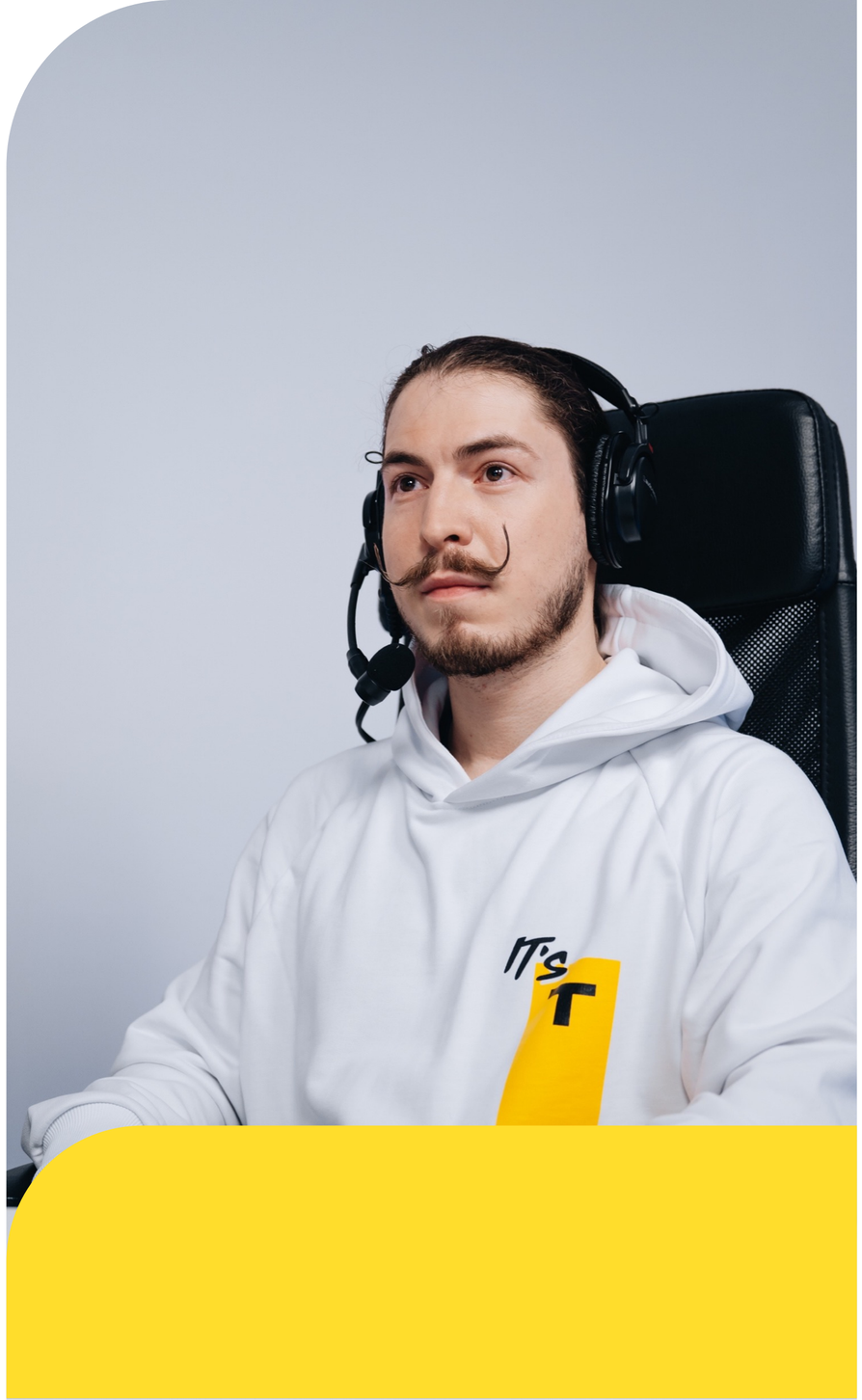
IT's TINKOFF

Python + Rust = ?

Как это работает

Данил Ахтаров

07.11.2023



Данил Ахтаров

Архитектор

github.com/daxartio

t.me/daxtar

Agenda

Наш путь сегодня

Введение

- Почему?
- Когда?
- Кто?

Как это работает

Как можно использовать другие языки в Python?

Практика

Быстрый старт

Напишем свою библиотеку с PyO3

Почему иногда “тупое” использование не даёт никаких результатов

Почему Rust?

Причины

Почему выбирают Rust?

- Скорость. Безопасность. Компилируемый ЯП без GC

Причины

Почему выбирают Rust?

- Скорость. Безопасность. Компилируемый ЯП без GC
- Microsoft переписывает ядро Windows на Rust
- В ядре андроида используется Rust ~21% нового кода
- В ядре Linux используется Rust

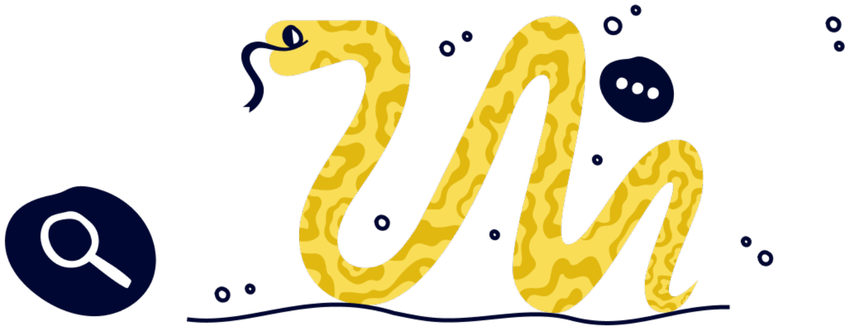
Причины

Почему выбирают Rust?

- Скорость. Безопасность. Компилируемый ЯП без GC
- Microsoft переписывает ядро Windows на Rust
- В ядре андроида используется Rust ~21% нового кода
- В ядре Linux используется Rust

“То, что Rust на хайпе — не просто так”

Когда нужно использовать Rust?



Когда нужно использовать Rust?

C/C++, Go

Если решили использовать другой ЯП, чтобы ускорить программу

CPU-bound задачи

Сериализация / десериализация

Шифрование / дешифрование

Производительность

Не можете горизонтально масштабировать приложение

Безопасность

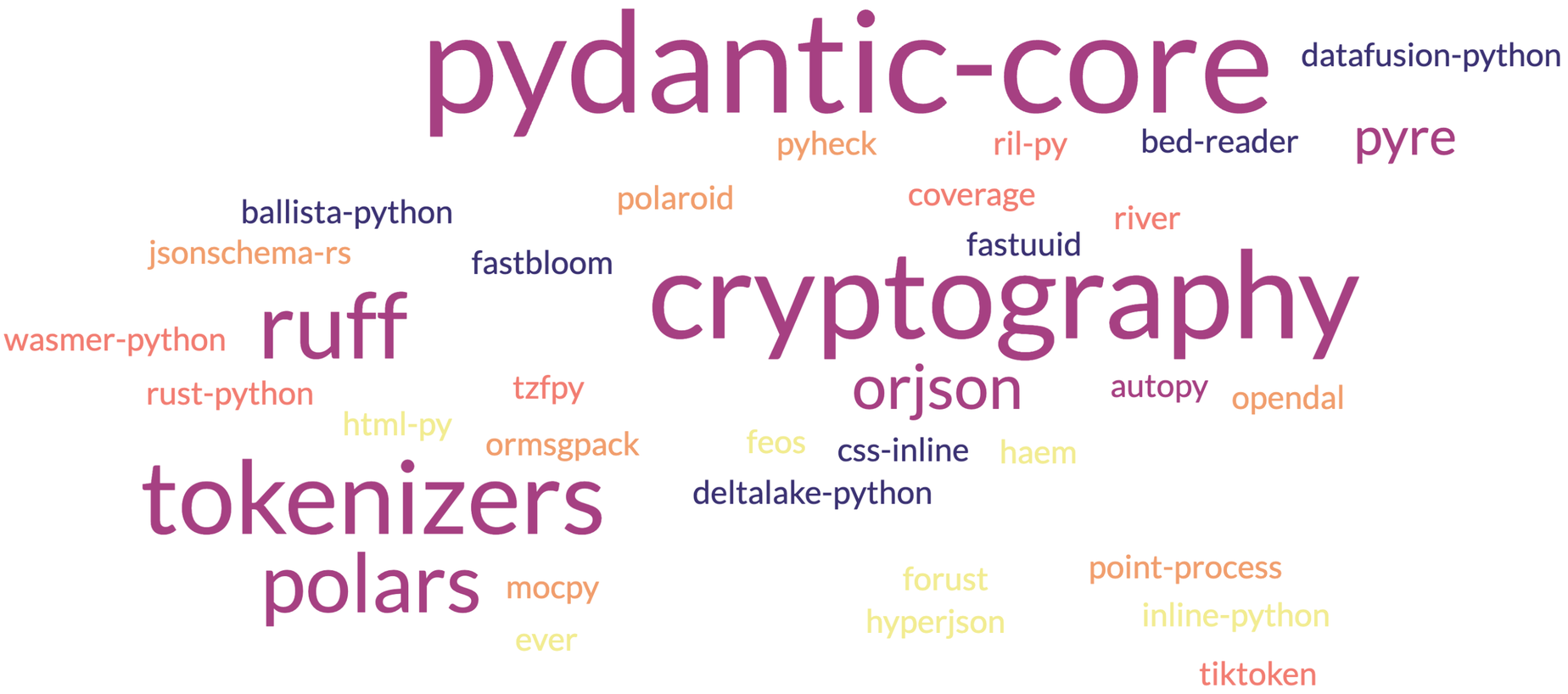
Если у вас чистая БЛ, вы можете написать её на Rust

Безопасность

Как в небольшой команде переписать узкие места на Rust

Solana

Библиотеки


pydantic-core
datafusion-python
pyheck
ril-py
bed-reader
pyre
ballista-python
polaroid
coverage
river
jsonschema-rs
fastbloom
fastuuid
cryptography
ruff
wasmer-python
rust-python
tzfpy
orjson
autopy
opendal
html-py
ormsgpack
feos
css-inline
haem
tokenizers
deltalake-python
polars
mocpy
forust
point-process
hyperjson
inline-python
ever
tiktoken


PyO3

FFI — Foreign function interface

— это механизм, с помощью которого программа, написанная на одном языке программирования, может вызывать подпрограммы или использовать услуги, написанные или скомпилированные на другом языке. © вики

ctypes

```
 1 import ctypes
 2
 3 libc = ctypes.cdll.LoadLibrary('libc.so.6')
 4
 5 # Define the function signature
 6 puts = libc.puts
 7 puts.argtypes = [ctypes.c_char_p]
 8 puts.restype = ctypes.c_int
 9
10 # Call the function
11 puts(b"Hello, world!")
```

ctypes

```python
import ctypes

# Allocate a block of memory in Python
buf = bytearray(b"Hello, world!")
print(buf)

# Get a pointer to the memory block
ptr = ctypes.cast(buf, ctypes.c_void_p)

# Call a C function with the pointer
libc = ctypes.cdll.LoadLibrary('libc.so.6')
libc.puts(ptr)
```

Pythonium Trioxide

Описание

PyO3 — это Rust библиотека для создания Python расширений
и встраивания Python в Rust.

Предоставляет высокоуровневый API

PyO3 построен на основе API Python C, но обеспечивает более безопасный и идиоматический интерфейс Rust.

Пример

maturin

maturin — это инструмент для создания и публикации пакетов Python на основе Rust с минимальной настройкой.

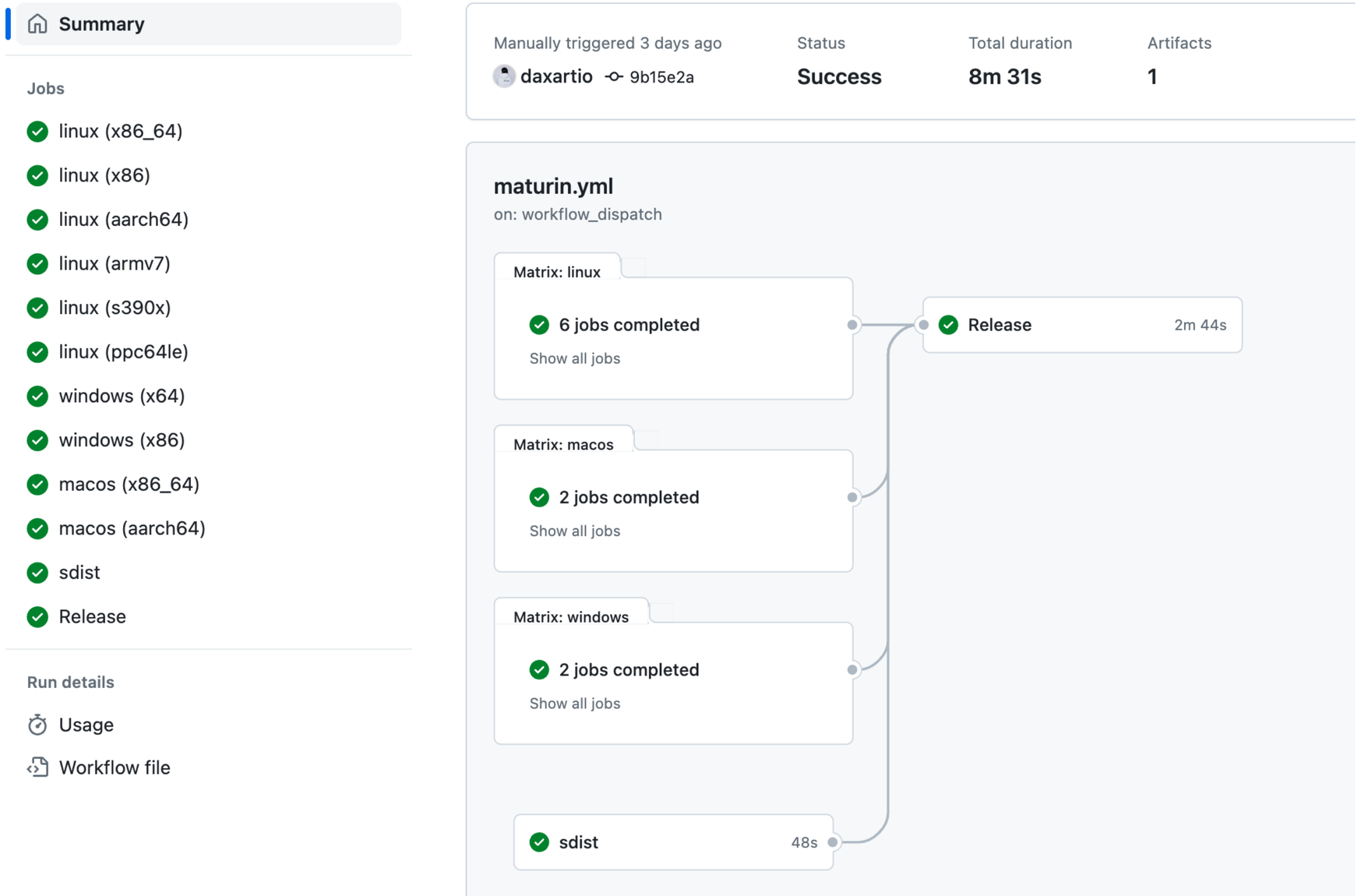
1. Создаем проект

```
$ pip install maturin
$ maturin init
$ maturin develop
```

2. Запускаем

```
$ python
>>> import mypackage
>>> mypackage.hello()
"Hello from Py03!"
```

Публикация пакета



Постановка задачи

Конвертация строки из `snake_case` в `camelCase`

Мы предоставляем `camelCase` формат данных API, а внутри используем `snake_case`

```python
from pydantic import BaseModel, Field

class User(BaseModel):
    first_name: str = Field(..., alias="firstName")
    last_name: str = Field(..., alias="lastName")
```

Alias generator

```
 1 from pydantic import BaseModel, ConfigDict
 2
 3
 4 class CamelAliases(BaseModel):
 5     model_config = ConfigDict(populate_by_name=True, alias_generator=to_camel)
 6
 7
 8 class User(CamelAliases):
 9     first_name: str
10     last_name: str
11
12 User(first_name="John", last_name="Doe").model_dump(by_alias=True)
13 '{"firstName": "John", "lastName": "Doe"}'
```

Реализация на Python

```
def to_camel(string: str) -> str:
    components = string.split("_")
    return components[0] + "".join(word.title() for word in components[1:])
```

Реализация на Rust

```rust
use pyo3::prelude::*;

/// Convert to camel case.
#[pyfunction]
fn to_camel(s: &str) -> PyResult<String> {
    let mut result = String::with_capacity(s.len());
    let mut capitalize_next = false;
    for c in s.chars() {
        if c == '_' {
            capitalize_next = true;
        } else {
            if capitalize_next {
                result.push(c.to_ascii_uppercase());
                capitalize_next = false;
            } else {
                result.push(c.to_ascii_lowercase());
            }
        }
    }
    Ok(result)
}
/// Casers package.
#[pymodule]
fn _casers(_py: Python, m: &PyModule) -> PyResult<()> {
    m.add_function(wrap_pyfunction!(to_camel, m)?)?;
    Ok(())
}
```

Реализация на Rust

```rust
use pyo3::prelude::*;

/// Convert to camel case.
#[pyfunction]
fn to_camel(s: &str) -> PyResult<String> {
    let mut result = String::with_capacity(s.len());
    let mut capitalize_next = false;
    for c in s.chars() {
        if c == '_' {
            capitalize_next = true;
        } else {
            if capitalize_next {
                result.push(c.to_ascii_uppercase());
                capitalize_next = false;
            } else {
                result.push(c.to_ascii_lowercase());
            }
        }
    }
    Ok(result)
}
/// Casers package.
#[pymodule]
fn _casers(_py: Python, m: &PyModule) -> PyResult<()> {
    m.add_function(wrap_pyfunction!(to_camel, m)?)?;
    Ok(())
}
```

Реализация на Rust

```rust
use pyo3::prelude::*;

/// Convert to camel case.
#[pyfunction]
fn to_camel(s: &str) -> PyResult<String> {
    let mut result = String::with_capacity(s.len());
    let mut capitalize_next = false;
    for c in s.chars() {
        if c == '_' {
            capitalize_next = true;
        } else {
            if capitalize_next {
                result.push(c.to_ascii_uppercase());
                capitalize_next = false;
            } else {
                result.push(c.to_ascii_lowercase());
            }
        }
    }
    Ok(result)
}
/// Casers package.
#[pymodule]
fn _casers(_py: Python, m: &PyModule) -> PyResult<()> {
    m.add_function(wrap_pyfunction!(to_camel, m)?)?;
    Ok(())
}
```

Реализация на Rust

```rust
use pyo3::prelude::*;

/// Convert to camel case.
#[pyfunction]
fn to_camel(s: &str) -> PyResult<String> {
    let mut result = String::with_capacity(s.len());
    let mut capitalize_next = false;
    for c in s.chars() {
        if c == '_' {
            capitalize_next = true;
        } else {
            if capitalize_next {
                result.push(c.to_ascii_uppercase());
                capitalize_next = false;
            } else {
                result.push(c.to_ascii_lowercase());
            }
        }
    }
    Ok(result)
}
/// Casers package.
#[pymodule]
fn _casers(_py: Python, m: &PyModule) -> PyResult<()> {
    m.add_function(wrap_pyfunction!(to_camel, m)?)?;
    Ok(())
}
```

Реализация на Rust

```rust
use pyo3::prelude::*;

/// Convert to camel case.
#[pyfunction]
fn to_camel(s: &str) -> PyResult<String> {
    let mut result = String::with_capacity(s.len());
    let mut capitalize_next = false;
    for c in s.chars() {
        if c == '_' {
            capitalize_next = true;
        } else {
            if capitalize_next {
                result.push(c.to_ascii_uppercase());
                capitalize_next = false;
            } else {
                result.push(c.to_ascii_lowercase());
            }
        }
    }
    Ok(result)
}
/// Casers package.
#[pymodule]
fn _casers(_py: Python, m: &PyModule) -> PyResult<()> {
    m.add_function(wrap_pyfunction!(to_camel, m)?)?;
    Ok(())
}
```

Python	Rust	Rust (Python-native)
`object`	-	`&PyAny`
`str`	`String`, `Cow<str>`, `&str`, `OsString`, `PathBuf`, `Path`	`&PyString`, `&PyUnicode`
`bytes`	`Vec<u8>`, `&[u8]`, `Cow<[u8]>`	`&PyBytes`
`bool`	`bool`	`&PyBool`
`int`	`i8`, `u8`, `i16`, `u16`, `i32`, `u32`, `i64`, `u64`, `i128`, `u128`, `isize`, `usize`, `num_bigint::BigInt`[1], `num_bigint::BigUint`[1]	`&PyLong`
`float`	`f32`, `f64`	`&PyFloat`
`complex`	`num_complex::Complex`[2]	`&PyComplex`
`list[T]`	`Vec<T>`	`&PyList`
`dict[K, V]`	`HashMap<K, V>`, `BTreeMap<K, V>`, `hashbrown::HashMap<K, V>`[3], `indexmap::IndexMap<K, V>`[4]	`&PyDict`

Реализация на Rust

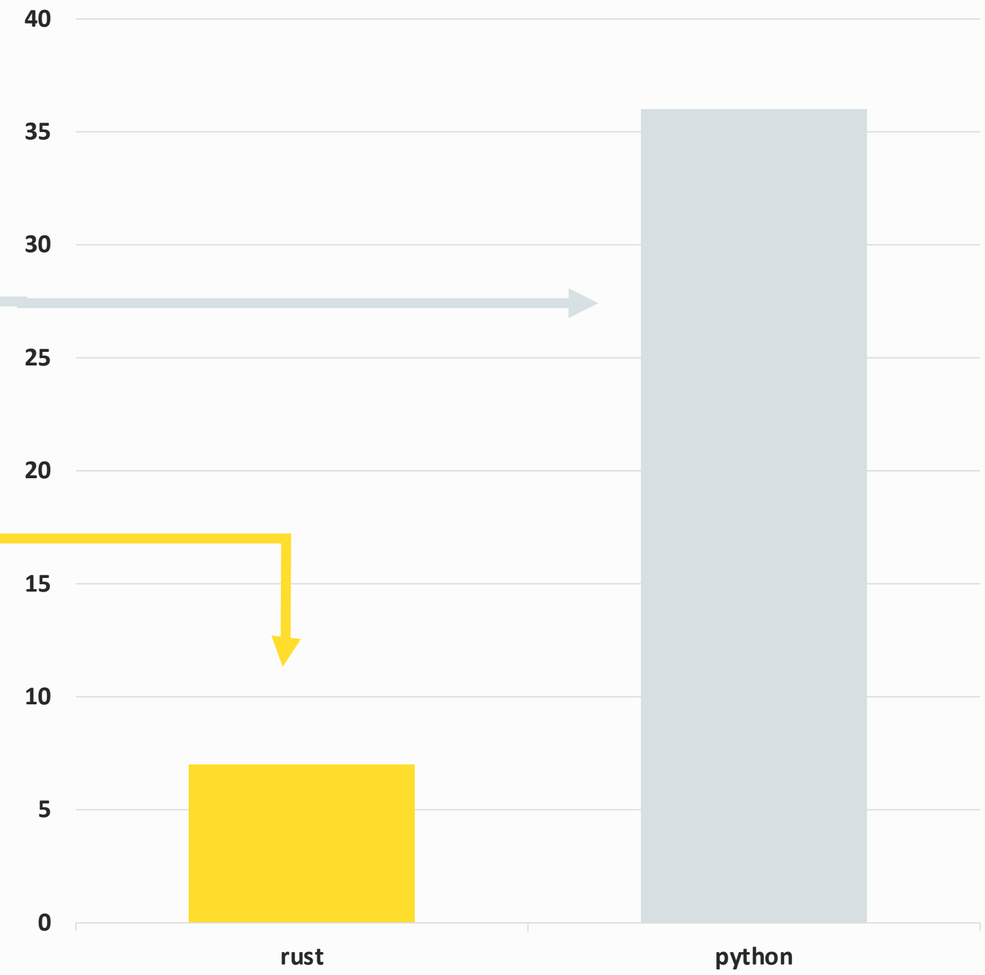
```rust
use pyo3::prelude::*;

/// Convert to camel case.
#[pyfunction]
fn to_camel(s: &str) -> PyResult<String> {
    let mut result = String::with_capacity(s.len());
    let mut capitalize_next = false;
    for c in s.chars() {
        if c == '_' {
            capitalize_next = true;
        } else {
            if capitalize_next {
                result.push(c.to_ascii_uppercase());
                capitalize_next = false;
            } else {
                result.push(c.to_ascii_lowercase());
            }
        }
    }
    Ok(result)
}
/// Casers package.
#[pymodule]
fn _casers(_py: Python, m: &PyModule) -> PyResult<()> {
    m.add_function(wrap_pyfunction!(to_camel, m)?)?;
    Ok(())
}
```

Benchmark

1149 байт

```python
def to_camel(string: str) -> str:
    components = string.split("_")
    return components[0] + "".join(word.title() for word in components[1:])
```

```rust
#[pyfunction]
fn to_camel(s: &str) -> PyResult<String> {
    let mut result = String::with_capacity(s.len());
    let mut capitalize_next = false;
    for c in s.chars() {
        if c == '_' {
            capitalize_next = true;
        } else {
            if capitalize_next {
                result.push(c.to_ascii_uppercase());
                capitalize_next = false;
            } else {
                result.push(c.to_ascii_lowercase());
            }
        }
    }
    Ok(result)
}
```



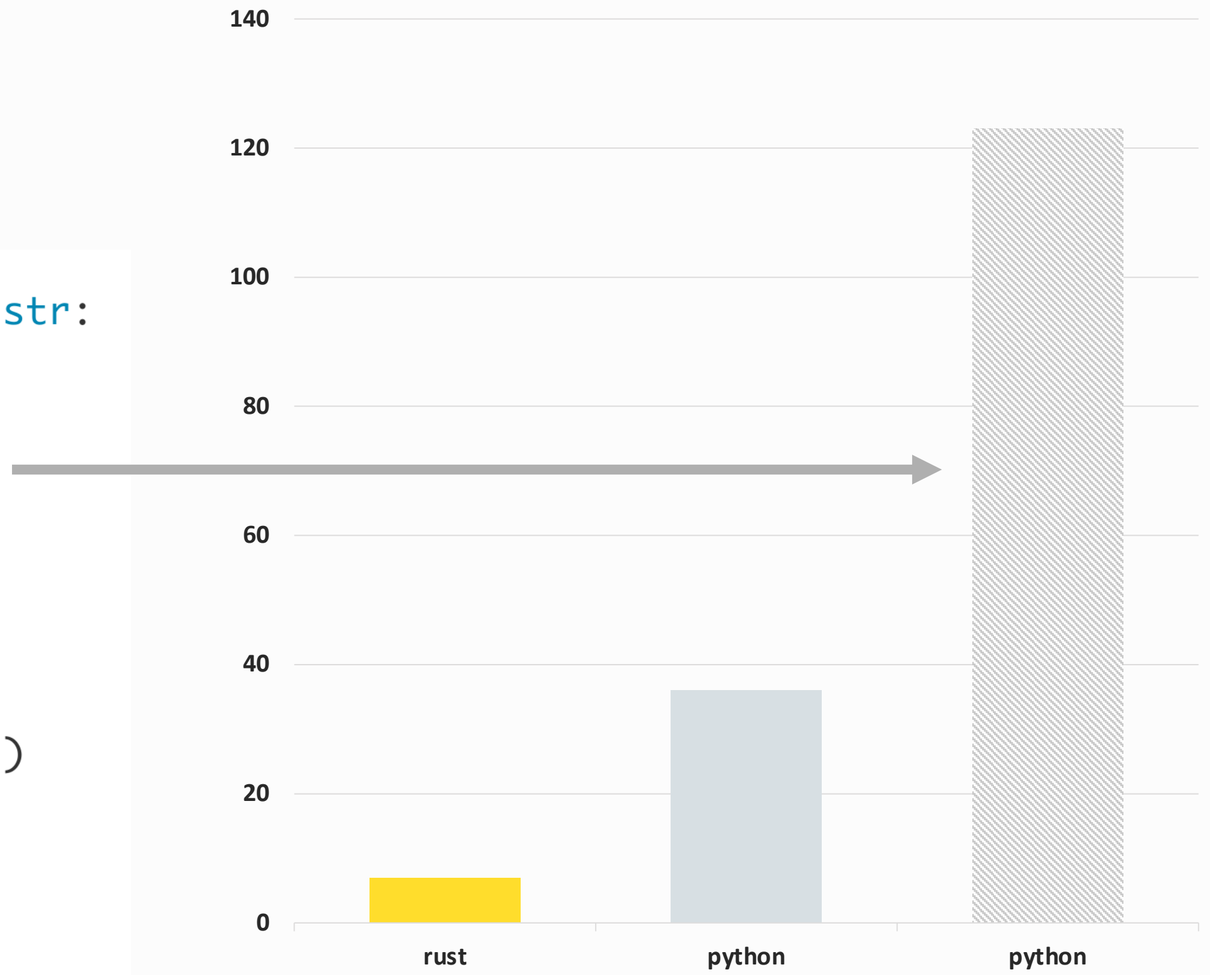
Чистый Python

```python
def pure_py_snake_to_camel(string: str) -> str:
    result = list(string)
    capitalize_next = False
    index = 0
    for char in string:
        if char == "_":
            capitalize_next = True
        else:
            if capitalize_next:
                result[index] = char.upper()
                capitalize_next = False
            else:
                result[index] = char
            index += 1
    return "".join(result)
```

Benchmark

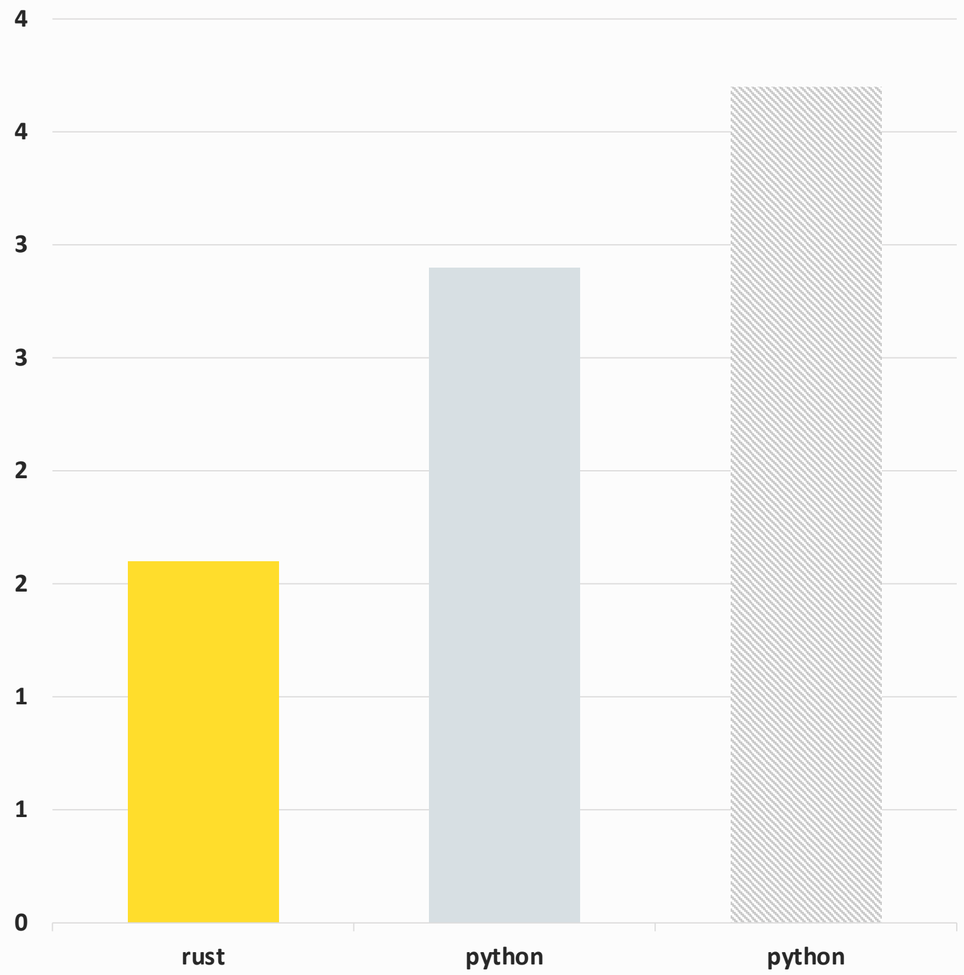
1149 байт

```python
def pure_py_snake_to_camel(string: str) -> str:
    result = list(string)
    capitalize_next = False
    index = 0
    for char in string:
        if char == "_":
            capitalize_next = True
        else:
            if capitalize_next:
                result[index] = char.upper()
                capitalize_next = False
            else:
                result[index] = char
            index += 1
    return "".join(result)
```

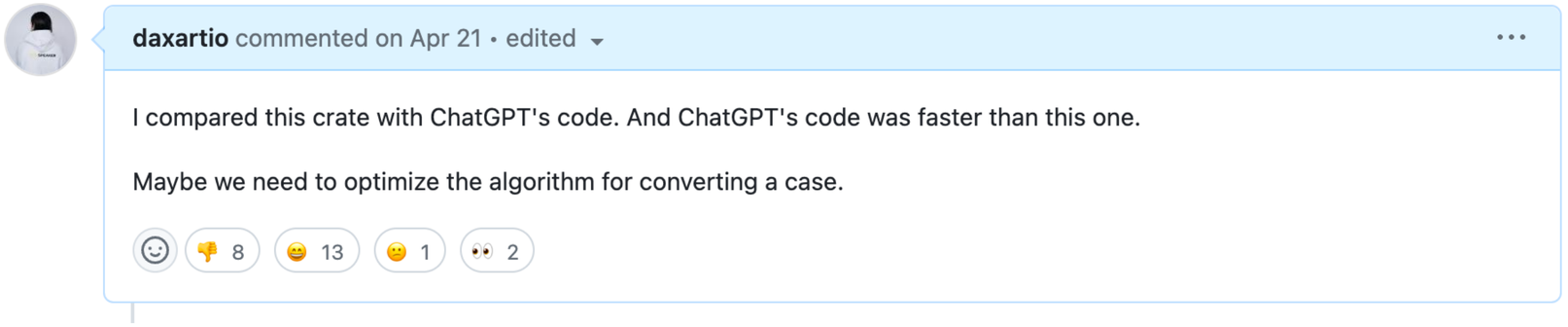


Benchmark

60 байт, до этого было 1149 байт



Использование сторонних библиотек на Rust




```
running 5 tests
test tests::it_works ... ignored
test tests::bench_to_camel1 ... bench:      22,035 ns/iter (+/- 7,051)
test tests::bench_to_camel2 ... bench:          85 ns/iter (+/- 27)
test tests::bench_to_snake1 ... bench:      19,016 ns/iter (+/- 1,645)
test tests::bench_to_snake2 ... bench:         226 ns/iter (+/- 43)

test result: ok. 0 passed; 0 failed; 1 ignored; 4 measured; 0 filtered out;
finished in 13.33s
```

https://github.com/rutrum/convert-case

Сложности

- Владение
- Заимствование
- Время жизни

Отладка

Итоги

- Посмотрели как использовать PyO3
- Нужно ли использовать Rust?
- Можем получить значительное ускорение
- Сложно поддерживать два стека

Ссылки

- https://github.com/daxartio/casers - исходный код библиотеки из примера
- https://docs.python.org/3/library/ctypes.html
- https://pyo3.rs - Главная страница PyO3
- https://github.com/rutrum/convert-case/issues/13 - issue
- https://www.opennet.ru/opennews/art.shtml?num=58249 Rust в Android
- https://youtu.be/bEP0YcOyuyE доклад про Rust

Спасибо за внимание

IT's TINKOFF

Данил Ахтаров | Архитектор

github.com/daxartio

t.me/daxtar