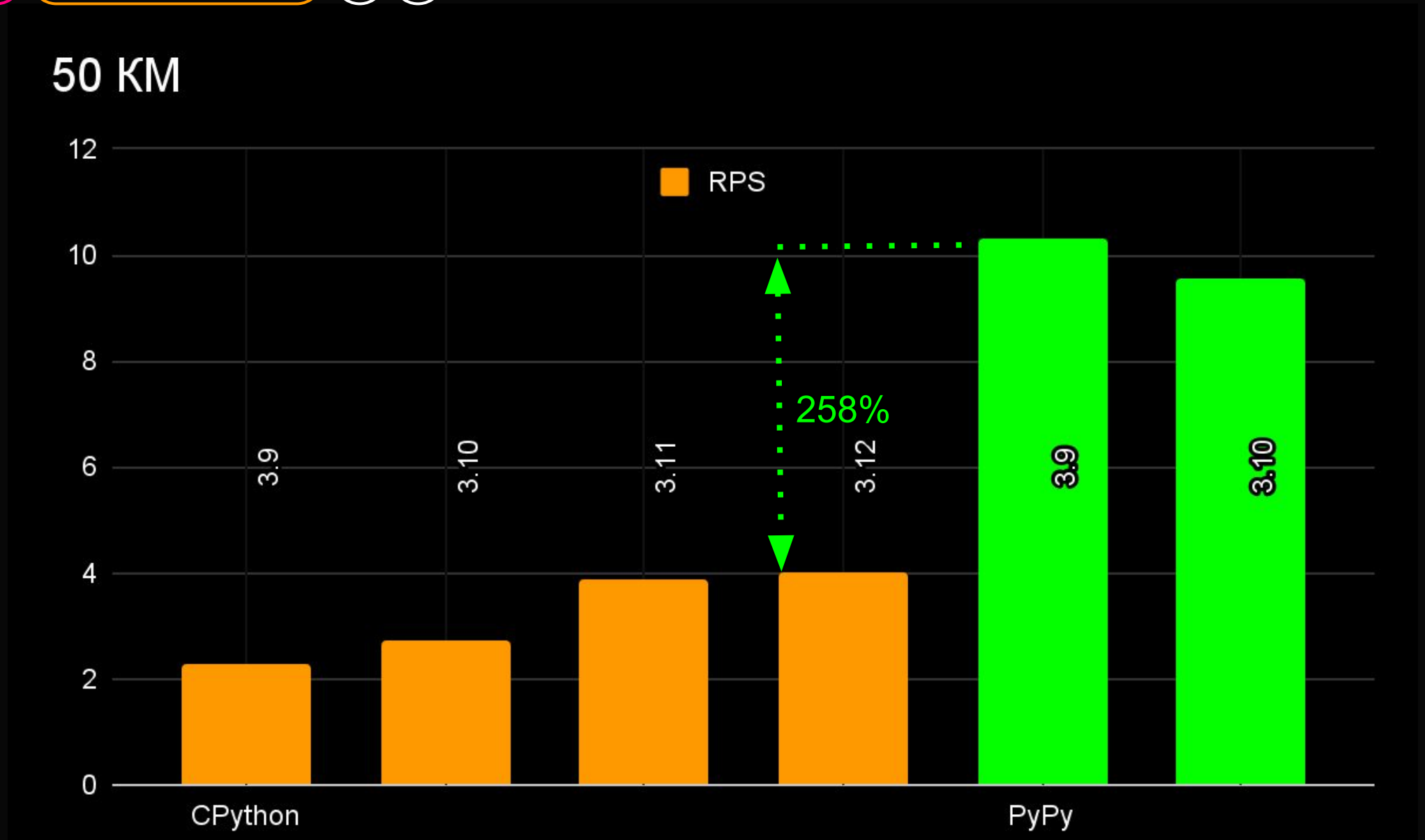

Pension Misfah
Old House
Jabaat al Baqt
The Wadi Inn
Hamra


10 KM
80
60
40
20
0
RPS
3.9
3.10
3.11
3.12
3.9
3.10
CPython
PyPy

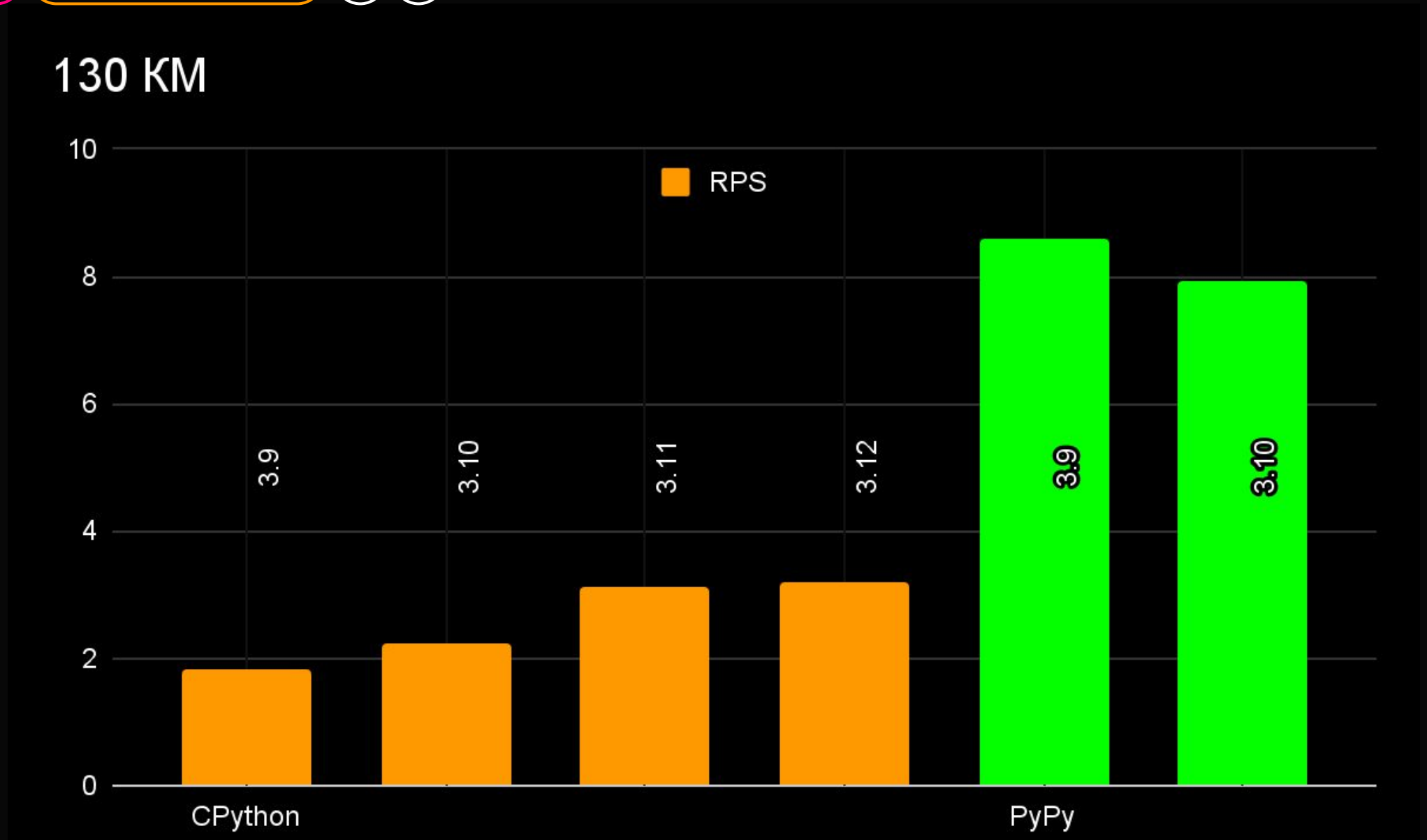

10 KM
80
60
40
20
0
RPS
210%
3.9
3.10
3.11
3.12
3.9
3.10
CPython
PyPy

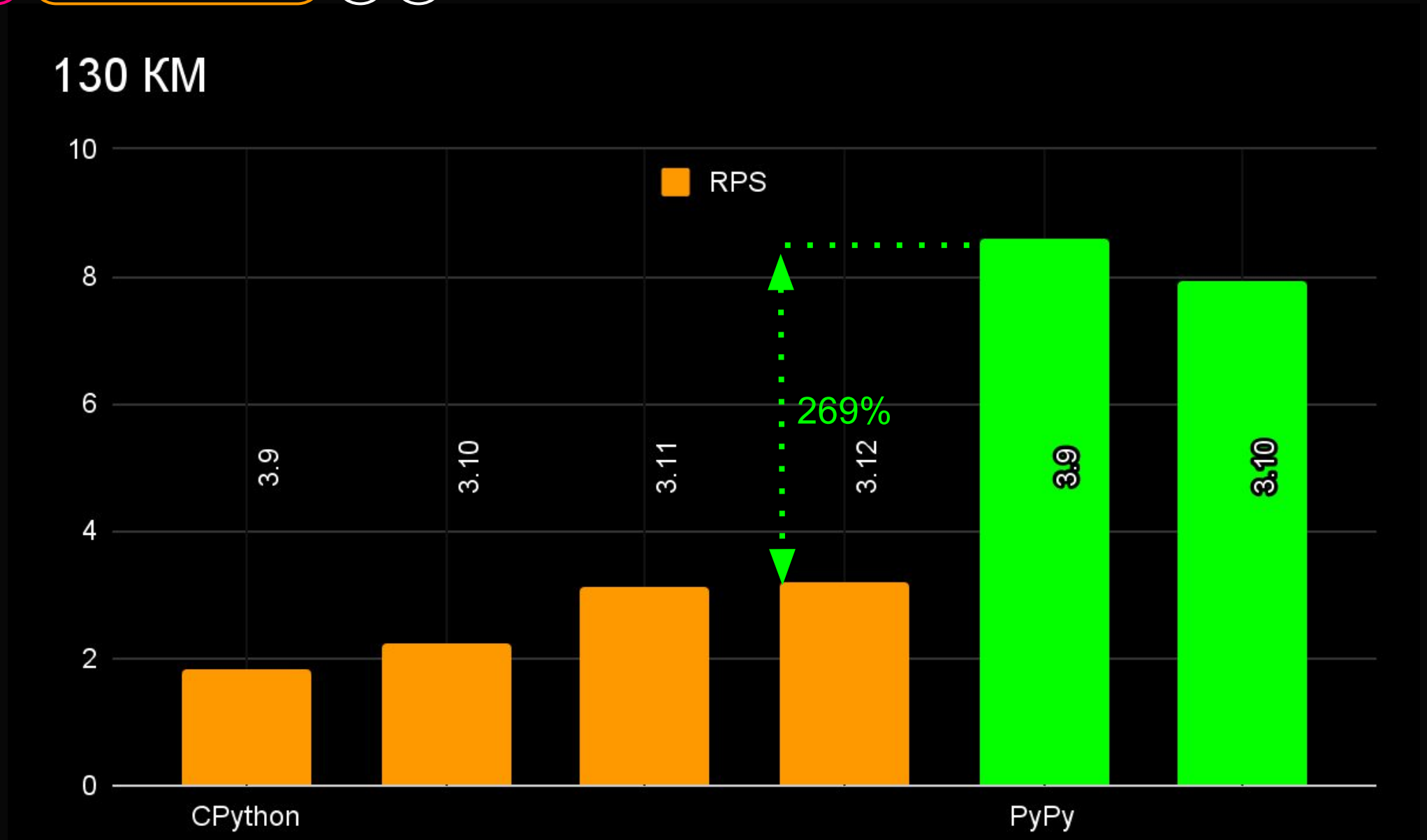

Al Hiyal
Mahbab
Kirub
Sawda
Jabal ash Sham
Al Masnaah
Al Ajal
Amq
Al Ambah
Al Ghabirah
Qarn Biman
Zammah
Al Jufar
Al Qasmitayn
Al Awabi
Layjah
Ad Das
Al Bir
Al Ayr
Al Hijir
Ghayl Wadi as Sayatin
Al Jahliyah
Al Mazarah
Bilad Aqabat al Hamrah
Dar al Qalah
Jabal Qiqat as Safil
Hillat al Quwaitah
Misfat al Abriyeen
Dar al Bayda'
Yawiyah
Qayut
Duwayrat al Luthayb
Mahbat
al Hota


50 KM
RPS
12
10
8
6
4
2
0
3.9
3.10
3.11
3.12
3.9
3.10
CPython
PyPy


50 KM
12
10
8
6
4
2
0
RPS
258%
3.9
3.10
3.11
3.12
3.9
3.10
CPython
PyPy

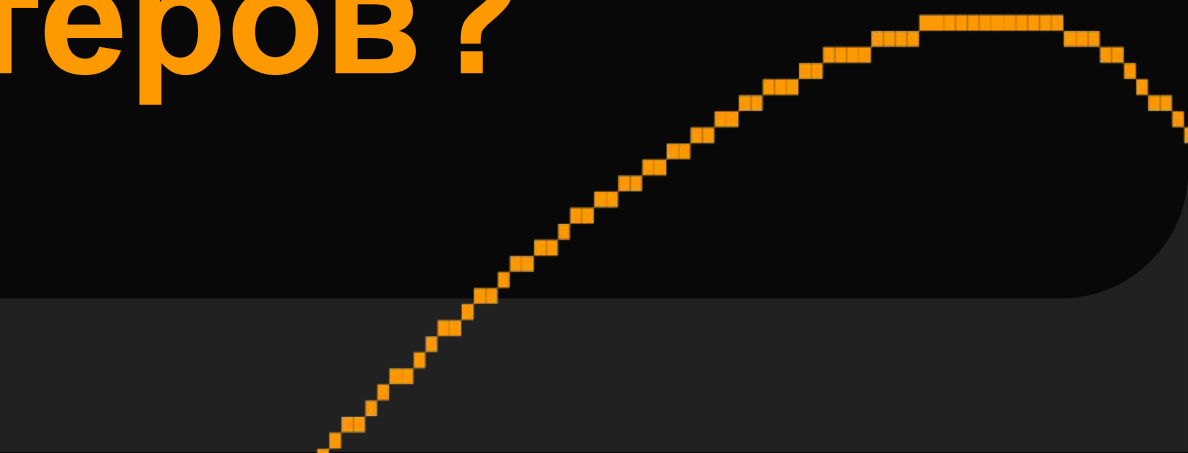

Al Ham
Dar al Qalah
Dar al Bayda
Mahbat
Ghubrat Tanuf
Rawdah
Al Abyad
Nizwa
Fuluj Sayghah
Rakhbah
Wasit
Misfah
Muaydin
Hayl al Liwa
Birkat al Mawz
Ras al Kabul
Al Ababah
Qarn al Khuzar
Al Ayn
Al Ala
Dann al Hamra
Al Huwir
Al Maqul
Al Gharir Alayah
Al Ghalail
Al Khutaymah
Qayut Halhal
Bilad Sayt
Al Ghabirah
Al Jufar
Al Rustaq
Al Musannah
Ath Tharmad
As Suwaiq
Sukhaylah Qarn
Wakan
Hadash
Shaww
Amti
Nakhal
Wadi Almaawil
Barka
Seeb
Sumail
Rusayl
Fanja
Masirat Nakhr
Dayqarah
Al Hiyal
Hazm Qāshi'
Maydan
Al Masnaah
Jabal ash Sham
Ghayl Wadi as Sayatin
Al Ajal
Al Jahliyah
Al Qalah


130 КМ
RPS
10
8
6
4
2
0
3.9
3.10
3.11
3.12
3.9
3.10
CPython
PyPy


130 КМ
10
8
6
4
2
0
RPS
269%
3.9
3.10
3.11
3.12
3.9
3.10
CPython
PyPy

PyPy ускоряет длительные вычисления в коде

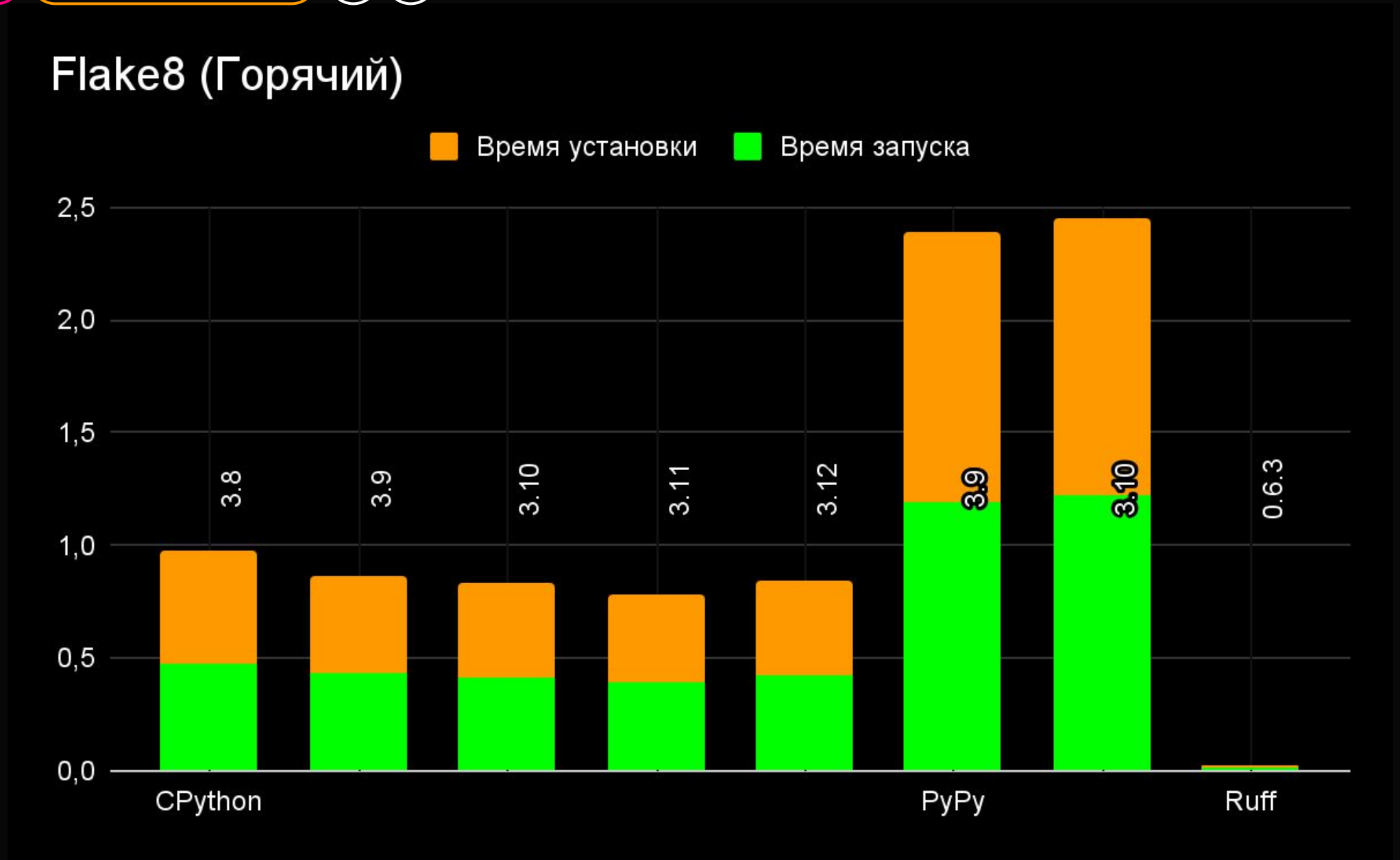
Что насчёт линтеров?




Flake8 (Холодный)
Время установки
Время запуска
8
6
4
2
0
3.8
3.9
3.10
3.11
3.12
3.9
3.10
0.6.3
CPython
PyPy
Ruff


MyPy (Холодный)
Время установки
Время запуска
40
30
20
10
0
3.8
3.9
3.10
3.11
3.12
3.9
3.10
CPython
PyPy

PyPy ускоряет холодный запуск линтера


Flake8 (Горячий)
Время установки
Время запуска
2,5
2,0
1,5
1,0
0,5
0,0
3.8
3.9
3.10
3.11
3.12
3.9
3.10
0.6.3
CPython
PyPy
Ruff


MyPy (Горячий)
Время установки
Время запуска
0,8
0,6
0,4
0,2
0,0
3.8
3.9
3.10
3.11
3.12
3.9
3.10
CPython
PyPy

JIT не успевает прогреться при повторном запуске линтера

Недостатки PyPy

Недостатки PyPy

- Время на прогрев JIT

Недостатки PyPy

- Время на прогрев JIT
- Потребление памяти

Недостатки PyPy

- Время на прогрев JIT
- Потребление памяти
- Совместимость с библиотеками

Недостатки PyPy

- Время на прогрев JIT
- Потребление памяти
- Совместимость с библиотеками
- Отставание версий

Альтернативы PyPy

Альтернативы PyPy

- mypyc

Альтернативы PyPy

- mypyc
- cython

Альтернативы PyPy

- mypyc
- cython
- nuitka

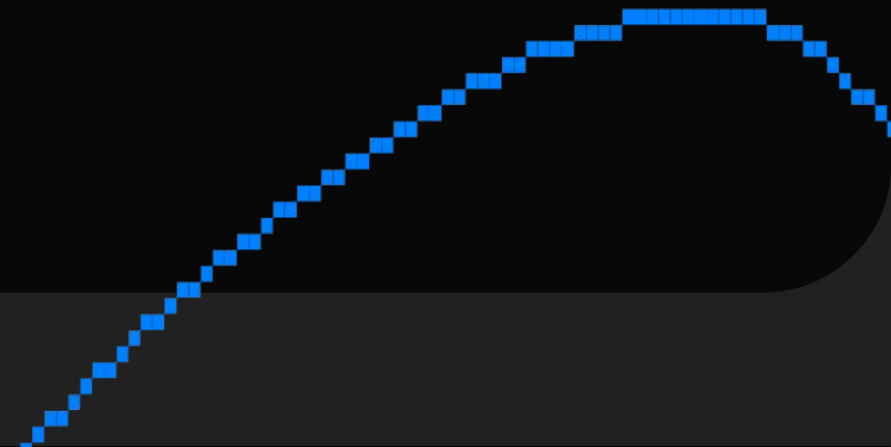
Альтернативы PyPy

- mypyc
- cython
- nuitka
- numba

Альтернативы PyPy

- mypyc
- cython
- nuitka
- numba
- mojo

PyPy 3.10 vs Python 3.13



Что нового в Python 3.13

- Free-threaded CPython (PEP 703)
- Экспериментальный JIT-компилятор (PEP 744)
- Точечные оптимизации

Что нового в PyPy

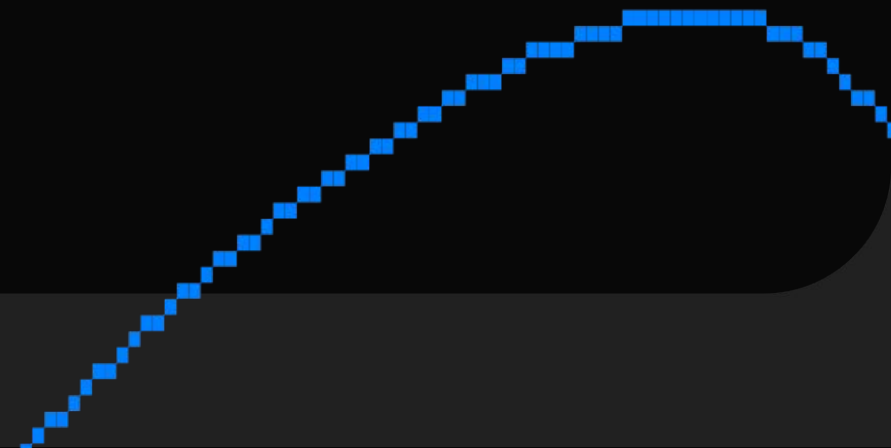
- JIT-оптимизации целочисленных операций (knownbits)

```
x = a | 1

...

if x & 1:
    ...

else:
    ...
```

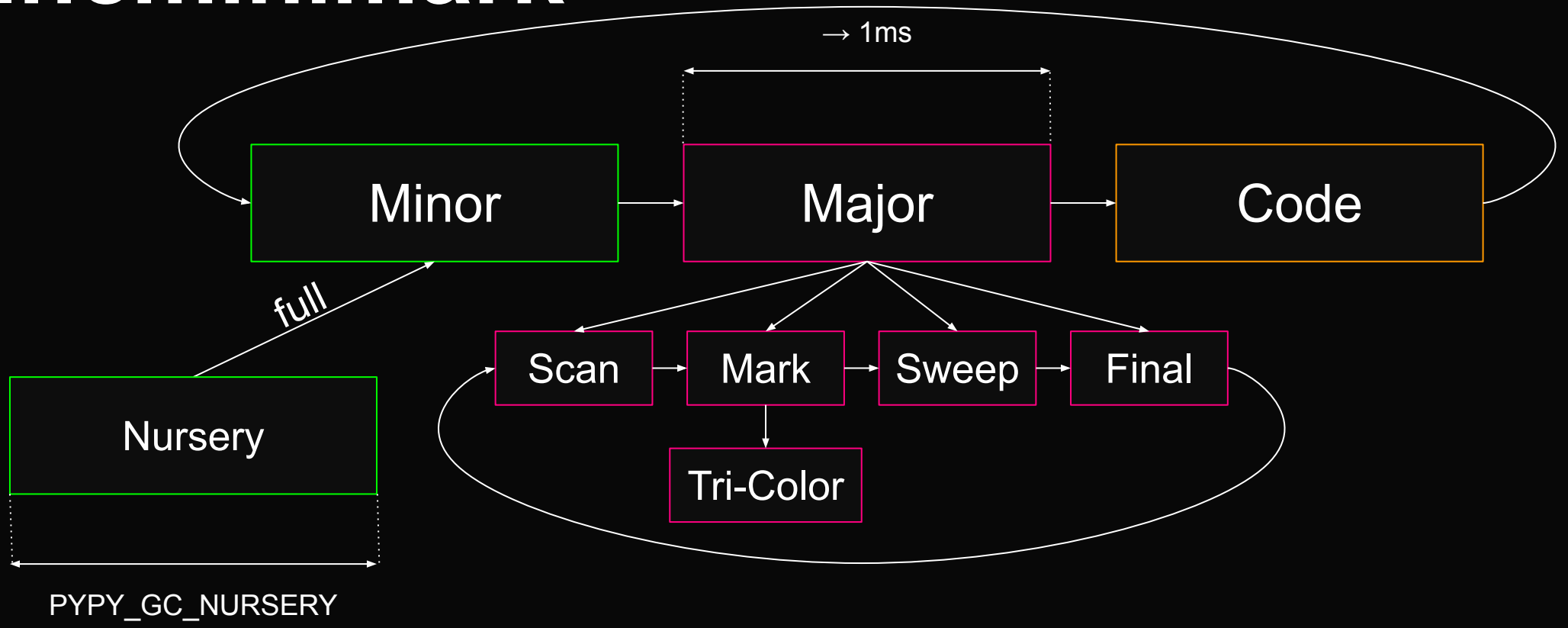
Что нового в PyPy

- JIT-оптимизации целочисленных операций (knownbits)
- Поддержка RISC-V для JIT
- Улучшения REPL
- HPY 0.9 / CFFI 1.16 / C API
- Security fixes / backports

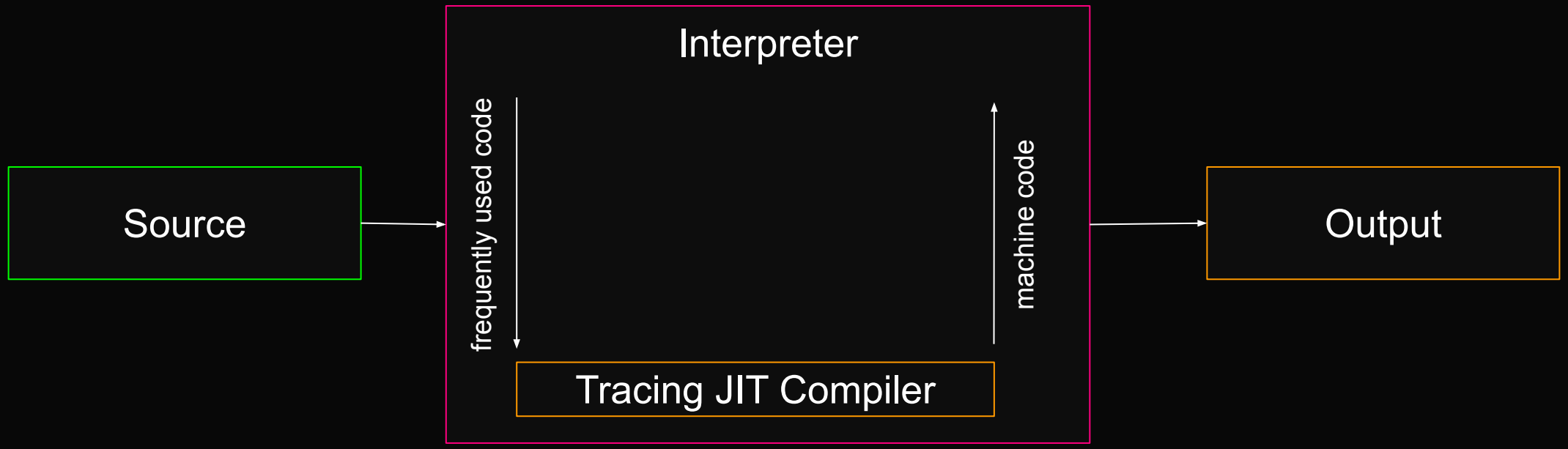
Что ещё важно знать про PyPy



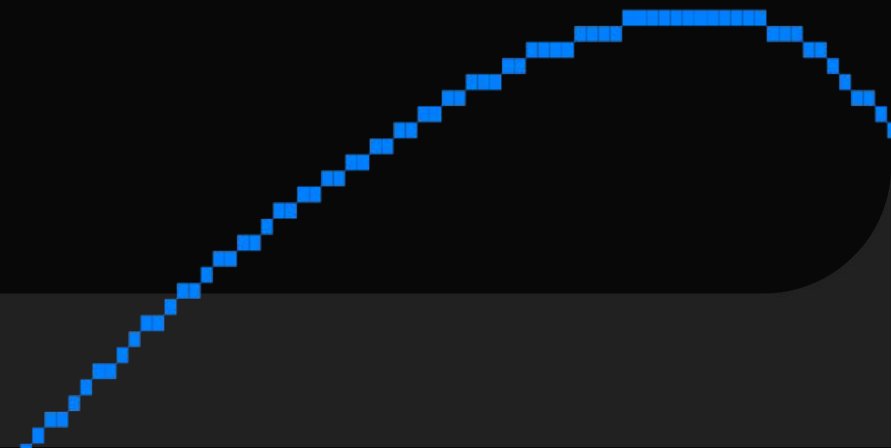
Incminimark

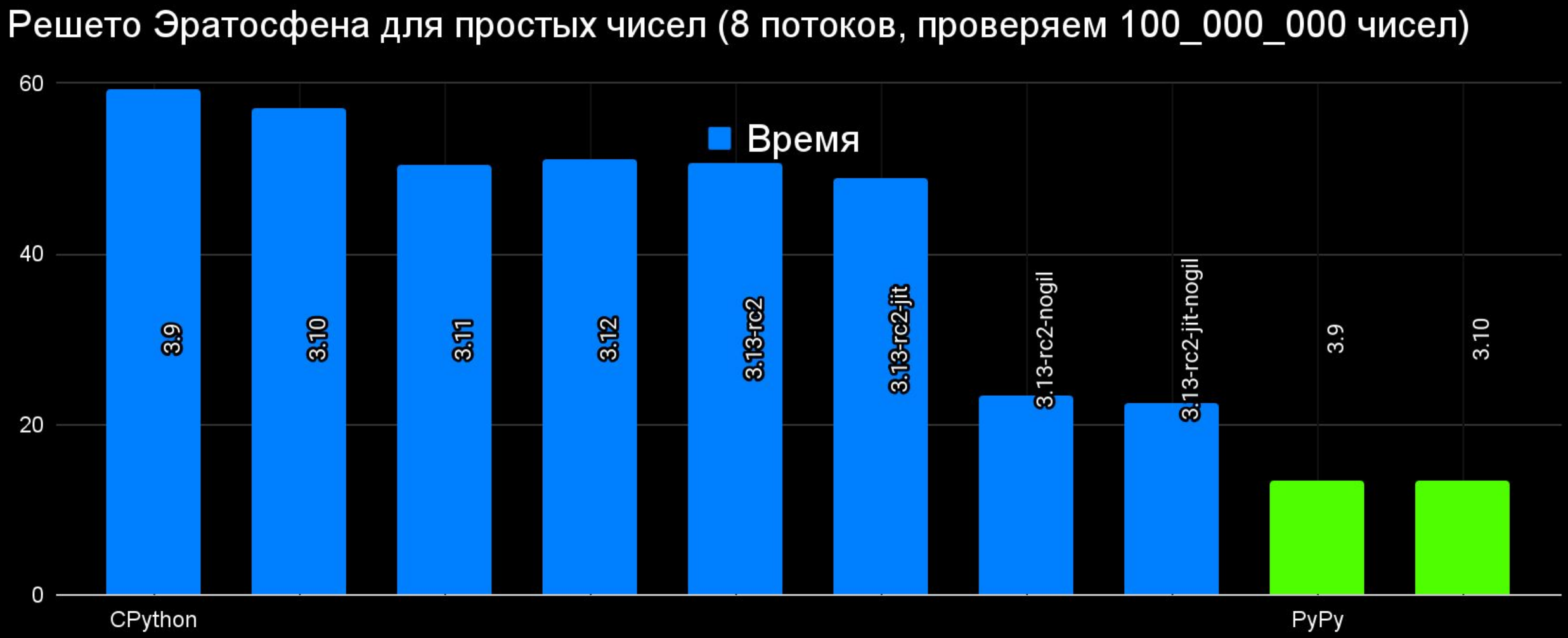


Tracing JIT

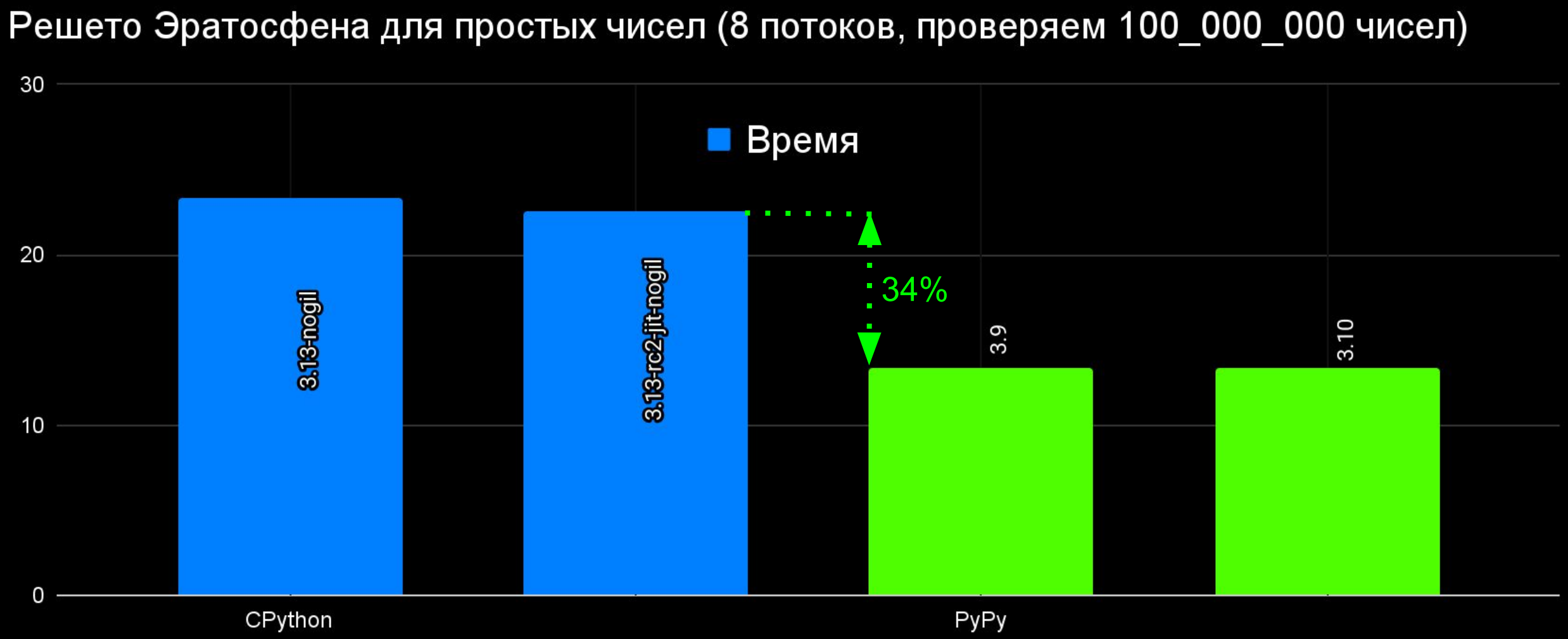

Interpreter
frequently used code
machine code
Source
Output
Tracing JIT Compiler

А что там с nogil?




Решето Эратосфена для простых чисел (8 потоков, проверяем 100_000_000 чисел)
60
40
20
0
Время
3.9
3.10
3.11
3.12
3.13-rc2
3.13-rc2-jit
3.13-rc2-nogil
3.13-rc2-jit-nogil
3.9
3.10
CPython
PyPy

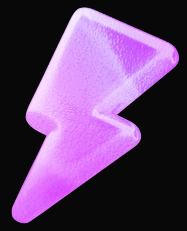

Решето Эратосфена для простых чисел (8 потоков, проверяем 100_000_000 чисел)
30
20
10
0
Время
3.13-nogil
3.13-rc2-jit-nogil
3.9
3.10
CPython
PyPy


Решето Эратосфена для простых чисел (8 потоков, проверяем 100_000_000 чисел)
30
20
10
0
Время
3.13-nogil
3.13-rc2-jit-nogil
34%
3.9
3.10
CPython
PyPy

PyPy неплохо показывает себя в многопоточных задачах

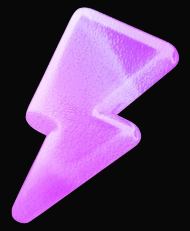
Выводы

Ещё раз о главном



Ускоряет вычисления
и алгоритмы

Лучше всего работает
с чистым Python



Не полностью
совместим с C API

Стоит ли использовать PyPy

Стоит ли использовать PyPy?

Да, если:

- Преобладают CPU-bound операции
- Преобладает чистый Python

Нет, если:

- Преобладают IO-bound операции
- Используется C API*
- Требуется Python > 3.10

Спасибо за внимание

sharupoff_code

sharupoff

Контур

Андрей Шарапов