Introduction to Zig Programming Language

Raman Deep Singh

USES OF THE ZIG PROGRAMMING LANGUAGE

If you're asking “What can I actually build with Zig?”, here are practical examples.

1. System software

- Operating-system components
- System utilities
- File-system tools
- Memory managers
- Device drivers
- Process managers

2. Fast applications

- Image/video processing
- Compression tools
- Encryption libraries
- Data-processing programs
- High-performance parsers

3. Web servers and networking

- HTTP servers
- REST APIs
- TCP/UDP servers
- WebSocket servers
- Network proxies
- Network utilities

Typical architecture:

Client

↓

Zig HTTP Server

↓

Database

↓

JSON Response

4. Game development

- Game engines
- 2D games
- 3D games
- Physics systems
- Graphics programming
- Game networking

5. Embedded/IoT

- IoT devices
- Robotics
- Sensors
- Microcontrollers
- Hardware control

Typical architecture:

Zig program

↓

Microcontroller

↓

Temperature sensor

↓

Motor / LED / Display

6. Command-line tools

- File converters
- Log analyzers
- Backup utilities
- Password generators
- JSON formatters
- System monitors

Zig produces native executables, so compiled tools can be distributed without requiring a runtime like Python.

7. C/C++ libraries

Zig can interact with existing C code, making it useful for integrating existing C libraries without rewriting them.

8. Python + Zig

Python can be used for the high-level application while Zig handles performance-critical code.

Typical architecture:

Python application

↓

Zig module

↓

Performance-critical code

9. AI/ML infrastructure

Zig is not normally the first choice for training neural networks, where Python + PyTorch/TensorFlow are more common. However, Zig can be useful for:

- AI inference engines
- Tokenizers
- Tensor operations
- Quantization

- Data preprocessing
- CPU-optimized algorithms
- Lightweight AI runtimes

Typical architecture:

Python

↓

AI model / application

↓

Zig

↓

Fast inference / preprocessing

10. Build systems

Zig has its own build system for managing compilation and dependencies in complex projects.

WHAT SHOULD YOU USE ZIG FOR?

Use Python when you want to build something quickly.

Use Zig when you need:

- Speed
- Low-level control
- Small native executables
- Predictable memory usage

Examples of Zig Programs

Program to print hello world

```
const std = @import("std");

pub fn main() void {
    std.debug.print("Hello, World!\n", .{});
```

```
}
```

Program to display a number

```
const std = @import("std");

pub fn main() !void {
    var age: i64 = 25;
    age = 26;
    std.debug.print("Age is {d}\n", .{age});
}
```

Program for if statement

```
const std = @import("std");

pub fn main() !void {
    const age: i32 = 20;

    if (age >= 18) {
        std.debug.print("Adult\n", .{});
    } else {
        std.debug.print("Minor\n", .{});
    }
}
```

Program for a while loop

```
const std = @import("std");

pub fn main() !void {
    var i: i32 = 0;

    while (i < 5) : (i += 1) {
        std.debug.print("{}\n", .{i});
```

```
    }
}
```

Program to create a function to add two numbers

```
const std = @import("std");

fn add(a: i32, b: i32) i32 {
    return a + b;
}

pub fn main() void {
    const result = add(10, 20);

    std.debug.print("Result: {}\n", .{result});
}
```
