

Rashtriya Raksha University

**School of Information Technology, Artificial Intelligence & Cyber Security
(SITAICS)**

At- Lavad, Dahegam, Gandhinagar, Gujarat-382305



CERTIFICATE

Program: **B. TECH(CSE)** Subject: **Artificial Intelligence** Enrollment No: **220031101611059** Year: **2022-2026** Semester: **5th Semester**. This is certifying that **Miss Madhuja Deb Adhikari** has satisfactory completed **9** out of **9** experiments in the practical work prescribed by SITAICS **AI** laboratory.

Ms. Aakansha Saxena

SUBJECT INCHARGE

INDEX

ENROLLMENT:220031101611059

NAME: **MADHUJA DEB ADHIKARI**

SUBJECT: **ARTIFICIAL INTELLIGENCE**

SUBJECT(CODE): **5A24ARI**

SR. NO.	DATE	TITLE
1	31/07/24	Develop a case study demonstrating the application of Artificial Intelligence within a specific domain (such as Healthcare, Automation, Finance, etc.). Plagiarism should be less than 10%. Do include introduction, challenges & cons.
2	12/08/24	<ol style="list-style-type: none">1. Study of Prolog Programming-Mention its basics, Facts, Rules, Predicates, Syntax with examples.2. Write simple facts along with queries in Prolog for the following sentences<ul style="list-style-type: none">• 1. Ram likes mango.• 2. Seema is a girl.• 3. Bill likes candy.• 4. Rose is red.• 5. John owns platinum.• 6. David likes mary.
3	21/08/24	<p>(1) Write a program in Prolog along with queries to create a database for hobbies of five different persons.</p> <p>(2) Write a program in Prolog for the following condition- There are two person's named Mary and John. Mary's likings are chocolates, wine and burger. John's likings are Mary, wine and burger.</p> <p>i) Find out the common likings of John and Mary by using a single query.</p> <p>ii) Do John likes Mary and Mary likes John?</p>

		<p>(3) Write a predicate in Prolog along with queries for the following:</p> <ul style="list-style-type: none"> i) Convert Centigrade temperature to Fahrenheit ii) Check if the temperature is below freezing
4	04/09/24	<p>Write a Prolog Program to solve the Monkey Banana Problem.</p> <p>Imagine a room containing a monkey, box and some bananas. These bananas have been hanging from the center of the ceiling. If the monkey is clever enough, the monkey can reach the banana by placing the box directly below the banana and by climbing on that box. The problem is to prove the monkey can reach bananas. The monkey wants it but cannot jump high enough from the floor. At the window of the room there is a box that the monkey can use. The monkey can perform following actions-</p> <ol style="list-style-type: none"> 1. Walk on the floor 2. Climb the box 3. Push the box around 4. Grasp the banana if the monkey standing on the box and the box is directly under the banana
5	11/09/24	<p>Implement any Five (Basic/ Repositioning) operations on the list in prolog.</p> <ol style="list-style-type: none"> 1. Membership Checking. 2. Length Calculation 3. Concatenation. 4. Delete Items 5. Append Items 6. Insert Items 7. Permutation 8. Reverse Items 9. Shift Items 10. Order Items

6	18/09/24	<p>Implement any 4 libraries in python mentioned below:</p> <ul style="list-style-type: none">• Numpy• Numba• NumExpr• Scipy• Pandas• Sympy• Matplotlib
7	25/09/24	Implement Water Jug Problem. “You are given two jugs, a 4-litre one and a 3-litre one. Neither has any measuring markers on it. There is a pump that can be used to fill the jugs with water. How can you get exactly 2 litres of water into 4-litre jug.”
8	09/10/24	Implement Preprocessing steps for Text Dataset. For Text dataset - apply Preprocessing steps tokenization, lemmatization, stemming, stop words removal, bag of words etc.
9	06/11/24	Apply CNN on Image Dataset. Given a set of labeled images of cats and dogs, built or apply a model that could be used to classify a set of new images as cats or dogs.

PRACTICAL – 1

AIM: Develop a case study demonstrating the application of Artificial Intelligence within a specific domain (such as Healthcare, Automation, Finance, etc.). Plagiarism should be less than 10%. Do include introduction, challenges & cons.

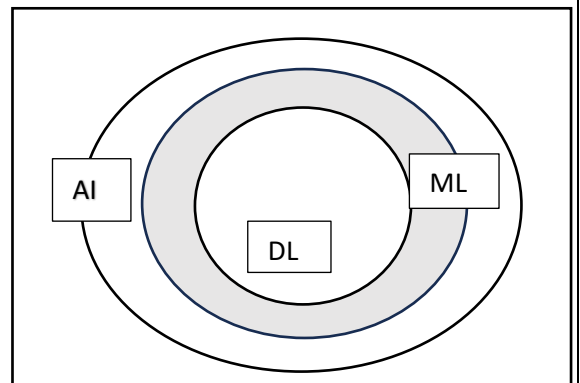
IMPLEMENTATION:

ARTIFICIAL INTELLIGENCE IN HEALTH CARE

INTRODUCTION:

- **What is artificial intelligence:**

Artificial intelligence is a technology that enables computers & machines to simulate human intelligence and problem-solving capabilities. As a field of computer science, artificial intelligence encompasses (and is often mentioned together with) machine learning and deep learning. These disciplines involve the development of AI algorithms, modeled after the decision-making processes of the human brain, that can 'learn' from available data and make increasingly more accurate classifications or prediction over time.



- **AI in health care:**

Artificial intelligence (AI) has the potential to revolutionize the healthcare industry in numerous ways. It can enhance patient care, improve diagnostics, streamline administrative tasks, and aid in drug discovery.

- **APPLICATION:**

1. Medical Imaging and Diagnostics, Image Analysis
2. Drug Discovery and Development, Drug Target Identification
3. Personalized Medicine
4. Predictive Analytics
5. Virtual Health Assistants:
6. Electronic Health Records (EHRs)
7. Drug Adverse Event Monitoring
8. Robotic Surgery
9. Fraud Detection and Billing Optimization
10. Clinical Trial Optimization
11. Patient Risk Stratification
12. Mental Health Support
13. Remote Patient Monitoring

14. Healthcare Operations Management

15. Medical Research

- **ROBOTIC SURGERY:**

Robotic surgery is an advance surgical technique that employs robotic systems to assist surgeons in performing complex procedures with enhanced precision, flexibility, and control. This technology has gained traction in various medical fields, including urology, gynaecology. Hospitals use AI and robots to help with everything from minimally invasive procedure to open heart surgery. Surgeons can control a robot's mechanical arms while seated at a computer console as the robot gives the doctor a three-dimensional, magnified view of the surgical site. The surgeon then leads other team members who work closely with the robot through the entire operation. Robot-assisted surgeries have led to fewer surgery-related complication, less pain and a quicker recovery time.

CHALLENGES:

Despite its benefits, robotic surgery faces several challenges, like:

- **Data privacy and security:** Patient data is highly sensitive and ensuring its security while using AI systems is a significant challenge. Breaches can lead to serve consequences for patients and healthcare organization.
- **Regulatory hurdles:** The healthcare industry is heavily regulated, and navigating the regulatory landscape for AI technologies can be complex and time-consuming.
- **Data quality and standardization:** AI systems require high-quality, standardized data for training. Variability in data sources and formats can hinder the effectiveness of AI algorithms.
- **Integration with Existing Systems:** Incorporating AI solutions into existing healthcare IT systems can be changes in workflows.
- **Bias and fairness:** AI algorithms can inadvertently perpetuate or exacerbate biases present in training data, leading to unequal treatment outcomes across different demographic groups.
- **High costs:** The purchase and maintenance of robotic surgical systems can be prohibitively expensive for many healthcare facilities. This financial burden may limit access to robotic surgery for patients, particularly in underserved areas.
- **Training and skill development:** Surgeons required specialized training to operate system effectively. The learning curve can be steep, and not all surgeons may have access to adequate training program.
- **Limited Availability:** Not all hospitals are equipped with robotic surgeries can vary and some insurers may not cover certain procedures, creating financial barriers for patients.

ADVANTAGES:

- **High quality patient care:** Medical robots support minimally invasive procedures, customized and frequent monitoring for patients with chronic disease, intelligent therapeutics, and social engagement for elderly patients. In aa robots alleviate workloads, nurses and other caregivers can offer patients more empathy and human interaction, which can promote long term well-begin.
- **Enhanced Precision:** Robotic instruments offer greater dexterity and stability compared to human hands, allowing for more precise movements during delicate procedure.
- **Improved visualization:** Robotic systems provide high-definition 3D visualization of the surgical field, enabling surgeons to see intricate details that may not be visible through traditional techniques.
- **Safe work Environment:** To help keep healthcare workers safe, AMRs are used to transport supplies and linens in hospital where pathogen exposure is a risk. Cleaning and disinfection robots limit pathogen exposure while helping reduce hospital acquired infection (HAIs)—and hundreds of healthcare facilities are already using them. Social robot, a type of AMR, also help with heavy lifting, such as moving beds or patients, which reduces physical strain on healthcare workers.

DISADVANTAGES:

- **High initial investment:** The cost of acquiring robotic surgical systems can be substantial, which may deter hospitals from adopting this technology.
- **Potential for over Reliance:** Surgeons may become overly dependent on robotic systems, potentially diminishing their manual surgical skills over time.
- **Longer setup time:** Preparing robotic systems for surgery can take longer than traditional setups, potentially delaying procedures and increasing operating room costs.
- **Risks of complication:** While robotic surgery can reduce certain risks, it is not without complications. Issues such as instrument malfunction or improper use can lead to adverse outcomes.
- **Ethical Concerns:** The use of AI raises ethical questions regarding consent, accountability for decisions made by algorithms, and the potential for dehumanization in patient care.
- **Job Displacement:** Automation of certain tasks may lead to job losses or shifts in roles within the healthcare workforce.

- **Costly operation:** Medical technology is costly- especially if it needs to be built into the physical design of the office. AI is often only available to more advanced medical facilities that have research departments as well as developmental centers. This can be an issue for the smaller medical offices that want to incorporate all that AI has to offer.
- **Risk of Infection:** Just like any procedure, infection can occur. Even robotics in healthcare offices can still pose this potential threat. Issues may surface if the patient begins bleeding unexpectedly when a robot is the only “Professional” present. It could take time before human workers arrived at the scene.

PRACTICAL – 2 [A]

AIM: Study of Prolog Programming-Mention its basics, Facts, Rules, Predicates, Syntax with examples.

IMPLEMENTATION:

Prolog (Programming in Logic) is a logic programming language widely used in artificial intelligence, computational linguistics, and symbolic reasoning. Its key features include the use of logical relationships, pattern matching, and backtracking to solve problems.

BASICS OF PROLOG

1. **Logic-Based Paradigm:** Prolog programs consist of facts, rules, and queries, representing knowledge about a domain.
2. **Declarative Nature:** Instead of specifying how to solve a problem, Prolog focuses on what the problem is.
3. **Backward Chaining:** Prolog works by attempting to prove goals by reasoning backward from them.
4. **Backtracking:** Prolog systematically searches through possible solutions if multiple solutions exist or if an initial attempt fails.

COMPONENTS OF PROLOG:

1. FACTS:

A fact is a predicate expression that makes a declarative statement about the problem domain. Whenever a variable occurs in a prolog expression, it is assumed to be universally quantified. NOTE that all prolog sentences must end with a period.

Syntax

predicate(arguments).

Example:

```
/* sita is a girl */
```

```
girl(sita).
```

2. RULES:

A rule is a predicate expression that uses logical implication($:-$) to describe a relationship among facts. Thus a prolog rule takes the form

left_hand_side $:-$ right_hand_side .

This sentence is interpreted as: left_hand_side $:-$ right_hand_side

.The left_hand_side is restricted to a single, positive, literal, which means it must consist of a positive atomic expression. It cannot be negated and it cannot contain logical connectives.

This notation is known as a Horn clause. In Horn clause logic, the left hand side of the clause is the conclusion, and must be a single positive literal. The right hand side contains the premises. The Horn clause calculus is equivalent to the first-order predicate calculus.

Syntax

head $:-$ body.

- The head is true if the body (conditions) is true.

Example:

```
/* if x is human then x is mortal */  
mortal(x), human(x).
```

3. PREDICATE:

Prolog predicate is the method to contain the argument and return the boolean values such as true or false. It is a function to operate and return given values, variables, or arguments using a prolog programming language.

Example:

```
/* kumud like mango */  
like(kumud,mango).
```

4. QUERIES:

Queries are questions posed to the Prolog system to determine if certain facts or rules hold true.

Syntax

?- goal.

Examples

?- cat(tom). % Is Tom a cat?

?- likes(john, pizza). % Does John like pizza?

PROLOG SYNTAX

1. Variables:

- Start with an uppercase letter or underscore (_).
- Examples: X, Person, _Temp

2. Constants:

- Lowercase letters or numbers.
- Examples: john, pizza, 23.

3. Atoms:

- Represent symbolic constants.
- Examples: apple, x1.

4. Compound Terms:

- Represent structured data.
- Syntax: functor(arguments).
- Example: likes(john, pizza).

5. Lists:

- Represent sequences of elements.
- Syntax: [Head | Tail].
- Example: [1, 2, 3], [H | T].

EXAMPLES IN PROLOG

1. Defining Facts:

parent(john, mary).

parent(mary, susan).

2. Defining Rules:

grandparent(X, Y) :- parent(X, Z), parent(Z, Y).

3. Querying the Knowledge Base:

?- parent(john, mary).

true.

?- grandparent(john, susan).

true.

4. Working with Lists:

member(X, [X | _]). % X is a member if it is the head

member(X, [_ | Tail]) :- member(X, Tail). % Or if it is in the tail

?- member(2, [1, 2, 3]).

true.

5. Recursion:

factorial(0, 1). % Base case

factorial(N, Result) :-

 N > 0,

 N1 is N - 1,

 factorial(N1, SubResult),

 Result is N * SubResult.

?- factorial(5, R).

$R = 120$.

PROLOG EXECUTION WORKFLOW

1. Define facts and rules in a .pl file (e.g., example.pl).
2. Load the file in a Prolog interpreter (e.g., SWI-Prolog) using:
?- consult('example.pl').
3. Execute queries interactively:
?- query.

PRACTICAL – 2 [B]

AIM: Write simple facts along with queries in Prolog for the following sentences

1. Ram likes mango.
2. Seema is a girl.
3. Bill likes candy.
4. Rose is red.
5. John owns platinum.
6. David likes mary.

IMPLEMENTATION:

CODE:

likes(ram ,mango).

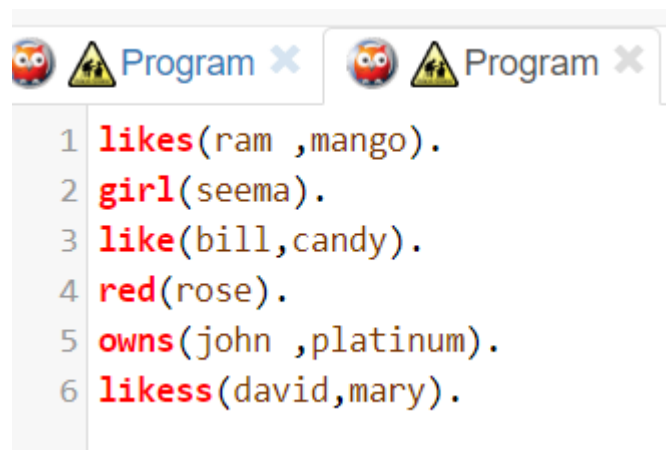
girl(seema).

like(bill,candy).

red(rose).

owns(john ,platinum).

likess(david,mary).

A screenshot of a Prolog code editor window. The window has a title bar with two tabs, each labeled 'Program' with a close button. The code is displayed in a monospaced font with syntax highlighting: 'likes(ram ,mango).', 'girl(seema).', 'like(bill,candy).', 'red(rose).', 'owns(john ,platinum).', and 'likess(david,mary).' are all in red. Line numbers 1 through 6 are visible on the left side of the code area.

```
1 likes(ram ,mango).
2 girl(seema).
3 like(bill,candy).
4 red(rose).
5 owns(john ,platinum).
6 likess(david,mary).
```

QUERIES:



The screenshot shows a Prolog IDE interface with a list of queries and their results. Each query is in a separate row with a gear icon on the left and control icons (download, zoom, close) on the right. The results are displayed below each query.

- Query: `likes(Who,mango).`
Result: **Who** = ram
- Query: `girl(Who).`
Result: **Who** = seema
- Query: `like(bill,What).`
Result: **What** = candy
- Query: `red(What).`
Result: **What** = rose
- Query: `owns(Who,What).`
Result: **What** = platinum,
Who = john
- Query: `likess(Who,Who).`
Result: **false**
- Query: `likess(Who,What).`
Result: **What** = mary,
Who = david
- Query: `likess(Who,mary).`
Result: **Who** = david

At the bottom, there is a search bar with the text `?- likess(Who,mary).` and three tabs: Examples, History, and Solutions. To the right of the tabs is a checkbox labeled "table results" and a blue "Run" button.

PRACTICAL – 3 [A]

AIM: Write a program in Prolog along with queries to create a database for hobbies of five different persons.

IMPLEMENTATION:**CODE:**

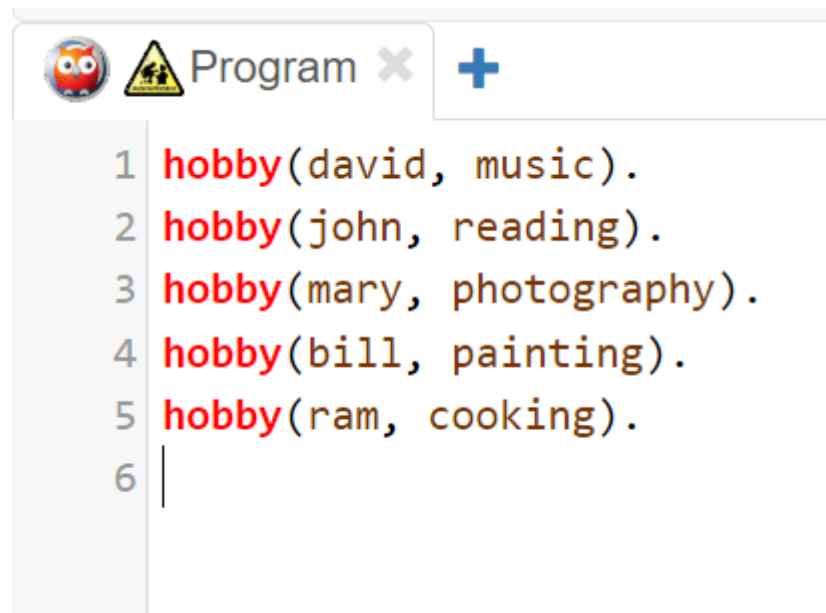
hobby(david,music).

hobby(john,reading).

hobby(mary,photography).

hobby(bill,painting).

Hobby(ram,cooking).



```
1 hobby(david, music).
2 hobby(john, reading).
3 hobby(mary, photography).
4 hobby(bill, painting).
5 hobby(ram, cooking).
6 |
```


PRACTICAL – 3 [B]

AIM: Write a program in Prolog for the following condition-

There are two person's named Mary and John. Mary's likings are chocolates, wine and burger. John's likings are Mary, wine and burger.

- 1) Find out the common likings of John and Mary by using a single query.
- 2) Do John likes Mary and Mary likes John?

IMPLEMENTATION:

likes(mary,wine).

likes(mary,chocolate).

likes(mary,burger).

likes(john,mary).

likes(john,wine).

likes(john,burger).



```
1 likes(mary,wine).
2 likes(mary,chocolate).
3 likes(mary,burger).
4 likes(john,mary).
5 likes(john,wine).
6 likes(john,burger).
```

QUERIES:



likes(john,mary), likes(mary,john)

false

?-

likes(john,mary), likes(mary,john)



likes(mary,X), likes(john,X)

X

wine

burger

?-

likes(mary,X), likes(john,X)

PRACTICAL – 3 [C]

AIM: Write a predicate in Prolog along with queries for the following:

- i) Convert Centigrade temperature to Fahrenheit
- ii) Check if the temperature is below freezing

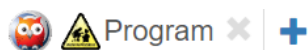
IMPLEMENTATION:

Write a predicate in prolog along with queries for the following:

- i) Convert centigrade temperature to Fahrenheit
- ii) Check if the temperature is below freezing

`c_to_f(C,F):- F is ((C * (9/5) + 32)).`

`Freeze(F):- (F < 32).`



Program



```
1 %Write a predicate in Prolog along with queries for the following:
2 %i) Convert Centigrade temperature to Fahrenheit
3 %ii) Check if the temperature is below freezing
4
5 c_to_f(C, F):- F is ((C * (9/5) + 32)).
6 freeze(F):- (F < 32).
7
```

A

50.0

?- c_to_f(10,A).



freeze(45).

false

?-

freeze(45).



freeze(30).

true

?-

freeze(30).

PRACTICAL – 4

AIM: Write a Prolog Program to solve the Monkey Banana Problem

Imagine a room containing a monkey, box and some bananas. These bananas have been hanging from the center of the ceiling. If the monkey is clever enough, the monkey can reach the banana by placing the box directly below the banana and by climbing on that box. The problem is to prove the monkey can reach bananas. The monkey wants it but can not jump high enough from the floor. At the window of the room there is a box that the monkey can use. The monkey can perform following actions-

1. Walk on the floor
2. Climb the box
3. Push the box around
4. Grasp the banana if the monkey standing on the box and the box is directly under the banana

IMPLEMENTATION:

% Facts

monkey(john). % John is a monkey

banana(yellow). % There is a banana (yellow)

near(john, yellow). % John is near the yellow banana

% Rule: If a monkey is near a banana, it can get it

can_get_banana(Monkey, Banana) :-

 monkey(Monkey),

 banana(Banana),

 near(Monkey, Banana).

OUTPUT:

can_get_banana(Monkey, Banana)

Monkey	Banana	
john	yellow	1

?- can_get_banana(Monkey, Banana)

Examples▲ History▲ ☒ Table results▲ Run!

PRACTICAL – 5

AIM: Implement any Five (Basic/ Repositioning) operations on the list in prolog.

1. Membership Checking.
2. Length Calculation
3. Concatenation.
4. Delete Items
5. Append Items
6. Insert Items
7. Permutation
8. Reverse Items
9. Shift Items
10. Order Items

IMPLEMENTATION:

1. MEMBERSHIP CHECKING

member(X, [X|_]).

member(X, [_|Tail]) :-

member(X, Tail).

Query:

member(3, [1, 2, 3, 4]).

Output:



Program



```

1 member(X, [X|_]).
2 member(X, [_|Tail]) :-
3   member(X, Tail).

```



member(3, [1, 2, 3, 4]).



true

1



?-

member(3, [1, 2, 3, 4]).

Examples▲

History▲

Solutions▲

☒ table results

Run!

2. LENGTH CALCULATION

length_of_list([], 0).

length_of_list([_ | T], N) :- length_of_list(T, N1), N is N1 + 1.

Query:

length_of_list([a, b, c, d], L).

Output:



length_of_list([a, b, c, d], L).



L

4

1

3. CONCATENATION





`concatenate([], L, L).`

`concatenate([H | T], L, [H | R]) :- concatenate(T, L, R).`

Query:

`concatenate([1, 2], [3, 4], Result).`

Output:

 <code>concatenate([1, 2], [3, 4],</code>		  
Result).		
Result		
[1, 2, 3, 4]		1

4. DELETE ITEMS

`delete_item(_, [], []).`





`delete_item(X, [X | T], T).`

`delete_item(X, [H | T], [H | R]) :- delete_item(X, T, R).`

Query:

`delete_item(b, [a, b, c, d], Result).`

Output:

 <code>delete_item(b, [a, b, c, d],</code>		  
Result).		
Result		
[a, c, d]		1





5. APPEND ITEMS

```
append_item(X, [], [X]).
append_item(X, [H | T], [H | R]) :- append_item(X, T, R).
```

Query:

```
append_item(e, [a, b, c, d], Result).
```

Output:

 <code>append_item(e, [a, b, c, d],</code>   	
<code>Result).</code>	
Result	
<code>[a, b, c, d, e]</code>	1





6. INSERT ITEMS

```
insert_at(X, 1, L, [X | L]).
insert_at(X, N, [H | T], [H | R]) :- N > 1, N1 is N - 1, insert_at(X, N1, T, R).
```

Query:

```
insert_at(10, 3, [1, 2, 3, 4, 5], Result).
```

Output:

 <code>insert_at(10, 3, [1, 2, 3, 4, 5],</code>   	
<code>Result).</code>	
Result	
<code>[1, 2, 10, 3, 4, 5]</code>	1


7. PERMUTATION

```
permutation([], []).
permutation(L, [H | T]) :- select(H, L, Rest), permutation(Rest, T).
```

Query:

```
permutation([1, 2, 3], Result).
```

Output:

 <code>permutation([1, 2, 3],</code> ⬇ — ✕	
<code>Result).</code>	
Result	
<code>[1, 2, 3]</code>	1

8. REVERSE ITEMS


`reverse_list([], []).`

`reverse_list([H | T], R) :- reverse_list(T, RT), append(RT, [H], R).`

Query:

`reverse_list([a, b, c, d], Result).`

Output:

 <code>reverse_list([a, b, c, d],</code> ⬇ — ✕	
<code>Result).</code>	
Result	
<code>[d, c, b, a]</code>	1


9. SHIFT ITEMS

`shift_left([H | T], R) :- append(T, [H], R).`

Query:

`shift_left([1, 2, 3, 4], Result).`

Output:

 <code>shift_left([1, 2, 3, 4], Result).</code> ⬇ — ✕	
Result	
<code>[2, 3, 4, 1]</code>	1





10. ORDER ITEMS

`shift_left([H | T], R) :- append(T, [H], R).`

Query:

`shift_left([1, 2, 3, 4], Result).`

Output:

 <code>shift_left([1, 2, 3, 4], Result).</code>   	
Result	
[2, 3, 4, 1]	1

PRACTICAL – 6

AIM: Implement any 4 libraries in python mentioned below:

- Numpy
- Numba
- NumExpr
- Scipy
- Pandas
- Sympy
- Matplotlib

IMPLEMENTATION:

1. Numpy Implementation:

CODE:

```
4. import numpy as np
5. arr=np.array([1,2,3,4,5])
6. print(arr[0]+arr[3])
```

OUTPUT:

5

CODE:

```
import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])

for x in arr:
    for y in x:
        print(y)
```

OUTPUT:

1
2
3
4
5

6

2. Numba implementation:

CODE:

```
import numba
#@numba.jit
def sum_square(x,y):
    return x**2 + y**2
result=sum_square(2,3)
print(result)
```

OUTPUT:

13

3. Numexpr implementation:

CODE:

```
import numexpr as ne
import numpy as np
a = np.random.rand(10000000)
b = np.random.rand(10000000)
ne.evaluate("a + 1")
ne.evaluate("a + b")
```

OUTPUT:

```
array([1.29948935, 0.75361894, 0.99861465, ..., 0.99823354, 0.24335867,
       0.47587374])
```

CODE:

```
ne.evaluate('a*b-4.1*a > 2.5*b')
```

OUTPUT:

```
array([False, False, False, ..., False, False, False])
```

CODE:

```
%%timeit
a * b - 4.1 * a > 2.5 * b
```

OUTPUT:

15.9 ms \pm 1.42 ms per loop (mean \pm std. dev. of 7 runs, 100 loops each)

CODE:

```
%%timeit
ne.evaluate('a*b-4.1*a > 2.5*b')
```

OUTPUT:

4.51 ms \pm 199 μ s per loop (mean \pm std. dev. of 7 runs, 100 loops each)

CODE:

```
%%timeit
ne.evaluate("sin(a) + arcsinh(a/b)")
```

OUTPUT:

38 ms \pm 651 μ s per loop (mean \pm std. dev. of 7 runs, 10 loops each)

CODE:

```
%%timeit
np.sin(a) + np.arcsinh(a / b)
```

OUTPUT:

59.5 ms \pm 1.33 ms per loop (mean \pm std. dev. of 7 runs, 10 loops each)

4. Scipy Implementation :**CODE:**

```
from scipy import constants
print(constants.pi)
```

OUTPUT:

3.141592653589793

CODE:

```
print(constants.kibi)      #1024
print(constants.mebi)     #1048576
print(constants.gibi)      #1073741824
print(constants.tebi)      #1099511627776
```

OUTPUT:

1024

1048576

1073741824

1099511627776

CODE:

```
print(constants.minute)    #60.0
print(constants.hour)      #3600.0
print(constants.day)       #86400.0
print(constants.week)      #604800.0
```

OUTPUT:

60.0

3600.0

86400.0

604800.0

5. Pandas Implementation:**CODE:**

```
import pandas as pd

mydataset = {
    'cars': ["BMW", "Volvo", "Ford"],
    'passings': [3, 7, 2]
}

myvar = pd.DataFrame(mydataset)

print(myvar)
```


OUTPUT:

cars passings

0	BMW	3
1	Volvo	7
2	Ford	2

CODE:

```
print(pd.__version__)
```

OUTPUT:

2.1.4

CODE:

```
a = [1, 7, 2]

myvar = pd.Series(a, index = ["x", "y", "z"])

print(myvar)
```

OUTPUT:

x	1
y	7
z	2

dtype: int64

CODE:

```
import pandas as pd

data = {
    "calories": [420, 380, 390],
    "duration": [50, 40, 45]
}

df = pd.DataFrame(data, index = ["day1", "day2", "day3"])

print(df)
```

OUTPUT:

	calories	duration
day1	420	50
day2	380	40
day3	390	45

CODE:

```
#refer to the named index:  
print(df.loc["day2"])
```

OUTPUT:

calories 380
duration 40
Name: day2, dtype: int64

6. Sympy implementation:**CODE:**

```
from sympy import *  
  
a = Rational(5, 8)  
print("value of a is :" + str(a))  
  
b = Integer(3.579)  
print("value of b is :" + str(b))
```

OUTPUT:

value of a is :5/8
value of b is :3

CODE:

```
sympify("10/5+4/2")
```

OUTPUT:

4

7. Matplotlib implementation:

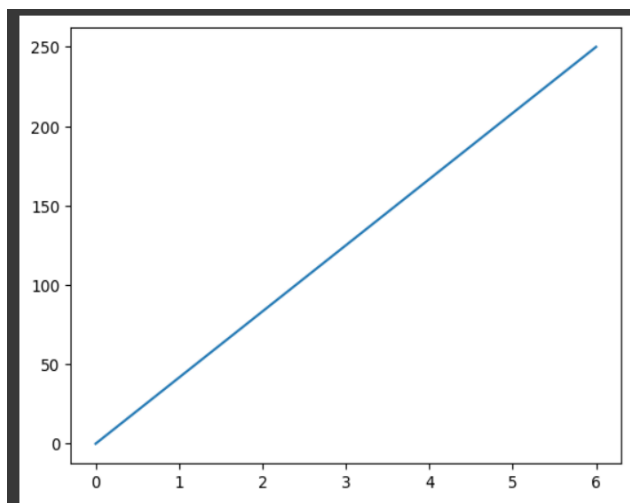
CODE:

```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.array([0, 6])
ypoints = np.array([0, 250])

plt.plot(xpoints, ypoints)
plt.show()
```

OUTPUT:



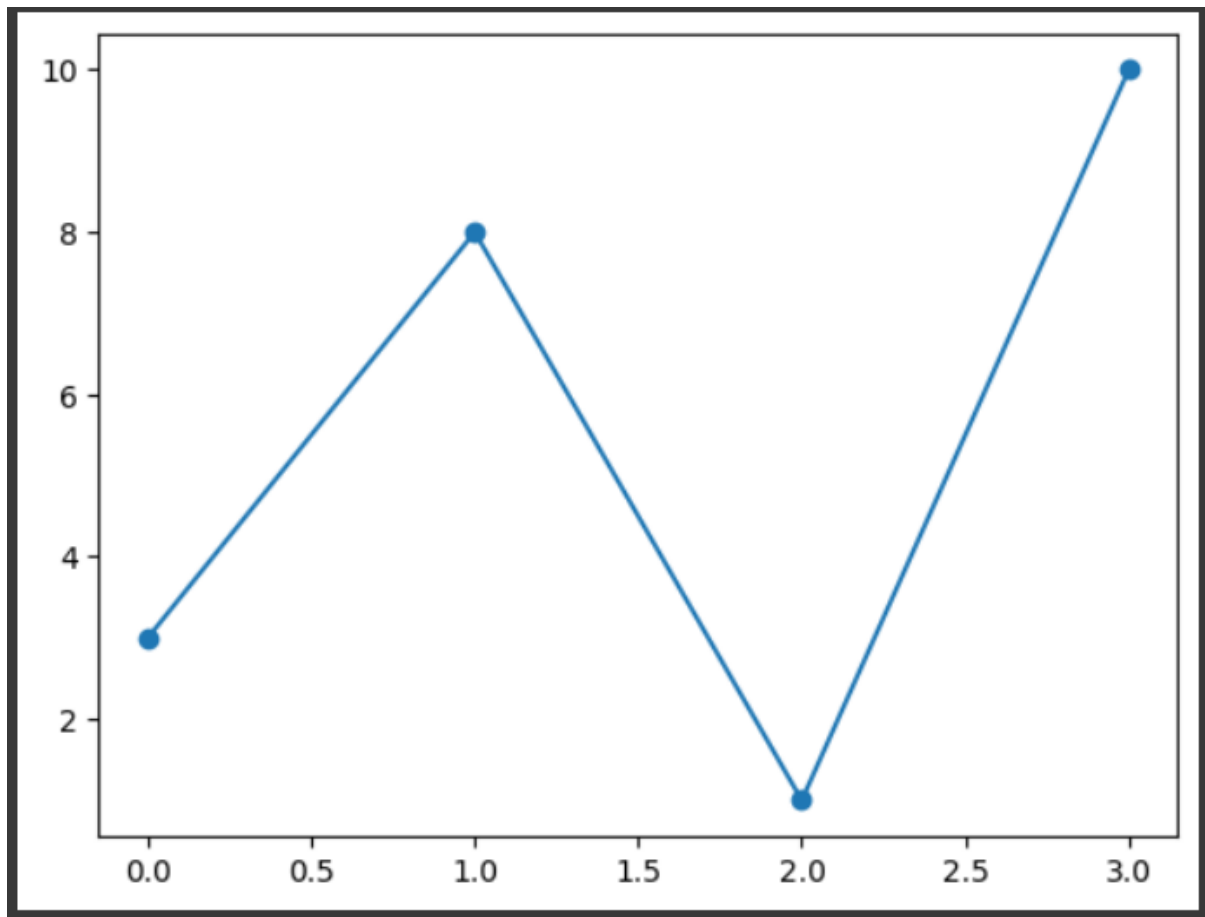
CODE:

```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, marker = 'o')
plt.show()
```

OUTPUT:

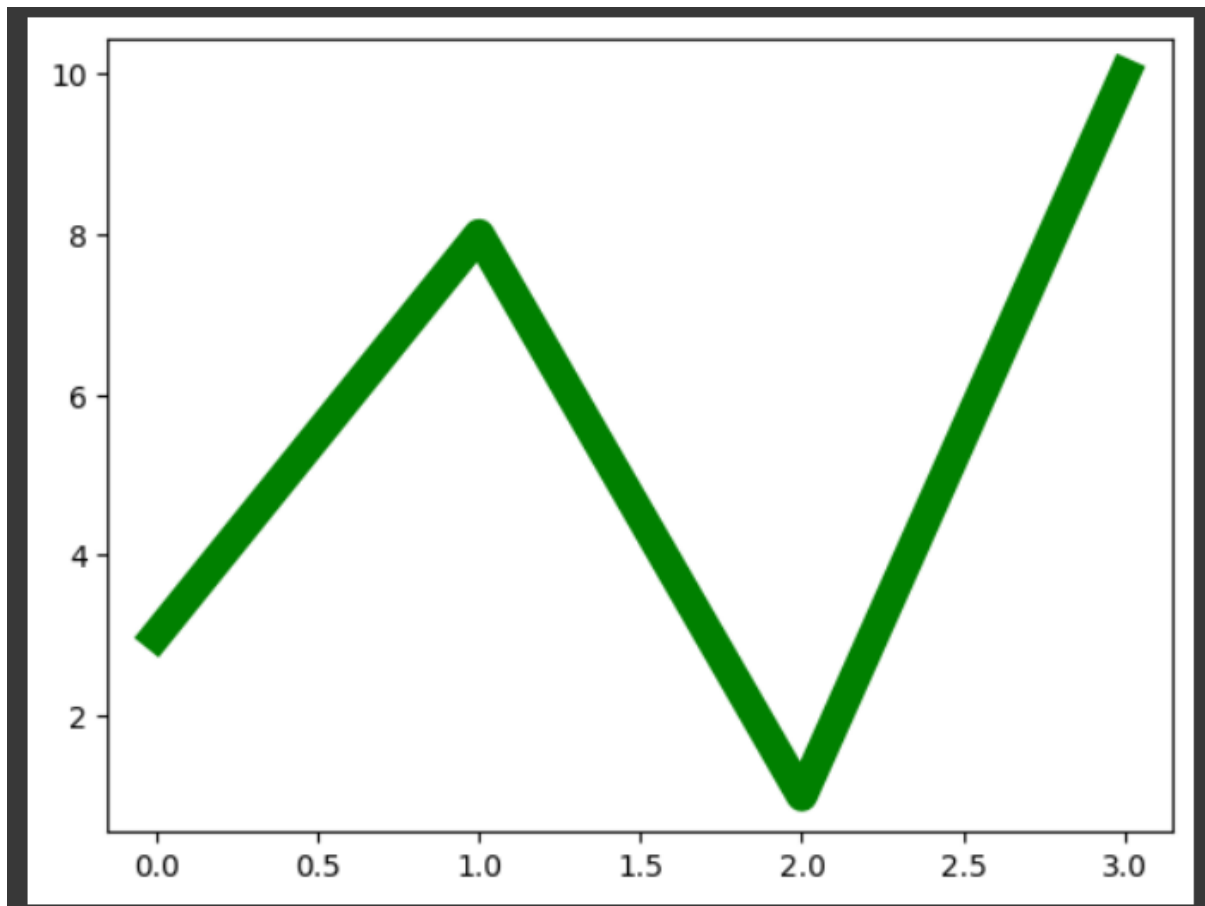
**CODE:**

```
import matplotlib.pyplot as plt
import numpy as np

ypoints = np.array([3, 8, 1, 10])

plt.plot(ypoints, color='green', linewidth='10.5')
plt.show()
```

OUTPUT:



PRACTICAL – 7

AIM: Implement Water Jug Problem. “You are given two jugs, a 4-litre one and a 3-litre one. Neither has any measuring markers on it. There is a pump that can be used to fill the jugs with water. How can you get exactly 2 litres of water into 4-litre jug.”

IMPLEMENTATION:

```
def water_jug_solution():  
    x, y = 0, 0 #x = 4L jug, y = 3L jug  
    steps = []  
  
    x = 4 #Fill 4L jug  
    steps.append((x, y))  
  
    #Pour water from 4L jug into 3L jug until 3L jug is full  
    x, y = 1, 3  
    steps.append((x, y))  
  
    y = 0 #Empty 3L jug  
    steps.append((x, y))  
  
    #Pour remaining 1L from the 4L jug into 3L jug  
    x, y = 0, 1  
    steps.append((x, y))  
  
    x = 4 #Fill 4L jug again  
    steps.append((x, y))  
  
    #Pour water from 4L jug into 3L jug to get 2L in 4L jug  
    x, y = 2, 3  
    steps.append((x, y))
```

```
return steps
```

```
solution = water_jug_solution()
```

```
print("Solution steps:")
```

```
for step in solution:
```

```
    print(step)
```

OUTPUT:

Output

Solution steps:

(4, 0)

(1, 3)

(1, 0)

(0, 1)

(4, 1)

(2, 3)

=== Code Execution Successful ===

PRACTICAL – 8

AIM: Implement Preprocessing steps for Text Dataset. For Text dataset- apply Preprocessing steps tokenization, lemmatization, stemming, stop words removal, bag of words.

IMPLEMENTATION:

```
import nltk

from nltk.tokenize import word_tokenize

from nltk.stem import PorterStemmer, WordNetLemmatizer

from nltk.corpus import stopwords

from sklearn.feature_extraction.text import CountVectorizer

import pandas as pd


text = "I love programming. Python is great and I love Python programming."


# 1. Tokenization

tokens = word_tokenize(text)

print("Tokenized Text:", tokens)


# 2. Removing Stop Words

stop_words = set(stopwords.words('english'))

filtered_tokens = [word for word in tokens if word.lower() not in stop_words]

print("After Stop Words Removal:", filtered_tokens)


# 3. Stemming

stemmer = PorterStemmer()

stemmed_tokens = [stemmer.stem(word) for word in filtered_tokens]

print("Stemmed Tokens:", stemmed_tokens)


# 4. Lemmatization

lemmatizer = WordNetLemmatizer()
```



```
lemmatized_tokens = [lemmatizer.lemmatize(word) for word in stemmed_tokens]
print("Lemmatized Tokens:", lemmatized_tokens)
```

5. Bag of Words

```
vectorizer = CountVectorizer()
X = vectorizer.fit_transform([" ".join(lemmatized_tokens)])
df = pd.DataFrame(X.toarray(), columns=vectorizer.get_feature_names_out())
print("Bag of Words Representation:\n", df)
```

OUTPUT:

```
[Running] python -u "d:\Semester 5\AI\preprocessing\preprocessing.py"
Tokenized Text: ['I', 'love', 'programming', '.', 'Python', 'is', 'great', 'and', 'I', 'love', 'Python', 'programming', '.']
After Stop Words Removal: ['love', 'programming', '.', 'Python', 'great', 'love', 'Python', 'programming', '.']
Stemmed Tokens: ['love', 'program', '.', 'python', 'great', 'love', 'python', 'program', '.']
Lemmatized Tokens: ['love', 'program', '.', 'python', 'great', 'love', 'python', 'program', '.']
Bag of Words Representation:
      great love program python
0         1   2         2     2

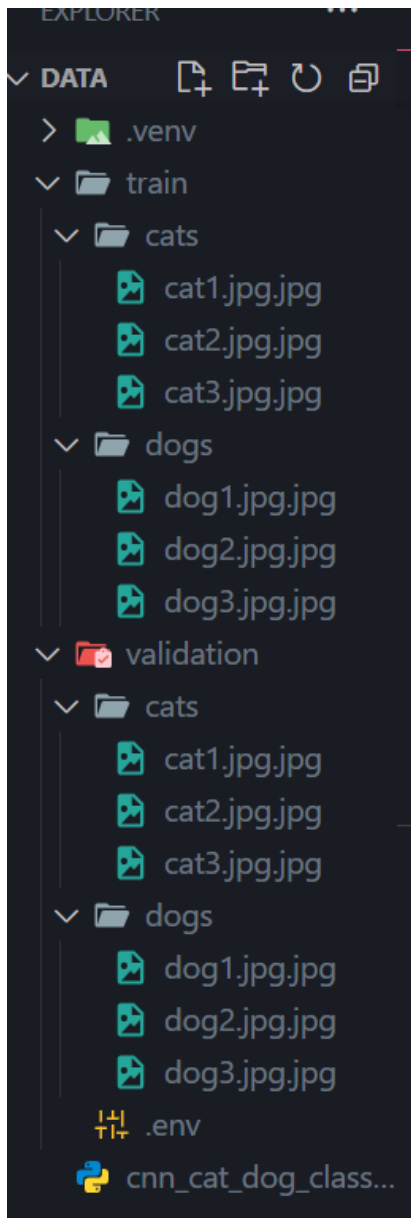
[Done] exited with code=0 in 4.173 seconds
```

PRACTICAL – 9

AIM: Apply CNN on Image Dataset. Given a set of labeled images of cats and dogs, built or apply a model that could be used to classify a set of new images as cats or dogs.

IMPLEMENTATION:

DIRECTORY STRUCTURE:



Cnn_cat_dog_classifier.py:

```
import tensorflow as tf
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
from tensorflow.keras import layers, models

import matplotlib.pyplot as plt

import os

# Directory paths for dataset
train_dir = r'D:\Semester 5\AI\data\train'
validation_dir = r'D:\Semester 5\AI\data\validation'

# Image Preprocessing
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)

validation_datagen = ImageDataGenerator(rescale=1./255)

# Load images from the directories
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='binary'
)

validation_generator = validation_datagen.flow_from_directory(
```

```
validation_dir,  
target_size=(150, 150),  
batch_size=32,  
class_mode='binary'  
)  
  
# Build the CNN Model  
model = models.Sequential([  
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150, 3)),  
    layers.MaxPooling2D(2, 2),  
  
    layers.Conv2D(64, (3, 3), activation='relu'),  
    layers.MaxPooling2D(2, 2),  
  
    layers.Conv2D(128, (3, 3), activation='relu'),  
    layers.MaxPooling2D(2, 2),  
  
    layers.Flatten(),  
    layers.Dense(512, activation='relu'),  
  
    layers.Dense(1, activation='sigmoid') # Binary output  
)  
  
# Compile the model  
model.compile(  
    loss='binary_crossentropy',  
    optimizer='adam',  
    metrics=['accuracy']  
)  
  
# Train the model
```

```
history = model.fit(
    train_generator,
    steps_per_epoch = len(train_generator) // 32, # Number of batches per epoch
    epochs=20,
    validation_data=validation_generator,
    validation_steps = len(validation_generator) // 32 # Number of validation batches per epoch
)
```

```
# Evaluate the model
```

```
val_loss, val_accuracy = model.evaluate(validation_generator)
print(f"Validation Loss: {val_loss}")
print(f"Validation Accuracy: {val_accuracy}")
```

```
# Plot training and validation accuracy
```

```
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()
```

```
# Plot training and validation loss
```

```
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
# Save the trained model  
model.save('cnn_cat_dog_classifier.h5')
```

OUTPUT:

```

PS D:\Semester 5\AI\data> python cnn_cat_dog_classifier.py
2024-11-16 11:52:09.334269: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable 'TF_ENABLE_ONEDNN_OPTS=0'.
2024-11-16 11:52:10.299533: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results due to floating-point round-off errors from different computation orders. To turn them off, set the environment variable 'TF_ENABLE_ONEDNN_OPTS=0'.
Found 6 images belonging to 2 classes.
Found 6 images belonging to 2 classes.
C:\Users\nehap\AppData\Roaming\Python\Python312\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pass an 'input_shape'/'input_dim' argument to a layer. When using Sequential models, prefer using an 'Input(shape)' object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
2024-11-16 11:52:12.585536: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instructions in performance-critical operations.
To enable the following instructions: AVX2 AVX512F AVX512_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
C:\Users\nehap\AppData\Roaming\Python\Python312\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your 'PyDataset' class should call 'super().__init__(**kwargs)' in its constructor. '**kwargs' can include 'workers', 'use_multiprocessing', 'max_queue_size'. Do not pass these arguments to 'fit()', as they will be ignored.
  self._warn_if_super_not_called()
Epoch 1/20
2024-11-16 11:52:14.299757: I tensorflow/core/framework/local_rendezvous.cc:405] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
[[{{(node IteratorGetNext)}}]]
C:\Python312\Lib\contextlib.py:158: UserWarning: Your input ran out of data; interrupting training. Make sure that your dataset or generator can generate at least 'steps_per_epoch * epochs' batches. You may need to use the '.repeat()' function when building your dataset.
  self.gen.throw(value)
1/100 1s 210ms/step - accuracy: 0.5000 - loss: 0.7038 C:\Users\nehap\AppData\Roaming\Python\Python312\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:121: UserWarning: Your 'PyDataset' class should call 'super().__init__(**kwargs)' in its constructor. '**kwargs' can include 'workers', 'use_multiprocessing', 'max_queue_size'. Do not pass these arguments to 'fit()', as they will be ignored.
  self._warn_if_super_not_called()
2024-11-16 11:52:14.856924: I tensorflow/core/framework/local_rendezvous.cc:405] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
[[{{(node IteratorGetNext)}}]]
100/100 2s 6ms/step - accuracy: 0.5000 - loss: 0.7038 - val_accuracy: 0.5000 - val_loss: 2.9248
Epoch 2/20
1/100 1s 44s 452ms/step - accuracy: 0.5000 - loss: 2.9313 2024-11-16 11:52:15.570636: I tensorflow/core/framework/local_rendezvous.cc:405] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
[[{{(node IteratorGetNext)}}]]
100/100 1s 3ms/step - accuracy: 0.5000 - loss: 2.9313 - val_accuracy: 0.5000 - val_loss: 2.2070
Epoch 3/20
100/100 1s 3ms/step - accuracy: 0.5000 - loss: 2.1611 - val_accuracy: 0.5000 - val_loss: 1.1211
Epoch 4/20
1/100 1s 44s 449ms/step - accuracy: 0.5000 - loss: 1.1130 2024-11-16 11:52:16.966247: I tensorflow/core/framework/local_rendezvous.cc:405] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
[[{{(node IteratorGetNext)}}]]
100/100 1s 3ms/step - accuracy: 0.5000 - loss: 1.1130 - val_accuracy: 1.0000 - val_loss: 0.6693
Epoch 5/20

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
python + - [ ] [ ] ... x x

Epoch 5/20
100/100 1s 3ms/step - accuracy: 0.6667 - loss: 0.6818 - val_accuracy: 0.5000 - val_loss: 0.6946
Epoch 6/20
100/100 1s 3ms/step - accuracy: 0.5000 - loss: 0.6946 - val_accuracy: 0.5000 - val_loss: 0.6807
Epoch 7/20
100/100 1s 3ms/step - accuracy: 0.5000 - loss: 0.6701 - val_accuracy: 0.8333 - val_loss: 0.6241
Epoch 8/20
1/100 1s 48s 494ms/step - accuracy: 0.6667 - loss: 0.6598 2024-11-16 11:52:19.980784: I tensorflow/core/framework/local_rendezvous.cc:405] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
[[{{(node IteratorGetNext)}}]]
100/100 1s 3ms/step - accuracy: 0.6667 - loss: 0.6598 - val_accuracy: 0.5000 - val_loss: 0.6683
Epoch 9/20
100/100 1s 3ms/step - accuracy: 0.5000 - loss: 0.6940 - val_accuracy: 1.0000 - val_loss: 0.6006
Epoch 10/20
100/100 1s 3ms/step - accuracy: 0.8333 - loss: 0.6242 - val_accuracy: 0.8333 - val_loss: 0.5579
Epoch 11/20
100/100 1s 3ms/step - accuracy: 0.6667 - loss: 0.5941 - val_accuracy: 0.6667 - val_loss: 0.5608
Epoch 12/20
100/100 1s 3ms/step - accuracy: 0.6667 - loss: 0.6074 - val_accuracy: 0.8333 - val_loss: 0.4960
Epoch 13/20
100/100 1s 3ms/step - accuracy: 0.5000 - loss: 0.6659 - val_accuracy: 1.0000 - val_loss: 0.4621
Epoch 14/20
100/100 1s 3ms/step - accuracy: 1.0000 - loss: 0.4802 - val_accuracy: 0.5000 - val_loss: 0.6172
Epoch 15/20
100/100 1s 3ms/step - accuracy: 0.5000 - loss: 0.6129 - val_accuracy: 0.8333 - val_loss: 0.4424
Epoch 16/20
endezvous.cc:405] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
[[{{(node IteratorGetNext)}}]]
100/100 1s 3ms/step - accuracy: 0.5000 - loss: 0.6673 - val_accuracy: 0.8333 - val_loss: 0.4115
Epoch 17/20
100/100 1s 3ms/step - accuracy: 0.8333 - loss: 0.5463 - val_accuracy: 0.8333 - val_loss: 0.4328
Epoch 18/20
100/100 1s 3ms/step - accuracy: 0.8333 - loss: 0.4374 - val_accuracy: 0.6667 - val_loss: 0.4739
Epoch 19/20
100/100 1s 3ms/step - accuracy: 0.6667 - loss: 0.4792 - val_accuracy: 1.0000 - val_loss: 0.3008
Epoch 20/20
100/100 1s 3ms/step - accuracy: 1.0000 - loss: 0.2976 - val_accuracy: 1.0000 - val_loss: 0.2381
1/1 0s 250ms/step - accuracy: 1.0000 - loss: 0.2381
Validation Loss: 0.23811523616313934
Validation Accuracy: 1.0

```

