pragma solidity 0.6.12;

contract Basics {

//integer, string, bool, bytes

//arrays, enum, struct, mapping (dictionary in python)

int public a=4; // default =0

int public temp1=10;

int public temp2=20;

address public owner;

int public sum;

int8 public b=10;

int16 public c=20;

//unsigned integers

//uint8 = 256 - 0 to 255

uint8 public d=30;

uint public e=19;

//address

address payable public investor;

//string

string public name="Raghu"; //default - ""

//bool

bool public tf = true; //default - false

//bytes

bytes public byteSample = "abc"; //0x616263

bytes public byteSample2 = "ABC"; //0x414243

//arrays - combination of same type of data

string[] public students=["Raghu","John"]; //dynamic array

string[15] public students2; //fixed length arrays

int[] public sampleint=[1,2,3,4];

// push, pop, length - are possible in solidity also

//mapping

mapping(string=>bool) public map;

uint public msgValue;

//enum - Enumeration - save storage, confine options for users

enum Cities {LosAngeles, California, SanJose, Toronto}

Cities public c1 = Cities.LosAngeles;

Cities public c2 = Cities.SanJose;

//struct

struct Student {

string name;

bool attendance;

int rollNo;

int marks;

}

Student public student1;

Student public student2;

Student public student3;

//Functions

function setSum(int \_a, int \_b) public { //setter function

sum = \_a + \_b;

}

//getter function

function getSum() public view returns(int16 balance,int marks) {

//wants you to be specific

return (c,a);

}

function getSum2() public pure returns(int16 balance,bool attendance) {

//wants you to be specific

return (20,true); //this does not set value nor read the value

}

function setString(string memory \_name) public {

name = \_name;

}

function getString() public view returns(string memory studentName) {

return name;

}

function setVariables() public {

students.push("Alice");

// students[3]="Bob";

students2[0]="Raghu";

map["Barrack Obama"] = true;

student1.name = "Raghu";

student1.attendance = true;

student1.rollNo = 34;

student1.marks = 25;

// global variables vs local variables

Student memory tempStudent = Student ({

name: "Raghu",

attendance: true,

rollNo : 25,

marks:99

});

student2 = tempStudent;

tempStudent = Student ("John",true,36,80);

student3 = tempStudent;

}

function swap() public {

int swapTemp = temp2; // no need to pay gas fee for a local variable

temp2 = temp1;

temp1 = swapTemp;

}

// payable function

function payMoney() payable public {

//do something

}

function getbalance() public view returns(uint) {

return address(this).balance ;

}

function sendMoney() public {

investor = 0xE7feb7939C198BE7E770b803aD1f9B6971466F98;

investor.transfer(2 ether);

address payable investor2 = 0x4B87b054D9095650c5233CB0d3902615b99C8085;

investor2.transfer(1.5 ether);

}

function getValues() public view returns(uint blockNumber,

uint blockTimestamp,

address msgSender,

bytes calldata msgData) {

// block number - uint

// block timestamp - uint

// msg sender - address

// msg data - bytes

return (block.number,block.timestamp,msg.sender,msg.data);

}

function payToContract() payable public {

msgValue = msg.value;

}

}