AI Lab\BFS.py

```python
from collections import deque

class Graph:
    def __init__(self, directed=False):
        self.graph = {}
        self.directed = directed

    def add_edge(self, u, v):
        if u not in self.graph:
            self.graph[u] = []
        self.graph[u].append(v)

        if not self.directed:
            if v not in self.graph:
                self.graph[v] = []
            self.graph[v].append(u)

    def bfs(self, start_vertex):
        visited = set()
        queue = deque([start_vertex])
        visited.add(start_vertex)

        traversal = []

        while queue:
            vertex = queue.popleft()
            traversal.append(vertex)

            for neighbor in self.graph.get(vertex, []):
                if neighbor not in visited:
                    visited.add(neighbor)
                    queue.append(neighbor)

        return traversal

if __name__ == "__main__":
    g = Graph()
    g.add_edge(0, 1)
    g.add_edge(0, 2)
    g.add_edge(1, 2)
    g.add_edge(2, 0)
    g.add_edge(2, 3)
    g.add_edge(3, 3)

    print("BFS traversal starting from vertex 2:")
    print(g.bfs(2))

```


>_pwsh
Python_LocalVC
master ≡ ?1
2ms
>> python -u "d:\SelfRepoClone\Python_LocalVC\AI Lab\BFS.py"
BFS traversal starting from vertex 2:
[2, 0, 1, 3]

AI Lab\Water_Jug_Problem.py

```python
from collections import deque

x = int(input("Enter capacity of Jug 1: "))
y = int(input("Enter capacity of Jug 2: "))
target = int(input("Enter Target to achieve: "))

def gcd(a, b):
    while b:
        a, b = b, a % b
    return a

if target > max(x, y):
    print("No solution possible: Target exceeds capacity of both jugs")
elif target % gcd(x, y) != 0:
    print("No solution possible: Target cannot be measured with these jug sizes")
else:

    visited = set()
    queue = deque([(0, 0, [])])
    solution_found = False

    while queue and not solution_found:
        jug1, jug2, steps = queue.popleft()

        if jug1 == target or jug2 == target:
            print("Solution found:")
            for i, step in enumerate(steps, 1):
                print(f"{i}. {step}")
            solution_found = True

        if (jug1, jug2) in visited:
            continue

        visited.add((jug1, jug2))

        if jug1 < x:
            queue.append((x, jug2, steps + [f"Fill jug 1: ({x}, {jug2})"]))

        if jug2 < y:
            queue.append((jug1, y, steps + [f"Fill jug 2: ({jug1}, {y})"]))

        if jug1 > 0:
            queue.append((0, jug2, steps + [f"Empty jug 1: (0, {jug2})"]))

        if jug2 > 0:
            queue.append((jug1, 0, steps + [f"Empty jug 2: ({jug1}, 0)"]))

        if jug1 > 0 and jug2 < y:
            pour = min(jug1, y - jug2)
            queue.append((jug1 - pour, jug2 + pour, steps + [f"Pour jug 1 to jug 2: ({jug1 - pour}, {jug2 + pour})"]))

```

```
        if jug2 > 0 and jug1 < x:
            pour = min(jug2, x - jug1)
            queue.append((jug1 + pour, jug2 - pour, steps + [f"Pour jug 2 to jug 1: ({jug1 + pour}, {jug2 - pour})"]))

    if not solution_found:
        print("No solution found")
```

```
>_pwsh  Python_LocalVC  master ≡ ?1  56s 844ms
>> python -u "d:\SelfRepoClone\Python_LocalVC\AI Lab\Water_Jug_Problem.py"
Enter capacity of Jug 1: 5
Enter capacity of Jug 2: 3
Enter Target to achieve: 4
Solution found:
1. Fill jug 1: (5, 0)
2. Pour jug 1 to jug 2: (2, 3)
3. Empty jug 2: (2, 0)
4. Pour jug 1 to jug 2: (0, 2)
5. Fill jug 1: (5, 2)
6. Pour jug 1 to jug 2: (4, 3)
```

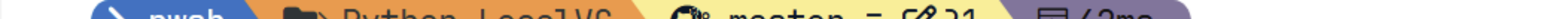
AI Lab\PunctuationRemoval.py

```python
import string

def remove_punctuation(input_string):
    translator = str.maketrans('', '', string.punctuation)
    return input_string.translate(translator)


sample_text = "Hi, I am Under the water! Here its too much raining :("
clean_text = remove_punctuation(sample_text)
print(f"Original: {sample_text}")
print(f"Cleaned: {clean_text}")

```




```
>> python -u "d:\SelfRepoClone\Python_LocalVC\AI Lab\PunctuationRemoval.py"
Original: Hi, I am Under the water! Here its too much raining :(
Cleaned: Hi I am Under the water Here its too much raining
```

AI Lab\sentenceOrder.py

```python
def sort_sentence(sentence):
    words = sentence.split()
    words.sort(key=str.lower)
    sorted_sentence = ' '.join(words)
    return sorted_sentence

input_sentence = input("Enter a sentence: ")
result = sort_sentence(input_sentence)
print("Sorted sentence:", result)
```


>_pwsh
Python_LocalVC
master ≡ ?1
49ms
>> python -u "d:\SelfRepoClone\Python_LocalVC\AI Lab\sentenceOrder.py"
Enter a sentence: The Fool that doesn't belong to this era;
Sorted sentence: belong doesn't era; Fool that The this to

AI Lab\Hangman Game\hangman.py

```python
import random

words = ['python', 'hangman', 'programming', 'computer', 'algorithm', 'database']
word = random.choice(words)
guessed = set()
attempts = 6

while attempts > 0:
    display = ''.join([letter if letter in guessed else '_' for letter in word])
    print(f"\nWord: {display}")
    print(f"Attempts left: {attempts}")
    print(f"Guessed: {', '.join(sorted(guessed))}")

    if display == word:
        print("\nYou won!")
        break

    guess = input("Guess a letter: ").lower()

    if len(guess) != 1 or not guess.isalpha():
        print("Please enter a single letter")
        continue

    if guess in guessed:
        print("Already guessed")
        continue

    guessed.add(guess)

    if guess not in word:
        attempts -= 1
        print("Wrong!")
else:
    print(f"\nGame over! The word was: {word}")
```

```
>_pwsh  Python_LocalVC  master ≡ ?5 ~2  67ms
>> python -u "d:\SelfRepoClone\Python_LocalVC\AI Lab\Hangman Game\hangman.py"

Word: ________
Attempts left: 6
Guessed:
Guess a letter: m
Wrong!

Word: ________
Attempts left: 5
Guessed: m
Guess a letter: e

Word: _______e
Attempts left: 5
Guessed: e, m
Guess a letter: d

Word: d______e
Attempts left: 5
Guessed: d, e, m
Guess a letter: a

Word: da_a_a_e
Attempts left: 5
Guessed: a, d, e, m
Guess a letter: t

Word: data_a_e
Attempts left: 5
Guessed: a, d, e, m, t
Guess a letter: b

Word: databa_e
Attempts left: 5
Guessed: a, b, d, e, m, t
Guess a letter: s

Word: database
Attempts left: 5
Guessed: a, b, d, e, m, s, t

You won!
```

AI Lab\tic_tac_toe.py

```

WIN_LINES = [
    (0, 1, 2), (3, 4, 5), (6, 7, 8),
    (0, 3, 6), (1, 4, 7), (2, 5, 8),
    (0, 1, 2), (2, 4, 6), (0, 4, 8)
]

def print_board(b):
    rows = [b[0:3], b[3:6], b[6:9]]
    print("\n  " + " | ".join(c if c != " " else str(i+1) for i, c in enumerate(b[:3])))
    print(" ---+---+---")
    print("  " + " | ".join(c if c != " " else str(i+1) for i, c in enumerate(b[3:6], start=3)))
    print(" ---+---+---")
    print("  " + " | ".join(c if c != " " else str(i+1) for i, c in enumerate(b[6:9], start=6)))
    print()

def winner(b):
    for a, c, d in {(0,1,2),(3,4,5),(6,7,8),(0,3,6),(1,4,7),(2,5,8),(0,4,8),(2,4,6)}:
        if b[a] != " " and b[a] == b[c] == b[d]:
            return b[a]
    if all(x != " " for x in b):
        return "D"
    return None

def get_move(b, player):
    while True:
        try:
            m = input(f"Player {player}, enter 1-9: ").strip()
            if m.lower() in {"q", "quit", "exit"}:
                return -1
            n = int(m)
            if 1 <= n <= 9 and b[n-1] == " ":
                return n-1
            print("Invalid move.")
        except ValueError:
            print("Enter a number 1-9.")

def game():
    b = [" "] * 9
    turn = "X"
    print_board(b)
    while True:
        idx = get_move(b, turn)
        if idx == -1:
            print("Game aborted.")
            return
        b[idx] = turn
        print_board(b)
```

```
        w = winner(b)
        if w == "X" or w == "O":
            print(f"Player {w} wins!")
            break
        if w == "D":
            print("Draw.")
            break
        turn = "O" if turn == "X" else "X"

def main():
    while True:
        game()
        again = input("Play again? (y/n): ").strip().lower()
        if again not in {"y", "yes"}:
            break

if __name__ == "__main__":
    main()
```

```
>_pwsh  Python_LocalVC  master ≡ ?1  29ms
>> python -u "d:\SelfRepoClone\Python_LocalVC\AI Lab\tic_tac_toe.py"

 1 | 2 | 3
---+---+---
 4 | 5 | 6
---+---+---
 7 | 8 | 9

Player X, enter 1-9: 1

 X | 2 | 3
---+---+---
 4 | 5 | 6
---+---+---
 7 | 8 | 9

Player O, enter 1-9: 9

 X | 2 | 3
---+---+---
 4 | 5 | 6
---+---+---
 7 | 8 | O

Player X, enter 1-9: 3

 X | 2 | X
---+---+---
 4 | 5 | 6
---+---+---
 7 | 8 | O

Player O, enter 1-9: 5

 X | 2 | X
---+---+---
 4 | O | 6
---+---+---
 7 | 8 | O

Player X, enter 1-9: 2

 X | X | X
---+---+---
 4 | O | 6
---+---+---
 7 | 8 | O

Player X wins!
Play again? (y/n): n
```

AI Lab\exp_07\stop_words.py

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

nltk.download('punkt', quiet=True)
nltk.download('stopwords', quiet=True)

with open('D:/SelfRepoClone/Python_LocalVC/AI Lab/exp_07/input.txt', 'r') as file:
    text = file.read()

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

filtered_words = [word for word in words if word.lower() not in stop_words]

filtered_text = ' '.join(filtered_words)

print("Original text:")
print(text)
print("\nFiltered text:")
print(filtered_text)

with open('output.txt', 'w') as file:
    file.write(filtered_text)
```

```
>_pwsh  Python_LocalVC  master ≡ ?2 ~1  2ms                    13,23:19
>> C:\Users\adtya\AppData\Local\Programs\Python\Python312\python.exe -u "d:\SelfRepoClone\Python_LocalVC\AI Lab\exp_07\stop_words.py"
Original text:
The quick brown fox jumps over the lazy dog near the old wooden bridge by the river. In the morning, when the sun rises above the mountains, all the birds begin to sing their beautiful songs. The children were playing with their toys under the big tree while their parents sat on the bench and watched them carefully. There are many books on the shelf in the library where students come to study for their exams. This is a test passage that contains numerous common stop words such as the, is, are, in, on, at, to, for, of, and, a, an, with, by, from, about, between, during, before, after, while, because, although, since, until, unless, when, where, which, who, whom, whose, what, how, why, if, then, else, or, nor, but, yet, so, as, than, through, into, onto, upon, within, without, throughout, and many more.

Filtered text:
quick brown fox jumps lazy dog near old wooden bridge river . morning , sun rises mountains , birds begin sing beautiful songs . children playing toys big tree parents sat bench watched carefully . many books shelf library students come study exams . test passage contains numerous common stop words , , , , , , , , , , , , , , , , , , , , , , , , , , although , since , , unless , , , , , , , whose , , , , , , , else , , , , yet , , , , , , onto , upon , within , without , throughout , many .
```

AI Lab\Exp_08\stemmingnltk.py

```python
import nltk
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize

nltk.download('punkt', quiet=True)

def stem_sentence(sentence):
    stemmer = PorterStemmer()
    words = word_tokenize(sentence)
    stemmed_words = [stemmer.stem(word) for word in words]
    return ' '.join(stemmed_words)

if __name__ == "__main__":
    sentence = "The runners were running and jumping over the obstacles quickly"
    print(f"Original: {sentence}")
    print(f"Stemmed: {stem_sentence(sentence)}")
```

```
>_pwsh  Python_LocalVC  master ≡ ?1  6ms
>> python -u "d:\SelfRepoClone\Python_LocalVC\AI Lab\Exp_08\stemmingnltk.py"
Original: The runners were running and jumping over the obstacles quickly
Stemmed: the runner were run and jump over the obstacl quickli
```

AI Lab\Exp_09\partsOfspeech.py

```python
import nltk
from nltk import pos_tag
from nltk.tokenize import word_tokenize

def ensure_nltk_resources():
    resources = [
        ('tokenizers/punkt', 'punkt'),
        ('tokenizers/punkt_tab', 'punkt_tab'),
        ('taggers/averaged_perceptron_tagger', 'averaged_perceptron_tagger'),
        ('taggers/averaged_perceptron_tagger_eng', 'averaged_perceptron_-
tagger_eng')
    ]

    print("Checking NLTK resources...")
    for path, resource in resources:
        try:
            nltk.data.find(path)
        except LookupError:
            print(f"Downloading missing resource: {resource}")
            try:
                nltk.download(resource, quiet=True)
            except Exception as e:
                print(f"Note: Could not download {resource}. Error: {e}")

def get_pos_tags(sentence):
    tokens = word_tokenize(sentence)
    tagged_tokens = pos_tag(tokens)
    return tagged_tokens

def print_explained_tags(tagged_tokens):
    tag_descriptions = {
        'CC': 'Coordinating conjunction',
        'CD': 'Cardinal number',
        'DT': 'Determiner',
        'EX': 'Existential there',
        'IN': 'Preposition or subordinating conjunction',
        'JJ': 'Adjective',
        'JJR': 'Adjective, comparative',
        'JJS': 'Adjective, superlative',
        'NN': 'Noun, singular or mass',
        'NNS': 'Noun, plural',
        'NNP': 'Proper noun, singular',
        'NNPS': 'Proper noun, plural',
        'RB': 'Adverb',
        'RBR': 'Adverb, comparative',
        'RBS': 'Adverb, superlative',
        'VB': 'Verb, base form',
        'VBD': 'Verb, past tense',
        'VBG': 'Verb, gerund or present participle',
        'VBN': 'Verb, past participle',
        'VBP': 'Verb, non-3rd person singular present',
        'VBZ': 'Verb, 3rd person singular present',
```

```
    }

    print(f"\n{'WORD':<15} {'TAG':<10} {'DESCRIPTION'}")
    print("-" * 50)

    for word, tag in tagged_tokens:
        description = tag_descriptions.get(tag, "Other/Special Symbol")
        print(f"{word:<15} {tag:<10} {description}")

if __name__ == "__main__":
    ensure_nltk_resources()

    text = "The quick brown fox jumps over the lazy dog."

    print(f"\nProcessing sentence: \"{text}\"")

    tags = get_pos_tags(text)

    print("\nRaw Output (List of Tuples):")
    print(tags)

    print_explained_tags(tags)
```

```
>_pwsh  Python_LocalVC  master ≡ ~1  1ms                    Python_LocalVC 3.11.7  25,20:11
>> python -u "d:\SelfRepoClone\Python_LocalVC\AI Lab\Exp_09\partsOfspeech.py"
Checking NLTK resources...

Processing sentence: "The quick brown fox jumps over the lazy dog."

Raw Output (List of Tuples):
[('The', 'DT'), ('quick', 'JJ'), ('brown', 'NN'), ('fox', 'NN'), ('jumps', 'VBZ'), ('over', 'IN'), ('the', 'DT'), ('lazy', 'JJ'), ('dog', 'NN'), ('.'
, '.')]

WORD            TAG         DESCRIPTION
-------------------------------------------------
The             DT          Determiner
quick           JJ          Adjective
brown           NN          Noun, singular or mass
fox             NN          Noun, singular or mass
jumps           VBZ         Verb, 3rd person singular present
over            IN          Preposition or subordinating conjunction
the             DT          Determiner
lazy            JJ          Adjective
dog             NN          Noun, singular or mass
.               .           Other/Special Symbol
```

Exp_10\lemmatization.py

```python
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet

# Download required resources
nltk.download('wordnet', quiet=True)
nltk.download('omw-1.4', quiet=True)
nltk.download('averaged_perceptron_tagger', quiet=True)

def get_wordnet_pos(tag):
    """Convert POS tag to wordnet format"""
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN

def lemmatize_text(text):
    """Lemmatize text with POS tagging"""
    lemmatizer = WordNetLemmatizer()
    tokens = nltk.word_tokenize(text)
    pos_tags = nltk.pos_tag(tokens)

    lemmatized = [lemmatizer.lemmatize(word, get_wordnet_pos(pos)) for word, pos in pos_tags]
    return lemmatized

def simple_lemmatize(words):
    """Simple lemmatization without POS tagging"""
    lemmatizer = WordNetLemmatizer()
    return [lemmatizer.lemmatize(word) for word in words]

if __name__ == "__main__":
    sample_text = "The cats are running and jumping over the fences"

    # With POS tagging
    result = lemmatize_text(sample_text)
    print("Lemmatized (with POS):", ' '.join(result))

    # Without POS tagging
    words = nltk.word_tokenize(sample_text)
    simple_result = simple_lemmatize(words)
    print("Lemmatized (simple):", ' '.join(simple_result))
```

```
>_pwsh  Python_LocalVC  master ≡ ?1  1ms
>> python -u "d:\SelfRepoClone\Python_LocalVC\Exp_10\lemmatization.py"
Lemmatized (with POS): The cat be run and jumping over the fence
Lemmatized (simple): The cat are running and jumping over the fence
```

AI Lab\Exp_11\TextClassification.py

```python
import nltk
from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet

nltk.download('punkt', quiet=True)
nltk.download('wordnet', quiet=True)
nltk.download('averaged_perceptron_tagger', quiet=True)
nltk.download('omw-1.4', quiet=True)

def get_wordnet_pos(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN

def lemmatize_text(text):
    lemmatizer = WordNetLemmatizer()
    tokens = nltk.word_tokenize(text)
    pos_tags = nltk.pos_tag(tokens)

    lemmatized = [lemmatizer.lemmatize(word, get_wordnet_pos(tag)) for word, tag in pos_tags]
    return lemmatized

if __name__ == "__main__":
    sample_text = "The striped bats are hanging on their feet for best"

    print("Original text:", sample_text)
    print("Lemmatized:", lemmatize_text(sample_text))

    test_words = ["running", "ran", "runs", "better", "best", "geese", "mice"]
    lemmatizer = WordNetLemmatizer()

    print("\nWord lemmatization examples:")
    for word in test_words:
        print(f"{word} -> {lemmatizer.lemmatize(word, wordnet.VERB)}")
```

```
>_pwsh  Python_LocalVC  master ≡ ?1  7s 642ms
>> python -u "d:\SelfRepoClone\Python_LocalVC\AI Lab\Exp_11\TextClassification.py"
Original text: The striped bats are hanging on their feet for best
Lemmatized: ['The', 'striped', 'bat', 'be', 'hang', 'on', 'their', 'foot', 'for', 'best']

Word lemmatization examples:
running -> run
ran -> run
runs -> run
better -> better
best -> best
geese -> geese
mice -> mice
```