```
explain :-
 write('You play X by entering integer positions followed by a period.'),
 nl,
 display([1,2,3,4,5,6,7,8,9]).

playfrom(b) :- win(b, x), write('You win!').
playfrom(b) :- win(b, o), write('I win!').
playfrom(b) :- read(N),
 xmove(b, N, Newb),
 display(Newb),
 orespond(Newb, Newnewb),
 display(Newnewb),
 playfrom(Newnewb).
```

Output:-

```
?- playo.
You play X by entering integer positions followed by a period.
[1,2,3]
[4,5,6]
[7,8,9]

|: 1.
[x,b,b]
[b,b,b]
[b,b,b]

[x,o,b]
[b,b,b]
[b,b,b]

|: 4.
[x,o,b]
[x,b,b]
[b,b,b]

[x,o,b]
[x,b,b]
[o,b,b]

|: 9.
[x,o,b]
[x,b,b]
[o,b,x]

[x,o,b]
[x,o,b]
[o,b,x]

|: 8.
[x,o,b]
[x,o,b]
[o,x,x]

[x,o,o]
[x,o,b]
[o,x,x]

I win!
true .
```

Experiment – 10

Aim:- Write a program to implement the Hangman game using Python.

Code:-

```
import random as r
d=r.choice

def h():
 w=d(["python","java","kotlin","js","hangman","dev","code","YPS"]);g=set();a=6
 print("Hangman!")
 while a:
  print(" ".join(l if l in g else "_" for l in w))
  i=input("Guess:").lower()
  if len(i)!=1 or not i.isalpha()or i in g:continue
  g.add(i);a-=(i not in w)
  if all(l in g for l in w):print("Win!",w);return
 print("Lost!",w)
h()
```

output:

```
Hangman!
_ _ _ _ _ _
Guess:k
_ _ _ _ _ _
Guess:h
_ _ _ h _ _
Guess:p
p _ _ h _ _
Guess:y
p y _ h _ _
Guess:t
p y t h _ _
Guess:o
p y t h o _
Guess:n
Win! python
```

Experiment – 11

Code:-

```
import nltk
from nltk.stem import PorterStemmer as P
from nltk.tokenize import word_tokenize as w
nltk.download('punkt')
nltk.download('punkt_tab')
s=input("Enter: ")
print(" ".join(P().stem(i) for i in w(s)))
```

output:

```
[nltk_data] Downloading package punkt to /home/abhay/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package punkt_tab to /home/abhay/nltk_data...
[nltk_data]   Package punkt_tab is already up-to-date!
Enter: running
run
```

Experiment – 12

Code:-

```
import nltk
from nltk.tokenize import word_tokenize as w
from nltk import pos_tag as p
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
print(p(w(input("Enter: "))))
```

output:

```
[nltk_data] Downloading package punkt to /home/abhay/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /home/abhay/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
Enter: I am going to movie.
[('I', 'PRP'), ('am', 'VBP'), ('going', 'VBG'), ('to', 'TO'), ('movie',
'NN'), ('.', '.')]
```

Experiment – 13

Code:-

```
import nltk
from nltk.stem import WordNetLemmatizer as L
from nltk.tokenize import word_tokenize as w
from nltk.corpus import wordnet as wn
nltk.download('punkt')
nltk.download('omw-1.4')
l=L()
def f(wd):return l.lemmatize(wd,pos=wn.VERB)
print(" ".join(f(i) for i in w(input("Enter: "))))
```

Output:

```
[nltk_data] Downloading package punkt to /home/abhay/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package omw-1.4 to /home/abhay/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
Enter: running
run
```

Experiment – 14

Code:-

```
import nltk
from nltk.classify import NaiveBayesClassifier
from nltk.tokenize import word_tokenize
nltk.download('punkt')
def fs(sen):
    return {word: True for word in word_tokenize(sen)}
td = []
with open("td.txt", "r") as f:
    for line in f:
        parts = line.strip().rsplit(" ", 1)  # Split only at the last space
        if len(parts) == 2 and parts[1] in ["positive", "negative"]:
            td.append((parts[0], parts[1]))
train_set = [(fs(text), label) for text, label in td]
classifier = NaiveBayesClassifier.train(train_set)
sen = input("Enter a sen: ")

print("Sentiment:", classifier.classify(fs(sen)))
```

Output:-

```
[nltk_data] Downloading package punkt to /home/abhay/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
Enter a sentence: this movie is so great!!!
Sentiment: positive
```