1 RegexCode  
print("MANOJ R, USN:1AY24AI068, SEC:O")  
import os

import re

def search\_text\_files(folder\_path, regex\_pattern):

try:

pattern = re.compile(regex\_pattern)

except re.error as e:

print(f"Invalid regular expression: {e}")

return

print("\n--- Matching Lines ---\n")

for filename in os.listdir(folder\_path):

if filename.endswith(".txt"):

file\_path = os.path.join(folder\_path, filename)

try:

with open(file\_path, 'r', encoding='utf-8') as file:

for line\_number, line in enumerate(file, 1):

if pattern.search(line):

print(f"{filename} [Line {line\_number}]: {line.strip()}")

except Exception as e:

print(f"Could not read file {filename}: {e}")

if \_\_name\_\_ == "\_\_main\_\_":

folder = input("Enter the path to the folder: ").strip().strip('"')

regex = input("Enter a regular expression to search for: ").strip()

search\_text\_files(folder, regex)

O/P:MANOJ R, USN:1AY24AI068, SEC:O

Enter the path to the folder: C:\Users\91636\OneDrive\Desktop\New folder

Enter a regular expression to search for: panda

--- Matching Lines ---

New Text Document.txt [Line 1]: the ADJECTIVE panda walked to the NOUN and then

VERB. A nearby NOUN was unaffected by these events

Your Name.txt [Line 1]: the ADJECTIVE panda walked to the NOUN and then VERB. A nearby NOUN was unaffected by these events

2 MadLibs   
print("MANOJ R, USN:1AY24AI068, SEC:O")  
import re  
def mad\_libs(input\_filename, output\_filename):  
 # Read the content of the file  
 with open(input\_filename, 'r') as file:  
 content = file.read()  
 placeholders = re.findall(r'\b(ADJECTIVE|NOUN|VERB|ADVERB)\b', content)  
 for word in placeholders:  
 article = "an" if word[0] in "AEIOU" else "a"  
 user\_input = input(f"Enter {article.lower()} {word.lower()}: ")  
 # Replace only the first occurrence each time  
 content = content.replace(word, user\_input, 1)  
 print("\nGenerated Mad Libs:\n")  
 print(content)  
 with open(output\_filename, 'w') as file:  
 file.write(content)  
 print(f"\nSaved to '{output\_filename}'")  
input\_file = input("Enter the path to your Mad Libs template file: ").strip()  
output\_file = "madlib\_result.txt"  
mad\_libs(input\_file, output\_file)

O/P: MANOJ R, USN:1AY24AI068, SEC:O

Enter the path to your Mad Libs template file: C:\Users\USER\Desktop\AIT\SEM 2\PYT

3 Selective Copy  
print("MANOJ R, USN:1AY24AI068, SEC:O")  
import os  
import shutil  
def copy\_files\_with\_extension(source\_folder, destination\_folder, file\_extension):  
 os.makedirs(destination\_folder, exist\_ok=True)  
 file\_extension = file\_extension.lower()  
 count = 0  
 for foldername, subfolders, filenames in os.walk(source\_folder):  
 for filename in filenames:  
 if filename.lower().endswith(file\_extension):  
 source\_path = os.path.join(foldername, filename)  
 destination\_path = os.path.join(destination\_folder, filename)  
 if os.path.exists(destination\_path):  
 base, ext = os.path.splitext(filename)  
 i = 1  
 while os.path.exists(destination\_path):  
 destination\_path = os.path.join(destination\_folder, f"{base}\_{i}{ext}")  
 i += 1  
 shutil.copy2(source\_path, destination\_path)  
 print(f"Copied: {source\_path} → {destination\_path}")  
 count += 1  
 print(f"\nTotal files copied: {count}")  
source = input("Enter the path to the source folder: ").strip()  
destination = input("Enter the path to the destination folder: ").strip()  
extension = input("Enter the file extension to search for (e.g., .pdf, .jpg): ").strip()  
copy\_files\_with\_extension(source, destination, extension)

O/P: MANOJ R, USN:1AY24AI068, SEC:O

Enter the path to the source folder: C:\Users\USER\Desktop\AIT\SEM 2\PYT  
Enter the path to the destination folder: C:\Users\USER\Desktop\AIT\SEM 2\Syllabus  
Enter the file extension to search for (e.g., .pdf, .jpg): .pdf

Copied: C:\Users\USER\Desktop\AIT\SEM 2\PYT\Introduction to Python (YouTube @ManojPN) Module 1.pdf → C:\Users\USER\Desktop\AIT\SEM 2\Syllabus\Introduction to Python (YouTube @ManojPN) Module 1.pdf  
Copied: C:\Users\USER\Desktop\AIT\SEM 2\PYT\Lab Manual.pdf → C:\Users\USER\Desktop\AIT\SEM 2\Syllabus\Lab Manual.pdf  
  
Total files copied: 2

4 Deleting Unneeded Files  
print("MANOJ R, USN:1AY24AI068, SEC:O")  
import os  
def find\_large\_files(folder\_path, size\_threshold\_mb=100):  
 size\_threshold\_bytes = size\_threshold\_mb \* 1024 \* 1024  
 large\_files\_found = 0  
 print(f"\nScanning for files larger than {size\_threshold\_mb}MB in:\n{os.path.abspath(folder\_path)}\n")  
 for foldername, subfolders, filenames in os.walk(folder\_path):  
 for filename in filenames:  
 try:  
 file\_path = os.path.join(foldername, filename)  
 file\_size = os.path.getsize(file\_path)  
 if file\_size > size\_threshold\_bytes:  
 size\_mb = file\_size / (1024 \* 1024)  
 print(f"{os.path.abspath(file\_path)} - {size\_mb:.2f} MB")  
 large\_files\_found += 1  
 except (PermissionError, FileNotFoundError):  
 continue  
 if large\_files\_found == 0:  
 print("No files larger than the specified size were found.")  
 else:  
 print(f"\nTotal large files found: {large\_files\_found}")  
folder = input("Enter the path to the folder to scan: ").strip()  
find\_large\_files(folder)

O/P: MANOJ R, USN:1AY24AI068, SEC:O

Enter the path to the folder to scan: C:\Users\USER\Desktop\AIT\SEM 2\PYT

Scanning for files larger than 100MB in:  
C:\Users\USER\Desktop\AIT\SEM 2\PYT  
  
No files larger than the specified size were found.

5 Filling in the Gaps  
print("MANOJ R, USN:1AY24AI068, SEC:O")  
import os  
import re  
def get\_numbered\_files(folder, prefix, extension):  
 pattern = re.compile(rf'^{re.escape(prefix)}(\d+){re.escape(extension)}$')  
 numbered\_files = []  
   
 for filename in os.listdir(folder):  
 match = pattern.match(filename)  
 if match:  
 num = int(match.group(1))  
 numbered\_files.append((num, filename))  
 return sorted(numbered\_files)  
def close\_gaps(folder, prefix, extension):  
 files = get\_numbered\_files(folder, prefix, extension)  
 if not files:  
 print("No matching files found.")  
 return  
 next\_expected = 1  
 for actual\_num, filename in files:  
 if actual\_num != next\_expected:  
 new\_name = f"{prefix}{str(next\_expected).zfill(3)}{extension}"  
 print(f"Renaming {filename} → {new\_name}")  
 os.rename(os.path.join(folder, filename), os.path.join(folder, new\_name))  
 next\_expected += 1  
 print("Gaps closed.")  
def insert\_gap(folder, prefix, extension, insert\_position):  
 files = get\_numbered\_files(folder, prefix, extension)  
 if not files:  
 print("No matching files found.")  
 return  
 for num, filename in sorted(files, reverse=True):  
 if num >= insert\_position:  
 new\_num = num + 1  
 new\_name = f"{prefix}{str(new\_num).zfill(3)}{extension}"  
 print(f"Renaming {filename} → {new\_name}")  
 os.rename(os.path.join(folder, filename), os.path.join(folder, new\_name))  
 print(f"Gap inserted at position {insert\_position}.")  
def main():  
 print("Choose an operation:")  
 print("1. Close gaps in numbered files")  
 print("2. Insert a gap at a specific position")  
 choice = input("Enter 1 or 2: ").strip()  
 folder = input("Enter folder path: ").strip()  
 prefix = input("Enter file prefix (e.g., spam): ").strip()  
 extension = input("Enter file extension (e.g., .txt): ").strip()  
 if choice == '1':  
 close\_gaps(folder, prefix, extension)  
 elif choice == '2':  
 pos = int(input("Enter position number to insert gap at (e.g., 2 for spam002): "))  
 insert\_gap(folder, prefix, extension, pos)  
 else:  
 print("Invalid choice.")  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

O/P: MANOJ R, USN:1AY24AI068, SEC:O  
Choose an operation:  
1. Close gaps in numbered files  
2. Insert a gap at a specific position

Enter 1 or 2: 2  
Enter folder path: C:\Users\USER\Desktop\AIT\SEM 2\PYT  
Enter file prefix (e.g., spam): Lab  
Enter file extension (e.g., .txt): .pdf  
Enter position number to insert gap at (e.g., 2 for spam002): 3

No matching files found.

6 Debugging Coin Toss  
print("MANOJ R, USN:1AY24AI068, SEC:O")  
import random  
guess = ''  
while guess not in ('heads', 'tails'):  
 print('Guess the coin toss! Enter heads or tails:')  
 guess = input().lower()  
toss = random.randint(0, 1)   
toss\_str = 'heads' if toss == 1 else 'tails'  
if guess == toss\_str:  
 print('You got it!')  
else:  
 print('Nope! Guess again!')  
 guess = ''  
 while guess not in ('heads', 'tails'):  
 guess = input().lower()  
 if guess == toss\_str:  
 print('You got it!')  
 else:  
 print('Nope. You are really bad at this game.')

O/P: MANOJ R, USN:1AY24AI068, SEC:O  
Guess the coin toss! Enter heads or tails:

Heads

Nope! Guess again!

Tails

You got it!

7 Circle  
print("MANOJ R, USN:1AY24AI068, SEC:O")  
import math  
class Point:  
 def \_\_init\_\_(self, x, y):  
 self.x = x  
 self.y = y  
class Rectangle:  
 def \_\_init\_\_(self, corner, width, height):  
 self.corner = corner  
 self.width = width  
 self.height = height  
class Circle:  
 def \_\_init\_\_(self, center, radius):  
 self.center = center # Point  
 self.radius = radius  
def distance(p1, p2):  
 """Calculate Euclidean distance between two points"""  
 return math.hypot(p1.x - p2.x, p1.y - p2.y)  
  
def point\_in\_circle(circle, point):  
 """Return True if point lies in or on the circle"""  
 return distance(circle.center, point) <= circle.radius  
def get\_rect\_corners(rect):  
 """Return list of 4 corner points of the rectangle"""  
 x, y = rect.corner.x, rect.corner.y  
 return [  
 Point(x, y),   
 Point(x + rect.width, y),   
 Point(x, y + rect.height),  
 Point(x + rect.width, y + rect.height)   
 ]  
def rect\_in\_circle(circle, rect):  
 """Return True if all corners of rect lie in or on circle"""  
 return all(point\_in\_circle(circle, corner) for corner in get\_rect\_corners(rect))  
def rect\_circle\_overlap(circle, rect):  
 """Return True if any corner of rect lies inside circle"""  
 return any(point\_in\_circle(circle, corner) for corner in get\_rect\_corners(rect))  
circle = Circle(Point(150, 100), 75)  
rect1 = Rectangle(Point(120, 80), 40, 30)   
rect2 = Rectangle(Point(100, 50), 100, 100)   
rect3 = Rectangle(Point(300, 300), 20, 20)   
print("rect1 is inside circle:", rect\_in\_circle(circle, rect1))   
print("rect2 is inside circle:", rect\_in\_circle(circle, rect2))   
print("rect2 overlaps circle:", rect\_circle\_overlap(circle, rect2))   
print("rect3 overlaps circle:", rect\_circle\_overlap(circle, rect3))

O/P: MANOJ R, USN:1AY24AI068, SEC:O  
rect1 is inside circle: True  
rect2 is inside circle: True  
rect2 overlaps circle: True  
rect3 overlaps circle: False

8 DateTime  
print("MANOJ R, USN:1AY24AI068, SEC:O")  
from datetime import datetime, timedelta  
def current\_day\_of\_week():  
 today = datetime.today()  
 print("Today's date:", today.strftime("%Y-%m-%d"))  
 print("Day of the week:", today.strftime("%A"))  
def birthday\_info():  
 bday\_str = input("Enter your birthday (YYYY-MM-DD): ")  
 bday = datetime.strptime(bday\_str, "%Y-%m-%d")  
 now = datetime.now()  
 age = now.year - bday.year  
 if (now.month, now.day) < (bday.month, bday.day):  
 age -= 1  
 print(f"You are {age} years old.")  
 next\_birthday = datetime(now.year, bday.month, bday.day)  
 if next\_birthday < now:  
 next\_birthday = datetime(now.year + 1, bday.month, bday.day)  
 time\_left = next\_birthday - now  
 print(f"Time until next birthday: {time\_left.days} days, "  
 f"{time\_left.seconds // 3600} hours, "  
 f"{(time\_left.seconds % 3600) // 60} minutes,"  
 f"{time\_left.seconds % 60} seconds.")  
def double\_day(birth1\_str, birth2\_str):  
 birth1 = datetime.strptime(birth1\_str, "%Y-%m-%d")  
 birth2 = datetime.strptime(birth2\_str, "%Y-%m-%d")  
 if birth1 > birth2:  
 older, younger = birth2, birth1  
 else:  
 older, younger = birth1, birth2  
 diff = younger - older  
 double\_day = younger + diff  
 print("Double Day (when older is twice as old):", double\_day.date())  
def n\_times\_day(birth1\_str, birth2\_str, n):  
 birth1 = datetime.strptime(birth1\_str, "%Y-%m-%d")  
 birth2 = datetime.strptime(birth2\_str, "%Y-%m-%d")  
 if birth1 > birth2:  
 older, younger = birth2, birth1  
 else:  
 older, younger = birth1, birth2  
 diff = younger - older  
 target\_day = younger + diff / (n - 1)  
 print(f"{n}-Times Day:", target\_day.date())  
  
# ----------- Run Example --------  
if \_\_name\_\_ == "\_\_main\_\_":  
 print("1. Current day of the week:")  
 current\_day\_of\_week()  
 print("\n2. Birthday info:")  
 birthday\_info()  
 print("\n3. Double Day Example:")  
 double\_day("1990-01-01", "2000-01-01")  
 print("\n4. n-Times Day Example (n=3):")  
 n\_times\_day("1990-01-01", "2000-01-01", 3)

O/P: MANOJ R, USN:1AY24AI068, SEC:O  
1. Current day of the week:  
Today's date: 2025-06-15  
Day of the week: Sunday  
  
2. Birthday info:

Enter your birthday (YYYY-MM-DD): 2006-12-30

You are 18 years old.  
Time until next birthday: 197 days, 13 hours, 21 minutes,45 seconds.  
  
3. Double Day Example:  
Double Day (when older is twice as old): 2009-12-31  
  
4. n-Times Day Example (n=3):  
3-Times Day: 2004-12-31

9 Average  
print("MANOJ R, USN:1AY24AI068, SEC:O")  
  
class Time:  
 def \_\_init\_\_(self, hours=0, minutes=0, seconds=0):  
 self.hours = hours  
 self.minutes = minutes  
 self.seconds = seconds  
  
 def total\_seconds(self):  
 """Returns total time in seconds"""  
 return self.hours \* 3600 + self.minutes \* 60 + self.seconds  
  
 @classmethod  
 def from\_seconds(cls, total\_secs):  
 """Creates a Time object from total seconds"""  
 hours = total\_secs // 3600  
 remainder = total\_secs % 3600  
 minutes = remainder // 60  
 seconds = remainder % 60  
 return cls(int(hours), int(minutes), int(seconds))  
  
 def print\_time(self):  
 """Nicely formats and prints time"""  
 print(f"{self.hours:02d}:{self.minutes:02d}:{self.seconds:02d}")  
  
 def \_\_str\_\_(self):  
 return f"{self.hours:02d}:{self.minutes:02d}:{self.seconds:02d}"  
  
def mul\_time(time\_obj, factor):  
 """Multiplies a Time object by a number"""  
 total\_secs = time\_obj.total\_seconds() \* factor  
 return Time.from\_seconds(total\_secs)  
  
def avg\_pace(finishing\_time, distance):  
 """Returns average pace (time per unit distance)"""  
 return mul\_time(finishing\_time, 1 / distance)  
  
# ----------- Example Usage -----------  
if \_\_name\_\_ == "\_\_main\_\_":  
 # Suppose you ran a 10-mile race in 1 hour 30 minutes (01:30:00)  
 finish\_time = Time(1, 30, 0)  
 distance = 10   
 print("Finishing Time:", finish\_time)  
 pace = avg\_pace(finish\_time, distance)  
 print("Average Pace (per mile):", pace)

O/P: MANOJ R, USN:1AY24AI068, SEC:O  
Finishing Time: 01:30:00  
Average Pace (per mile): 00:09:00

10 Atrribute  
print("MANOJ R, USN:1AY24AI068, SEC:O")  
from \_\_future\_\_ import print\_function, division  
class Time:  
 """Represents the time of day as seconds since midnight."""  
 def \_\_init\_\_(self, hour=0, minute=0, second=0):  
 self.seconds = hour \* 3600 + minute \* 60 + second  
 def \_\_str\_\_(self):  
 h, rem = divmod(self.seconds, 3600)  
 m, s = divmod(rem, 60)  
 return '%.2d:%.2d:%.2d' % (h, m, s)  
 def print\_time(self):  
 print(str(self))  
 def time\_to\_int(self):  
 """Returns seconds since midnight."""  
 return self.seconds  
 def is\_after(self, other):  
 return self.seconds > other.seconds  
 def \_\_add\_\_(self, other):  
 if isinstance(other, Time):  
 return self.add\_time(other)  
 else:  
 return self.increment(other)  
 def \_\_radd\_\_(self, other):  
 return self.\_\_add\_\_(other)  
 def add\_time(self, other):  
 assert self.is\_valid() and other.is\_valid()  
 return int\_to\_time(self.seconds + other.seconds)  
 def increment(self, seconds):  
 return int\_to\_time(self.seconds + seconds)  
 def is\_valid(self):  
 return 0 <= self.seconds < 24 \* 3600  
def int\_to\_time(seconds):  
 """Creates a Time object from seconds since midnight."""  
 return Time(0, 0, seconds)  
def main():  
 start = Time(9, 45, 0)  
 start.print\_time()  
 end = start.increment(1337)  
 end.print\_time()  
 print('Is end after start?')  
 print(end.is\_after(start))  
 print('Using \_\_str\_\_')  
 print(start, end)  
 start = Time(9, 45)  
 duration = Time(1, 35)  
 print(start + duration)  
 print(start + 1337)  
 print(1337 + start)  
 print('Example of polymorphism')  
 t1 = Time(7, 43)  
 t2 = Time(7, 41)  
 t3 = Time(7, 37)  
 total = sum([t1, t2, t3])  
 print(total)  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

O/P: MANOJ R, USN:1AY24AI068, SEC:O  
09:45:00  
10:07:17  
Is end after start?  
True  
Using \_\_str\_\_  
09:45:00 10:07:17  
11:20:00  
10:07:17  
10:07:17  
Example of polymorphism  
23:01:00

11 Kangaroo  
print("MANOJ R, USN:1AY24AI068, SEC:O")  
class Kangaroo:  
 def \_\_init\_\_(self):  
 # Each kangaroo gets its own pouch list  
 self.pouch\_contents = []  
 def put\_in\_pouch(self, item):  
 self.pouch\_contents.append(item)  
 def \_\_str\_\_(self):  
 contents = []  
 for item in self.pouch\_contents:  
 if isinstance(item, Kangaroo):  
 # Indent nested kangaroos  
 contents.append(" " + str(item).replace("\n", "\n "))  
 else:  
 contents.append(str(item))  
 return "Kangaroo with pouch contents:\n" + "\n".join(contents)  
# ----------- Test Code -----------  
if \_\_name\_\_ == "\_\_main\_\_":  
 kanga = Kangaroo()  
 roo = Kangaroo()  
 kanga.put\_in\_pouch("bottle")  
 kanga.put\_in\_pouch("snack")  
 kanga.put\_in\_pouch(roo)  
 print("Kanga:")  
 print(kanga)  
 print("\nRoo:")  
 print(roo)

O/P: MANOJ R, USN:1AY24AI068, SEC:O  
Kanga:  
Kangaroo with pouch contents:  
bottle  
snack  
 Kangaroo with pouch contents:  
   
  
Roo:  
Kangaroo with pouch contents: