




# Edge Computing

Tassadaq Hussain Cheema

Professor EE Department NAMAL University

February 27, 2018

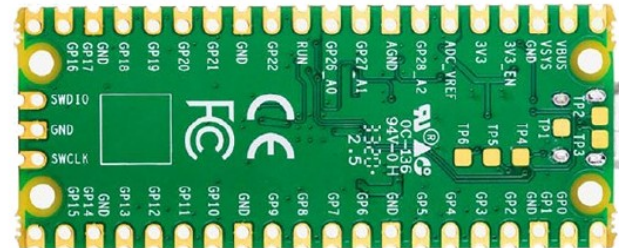
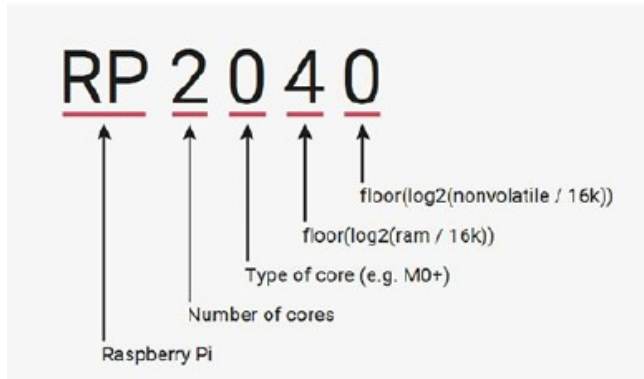
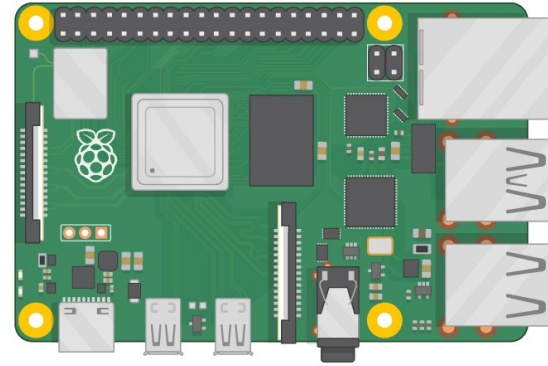


Edge is about processing data closer to where it's being generated, enabling processing at greater speeds and volumes, leading to greater action-led results in real time.

- Hardware Architecture
- Operating System
- Application Development Framework

# Hardware Architecture

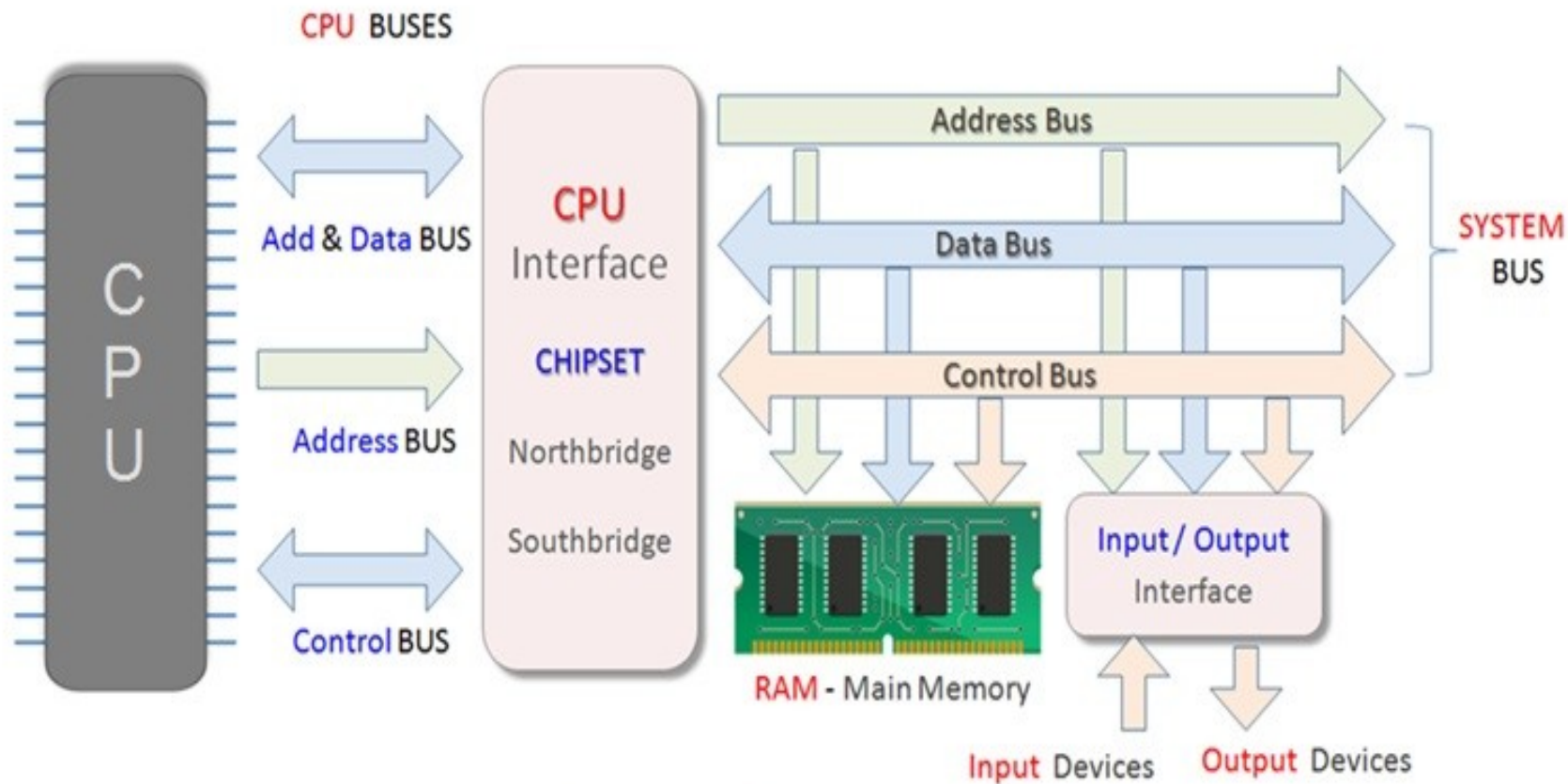
- Single Board Computer
  - Raspbberypi
  - Raspbberypi Pico



The Raspberry Pi Pico - bottom view (courtesy of raspberrypi.org)

# Hardware Specifications

- Processor Architecture
  - Number of Cores (ALU) and Operating Frequency
- Processor Local Bus
  - Instruction and Data Bus
  - Program and Data Memory
  - Local Memory (Cache and ScratchPad)
- Processor External Bus (Peripheral Bus)
  - Main Memory
  - DMA, PWM, ADC, DAC
  - I/O Interfaces



	Raspberry Pi 3 [7]	Odroid C-2 [11]	Odroid XU-4 [3, 10]
SoC	Broadcom BCM2837	Amlogic S905	Samsung Exynos 5422
CPU	4×ARM Cortex-A53 1.2GHz, in-order	4×ARM Cortex-A53 2.0GHz, in-order	ARM big.LITTLE octa core 4×A7, 1.5GHz, in-order 4×A15, 2.0GHz, out-of-order
Arch	ARMv8-A (64 bit)	ARMv8-A (64 bit)	ARMv7-A (32 bit)
L1\$ (I/D)	32KB/32KB	32KB/32KB	32KB/32KB
L2\$	512KB	512KB	512KB (A7), 2MB (A15)
Memory	1GB LPDDR2 900 MHz	2GB DDR3 32 bit / 912Mhz	2GB LPDDR3 32 bit / 933MHz, PoP
GPU	BCM VideoCore IV	ARM Mali-450	ARM Mali-T628 MP6
Compute	no	no	OpenCL 1.1
OS	Ubuntu MATE 15.10	Ubuntu MATE 16.04	Ubuntu MATE 15.10
Kernel	4.1.18-v7+ (armv7l)	3.14.29-29 (aarch64)	3.10.96-78 (armv7l, HMP)
Compiler	GCC v5.2.1	GCC v5.3.1	GCC v5.2.1

# RPI-Pico: Hardware Specifications

- **Operating Frequency and Processor: Dual ARM Cortex-M0+ @ 133MHz**
- RAM and ROM: 264kB on-chip SRAM in six independent banks, Support for up to 16MB of off-chip flash memory
- **External Peripherals :**
  - DMA controller
  - Fully connected AHB crossbar
  - Interpolator and integer divider peripherals
  - On-chip programmable LDO to generate core voltage
  - 2 on-chip PLLs to generate USB and core clocks
  - 30 GPIO pins, 4 of which can be used as
    - analogue inputs
  - Serial: 2 UARTs, 2 SPI controllers, 2 I2C controllers
  - 16 PWM channels
  - USB 1.1 controller and PHY, with host and device support
  - 8 PIO state machines

Edge is about processing data closer to where it's being generated, enabling processing at greater speeds and volumes, leading to greater action-led results in real time.

- Hardware Architecture
- **Operating System**
- Application Development Framework

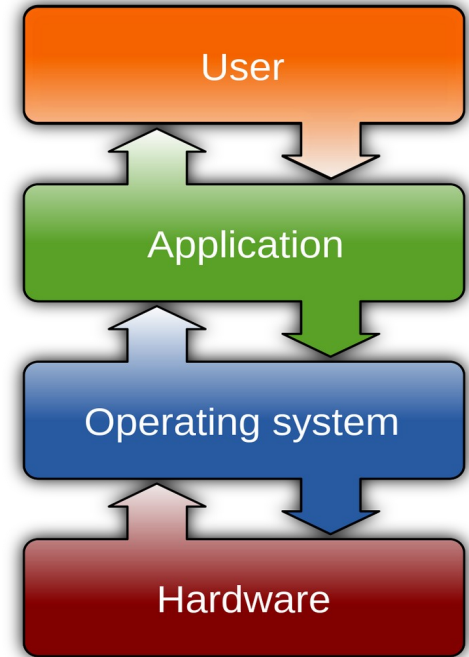


# Operating System

- An Operating System (OS) is a software program that acts as an intermediary between the computer hardware and the application software.
- It manages the computer's hardware resources, such as the central processing unit (CPU), memory, disk drives, and input/output devices, and provides services to application software, such as file management, memory management, and process scheduling. It also provides a user interface for interacting with the computer and running application programs.

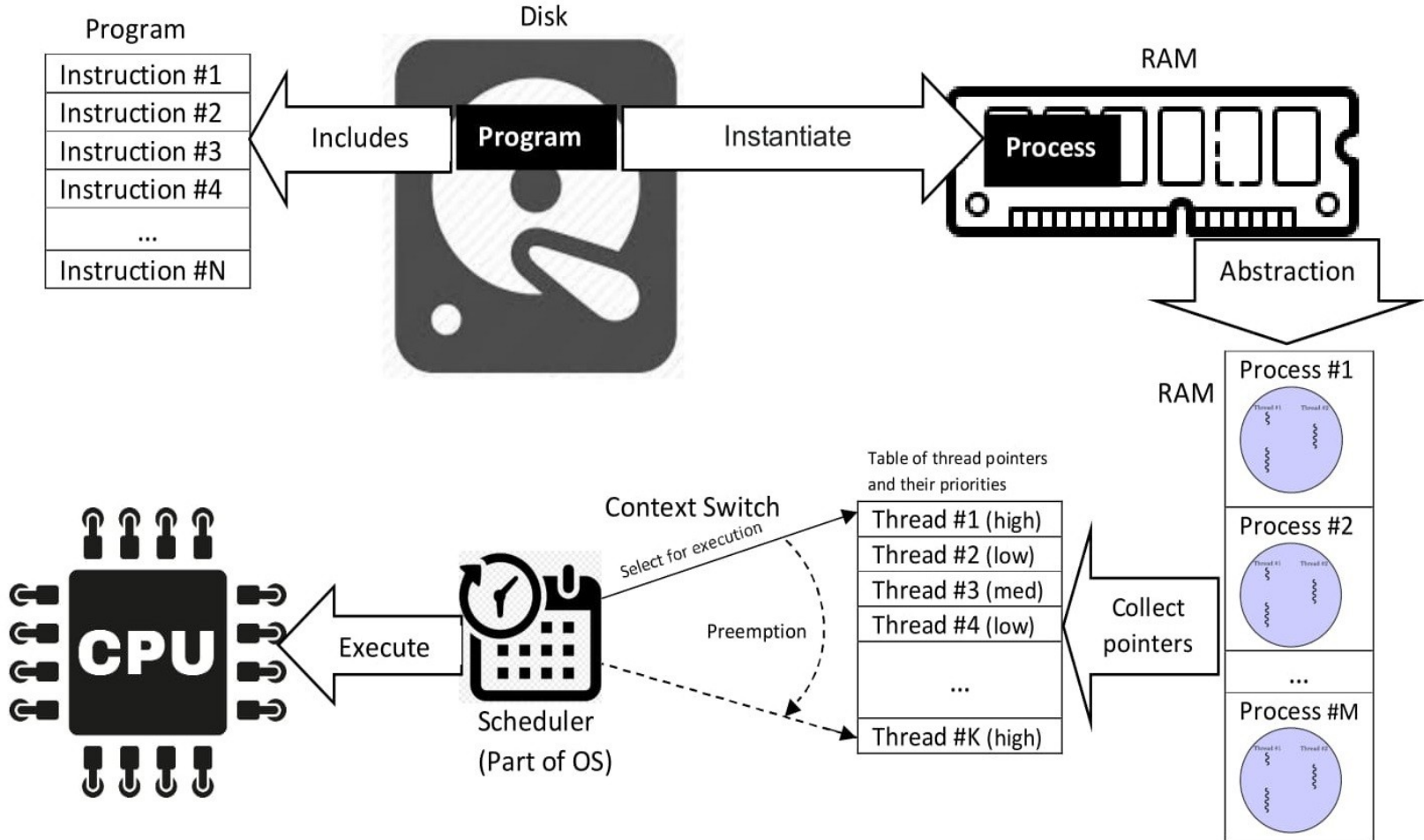
**Task Manager, Memory Manager, File Manager, Network Manager, Power Manager, I/O Manager**

- Examples of popular operating systems include Windows, macOS, Linux, and Android.

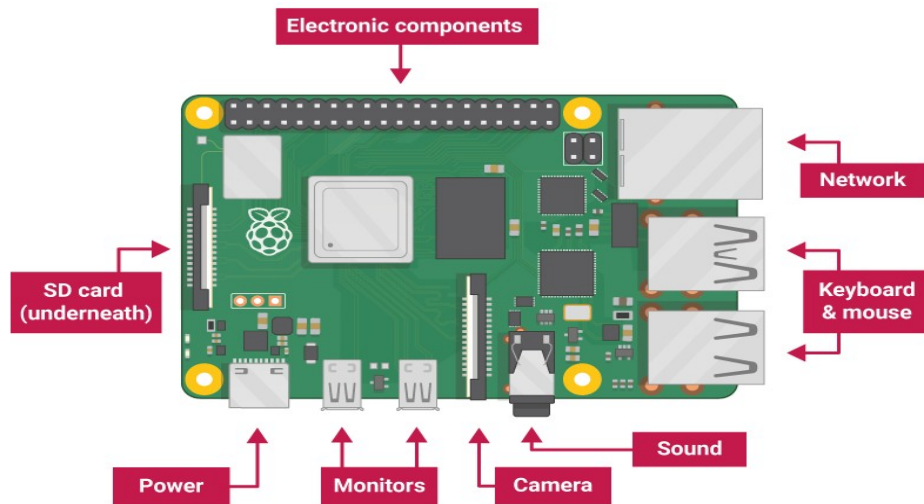


# Types of OS

- Single-user, single-tasking OS: This is the simplest type of operating system and can only support one user and one task at a time. Examples include MS-DOS and early versions of Apple's Mac OS.
- Single-user, multi-tasking OS: This type of operating system can run multiple applications simultaneously, but only supports one user at a time. Examples include Windows, macOS, and Linux.
- Multi-user OS: This type of operating system can support multiple users running multiple applications simultaneously. Examples include UNIX and mainframe operating systems.
- Real-time OS: This type of operating system is designed to handle time-sensitive applications, such as those used in robotics, aerospace, and industrial control systems. Examples include VxWorks and QNX.
- Mobile OS: This type of operating system is designed for mobile devices, such as smartphones and tablets. Examples include Android, iOS, and Windows Mobile.
- Embedded OS: This type of operating system is designed to run on specialized devices, such as digital cameras, printers, and routers. Examples include Embedded Linux and Windows Embedded.
- Network OS: This type of operating system is designed to manage and coordinate multiple computers and devices on a network. Examples include Novell NetWare, Windows Server, and Linux server.
- Server OS: This type of operating system is designed specifically to run server applications, such as web servers, email servers, and database servers. Examples include Windows Server, Linux server, and macOS Server.



# Steps Installing OS

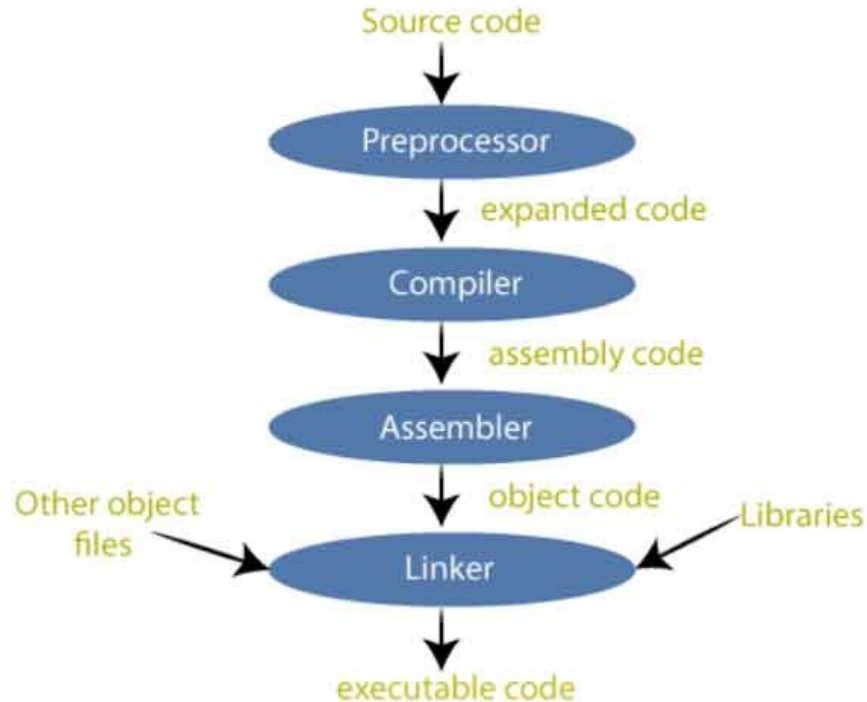


- Download OS Image
- Boot Loader: SD Card
  - Stored in the first sector of the SD card.
- Insert and Run
- Firmware:
  - Collection of low-level software components that interact directly with the hardware of the device.

Edge is about processing data closer to where it's being generated, enabling processing at greater speeds and volumes, leading to greater action-led results in real time.

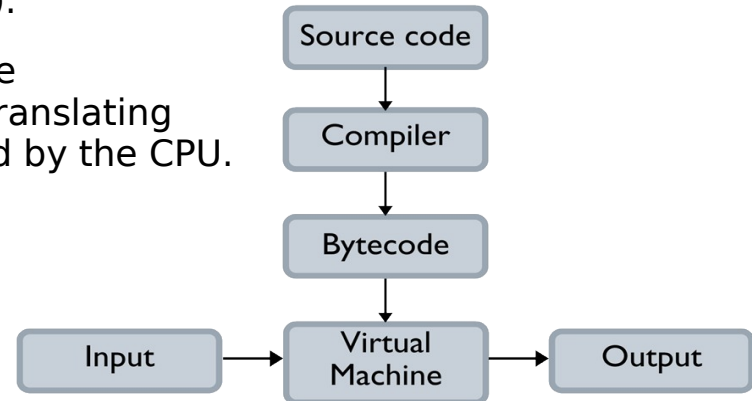
- Hardware Architecture
- Operating System
- **Application Development Framework**

# Application Development Framework



# Python Execution Flow

- **Tokenization:** When you run a Python program, the Python interpreter first tokenizes the source code. This means it breaks the code down into a sequence of tokens (e.g. keywords, identifiers, operators, etc.), each of which represents a specific element of the code.
- **Parsing:** Once the source code has been tokenized, the Python interpreter then parses it to build an abstract syntax tree (AST). The AST is a tree-like structure that represents the syntactic structure of the code, and it's used by the interpreter to determine the meaning of the code.
- **Compilation:** After the AST has been built, the Python interpreter then compiles it into bytecode. Bytecode is a lower-level, platform-independent representation of the code that can be executed by the Python virtual machine (PVM).
- **Execution:** Finally, the Python interpreter executes the bytecode using the PVM. The PVM is responsible for translating the bytecode into machine code that can be executed by the CPU.

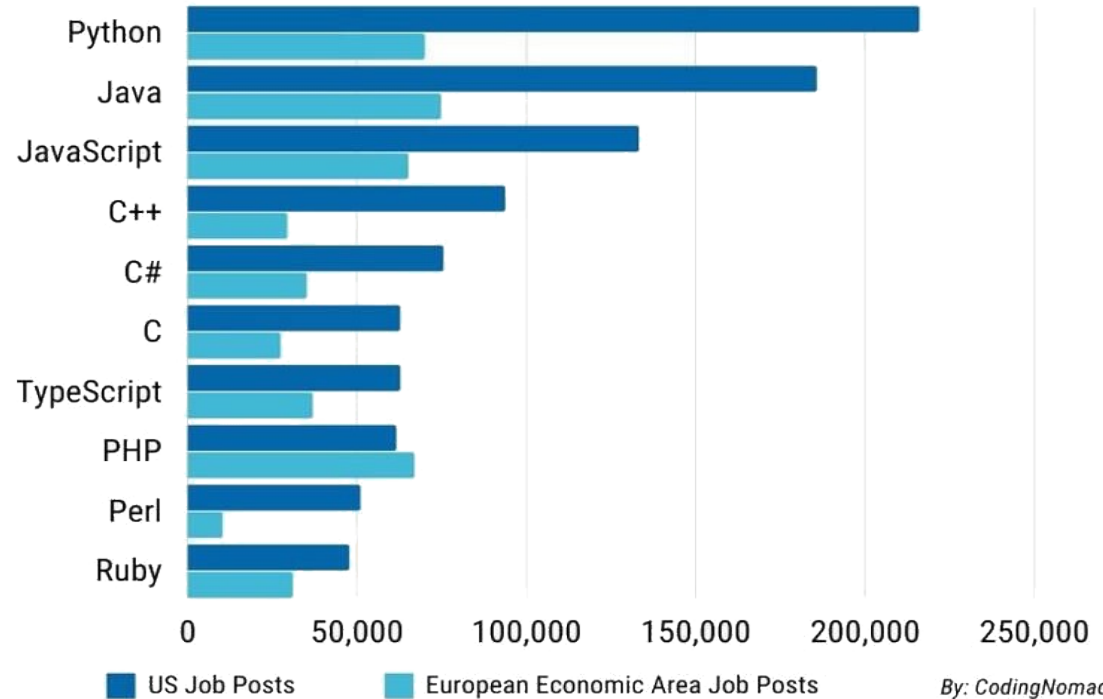


# Application Development Languages

- Assembly
- C/C++
- Python
- 

## Most in-demand programming languages of 2022

*Based on LinkedIn job postings in the USA & Europe*





# Edge Computing: Application Development

- **Hardware**
  - Raspberry-pi
  - Raspberry-pi Pico
- **Linux OS**
- **Thonny**
  - Micro-Python

# Thonny: Application Development Environment

- **Download Thonny Application Development**
  - <https://thonny.org/>
- **Configuration (HandsOn)**

## Generic Code Structure

#Add libraries

#Add global variables

#Initialization (Peripherals etc.)

#Read Data

#Store it on local memory

#Apply local processing

#Transfer / Store / Display

# LED Blinking

```
from machine import Pin
```

```
led = Pin(25, Pin.OUT)
```

```
led.toggle()
```

# Read from ADC

```
from machine import ADC, Pin
```

```
import time
```

```
adc = ADC(Pin(26))
```

```
while True:
```

```
    print(adc.read_u16())
```

```
    time.sleep(1)
```

```
from machine import Pin, Timer
```

```
led = Pin(25, Pin.OUT)
```

```
timer = Timer()
```

```
def blink(timer):
```

```
    led.toggle()
```

```
timer.init(freq=2.5, mode=Timer.PERIODIC, callback=blink)
```

# ADC and Temperature Sensor

```
import machine
```

```
import utime
```

```
sensor_temp = machine.ADC(4)
```

```
conversion_factor = 3.3 / (65535)
```

```
while True:
```

```
    reading = sensor_temp.read_u16() * conversion_factor
```

```
    temperature = 27 - (reading - 0.706)/0.001721
```

```
    print(temperature)
```

```
    utime.sleep(2)
```

## Tasks:

- Read multi-sensor Data
- Store it local data structure (As per specification of sensor and application requirement)
- Perform pre-processing (Signal Conditioning, Enhancement, noise removal etc)
- Transfer Data