Integración de Rust y Python con PyO3

Hermilo

July 23, 2023

¿Qué es PyO3?

PyO3 es un proyecto desarrollado en Rust que permite la integración con Python.

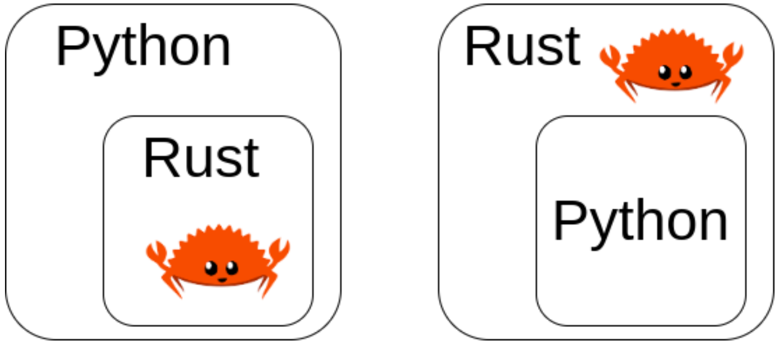


Figure 1: Tomado de Hewitt 2025

- Guía para el desarrollador : https://pyo3.rs
- Documentación : https://docs.rs/pyo3
- Github : https://github.com/pyo3/pyo3

Proyectos de soporte

- **maturin** : CLI build backend.
- **setuptools-rust** : adiciona Rust a proyectos de setuptools.
- **rust-numpy** : numpy interoperability.

La filosofía de PyO3 en el ecosistema de Python

PyO3 agrega el poder y precisión de Rust al ecosistema de Python. No es una sustitución. Es complementario.

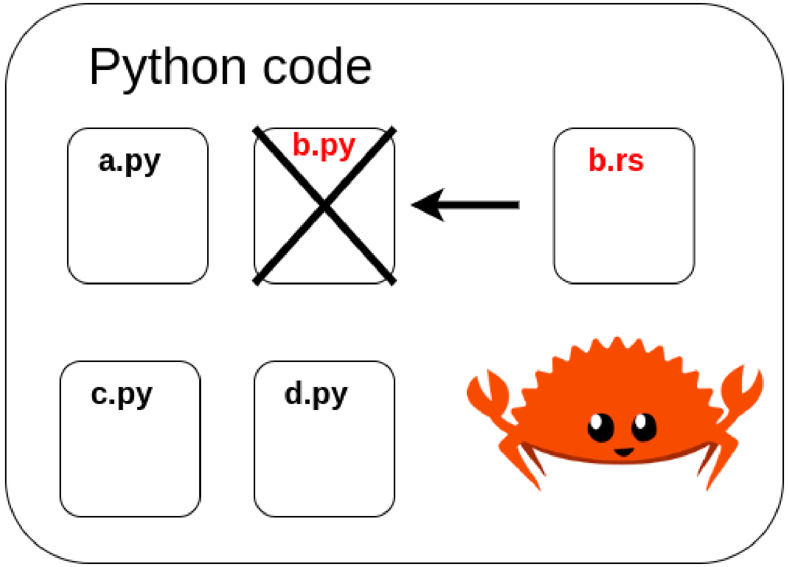


Figure 2: Tomado de Hewitt 2025

Cómo funciona PyO3

PyO3 user place procedural macro (“proc macro”) attributes on their Rust code.

These generate Rust code calling Python’s C API to define Python functions, classes and modules.

```rust
#[pyfunction]
fn my_rust_function(){...}
```

🦀 Rust

Cómo funciona PyO3

PyO3 user place procedural macro (“proc macro”) attributes on their Rust code.

These generate Rust code calling Python’s C API to define Python functions, classes and modules.

🦀 Rust
```rust
#[pyfunction]
fn my_rust_function(){...}
```

🦀 Rust
```rust
unsafe extern "C" fn __wrap(){...}

PyMethodDef{
    ml_meth: __wrap as *mut c_void,
    ...
}
```

Cómo funciona PyO3

PyO3 user place procedural macro (“proc macro”) attributes on their Rust code.

These generate Rust code calling Python’s C API to define Python functions, classes and modules.

```rust
#[pyfunction]
fn my_rust_function(){...}
```

```rust
unsafe extern "C" fn __wrap(){...}

PyMethodDef{
    ml_meth: __wrap as *mut c_void,
    ...
}
```

Tools like maturin and setuptools-rust handle the task of compiling the Rust code to a library placed where Python can consume it.

b.so

b.pyd

¿Cómo consume Python las extensiones?

- Python’s “import” statement is tipically used to load a Python file (module).

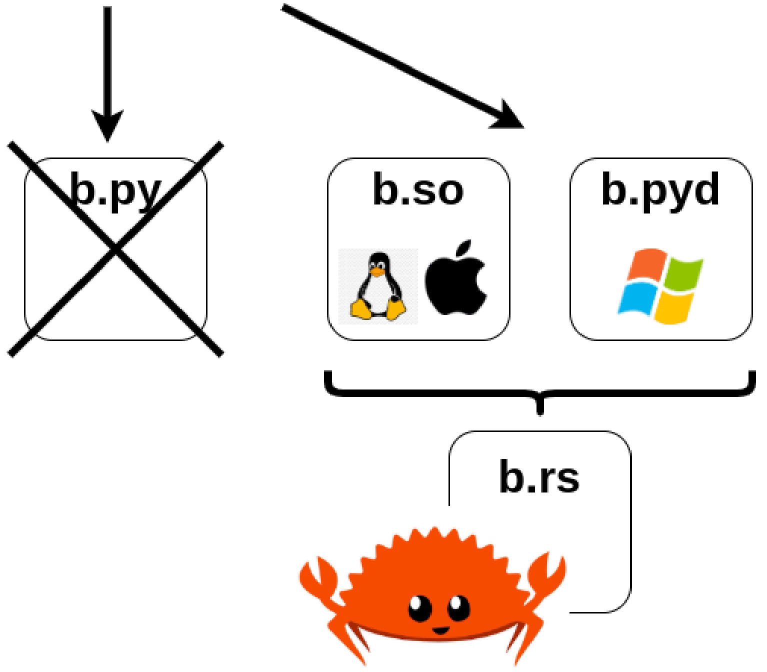
```python
import b
```

Python

¿Cómo consume Python las extensiones?

- Python’s “import” statement is tipically used to load a Python file (module).
- It can load an “extension module” from a compiled library compatible with Python’s ABI.
- This is used widely, e.g. Cython, C, C++, and Rust.

```python
import b
```



Ejemplo

Programa que cuenta ocurrencias en un texto:

Python

```python
def count_ocurrences(
        contents: str,
        needle: str,
        ) -> int:

        total = 0

        for line in contents.splitlines():
            for word in line.split():
                if word == needle:
                    total+=1

        return total
```

Ejemplo

PyO3 translation of this function is very mechanical :

Python

```python
def count_ocurrences(
        contents: str,
        needle: str,
        ) -> int:

        total = 0

        for line in contents.splitlines():
            for word in line.split():
                if word == needle:
                    total+=1

        return total
```

Rust

```rust
#[pyfunction]
fn count_ocurrences(
        contents: &str,
        needle: &str,
) -> usize {
    let mut total = 0;
    for line in contents.lines(){
        for word in line.split(" "){
            if word == needle{
                total += 1;
            }
        }
    }
    total
}
```

Ejemplo

PyO3 translation of this function is very mechanical:

🐍 Python

```python
def count_ocurrences(
        contents: str,
        needle: str,
        ) -> int:

        total = 0

        for line in contents.splitlines():
            for word in line.split():
                if word == needle:
                    total+=1

        return total
```

🦀 Rust

```rust
#[pyfunction]
fn count_ocurrences(
        contents: &str,
        needle: &str,
) -> usize {
    let mut total = 0;
    for line in contents.lines(){
        for word in line.split(" "){
            if word == needle{
                total += 1;
            }
        }
    }
    total
}
```

~ 2-4X faster (Python 3.12)

```rust
/// A Python module implemented in Rust
#[pyo3::pymodule]
mod hello_pyo3{
    use pyo3::prelude::*;
    /// Counts the number of occurrences of `needle` in `contents`.
    #[pyfunction]
    fn count_ocurrences(contents: &str, needle: &str) -> usize:
        let mut total = 0;
        for line in contents.lines(){
            for word in line.split(" "){
                if word == needle{
                    total += 1;
                }
            }
        }
        total

}
```

Rust Source

🦀 Rust

```rust
/// A Python module implemented in Rust
#[pyo3::pymodule]
mod hello_pyo3{
    use pyo3::prelude::*;
    /// Counts the number of occurrences of `needle` in
    `contents`.
    #[pyfunction]
    fn count_ocurrences(contents: &str, needle: &str) -> usize:
        let mut total = 0;
        for line in contents.lines(){
            for word in line.split(" "){
                if word == needle{
                    total += 1;
                }
            }
        }
        total

}
```

🐍 Python

```python
import hello_pyo3

contents = "a b c d"

hello_pyo3.count_ocurrences(contents, needle = "a")
```

Python API

Rust Source

What does the interpreter do when we call this function?

🐍 Python

```python
import hello_pyo3

contents = "a b c d"

hello_pyo3.count_ocurrences(contents, needle = "a")
```