**Exp - 8 REGULAR EXPRESSIONS AND FILE HANDELING**

1. Regular Expressions in Python

A Regular Expression (RegEx) is a tool used to search, match, or manipulate text using specific patterns. Python provides the built-in re module to work with regular expressions.

To use regular expressions in Python, you must first import the re module using: import re

**Key Functions in re Module**:

1.re.match() checks for a match only at the beginning of the string.

2.re.search() scans the entire string and returns the first match found.

3.re.findall() returns a list of all matching substrings.

4.re.sub() is used to replace parts of the string that match a pattern.

5.re.split() is used to split a string based on a regular expression.

Common Metacharacters and Special Sequences:

In regular expressions:

A dot (.) matches any single character except newline.

A caret (^) matches the beginning of a string.

A dollar sign ($) matches the end of a string.

Square brackets ([]) are used to match a set of characters.

\d matches any digit (0-9), and \D matches any non-digit.

\w matches any alphanumeric character and underscore, while \W matches non-alphanumeric characters.

\s matches whitespace, and \S matches non-whitespace.

The plus sign (+) matches one or more occurrences.

The asterisk (\*) matches zero or more occurrences.

The question mark (?) matches zero or one occurrence.

Curly braces ({n}) specify an exact number of matches.

**Examples**:

1.Matching a specific word at the start of a string:

import re

result = re.match("Hello", "Hello world!")

print(result.group()) # Output: Hello

2.Finding a phone number in text:

text = "Contact: 9876543210"

match = re.search(r'\d{10}', text)

print(match.group()) # Output: 9876543210

3.Finding all digits in a string:

text = "a1 b2 c3"

print(re.findall(r'\d', text)) # Output: ['1', '2', '3']

4.Replacing digits with #:

text = "Room 101"

print(re.sub(r'\d+', '#', text)) # Output: Room #

5.Splitting text based on a comma:

text = "apple,banana,grape"

print(re.split(",", text)) # Output: ['apple', 'banana', 'grape']

……………………………..

2. File Handling in Python

File handling in Python refers to the process of reading from and writing to files. It allows data to be stored permanently on disk and accessed later.

To handle files, Python uses the open() function, which takes two arguments:

1. The file name

2. The mode of operation (like read or write)

File Modes:

'r' is used to open a file for reading. The file must exist.

'w' is used to open a file for writing. It will create a new file if it doesn’t exist or overwrite an existing one.

'a' is used to append data to the end of the file.

'x' creates a new file and throws an error if the file already exists.

'b' opens a file in binary mode.

't' is the default mode and opens files in text mode.

Reading from a File:

To read from a file, you open it in read mode:

f = open("sample.txt", 'r')

content = f.read()

print(content)

f.close()

You can also read one line at a time using readline():

f = open("sample.txt", 'r')

print(f.readline())

f.close()

Or read all lines into a list using readlines():

f = open("sample.txt", 'r')

lines = f.readlines()

for line in lines:

print(line.strip())

f.close()

Writing to a File:

To write data to a file, you open it in write ('w') or append ('a') mode:

f = open("sample.txt", 'w')

f.write("This is a new file.")

f.close()

Appending to the file instead of overwriting:

f = open("sample.txt", 'a')

f.write("\nThis is additional content.")

f.close()

Using with Statement:

The with statement automatically closes the file after the block ends. This is a recommended way to handle files.

with open("sample.txt", 'r') as f:

print(f.read())

Checking File Existence and Deleting Files:

You can check if a file exists using the os module:

import os

if os.path.exists("sample.txt"):

print("File exists.")

else:

print("File not found.")

To delete a file:

import os

os.remove("sample.txt")

……………………………………..

Codes —-

**#REGULAR EXPRESSIONS**

PROGRAM-1

import re

string="She sells sea shells on the sea shore"

pattern1="She"

if re.match(pattern1,string):

print("match found")

else:

print("match not found")

pattern2="show"

if re.match(pattern2,string):

print("match found")

else:

print("match not found")

#OUTPUT

match found

match not found

PROGRAM 2

import re

string="She sells sea shells on the sea shore"

patternl="sea"

repl="ocean"

new string=re.sub(pattern1,repl,string,1)

print(new\_string)

#OUTPUT

she sells ocean shells on the sea shore

**#FILE HANDLING**

\*PROGRAM-1 (EX-14.1)

def sed(pattern, replacement, input\_filename, output\_filename):

try:

with open(input filename, 'r') as infile:

content = infile.read()

modified content = content.replace(pattern, replacement)

with open(output filename, 'w') as outfile:

outfile.write(modified content)

print(f"Successfully written to {output\_filename}")

except FileNotFoundError:

print(f'Error: The file '{input filename}' was not found.")

except IOError as e:

print(f"Error: An I/O error occurred: {e}")

except Exception as e: print(f"An unexpected error occurred: {e}")

\*PROGRAM-2(EX-14.2)

import shelve

import anagram\_sets

def store\_anagrams(filename, anagram dict):

"""Stores the anagram dictionary in a shelf file."""

try:

with shelve.open(filename, 'c') as shelf:

for key, value in anagram dict.items():

shelf[key] = value

print(f"Anagrams stored successfully in {filename}")

except Exception as e:

print(f"Error storing anagrams: {e}")

def read\_anagrams(filename, word):

"""Reads the anagram list for a given word from the shelf file.”

lookup

try:

key = anagram\_sets.signature(word) # Generate the key for

with shelve.open(filename, 'r') as shelf:

return shelf.get(key, [])

except Exception as e:

print(f"Error reading anagrams: {e}")

return [ ]

\*PROGRAM-3 (EX-14.3)

import os

import hashlib

import subprocess

def find\_files\_with\_suffix(directory, suffix):

"""Recursively finds all files with the given suffix in the specified directory.

file\_list = []

for root,, files in os.walk(directory):

for file in files:

if file.endswith(suffix):

file\_list.append(os.path.join(root, file))

return file\_list

def compute\_md5(file\_path):

"""Computes the MD5 checksum of a file. """

hasher = hashlib.md5()

try:

with open(file\_path, 'rb') as f:

while chunk := f.read(4096):

hasher.update(chunk)

return hasher.hexdigest()

except Exception as e:

print(f"Error computing MD5 for {file\_path}: {e}")

return None

def find\_duplicates(directory, suffix=".mp3"):

"""Finds duplicate files based on MD5 checksum. ""

files = find files with suffix(directory, suffix)

checksum\_dict = {}

duplicates = {}

for file in files:

checksum = compute\_md5(file)

if checksum:

if checksum in checksum\_dict:

duplicates.setdefault(checksum, []).append(file)

else:

checksum\_dict[checksum] = file

return {k: v for k, v in duplicates.items() if len(v)> 1}

def confirm duplicates(filel, file2):

""Uses the Unix diff command to compare two files. """

try:

result = subprocess.run(["diff", file1, file2],

capture\_output=True, text=True)

return result.stdout ==

except Exception as e:

print(f"Error running diff command: {e}")

return False