Python Assignment 3

Q25. WAP to combine each line of first file with corresponding line in second file

def combine\_files(file1, file2):  
 with open(file1, 'r') as f1, open(file2, 'r') as f2:  
 lines1 = f1.readlines()  
 lines2 = f2.readlines()  
  
 # Make sure both files have the same number of lines  
 if len(lines1) != len(lines2):  
 print("Error: The two files must have the same number of lines.")  
 return  
  
 for line1, line2 in zip(lines1, lines2):  
 combined\_line = line1.strip() + ' ' + line2.strip() + '\n'  
 print(combined\_line)  
  
 print("Lines combined successfully!")  
  
  
# Usage Example  
file1 = 'file1.txt' # Replace with the path to your first file  
file2 = 'man.txt' # Replace with the path to your second file  
  
  
combine\_files(file1, file2)

Q24.WAP to print each line of a file in reverse Order

def print\_lines\_reverse(filename):  
 try:  
 with open(filename, 'r') as file:  
 lines = file.readlines()  
 print("Original Data: ", lines)  
 for line in reversed(lines):  
 reversed\_line = line.rstrip()[::-1]  
 print(reversed\_line)  
 except FileNotFoundError:  
 print(f"File '{filename}' not found.")  
  
# Usage example  
filename = 'example.txt' # Replace with the path to your file  
print\_lines\_reverse("Q24.txt")

Q23. You are given a file called grades.txt, where each line contains a one-word student username and three test scores separated by spaces, like below:

GWashington 83 77 54

JAdams 86 69 90

Nikita 78 90 46

Nisha 76 23 98

Write code that scans through file and determines how many students passed all three tests

**Solution:**

# Open the grades.txt file  
with open('grades.txt', 'r') as file:  
 passed\_count = 0  
  
 # Read each line in the file  
 for line in file:  
 # Split the line into words and scores  
 data = line.strip().split()  
 print(data)  
 # Extract the username and scores  
 username = data[0]  
 scores = [int(score) for score in data[1:]]  
  
 # Check if all scores are above or equal to the passing threshold  
 if all(score >= 60 for score in scores):  
 passed\_count += 1  
  
# Print the number of students who passed all three tests  
print(f"The number of students who passed all three tests: {passed\_count}")

Q22. Write a python program to read "employee.txt" file and display the alternate employee records.

**Solution:**

def display\_alternate\_records(filename):  
 try:  
 with open(filename, 'r') as file:  
 lines = file.readlines()  
 alternate\_records = [lines[i] for i in range(len(lines)) if i % 2 == 0]  
 for record in alternate\_records:  
 print(record.strip())  
 except FileNotFoundError:  
 print(f"File '{filename}' not found.")  
  
  
filename = 'employee.txt'  
display\_alternate\_records(filename)

**Q – 21.** Write a python program to create an employee.txt file and store employee name and address in the file.

**SOLUTION:**

def create\_employee\_file():  
 # Open the file in write mode  
 with open("employee.txt", "w") as file:  
 while True:  
 employee\_name = input("Enter the employee name (or 'q' to quit): ")  
 if employee\_name == "q":  
 break  
  
 employee\_address = input("Enter the employee address: ")  
  
 # Write the employee name and address to the file  
 file.write(f"Name: {employee\_name}\nAddress: {employee\_address}\n\n")  
  
 print("Employee file created successfully.")  
  
  
# Call the function to create the employee file  
create\_employee\_file()

Q – 20. Develop a program to remove the comment character from all the lines in a given file. Accept file name from user

**SOLUTION**

def remove\_comments(file\_name):  
 # Read the contents of the file  
 with open(file\_name, 'r') as file:  
 lines = file.readlines()  
  
 # Remove the comment character from each line  
 modified\_lines = []  
 for line in lines:  
 if '#' in line:  
 # Remove everything after the comment character  
 line = line[:line.index('#')]  
 modified\_lines.append(line)  
  
 # Write the modified lines back to the file  
 with open(file\_name, 'w') as file:  
 file.writelines(modified\_lines)  
  
 print(f"Comments removed from '{file\_name}'.")  
  
  
# Accept the file name from the user  
file\_name = input("Enter the file name: ")  
  
# Remove comments from the file  
remove\_comments(file\_name)

Q – 19. Python Program to Append the Contents of One File to Another File

**SOLUTION**:

def append\_file(source\_file, destination\_file):  
 try:  
 with open(source\_file, 'r') as source:  
 with open(destination\_file, 'a') as destination:  
 contents = source.read()  
 destination.write(contents)  
 print(f"Successfully appended contents of '{source\_file}' to '{destination\_file}'.")  
 except FileNotFoundError:  
 print("File not found.")  
 except Exception as e:  
 print(f"An error occurred: {str(e)}")  
  
  
# Usage example  
append\_file('grades.txt', 'employee.txt')

Q – 18. Develop a program to Print the number of line,words,and characters present in the given file. Accept the file name from user. Handle necessary exceptions

**SOLUTION**:

def count\_file\_stats(file\_name):  
 try:  
 with open(file\_name, 'r') as file:  
 lines = file.readlines()  
 line\_count = len(lines)  
 word\_count = 0  
 char\_count = 0  
  
 for line in lines:  
 words = line.split()  
 word\_count += len(words)  
 char\_count += len(line)  
  
 print("Number of lines:", line\_count)  
 print("Number of words:", word\_count)  
 print("Number of characters:", char\_count)  
  
 except FileNotFoundError:  
 print("File not found. Please check the file name and try again.")  
  
 except PermissionError:  
 print("Permission denied. Please make sure you have the necessary permissions to access the file.")  
  
 except Exception as e:  
 print("An error occurred:", str(e))  
  
  
file\_name = input("Enter the file name: ")  
count\_file\_stats(file\_name)

Q – 17. Write a function to compute 5/0 and use try/except to catch the exceptions.

**SOLUTION**:

def divide\_by\_zero():  
 try:  
 result = 5/0  
 return result  
 except ZeroDivisionError:  
 print("Error: Division by zero is not allowed!")  
  
divide\_by\_zero()

Q – 16. Write a python program to check the given number is prime or not. Handle Suitable Exception

**SOLUTION**:

def is\_prime(n):  
 if n <= 1:  
 return False  
  
 # Check for factors from 2 to the square root of n  
 for i in range(2, int(n\*\*0.5) + 1):  
 if n % i == 0:  
 return False  
  
 return True  
  
try:  
 number = int(input("Enter a number: "))  
 if is\_prime(number):  
 print(f"{number} is a prime number.")  
 else:  
 print(f"{number} is not a prime number.")  
except ValueError:  
 print("Invalid input. Please enter a valid integer.")  
except Exception as e:  
 print(f"An error occurred: {str(e)}")

Q – 15. Write a multithreading program where one thread prints square of a number and another thread prints factorial of a number. Also display the total time taken for the execution

**SOLUTION**:

import threading  
import time  
import math  
  
  
def calculate\_square(number):  
 print("Calculating square...")  
 time.sleep(1) # Simulating some computation time  
 square = number \*\* 2  
 print(f"Square of {number} is: {square}")  
  
  
def calculate\_factorial(number):  
 print("Calculating factorial...")  
 time.sleep(1) # Simulating some computation time  
 factorial = math.factorial(number)  
 print(f"Factorial of {number} is: {factorial}")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 number = 5  
  
 start\_time = time.time()  
  
 # Create the threads  
 square\_thread = threading.Thread(target=calculate\_square, args=(number,))  
 factorial\_thread = threading.Thread(target=calculate\_factorial, args=(number,))  
  
 # Start the threads  
 square\_thread.start()  
 factorial\_thread.start()  
  
 # Wait for the threads to complete  
 square\_thread.join()  
 factorial\_thread.join()  
  
 end\_time = time.time()  
 total\_time = end\_time - start\_time  
 print(f"Total time taken: {total\_time} seconds")

Q – 14. Write a programm for synchronization of Threads using RLOCK. Accept the two numbers from user and calculate factorial of both Numbers simultaneously

**SOLUTION:**

import threading  
  
  
def factorial(n, lock):  
 result = 1  
 for i in range(1, n + 1):  
 result \*= i  
 with lock:  
 print(f"Factorial of {n}: {result}")  
  
  
def main():  
 num1 = int(input("Enter the first number: "))  
 num2 = int(input("Enter the second number: "))  
  
 lock = threading.RLock()  
  
 thread1 = threading.Thread(target=factorial, args=(num1, lock))  
 thread2 = threading.Thread(target=factorial, args=(num2, lock))  
  
 thread1.start()  
 thread2.start()  
  
 thread1.join()  
 thread2.join()  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

**Q – 13.** Write program for checking length of password. raise an exception if it is not of correct length

**SOLUTION:**

MIN\_PASSWORD\_LENGTH = 8  
MAX\_PASSWORD\_LENGTH = 16  
  
def check\_password\_length(password):  
 if len(password) < MIN\_PASSWORD\_LENGTH or len(password) > MAX\_PASSWORD\_LENGTH:  
 raise ValueError(f"Password must be between {MIN\_PASSWORD\_LENGTH} and {MAX\_PASSWORD\_LENGTH} characters long.")  
 else:  
 print("Password length is valid.")  
  
# Example usage:  
try:  
 password = input("Enter your password: ")  
 check\_password\_length(password)  
except ValueError as error:  
 print(str(error))

Q – 12. Write a program to check the age of person for voting. raise an exception if he is not an eligible person for voting(check for negative values also)

**SOLUTION:**

def check\_voting\_eligibility(age):  
 if age < 0:  
 raise ValueError("Age cannot be negative.")  
 elif age < 18:  
 raise ValueError("You are not eligible to vote.")  
 else:  
 print("You are eligible to vote!")  
  
  
try:  
 age\_input = int(input("Enter your age: "))  
 check\_voting\_eligibility(age\_input)  
except ValueError as e:  
 print("Error:", str(e))

**Q – 11.** Write a program to raise an exception if the user enters name of an employee which is not in predifined list of Employee (note: create a list of employee)

**SOLUTION:**

# Predefined list of employees  
employee\_list = ['John', 'Mary', 'Tom', 'Emily']  
  
  
# Function to check if the entered name is in the list  
def check\_employee(name):  
 if name not in employee\_list:  
 raise ValueError("Employee not found: {}".format(name))  
  
  
# Main program  
try:  
 # Prompt user for input  
 user\_input = input("Enter the name of an employee: ")  
  
 # Check if the entered name is in the list  
 check\_employee(user\_input)  
  
 # If no exception is raised, print success message  
 print("Employee found!")  
  
except ValueError as e:  
 # Print error message  
 print(str(e))

Q – 10. Write a Regular Expression to represent all PIBM language (Your own language) identifiers. Rules:   
1. The allowed characters are a-z,A-Z,0-9,#.   
2. The first character should be a lower case alphabet symbol from a to k.   
3. The second character should be a digit divisible by 3.   
4. The length of identifier should be at least 2.   
 Write a python program to check whether the given string is PIBM language identifier or not?

**SOLUTION:**

import re  
  
pattern = r'^[a-k][0369][a-zA-Z0-9#]\*$'  
  
def is\_pibm\_identifier(identifier):  
 if re.match(pattern, identifier):  
 return True  
 else:  
 return False  
  
   
identifier1 = "a3B9#"  
identifier2 = "l60Xyz"  
identifier3 = "c303"  
  
print(is\_pibm\_identifier(identifier1)) # True  
print(is\_pibm\_identifier(identifier2)) # False  
print(is\_pibm\_identifier(identifier3)) # False

**Q – 9.** Write a Regular Expression to represent all 10 digit mobile numbers.   
Rules:   
1. Every number should contains exactly 10 digits.   
2. The first digit should be 7 or 8 or 9   
Write a Python Program to check whether the given number is valid mobile number or not?

**SOLUTION:**

import re  
  
def is\_valid\_mobile\_number(number):  
 pattern = r"^[789]\d{9}$"  
 return bool(re.match(pattern, number))  
  
  
number = input("Enter a mobile number: ")  
if is\_valid\_mobile\_number(number):  
 print("Valid mobile number")  
else:  
 print("Invalid mobile number")

Q – 8. Write a Python code to validate Email ID.

**SOLUTION:**

import re  
  
def validate\_email(email):  
 pattern = r'^[\w\.-]+@[\w\.-]+\.\w+$'  
 if re.match(pattern, email):  
 return True  
 else:  
 return False  
  
# Example usage  
email = input("Enter an email ID: ")  
if validate\_email(email):  
 print("Valid email ID.")  
else:  
 print("Invalid email ID.")

Q – 7. Write a Python program to check the validity of a password (input from users).  
Validation :  
At least 1 letter between [a-z] and 1 letter between [A-Z].  
At least 1 number between [0-9].  
At least 1 character from [$#@].  
Minimum length 6 characters.  
Maximum length 16 characters.

**SOLUTION:**

import re  
  
  
def validate\_password(password):  
 # Check length  
 if len(password) < 6 or len(password) > 16:  
 return False  
  
 # Check lowercase letter  
 if not re.search(r'[a-z]', password):  
 return False  
  
 # Check uppercase letter  
 if not re.search(r'[A-Z]', password):  
 return False  
  
 # Check number  
 if not re.search(r'[0-9]', password):  
 return False  
  
 # Check special character  
 if not re.search(r'[$#@]', password):  
 return False  
  
 return True  
  
  
# Get password from user  
password = input("Enter a password: ")  
  
# Validate password  
if validate\_password(password):  
 print("Valid password")  
else:  
 print("Invalid password")

Q – 6. Define a class named Shape and its subclass Square. The Square class has an init function which takes a length as argument. Both classes have a area function which can print the area of the shape where Shape's area is 0 by default.

**SOLUTION:**

class Shape:  
 def \_\_init\_\_(self):  
 self.area = 0  
  
 def calculate\_area(self):  
 print("Area of the shape:", self.area)  
  
  
class Square(Shape):  
 def \_\_init\_\_(self, length):  
 super().\_\_init\_\_()  
 self.length = length  
  
 def calculate\_area(self):  
 self.area = self.length \*\* 2  
 print("Area of the square:", self.area)  
   
  
shape = Shape()  
shape.calculate\_area() # Output: Area of the shape: 0  
  
square = Square(5)  
square.calculate\_area() # Output: Area of the square: 25

Q – 5.Write a Python class named Circle constructed by a radius and two methods which will compute the area and the perimeter of a circle.

**SOLUTION:**

import math  
  
class Circle:  
 def \_\_init\_\_(self, radius):  
 self.radius = radius  
  
 def area(self):  
 return math.pi \* (self.radius \*\* 2)  
  
 def perimeter(self):  
 return 2 \* math.pi \* self.radius  
  
  
# Create a circle with radius 5  
my\_circle = Circle(5)  
  
# Calculate and print the area  
print("Area:", my\_circle.area())  
  
# Calculate and print the perimeter  
print("Perimeter:", my\_circle.perimeter())  
Area: 78.53981633974483  
Perimeter: 31.41592653589793

Q – 4.Define a class which has at least two methods:  
getString: to get a string from console input  
printString: to print the string in upper case.

**SOLUTION:**

class StringManipulator:  
 def \_\_init\_\_(self):  
 self.string = ""  
  
 def getString(self):  
 self.string = input("Enter a string: ")  
  
 def printString(self):  
 print(self.string.upper())  
# Create an instance of StringManipulator  
manipulator = StringManipulator()  
  
# Get a string from the user  
manipulator.getString()  
  
# Print the string in uppercase  
manipulator.printString()

Q – 3. Write a class called Product. The class should have fields called name, amount, and price, holding the product’s name, the number of items of that product in stock, and the regular price of the product. There should be a method get\_price that receives the number of items to be bought and returns a the cost of buying that many items, where the regular price is charged for orders of less than 10 items, a 10% discount is applied for orders of between 10 and 99 items, and a 20% discount is applied for orders of 100 or more items. There should also be a method called make\_purchase that receives the number of items to be bought and decreases amount by that much

**SOLUTION:**

class Product:  
 def \_\_init\_\_(self, name, amount, price):  
 self.name = name  
 self.amount = amount  
 self.price = price  
  
 def get\_price(self, quantity):  
 if quantity < 10:  
 return self.price \* quantity  
 elif 10 <= quantity <= 99:  
 discounted\_price = self.price \* 0.9 # 10% discount  
 return discounted\_price \* quantity  
 else:  
 discounted\_price = self.price \* 0.8 # 20% discount  
 return discounted\_price \* quantity  
  
 def make\_purchase(self, quantity):  
 if quantity <= self.amount:  
 self.amount -= quantity  
 return True  
 else:  
 return False  
# Create a Product instance  
product = Product("Apple", 50, 1.0)  
  
# Get the price for buying 5 items  
price = product.get\_price(5)  
print("Price for 5 items:", price)  
  
# Make a purchase of 5 items  
if product.make\_purchase(5):  
 print("Purchase successful. Remaining amount:", product.amount)  
else:  
 print("Insufficient stock.")  
  
# Get the price for buying 15 items  
price = product.get\_price(15)  
print("Price for 15 items:", price)  
  
# Make a purchase of 15 items  
if product.make\_purchase(15):  
 print("Purchase successful. Remaining amount:", product.amount)  
else:  
 print("Insufficient stock.")

Q – 2.Write a class called “Investment” with fields called ‘principal’ and ‘interest’. The constructor should set the values of those fields. There should be a method called value\_after that returns the value of the investment after n years. The formula for this is p(1 + i) n , where p is the principal, and i is the interest rate.

**SOLUTION:**

class Investment:  
 def \_\_init\_\_(self, principal, interest):  
 self.principal = principal  
 self.interest = interest  
  
 def value\_after(self, years):  
 value = self.principal \* (1 + self.interest) \*\* years  
 return value  
  
   
investment = Investment(1000, 0.05) # Create an Investment object with principal of 1000 and interest rate of 0.05  
value\_after\_5\_years = investment.value\_after(5) # Calculate the value after 5 years  
print(value\_after\_5\_years) # Output: 1276.2815625000003

Q – 1. Create class called 'library' with data attributes like Acc-Number, publisher, title and Author, The methods of class should include  
i) Read()-Acc-Number,title,Author,publisher  
ii)Compute()-to accept the number of day late, calculate and display the fine fine charged at the rate of Rupees 5/- per day  
iii)Display the data

**SOLUTION:**

class Library:  
 def \_\_init\_\_(self, acc\_number, publisher, title, author):  
 self.acc\_number = acc\_number  
 self.publisher = publisher  
 self.title = title  
 self.author = author  
  
 def read(self):  
 print("Accession Number:", self.acc\_number)  
 print("Title:", self.title)  
 print("Author:", self.author)  
 print("Publisher:", self.publisher)  
  
 def compute(self, days\_late):  
 fine = 5 \* days\_late  
 print("Fine charged: Rs", fine, "/-")  
  
 def display(self):  
 print("Accession Number:", self.acc\_number)  
 print("Title:", self.title)  
 print("Author:", self.author)  
 print("Publisher:", self.publisher)  
  
  
# Example usage:  
book = Library(123456, "Publisher X", "Book Title", "Author Name")  
book.read() # Display book information  
book.compute(7) # Calculate fine for 7 days late  
book.display() # Display book information again