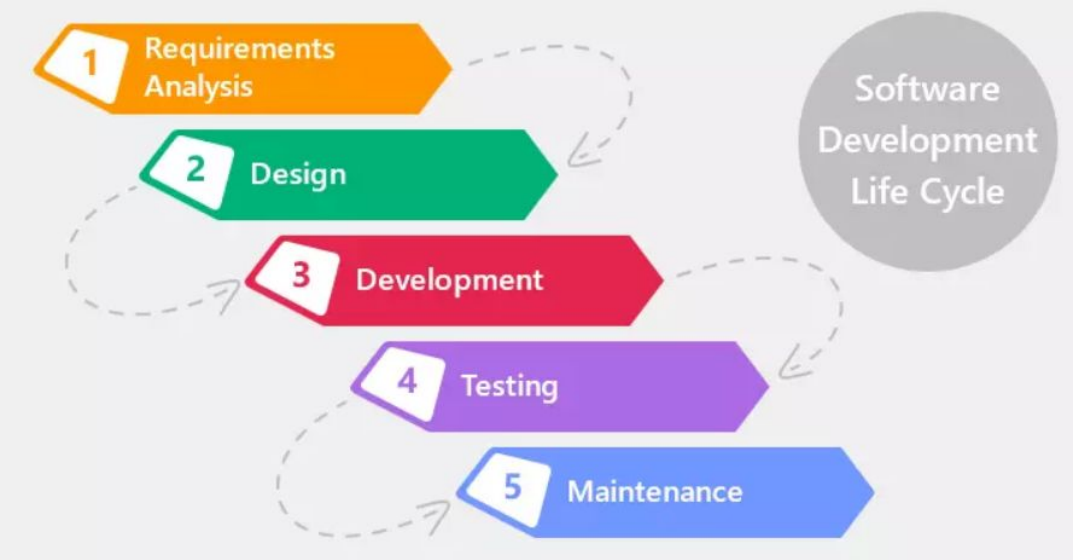
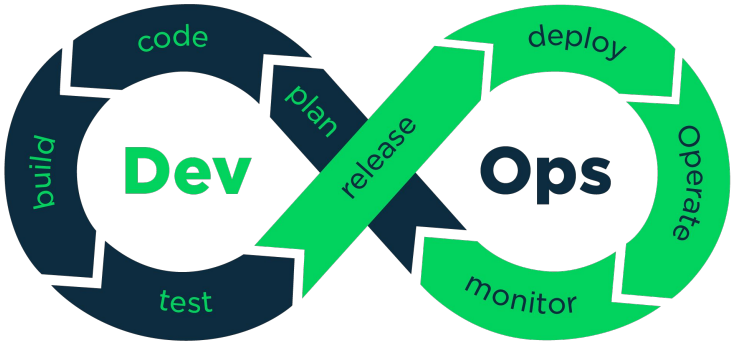
Software Technology

Content

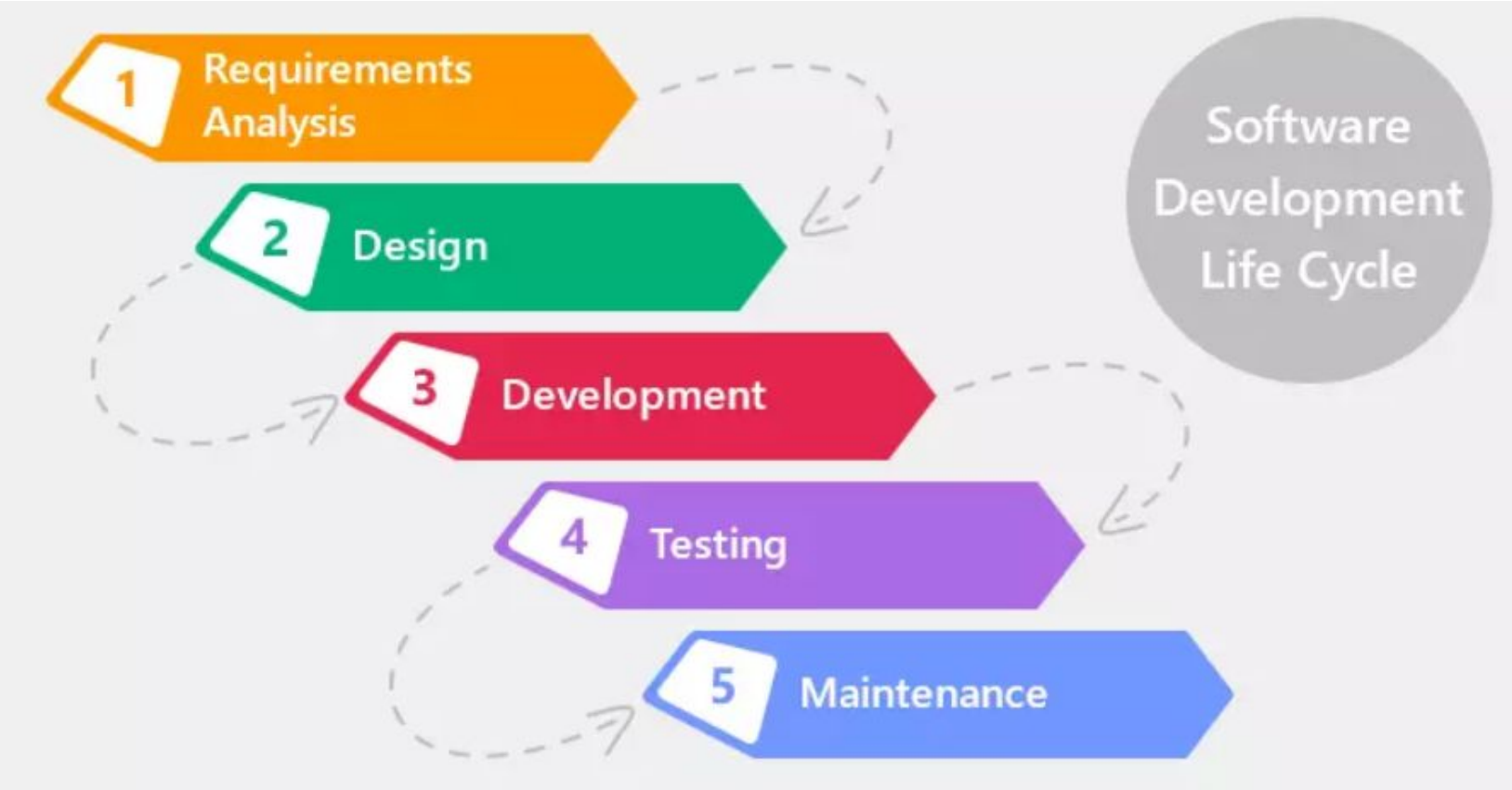
- Software Development Life Cycle
- Software Development Framework
- Software Programming Language
- MVC Pattern using Python

Software Development Life Cycle



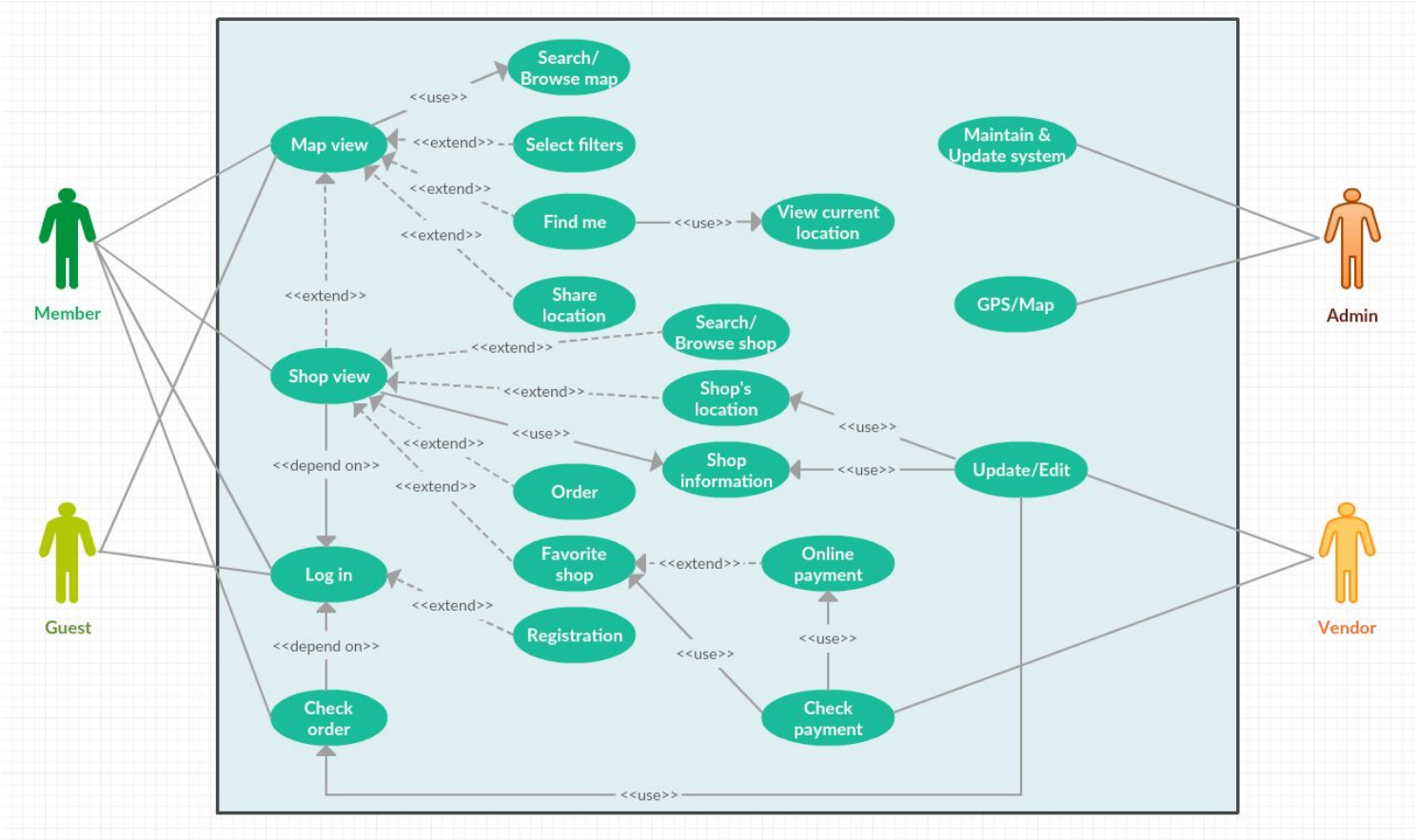


Software Development Life Cycle



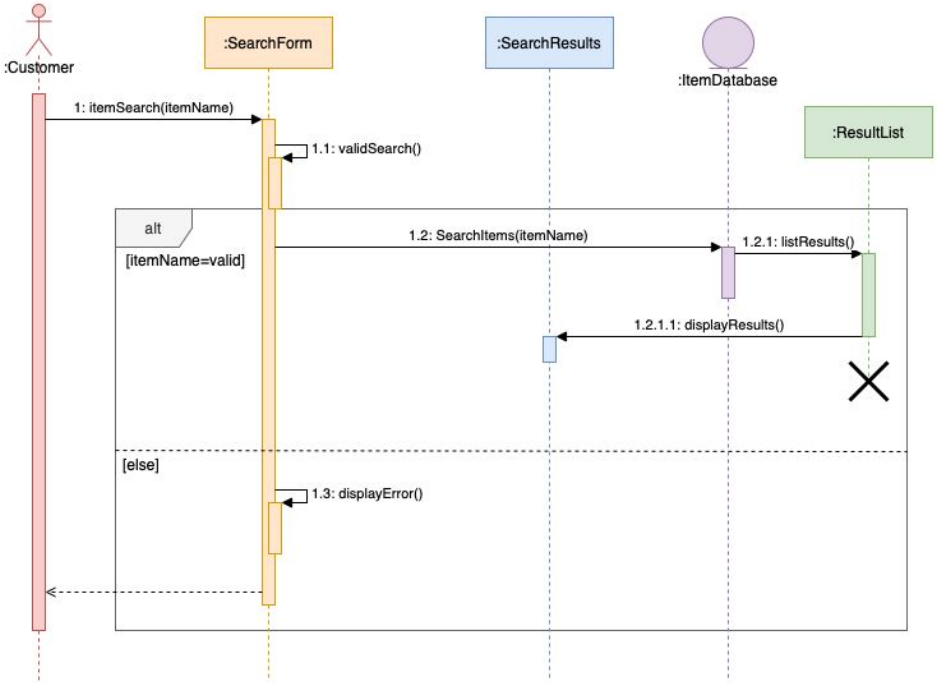
- A developer mainly works in the **Development** phase

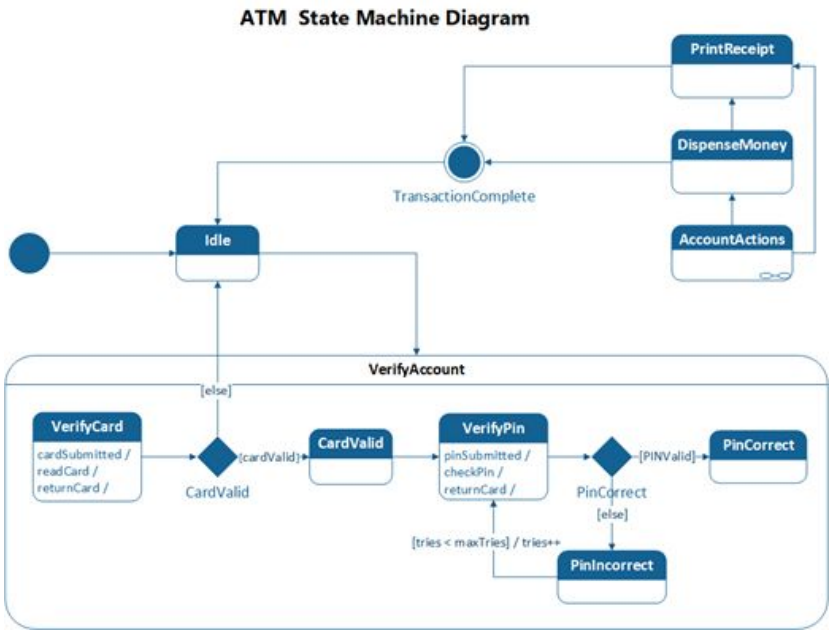
Requirement Analysis



- Use-case diagram is used to model system features

Design



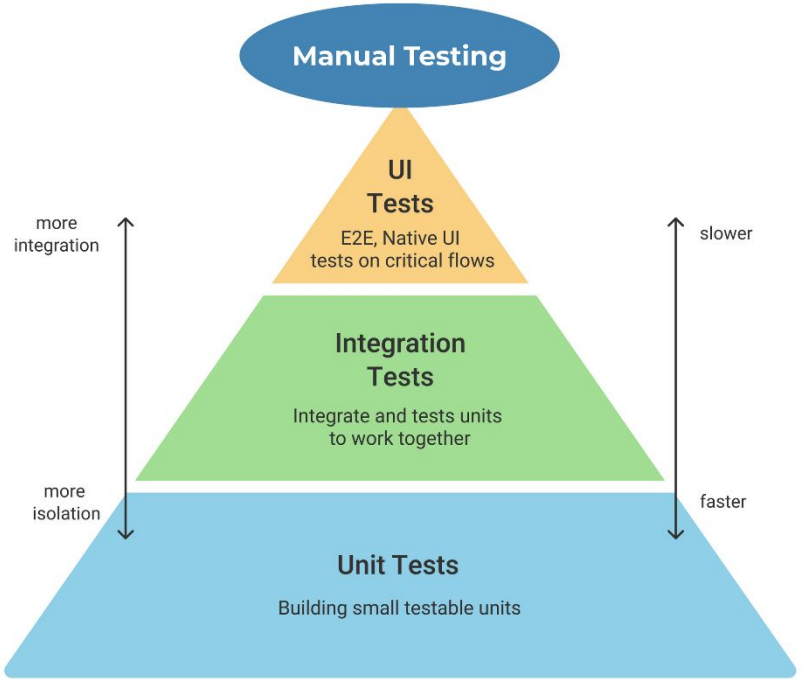


- Sequence diagram, Flowchart, State Machine, ….

Development

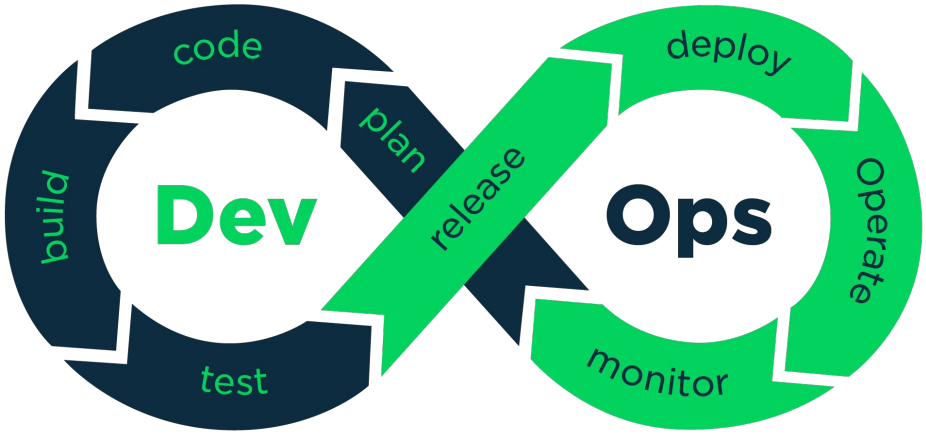
- CAD Tools: Github, SourceTree
- Software Editors: Visual studio code
- Project template: MVC model

Software Testing



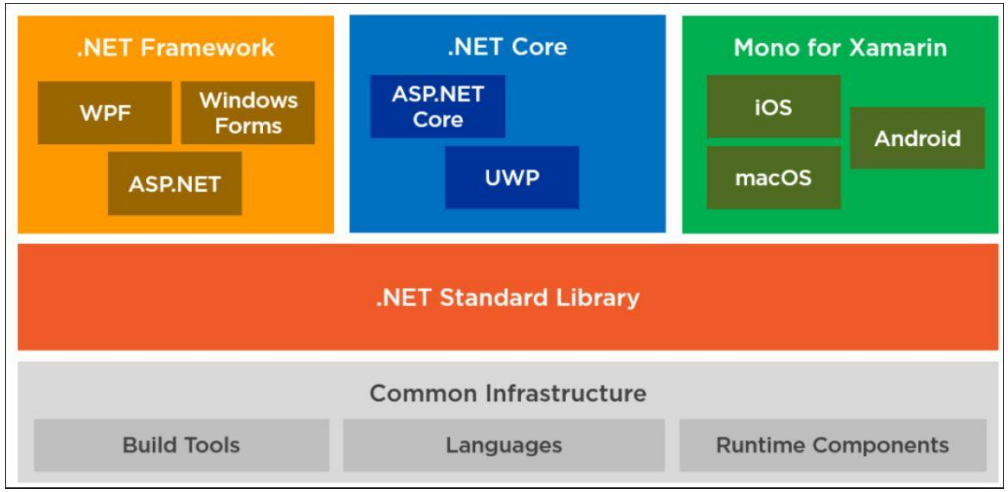
- Validation and Performance
- Testing reports

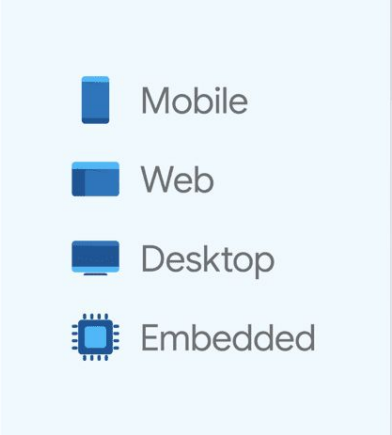
Maintenance or Operations (DevOps)



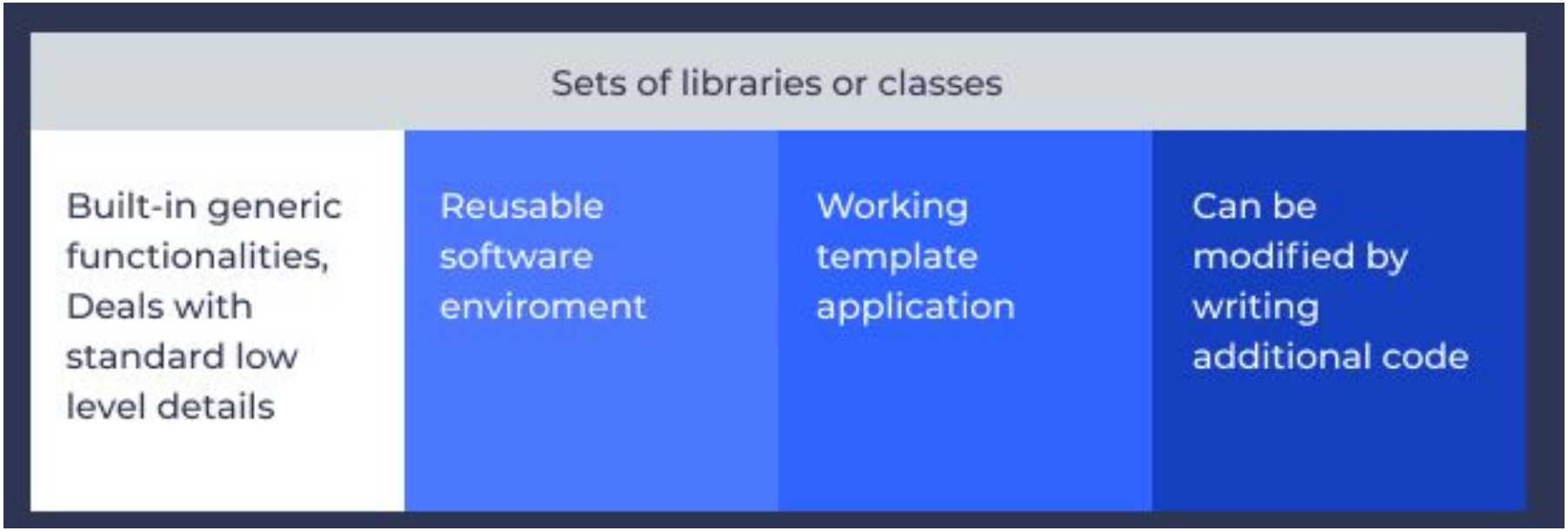
- The most important phase to upgrade the project:
  - Error tracking
  - CD and CI

Software Development Framework



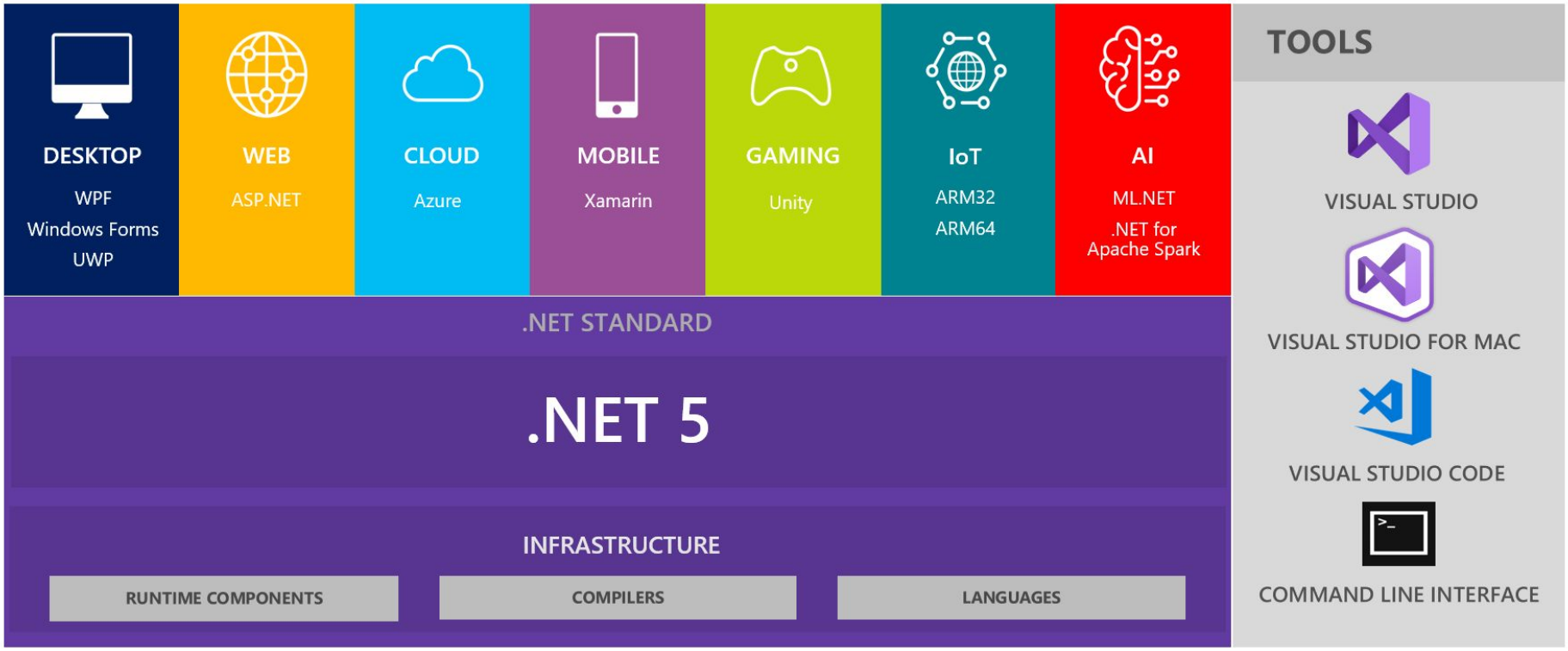


What is Software Development Framework?



- Supported libraries, SDK, programming languages for software development and deployment

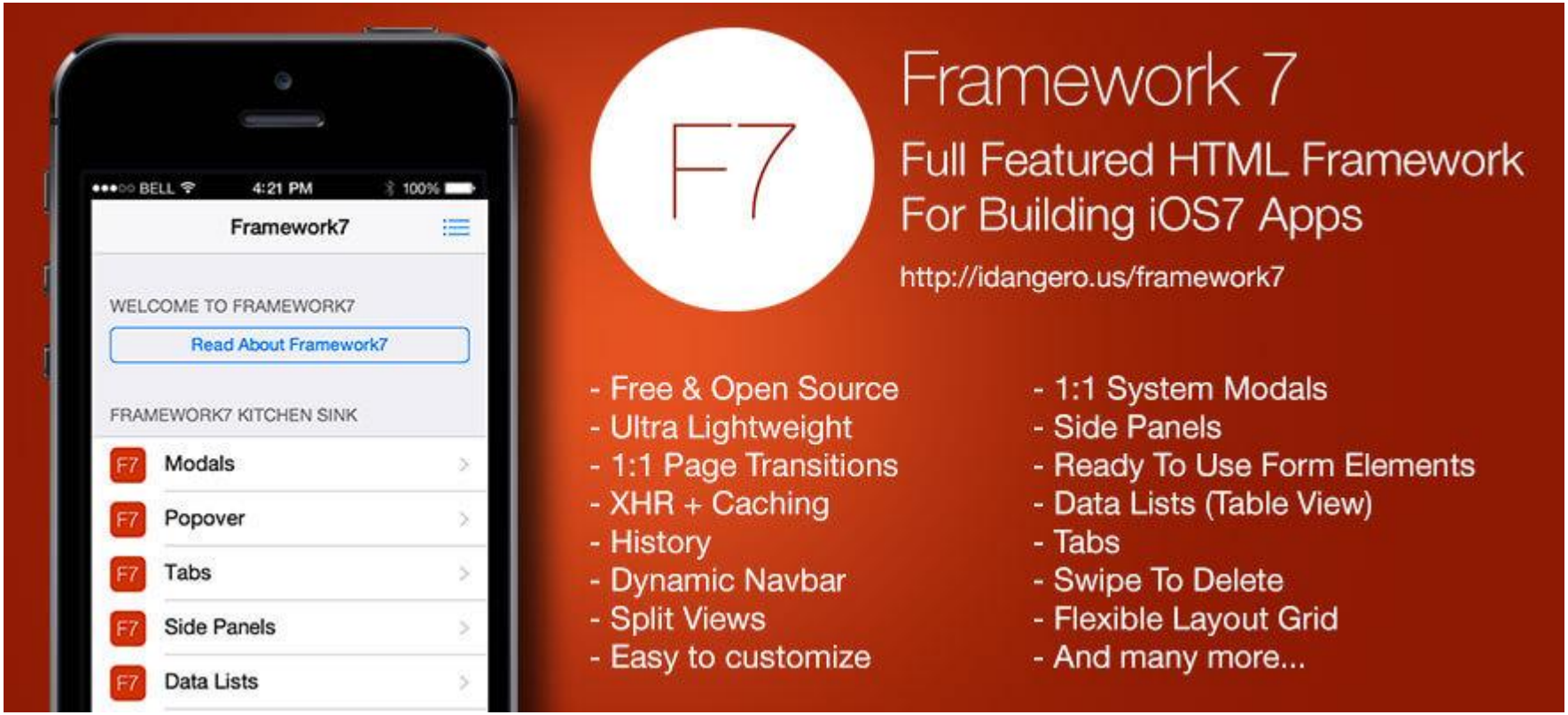
Dot Net Framework



- Unify framework for **Desktop Applications**
- **C, C++ and C#** are the most popular languages
- Visual studio IDE: Drag and Drop!!!

Framework 7 – React JS

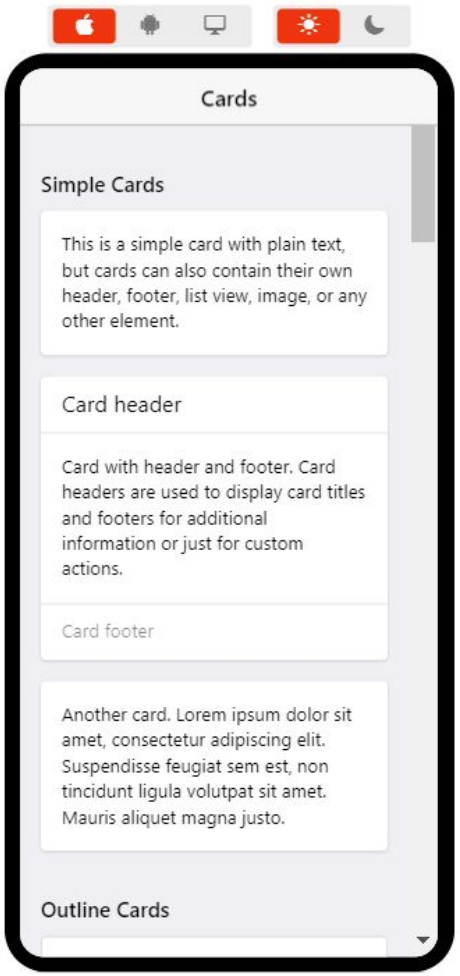
- https://framework7.io/react/



Example of Framework 7

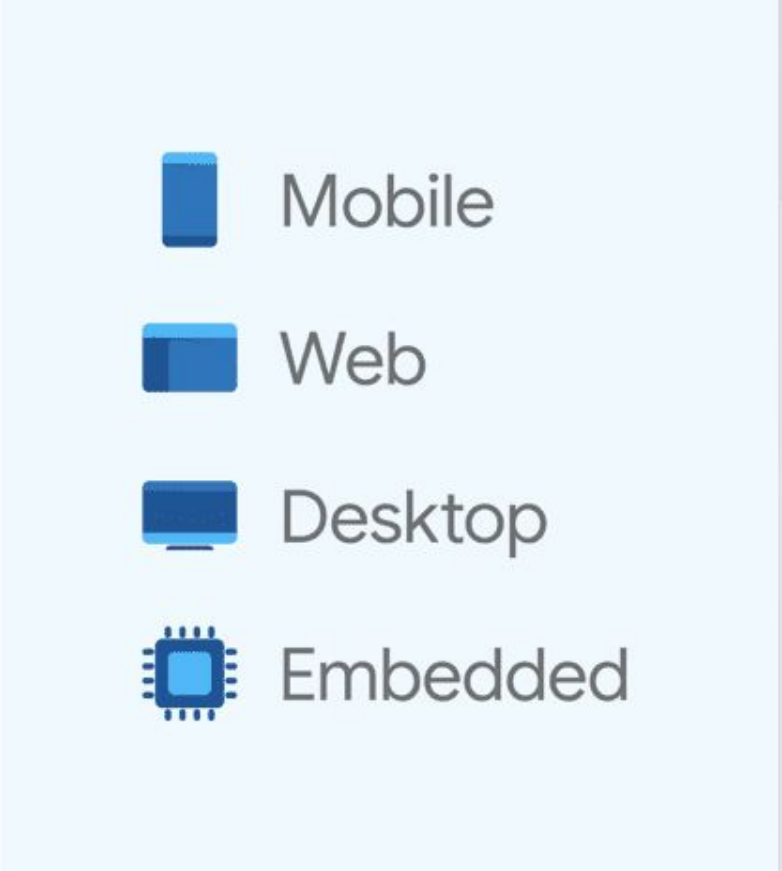
Examples

```
import React from 'react';
import {
  Page,
  Navbar,
  BlockTitle,
  Card,
  CardHeader,
  CardContent,
  CardFooter,
  Link,
  List,
  ListItem,
} from 'framework7-react';
import './cards.css';

export default () => (


  <Page>
    <Navbar title="Cards" />
    <BlockTitle>Simple Cards</BlockTitle>
    <Card content="This is a simple card with plain text, but cards can also contain their
own header, footer, list view, image, or any other element."></Card>
    <Card
      title="Card header"
      content="Card with header and footer. Card headers are used to display card titles and
footers for additional information or just for custom actions."
      footer="Card footer"
    ></Card>
    <Card content="Another card. Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Suspendisse feugiat sem est, non tincidunt ligula volutpat sit amet. Mauris aliquet magna
justo. "></Card>

    <BlockTitle>Outline Cards</BlockTitle>
    <Card
```



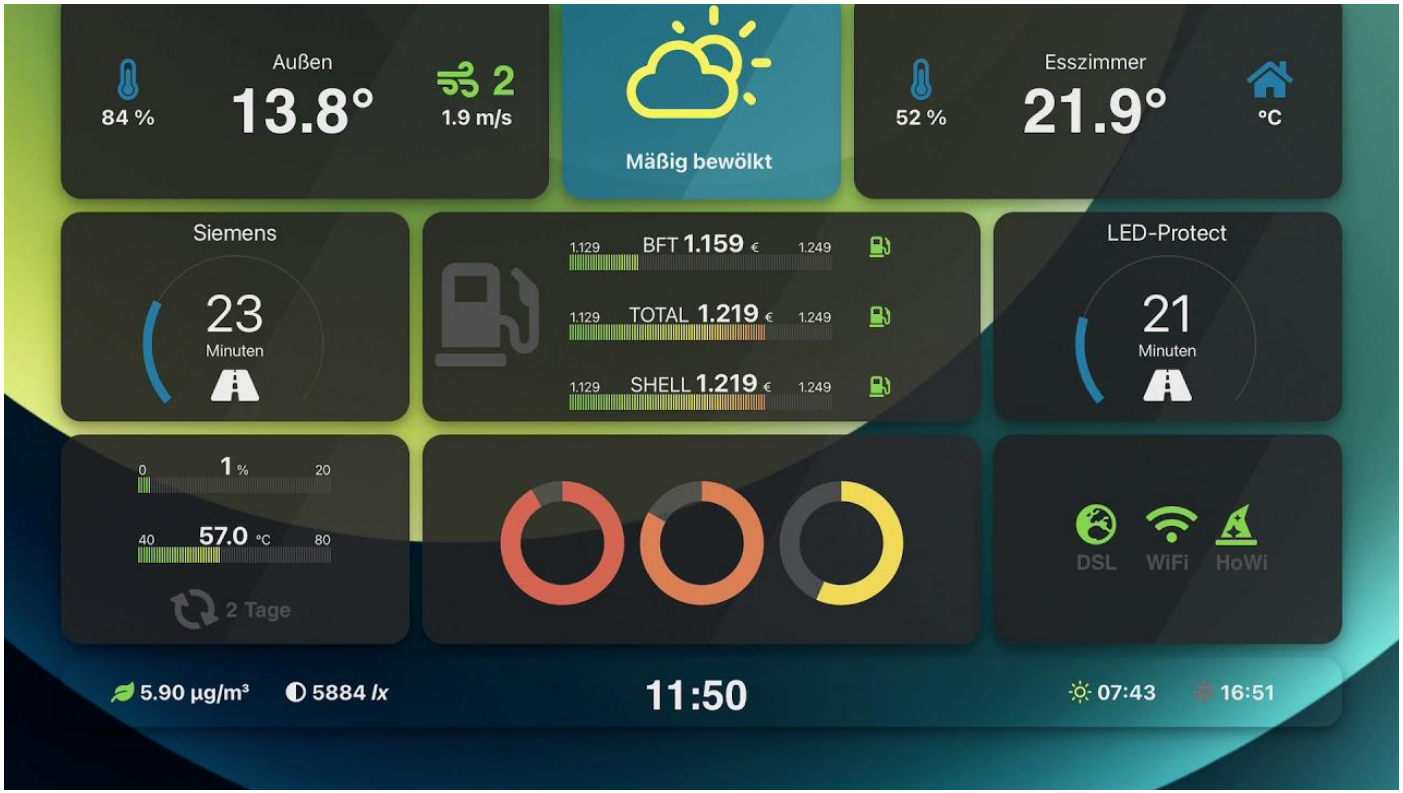
- Framework 7 Web-App development environment

Android Studio and Flutter



- Flutter is an Android Studio plug-in:
  - Access to the hardware of the mobile device (Bluetooth, NFC, Wifi, …)

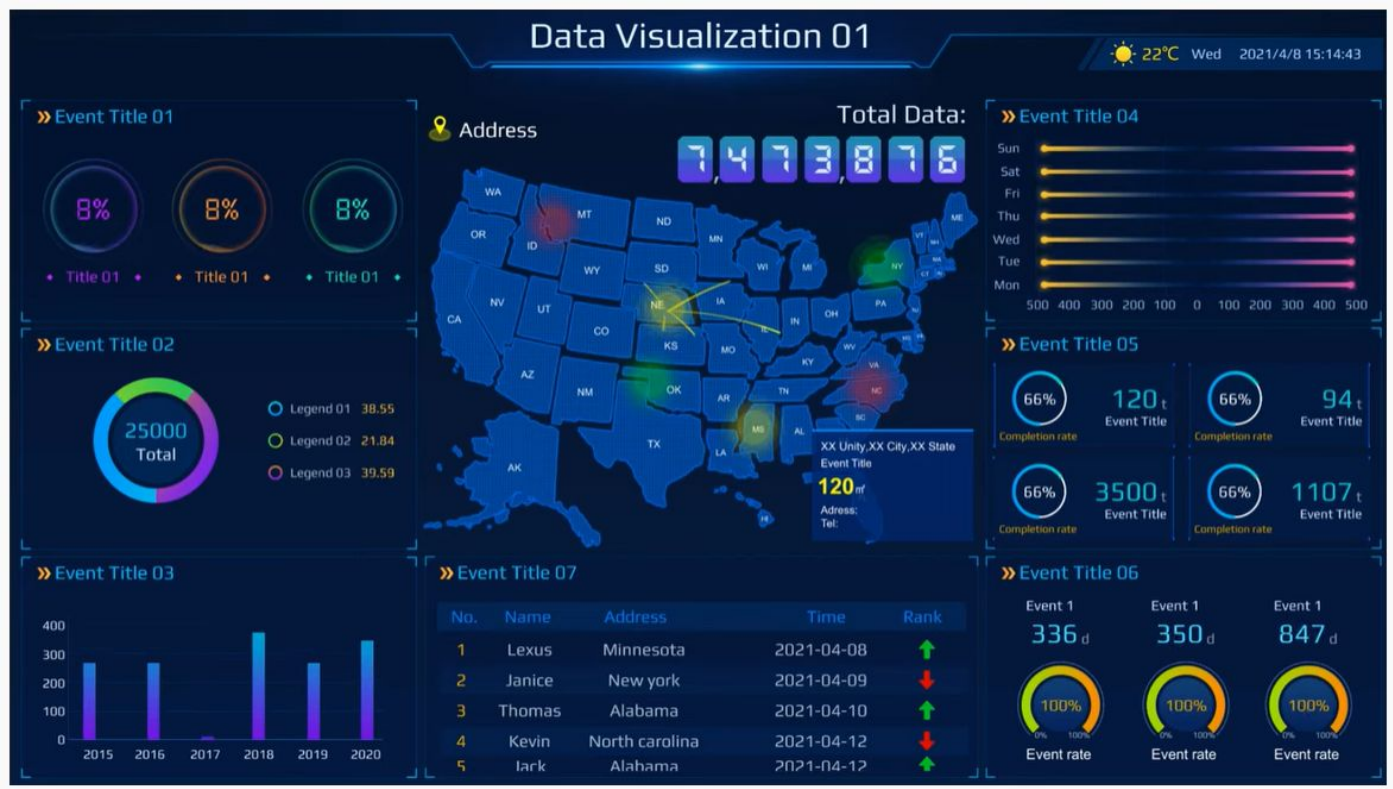
Unity 3D



- A product from Microsoft
- Not only for game developers, but also a cross-framework for software developemt
- C# programing language

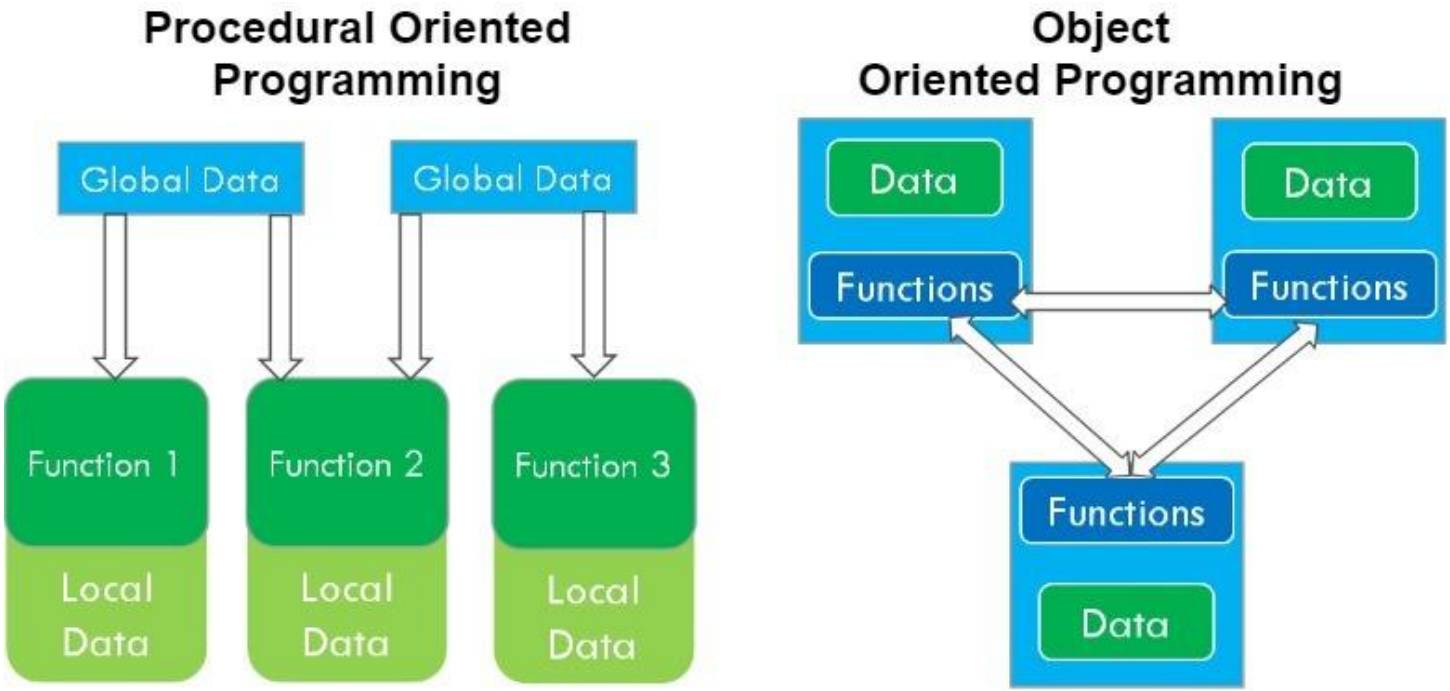
Unity Asset Store

- https://assetstore.unity.com/templates
- https://www.youtube.com/watch?v=uZaFHx1daLU



Software Programming Language

Procedure and OOP Programming Language

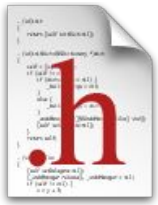


- In procedural programming **major focus is on functions** rather than data
- In Object Oriented Programming, focus is given on **data and how to access that data** and the real world scenarios share more resemblance

Basic concepts of OOP

- Classes & Objects:
  - A class is a template which consists of data members
  - ***An Object is a variable of type Class***
- Inheritance:
  - When one class acquires all the properties and behaviors of parent class
- Polymorphism:
  - When one task is performed by different ways
- Abstraction:
  - **Hiding internal details** and showing functionality is known as **abstraction**
- Encapsulation:
  - **Binding (or wrapping) code and data** together into a **single unit**

C/C++ Programming Language



- The best performance program
- **Compiler** programing language

```
#ifndef __LED_H_
#define __LED_H_

#include <system_lib.h>
#include "user_lib.h"
#include "UserFolder/lib.h"

extern int T_on;
extern int T_off;

void setOn(long duration);
void setoff(long duration);

#endif
```

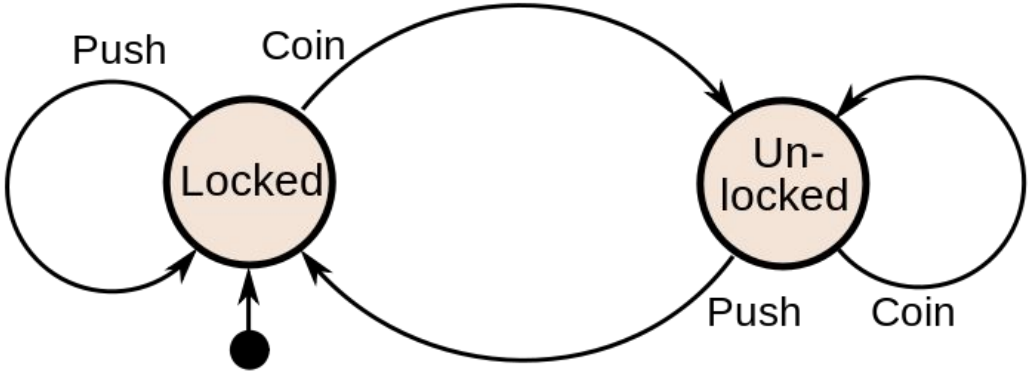
```
#include "led.h"

int T_on;
int T_off;
int counter;

void setOn(long duration){
    //TODO: set LED on here
}
void setoff(long duration){
   //TODO: set LED off here
}
void delay(long duration){
  //TODO: set delay here
}
```

Finite State Machine (FSM)

- **Finite-State Machine** (**FSM**) or **Deterministic Finite Automata** (**DFA**), **finite automaton**, or simply a **state machine**, is a mathematical model of computation

Current State	Input	Next State
Locked	Coin	Unlocked
	Push	Locked
Unlocked	Coin	Unlocked
	Push	Locked

A turnstile



Finite State Machine Programming

```
while(1){
  switch(status){
    case LOCKED:
      lock_turnstile(); //operation in a state
      if(Coin == true)  //transition condition
         status = UNLOCKED; //next state
      break;
    case UNLOCKED:
      unlock_turnstile(); //operation in a state
      if(Push == true)    //transition condition
         status = LOCKED; //next state
      break;
    default:
      break;
    }
}
```

Python Programing Language

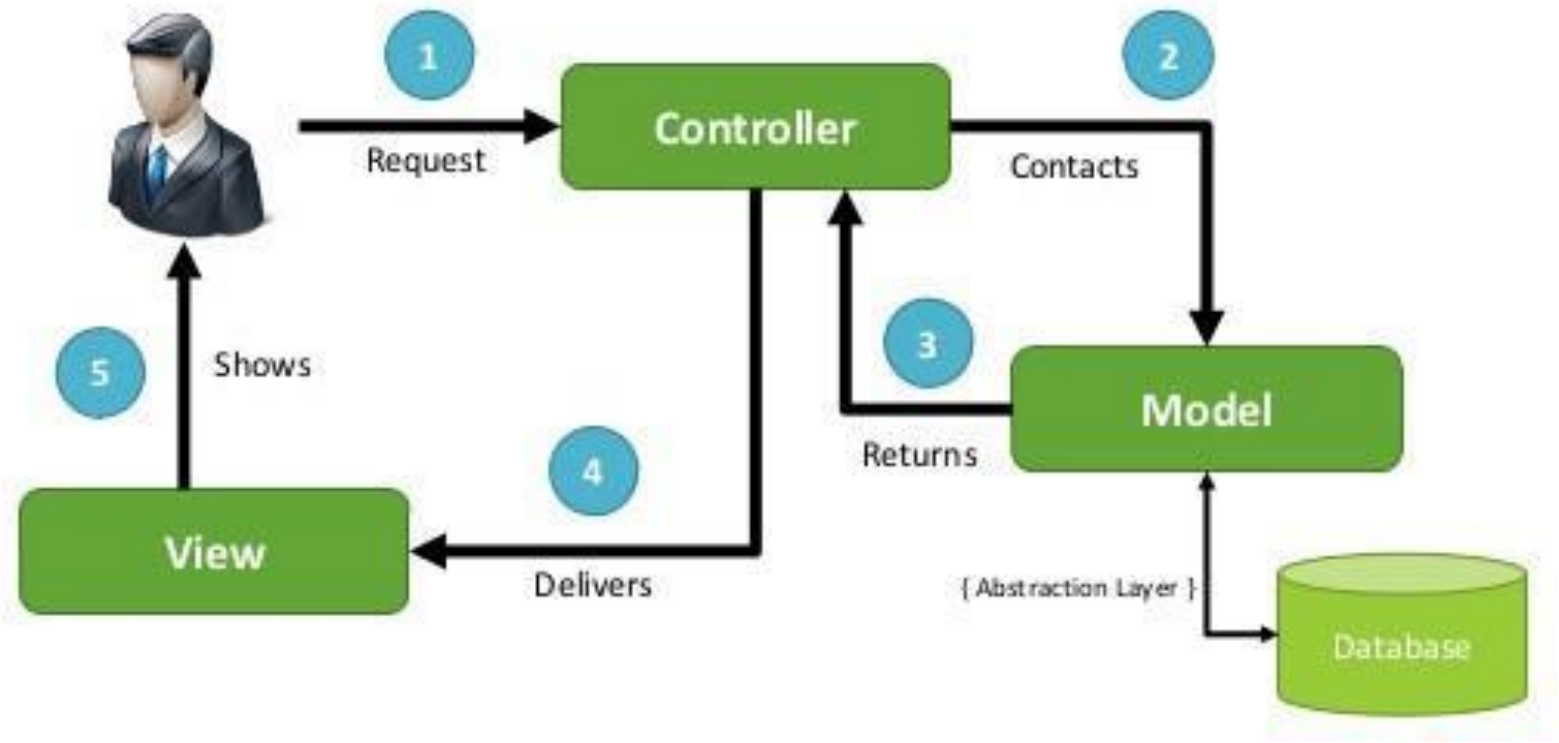
- AI, Data science and Deep Learning
- Interpreter programing language

Class in Python

```
class Student:

    def __init__(self, _name, _age):
        self.name = _name
        self.age = _age

    def setName(self, _name):
        self.name = _name
```

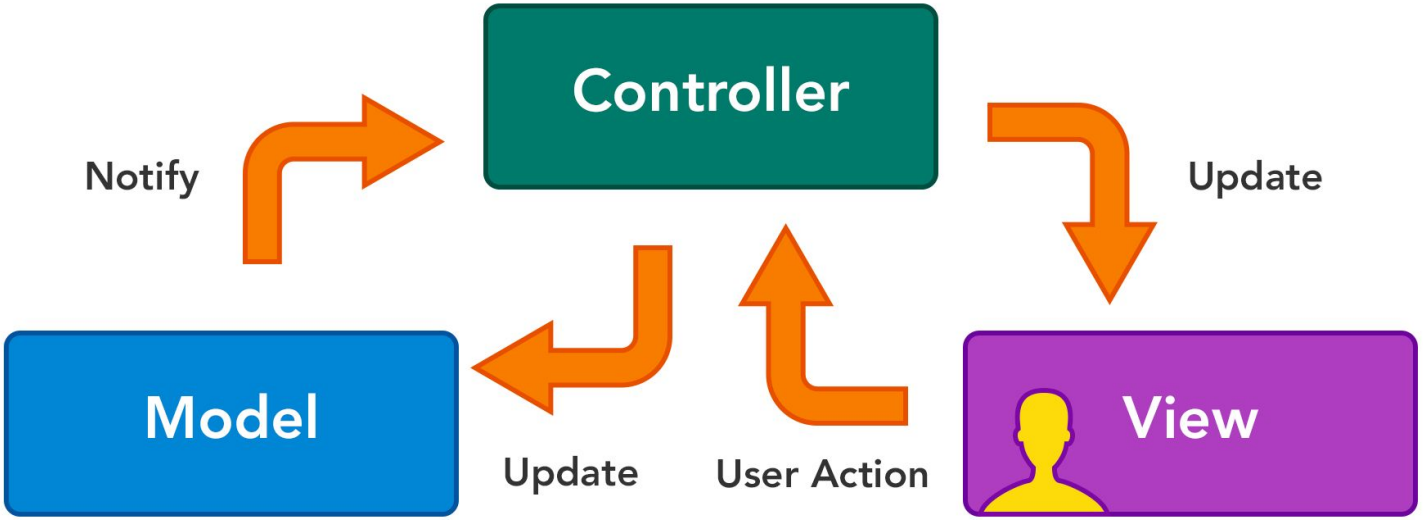
- Flexible and high level approach

Model View Controller



- A pattern for OOP program

MVC using Python



Step 1: Create a new project in Python

- Create Model folder:
  - Student.py
- Create Controller folder:
  - StudentController.py

- The main.py is the View of the project

Step 2: Implement the Student Class

```
class Student:

    def __init__(self,
_name, _age):
        self.name = _name
        self.age = _age

    def setName(self,
_name):
        self.name = _name
    def getName(self):
        return self.name

    def setAge(self, _age):
        self.age = _age
    def getAge(self):
        return self.age
```

- The fields are **highly related to the database properties**

Step 3: Implement the Controller

```
class StudentController:

    def insertStudent(self, student):
        print(student.name, student.age)
```

- Data manipulation: insert, update or delete
- Data storage: database or file management system