```
1.
import re

def validate_pan(pan):
    # Define the regex pattern for a valid PAN card number
    pattern = r'^[A-Z]{5}[0-9]{4}[A-Z]$'

    # Check if the PAN card number matches the pattern
    if re.match(pattern, pan):
        return "Valid"
    else:
        return "Invalid"

# Test cases
print(validate_pan("BIIPR2222K"))  # Output: Valid
print(validate_pan("Azzrt@345l"))  # Output: Invalid
print(validate_pan("biips1234l"))  # Output: Invalid

# ^[A-Z]{5}[0-9]{4}[A-Z]$: This regex pattern checks the following:
# ^[A-Z]{5}: The first 5 characters are uppercase alphabets.
# [0-9]{4}: The next 4 characters are digits.
# [A-Z]$: The last character is an uppercase alphabet.
# The re.match() function checks if the input matches this pattern exactly.
```

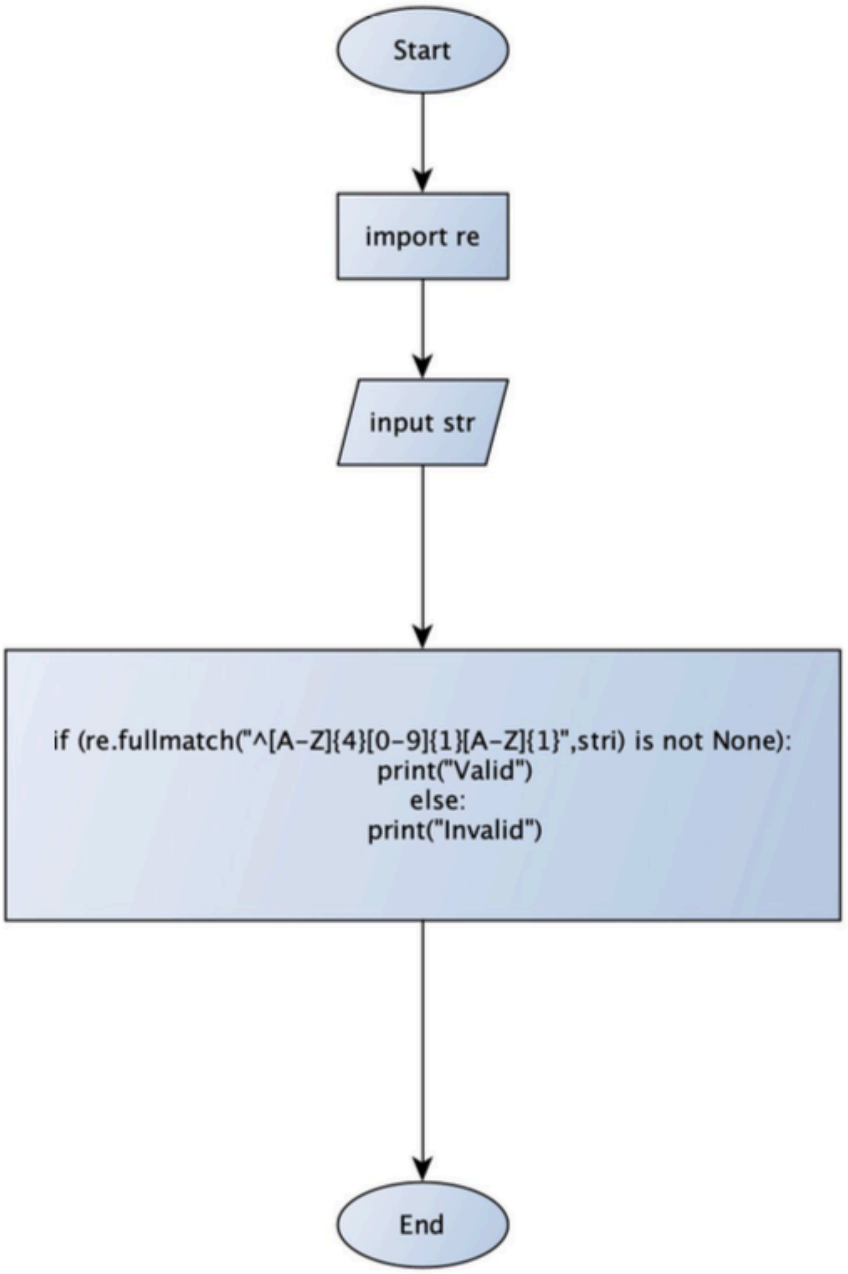
ALGO
Input Prompt:

Prompt the user to enter their PAN (Permanent Account Number).
Read Input:
Store the user input in a variable, n.

Define Regex Pattern:
Create a regular expression pattern to match a valid PAN number:
^[A-Z]{5}\d{4}[A-Z]$
^: Start of the string.
[A-Z]{5}: Match exactly 5 uppercase letters.
\d{4}: Match exactly 4 digits.
[A-Z]: Match exactly 1 uppercase letter.
$: End of the string.

Perform Regex Match:
Use re.fullmatch() to check if the entire string n matches the defined pattern.

Validation Check:
If the match is found:
Print a message stating the PAN number is valid
Otherwise:
Print a message stating the PAN number is invalid.
'''


Start
import re
input str
if (re.fullmatch("^[A-Z]{4}[0-9]{1}[A-Z]{1}",stri) is not None):
print("Valid")
else:
print("Invalid")
End

2.

```
import re

def validate_password(password):
    # Check for minimum and maximum length
    if len(password) < 8 or len(password) > 16:
        return "Invalid"

    password_pattern =
r"^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[_@$])[A-Za-z\d_@$]{8,16}$"

    if re.match(password_pattern, password):
        return "Valid"
    else:
        return "Invalid"

# Test cases
print(validate_password("Password123@"))  # Expected Output: Valid
print(validate_password("pass@"))         # Expected Output: Invalid (too short, no
uppercase)
print(validate_password("Password"))      # Expected Output: Invalid (no digit, no
special character)
print(validate_password("P@ssword12345")) # Expected Output: Valid
```

**'''algo**
**Input Prompt:**
**Prompt the user to enter their password.**

**Read Input:**
**Store the user input in a variable, n.**

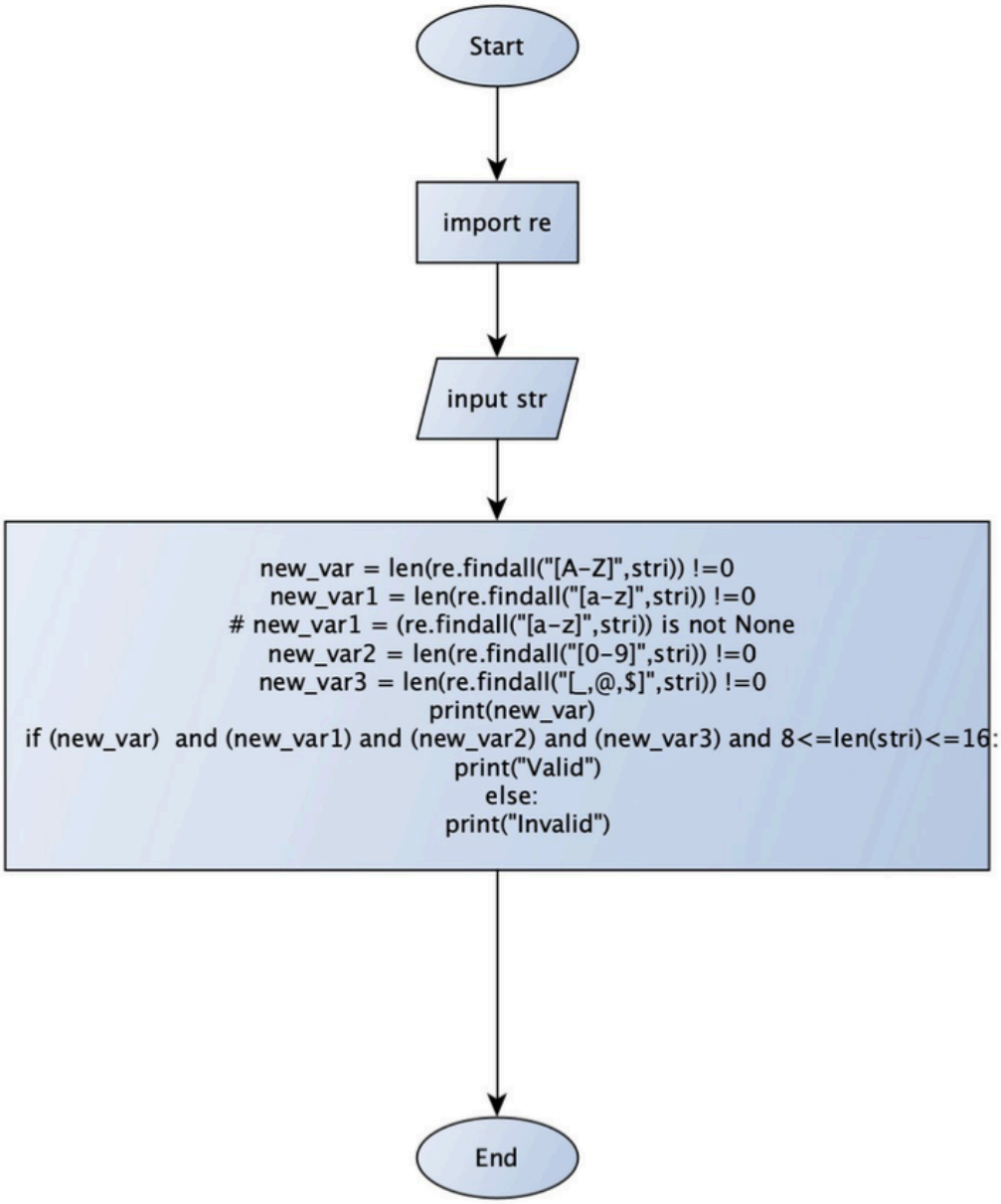
**Define Regex Pattern:**
**Create a regular expression pattern to match a valid password:**
**^(?=.+[A-Z])(?=.+\d)(?=.+[_@$])[a-zA-Z\d_@$]{8,16}$**

**^: Start of the string.**
**(?=.+[A-Z]): Assert that there is at least one uppercase letter somewhere in the string.**
**(?=.+\d): Assert that there is at least one digit somewhere in the string.**
**(?=.+[_@$]): Assert that there is at least one of the special characters _, @, or $ somewhere in the string.**
**[a-zA-Z\d_@$]{8,16}: Ensure the string is 8 to 16 characters long and contains only the specified characters.**
**$: End of the string.**

**Perform Regex Match:**
**Use re.fullmatch() to check if the entire string n matches the defined pattern.**

**Validation Check:**
**If matches is not None (indicating a match is found):**
**Print a message stating the password is valid.**

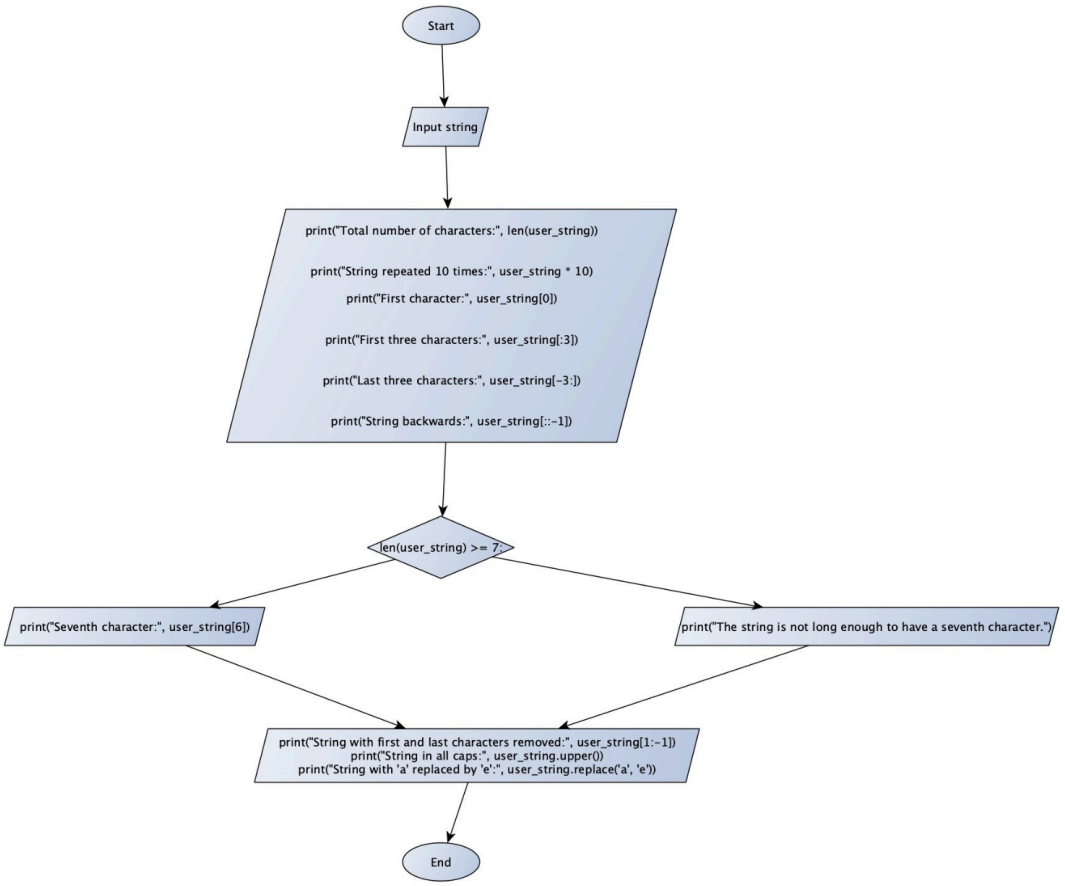
**Otherwise:**
**Print a message stating the password is invalid.**
**'''**


Start
import re
input str
new_var = len(re.findall("[A-Z]",stri)) !=0
new_var1 = len(re.findall("[a-z]",stri)) !=0
# new_var1 = (re.findall("[a-z]",stri)) is not None
new_var2 = len(re.findall("[0-9]",stri)) !=0
new_var3 = len(re.findall("[_,@,$]",stri)) !=0
print(new_var)
if (new_var) and (new_var1) and (new_var2) and (new_var3) and 8<=len(stri)<=16:
print("Valid")
else:
print("Invalid")
End

**3.**

```
user_string = input("Enter a string: ")
print("Total number of characters:", len(user_string))
print("String repeated 10 times:", user_string * 10)
print("First character:", user_string[0])
print("First three characters:", user_string[:3])
print("Last three characters:", user_string[-3:])
print("String backwards:", user_string[::-1])
if len(user_string) >= 7:
   print("Seventh character:", user_string[6])
else:
   print("The string is not long enough to have a seventh character.")
print("String with first and last characters removed:", user_string[1:-1])
print("String in all caps:", user_string.upper())
print("String with 'a' replaced by 'e':", user_string.replace('a', 'e'))
```

**• Prompt the user to enter a string and store it in user_string.**
**• Display Total Number of Characters:**
**• Use len() to calculate and print the total number of characters in user_string.**
**• Display String Repeated 10 Times:**
**• Print user_string repeated 10 times using user_string * 10.**
**• Display First Character:**
**• Access and print the first character of user_string using user_string[0].**
**• Display First Three Characters:**
**• Slice and print the first three characters of user_string using user_string[:3].**
**• Display Last Three Characters:**
**• Slice and print the last three characters of user_string using user_string[-3:].**
**• Display String Backwards:**
**• Reverse and print user_string using the slicing user_string[::-1].**
**• Display Seventh Character (if String is Long Enough):**
**• Check if the length of user_string is 7 or more.**
**• If yes, print the seventh character using user_string[6].**
**• If no, print a message indicating the string is not long enough to have a seventh character.**
**• Display String with First and Last Characters Removed:• Print user_string with the first and last characters removed by using user_string[1:-1].**
**• Display String in All Uppercase:**
**• Print user_string converted to uppercase using user_string.upper().**
**• Display String with 'a' Replaced by 'e':**
**• Use user_string.replace('a', 'e') to replace all occurrences of 'a' with 'e' and print the result.**


Start
Input string
print("Total number of characters:", len(user_string))
print("String repeated 10 times:", user_string * 10)
print("First character:", user_string[0])
print("First three characters:", user_string[:3])
print("Last three characters:", user_string[-3:])
print("String backwards:", user_string[::-1])
len(user_string) >= 7:
print("Seventh character:", user_string[6])
print("The string is not long enough to have a seventh character.")
print("String with first and last characters removed:", user_string[1:-1])
print("String in all caps:", user_string.upper())
print("String with 'a' replaced by 'e':", user_string.replace('a', 'e'))
End

4.

```
def contains_007(nums):
    code = [0, 0, 7]

    for num in nums:
        if num == code[0]:  # Check if the number matches the first element in the 'code' sequence
            code.pop(0)     # Remove the first element from 'code'
        if not code:        # If 'code' is empty, it means 0, 0, 7 sequence was found
            return True

    return False  # Return False if 'code' sequence is not found

# Test cases
print(contains_007([1, 2, 4, 0, 0, 7, 5]))  # Output: True
print(contains_007([1, 0, 2, 4, 0, 5, 7]))  # Output: True
print(contains_007([1, 7, 2, 0, 4, 5, 0]))  # Output: False
```

Start

Import the re module:

```
import re
```

Define the contains_007(nums) function:

Function to check if the list contains the sequence 007.

Convert List to String:

Convert the list nums to a string num_str using ''.join(map(str, nums)).

Define the Regex Pattern:

Create the regex pattern 0.*0.*7.

Search for Pattern in String:

Use re.search(r'0.*0.*7', num_str).

Return Result:

If the pattern is found, return True.

Otherwise, return False.

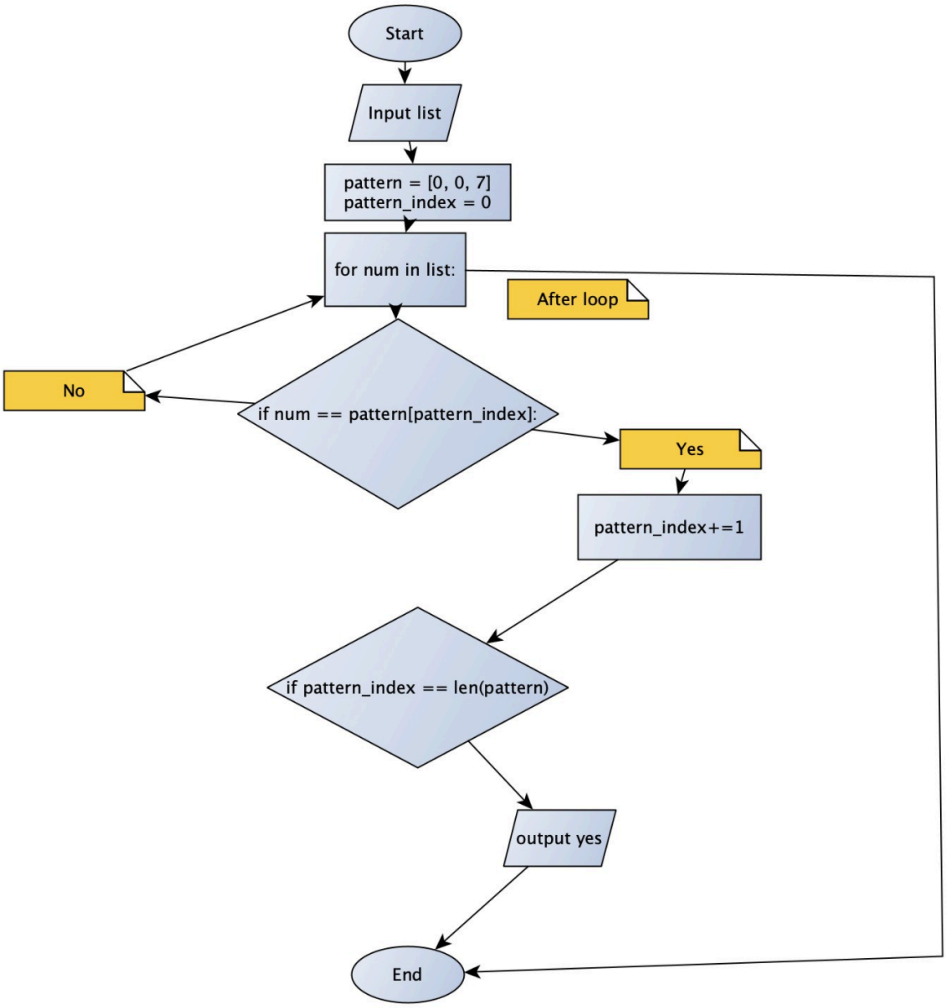
Example Usage:

Call contains_007 with lists and print the result:

```
print(contains_007([1, 2, 4, 0, 0, 7, 5]))

print(contains_007([1, 0, 2, 4, 0, 5, 7]))

print(contains_007([1, 7, 2, 0, 4, 5, 0]))
'''
```


Start
Input list
pattern = [0, 0, 7]
pattern_index = 0
for num in list:
After loop
No
if num == pattern[pattern_index]:
Yes
pattern_index+=1
if pattern_index == len(pattern)
output yes
End

**5.**

```
def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(n**0.5) + 1):
        if n % i == 0:
            return False
    return True

def twin_primes(limit):
    for num in range(3, limit - 1, 2):  # Start from 3, consider only odd numbers
        if is_prime(num) and is_prime(num + 2):
            print(f"{num}, {num + 2} - Twin Prime")

# Run the function for limit 1000
twin_primes(1000)
```

**Start**

**Define prime(x) function:**

**Loop through potential divisors:**

**For each i from 2 to x-1:**

**If x is divisible by i (i.e., x % i == 0):**

**Return False (indicating x is not a prime number).**

**Return True (indicating x is a prime number if no divisors are found).**

**Define twin_prime(limit) function:**

**Initialize an empty list list to store twin primes (though it's not used later).**

**Loop through odd numbers from 3 to limit, skipping even numbers (as they can't be prime):**

**For each num in this range:**

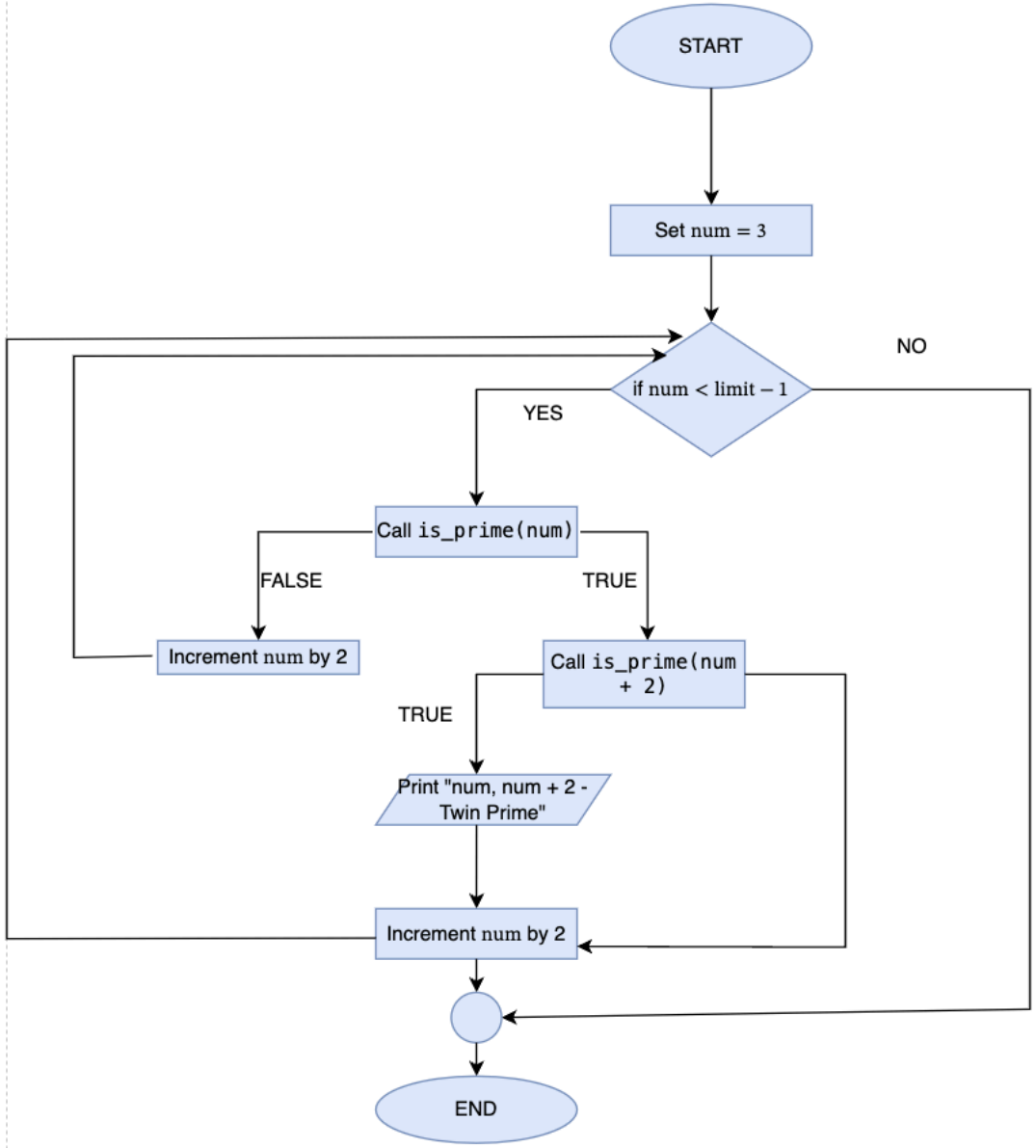
**Check if both num and num + 2 are prime by calling the prime function.**

**If both are prime:**

**Print that num and num + 2 are twin primes.**

**Execute twin_prime function:**

**Call twin_prime(1000) to find and print all twin primes less than 1000.'''**


START
Set num = 3
if num < limit – 1
NO
YES
Call is_prime(num)
FALSE
TRUE
Increment num by 2
Call is_prime(num + 2)
TRUE
Print "num, num + 2 - Twin Prime"
Increment num by 2
END

6

```
import string

def is_pangram(sentence):
    # Convert the sentence to lowercase and use a set to store unique letters
    letters = set(sentence.lower())

    # Check if all letters in the alphabet are in the set of characters from the sentence
    return set(string.ascii_lowercase).issubset(letters)

# Test cases
print(is_pangram("The quick brown fox jumps over the lazy dog"))  # Output: True
print(is_pangram("Hello, World!"))

'''
```

Import the re module:

Import the re module to use regular expressions.

Define the function is_pangram(input_string):

This function will check if the input string is a pangram.

Convert to Lowercase:

Convert the input_string to lowercase to ensure the check is case-insensitive.

Find Unique Alphabetic Characters:

Use re.findall with the regex pattern [a-z] to find all alphabetic characters in the string.

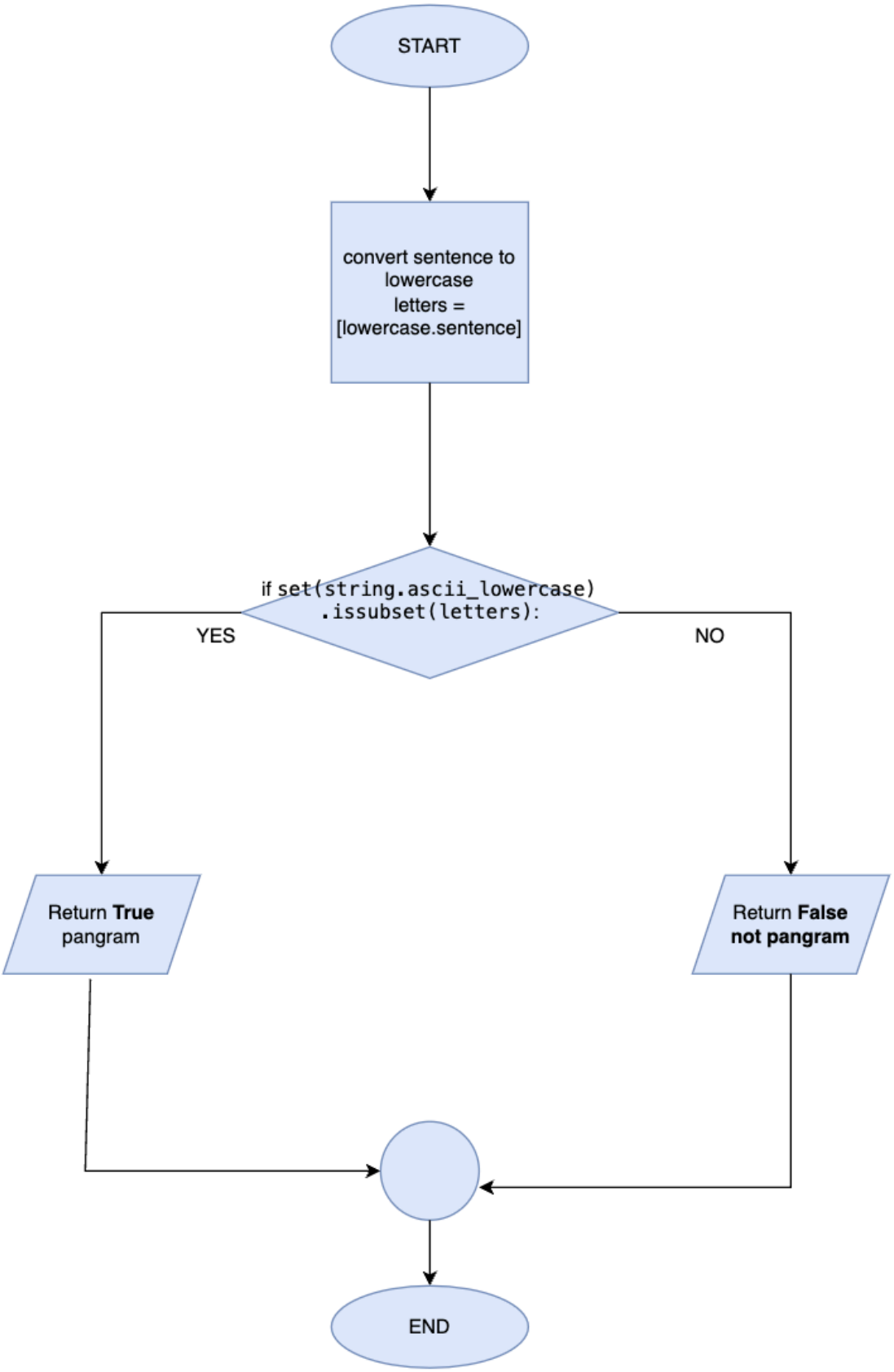
Convert the list of found characters to a set to get unique characters.

Check if All Letters are Present:

Check if the length of the set of unique characters is 26 (the number of letters in the English alphabet).

Return the Result:

Return True if the length is 26, otherwise return False.


START
convert sentence to lowercase
letters = [lowercase.sentence]
if set(string.ascii_lowercase) .issubset(letters):
YES
NO
Return True pangram
Return False not pangram
END

7.

```
import re

def extract_emails(filename):
    with open(filename, 'r') as file:
        content = file.read()

    # Regular expression pattern for email
    email_pattern = r'[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}'

    # Find all email addresses in the file content
    emails = re.findall(email_pattern, content)

    return emails

# Example usage
# Assuming the file is named 'textfile.txt' and contains the sample text
emails = extract_emails('/Users/vedikagoyal/Desktop/vs code/PAT 2/textfile.txt')
print(emails)
```

Start

Import the re module:

import re

Define the text variable:

Store the given text in a variable called text.

Define the regex pattern:

Create the regex pattern [a-zA-Z0-9]+@[a-zA-Z0-9]+\.com.

Find all matches using re.finditer:

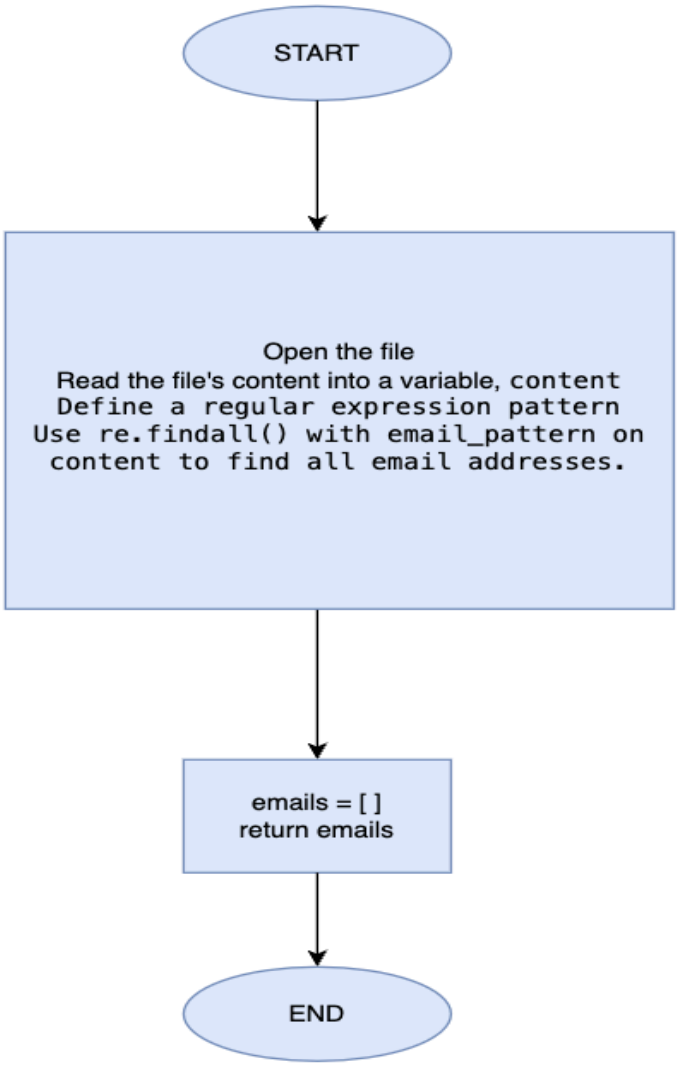
f = re.finditer(r'[a-zA-Z0-9]+@[a-zA-Z0-9]+\.com', text)

Iterate over matches:

for i in f:

Print each match: print(i)

End'''


START
Open the file
Read the file's content into a variable, content
Define a regular expression pattern
Use re.findall() with email_pattern on
content to find all email addresses.
emails = [ ]
return emails
END

**8.**

```
def count_file_content(filename):
    with open(filename, 'r') as file:
        lines = file.readlines()  # Read all lines in the file

    # Count lines, words, and characters
    line_count = len(lines)
    word_count = sum(len(line.split()) for line in lines)
    char_count = sum(len(line) for line in lines)

    return line_count, word_count, char_count

# Example usage
# Assuming the file is named 'textfile.txt'
line_count, word_count, char_count = count_file_content('/Users/vedikagoyal/Desktop/vs code/PAT 2/hehehe.txt')
print(f"Lines: {line_count}")
print(f"Words: {word_count}")
print(f"Characters: {char_count}")
```

**Start**

**Open File:**

**Open the file named filename in read mode.**

**Read Lines:**

**Read all the lines of the file into a list called text.**

**Count Lines:**

**Calculate the number of lines by determining the length of the text list and store it in num_lines.**

**Join Text:**

**Join all lines in the text list into a single string called joined_text.**

**Split Words:**

**Split the joined_text string into words using the split() method and store the result in a list called words.**

**Count Words:**

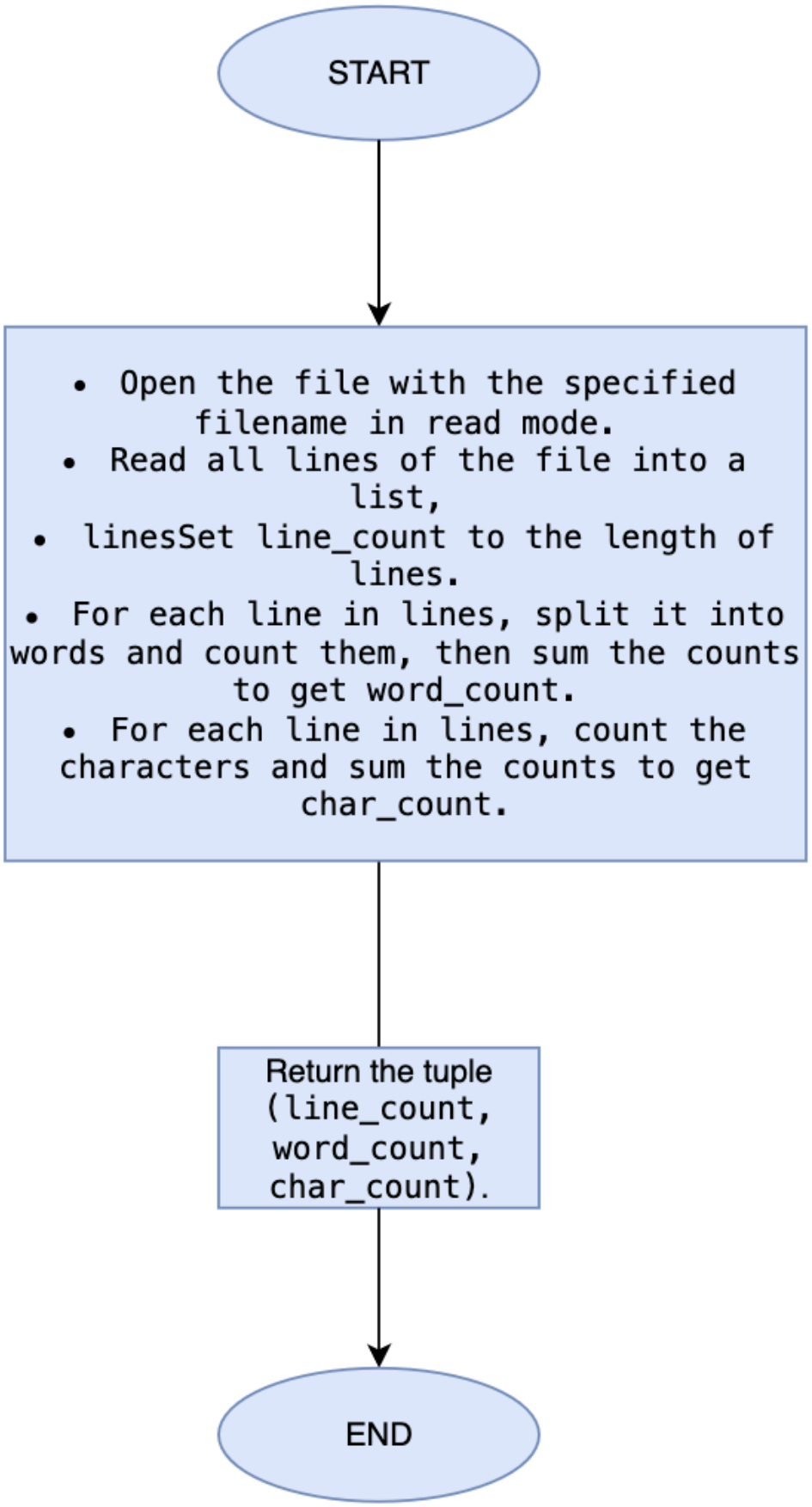
**Calculate the number of words by determining the length of the words list and store it in num_words.**

**Count Characters:**

**Calculate the number of characters by determining the length of the joined_text string and store it in num_chars.**

**Print Results:**

**Print the values of num_lines, num_words, and num_chars with appropriate labels.**


START
• Open the file with the specified filename in read mode.
• Read all lines of the file into a list,
• linesSet line_count to the length of lines.
• For each line in lines, split it into words and count them, then sum the counts to get word_count.
• For each line in lines, count the characters and sum the counts to get char_count.
Return the tuple (line_count, word_count, char_count).
END

9.

```
def count_word_occurrences(filename, word):
    with open(filename, 'r') as file:
        content = file.read().lower()  # Read the content and convert to lowercase for case-insensitive search

    # Count occurrences of the word
    word_count = content.split().count(word.lower())

    return word_count

# Example usage
# Assuming the file is named 'textfile.txt' and we are searching for the word "example"
word = "word"
count = count_word_occurrences('/Users/vedikagoyal/Desktop/vs code/PAT 2/word.txt', word)
print(f"The word '{word}' occurs {count} times in the file.")
```

**Prompt User for Input:**

**Ask the user to enter the word they want to search.**

**Read Input:**

**Store the user's input in a variable word.**

**Open the File:**

**Open the file named 1.txt in read mode.**

**Read File Content:**

**Read the entire content of the file into a variable text.**

**Convert to Lowercase:**

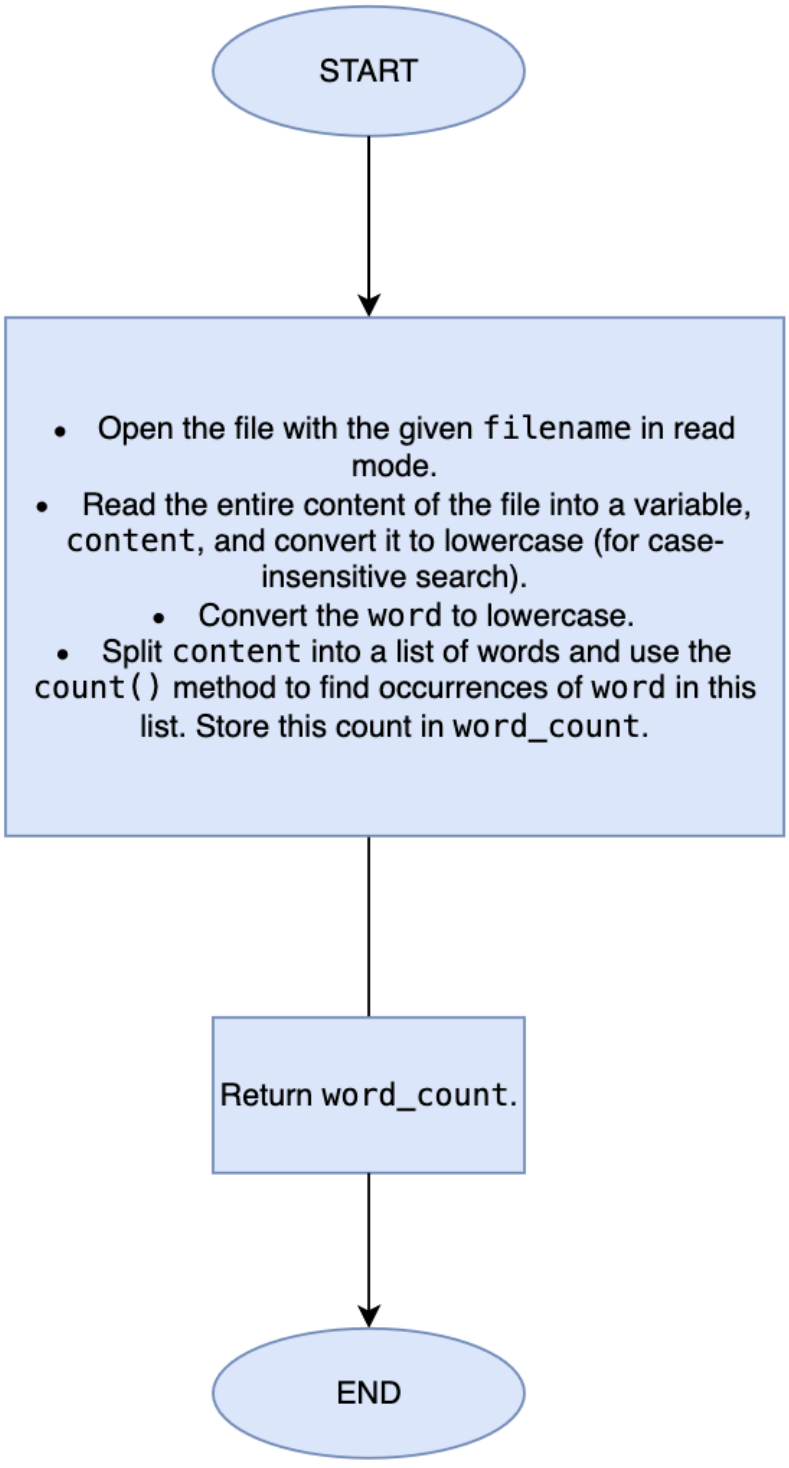
**Convert both the file content text and the input word to lowercase to ensure a case-insensitive search.**

**Count Occurrences:**

**Use the count method to find the number of occurrences of the word in the file content.**

**Print Result:**

**Print the number of occurrences.'''**


START
Open the file with the given filename in read mode.
Read the entire content of the file into a variable, content, and convert it to lowercase (for case-insensitive search).
Convert the word to lowercase.
Split content into a list of words and use the count() method to find occurrences of word in this list. Store this count in word_count.
Return word_count.
END