## INDEX

|  |  |  |
| --- | --- | --- |
| **SR NO.** | **PRACTICAL** | **SIGN** |
| 1 | * Write the following programs for Blockchain in Python:   a. A simple client class that generates the private and public keys by using the built-in Python RSA algorithm and testing it. |  |
|  | b. A transaction class to send and receive money and test it. |  |
|  | c. Create multiple transactions and display them. |  |
|  | d. Create a blockchain, a genesis block, and  execute it. |  |
|  | e. Create a mining function and test it. |  |
|  | f. Add blocks to the miner and dump the  blockchain. |  |
| 2 | Install and configure Go Ethereum and the Mist  browser. |  |
| 3 | Implement and demonstrate the use of the following in Solidity:  a. Variable, Operators, Loops, Decision Making, Strings, Arrays, Enums, Structs, Mappings, Conversions, Ether Units, Special Variables. |  |
|  | b. Functions, Function Modifiers, View functions, Pure Functions, Fallback Function, Function Overloading, Mathematical functions, Cryptographic functions. |  |
| 4 | Implement and demonstrate the use of the following in Solidity:  a. Contracts, Inheritance, Interfaces |  |
|  | b. Libraries, Assembly, and Error handling. |  |
| 5 | Install hyperledger fabric and composer. Deploy and execute the application. |  |
| 6 | Create your own blockchain and demonstrate its use. |  |

**PRACTICAL 1**

## 1. Write the following programs for Blockchain in Python:

1. **A simple client class that generates the private and public keys by using the built-in Python RSA algorithm and test it.**

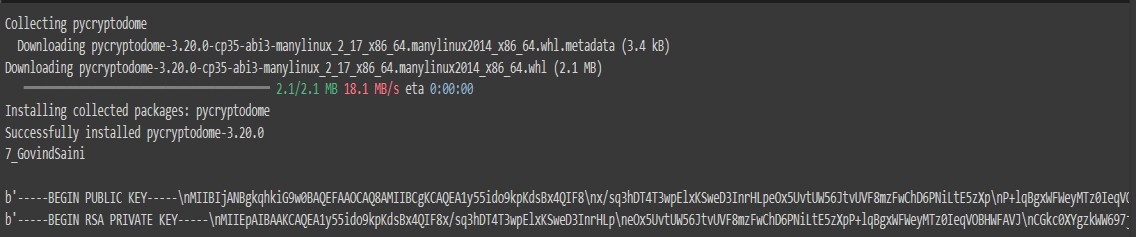
!pip install pycryptodome

from Crypto.PublicKey import RSA key = RSA.generate(2048)

p\_key = key.public\_key().export\_key("PEM") priv\_key = key.export\_key("PEM") print("7\_GovindSaini \n")

print(p\_key) print(priv\_key)

**OUTPUT:**



## A transaction class to send and receive money and test it.

class Bank:

def init (self):

self.balance = 0 print("Saket College")

print("The account is created ") def deposit(self):

amount= float(input("Enter the amount to be deposited:")) self.balance=self.balance + amount

print("The deposit is successful and the balance is in the account is %f" %self.balance ) def withdraw(self):

amount=float(input("Enter the amount to be withdrawn:")) if (self.balance>=amount):

self.balance=self.balance-amount

print("The withdraw is successful and the balance is in the account is %f"

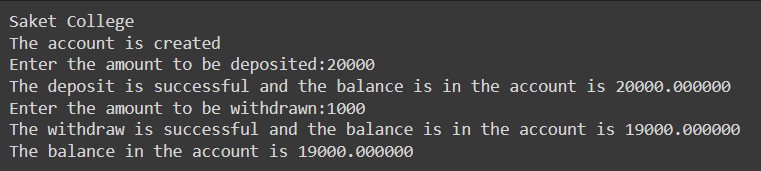
%self.balance) else:

print('Insufficient balance') def enquiry(self):

print("The balance in the account is %f" %self.balance) acc=Bank()

acc.deposit() acc.withdraw() acc.enquiry()

**OUTPUT:**



## Create multiple transactions and display them.

blockchain=[] print("Saket College") def get\_last\_value():

return(blockchain[-1])

def add\_value(sender,reicipent,amount=1.0): transaction={'sender':sender, 'reicipent':reicipent, 'amount':amount} blockchain.append(transaction)

def get\_transaction\_value(): tx\_sender=input('Enter the sender name:')

tx\_reicipent=input(' Enter the recipient of the transaction:') tx\_amount=float(input('Enter the transaction amount:')) return(tx\_sender,tx\_reicipent,tx\_amount)

def print\_block():

for block in blockchain:

print("Saket College\n") print("Here is your block") print(block)

tx\_sender=input('Enter the sender name:') tx\_reicipent=input(' Enter the recipient of the transaction:') tx\_amount=float(input('Enter the transaction amount:')) return tx\_sender,tx\_reicipent,tx\_amount

def print\_block():

for block in blockchain: print("Here is your block") print(block)

again=True

while again==True: tx=get\_transaction\_value() s,r,a=tx

add\_value(s,r,a) print(blockchain)

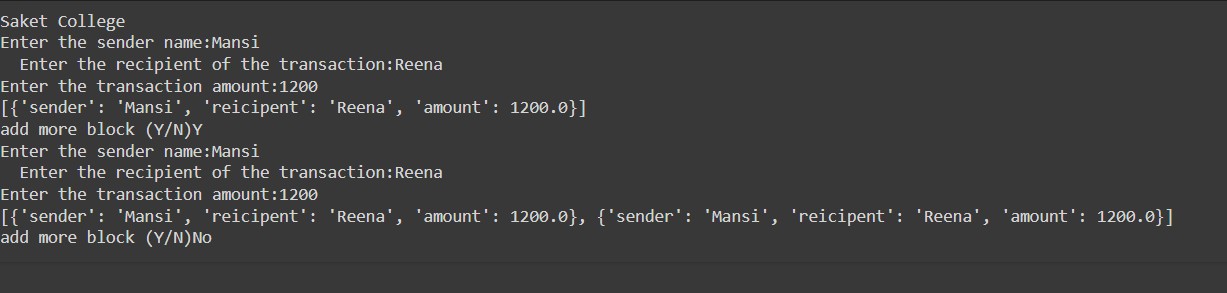
more=input("add more block (Y/N)") if more.lower()=="y":

again=True

else:

again=False

**OUTPUT:**



## Create a blockchain, a genesis block, and execute it.

genesis\_block={

'previous\_hash':'', 'index':0, 'transaction':[], 'nonce':23

}

blockchain=[genesis\_block] def get\_last\_value():

return(blockchain[-1])

def add\_value(sender,reicipent,amount=1.0): transaction={'sender':sender, 'reicipent':reicipent, 'amount':amount} open\_transactions.append(transaction)

def get\_transaction\_value(): tx\_sender=input('Enter the sender name:')

tx\_reicipent=input(' Enter the recipient of the transaction:') tx\_amount=float(input('Enter the transaction amount:')) return(tx\_sender,tx\_reicipent,tx\_amount)

def add\_value(sender,reicipent,amount=1.0): transaction={'sender':sender, 'reicipent':reicipent, 'amount':amount} open\_transactions.append(transaction)

def get\_transaction\_value(): tx\_sender=input('Enter the sender name:')

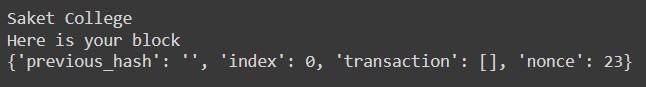
tx\_reicipent=input(' Enter the recipient of the transaction:') tx\_amount=float(input('Enter the transaction amount:')) return tx\_sender,tx\_reicipent,tx\_amount

def get\_user\_choice(): user\_input=input("Enter your choice:") return user\_input

def print\_block():

for block in blockchain: print('Saket College') print("Here is your block")

print(block) print\_block()

**OUTPUT:**

## Create a mining function and test it.

from hashlib import sha256

MAX\_NONCE=100000000

print("Saket College\n") def SHA256(text):

return sha256(text.encode('utf-8')).hexdigest()

def mine(block\_number,transactions,previous\_hash,prefix\_zeros): prefix\_str='0'\*prefix\_zeros

for nonce in range(MAX\_NONCE): text=str(block\_number)+transactions+previous\_hash+str(nonce) new\_hash=SHA256( text)

if new\_hash.startswith(prefix\_str):

print("yay!Successfully mined bitcoins with nonce value:{nonce} ") return new\_hash

return BaseException(f"Couldn't find correct has after trying {MAX\_NONCE} times") if name ==" main ":

transactions='''

Usha ->Karthik->20,

Saroj->Pooja->45 '''

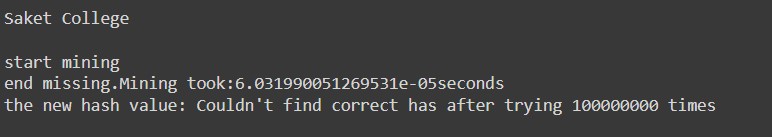
import time difficulty=4 start=time.time() print("start mining")

new\_hash=mine(5,transactions,'000000000xkjfhsdfshfoisafhyuskdhfksduyfh',difficulty) total\_time=str((time.time()-start))

print(f"end missing.Mining took:{total\_time}seconds")

print("the new hash value:",new\_hash)

**OUTPUT:**



## Add blocks to the miner and dump the blockchain.

import datetime import hashlib class Block:

def init (self, data): self.blockNo = 0 self.data = data self.next = None self.hash = None self.nonce = 0

self.previous\_hash = 0x0

self.timestamp = datetime.datetime.now() def calculate\_hash(self):

h = hashlib.sha256() h.update(

str(self.nonce).encode('utf-8') + str(self.data).encode('utf-8') + str(self.previous\_hash).encode('utf-8') + str(self.timestamp).encode('utf-8') + str(self.blockNo).encode('utf-8')

)

return h.hexdigest() def str (self):

return "Block Hash: " + str(self.calculate\_hash()) + "\nBlockNo: " + str(self.blockNo) + "\nBlock Data: " + str(self.data)

class Blockchain: def init (self):

self.diff = 20

self.maxNonce = 2 \*\* 32 self.target = 2 \*\* (256 - self.diff) self.block = Block("Genesis") self.head = self.block self.dummy = self.head

def add(self, block):

block.previous\_hash = self.head.calculate\_hash() block.blockNo = self.head.blockNo + 1 self.head.next = block

self.head = self.head.next def mine(self, block):

for n in range(self.maxNonce):

if int(block.calculate\_hash(), 16) <= self.target: self.add(block)

print(block) break

else:

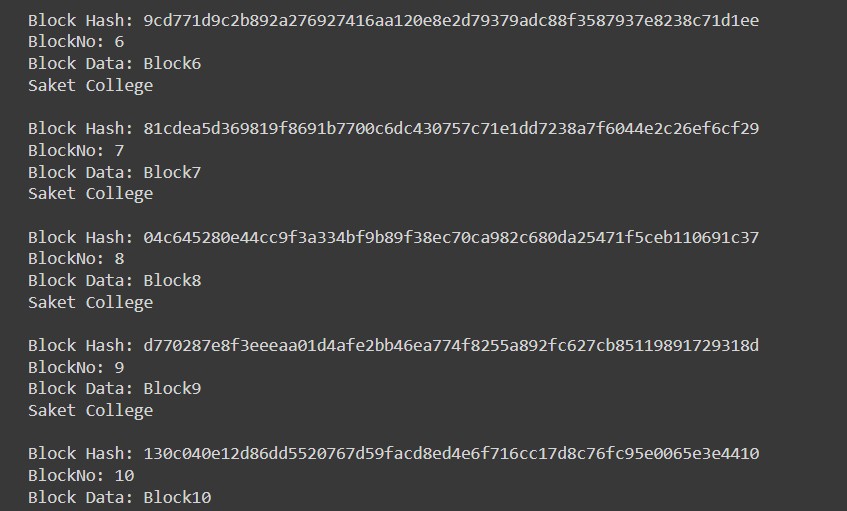
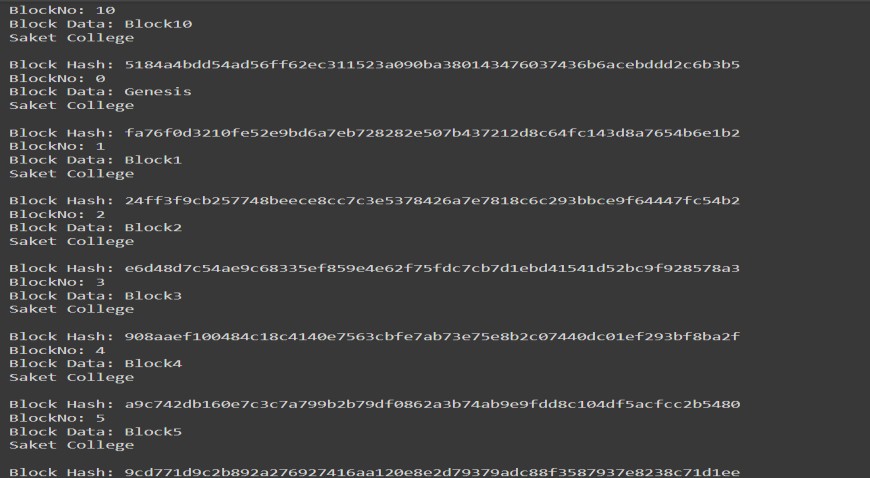
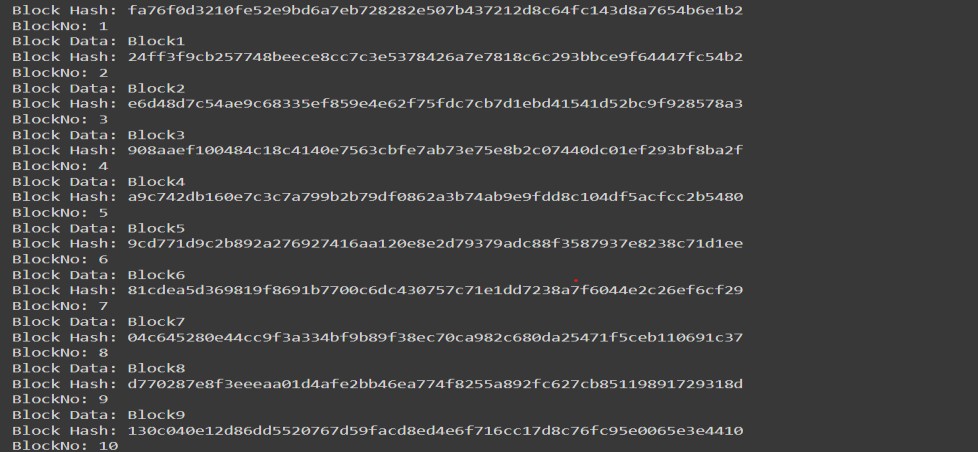
block.nonce += 1 blockchain = Blockchain() for n in range(10):

blockchain.mine(Block("Block" + str(n + 1)))

while blockchain.dummy is not None: print("Saket College\n") print(blockchain.dummy)

blockchain.dummy = blockchain.dummy.next

**OUTPUT:**



## PRACTICAL 2

**Aim: Install and configure Go Ethereum and the Mist browser.**

**Installing GETH (Go Ethereum).**

**Step 1:** Go to the website <https://geth.ethereum.org/downloads/>

**Step 2**: From stable releases Geth 1.5.8 (kind = installer).

**Step 3:** Once downloaded run it then click next.

**Step 4:** Select Geth and Development tools click next.

**Step 5:** Select the location to install click next.

**Step 6:** Once Installation is finished Click Close and it's done.

**Installing Mist Browser**

**Step 1:** <https://github.com/ethereum/mist/releases>

**Step 2:** Under Ethereum Wallet and Mist 0.8.9 –“The wizard” download mist-installer-0-8-9.exe

**Step 3:** For installation click, I agree -> next -> install

**Run Mist**

**Step 1:** Open the Mist from the start menu.

**Step 2:** It will start downloading Blockchain data once you open it.

**Step 3:** Once it finishes downloading it is ready to use.

**Run Geth**

**Step 1:** Open CMD

**Step 2:** Type GETH and press enter

**Step 3:** After it finishes loading press ctrl+c to exit the process.

**Step 4:** Now it’s ready to use

## PRACTICAL 3

**Aim: Implement and demonstrate the use of the following in Solidity.**

## A. Variable, Operators, Loops, Decision Making, Strings, Arrays, Enums, Structs, Mappings, Conversions, Ether Units, Special Variables.

### Variable

pragma solidity ^0.5.0; contract variable\_demo {

uint256 sum = 4; //state variable uint256 x;

address a;

string s = "welcome"; function add(uint256) public {

uint256 y = 2; //local variable sum = sum+x+y: sum = sum + x + y;

}

function display() public view returns (uint256) { return sum;

}

function displayMsg() public view returns (string memory) { return s;

}

}

## OUTPUT:

1. **Operators**

pragma solidity ^0.5.0; contract SolidityTest {

uint16 public a = 20; uint16 public b = 10; uint256 public sum;

uint256 public diff; uint256 public mul; uint256 public div; uint256 public mod; uint256 public dec; uint256 public inc; constructor() public {

sum = a + b; diff = a - b; mul = a \* b; div = a / b; mod = a % b;

dec = b - 1; // Decrement operation inc = a + 1; // Increment operation

}

}

## OUTPUT:

1. **LOOP** pragma solidity ^0.5.0; contract LoopDemo {

uint[] public data;

function forDemo() public returns (uint[] memory) { for (uint i = 0; i < 10; i++) {

data.push(i)

}

return data;

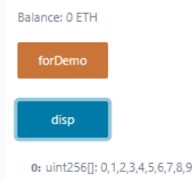
}

function disp() public view returns (uint[] memory) { return data;

}

}

## OUTPUT:



1. **DECISION MAKING**

pragma solidity ^0.5.0; contract IfElseDemo {

uint i = 10;

function decisionMaking() public view returns (string memory) { if (i % 2 == 0) {

return "Even";

} else {

return "Odd";

}

}

}

## OUTPUT:

1. **STRINGS**

pragma solidity ^0.5.0; contract LearningStrings {

string public text; function getText()

public view returns (string memory) {

return text;

}

function setText() public { text = "hello";

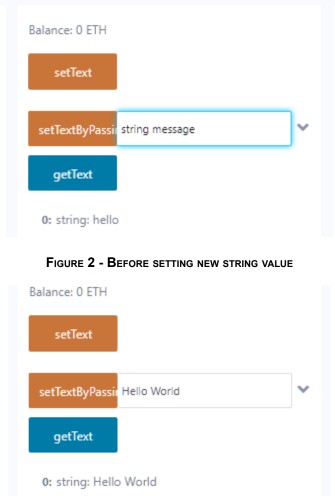
}

function setTextByPassing(string memory message) public { text = message;

}

}

## OUTPUT:



1. **ARRAYS**

pragma solidity ^0.5.0; contract ArrayDemo {

// Static Array

uint[6] public arr2 = [10, 20, 30, 0, 0, 0];

function dispStaticArray() public view returns (uint[6] memory) { return arr2;

}

// Dynamic Array uint public x = 5; uint[] public arr1;

function arrayDemo() public { while (x > 0) {

arr1.push(x); x = x - 1;

}

}

function dispDynamicArray() public view returns (uint[] memory) { return arr1;

}

}

## OUTPUT:

1. **ENUMS** pragma solidity ^0.5.0; contract EnumDemo {

// Define an enum for week days enum WeekDays {

Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday

}

WeekDays public week; WeekDays public choice;

WeekDays public constant default\_value = WeekDays.Sunday; function setValue() public {

choice = WeekDays.Tuesday;

}

function getChoice() public view returns (WeekDays) { return choice;

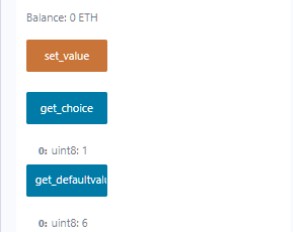
}

function getDefaultValue() public view returns (WeekDays) { return default\_value;

}

}

## OUTPUT:



1. **Structs**

pragma solidity ^0.5.0; contract StructDemo {

struct Book { string name; string author; uint256 id;

bool availability;

}

Book public book2;

Book public book1 = Book("A Little Life", "Hanya Yanagihara", 2, false); function setDetails() public {

book2 = Book("Almond", "Sohn Won-pyung", 1, true);

}

function bookInfo() public view returns (

string memory, string memory, uint256,

bool

)

{

return (book1.name, book1.author, book1.id, book1.availability);

}

function getDetails() public view returns (

string memory, string memory, uint256,

bool

)

{

return (book2.name, book2.author, book2.id, book2.availability);

}

}

## OUTPUT:

1. **Mappings**

pragma solidity ^0.5.0; contract LedgerBalance {

mapping(address => uint256) public balances; function updateBalance(uint256 newBalance) public {

balances[msg.sender] = newBalance;

}

}

contract Updater { LedgerBalance ledgerBalance;

// Set the address of the deployed LedgerBalance contract constructor(address \_ledgerBalanceAddress) public {

ledgerBalance = LedgerBalance(\_ledgerBalanceAddress);

}

function updateBalance(uint256 newBalance) public { ledgerBalance.updateBalance(newBalance);

}

function getBalance() public view returns (uint256) { return ledgerBalance.balances(msg.sender);

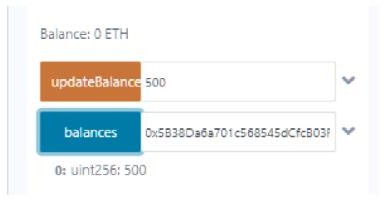
}

}

## OUTPUT:



Before Updating Balance



**After updating Balance**

## CONVERSIONS

pragma solidity ^0.8.0; contract ExplicitConversion {

function convert() public pure returns (bytes memory) { string memory str = "Hello World";

bytes memory b = bytes(str); return b;

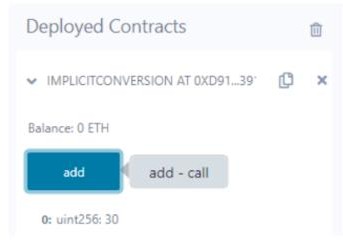
}

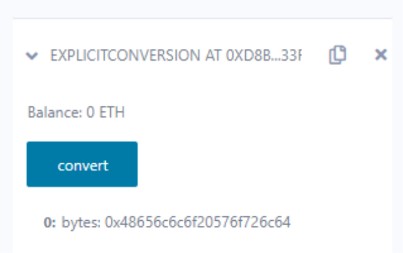
}

**Step 1:** Deploy both contracts



**Step 2**:Open Implicit Conversion and click on add button to sum and display Value



**Step 3**: Open Explicit Conversion and click on convert button

1. **Ether Units** pragma solidity ^0.8.0; contract SolidityTest {

function convertAmountToWei(uint256 amount) public pure returns (uint256)

{

return amount \* 1;

}

function convertAmountToEther(uint256 amount) public pure returns (uint256)

{

return amount \* 1 ether;

}

function convertAmountToGwei(uint256 amount) public pure returns (uint256)

{

return amount \* 1 gwei;

}

function convertSecondsToMins(uint256 seconds) public pure returns (uint256)

{

return seconds / 60;

}

function convertSecondsToHours(uint256 seconds) public pure returns (uint256)

{

return seconds / 3600;

}

function convertMinsToSeconds(uint256 mins) public pure returns (uint256)

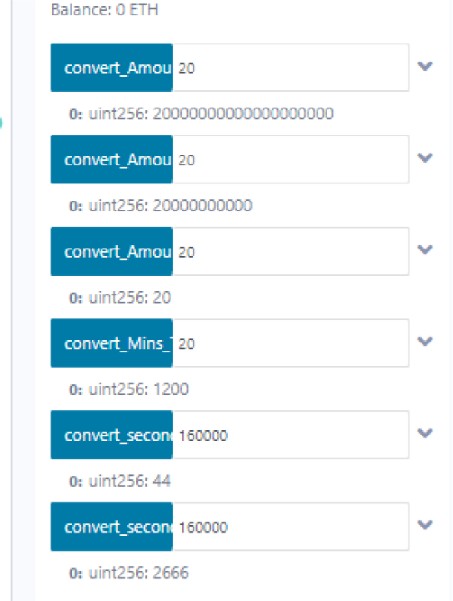
{

return mins \* 60;

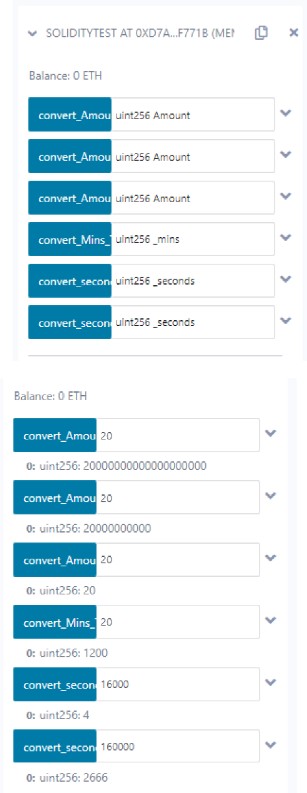
}

}

## OUTPUT:



**Step 1 :** Provide values to each function and click on them



## Special Variables.

pragma solidity ^0.8.0; contract SpecialVariables {

mapping(address => uint256) private rollNo; function setRollNo(uint256 \_myNumber) public {

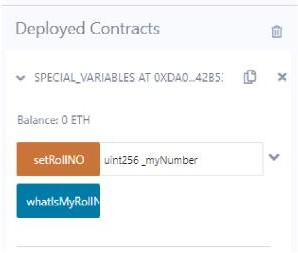
rollNo[msg.sender] = \_myNumber;

}

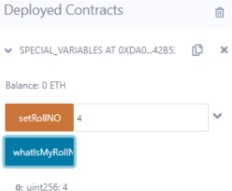
function whatIsMyRollNumber() public view returns (uint256) { return rollNo[msg.sender];

}

}

**Step 1:**Deploy contract Special Variables

**Step 2:** Input a number for setRollNO function and click on it & whatIsMyRollNumber button



## b.Functions, Function Modifiers, View functions, Pure Functions, Fallback Function, Function Overloading, Mathematical functions, Cryptographic functions.

1. **Functions**

pragma solidity >=0.4.22 <0.9.0; contract Test {

function return\_example() public

pure returns (

uint256 sum, uint256 prod, uint256 diff,

string memory message

)

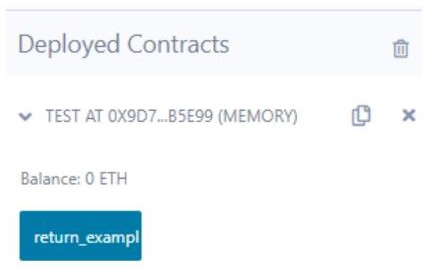
{

uint256 num1 = 10; uint256 num2 = 16; sum = num1 + num2; prod = num1 \* num2; diff = num2 - num1;

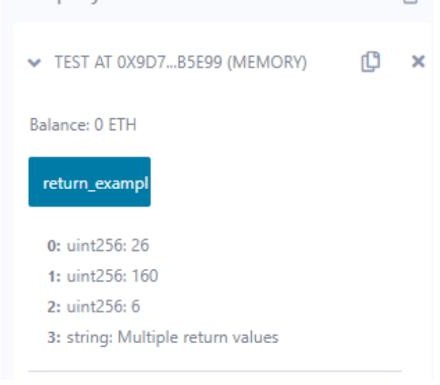
message = "Multiple return values"; return (sum, prod, diff, message);

}

}

**Step 1:** Deploy Test Contract

**Step 2:** Click on the return\_example button to display all values



1. **Function Modifiers** pragma solidity ^0.5.0; contract ExampleContract {

address public owner; uint256 public counter;

// Constructor to initialize the owner constructor() public {

owner = msg.sender;

}

modifier onlyOwner() {

require(msg.sender == owner, "Only the contract owner can call");

\_;

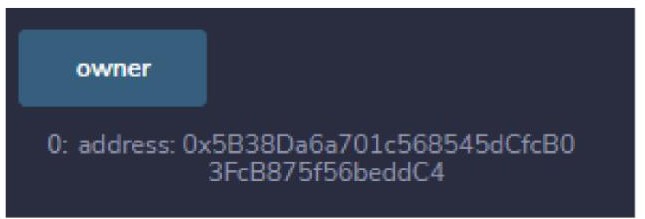
}

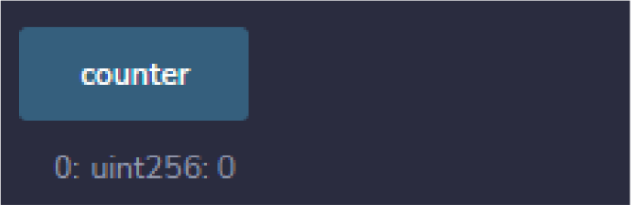
function incrementCounter() public onlyOwner { counter++;

}

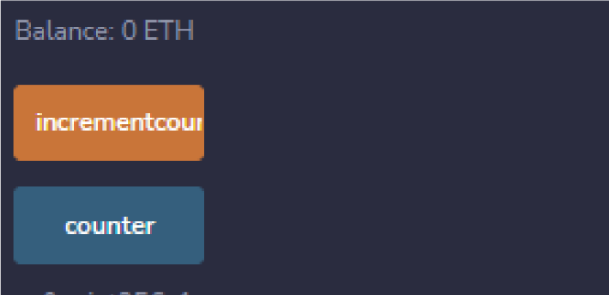
}

**Step 1:** Click on owner button



**Step 2->** Click on counter button initially it is 0.

**Step 3->** Then click on the increment counter button and again click on the counter button, the counter has been increased



## View Functions

pragma solidity ^0.5.0; contract ViewDemo { uint256 num1 = 2; uint256 num2 = 4;

function getResult() public view returns (uint256 product, uint256 sum) { product = num1 \* num2;

sum = num1 + num2;

}

}

## OUTPUT:

1. **Pure Functions**

pragma solidity ^0.5.0; contract PureDemo {

function getResult() public pure returns (uint256 product, uint256 sum) { uint256 num1 = 2;

uint256 num2 = 4; product = num1 \* num2; sum = num1 + num2;

}

}

## OUTPUT:



1. **Fallback Function**

pragma solidity ^0.5.12; contract A {

uint256 public n;

function set(uint256 value) external { n = value;

}

function() external payable { n = 0;

}

}

contract Example {

function callA(A a) public returns (bool) {

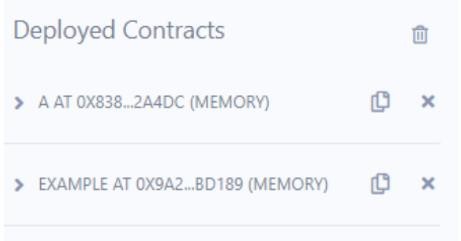
(bool success, ) = address(a).call(abi.encodeWithSignature("set(uint256)", 42)); require(success);

address payable payableA = address(uint160(address(a))); return (payableA.send(2 ether));

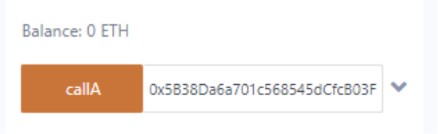
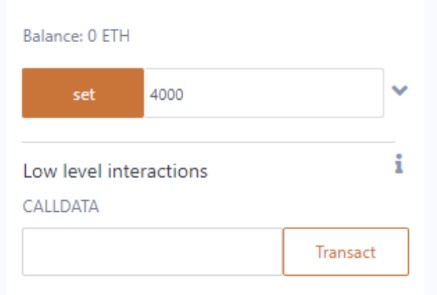
}

}

**Step 1:**Deploy both A & example contracts



**Step 2:** Provide values to both deployed contracts accordingly(use any address)



## Function Overloading

pragma solidity ^0.8.0;

contract OverloadingExample {

function add(uint256 a, uint256 b) public pure returns (uint256) { return a + b;

}

function add(string memory a, string memory b) public pure returns (string memory) { return string(abi.encodePacked(a, b));

}

}

**Step 1:**Deploy Overloading Example contract



**Step 2:** Give integer and string values to both add functions as below



## Mathematical functions

pragma solidity ^0.5.0; contract Test {

function CallAddMod() public pure returns (uint) { return addmod(7, 3, 3);

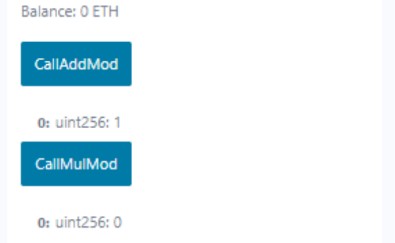
}

function CallMulMod() public pure returns (uint) { return mulmod(7, 3, 3);

}

}

## OUTPUT:



1. **Cryptographic functions.**

pragma solidity ^0.5.0; contract Test {

function callKeccak256() public pure returns (bytes32 result) { return keccak256(abi.encodePacked("BLOCKCHAIN"));

}

function callSha256() public pure returns (bytes32 result) { return sha256(abi.encodePacked("BLOCKCHAIN"));

}

function callRipemd160() public pure returns (bytes20 result) { return ripemd160(abi.encodePacked("BLOCKCHAIN"));

}

}

## OUTPUT:



**PRACTICAL 4**

## Implement and demonstrate the use of the following in Solidity: Contracts, Inheritance, Interfaces

1. **Contracts**

pragma solidity ^0.5.0; contract ContractDemo {

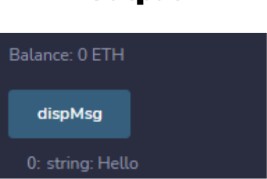
string public message = "Hello";

function dispMsg() public view returns (string memory) { return message;

}

}

**OUTPUT:**



## Inheritance

pragma solidity >=0.4.22 <0.6.0; contract Parent {

uint256 internal sum; function setValue() external {

uint256 a = 10; uint256 b = 20; sum = a + b;

}

}

contract Child is Parent {

function getValue() external view returns (uint256) { return sum;

}

}

contract Caller {

Child cc = new Child();

function testInheritance() public returns (uint256) { cc.setValue();

return cc.getValue();

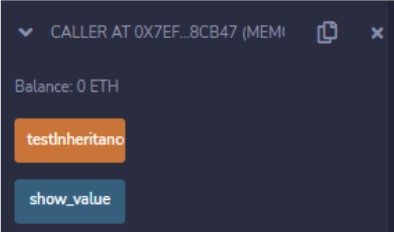
}

function showValue() public view returns (uint256) { return cc.getValue();

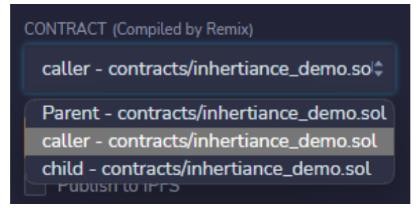
}

}

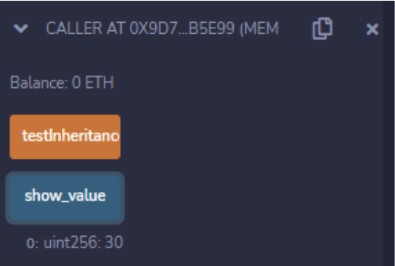
## OUTPUT:



### Flow of execution

**Step 1:**Select caller contract to deploy in Contract and deploy.

**Step 2:** Click test Inheritance and then click on show\_value to view the value.



1. **Interfaces** pragma solidity ^0.5.0; interface Calculator {

function getResult() external view returns (uint);

}

contract Test is Calculator { constructor() public {}

function getResult() external view returns (uint) { uint a = 1;

uint b = 2;

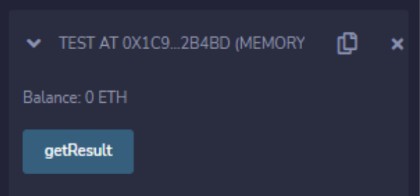
uint result = a + b; return result;

}

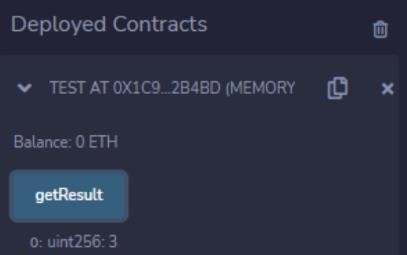
}

## OUTPUT:

Flow of execution:



Click on getResult to display the sum



## Libraries, Assembly, and Error Handling.

1. **Libraries**

### myLib.sol Code

pragma solidity >=0.7.0 <0.9.0; library MyMathLib {

function sum(uint256 a, uint256 b) external pure returns (uint256) { return a + b;

}

function exponent(uint256 a, uint256 b) external pure returns (uint256) { return a\*\*b;

}

}

### using\_library.sol Code

pragma solidity >=0.7.0 <0.9.0;

import "./myLIB.sol"; // Adjust the path according to your project structure contract UseLib {

function getsum(uint256 x, uint256 y) public pure returns (uint256) { return MyMathLib.sum(x, y);

}

function getexponent(uint256 x, uint256 y) public pure returns (uint256) { return MyMathLib.exponent(x, y);

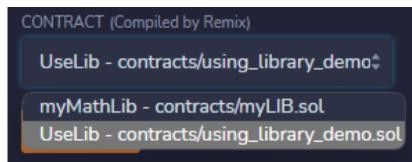
}

}

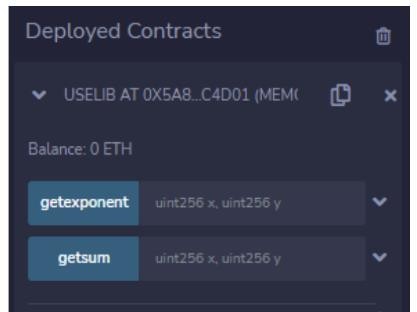
## OUTPUT:

### Flow of execution

**Step 1:**Change contract to UseLib and deploy.



**Step 2 :**The deployed contract should be same as below.



**Step 3:**Input values to both getexponent and getsum functions as below



**Step 4:** Execute both functions. You will get below output



## Assembly

pragma solidity >=0.4.16 <0.9.0; contract InlineAssembly {

// Defining function

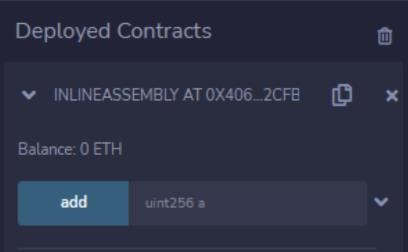
function add(uint256 a) public pure returns (uint256 b) { assembly {

// Perform addition let c := add(a, 16)

// Set value of b as c + 12 b := add(c, 12)

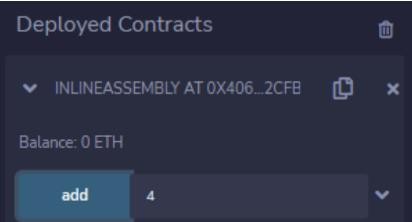
}

## OUTPUT:



### Flow of execution

**Step 1:**Input a number for add function



**Step 2:** Click add to output sum



## Error Handling

pragma solidity ^0.5.17; contract ErrorDemo {

function getSum(uint256 a, uint256 b) public pure returns (uint256) { uint256 sum = a + b;

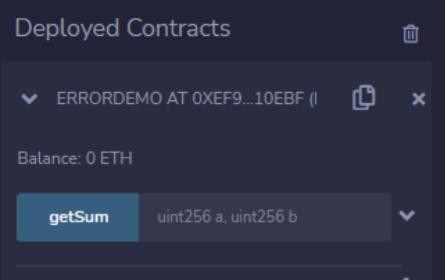
// Using require for input validation

require(sum < 255, "Sum exceeds maximum value of 254");

// Using assert to check internal invariants assert(sum < 255);

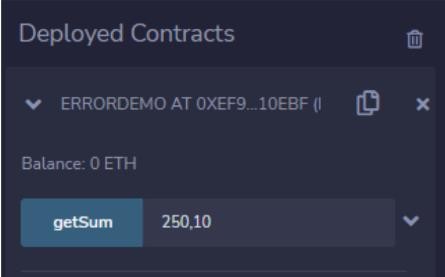
return sum;

## OUTPUT:

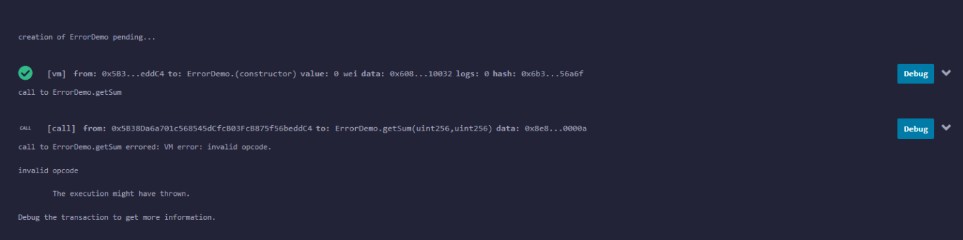


### Flow of execution

**Step 1:**Provide some values and press on getSum



**Step 2:**Check terminal panel



## PRACTICAL 5

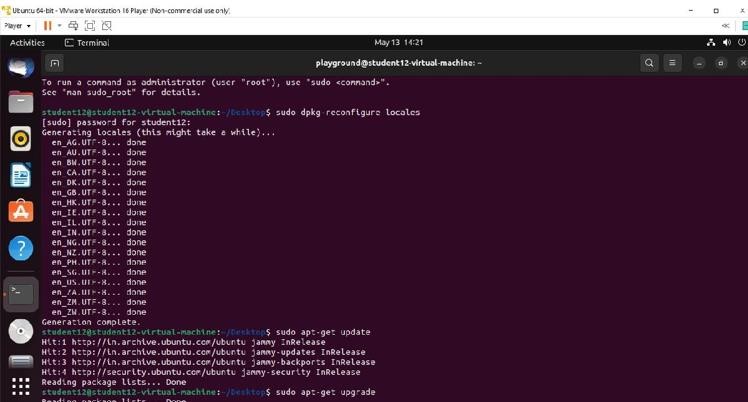
**Aim:** Install hyperledger fabric and composer. Deploy and execute the application.

1. Download [VMware Player,](https://www.vmware.com/products/workstation-player.html) you can use [virtualbox](https://www.virtualbox.org/wiki/Downloads) as an alternate.
2. Download [Ubuntu ISO](https://www.ubuntu.com/download/desktop)
3. Install vmware player
4. Create VM of Ubuntu using vmware player

### Prepare the VM

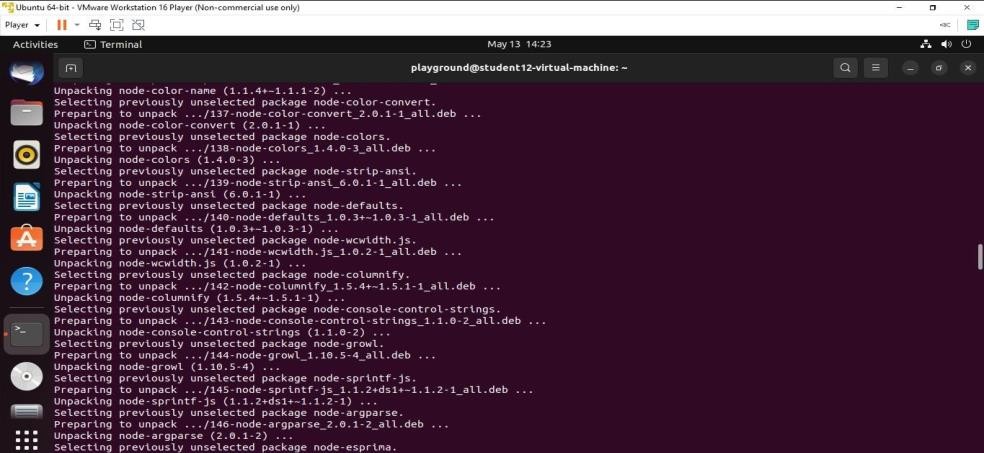
$ sudo dpkg-reconfigure locales // choose en\_US.UTF -8 if in doubt

$ sudo apt-get update

$ sudo apt-get upgrade

### Install pre-requists

$ sudo apt-get install curl git docker.io docker-compose golang nodejs npm



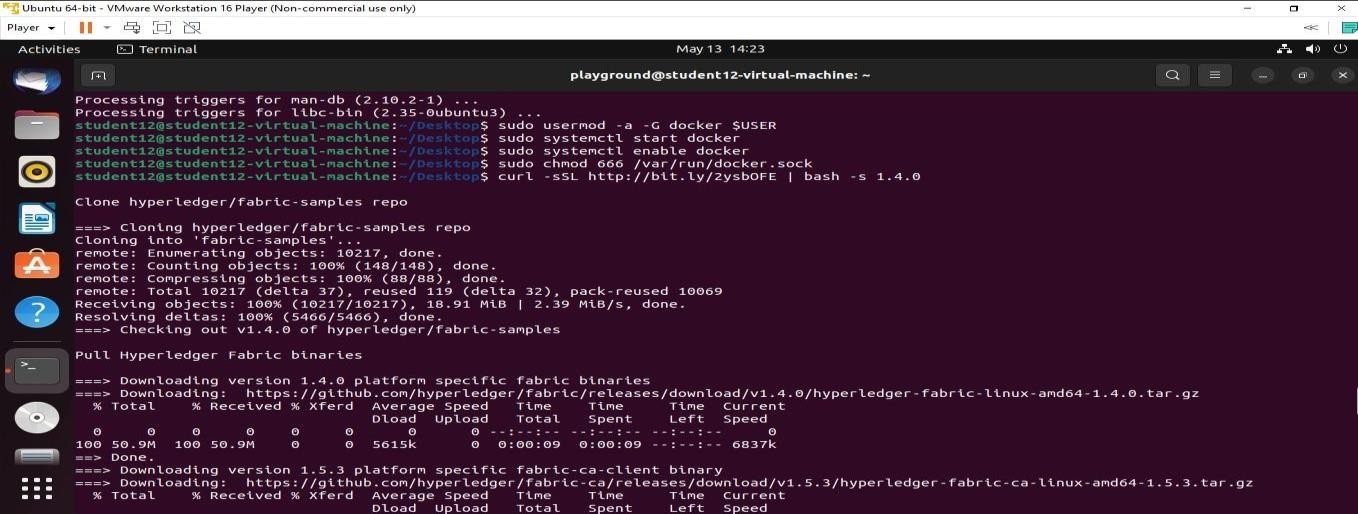
### Install Docker

$ sudo usermod -a -G docker $USER

$ sudo systemctl start docker

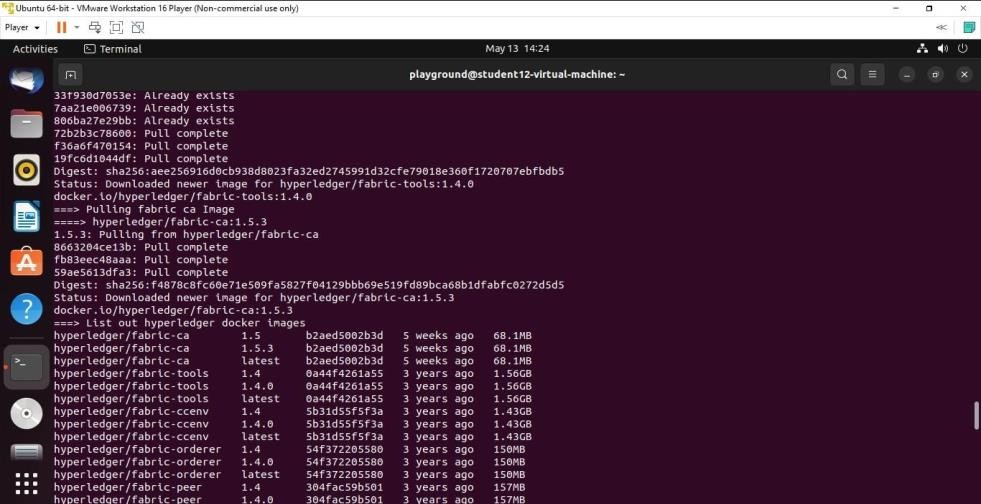
$ sudo systemctl enable docker

$ sudo chmod 666 /var/run/docker.sock



### Install Hyperledger Fabric

1. Check the latest version of [fabric repository,](https://github.com/hyperledger/fabric) at the time of writting this post, it is 1.4 2. Install Fabric



$ curl -sSL <http://bit.ly/2ysbOFE>|

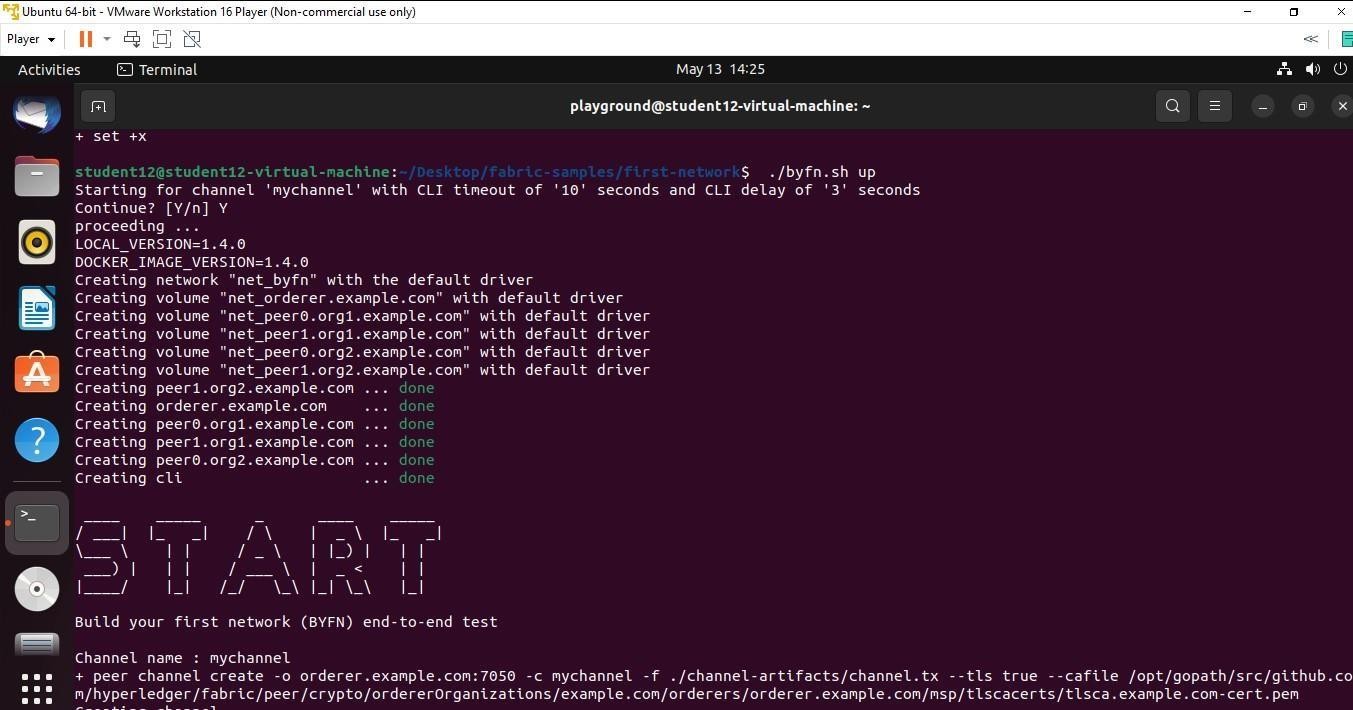
1. Check if fabric is installed, you should see big "END" once done

$ cd fabric-samples/first-network

$ ./byfn.sh generate

$ ./byfn.sh up





### Check if fabric docker is running smoothly

1. **Stop the network**

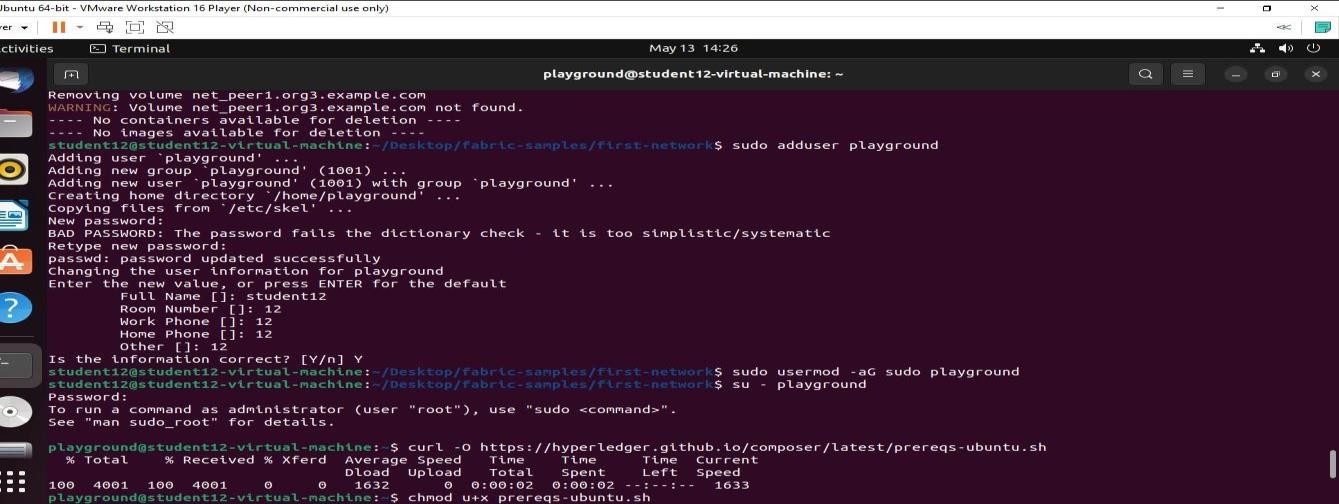
$ ./byfn.sh down

### Install Composer

1. Create new user, when asked about the full name, use something different than the full name used of the main user, to avoid confusion the next time you log on.

Note: if you need to keep things with the same user, jump to step number 4 followed by step number 7 directly

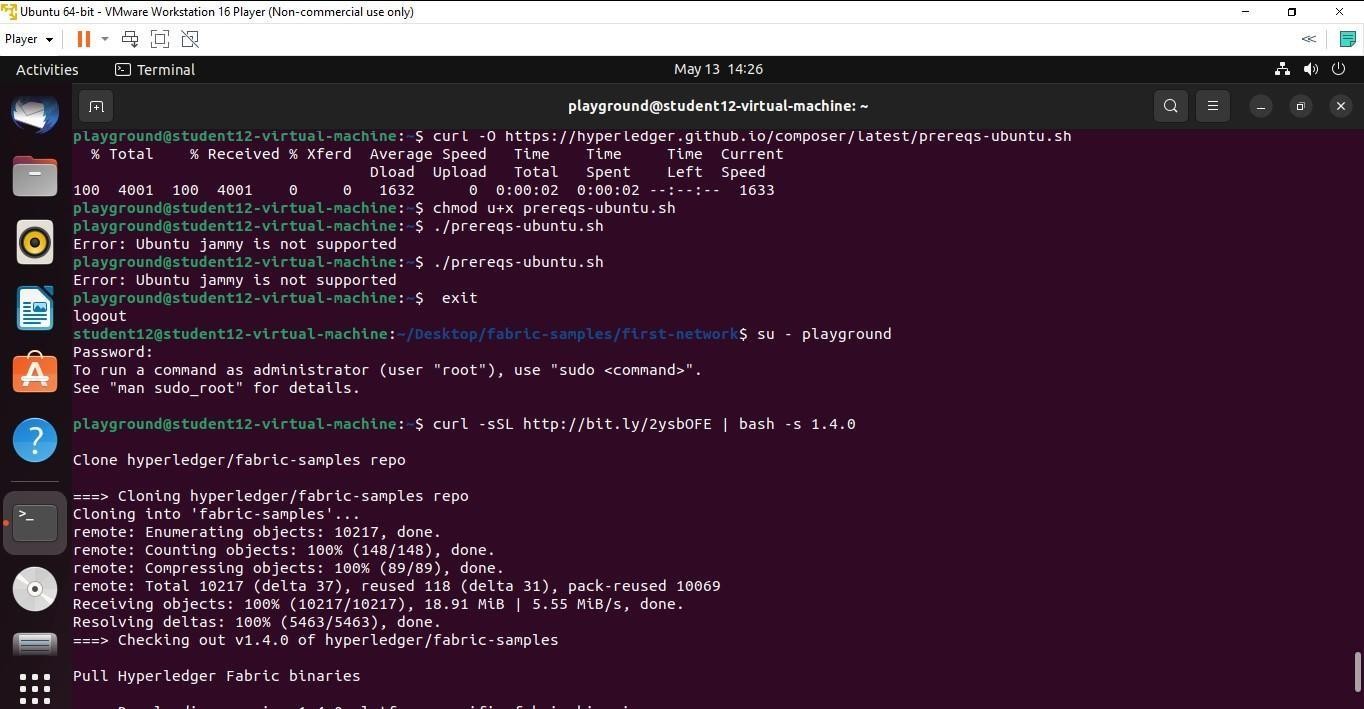
$ sudo adduser playground



### Set permission for the new user sudo usermod -aG sudo playground

1. **Login as the new user**

### su - playground



1. **Install the prerequisites by getting and running the script from github. It will ask for the password of “playground” account to proceed.**

$ curl -O https://hyperledger.github.io/composer/latest/prereqs-ubuntu.sh

$ chmod u+x prereqs-ubuntu.sh

$ ./prereqs-ubuntu.sh

### Logout and login with the new user to get things activated properly

$ exit

$ su - playground

### Install components needed for running Hyperledger Fabric

$ curl -sSL <http://bit.ly/2ysbOFE>| bash -s 1.4.0

### Install components needed for running Hyperledger Composer

$ npm install -g composer-cli composer-rest-server generator-hyperledger- composer yo composerplayground

### Start Composer

$ composer-playground

## PRACTICAL 6

**Create your own blockchain and demonstrate its use. Code:**

import time import hashlib

class Block(object):

def init (self, index, proof\_number, previous\_hash, data, timestamp=None): self.index = index

self.proof\_number = proof\_number self.previous\_hash = previous\_hash self.data = data

self.timestamp = timestamp or time.time() @property

def compute\_hash(self):

string\_block = "{}{}{}{}{}".format(

self.index, self.proof\_number, self.previous\_hash, self.data, self.timestamp) return hashlib.sha256(string\_block.encode()).hexdigest()

def repr (self):

return "{} - {} - {} - {} - {}".format(

self.index, self.proof\_number, self.previous\_hash, self.data, self.timestamp) class BlockChain(object):

def init (self): self.chain = [] self.current\_data = [] self.nodes = set() self.build\_genesis()

def build\_genesis(self): self.build\_block(proof\_number=0, previous\_hash='0')

def build\_block(self, proof\_number, previous\_hash): block = Block(

index=len(self.chain), proof\_number=proof\_number, previous\_hash=previous\_hash, data=self.current\_data

)

self.current\_data = [] self.chain.append(block) return block

@staticmethod

def confirm\_validity(block, previous\_block): if previous\_block.index + 1 != block.index:

return False

elif previous\_block.compute\_hash != block.previous\_hash: return False

elif block.timestamp <= previous\_block.timestamp: return False

return True

def get\_data(self, sender, receiver, amount): self.current\_data.append({

'sender': sender, 'receiver': receiver, 'amount': amount

})

return True @staticmethod

def proof\_of\_work(last\_proof): proof = 0

while not BlockChain.valid\_proof(last\_proof, proof): proof += 1

return proof @staticmethod

def valid\_proof(last\_proof, proof):

guess = f'{last\_proof}{proof}'.encode() guess\_hash = hashlib.sha256(guess).hexdigest() return guess\_hash[:4] == "0000"

@property

def latest\_block(self): return self.chain[-1]

def chain\_validity(self): previous\_block = self.chain[0] block\_index = 1

while block\_index < len(self.chain): block = self.chain[block\_index]

if not self.confirm\_validity(block, previous\_block): return False

previous\_block = block block\_index += 1

return True

def block\_mining(self, details\_miner): self.get\_data(

sender="0", # it implies that this node has created a new block receiver=details\_miner,

amount=1,

)

last\_block = self.latest\_block last\_proof\_number = last\_block.proof\_number

proof\_number = self.proof\_of\_work(last\_proof\_number) last\_hash = last\_block.compute\_hash

block = self.build\_block(proof\_number, last\_hash) return vars(block)

def create\_node(self, address): self.nodes.add(address) return True

@staticmethod

def get\_block\_object(block\_data): return Block(

block\_data['index'], block\_data['proof\_number'], block\_data['previous\_hash'], block\_data['data'], timestamp=block\_data['timestamp']

)

blockchain = BlockChain()

print("GET READY MINING ABOUT TO START")

print(blockchain.chain)

last\_block = blockchain.latest\_block last\_proof\_number = last\_block.proof\_number

proof\_number = blockchain.proof\_of\_work(last\_proof\_number) blockchain.get\_data(

sender="0", # this means that this node has constructed another block receiver="sharik",

amount=1 # building a new block (or figuring out the proof number) is rewarded with 1

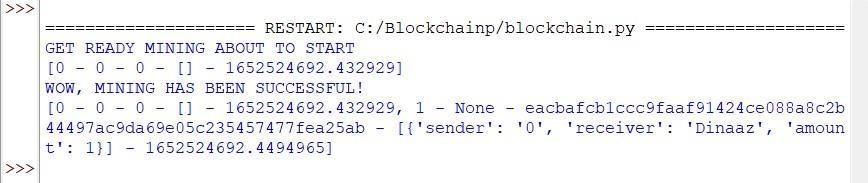
)

last\_hash = last\_block.compute\_hash

block = blockchain.build\_block(proof\_number, last\_hash) print("WOW, MINING HAS BEEN SUCCESSFUL!")

print(blockchain.chain)

**OUTPUT:**



# NATURAL LANGUAGE PROCESSING

# D:\IMP.DATA\Desktop\Shital\Printing Banner,Banners,Certificates\NEW_LOGO-removebg-preview.png

# D:\IMP.DATA\Desktop\Shital\Untitled-1-removebg-preview.png

**SAKET GYANPEETH’S**

**SAKET COLLEGE OF ARTS, SCIENCE AND COMMERCE**

**(Permanently Affiliated to University of Mumbai)**

**NAAC Accredited B Grade**

Saket Vidyanagari Marg, Chinchpada Road, Katemanivali, Kalyan (East) – 421306, Dist. Thane (MAH)

Department of Information Technology

**CERTIFICATE**

**This is to certify that**

**---** of **MSc INFORMATION TECHNOLOGY PART-II** Class has satisfactorily carried out the required practicals in the

subject.

**NATURAL LANGUAGE PROCESSING**

For the Academic year 2023 – 2024

**Practical In-Charge Head of Department External Examiner**

# INDEX

|  |  |  |
| --- | --- | --- |
| SR NO: | PRACTICAL | SIGN |
| 1 | Install NLTKConvert the given text to speechConvert audio file Speech to Text |  |
| 2 | Study of various Corpus – Brown, Inaugural,Reuters, under various methods like fields, raw, words, scents, categories. |  |
| 3 | Study of Wordnet Dictionary with methods as synsets, definitions, examples, antonyms,Study lemmas, hyponyms, hypernyms. |  |
| 4 | Text Tokenization using Python, Regular Expressions |  |
| 5 | Illustrate part of speech tagging. |  |
| 6 | Finite state automata |  |
| 7 | Study Porter Stemmer, Lancaster Stemmer, Regexp Stemmer, Snowball Stemmer, WordNet Lemmatizer |  |
| 8 | Implement Naive Bayes |  |
| 9 | Speech Tagging using spacy,nltkcode,Statistical ,Probabilistic parsersss |  |

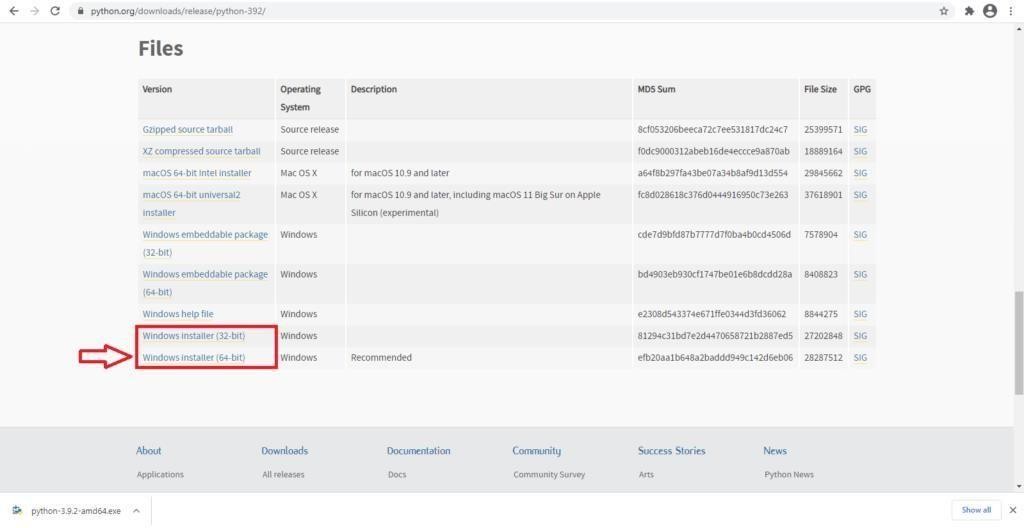
# PRACTICAL NO. 1

### Install NLTK

**Python 3.9.2 Installation on Windows**

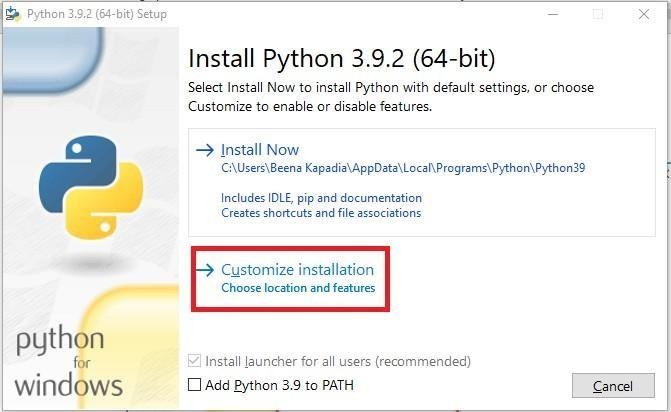
**Step 1)** **Go to link** https:/[/www.p](http://www.python.org/downloads/)y[thon.org/downloads/,](http://www.python.org/downloads/) **and select the latestversion for windows.**

**Note**: If you don't want to download the latest version, you can visit the download tab and see all releases.

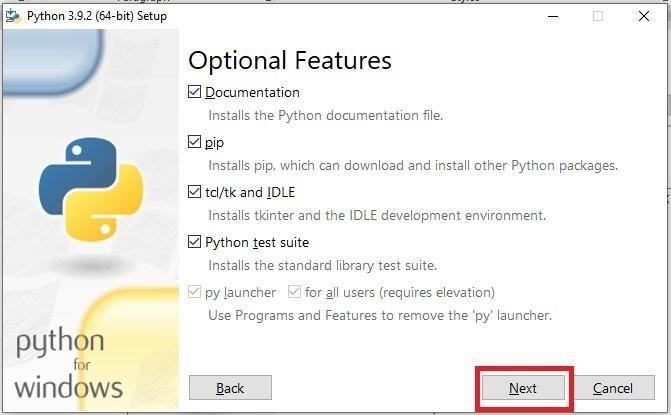


**Step 2)** Click on the Windows installer (64 bit)

**Step 3)** Select Customize Installation

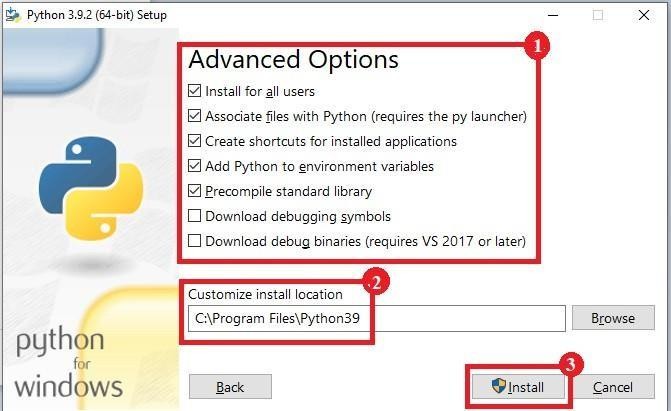


**Step 4)** Click NEXT



**Step 5)** In next screen

* + 1. Select the advanced options
    2. Give a Custom install location. Keep the default folder as c:\Programfiles\Python39
    3. Click Install

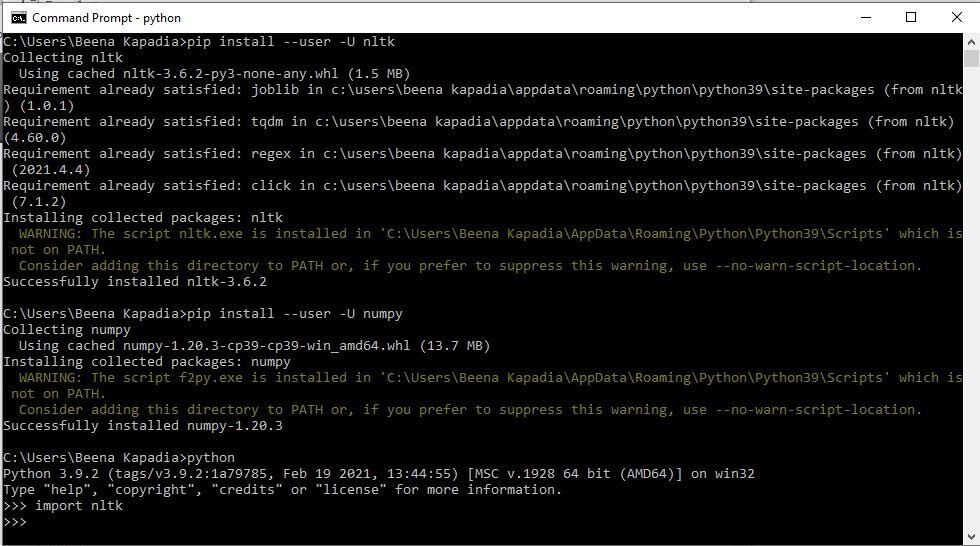


**Step 6)** Click Close button once install is done.

**Step 7) open** command prompt window and run the following commands: C:\Users\Suraj Maurya>pip install --upgrade pip

C:\Users\Suraj Maurya> pip install --user -U nltk C:\Users\SurajMaurya> >pip install --user -U numpy C:\Users\Suraj Maurya>python

>>> import nltk

>>>

* 1. **Convert the given text to speech.Source code:**

# text to speech pip install gtts

# pip install playsound

from playsound import playsound

# import required for text to speech conversion from gtts import gTTS

mytext = "Welcome to Natural Language programming" language = "en"

myobj = gTTS(text=mytext, lang=language, slow=False) myobj.save("myfile.mp3")

playsound("myfile.mp3")

## Output:

welcomeNLP.mp3 audio file is getting created and it plays the file with playsound() method, while running the program.

## Convert audio file Speech to Text.Source code:

Note: required to store the input file "male.wav" in the current folder before running the program.

#pip3 install SpeechRecognition pydub

import speech\_recognition as sr filename = "male.wav"

# initialize the recognizer r = sr.Recognizer()

# open the file

with sr.AudioFile(filename) as source:

# listen for the data (load audio to memory) audio\_data = r.record(source)

# recognize (convert from speech to text) text = r.recognize\_google(audio\_data) print(text)

Input:

male.wav (any wav file)

## Output:

**PRACTICAL NO. 2**

**Aim: Study of various Corpus – Brown, Inaugural, Reuters, under various methods like fields, raw, words, sents, categories.**

1. **Create and use your own corpora (plaintext, categorical)**

## Study Conditional frequency distributions

1. **Study of tagged corpora with methods like tagged\_sents, tagged\_words.**

## Write a program to find the most frequent noun tags.

1. **Map Words to Properties Using Python Dictionaries**

## Study DefaultTagger, Regular expression tagger, UnigramTagger

1. **Find different words from a given plain text without any space by comparing this text with a given corpus of words. Also find the score of words.**

## Study of various Corpus – Brown, Inaugural, Reuters, udhr with various methods like fields, raw, words, sents, categories,

### SOURCE CODE: '''NLTK includes a small selection of texts from the Project

### brown electronic text archive, which contains some 25,000 free electronic books, hosted at [http://www.brown.org/.](http://www.brown.org/) We begin by getting the Python interpreter to load the

### NLTK package, then ask to see nltk.corpus.brown.fileids(), the file identifiers

### in this corpus:'''

import nltk

from nltk.corpus import brown

print ('File ids of brown corpus\n',brown.fileids())

'''Let’s pick out the first of these texts — Emma by Jane Austen — and give it a short name, emma, then find out how many words it contains:'''

ca01 = brown.words('ca01')

# display first few words

print('\nca01 has following words:\n',ca01)

# total number of words in ca01 print('\nca01 has',len(ca01),'words')

#categories or files

print ('\n\nCategories or file in brown corpus:\n') print (brown.categories())

'''display other information about each text, by looping over all the values of fileid corresponding to the brown file identifiers listed earlier and then computing statistics for each text.'''

print ('\n\nStatistics for each text:\n') print('AvgWordLen\tAvgSentenceLen\tno.ofTimesEachWordAppearsOnAvg\t\tFile

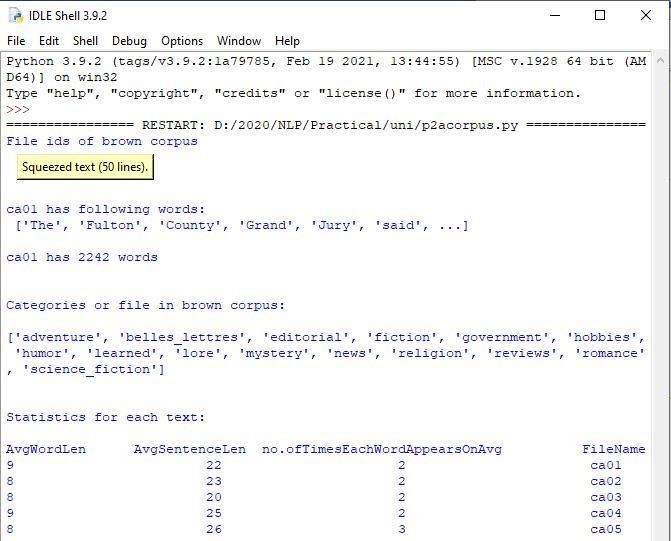
Name') for fileid in brown.fileids():

num\_chars = len(brown.raw(fileid)) num\_words = len(brown.words(fileid)) num\_sents = len(brown.sents(fileid))

num\_vocab = len(set([w.lower() for w in brown.words(fileid)]))

print (int(num\_chars/num\_words),'\t\t\t', int(num\_words/num\_sents),'\t\t\t', int(num\_words/num\_vocab),'\t\t\t', fileid)

## OUTPUT:



### Create and use your own corpora (plaintext, categorical)

### source code:

### '''NLTK includes a small selection of texts from the Project filelist electronic text archive, which contains some 25,000 free electronic books, hosted at [http://www.filelist.org/.](http://www.filelist.org/) We begin by getting the Python interpreter to load the NLTK package, then ask to see nltk.corpus.filelist.fileids(), the file identifiers in this corpus:'''

import nltk

from nltk.corpus import PlaintextCorpusReader

corpus\_root = 'D:/2020/NLP/Practical/uni'

filelist = PlaintextCorpusReader(corpus\_root, '.\*') print ('\n File list: \n')

print (filelist.fileids()) print (filelist.root)

'''display other information about each text, by looping over all the values of fileid corresponding to the filelist file identifiers listed earlier and then computing statistics for each text.'''

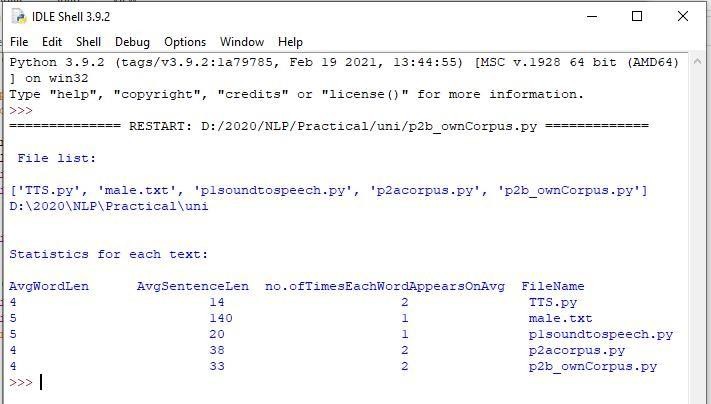
print ('\n\nStatistics for each text:\n') print('AvgWordLen\tAvgSentenceLen\tno.ofTimesEachWordAppearsOnAvg\tFileName')

for fileid in filelist.fileids():

num\_chars = len(filelist.raw(fileid)) num\_words = len(filelist.words(fileid)) num\_sents = len(filelist.sents(fileid))

num\_vocab = len(set([w.lower() for w in filelist.words(fileid)]))

print (int(num\_chars/num\_words),'\t\t\t', int(num\_words/num\_sents),'\t\t\t', int(num\_words/num\_vocab),'\t\t', fileid)

**OUTPUT:**

### Study Conditional frequency distributions source code:

#process a sequence of pairs

text = ['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]

pairs = [('news', 'The'), ('news', 'Fulton'), ('news', 'County'), ...] import nltk

from nltk.corpus import brown fd = nltk.ConditionalFreqDist(

(genre, word)

for genre in brown.categories()

for word in brown.words(categories=genre))

genre\_word = [(genre, word)

for genre in ['news', 'romance']

for word in brown.words(categories=genre)] print(len(genre\_word))

print(genre\_word[:4]) print(genre\_word[-4:])

cfd = nltk.ConditionalFreqDist(genre\_word) print(cfd)

print(cfd.conditions())

print(cfd['news']) print(cfd['romance']) print(list(cfd['romance']))

from nltk.corpus import inaugural cfd = nltk.ConditionalFreqDist(

(target, fileid[:4])

for fileid in inaugural.fileids() for w in inaugural.words(fileid) for target in ['america', 'citizen'] if w.lower().startswith(target))

from nltk.corpus import udhr

languages = ['Chickasaw', 'English', 'German\_Deutsch', 'Greenlandic\_Inuktikut', 'Hungarian\_Magyar', 'Ibibio\_Efik']

cfd =nltk.ConditionalFreqDist ((lang, len(word))

for lang in languages

for word in udhr.words(lang + '-Latin1'))

cfd.tabulate(conditions=['English', 'German\_Deutsch'], samples=range(10), cumulative=True)

## OUTPUT:

* 1. **Study of tagged corpora with methods like tagged\_sents, tagged\_words.**

## Source code:

import nltk

from nltk import tokenize nltk.download('punkt') nltk.download('words')

para = "Hello! My name is Suraj Maurya. Today you'll be learning NLTK." sents = tokenize.sent\_tokenize(para)

print("\nsentence tokenization\n===================\n",sents)

# word tokenization

print("\nword tokenization\n===================\n") for index in range(len(sents)):

words = tokenize.word\_tokenize(sents[index]) print(words)

## OUTPUT:

### Write a program to find the most frequent noun tags.

### Code:

import nltk

from collections import defaultdict

text = nltk.word\_tokenize("Nick likes to play football. Nick does not like to play cricket.")

tagged = nltk.pos\_tag(text) print(tagged)

# checking if it is a noun or not addNounWords = []

count=0

for words in tagged:

val = tagged[count][1]

if(val == 'NN' or val == 'NNS' or val == 'NNPS' or val == 'NNP'): addNounWords.append(tagged[count][0])

count+=1

print (addNounWords) temp = defaultdict(int)

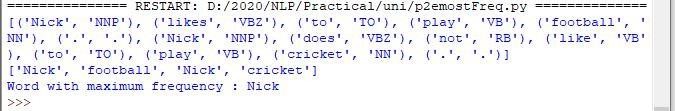
# memoizing count

for sub in addNounWords: for wrd in sub.split(): temp[wrd] += 1

# getting max frequency

res = max(temp, key=temp.get)

# printing result

print("Word with maximum frequency : " + str(res)) **OUTPUT:**

## Map Words to Properties Using Python Dictionariescode:

#creating and printing a dictionay by mapping word with its properties thisdict = {

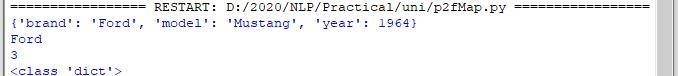
"brand": "Ford",

"model": "Mustang", "year": 1964

}

print(thisdict) print(thisdict["brand"]) print(len(thisdict)) print(type(thisdict))

## OUTPUT:



* 1. **Study i) DefaultTagger, ii) Regular expression tagger, iii) UnigramTagger**

## DefaultTagg ercode:

import nltk

from nltk.tag import DefaultTagger exptagger = DefaultTagger('NN') from nltk.corpus import treebank

testsentences = treebank.tagged\_sents() [1000:] print(exptagger.evaluate (testsentences))

#Tagging a list of sentences import nltk

from nltk.tag import DefaultTagger exptagger = DefaultTagger('NN')

print(exptagger.tag\_sents([['Hi', ','], ['How', 'are', 'you', '?']]))

## output

### Regular expression tagger, code:

from nltk.corpus import brown from nltk.tag import RegexpTagger

test\_sent = brown.sents(categories='news')[0] regexp\_tagger = RegexpTagger(

[(r'^-?[0-9]+(.[0-9]+)?$', 'CD'), # cardinal numbers (r'(The|the|A|a|An|an)$', 'AT'), # articles (r'.\*able$', 'JJ'), # adjectives

(r'.\*ness$', 'NN'), # nouns formed from adjectives (r'.\*ly$', 'RB'), # adverbs

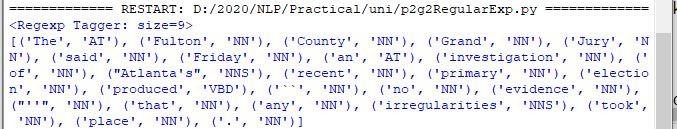
(r'.\*s$', 'NNS'), # plural nouns

(r'.\*ing$', 'VBG'), # gerunds

(r'.\*ed$', 'VBD'), # past tense verbs (r'.\*', 'NN') # nouns (default)

])

print(regexp\_tagger) print(regexp\_tagger.tag(test\_sent)) **output:**



## UnigramTager code:

# Loading Libraries

from nltk.tag import UnigramTagger from nltk.corpus import treebank

# Training using first 10 tagged sentences of the treebank corpus as data. # Using data

train\_sents = treebank.tagged\_sents()[:10]

# Initializing

tagger = UnigramTagger(train\_sents)

# Lets see the first sentence

# (of the treebank corpus) as list print(treebank.sents()[0]) print('\n',tagger.tag(treebank.sents()[0]))

#Finding the tagged results after training. tagger.tag(treebank.sents()[0])

#Overriding the context model

tagger = UnigramTagger(model ={'Pierre': 'NN'}) print('\n',tagger.tag(treebank.sents()[0]))

## output:

* 1. **Find different words from a given plain text without any space by comparing this text with a given corpus of words. Also find the score of words.**

## Question:

Initialize the hash tag test data or URL test data and convert to plain text without any space.. Read a text file of different words and compare the plain text data with the words exist in that text file and find out different words available in that plain text. Also find out how many words could be found. (for example, text = "#whatismyname" or text = [www.whatismyname.com.](http://www.whatismyname.com/) Convert that to plain text without space as: whatismyname and read text file as words.txt. Now compare plain text with words given in a file and find the words form the plain text and the count of words which could be found)

## Source code:

from future import with\_statement #with statement for reading file import re # Regular expression

words = [] # corpus file words testword = [] # test words

ans = [] # words matches with corpus

print("MENU")

print(" ")

print(" 1 . Hash tag segmentation ") print(" 2 . URL segmentation ")

print("enter the input choice for performing word segmentation") choice = int(input())

if choice == 1:

text = "#whatismyname" # hash tag test data to segment print("input with HashTag",text) pattern=re.compile("[^\w']") a = pattern.sub('', text)

elif choice == 2:

text = "[www.whatismyname.com](http://www.whatismyname.com/)" # url test data to segment print("input with URL",text)

a=re.split('\s|(?<!\d)[,.](?!\d)', text)

splitwords = ["www","com","in"] # remove the words which is containg in the list a ="".join([each for each in a if each not in splitwords])

else:

print("wrong choice...try again") print(a)

for each in a:

testword.append(each) #test word

test\_lenth = len(testword) # lenth of the test data

# Reading the corpus

with open('words.txt', 'r') as f: lines = f.readlines()

words =[(e.strip()) for e in lines]

def Seg(a,lenth): ans =[]

for k in range(0,lenth+1): # this loop checks char by char in the corpus

if a[0:k] in words:

print(a[0:k],"-appears in the corpus") ans.append(a[0:k])

break if ans != []:

g = max(ans,key=len) return g

test\_tot\_itr = 0 #each iteration value

answer = [] # Store the each word contains the corpus Score = 0 # initial value for score

N = 37 # total no of corpus M = 0

C = 0

while test\_tot\_itr < test\_lenth: ans\_words = Seg(a,test\_lenth) if ans\_words != 0:

test\_itr = len(ans\_words) answer.append(ans\_words) a = a[test\_itr:test\_lenth] test\_tot\_itr += test\_itr

Aft\_Seg = " ".join([each for each in answer]) # print segmented words in the list print("output")

print(" ")

print(Aft\_Seg) # print After segmentation the input

# Calculating Score C = len(answer)

score = C \* N / N # Calculate the score print("Score",score)

## Input:

### Words.txt

check domain big rocks name cheap being human current rates ought to

go down apple domains honesty hour follow

back social media 30

seconds earth this

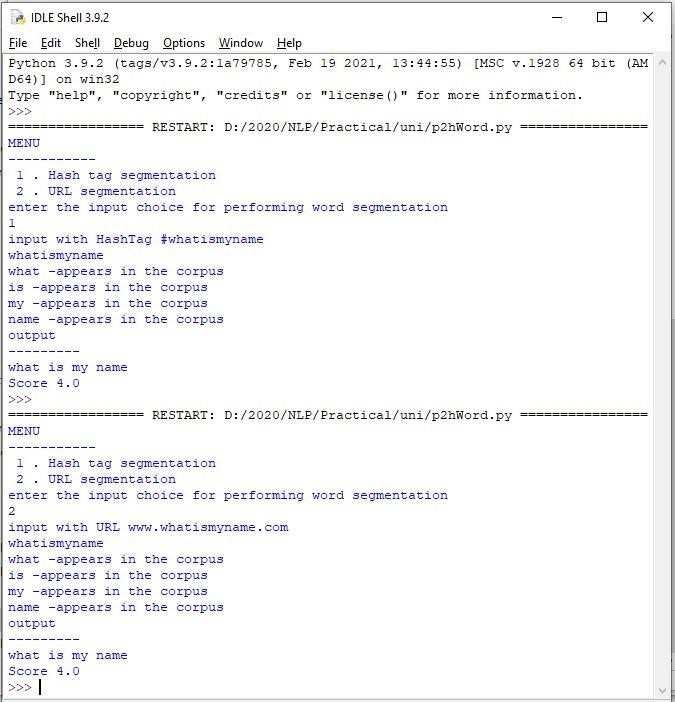
is insane it

time what is my

name let

us go

## Output:



### Practical No: 3

a. **Study of Wordnet Dictionary with methods as synsets, definitions, examples, antonyms**

### Source code:

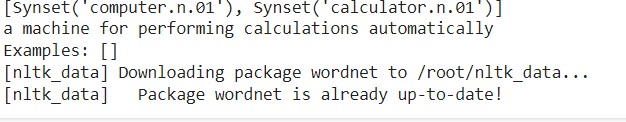
import nltk nltk.download('wordnet')

from nltk.corpus import wordnet print(wordnet.synsets("computer"))

# definition and example of the word ‘computer’ print(wordnet.synset("computer.n.01").definition()) #examples

print("Examples:", wordnet.synset("computer.n.01").examples())

### Output:



**B Study lemmas, hyponyms, hypernyms. Source code:**

import nltk

from nltk.corpus import wordnet print(wordnet.synsets("computer"))

print(wordnet.synset("computer.n.01").lemma\_names()) #all lemmas for each synset. for e in wordnet.synsets("computer"):

print(f'{e} --> {e.lemma\_names()}')

print(wordnet.lemma('computer.n.01.computing\_device').name())

#Hyponyms give abstract concepts of the word that are much more specific #the list of hyponyms words of the computer

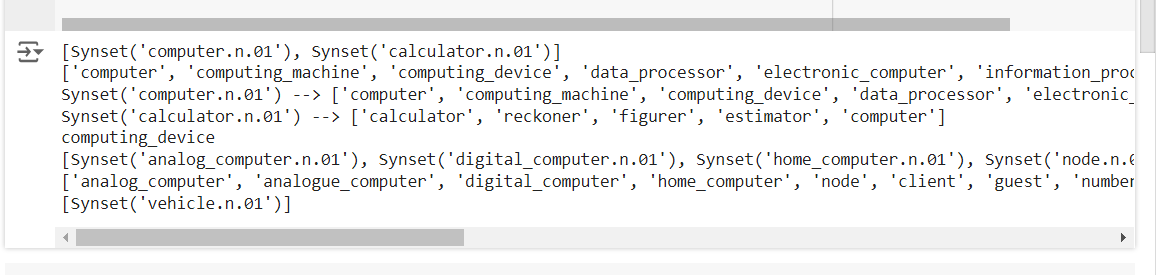
# Removed unnecessary import statement syn = wordnet.synset('computer.n.01') print(syn.hyponyms())

print([lemma.name() for synset in syn.hyponyms() for lemma in synset.lemmas()]) #the semantic similarity in WordNet

vehicle = wordnet.synset('vehicle.n.01') car = wordnet.synset('car.n.01')

print(car.lowest\_common\_hypernyms(vehicle))

### Output:



**C Write a program using python to find synonym and antonym of word "active" using Wordnet.**

### Source Code:

from nltk.corpus import wordnet print( wordnet.synsets("active"))

print(wordnet.lemma('active.a.01.active').antonyms())

### Output:



**D Compare two nouns Source Code:**

import nltk

from nltk.corpus import wordnet

syn1 = wordnet.synsets('football') syn2 = wordnet.synsets('soccer')

# A word may have multiple synsets, so need to compare each synset of word1 with synset of word2 for s1 in syn1:

for s2 in syn2:

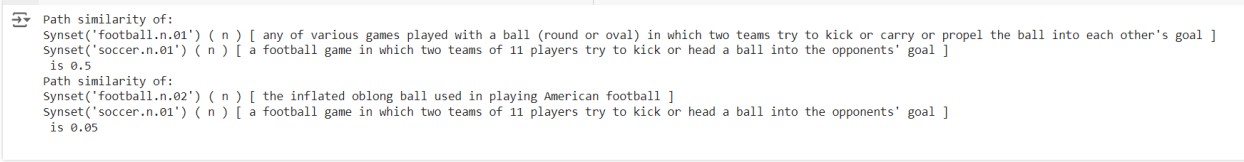
print("Path similarity of: ")

print(s1, '(', s1.pos(), ')', '[', s1.definition(), ']')

print(s2, '(', s2.pos(), ')', '[', s2.definition(), ']') print(" is", s1.path\_similarity(s2))

print()

### Output:



**E Handling stopword:**

### Using nltk Adding or Removing Stop Words in NLTK's Default Stop WordList

**Source Code:**

import nltk

from nltk.corpus import stopwords nltk.download('stopwords')

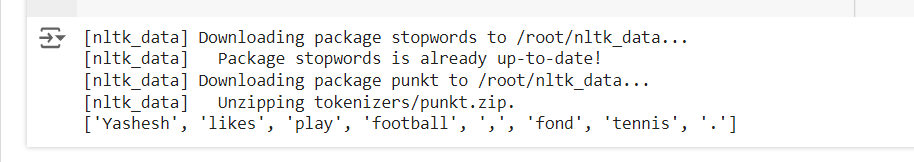
nltk.download('punkt') # Download the 'punkt' resource for sentence tokenization from nltk.tokenize import word\_tokenize

text = "Yashesh likes to play football, however he is not too fond of tennis." text\_tokens = word\_tokenize(text)

tokens\_without\_sw = [word for word in text\_tokens if not word in stopwords.words()]

print(tokens\_without\_sw)

### Output:



1. **Using Gensim Adding and Removing Stop Words in Default Gensim Stop Words List Source Code:**

import gensim

from gensim.parsing.preprocessing import remove\_stopwords

text = "Yashesh likes to play football, however he is not too fond of tennis." filtered\_sentence = remove\_stopwords(text)

print(filtered\_sentence)

all\_stopwords = gensim.parsing.preprocessing.STOPWORDS print(all\_stopwords)

'''The following script adds likes and play to the list of stop words in Gensim:''' from gensim.parsing.preprocessing import STOPWORDS

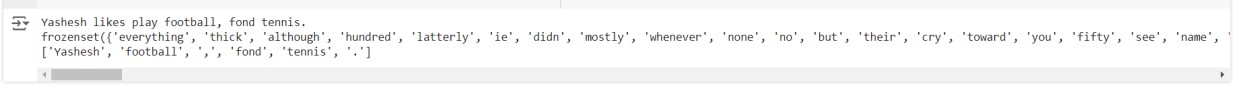
all\_stopwords\_gensim = STOPWORDS.union(set(['likes', 'play']))

text = "Yashesh likes to play football, however he is not too fond of tennis." text\_tokens = word\_tokenize(text)

tokens\_without\_sw = [word for word in text\_tokens if not word in all\_stopwords\_gensim]

print(tokens\_without\_sw)

### Output:



1. Using Spacy Adding and Removing Stop Words in Default Spacy Stop WordsList

### Source Code:

import spacy import nltk

from nltk.tokenize import word\_tokenize sp = spacy.load('en\_core\_web\_sm')

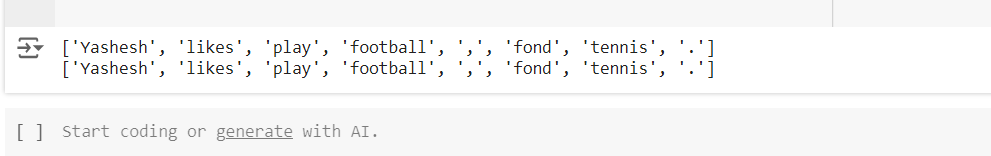
#add the word play to the NLTK stop word collection all\_stopwords = sp.Defaults.stop\_words all\_stopwords.add("play")

text = "Yashesh likes to play football, however he is not too fond of tennis." text\_tokens = word\_tokenize(text)

tokens\_without\_sw = [word for word in text\_tokens if not word in all\_stopwords] print(tokens\_without\_sw)

#remove 'not' from stop word collection all\_stopwords.remove('not') tokens\_without\_sw = [word for word in text\_tokens if not word in all\_stopwords] print(tokens\_without\_sw)

### Output:



**PRACTICAL 4 : Text Tokenization**

### Tokenization using Python’s split() Source Code:

text = """ This tool is an a beta stage. Alexa developers can use Get Metrics API to seamlessly analyse

metric. It also supports custom skill model, prebuilt Flash Briefing model, and the Smart Home Skill API.

You can use this tool for creation of monitors, alarms, and dashboards that spotlight changes.

The release of these three tools will enable developers to create visual rich skills for Alexa devices with screens. Amazon describes these tools as the collection of tech and tools for creating visually rich and interactive voice experiences. """

data = text.split('.') for i in data:

print (i)

### Output:



* 1. **Tokenization using Regular Expressions (RegEx) Source Code:**

import nltk

from nltk.tokenize import RegexpTokenizer # Import RegexpTokenizer

# Create a reference variable for Class RegexpTokenizer tk = RegexpTokenizer('\s+', gaps = True)

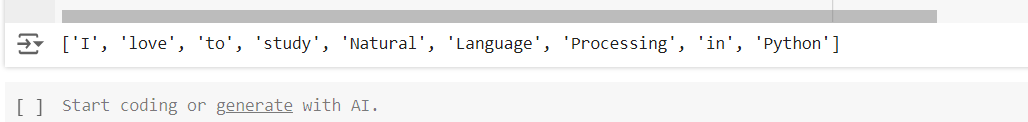
# Create a string input

str = "I love to study Natural Language Processing in Python" # Avoid using 'str' as a variable name, as it's a built-in type

# Use tokenize method tokens = tk.tokenize(str)

print(tokens)

### Output:



* 1. **Tokenization using NLTK Source Code:**

import nltk

from nltk.tokenize import word\_tokenize

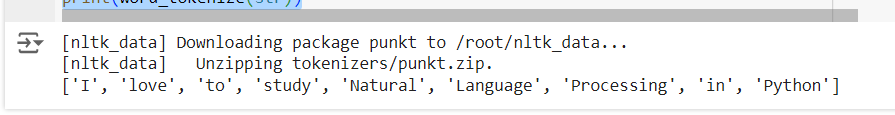
# Download the 'punkt' resource nltk.download('punkt')

# Create a string input

str = "I love to study Natural Language Processing in Python" # Consider renaming this variable to avoid shadowing the built-in 'str' type

# Use tokenize method print(word\_tokenize(str))

### Output:



* 1. **Tokenization using the spaCy library**

### Source Code:

import spacy

nlp = spacy.blank("en") # Create a string input

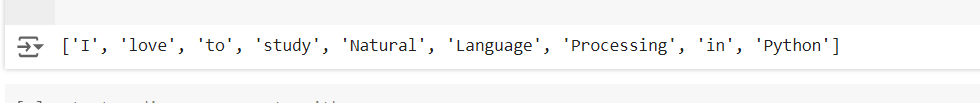
str = "I love to study Natural Language Processing in Python"

# Create an instance of document;

# doc object is a container for a sequence of Token objects. doc = nlp(str)

# Read the words; Print the words # words = [word.text for word in doc] print(words)

### Output:



* 1. **Tokenization using Keras Source Code:**

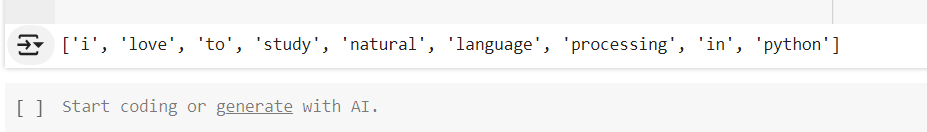
import keras

from keras.preprocessing.text import text\_to\_word\_sequence # Create a string input

str = "I love to study Natural Language Processing in Python" # tokenizing the text

tokens = text\_to\_word\_sequence(str) print(tokens)

### Output:



* 1. **Tokenization using Gensim Source Code:**

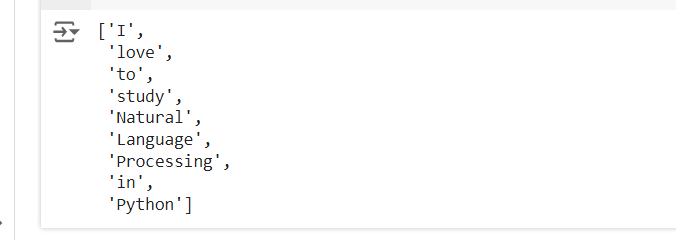
from gensim.utils import tokenize # Create a string input

str = "I love to study Natural Language Processing in Python"

# tokenizing the text

list(tokenize(str))

### Output:



### PRACTICAL 5: Illustrate part of speech tagging.

1. **Sentence tokenization, word tokenization, Part of speech Tagging and chunking of user defined text.**

### Source Code:

import nltk

from nltk import tokenize nltk.download('punkt') from nltk import tag from nltk import chunk

nltk.download('averaged\_perceptron\_tagger') nltk.download('maxent\_ne\_chunker') nltk.download('words')

para = "Hello! My name is Roman Reigns. Today you'll be learning NLTK." sents = tokenize.sent\_tokenize(para)

print("\nsentence tokenization\n===================\n",sents)

# word tokenization

print("\nword tokenization\n===================\n") for index in range(len(sents)):

words = tokenize.word\_tokenize(sents[index]) print(words)

# POS Tagging tagged\_words = []

for index in range(len(sents)): tagged\_words.append(tag.pos\_tag(words))

print("\nPOS Tagging\n===========\n",tagged\_words)

# chunking tree = []

for index in range(len(sents)): tree.append(chunk.ne\_chunk(tagged\_words[index])) print("\nchunking\n========\n", tree)

### OUTPUT:

1. **Named Entity recognition using user**

### Source Code:

import spacy

# Load English tokenizer, tagger, parser and NER

nlp = spacy.load("en\_core\_web\_sm") # removed comment from this line

# Process whole documents

text = ("When Sebastian Thrun started working on self-driving cars at " "Google in 2007, few people outside of the company took him " "seriously. “I can tell you very senior CEOs of major American "

"car companies would shake my hand and turn away because I wasn’t " "worth talking

to,” said Thrun, in an interview with Recode earlier " "this week.") doc = nlp(text)

# Analyse syntax

print("Noun phrases:", [chunk.text for chunk in doc.noun\_chunks])

print("Verbs:", [token.lemma\_ for token in doc if token.pos\_ == "VERB"])

### Output:

### 

## PRACTICAL 6: Finite state automata

1. **Define grammar using nltk. Analyze a sentence**

### Source Code:

import nltk

from nltk import tokenize

grammar1 = nltk.CFG.fromstring(""" S -> VP NP VP -> VP NP

NP -> Det NP

Det -> 'that'

NP -> singular Noun NP -> 'flight'

VP -> 'Book' """) # Fixed formatting of grammar rules sentence = "Book that flight"

all\_tokens = sentence.split() # Tokenize the sentence for index in range(len(sentence)):

print(all\_tokens)

parser = nltk.ChartParser(grammar1) for tree in parser.parse(all\_tokens):

print(tree)

tree.draw()

**OUTPUT:**



1. **Accept the input string with Regular expression of Finite**

### Source Code:

def FA(s):

if len(s)<3: #if the length is less than 3 then it can't be accepted, Therefore end the process.

return "Rejected"

if s[0]=='1': #first three characters are fixed. Therefore, checking them using index

if s[1]=='0':

if s[2]=='1':

# After index 2 only "1" can appear. Therefore break the process if any other character is detected

for i in range(3,len(s)): if s[i]!='1':

return "Rejected"

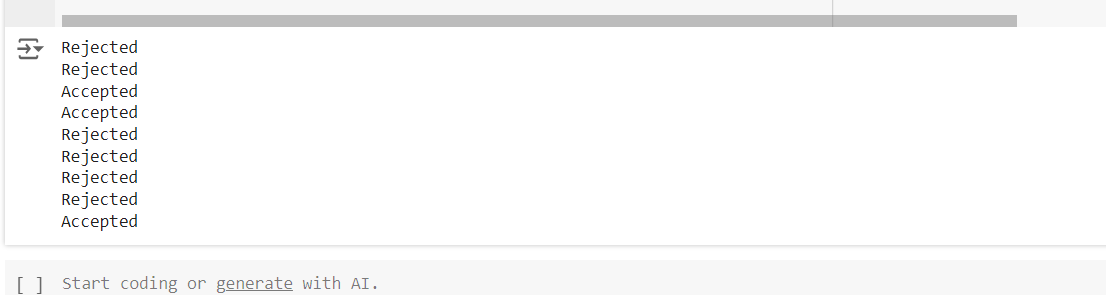
return "Accepted" # if all 4 nested if true return "Rejected" # else of 3rd if

return "Rejected" # else of 2nd if return "Rejected" # else of 1st if

inputs=['1','10101','101','10111','01010','100','','10111101','1011111']

for i in inputs: print(FA(i))

OUTPUT:



1. **Accept the input string with Regular expression of FA**

### Source Code:

def FA(s): size=0

#scan complete string and make sure that it contains only 'a' & 'b' for i in s:

if i=='a' or i=='b':

size+=1 # Indent this line to include it in the if block else:

return "Rejected"

#After checking that it contains only 'a' & 'b' #check it's length it should be 3 atleast

if size>=3:

pass # Add code here to continue the logic pass # Add code here to continue

if s[size-3]=='b':

if s[size-2]=='b':

if s[size-1]=='a':

return "Accepted" # if all 4 if true return "Rejected" # else of 4th if return "Rejected" # else of 3rd if

return "Rejected" # else of 2nd if return "Rejected" # else of 1st if

inputs=['bba', 'ababbba', 'abba','abb', 'baba','bbb',''] for i in inputs:

print(FA(i))

### Output:

### 

1. **Implementation of Deductive Chart Parsing using context free grammar and a given sentence.**

## Source Code:

import nltk nltk.download('punkt')

import nltk

from nltk import tokenize

# Corrected grammar definition grammar1 = nltk.CFG.fromstring(""" S -> NP VP

PP -> P NP

NP -> Det N | Det N PP | 'I' VP -> V NP | VP PP

Det -> 'a' | 'my'

N -> 'bird' | 'balcony' V -> 'saw'

P -> 'in'

""")

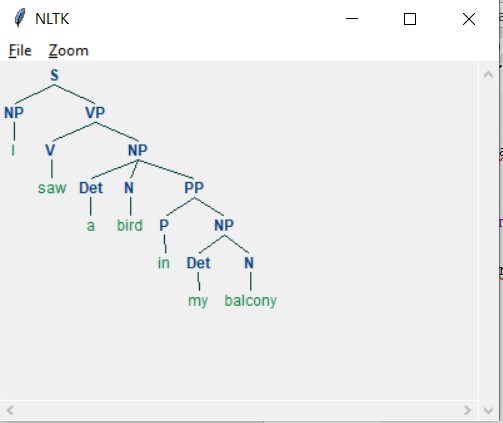
sentence = "I saw a bird in my balcony" # Moved tokenization outside the loop

all\_tokens = tokenize.word\_tokenize(sentence) print(all\_tokens)

parser = nltk.ChartParser(grammar1) for tree in parser.parse(all\_tokens):

print(tree) tree.draw()

## Output:



**PRACTICAL 7**

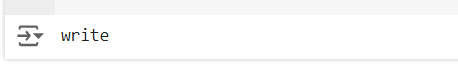
**Study Porter Stemmer, Lancaster Stemmer, Regex Stemmer,**

## Snowball Stemmer Study WordNet Lemmatizer

# PorterStemmer import nltk

from nltk.stem import PorterStemmer word\_stemmer = PorterStemmer() print(word\_stemmer.stem('writing'))

## Output:



#LancasterStemmer import nltk

from nltk.stem import LancasterStemmer Lanc\_stemmer = LancasterStemmer() print( Lanc\_stemmer.stem('writing'))

## Output:



#RegexpStemmer import nltk

from nltk.stem import RegexpStemmer

Reg\_stemmer = RegexpStemmer('ing$|s$|e$|able$', min=4) print(Reg\_stemmer.stem('writing'))

## Output:



#SnowballStemmer import nltk

from nltk.stem import SnowballStemmer english\_stemmer = SnowballStemmer('english') print(english\_stemmer.stem ('writing'))

#WordNetLemmatizer import nltk

from nltk.stem import WordNetLemmatizer

# Download the WordNet corpus if it hasn't been downloaded already nltk.download('wordnet')

lemmatizer = WordNetLemmatizer()

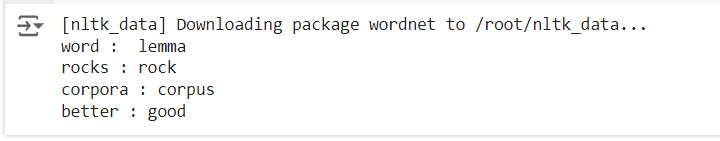
print("word :\tlemma")

print("rocks :", lemmatizer.lemmatize("rocks")) print("corpora :", lemmatizer.lemmatize("corpora"))

# a denotes adjective in "pos"

print("better :", lemmatizer.lemmatize("better", pos ="a"))

## Output:



**PRACTICAL 8: Implement Naive Bayes**

## classifierCode:

#pip install pandas #pip install sklearn

import pandas as pd import numpy as np

sms\_data = pd.read\_csv("spam.csv", encoding='latin-1') import re

import nltk

from nltk.corpus import stopwords

from nltk.stem.porter import PorterStemmer

stemming = PorterStemmer() corpus = []

for i in range (0,len(sms\_data)):

s1 = re.sub('[^a-zA-Z]',repl = ' ',string = sms\_data['v2'][i]) s1.lower()

s1 = s1.split()

s1 = [stemming.stem(word) for word in s1 if word not in set(stopwords.words('english'))]

s1 = ' '.join(s1) corpus.append(s1)

from sklearn.feature\_extraction.text import CountVectorizer countvectorizer =CountVectorizer()

x = countvectorizer.fit\_transform(corpus).toarray() print(x)

y = sms\_data['v1'].values print(y)

from sklearn.model\_selection import train\_test\_split x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size = 0.3, stratify=y,random\_state=2)

#Multinomial Naïve Bayes.

from sklearn.naive\_bayes import MultinomialNB multinomialnb = MultinomialNB() multinomialnb.fit(x\_train,y\_train)

# Predicting on test data:

y\_pred = multinomialnb.predict(x\_test)

print(y\_pred)

#Results of our Models

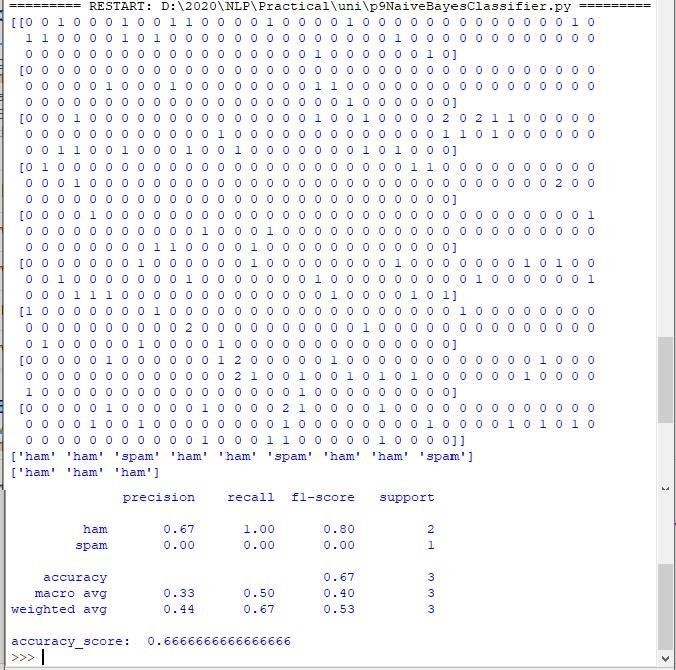
from sklearn.metrics import classification\_report, confusion\_matrix from sklearn.metrics import accuracy\_score

print(classification\_report(y\_test,y\_pred)) print("accuracy\_score: ",accuracy\_score(y\_test,y\_pred))

## input:

spam.csv file from github

## output:



**PRACTICAL 9: Speech Tagging**

## A] Speech tagging using spacy Source code:

import spacy

sp = spacy.load('en\_core\_web\_sm')

sen = sp(u"I like to play football. I hated it in my childhood though") print(sen.text)

print(sen[7].pos\_) print(sen[7].tag\_) print(spacy.explain(sen[7].tag\_)) for word in sen:

print(f'{word.text:{12}} {word.pos\_:{10}} {word.tag\_:{8}}

{spacy.explain(word.tag\_)}') # Indent this line to include it in the for loop

sen = sp(u'Can you google it?') word = sen[2]

print(f'{word.text:{12}} {word.pos\_:{10}} {word.tag\_:{8}}

{spacy.explain(word.tag\_)}')

sen = sp(u'Can you search it on google?') word = sen[5]

print(f'{word.text:{12}} {word.pos\_:{10}} {word.tag\_:{8}}

{spacy.explain(word.tag\_)}')

#Finding the Number of POS Tags

sen = sp(u"I like to play football. I hated it in my childhood though")

num\_pos = sen.count\_by(spacy.attrs.POS) # Remove extra num\_pos

for k,v in sorted(num\_pos.items()):

print(f'{k}. {sen.vocab[k].text:{8}}: {v}') # Indent this line to include it in the for loop

#Visualizing Parts of Speech Tags

# from spacy import displacy # Uncomment this line if you want to use displacy

sen = sp(u"I like to play football. I hated it in my childhood though") # displacy.serve(sen, style='dep', options={'distance': 120}) # Uncomment this line if you want to use displacy

## Output:



1. **Speech tagging using nltkcode**

## Source Code:

import nltk

from nltk.corpus import state\_union

from nltk.tokenize import PunktSentenceTokenizer

# Download the 'state\_union' corpus nltk.download('state\_union') nltk.download('punkt') nltk.download('averaged\_perceptron\_tagger')

# Create training and testing data

train\_text = state\_union.raw("2005-GWBush.txt") sample\_text = state\_union.raw("2006-GWBush.txt")

# Train the Punkt tokenizer

custom\_sent\_tokenizer = PunktSentenceTokenizer(train\_text)

# Tokenize the sample text

tokenized = custom\_sent\_tokenizer.tokenize(sample\_text)

def process\_content(): try:

for i in tokenized[:2]:

words = nltk.word\_tokenize(i)

tagged = nltk.pos\_tag(words)



print(tagged) except Exception as e:

print(str(e))

# Call the function to process the content process\_content()

**Output:**

1. **Statistical parsing: Source Code:**

import nltk

import nltk.parse.viterbi import nltk.parse.pchart

# Download the 'treebank' corpus nltk.download('treebank')

def give(t):

return t.label() == 'VP' and len(t) > 2 and t[1].label() == 'NP' and (t[2].label() == 'PP-DTV' or t[2].label() == 'NP') and \

('give' in t[0].leaves() or 'gave' in t[0].leaves())

def sent(t):

return ' '.join(token for token in t.leaves() if token[0] not in '\*- 0')

def print\_node(t, width):

output = "%s %s: %s / %s: %s" % \

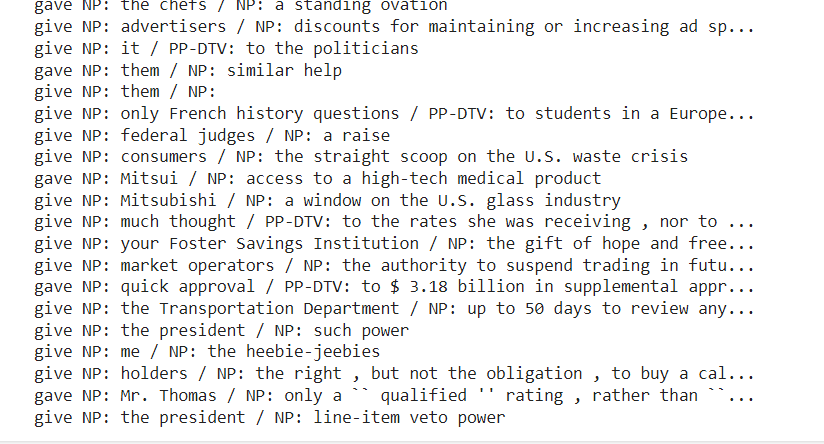
(sent(t[0]), t[1].label(), sent(t[1]), t[2].label(), sent(t[2])) if len(output) > width:

output = output[:width] + "..." print (output)

for tree in nltk.corpus.treebank.parsed\_sents(): for t in tree.subtrees(give):

print\_node(t, 72)

## Output:



1. **Probabilistic parser**

**Source Code:**

import nltk

from nltk import PCFG

grammar = PCFG.fromstring('''

NP -> NNS [0.5] | JJ NNS [0.3] | NP CC NP [0.2]

NNS -> "men" [0.1] | "women" [0.2] | "children" [0.3] | NNS CC NNS [0.4]

JJ -> "old" [0.4]

JJ -> "young" [0.6] # Moved comment to a separate line CC -> "and" [0.9] | "or" [0.1] ''')

print(grammar)

viterbi\_parser = nltk.ViterbiParser(grammar) token = "old men and women".split()

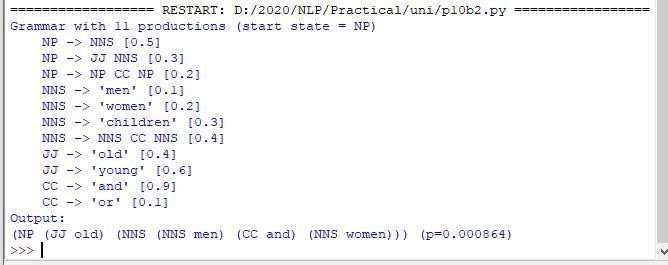
obj = viterbi\_parser.parse(token)

print("Output: ") for x in obj:

print(x)

**Output:**

**OUTPUT**



# DEEP LEARNING

# D:\IMP.DATA\Desktop\Shital\Printing Banner,Banners,Certificates\NEW_LOGO-removebg-preview.png

****

**SAKET GYANPEETH’S**

**SAKET COLLEGE OF ARTS, SCIENCE AND COMMERCE**

**(Permanently Affiliated to University of Mumbai)**

**NAAC Accredited B Grade**

Saket Vidyanagari Marg, Chinchpada Road, Katemanivali, Kalyan (East) – 421306, Dist. Thane (MAH)

Department of Information Technology

**CERTIFICATE**

**This is to certify that**

**---** of **MSc INFORMATION TECHNOLOGY PART-II** Class has satisfactorily carried out the required practicals in the subject.

**DEEP LEARNING**

For the Academic year 2023 – 2024

**Practical In-Charge Head of Department External Examiner**

# Index

|  |  |  |
| --- | --- | --- |
| Sr.  no. | Title | Sign |
| 1 | Performing matrix multiplication and finding eigen vectors and eigen values using TensorFlow |  |
| 2 | Solving XOR problem using deep feed forward network. |  |
| 3 | Implementing deep neural network for performing binary classification task. |  |
| 4 | a) Aim: Using deep feed forward network with two |  |
|  | hidden layers for performing multiclass classification |
|  | and predicting the class. |
|  | b) Aim: Using a deep feed forward network with two |
|  | hidden layers for performing classification and |
|  | predicting the probability of class. |
|  | c) Aim: Using a deep feed forward network with two hidden layers for performing linear regression and |
|  | predicting values. |
| 5 | a) Evaluating feed forward deep network for regression |  |
|  | using KFold cross validation. |
|  | b) Evaluating feed forward deep network for multiclass |
|  | Classification using KFold cross-validation. |
| 6 | Implementing regularization to avoid overfitting in binary classification. |  |
| 7 | Demonstrate recurrent neural network that learns to perform sequence analysis for stock price. |  |
| 8 | Performing encoding and decoding of images using deep  autoencoder. |  |
| 9 | Implementation of convolutional neural network to predict numbers from number images |  |
| 10 | Denoising of images using autoencoder. |  |

Practical No:1

## Aim: Performing matrix multiplication and finding eigen vectors and eigen values using TensorFlow.

import tensorflow as tf print("Matrix Multiplication Demo")

x=tf.constant([1,2,3,4,5,6],shape=[2,3]) print(x) y=tf.constant([7,8,9,10,11,12],shape=[3,2]) print(y)

z=tf.matmul(x,y) # Calculate the matrix product print("Product:",z) # Print the result on a separate line

e\_matrix\_A=tf.random.uniform([2,2],minval=3,maxval=10,dtype=tf.float32, name="matrixA")

print("Matrix A:\n{}\n\n".format(e\_matrix\_A)) eigen\_values\_A,eigen\_vectors\_A=tf.linalg.eigh(e\_matrix\_A) print("Eigen Vectors:\n{}\n\nEigen Values:\n{}\n".format(eigen\_vectors\_A,eigen\_values\_A))

**Eigenvectors and Eigenvalues**: Eigenvectors are unit vectors that mean

their length or magnitude is equal to one. They are often known as right vectors, which simply mean a column vector (as opposed to a row vector or a left vector).

A matrix contains only positive eigenvalues is known as a positive definite matrix, and if it contains all negative eigenvalues, it is known as a negative definite matrix.

# OUTPUT:

**Practical No:2**

## Aim: Solving XOR problem using deep feed forward network.

import numpy as np

from keras.layers import Dense from keras.models import Sequential model=Sequential()

model.add(Dense(units=2,activation='relu',input\_dim=2)) model.add(Dense(units=1,activation='sigmoid')) model.compile(loss='binary\_crossentropy',optimizer='adam',metrics=['acc uracy'])

print(model.summary()) print(model.get\_weights()) X=np.array([[0.,0.],[0.,1.],[1.,0.],[1.,1.]]) Y=np.array([0.,1.,1.,0.])

model.fit(X,Y,epochs=1000,batch\_size=4)

print(model.get\_weights()) print(model.predict(X,batch\_size=4))

**DEEP FEEDFORWARD NETWORK :-** Multi-layered Network of neurons is composed of many sigmoid neurons. MLNs are capable of handling the non-linearly separable data. The layers present between the input and output layers are called hidden layers. The hidden layers are used to handle the complex non-linearly separable relations between input and the output.

# OUTPUT:

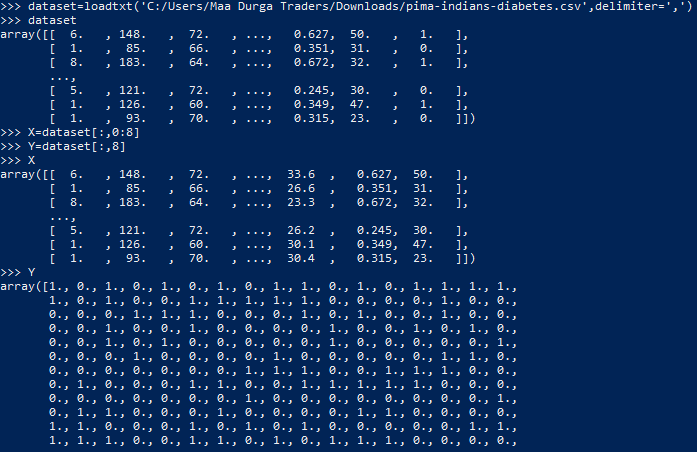
**Practical No:3**

## Aim: Implementing deep neural network for performing classification task.

**Problem statement:** the given dataset comprises of health information about diabetic women patient. we need to create deep feed forward network that will classify women suffering from diabetes mellitus as 1.

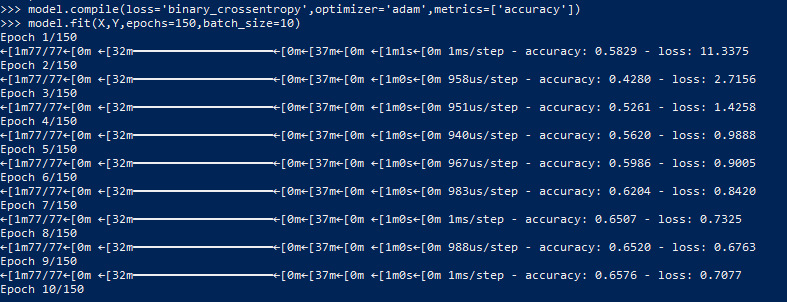




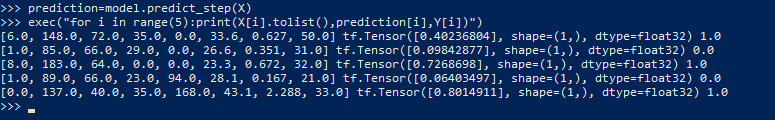


Creating model:



Compiling and fitting model:

Evaluating the accuracy:

Using model for prediction class:

# Practical No:4

## Aim: Using deep feed forward network with two hidden layers for performing classification and predicting the class.

from keras.models import Sequential from keras.layers import Dense

from sklearn.datasets import make\_blobs

from sklearn.preprocessing import MinMaxScaler import numpy as np # Import numpy for argmax function

X,Y=make\_blobs(n\_samples=100,centers=2,n\_features=2,random\_state=1) scalar=MinMaxScaler()

scalar.fit(X) X=scalar.transform(X) model=Sequential()

model.add(Dense(4,input\_dim=2,activation='relu')) model.add(Dense(4,activation='relu')) model.add(Dense(1,activation='sigmoid')) model.compile(loss='binary\_crossentropy',optimizer='adam') model.fit(X,Y,epochs=500)

Xnew,Yreal=make\_blobs(n\_samples=3,centers=2,n\_features=2,random\_state=1

)

Xnew=scalar.transform(Xnew)

# Use predict and argmax to get predicted classes Ynew\_probs = model.predict(Xnew)

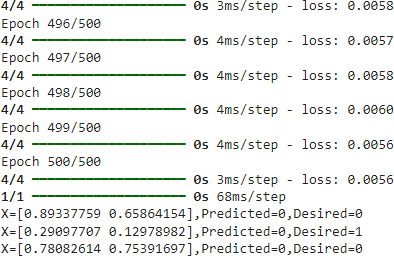
Ynew = np.argmax(Ynew\_probs, axis=-1)

for i in range(len(Xnew)):

# The print statement should be at the same indentation level as the 'for' loop

print("X=%s,Predicted=%s,Desired=%s"%(Xnew[i],Ynew[i],Yreal[i]))

# OUTPUT:



## Aim: Using a deep field forward network with two hidden layers for performing classification and predicting the probability of class.

from keras.models import Sequential from keras.layers import Dense

from sklearn.datasets import make\_blobs

from sklearn.preprocessing import MinMaxScaler import numpy as np # Import numpy for argmax function

X,Y=make\_blobs(n\_samples=100,centers=2,n\_features=2,random\_state=1) scalar=MinMaxScaler()

scalar.fit(X) X=scalar.transform(X) model=Sequential()

model.add(Dense(4,input\_dim=2,activation='relu')) model.add(Dense(4,activation='relu')) model.add(Dense(1,activation='sigmoid')) model.compile(loss='binary\_crossentropy',optimizer='adam') model.fit(X,Y,epochs=500) Xnew,Yreal=make\_blobs(n\_samples=3,centers=2,n\_features=2,random\_state=1

)

Xnew=scalar.transform(Xnew)

# Use predict and argmax to get predicted classes Y\_pred\_probs = model.predict(Xnew)

Yclass = np.argmax(Y\_pred\_probs, axis=1) # Get class with highest probability

Ynew=model.predict(Xnew) # predict\_proba is deprecated, use predict instead

for i in range(len(Xnew)):

print("X=%s,Predicted\_probability=%s,Predicted\_class=%s"%(Xnew[ i],Ynew[i],Yclass[i]))

# OUTPUT:

## Aim: Using a deep field forward network with two hidden layers for performing linear regression and predicting values.

from keras.models import Sequential from keras.layers import Dense

from sklearn.datasets import make\_regression from sklearn.preprocessing import MinMaxScaler

X,Y=make\_regression(n\_samples=100,n\_features=2,noise=0.1,random\_state=1

)

scalarX,scalarY=MinMaxScaler(),MinMaxScaler()

scalarX.fit(X) scalarY.fit(Y.reshape(100,1)) X=scalarX.transform(X) Y=scalarY.transform(Y.reshape(100,1)) model=Sequential()

model.add(Dense(4,input\_dim=2,activation='relu')) model.add(Dense(4,activation='relu')) model.add(Dense(1,activation='sigmoid')) model.compile(loss='mse',optimizer='adam') model.fit(X,Y,epochs=1000,verbose=0)

Xnew,a=make\_regression(n\_samples=3,n\_features=2,noise=0.1,random\_state= 1)

Xnew=scalarX.transform(Xnew)

Ynew=model.predict(Xnew)

for i in range(len(Xnew)):

print("X=%s,Predicted=%s"%(Xnew[i],Ynew[i]))

# OUTPUT:

**Practical No:5(a)**

## Aim: Evaluating feed forward deep network for regression using KFold cross validation.

**To Generate the CSV file.**

import pandas as pd import numpy as np

# Define the column names

columns = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS',

'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']

# Generate random data np.random.seed(0) # For reproducibility data = np.random.rand(100, 14)

# Scale data to appropriate ranges data[:, 0] \*= 10 # CRIM

data[:, 1] \*= 100 # ZN

data[:, 2] \*= 30 # INDUS

data[:, 3] = np.random.randint(0, 2, size=(100)) # CHAS as binary data[:, 4] \*= 1 # NOX

data[:, 5] = data[:, 5] \* 4 + 3 # RM

data[:, 6] \*= 100 # AGE

data[:, 7] \*= 12 # DIS

data[:, 8] = np.random.randint(1, 25, size=(100)) # RAD as integer data[:, 9] \*= 600 # TAX

data[:, 10] \*= 30 # PTRATIO

data[:, 11] \*= 400 # B

data[:, 12] \*= 40 # LSTAT

data[:, 13] \*= 50 # MEDV

# Create a DataFrame

df = pd.DataFrame(data, columns=columns)

# Save to CSV df.to\_csv('housing.csv', index=False)

## Code:

!pip install scikeras import pandas as pd

from keras.models import Sequential from keras.layers import Dense

from scikeras.wrappers import KerasRegressor # Import KerasRegressor from scikeras

from sklearn.model\_selection import cross\_val\_score from sklearn.model\_selection import KFold

from sklearn.preprocessing import StandardScaler from sklearn.pipeline import Pipeline

# load dataset

dataframe = pd.read\_csv("housing.csv") # Replace 'housing.csv' with the actual file path if it's in a different directory.

dataset = dataframe.values X = dataset[:,0:13]

Y = dataset[:,13] def wider\_model():

model=Sequential() model.add(Dense(15,input\_dim=13,kernel\_initializer='normal',act

ivation='relu'))

model.add(Dense(13,kernel\_initializer='normal',activation='relu

'))

model.add(Dense(1,kernel\_initializer='normal')) model.compile(loss='mean\_squared\_error',optimizer='adam')

return model estimators=[]

estimators.append(('standardize',StandardScaler())) estimators.append(('mlp',KerasRegressor(build\_fn=wider\_model,epochs=100

,batch\_size=5))) pipeline=Pipeline(estimators) kfold=KFold(n\_splits=10) results=cross\_val\_score(pipeline,X,Y,cv=kfold)

print("Wider: %.2f (%.2f) MSE" % (results.mean(), results.std()))

# OUTPUT:



(After changing neuron)

model.add(Dense(20, input\_dim=13,kernel\_initializer='normal',activation='relu'))



# Practical No: 5b

## Aim: Evaluating feed forward deep network for multiclass Classification using KFold cross-validation.

**To Generate the CSV file.**

import pandas as pd import numpy as np

# Define the column names

columns = ['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS',

'RAD', 'TAX', 'PTRATIO', 'B', 'LSTAT', 'MEDV']

# Generate random data np.random.seed(0) # For reproducibility data = np.random.rand(100, 14)

# Scale data to appropriate ranges data[:, 0] \*= 10 # CRIM

data[:, 1] \*= 100 # ZN

data[:, 2] \*= 30 # INDUS

data[:, 3] = np.random.randint(0, 2, size=(100)) # CHAS as binary data[:, 4] \*= 1 # NOX

data[:, 5] = data[:, 5] \* 4 + 3 # RM

data[:, 6] \*= 100 # AGE

data[:, 7] \*= 12 # DIS

data[:, 8] = np.random.randint(1, 25, size=(100)) # RAD as integer data[:, 9] \*= 600 # TAX

data[:, 10] \*= 30 # PTRATIO

data[:, 11] \*= 400 # B

data[:, 12] \*= 40 # LSTAT

data[:, 13] \*= 50 # MEDV

# Create a DataFrame

df = pd.DataFrame(data, columns=columns)

# Save to CSV df.to\_csv('flower.csv', index=False)

## Code:

#loading libraries

import pandas as pd # Import the pandas library with the alias 'pd' from keras.models import Sequential

from keras.layers import Dense

!pip install scikeras # Install the scikeras package

from scikeras.wrappers import KerasClassifier # Import from scikeras from sklearn.model\_selection import cross\_val\_score

from sklearn.model\_selection import KFold from sklearn.preprocessing import LabelEncoder #loading dataset

df=pd.read\_csv('flower.csv',header=0) # Tell pandas that the first row contains headers

print(df)

#splitting dataset into input and output variables X = df.iloc[:,0:4].astype(float)

y=df.iloc[:,4]

# ... rest of your code ... #print(X)

#print(y)

# ... (rest of your code) ...

#encoding string output into numeric output encoder=LabelEncoder()

encoder.fit(y) encoded\_y=encoder.transform(y)

# print(encoded\_y) # No need to print this

# Correctly create one-hot encoded target variable

num\_classes = len(np.unique(encoded\_y)) # Determine the number of unique classes

dummy\_Y = to\_categorical(encoded\_y, num\_classes=num\_classes) print(dummy\_Y)

def baseline\_model(): # create model

model = Sequential()

model.add(Dense(8, input\_dim=4, activation='relu'))

# Adjust the output layer to match the number of classes model.add(Dense(num\_classes, activation='softmax'))

# Compile model

model.compile(loss='categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

return model

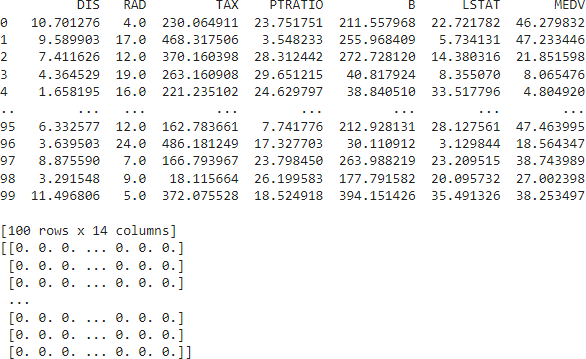
# ... (rest of your code) ... estimator=baseline\_model() estimator.fit(X,dummy\_Y,epochs=100,shuffle=True) action=estimator.predict(X)

for i in range(25): print(dummy\_Y[i]) print('^^^^^^^^^^^^^^^^^^^^^^')

for i in range(25):

print(action[i])

**OUTPUT:**



# Code 2:

**To Generate the CSV File.**

import pandas as pd

# Define the data data = {

'Name': ['Rose', 'Tulip', 'Sunflower', 'Lily', 'Orchid'],

'Color': ['Red', 'Yellow', 'Yellow', 'White', 'Purple'],

'PetalLength': [5.5, 4.2, 6.1, 4.5, 3.8],

'PetalWidth': [2.0, 1.5, 3.0, 2.5, 1.8]

}

# Create a DataFrame df = pd.DataFrame(data)

# Save to CSV df.to\_csv('Flower.csv', index=False)

**Code:**

!pip install scikeras import pandas

from keras.models import Sequential from keras.layers import Dense

from scikeras.wrappers import KerasClassifier from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import cross\_val\_score from sklearn.model\_selection import KFold

from tensorflow.keras.utils import to\_categorical

dataset=pandas.read\_csv("Flower.csv") # Read the CSV file with headers recognized

dataset1=dataset.values

X=dataset1[:,2:4].astype(float) # Select only the numeric columns 'PetalLength' and 'PetalWidth'

Y=dataset1[:,1] # Select the 'Color' column as labels print(Y)

encoder=LabelEncoder() encoder.fit(Y) encoder\_Y=encoder.transform(Y) print(encoder\_Y) dummy\_Y=to\_categorical(encoder\_Y) print(dummy\_Y)

def baseline\_model(): model=Sequential()

model.add(Dense(8,input\_dim=2,activation='relu')) # Adjust input\_dim to 2

model.add(Dense(4,activation='softmax')) # Change the output layer size to 4

model.compile(loss='categorical\_crossentropy',optimizer='adam',metric s=['accuracy'])

return model estimator=KerasClassifier(build\_fn=baseline\_model,epochs=100,batch\_size

=5)

# Change the number of splits to be less than or equal to the number of samples

kfold = KFold(n\_splits=5, shuffle=True) # Use 5 splits or less results = cross\_val\_score(estimator, X, dummy\_Y, cv=kfold) print("Baseline: %.2f%% (%.2f%%)" % (results.mean()\*100,

results.std()\*100))

## OUTPUT:



(Changing neuron) model.add(Dense(10,input\_dim=4,activation='relu'))



# Practical No. 6

## Aim: implementing regularization to avoid overfitting in binary classification.

from matplotlib import pyplot

from sklearn.datasets import make\_moons from keras.models import Sequential from keras.layers import Dense

X,Y=make\_moons(n\_samples=100,noise=0.2,random\_state=1) n\_train=30

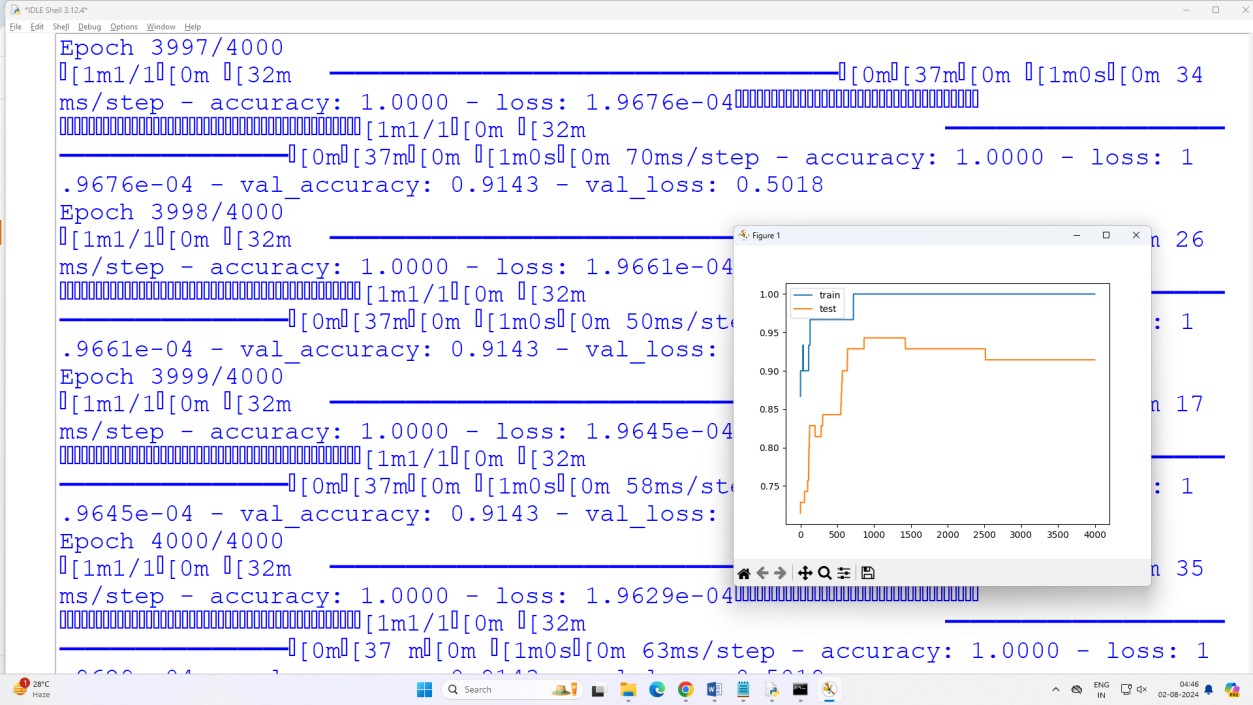
trainX,testX=X[:n\_train,:],X[n\_train:] trainY,testY=Y[:n\_train],Y[n\_train:] #print(trainX)

#print(trainY) #print(testX) #print(testY) model=Sequential()

model.add(Dense(500,input\_dim=2,activation='relu')) model.add(Dense(1,activation='sigmoid')) model.compile(loss='binary\_crossentropy',optimizer='adam',metrics=['acc uracy']) history=model.fit(trainX,trainY,validation\_data=(testX,testY),epochs=40 00)

pyplot.plot(history.history['accuracy'],label='train') pyplot.plot(history.history['val\_accuracy'],label='test') pyplot.legend()

pyplot.show()

**Output:**

The above code and resultant graph demonstrate overfitting with accuracy of testing data less than accuracy of training data also the accuracy of testing data increases once and then start decreases gradually.to solve this problem we can use regularization

Hence, we will add two lines in the above code as highlighted below to implement l2 regularization with alpha=0.001

from matplotlib import pyplot

from sklearn.datasets import make\_moons from keras.models import Sequential

from keras.layers import Dense from keras.regularizers import l2

X,Y=make\_moons(n\_samples=100,noise=0.2,random\_state=1) n\_train=30

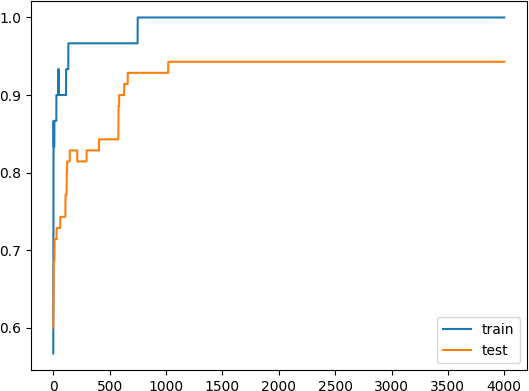
trainX,testX=X[:n\_train,:],X[n\_train:] trainY,testY=Y[:n\_train],Y[n\_train:] #print(trainX)

#print(trainY) #print(testX) #print(testY) model=Sequential()

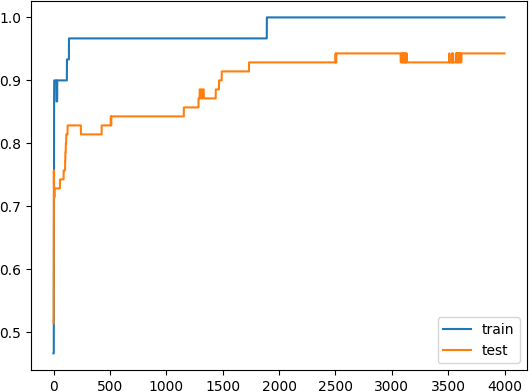
model.add(Dense(500,input\_dim=2,activation='relu',kernel\_regularizer=l2(0.001))) model.add(Dense(1,activation='sigmoid')) model.compile(loss='binary\_crossentropy',optimizer='adam',metrics=['accuracy']) history=model.fit(trainX,trainY,validation\_data=(testX,testY),epochs=4000) pyplot.plot(history.history['accuracy'],label='train') pyplot.plot(history.history['val\_accuracy'],label='test')

pyplot.legend() pyplot.show()

## Output:



By replacing l2 regularizer with l1 regularizer at the same learning rate 0.001 we get the following output.



By applying l1 and l2 regularizer we can observe the following changes in accuracy of both trainig and testing data. The changes in code are also highlighted.

from matplotlib import pyplot

from sklearn.datasets import make\_moons from keras.models import Sequential

from keras.layers import Dense from keras.regularizers import l1\_l2

X,Y=make\_moons(n\_samples=100,noise=0.2,random\_state=1) n\_train=30

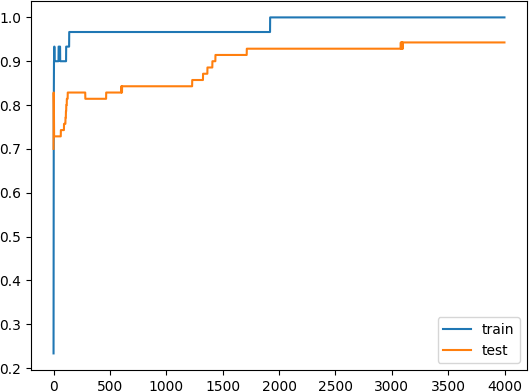
trainX,testX=X[:n\_train,:],X[n\_train:] trainY,testY=Y[:n\_train],Y[n\_train:] #print(trainX)

#print(trainY) #print(testX) #print(testY) model=Sequential()

model.add(Dense(500,input\_dim=2,activation='relu',kernel\_regularizer=l1\_l2(l1=0.001,l2=0.001))) model.add(Dense(1,activation='sigmoid')) model.compile(loss='binary\_crossentropy',optimizer='adam',metrics=['accuracy']) history=model.fit(trainX,trainY,validation\_data=(testX,testY),epochs=4000) pyplot.plot(history.history['accuracy'],label='train') pyplot.plot(history.history['val\_accuracy'],label='test')

pyplot.legend() pyplot.show()

**Output:**



# Practical No. 7

## Aim: Demonstrate recurrent neural network that learns to perform sequence analysis for stock price.

import numpy as np

import matplotlib.pyplot as plt import pandas as pd

from keras.models import Sequential from keras.layers import Dense from keras.layers import LSTM

from keras.layers import Dropout

from sklearn.preprocessing import MinMaxScaler dataset\_train=pd.read\_csv('Google\_Stock\_Price\_Train.csv') #print(dataset\_train) training\_set=dataset\_train.iloc[:,1:2].values #print(training\_set) sc=MinMaxScaler(feature\_range=(0,1)) training\_set\_scaled=sc.fit\_transform(training\_set) #print(training\_set\_scaled)

X\_train=[] Y\_train=[]

for i in range(60,1258): X\_train.append(training\_set\_scaled[i-60:i,0]) Y\_train.append(training\_set\_scaled[i,0])

X\_train,Y\_train=np.array(X\_train),np.array(Y\_train) print(X\_train) print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*') print(Y\_train) X\_train=np.reshape(X\_train,(X\_train.shape[0],X\_train.shape[1],1)) print('\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*') print(X\_train)

regressor=Sequential() regressor.add(LSTM(units=50,return\_sequences=True,input\_shape=(X\_train. shape[1],1)))

regressor.add(Dropout(0.2)) regressor.add(LSTM(units=50,return\_sequences=True)) regressor.add(Dropout(0.2)) regressor.add(LSTM(units=50,return\_sequences=True)) regressor.add(Dropout(0.2)) regressor.add(LSTM(units=50)) regressor.add(Dropout(0.2)) regressor.add(Dense(units=1))

regressor.compile(optimizer='adam',loss='mean\_squared\_error') regressor.fit(X\_train,Y\_train,epochs=100,batch\_size=32) dataset\_test=pd.read\_csv('Google\_Stock\_Price\_Test.csv') real\_stock\_price=dataset\_test.iloc[:,1:2].values

dataset\_total=pd.concat((dataset\_train['Open'],dataset\_test['Open']),ax is=0)

inputs=dataset\_total[len(dataset\_total)-len(dataset\_test)-60:].values inputs=inputs.reshape(-1,1)

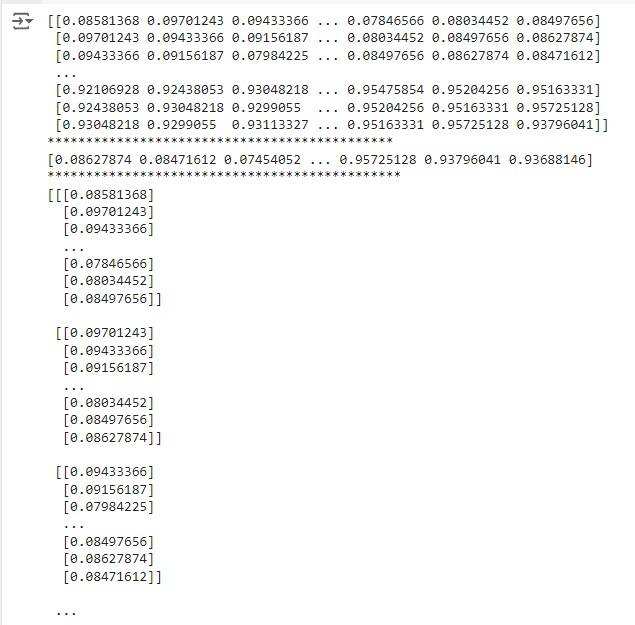
inputs=sc.transform(inputs) X\_test=[]

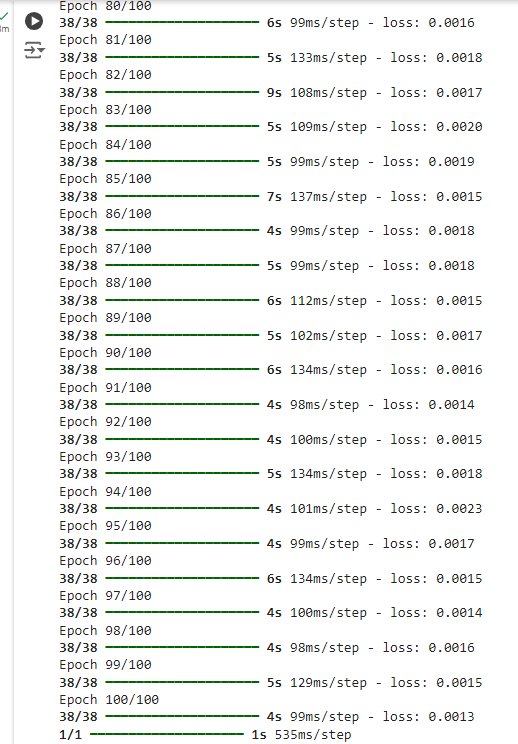
for i in range(60,80): X\_test.append(inputs[i-60:i,0])

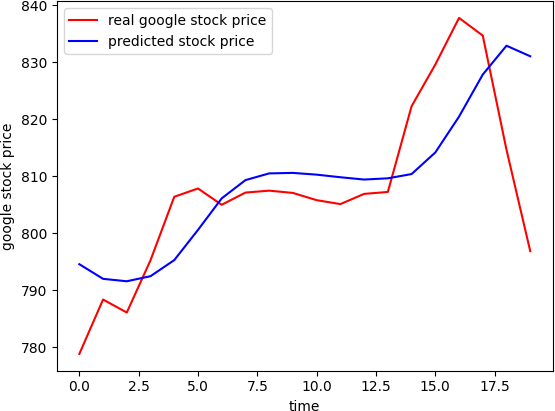
X\_test=np.array(X\_test) X\_test=np.reshape(X\_test,(X\_test.shape[0],X\_test.shape[1],1)) predicted\_stock\_price=regressor.predict(X\_test) predicted\_stock\_price=sc.inverse\_transform(predicted\_stock\_price) plt.plot(real\_stock\_price,color='red',label='real google stock price') plt.plot(predicted\_stock\_price,color='blue',label='predicted stock price')

plt.xlabel('time') plt.ylabel('google stock price') plt.legend()

plt.show()

**Output**





# Practical No:8

## Aim: Performing encoding and decoding of images using deep autoencoder.

import keras

from keras import layers

from keras.datasets import mnist import numpy as np encoding\_dim=32

#this is our input image input\_img=keras.Input(shape=(784,))

#"encoded" is the encoded representation of the input encoded=layers.Dense(encoding\_dim, activation='relu')(input\_img) #"decoded" is the lossy reconstruction of the input decoded=layers.Dense(784, activation='sigmoid')(encoded) #creating autoencoder model autoencoder=keras.Model(input\_img,decoded)

#create the encoder model encoder=keras.Model(input\_img,encoded) encoded\_input=keras.Input(shape=(encoding\_dim,)) #Retrive the last layer of the autoencoder model decoder\_layer=autoencoder.layers[-1]

#create the decoder model decoder=keras.Model(encoded\_input,decoder\_layer(encoded\_input)) autoencoder.compile(optimizer='adam',loss='binary\_crossentropy') #scale and make train and test dataset (X\_train,\_),(X\_test,\_)=mnist.load\_data() X\_train=X\_train.astype('float32')/255.

X\_test=X\_test.astype('float32')/255. X\_train=X\_train.reshape((len(X\_train),np.prod(X\_train.shape[1:]))) X\_test=X\_test.reshape((len(X\_test),np.prod(X\_test.shape[1:]))) print(X\_train.shape)

print(X\_test.shape)

#train autoencoder with training dataset autoencoder.fit(X\_train,X\_train, epochs=50,

batch\_size=256, shuffle=True,

validation\_data=(X\_test,X\_test)) encoded\_imgs=encoder.predict(X\_test) decoded\_imgs=decoder.predict(encoded\_imgs) import matplotlib.pyplot as plt

n = 10 # How many digits we will display plt.figure(figsize=(40, 4))

for i in range(10):

# display original

ax = plt.subplot(3, 20, i + 1) # Indent this line plt.imshow(X\_test[i].reshape(28, 28))

plt.gray()

ax.get\_xaxis().set\_visible(False) ax.get\_yaxis().set\_visible(False) # display encoded image

ax = plt.subplot(3, 20, i + 1 + 20) # Indent this line plt.imshow(encoded\_imgs[i].reshape(8,4))

plt.gray() ax.get\_xaxis().set\_visible(False) ax.get\_yaxis().set\_visible(False) # display reconstruction

ax = plt.subplot(3, 20, 2\*20 +i+ 1) # Indent this line plt.imshow(decoded\_imgs[i].reshape(28, 28)) plt.gray()

ax.get\_xaxis().set\_visible(False) ax.get\_yaxis().set\_visible(False)

plt.show() # This line should not be indented

**OUTPUT:**



# Practical No. 9

## Aim: Implementation of convolutional neural network to predict numbers from number images.

from keras.datasets import mnist

from keras.utils import to\_categorical from keras.models import Sequential

from keras.layers import Dense,Conv2D,Flatten import matplotlib.pyplot as plt

#download mnist data and split into train and test sets (X\_train,Y\_train),(X\_test,Y\_test)=mnist.load\_data() #plot the first image in the dataset plt.imshow(X\_train[0])

plt.show() print(X\_train[0].shape)

X\_train=X\_train.reshape(60000,28,28,1) X\_test=X\_test.reshape(10000,28,28,1) Y\_train=to\_categorical(Y\_train) Y\_test=to\_categorical(Y\_test) Y\_train[0]

print(Y\_train[0]) model=Sequential() #add model layers #learn image features

model.add(Conv2D(64,kernel\_size=3,activation='relu',input\_shape=(28,28, 1)))

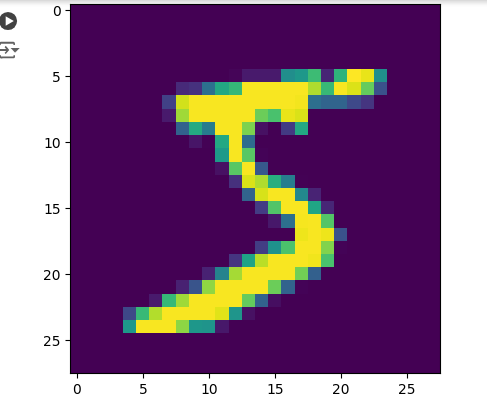
model.add(Conv2D(32,kernel\_size=3,activation='relu')) model.add(Flatten()) model.add(Dense(10,activation='softmax'))

model.compile(optimizer='adam',loss='categorical\_crossentropy',metrics= ['accuracy'])

#train model.fit(X\_train,Y\_train,validation\_data=(X\_test,Y\_test),epochs=3) print(model.predict(X\_test[:4]))

#actual results for 1st 4 images in the test set print(Y\_test[:4])

# Output:





**Practical No. 10**

## Aim: Denoising of images using autoencoder.

import keras

from keras.datasets import mnist from keras import layers

import numpy as np

from keras.callbacks import TensorBoard import matplotlib.pyplot as plt (X\_train,\_),(X\_test,\_)=mnist.load\_data() X\_train=X\_train.astype('float32')/255.

X\_test=X\_test.astype('float32')/255. X\_train=np.reshape(X\_train,(len(X\_train),28,28,1)) X\_test=np.reshape(X\_test,(len(X\_test),28,28,1)) noise\_factor=0.5

X\_train\_noisy=X\_train+noise\_factor\*np.random.normal(loc=0.0,scale=1.0,s ize=X\_train.shape) X\_test\_noisy=X\_test+noise\_factor\*np.random.normal(loc=0.0,scale=1.0,siz e=X\_test.shape)

X\_train\_noisy=np.clip(X\_train\_noisy,0.,1.) X\_test\_noisy=np.clip(X\_test\_noisy,0.,1.) n=10

plt.figure(figsize=(20,2)) for i in range(1,n+1):

ax=plt.subplot(1,n,i) # Indent this line plt.imshow(X\_test\_noisy[i].reshape(28,28)) # Indent this line plt.gray() # Indent this line ax.get\_xaxis().set\_visible(False) # Indent this line ax.get\_yaxis().set\_visible(False) # Indent this line

plt.show() input\_img=keras.Input(shape=(28,28,1))

x=layers.Conv2D(32,(3,3),activation='relu',padding='same')(input\_img) x=layers.MaxPooling2D((2,2),padding='same')(x) x=layers.Conv2D(32,(3,3),activation='relu',padding='same')(x) encoded=layers.MaxPooling2D((2,2),padding='same')(x) x=layers.Conv2D(32,(3,3),activation='relu',padding='same')(encoded) x=layers.UpSampling2D((2,2))(x) x=layers.Conv2D(32,(3,3),activation='relu',padding='same')(x) x=layers.UpSampling2D((2,2))(x) decoded=layers.Conv2D(1,(3,3),activation='sigmoid',padding='same')(x) autoencoder=keras.Model(input\_img,decoded) autoencoder.compile(optimizer='adam',loss='binary\_crossentropy') autoencoder.fit(X\_train\_noisy,X\_train,

epochs=3, batch\_size=128, shuffle=True,

validation\_data=(X\_test\_noisy,X\_test), callbacks=[TensorBoard(log\_dir='/tmo/tb',histogram\_freq=0,write\_graph=F alse)])

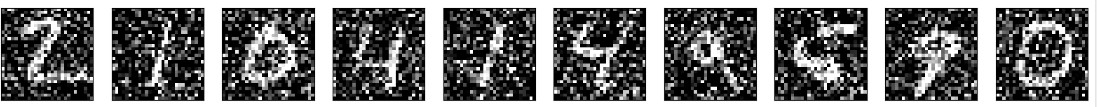
predictions=autoencoder.predict(X\_test\_noisy) m=10

plt.figure(figsize=(20,2)) for i in range(1,m+1):

ax=plt.subplot(1,m,i) # Indent this line plt.imshow(predictions[i].reshape(28,28)) # Indent this line plt.gray() # Indent this line ax.get\_xaxis().set\_visible(False) # Indent this line ax.get\_yaxis().set\_visible(False) # Indent this line

plt.show()

## OUTPUT:



After 3 epochs:

