ARE YOU A CARBON LIFE FORM?

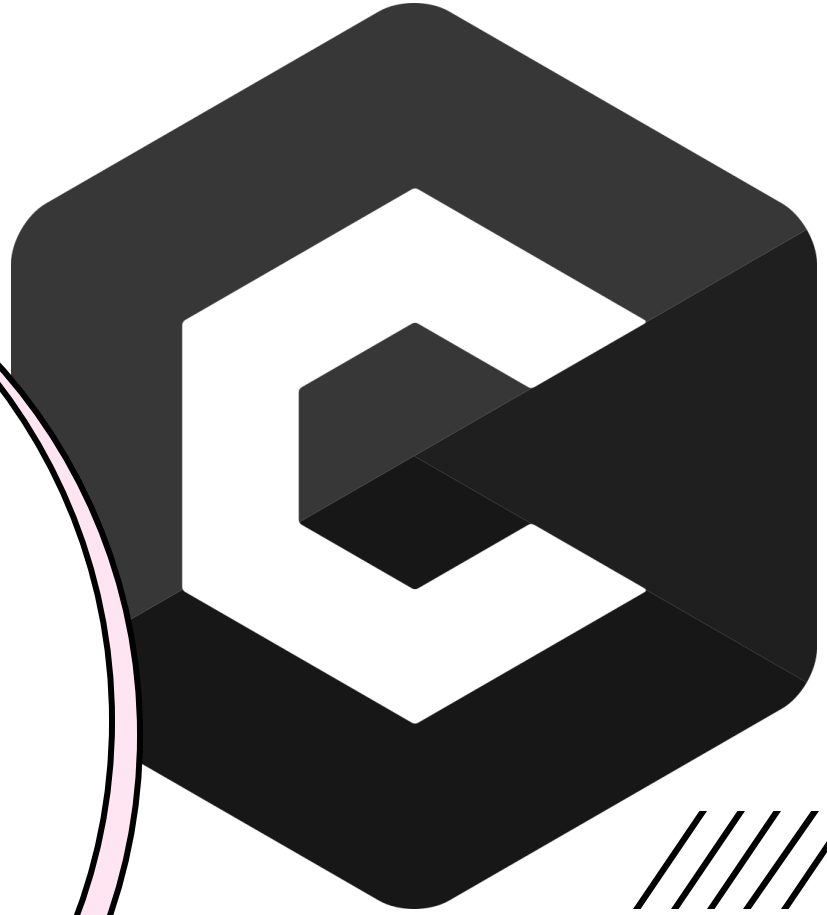
DO YOU HAVE THE MOJO TO LEARN ABOUT THESE LANGUAGES?

Made Swiftly By
Daniel and Kevin

Introduction

- The rise of new programming languages
  - Carbon
  - Mojo
  - Swift
- Why do we need new languages?
  - Evolving needs
  - Tech debt with old languages
  - Quick updates

CARBON



Purpose of Carbon

- Carbon is designed to support:
  - Performance critical software
  - Code that is easy to read, understand, and write
  - Interoperability with and migration from existing C++ code
- Why was Carbon created?
  - C++ is struggling to meet developers' needs
  - Carbon is fundamentally a successor language approach, rather than an attempt to incrementally evolve C++

Features/Paradigms

- Modern Generics System
  - Has checked definitions, while still supporting templates for seamless C++ interop
  - Generic definitions are fully type-checked
- Memory Safety
  - Safety, and especially memory safety, remains a key challenge for C++
  - Tracks uninitialized states better
  - Has a default debug build-mode that is both cheaper and more comprehensive than existing C++ build modes

Competition

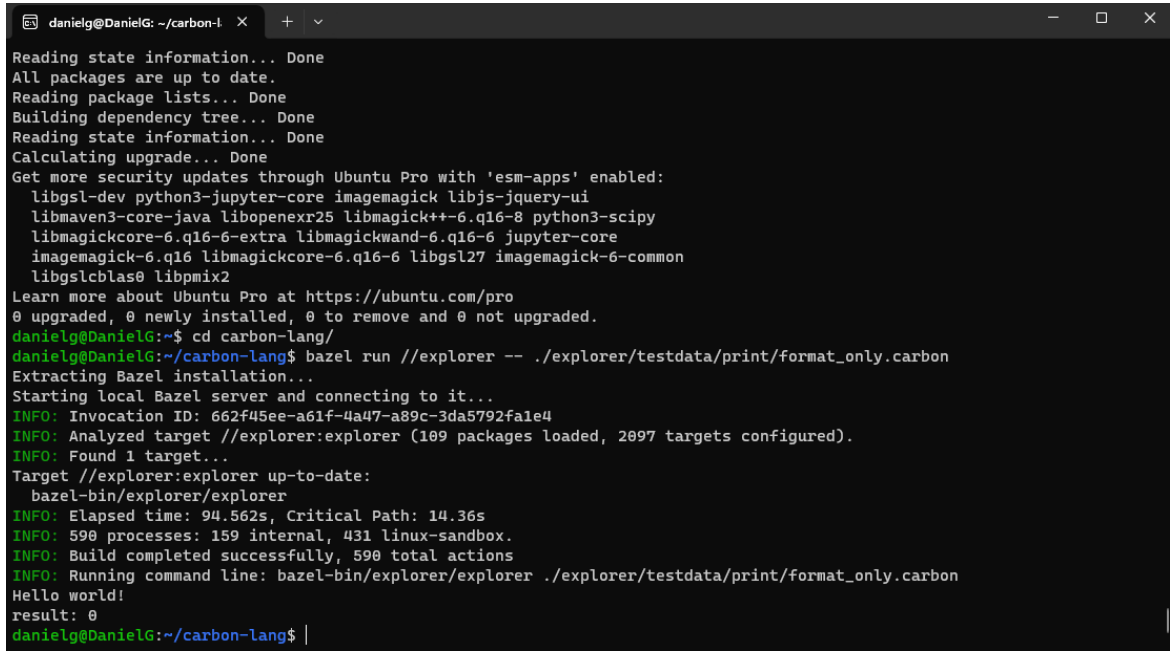
- Directly competes with C++
- Carbon has negligible popularity rating
- C++ has a 9.96% popularity rating
  - At rank #3

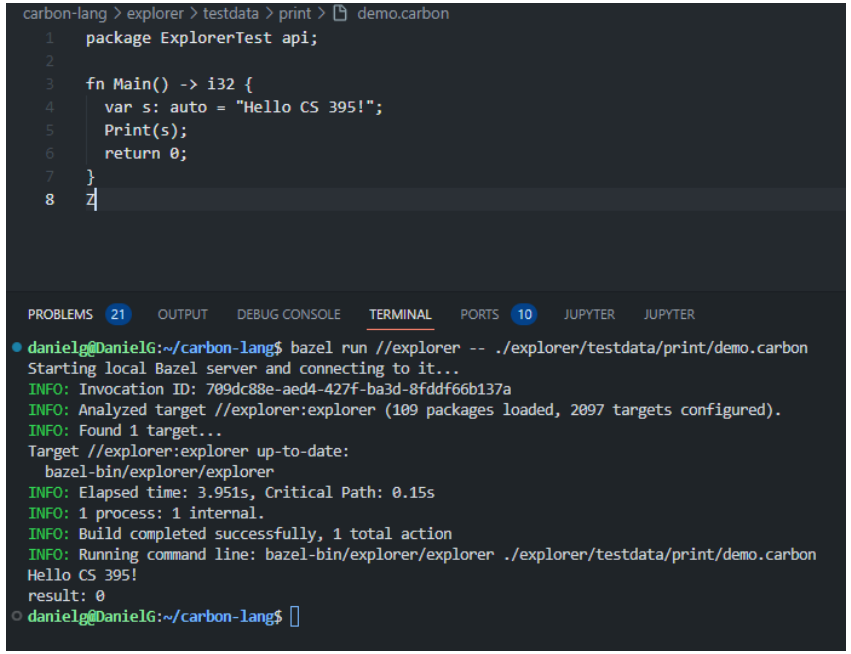
Carbon vs C++

```
// Carbon:
package Geometry api;
import Math;

class Circle {
  var r: f32;
};

fn PrintTotalArea(circles: Slice(Circle)) {
  var area: f32 = 0;
  for (c: Circle in circles) {
    area += Math.Pi * c.r * c.r;
  }
  Print("Total area: {0}", area);
}

fn Main() -> i32 {
  // A dynamically sized array, like `std::vector`.
  var circles: Array(Circle) = ({.r = 1.0},
                                {.r = 2.0});
  // Implicitly converts `Array` to `Slice`.
  PrintTotalArea(circles);
  return 0;
}
```

```
// C++:
#include <math.h>
#include <iostream>
#include <span>
#include <vector>

struct Circle {
  float r;
};

void PrintTotalArea(std::span<Circle> circles) {
  float area = 0;
  for (const Circle& c : circles) {
    area += M_PI * c.r * c.r;
  }
  std::cout << "Total area: " <<  area << "\n";
}

auto main(int argc, char** argv) -> int {
  std::vector<Circle> circles = {{1.0}, {2.0}};
  // Implicitly converts `vector` to `span`.
  PrintTotalArea(circles);
  return 0;
}
```

Building and Compiling Carbon





MOJO

Purpose of Mojo

- Mojo:
  - Created in 2023 by Chris Lattner: The mind behind the Swift programming language.
  - It was heavily influenced by Python: Combines Python's syntax and ecosystem, including its extensive modules.
  - Unifying Artificial Intelligence (AI) and Machine Learning (ML) Development: Aims to streamline the development and infrastructure process while also increasing preformace in artificial intelligence and machine learning applications.
  - Accessible to Researchers and Developers: Simplifies the transition from research to production, making advanced AI more attainable.

Features/Paradigms

- Mojo is a statically typed language, decreasing errors and debug times.
- Unlike python which is dynamically type.
- Apps can be natively compiled into a small, standalone, binary using 'mojo build'.
- Supports Just-in-time (JIT) compilation.
- Concurrency support: Improves code that runs in parallel.
- Automatic memory management

Competition

- Completes with Python.
  - It is 35000x faster than python
  - Mojo is compiled into machine code using Mojo-specific features utilizing the LLVM toolchain, whereas Python relies on runtime compilation
  - Mojo uses Multi-level Intermediate Representation (MLIR) allow it to run on a multitude of GPUs, TPUs, and the vector units.
  - Allowing seamless scalability
- Mojo is still new, being released in May of 2023

Mojo vs Python

CS395 > hello_world.mojo > ...

```mojo
struct HelloWorld:
    @staticmethod
    fn say_hello():
        print("Hello, World!")

fn main():
    let hello = HelloWorld

    hello.say_hello()

```

CS395 > hello_world.py

```python
class HelloWorld:
    def say_hello(self):
        print("Hello, World!")

hello = HelloWorld()

hello.say_hello()

```

SWIFT

Purpose of Swift

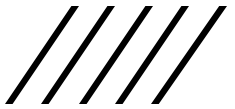
- Initially was developed only for native iOS development
- Apple created Swift as a replacement for all C-based languages
- Swift provides a more modern, safe, and efficient alternative to Objective-C

Features/Paradigms

- Swift supports inferred types to make code cleaner and less error prone
- Automatically managed memory
- Some additional features:
  - Tuples and multiple return values
  - Generics
  - Functional Programming Patterns
  - Advanced flow control with *do*, *guard*, *defer*, and *repeat*
- Safety:
  - Designed from the outset to be safer than C-based languages

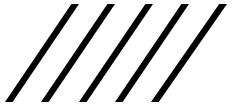
Competition

- Popularity Ranking:
  - Current customers: 19,341 companies that use Swift
  - 0.89% popularity rating
    - Ranked #16
  - Objective-C is rank #31 at 0.46%



Swift vs Objective-C

- Swift is statically typed, and Objective-C is dynamically types
- Swift has both structs and classes while Objective-C has only classes
- Swift is designed mainly for building apps for iOS, Mac, Apple TV, and Apple Watch
- Objective-C was created in the 1980s and lacked modern features



Swift Code

```swift
//
//  main.swift
//  test
//
//  Created by Daniel Garza on 1/22/24.
//

import Foundation

print("Hello CS 395!")

var myString: String = "Hello, Swift!"

var myInteger = 42

print(myString, myInteger)

```

The Conclusion

- Key Takeaways:
  - Carbon represents a departure from the incremental evolution of C++ and, instead, takes a fundamentally different approach as a successor language
  - Mojo hopes to be a better alternative to Python, while offering many benefits to Artificial Intelligence (AI) and Machine Learning (ML) workflows, that include speed increases and multi-systems support
  - Swift serves as the primary language for developing apps for Apple devices

References

- https://github.com/carbon-language/carbon-lang
- https://www.tiobe.com/tiobe-index/
- https://www.coursera.org/articles/programming-in-swift
- https://6sense.com/tech/languages/swift-market-share
- https://docs.modular.com/mojo/
- https://medium.com/coinmonks/introduction-to-mojo-the-programming-language-for-ai-which-35000x-faster-than-python-4fcd3a9cecab

QUESTIONS?