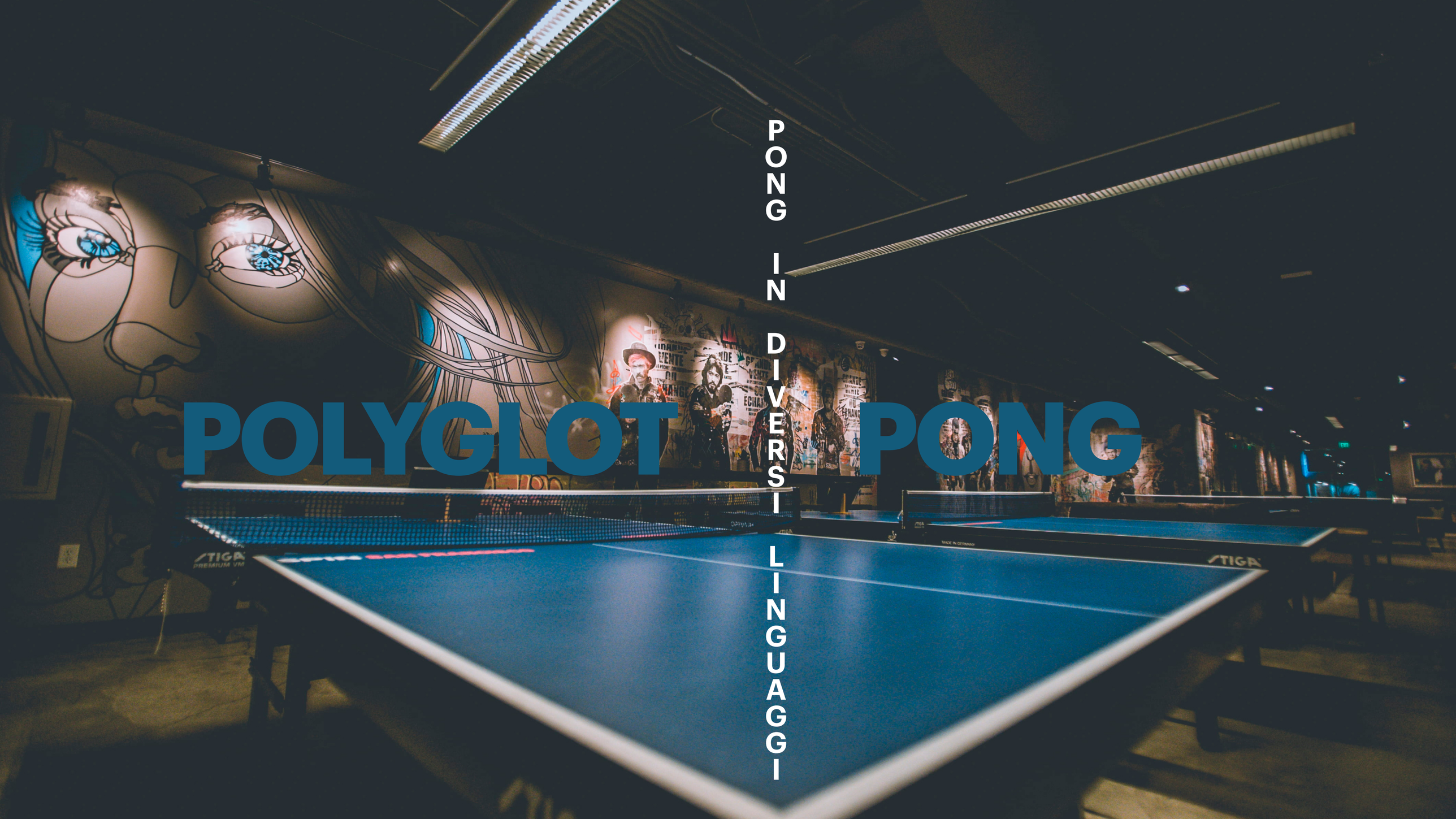

POLYGLOT PONG
PONG IN DIVERSI LINGUAGGI

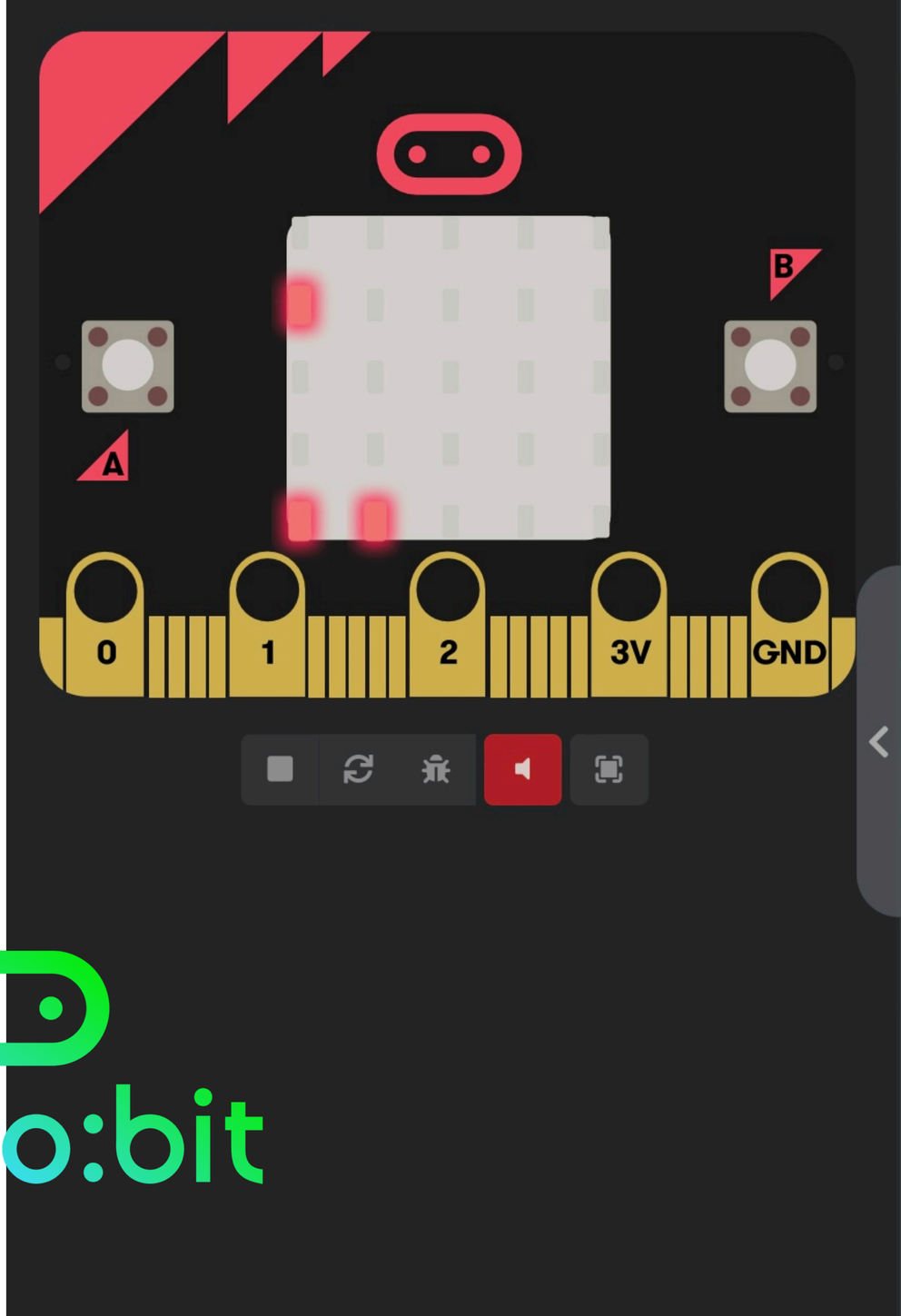
POLIGLOTTA?

INTRODUZIONE ALLA PROGRAMMAZIONE, TRA DIVERSI LINGUAGGI

- Demo Pong single-player
- Python vs JS
- Demo Pong multi-player Rust
- Rust vs C++
- 🏠

PEZZANO ENRICO, MANNAI MATTEO, ILARIA BRUZZONE

DEMO SINGLE PLAYER



micro:bit

Score: 3

Score: 3

pythong.py — terminal mult

pythong.py U | pong.js U

Users > enrico > Library > CloudStorage > OneDrive-unige.it > icdd > polyglot-pong > singleplayer > pythong.py

```python
bar_x = 0
score = 0
interval = 0
interval_step = 0
ball_x = 0
ball_y = 0
ball_dx = 0
ball_dy = 0
in_game = False

def on_button_pressed_a():
    global bar_x
    if bar_x > 0:
        led.unplot(bar_x + 1, 4)
        bar_x = bar_x - 1
        led.plot(bar_x, 4)
input.on_button_pressed(Button.A, on_button_pressed_a)

def on_button_pressed_b():
    global bar_x
    if bar_x < 3:
        led.unplot(bar_x, 4)
        bar_x = bar_x + 1
        led.plot(bar_x + 1, 4)
input.on_button_pressed(Button.B, on_button_pressed_b)

def on_forever():
    global score, interval, interval_step, ball_x, ball_y, ball_dx, ball_dy, bar_x, in_game
    score = 0
    interval = 500
    interval_step = 10
    ball_x = 3
    ball_y = 4
    ball_dx = -1
    ball_dy = -1
    bar_x = 0
    basic.show_string("GO")
    led.plot(ball_x, ball_y)
    led.plot(bar_x, 4)
    led.plot(bar_x + 1, 4)
    in_game = True
    while in_game:
        if ball_x + ball_dx > 4:
```

main* ⊗ 0 ⚠ 0 Live Share Git Graph

pong.js — terminal multipl

pythong.py U | pong.js U

Users > enrico > Library > CloudStorage > OneDrive-unige.it > icdd > polyglot-pong > singleplayer > pong.js > ...

```javascript
let bar_x = 0
let score = 0
let interval = 0
let interval_step = 0
let ball_x = 0
let ball_y = 0
let ball_dx = 0
let ball_dy = 0
let in_game = false
//
input.onButtonPressed(Button.A, function () {
    if (bar_x > 0) {
        led.unplot(bar_x + 1, 4)
        bar_x = bar_x - 1
        led.plot(bar_x, 4)
    }
})
//
input.onButtonPressed(Button.B, function () {
    if (bar_x < 3) {
        led.unplot(bar_x, 4)
        bar_x = bar_x + 1
        led.plot(bar_x + 1, 4)
    }
})
basic.forever(function () {
    score = 0
    interval = 500
    interval_step = 10
    ball_x = 3
    ball_y = 4
    ball_dx = -1
    ball_dy = -1
    bar_x = 0
    basic.showString("GO")
    led.plot(ball_x, ball_y)
    led.plot(bar_x, 4)
    led.plot(bar_x + 1, 4)
    in_game = true
    while (in_game) {
        if (ball_x + ball_dx > 4) {
            ball_dx = ball_dx * -1
        } else if (ball_x + ball_dx < 0) {
```

main* ⊗ 0 ⚠ 0 Live Share Git Graph

Score: 3

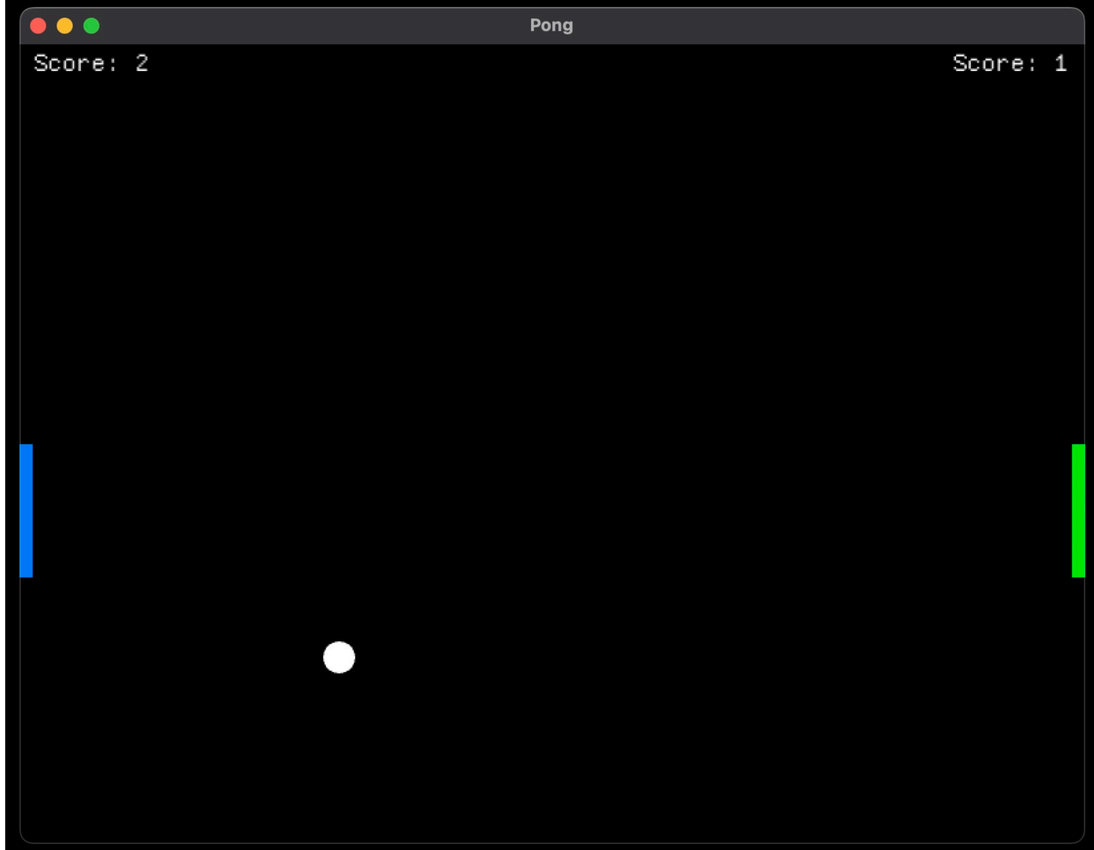
Score: 3

- Interpretato (Python VM)
- Sviluppo back-end
- No uso dei “;”, ma...
- ...indentazione per definire i code-block
- nomeVar = 7
- none
- Memoria per le variabili virtualmente infinta

- Interpretato (nel browser)
- Sviluppo back-end E front-end
- Sì uso dei “;” e ...
- ...parentesi {} per definire i code-block
- var nomeVar = 7;
- null
- Variabili allocate automaticamente

(De)Allocazione memoria automatica

DEMO MULTI PLAYER



Score: 3

Score: 3

score.rs — ter

score.cpp U | score.rs

pong > src > score.rs

```rust
use macroquad::prelude::*;

pub struct Score {
    pub player_1: u32,
    pub player_2: u32,
}

impl Score {

    pub fn new() -> Score {
        Score {
            player_1: 0,
            player_2: 0,
        }
    }

    pub fn increment_score(&mut self, player: u32) {
        if player == 1 {
            self.player_1 += 1;
        } else {
            self.player_2 += 1;
        }
    }

    pub fn draw(&self) {
        draw_text(&format!("Score: {}", self.player_1), 10.0, 20.0, 25.0, WHITE);
        draw_text(&format!("Score: {}", self.player_2), 700.0, 20.0, 25.0, WHITE);
    }

}
```

main | 0 ⚠ 0 | Live Share | Git Graph

score.cpp — terminal multi

score.cpp U | score.rs

Users > enrico > Library > CloudStorage > OneDrive-unige.it > icdd > polyglot-pong > singleplayer > score.cpp

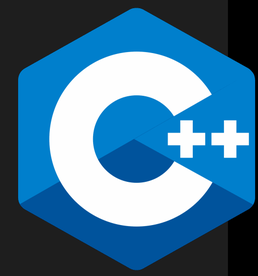
```cpp
#include <iostream>

struct Score {
  unsigned int player_1;
  unsigned int player_2;

  Score() : player_1(0), player_2(0) {}

  void increment_score(unsigned int player) {
    if (player == 1) {
      player_1++;
    } else {
      player_2++;
    }
  }

  void draw() {
    std::cout << "Score: " << player_1 << std::endl;
    std::cout << "Score: " << player_2 << std::endl;
  }
};
```



main* | 0 ⚠ 0 | Live Share | Git Graph

Score: 3

Score: 3

- È più sicuro a livello di memoria (safe mode)
- Ownership (Proprietà)
- Strumenti sta per creare pacchetti (Cargo)
- Compila leggermente più lento
- Type checking statico o dinamico
- Espressioni di tipo esplicite o implicite
- let i: i8 = 1;

- Meno sicuro, gestione manuale
- Dangling Pointers ☠️
- Nessun strumento standard
- Compila leggermente più veloce
- Type checking solo statico
- Espressioni di tipo solo esplicite
- int i = 1;

Entrambi compilano in linguaggio macchina

CODING IS TODAY'S LANGUAGE OF CREATIVITY. ALL OUR CHILDREN DESERVE A CHANCE TO BECOME CREATORS INSTEAD OF CONSUMERS OF COMPUTER SCIENCE.

MARIA KLAWE, PRESIDENT OF HARVEY MUDD COLLEGE

TAKE-HOME MESSAGE 

TROPPI LINGUAGGI?

- Ognuno ha le sue differenze (motivazioni storiche, memoria e garbage-collector)
- A seconda del contesto si sceglie il più adatto (tempi di compilazione, framework, etc)
- Difficoltà intrinseca
- Portabilità, cross-platform
- Didattica