TODO

Multi-plaform binding with Python!

🎤 **Speaker**: *Gianluca Aguzzi*, University of Bologna
📧 Email: gianluca.aguzzi@unibo.it
🌐 Personal site: https://cric96.github.io/
PDF slides @ https://cric96.github.io/phd-course-python-binding/index.pdf

Outline

- How to handle (conceptually) python - native interaction
- Main alternative in the current panorama
- A guided example with raylib

Create Binding from Native

Agenda 💭

- What you want to **expose**?
  - Low level or pythonic?
- How to **manage** the different types?
  - Marshalling?
- How to handle the **memory**?
  - GC vs Manual

What you want to expose?

- It is important to define what you want to expose to the Python side.
- Typically, native code is note pythonic, so you need to create a pythonic interface.
- **Flow**: Native ➡️ Direct Python Binding ➡️ Pythonic Interface

How to manage the different types?

- **Marshalling**: the process of transforming the data to be passed between the two platform.
- Two mindset:
  - C ➡️ Focused on performance,
  - Python ➡️ Focused on simplicity
- Examples:
  - Integers: C has int, short, long, long long, Python has int
  - Floats: C has float, double, Python has double

How to mannage memory?

Main alternatives

In python, there are several ways to create a binding with native code, from completaly manual to automatic.

- **ctypes**: Python standard library, it is a foreign function interface (FFI) for Python.
- **cffi**: A foreign function interface for Python calling C code.
- **Cython**: A language that makes writing C extensions for Python as easy as Python itself.
- **Swig**: A code generator for create several bidings in different languages (among them, Python).