Expt. No. 01 Page No. 01 Date 05-02-24.

Experiment-01

AIM:

write predicates one convert centrigrade temperatures to Fahrenheit the other checks if a temperature is below freezing using python.

Hardware/Software Required:

512GB, SSD, Intel core i5 PC, python 3.12.2 software.

CODE:

```
# Function to convert celsius to Fahrenheit
def celsius_to_fahrenheit(celsius_temp):
fahrenheit_temp = (celsius_temp * 9/5) + 32
return fahrenheit_temp

# Function to check if the temperature is below freezing
def is_below_freezing(celsius_temp):
if celsius_temp <0:
return True
else:
return False

# Example usage:
```

Teacher's Signature:

```
celsius_temperature = float (input ("Enter temperature in celsius:"))
Fahrenheit_temperature = celsius_to_Fahrenheit (celsius_temperature)
Print (f"(celsius_temperature)°c is equal to {Fahrenheit_temperature}°F")

if is_below_freezing (celsius_temperature):
Print ("The temperature is below freezing")

else:
print ("The temperature is above freezing")
```

Experiment-2

AIM :

write a program to remove punctuation from the given using python.

Hardware / software Required :

512GB, SSD, Intel core i5 PC, python 3.12.2 software.

code :

```
# Import the string library to get the punc-
tation characters
import string

# Define a Function to remove punctuations
from a given string
def remove_punctuations (input_string):

# Define a translation table to remove punctuation
translator = str.maketrans ('', '' string punctuation

# use the translate method to remove punctuation
return input_string translate (translator)

# Example usage
```

02 Expt. No. | 04 Page No. | Date

```
input_string = "Hello!!!, he said -- and went"
no_punctuation_string = remove_punctuations (input_string)
print (no_punctuation_string)
```

Neeta
12/02/2024

03 Expt. No. 05 Page No. 19-02-24 Date

Experiment-03

AIM:

write a program to sort the sentence in alphabetical order.

Hardware/software Required:

512GB, SSD, Intel core i5 PC, python 3.12.2 software.

code:

```
def sort_sentence_alphabetically (sentence):
# Split the sentence into words
words = sentence.split()
# Sort the words alphabetically
sorted_words = sorted(words)
# Join the sorted words back into a sentence
sorted_sentence = join(sorted_words)
return sorted_sentence

# Example sentence
example_sentence = "python is a powerfull programming
language"

# Sort the example sentence
sorted_example_sentence_alphabetically (example)
```

Teacher's Signature:..............................

```
print(sorted_example_sentence)
```

Monika.
19/02/2024

Experiment-04

AIM:

write a program to implement Tic-Tac-Toe gaming using python.

Hardware / software Required:

512 GB, SSD, Intel core i5 PC, Python 3.12.2 software

code:

```
    def print_board(board):
        for row in board:
            print("|".join(row))
            print("_"*9)

def check_win(board, player):
  win_conditions = [
[board[0][0], board[0][1], board[0],[2]],
[board[1][0], board[1][1], board[1][2]],
[board[2][0], board[2][1], board[2][2]],
[board[0][0], board[1][0], board[2][1]],
[board[0][1], board[1][1], board[2][1],
[board[0][2], board[1][2], board[2][2]],
[board[0][0], board[1][1], board[2][2]],
[board[0][2], board[1][1], board[2][0]],
  ]
```

Teacher's Signature:..........................

```
return [player, player, player] in win_conditions
def player_move (board, player):
while True:
 try:
  row = int (input (f" player {player} enter your
 move row (1-3)")) -1
 col = int (input (f" player {player}, enter your
move columm (1-3):")) -1
  if board [row] [col] = " ";
 board [row] [col] = player
  break
  else:
Print (" This position is already taken. please choose
another.")
except (value error, Index Error):
Print (" Invalid move. please enter row and column
as number from 1 to 3" )
def play_game():
board = [[" " For_ in range (3)] [ For_in range
(3)]]
current_player = "x"
 Print (" welcome to Tic-Tac-Toe!")
 For_in range (9):
Print_board (board)
Player_move (board, current_player)
If check_win (board, current_player):
```

```
print_board(board)
print(f"player {current_player} wins!")
return
current_player = "o" if current_player == 'x' else "x"

print_board(board)
print("It's a tie!")
```

Experiment-05

AIM :

write a program to implement Hangman game using python.

Hardware/Software Required :

512 GB, SSD, Intel core i5 PC, python 3.12.2 software.

code :

```
import random
# List of words to choose from
words = ['python', java, 'kotlin', javascript']
# Randomly select a word from the list
word = random.choice (words)
guessed_word = ['_' for _ in word] # placeholder
for guessed letters
attempts = 8 # Number of allowed incorrect
attempts
guessed_letters = set() # Keep track of guessed
letters
print ("HANGMAN")
while attempts > 0:
print ()
print ("join (guessed_word))
```

```
guess = input("Input a letter:")
# check if the ~~lete~~ letter was aready guessed
if guess in guessed_letters:
 print ("No improvements")
attempts =1
else:
guessed_letters.add(guess)

# check if the guessed letters in the word
 if guess in word:
 for i in range (len(word)):
 if word[i] == guess:
  guessed_word[i] = guess

# check if the word is fully guessed
  if '_' not in guessed_word:
   print ("you guessed the word!")
   print ("you survived!")
   break
 else
       print ("That letter doesn't ~~a~~ appear in
        the world")
       attempts =1
 if attempts == 0:
      print ("you lost!")
```

Teacher's Signature: 11/02/24

06 Expt. No. | 11 Page No. | 11/03/24 Date

Experiment-06

AIM :

Write a program to implement Breath First search Traversal using python.

Hardware/software Required :

512GB, SSD, Intel core i5 PC, python 3.12.2 software.

code :

```
graph = {
'5' = ['3', '7' ],
'3' = ['2','4' ],
'7' : ['8']
'2' : [ ],
'4' : ['8'],
 8' : [ ]
 }
visited = [ ] # List for visited nodes.
queue = [ ] # Initialize a queue

defbfs (visited, graph, node) : # function for BFS
visited.append (node)
 queue.append (node)

while queue : # creating loop to visit each mode
```

```
    m = queue.pop(0)
    Print (m, end=" ")

For neighbour in graph [m]:
if neighbour not in visited:
  visited.append (neighbour)
queue.append (neighbour)

# Driver code
  print ("Following is the Breadth - First search")
  bfs (visited, graph, 's'  # Function calling
```

Experiment - 7

AIM :

write a program to implement water jug problem using python.

Hardware | software Reqeired :

512 GB, SSD, Intel core i5 PC, python 3.12.2 software.

Theory :

you are given 2 jugs with the capacity 'm' and 'n' respectively. They are given empty. There is an unlimited Supply of water. you can either fill the capacity that is less than the given capacity of jugs. Now you are also given 'd'. using the 2 given jugs. you need to come up with a sol^n to have of water them and return the number of steps you took to reach that capacity.

code :

```
From collections import deque
def solution (a, b, target):
  m = {}
  issolvable = False
    path = []
    q = deque()
# Initializing with jugs being empty
    q.append((0,0))

while (len(q) >0):
      # current state
      u = q.popleft()
      if ((u[0], u[1]<0)):
         continue
     path.append([u[0], u[1]])
     m[(u[0], u[1])] = 1
    if (u[0] = = target      [1] = = target):
        issolvable = True

        if (u[0] = = Target):
            if(u[1]! =0):
         path. append[u[0], 0])
      else
          if (u[0]! =0):

                    path.append[0,u [i]]3
          sz = len(path)
          for i in range(sz):
          print("(", path[i][0]. ",",
                   path [i][1], ")")
          break.
```

```
q.append([u[0], b]) # Fill Jug2
q.append([a, u[1]]) # Fill Jug1

For ap in range (max(a, b) + 1):
    c = u[0] + ap
    d = u[1] - ap

if (c == a or (d == 0 and d>= 0)):
    q.append([c, d])

    c = u[0] - ap
    d = u[1] + ap
if ((c == 0 and c>= 0) or d == b)
    q.append([c, d])

  q.append([a, 0])
  q.append([0, b])

(not is solvable):
    print(" solution not possible")

if _ name _ == '_main_'':
   'Jug1, Jug2, target = 4, 5, 7
print(" path from initial state")
     " to solution state "
   solution(Jug1, Jug2, target)
```

08 Expt. No.　16 Page No.　1/04/24 Date

Experiment - 8

AIM :

write a program to stop words for given passage from a text file using NLTK.

Hardware/software Required :

512GB, SSD, Intel core i5 PC, Python 3.12.2 software.

code :

```
Import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

# Make sure to download the stop words set if you
haven't already
nltk.download (punkt')
nltk.download (stopwords')

# Function to remove stop words from a passage of
text.
 def remove_stop_words(file_path):
 # Read text from the file
 with open (file_path 'r' as file:
  text = file.read()
```

Teacher's Signature:

```
# Tokenize the text into words
 words = word_tokenize(text)

# Get the set of English stop words
  stop_words = set(stop words.words('english))

# Remove stop words from the tokenized word
list
filtered_text = [word for word in words if word.
lower() not in stop_words]

# Join words back to form the string without
Stop words
  filtered_text = " join(filtered_text)

return filtered_text

# usage.
File_path = path_to_your_text_file.txt # Replace
with your text file path
 processed_text = remove_stop_words(file_path)
 print(processed_text)
```

Experiment -09

AIM :-
write a program to impiement stenning for a given sentence using NLTK

Hardware / software Required :-
512GB, SSD, Intel core i5 PC, python 3.12.2 software

code :-

```
Import nltk
From nltk. steam import porter stemmer
From nltk. tokenize import word_ tokenize

# Make sure to download the punkt tokenizer model if
you haven't already nltk. download ('punkt')

# Function to stem words in a given sentence
 def stem_ sentence (sentence):
 # Tokenize the sentence into words
  words = word_ tokenize (sentence)

 # create a new porter stemmer
  stemmer = porter stemmer()

 # stem each word in the sentence
```

```
stemmed_words =[stemmer.stem(word) for
words]

# Join the stemmed words back into a string
stemmed _ sentence =' '.join(stemmed words)

return stemmed_sentence)
```

19/04/2024