# Rust programming

Module 1: Course introduction

# Why learn Rust?

# How to choose a language

What characteristics do you want?

1. Efficiency
2. Safety
3. Elegance
4. Practical relevance

Most languages tick two of these boxes, if you are lucky you get three.

# What Rust promises

1. Pedal to the metal
2. Comes with a warranty
3. Beautiful code
4. Rust is practical

# Pedal to the metal

- Compiled language, not interpreted
- State-of-the-art code generation using LLVM
- No garbage collector getting in the way of execution
- Usable in embedded devices, operating systems and demanding websites

# Rust comes with a warranty

- Strong type system helps prevent silly bugs
- Explicit errors instead of exceptions
- Type system tracks lifetime of objects
  - No more *"null pointer exception"*
- Programs don't trash your system accidentally
  - Warranty *can* be voided ( `unsafe` )

*"If it compiles, it is more often correct."*

# Rust code is elegant

- Data types can capture many problem domains
- Orthogonal, expression-oriented language
- Combine declarative and imperative paradigms
- Concise syntax instead of boilerplate
- Toolchain that suggests improvements to your code

# Rust is practical

- Can interface with legacy C code
- Supported on many platforms
- Active user base maintains a healthy ecosystem
- Adoption by Microsoft, Amazon, Google, …

# Why should *you* learn Rust?

- Learning a new language teaches you new tricks
  - You will also write better C/C++ code
- Rust is a young, but quickly growing platform
  - You can help shape its future
  - Demand for Rust programmers will increase!