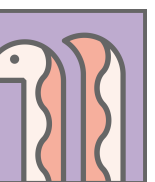


# Rust is easy

 Luca Baggi

 ML Engineer @Futura

 Organizer @Python Milano



# Rust is easy

Just trust the compiler™



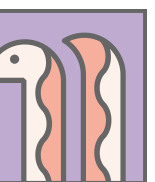
Luca Baggi



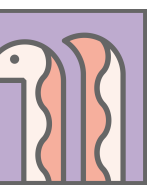
ML Engineer @Futura



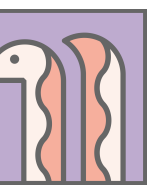
Organizer @Python Milano



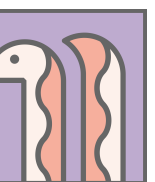
**Rust is ~~easy~~**



**Rust ~~is easy~~ has great tooling**

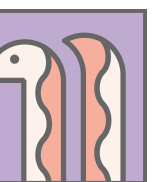


**Rust ~~is easy~~ has great docs**



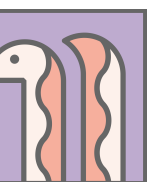
**Rust ~~is easy~~ has great docs**

That makes learning it way less frustrating



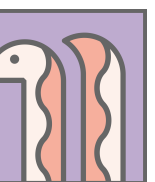
# Rust ~~is easy~~ has great docs

So much so, you don't have to read them



# Rust ~~is easy~~ has great docs

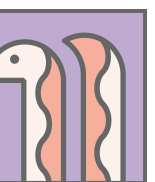
Like I did.





# Why should I learn Rust?

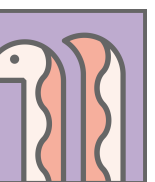
I am a Python user



# Why should I learn Rust?

I am a Python user

Python is “easy” (read: accessible). However, best practices, clean code, design patterns, etc... are not as straightforward.

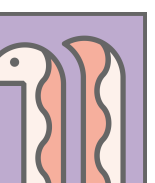


# Why should I learn Rust?

I am a Python user

Python is “easy” (read: accessible). However, best practices, clean code, design patterns, etc... are not as straightforward.

In other words, **anti-patterns and poor design choices** are easy to pick up, and **harder to get rid of**: if the code *just works*<sup>™</sup>, then there are less incentives to refactor it.



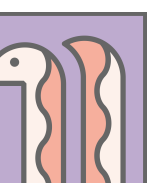
# Why should I learn Rust?

I am a Python user

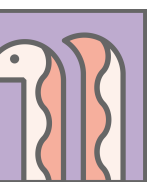
Python is “easy” (read: accessible). However, best practices, clean code, design patterns, etc... are not as straightforward.

In other words, **anti-patterns and poor design choices** are easy to pick up, and **harder to get rid of**: if the code *just works*<sup>™</sup>, then there are less incentives to refactor it.

Learning a bit of Rust can help us **write better Python code** from the start.



# Let's code!

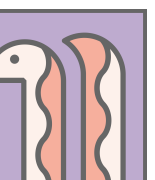


# A concrete example

```
from __future__ import annotations
```

```
from datetime import datetime
```

```
class User:  
    privilege: str  
    banned_at: datetime | None
```



# A concrete example

```
from __future__ import annotations
```

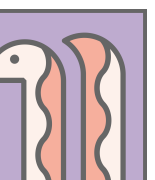
```
from datetime import datetime
```

```
from typing import Literal
```

```
class User:
```

```
    privilege: Literal["normal", "admin", "banned"]
```

```
    banned_at: datetime | None
```



# A concrete example

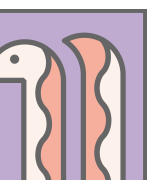
```
from __future__ import annotations
```

```
from datetime import datetime
```

```
from typing import Literal
```

```
class Banned:  
    banned_at: datetime
```

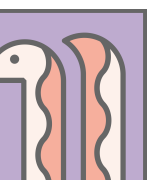
```
class User:  
    privilege: Literal["normal", "admin"] | Banned
```



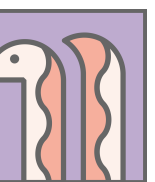


# A concrete example

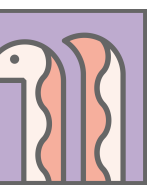
```
def process_user(user: User):  
    match user.privilege:  
        case Banned(banned_at=banned_at):  
            print(f"User banned at {banned_at}.")  
        case "admin":  
            print("User is an admin.")
```



# Questions?

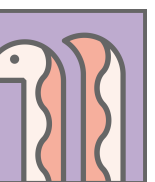


# Thank you!



# Thank you!

🐼 Tomorrow at 14:30, Room Pizza, I will present **Polars:**  
**is the great dataframe showdown finally over?**



# Thank you!

🙏 Feedback is very welcome! You can find me at [lucabaggi@duck.com](mailto:lucabaggi@duck.com), or feel free to connect on LinkedIn!

