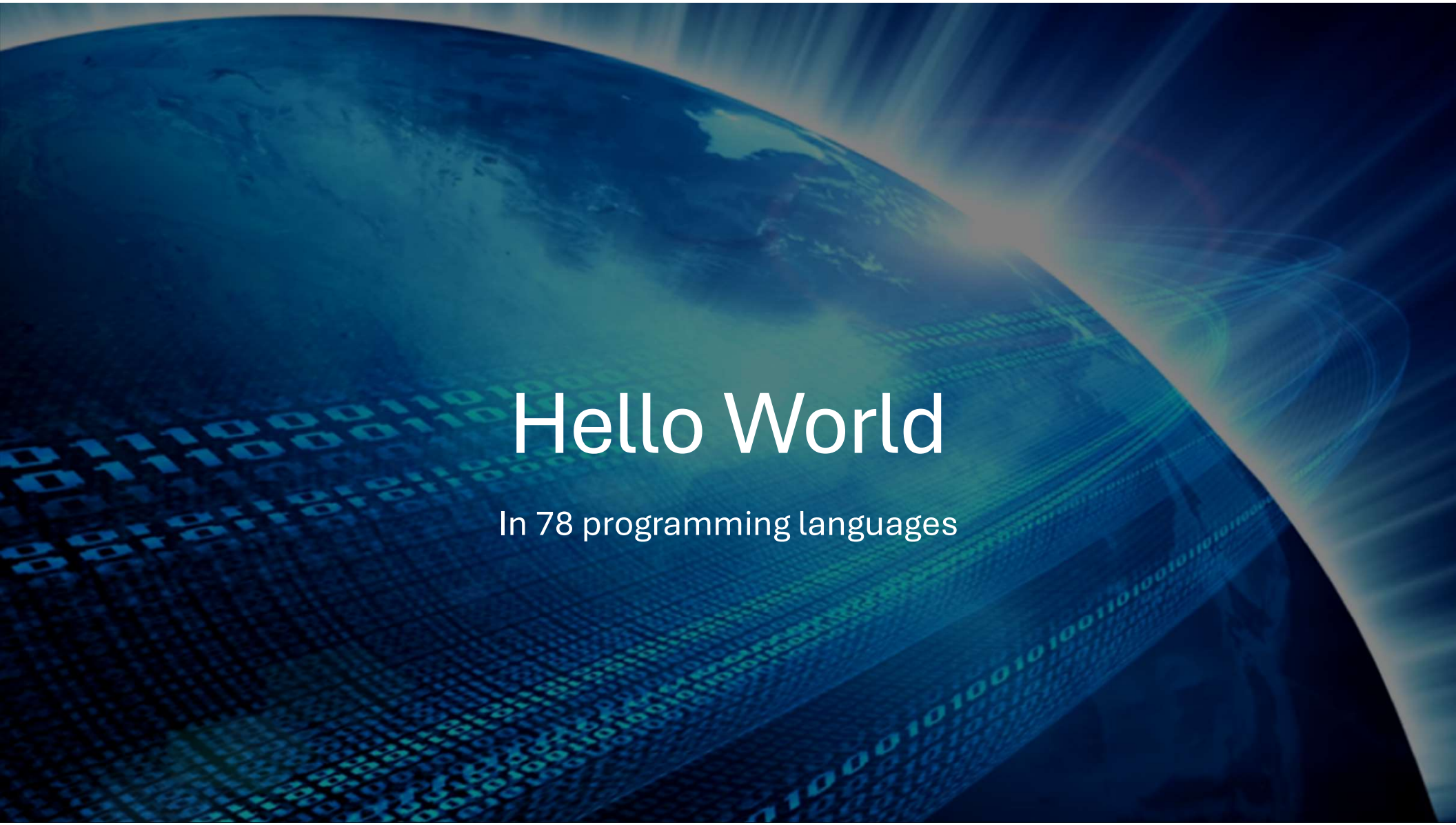
Hello World
In 78 programming languages

Python

```python
print('Hello, World!')
```

Julia

```julia
print("Hello, World!")
```

Java

```java
package com.example.helloworld;

public class HelloWorld {

    public static void main(String args[]) {
        System.out.print("Hello World!");
    }

}
```

C

```c
#include <stdio.h>

int main() {
    printf("Hello World");
    return 0;
}
```

C++

```cpp
#include <iostream>

int main() {
    std::cout << "Hello, World!" << std::endl;
    return 0;
}
```

C#

```csharp
using System;

class HelloWorld
{
    static void Main()
    {
        Console.WriteLine("Hello, World!");
    }
}
```

JavaScript

```
document.write('Hello, world!');
```

Perl

```
#!/usr/bin/perl
print "Hello World";
```

R

```
cat('Hello World')
```

Swift

```
import Foundation


print("Hello, World!")
```

Kotlin

```
fun main() {
    print("Hello World!\n")
}
```

Rust

```
fn main() {

    println!("Hello, world!");

}
```

HTML

```html
<!DOCTYPE html>
<html>
<head>
    <title>Hello, World!</title>
</head>
<body>
    <p>Hello, world!</p>
</body>
</html>
```

CSS

```css
body::before {
  content: "Hello World";
}
```

Ruby

```ruby
puts "Hello World"
```

SQL

```sql
SELECT 'Hello, World!' AS Greeting;
```

BASIC

```
10 PRINT "Hello, World!"
20 END
```

PHP

```php
<?php
echo "Hello, World!";
?>
```

Go

```go
package main

import "fmt"

func main() {
    fmt.Println("Hello, World")
}
```

TypeScript

```typescript
let message: string = 'Hello, World!';

console.log(message);
```

Z Shell

```zsh
echo "Hello, World!"
```

Scala

```scala
object HelloWorld {
  def main(args: Array[String]): Unit = {
    println("Hello, World!")
  }
}
```

Objective-C

```objc
#import <Foundation/Foundation.h>

int main (int argc, const char * argv[])
{
    NSAutoreleasePool *pool = [[NSAutoreleasePool alloc] init];
    NSLog (@"Hello, World!");

    [pool drain];
    return 0;
}
```

TCL

```tcl
puts "Hello, World!"
```

MATLAB

```
disp('Hello, World!');
```

Assembly

```
section .data
    hello db 'Hello, world!', 0

section .text
    global _start

_start:
    mov eax, 4
    mov ebx, 1
    mov ecx, hello
    mov edx, 13
    int 0x80

    mov eax, 1
    xor ebx, ebx
    int 0x80
```

Rub

```
cat('Hello, World!')
```

Delphi

```
program HelloWorld;
uses
  SysUtils;
begin
  WriteLn('Hello World');
end.
```

Cobra

```
class Program
    def main
        print "Hello, World!"
```

Dart

```
void main() {
    print("Hello, World!");
}
```

Solidity

```
pragma solidity ^0.4.22;
contract helloWorld {
 function renderHelloWorld () public pure returns (string) {
 return 'Hello, World!';
 }
}
```

Bash

```
#!/bin/bash
echo "Hello, World"
```

Ada

```
with Ada.Text_IO; use Ada.Text_IO;

procedure Hello_World is
begin
   Put_Line("Hello, World!");
end Hello_World;
```

Erlang

```
-module(hello_world).
-compile(export_all).

hello() ->
  io:format("hello, world~n").
```

Crystal

```
# file.cr
puts "Hello World"
```

Fortran

```
program helloworld
print *,'Hello World'
end program helloworld
```

XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<hello>World</hello>
```

K

```
"Hello, World!"
```

Haskell

```haskell
main :: IO ()
main = putStrLn "Hello, World!"
```

Lua

```lua
print("Hello, World!")
```

F#

```fsharp
open System

[<EntryPoint>]
let main argv =
    Console.WriteLine("Hello, World!")
    0
```

Clojure

```clojure
(println "Hello, World!")
```

Elixir

```
IO.puts "Hello, World!"
```

Elm

```
module Main exposing (..)


import Html exposing (text)


main =
    text "Hello, World!"
```

Nim

```
echo "Hello World"
```

Io

```
"hello world" println
```

B

```
main( ) {
    extern a, b, c;
    putchar(a); putchar(b); putchar(c);
putchar('!*n');
}

a 'hell';
b 'o, w';
c 'orld';
```

In B character constants are limited to ASCII characters, which is why the message is divided into multiple variables.

Cobol

```
    IDENTIFICATION DIVISION.
    PROGRAM-ID. HELLO.
    PROCEDURE DIVISION.
        DISPLAY 'Hello, World!'.
        STOP RUN.
```

IaC

```
resource "null_resource" "hello_world" {
  provisioner "local-exec" {
    command = "echo Hello, World!"
  }
}
```

HCL

```
output "greeting" {
  value = "Hello, World!"
}
```

AutoCode

```
BEGIN
  DISPLAY 'Hello, World!'
END.
```

A+

```
'Hello World'
```

E

```
System.out.println("Hello World")
```

Factor

```
USE: io
"Hello, World!" print
```

Joy

```
"Hello, World!" print
```

Hack

```
<?hh // strict

namespace
Hack\UserDocumentation\GettingStarted\HelloWorld;

<<__EntryPoint>>
function main(): void {
  echo "Hello, world!\n";
}
```

Oak

```
std := import('std')
std.println('Hello, World!')
```

Zig

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

pub fn main() void {
    const stdout = std.io.getStdOut().writer();
    try stdout.print("Hello, World!\n");
}
```

PeopleCode

```
Local string &message;
&message = "Hello, World!";
MessageBox(0, "", 0, 0, &message);
```

Maple

```
hello := proc()
    "Hello World";
end proc;
```

Boo

```
print "Hello, World!"
```

Caml

```
print_endline "Hello, world!";;
```

Clean

```
module hello
import StdEnv


Start = "Hello, world!"
```

CoffeeScript

```
console.log "Hello, World!"
```

bc

```
print "hello world"
```

AppleScript

```
display dialog "Hello, World"
```

V

```v
fn main() {
    println('hello world')
}
```

WebAssembly

```wasm
(module
    ;; Imports from JavaScript namespace
    (import  "console"  "log" (func  $log (param  i32  i32))) ;; Import log function
    (import  "js"  "mem" (memory  1)) ;; Import 1 page of memory (54kb)

    ;; Data section of our module
    (data (i32.const 0) "Hello World from WebAssembly!")

    ;; Function declaration: Exported as helloWorld(), no arguments
    (func (export  "helloWorld")
        i32.const 0  ;; pass offset 0 to log
        i32.const 29  ;; pass length 29 to log (strlen of sample text)
        call  $log
        )
)
```

Red

```
Red [Title: "Hello World in Red"]

print "Hello, World!"
```

Frege

```
module Main where


main args = do
    println "Hello, World!"
```

D

```d
import std.stdio;

void main()
{
    writeln("Hello, World!");
}
```

LiveScript

```
console.log 'Hello, World!'
```