import rust

https://github.com/timClicks/talks

Agenda

Confusing Concepts in Rust Made Easy

- Borrow checker
- Lifetimes
- Error handling
    - Option
    - Result

Using Rust in a Python program

Motivations

Introductions

```
>>> help(memory_safety)
```

TIMCLICKS.DEV

WHAT IS MEMORY SAFETY?

When you always get back exactly what you put into memory, then the program is "memory safe".

TIMCLICKS.DEV

```
>>> help(borrow_checker)
```

TIMCLICKS.DEV

```
a = "Hello, Wellington!"
del a
b = a
```

```
a = "Hello, Wellington!"
del a
b = a
```

```
Traceback (most recent call last):
  File "hello_wlg.py", line 3, in <module>
    b = a
NameError: name 'a' is not defined
```

```
import threading

analysis = "Wellington's coffee is better."

def coffee_report():
    print(analysis)

thread_1 = threading.Thread(target=coffee_report)
thread_1.start()

thread_1.join()
```

TIMCLICKS.DEV

```
$ python3 coffee_report.py
```

```
Wellington's coffee is better.
```

TIMCLICKS.DEV

```
import threading; import random; import time

analysis = "Wellington's coffee is better."

def coffee_report():
    time.sleep(random.random())
    print(analysis)

thread_1 = threading.Thread(target=coffee_report)
thread_1.start()
time.sleep(random.random())
del a
thread_1.join()
```

TIMCLICKS.DEV

```
$ python3 coffee_report_non_deterministic.py
```

Impossible to know
which outcome will occur.

TIMCLICKS.DEV

WHAT IS THE BORROW CHECKER?

The borrow checked is a component of the Rust compiler that ensures that all accesses to variables will succeed.

In Rust, AttributeError never occurs.

TIMCLICKS.DEV

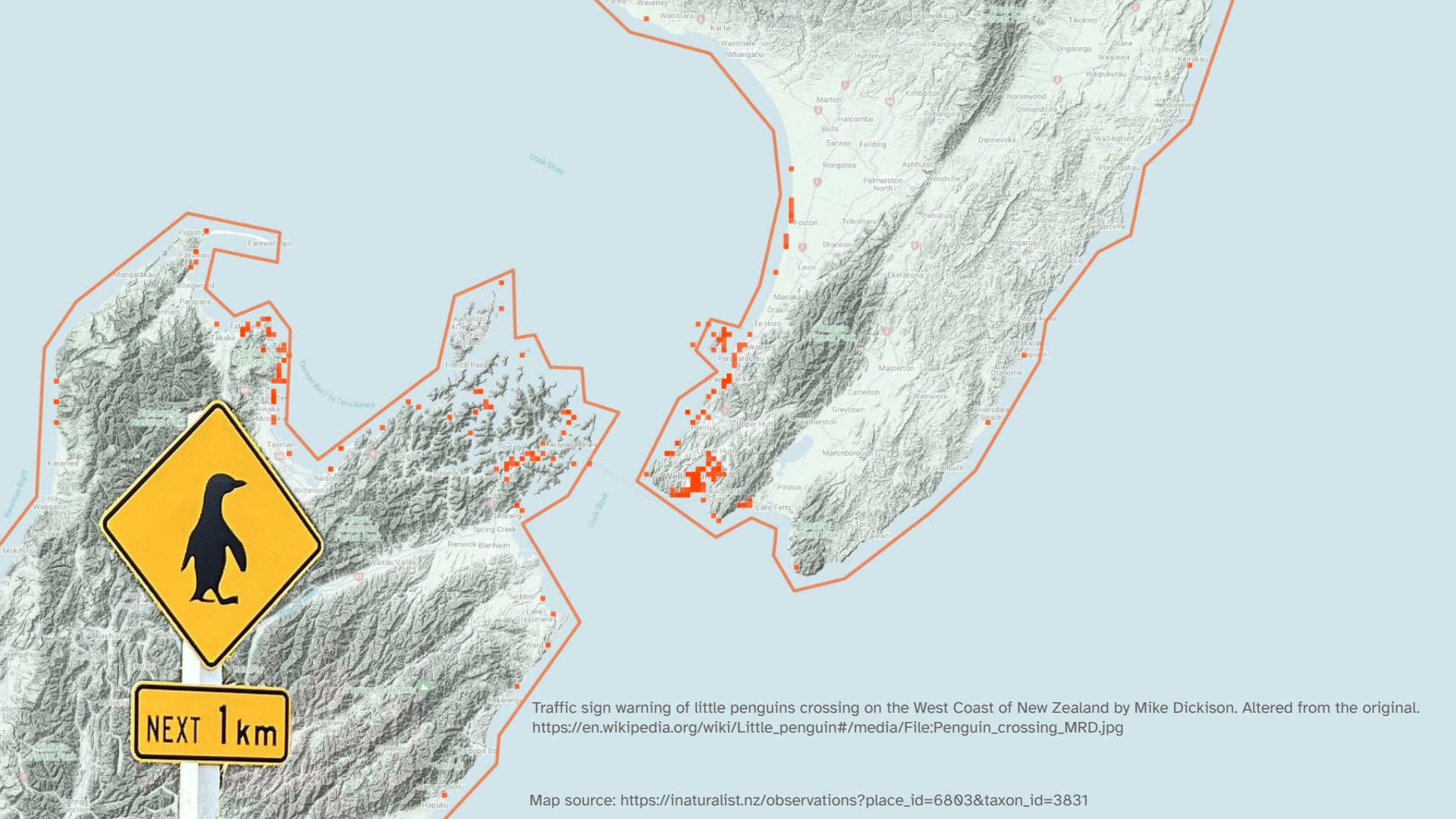
```
>>> help(lifetimes)
```

TIMCLICKS.DEV

WHAT IS A "LIFETIME"?

A lifetime is the span of logical time where accessing a variable is valid.

TIMCLICKS.DEV



Traffic sign warning of little penguins crossing on the West Coast of New Zealand by Mike Dickison. Altered from the original.
https://en.wikipedia.org/wiki/Little_penguin#/media/File:Penguin_crossing_MRD.jpg

Map source: https://inaturalist.nz/observations?place_id=6803&taxon_id=3831

```rust
#[derive(Debug)]
enum Penguin {
    Hoiho, Kororā, Tawaki,
}

fn main() {
    let sighting = Penguin::Kororā;
    println!("{sighting:?}");
}
```

Kororā

https://play.rust-lang.org/?edition=2021&gist=808a65c5cad4a4e520b1ba8c85289659

Learn more about New Zealand's penguin species here:
https://dxcprod.doc.govt.nz/nature/native-animals/birds/birds-a-z/penguins/

TIMCLICKS.DEV

```rust
#[derive(Debug)]
enum Penguin {
    Hoiho, Kororā, Tawaki,
}

fn main() {
    let sighting = Penguin::Kororā;

    println!("{sighting:?}");
}
```

https://play.rust-lang.org/?edition=2021&gist=967efe0984d7e23fac9b4eeb152edec1

Learn more about New Zealand's penguin species here:
https://dxcprod.doc.govt.nz/nature/native-animals/birds/birds-a-z/penguins/

TIMCLICKS.DEV

```rust
#[derive(Debug)]
enum Penguin {
    Hoiho, Kororā, Tawaki,
}

fn main() {
    let sighting = Penguin::Kororā;
    let another_sighting = sighting;
    println!("{sighting:?}");
}
```

error[E0382]: borrow of moved value: `sighting`

https://play.rust-lang.org/?edition=2021&gist=967efe0984d7e23fac9b4eeb152edec1

Learn more about New Zealand's penguin species here:
https://dxcprod.doc.govt.nz/nature/native-animals/birds/birds-a-z/penguins/

```
>>> help(rust_error_handling)
```

TIMCLICKS.DEV

HOW DOES ERROR HANDLING WORK IN RUST?

Errors are returned from functions as values.

There are no exceptions in Rust.

Errors that cannot be handled crash the program.

TIMCLICKS.DEV

Option<T>

TIMCLICKS.DEV

PYTHON

```
class Collection:
   items = []

   def last(self) → str | None:
       self.items[-1]
```

RUST

```
struct Collection<T> {
    items: Vec<T>
}

impl<T> Collection<T> {
    fn last(&self) → Option<&T> {
        self.items.get(self.items.len())
    }
}
```

TIMCLICKS.DEV

Result<T, E>

TIMCLICKS.DEV

Note to self: open up the playground and explain Result there

An extension

LET'S DO IT LIVE

https://github.com/PyO3/pyo3

About Me

TIMCLICKS.DEV

Q&A