

# **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.

# 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.

## Cómo funciona PyO3

PyO3 users 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.

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



# 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.

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

 Rust

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

```
1 unsafe extern "C" fn __wrap(){...}
2
3 PyMethodDef{
4     ml_meth: __wrap as *mut c_void,
5     ...
6 }
```

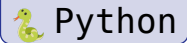
 Rust

**¿Cómo consume Python las extensiones?**

# Ejemplo

Programa que cuenta ocurrencias en un texto:


```
1  def count_ocurrences(  
2      contents: str,  
3      needle: str,  
4      ) -> int:  
5  
6      total = 0  
7  
8      for line in contents.splitlines():  
9          for word in line.split():  
10             if word == needle:  
11                 total+=1  
12  
13      return total
```




# Ejemplo

PyO3 translation of this function is very mechanical :

```
1  def count_ocurrences(  
2      contents: str,  
3      needle: str,  
4      ) -> int:  
5  
6      total = 0  
7  
8      for line in contents.splitlines():  
9          for word in line.split():  
10              if word == needle:  
11                  total+=1  
12  
13      return total
```

 Python

```
1  #[pyfunction]  
2  fn count_ocurrences(  
3      contents: &str,  
4      needle: &str,  
5  ) -> usize {  
6      let mut total = 0;  
7      for line in contents.lines(){  
8          for word in line.split(){  
9              if word == needle{  
10                  total += 1;  
11              }  
12          }  
13      }  
14      total  
15  }
```


 Rust




# Ejemplo

PyO3 translation of this function is very mechanical:

```
1  def count_ocurrences(  
2      contents: str,  
3      needle: str,  
4      ) -> int:  
5  
6      total = 0  
7  
8      for line in contents.splitlines():  
9          for word in line.split():  
10             if word == needle:  
11                 total+=1  
12  
13      return total
```

 Python

```
1  #[pyfunction]  
2  fn count_ocurrences(  
3      contents: &str,  
4      needle: &str,  
5  ) -> usize {  
6      let mut total = 0;  
7      for line in contents.lines(){  
8          for word in line.split(){  
9              if word == needle{  
10                 total += 1;  
11             }  
12         }  
13     }  
14     total  
15 }
```

 Rust

**~ 2-4X faster (Python 3.12)**

```
1  /// A Python module implemented in Rust
2  #[pyo3::pymodule]
3  mod hello_py3{
4      use pyo3::prelude::*;
5      /// Counts the number of occurrences of `needle` in
6      `contents`.
7      #[pyfunction]
8      fn count_ocurrences(contents: &str, needle: &str) -> usize{
9          let mut total = 0;
10         for line in contents.lines(){
11             for word in line.split(" "){
12                 if word == needle{
13                     total += 1;
14                 }
15             }
16         }
17         total
18     }
```



## Rust Source

```
1  /// A Python module implemented in Rust
2  #[pyo3::pymodule]
3  mod hello_py03{
4      use pyo3::prelude::*;
5      /// Counts the number of occurrences of `needle` in
6      `contents`.
7      #[pyfunction]
8      fn count_ocurrences(contents: &str, needle: &str) -> usize{
9          let mut total = 0;
10         for line in contents.lines(){
11             for word in line.split(" "){
12                 if word == needle{
13                     total += 1;
14                 }
15             }
16         }
17         total
18     }
```

 Rust

```
1  import hello_py03
2
3  contents = "a b c d"
4
5  hello_py03.count_ocurrences(contents, needle = "a")
```

 Python

## Python API

## Rust Source

# What does the interpreter do when we call this function?

```
1 import hello_py03
2
3 contents = "a b c d"
4
5 hello_py03.count_ocurrences(contents, needle = "a")
```

