# Define the documents

document1 = "The quick brown fox jumped over the lazy dog."

document2 = "The lazy dog slept in the sun."

# Step 1: Tokenize the documents

# Convert each document to lowercase and split it into words

tokens1 = document1.lower().split()

tokens2 = document2.lower().split()

# Combine the tokens into a list of unique terms

terms = list(set(tokens1 + tokens2))

# Step 2: Build the inverted index

# Create an empty dictionary to store the inverted index

inverted\_index = {}

# For each term, find the documents that contain it

for term in terms:

documents = []

if term in tokens1:

documents.append("Document 1")

if term in tokens2:

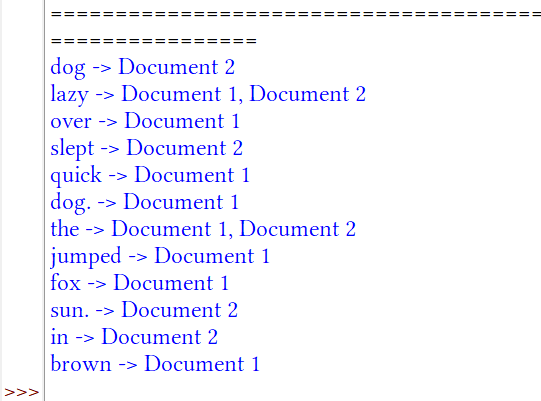
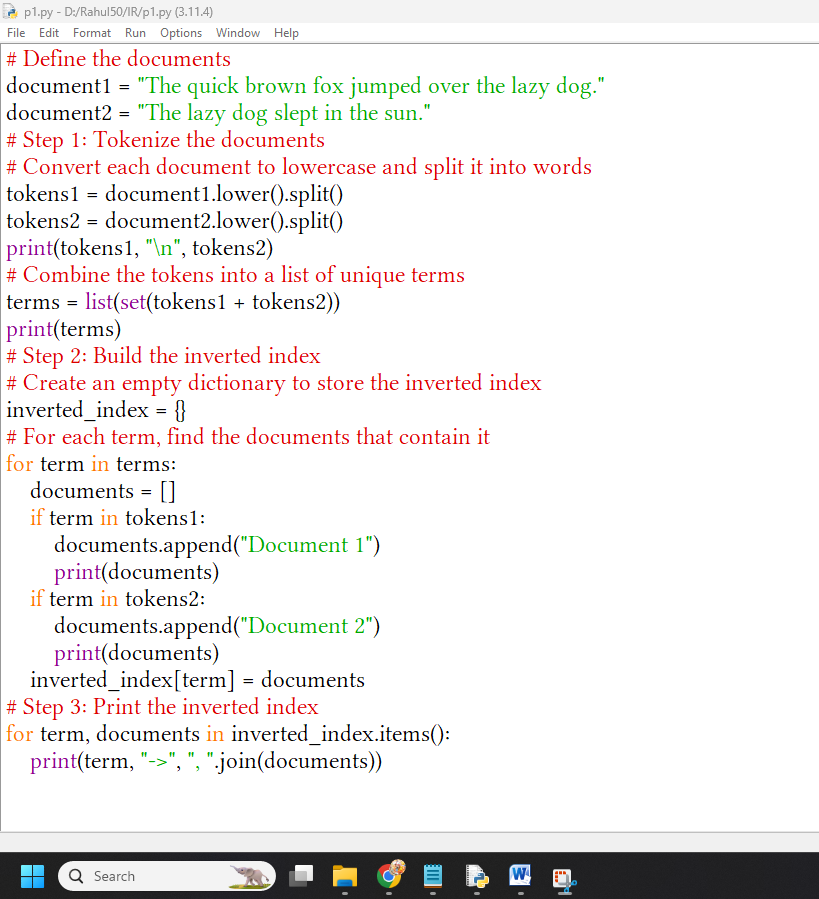
documents.append("Document 2")

inverted\_index[term] = documents

# Step 3: Print the inverted index

for term, documents in inverted\_index.items():

print(term, "->", ", ".join(documents))

documents = {

1: "apple banana orange",

2: "apple banana",

3: "banana orange",

4: "apple"

}

# Function to build an inverted index using dictionaries

def build\_index(docs):

index = {} # Initialize an empty dictionary to store the inverted index

for doc\_id, text in docs.items(): # Iterate through each document and its text

terms = set(text.split())

# Split the text into individual terms

for term in terms: # Iterate through each term in the document

if term not in index:

index[term] = {doc\_id} # If the term is not in the index, create a new set with document ID

else:

index[term].add(doc\_id) # If the term exists, add the document ID to its set

return index # Return the built inverted index

# Building the inverted index

inverted\_index = build\_index(documents)

# Function for Boolean AND operation using inverted index

def boolean\_and(operands, index):

if not operands: # If there are no operands, return all document IDs

return list(range(1, len(documents) + 1))

result = index.get(operands[0], set()) # Get the set of document IDs for the first operand

for term in operands[1:]: # Iterate through the rest of the operands

result = result.intersection(index.get(term, set())) # Compute intersection with sets of document IDs

return list(result) # Return the resulting list of document IDs

# Function for Boolean OR operation using inverted index

def boolean\_or(operands, index):

result = set() # Initialize an empty set to store the resulting document IDs

for term in operands: # Iterate through each term in the query

result = result.union(index.get(term, set())) # Union of sets of document IDs for each term

return list(result) # Return the resulting list of document IDs

# Function for Boolean NOT operation using inverted index

def boolean\_not(operand, index, total\_docs):

operand\_set = set(index.get(operand, set())) # Get the set of document IDs for the operand

all\_docs\_set = set(range(1, total\_docs + 1)) # Create a set of all document IDs

return list(all\_docs\_set.difference(operand\_set)) # Return documents not in the operand set

# Example queries

query1 = ["apple", "banana"] # Query for documents containing both "apple" and "banana"

query2 = ["apple", "orange"] # Query for documents containing "apple" or "orange"

# Performing Boolean Model queries using inverted index

result1 = boolean\_and(query1, inverted\_index) # Get documents containing both terms

result2 = boolean\_or(query2, inverted\_index) # Get documents containing either of the terms

result3 = boolean\_not("orange", inverted\_index, len(documents)) # Get documents not containing "orange"

# Printing results

print("Documents containing 'apple' and 'banana':", result1)

print("Documents containing 'apple' or 'orange':", result2)

print("Documents not containing 'orange':", result3)