**Program source Code:**

**guru\_lab05.py**

#importing operating system  
import os

print("Guruteja\_Kanderi")  
print("A20526883")  
print("Open source programming python")  
print("LAB\_05-File processing-Pay\_Roll\_Operations")

# Function to calculate the gross pay based on rate and hours per week included with overtime which has 1.% times more compared to fixed pay rate  
def calculate\_gross\_pay(rate, hours):  
 if hours <= 40:  
 # If the number of hours worked is less than or equal to 40,  
 # then the gross pay is simply the rate of pay multiplied by the number of hours worked.  
 gross\_pay = rate \* hours  
 else:  
 # Otherwise, the gross pay is the regular pay plus the overtime pay.  
 # The regular pay is the rate of pay multiplied by 40 hours.  
 # The overtime pay is the rate of pay multiplied by 1.5 times the number of overtime hours.  
 regular\_pay = rate \* 40  
 overtime\_hours = hours - 40  
 overtime\_pay = rate \* 1.5 \* overtime\_hours  
 gross\_pay = regular\_pay + overtime\_pay  
 return float(gross\_pay)  
  
def print\_allemp():  
 # This function prints the payroll report for all employees.  
 file\_name\_dir = "employees.txt"  
 try:  
 # Open the file in read mode.  
 with open(file\_name\_dir, "r") as emp\_file\_dir:  
 print("Gross Payroll Report for All Employees:")  
 print("-------------------------------------")  
 for line in emp\_file\_dir:  
 line = line.split(" ")  
 first\_name\_employee = line[0]  
 last\_name = line[1]  
 rate = float(line[2])  
 hours = float(line[3])  
 name = first\_name\_employee + " " + last\_name  
 gross\_pay = calculate\_gross\_pay(rate, hours)  
 overtime\_hours = hours - 40  
 if overtime\_hours < 0:  
 overtime\_hours = 0  
 overtime\_pay = rate \* 1.5 \* overtime\_hours  
 print(f"Employee Name: {name}") # Print employee name  
 print(f"Gross Amount: ${gross\_pay:.2f}") # Print gross pay with proper formatting  
 print(f"Total Hours Worked: {hours}") # Print total hours worked  
 print(f"Overtime Hours Worked: {overtime\_hours}") # Print overtime hours worked  
 print(f"Overtime Pay: ${overtime\_pay:.2f}") # Print overtime pay with proper formatting  
 print() # Add a gap between employees  
 print("-------------------------------------")  
 except FileNotFoundError:  
 # Print an error message if the file is not found.  
 print("Error: Payroll file not found.")  
  
def print\_employee():  
 file\_name = "employees.txt" # Assuming the file is named employees.txt  
 name = input("Enter the name of the employee: ")  
 try:  
 with open(file\_name, "r") as emp\_file:  
 print("Gross Payroll Report for Employee:")  
 print("---------------------------------")  
 for line in emp\_file:  
 line = line.split(" ")  
 first\_name = line[0]  
 last\_name = line[1]  
 rate = float(line[2])  
 hours = float(line[3])  
 if name == f"{first\_name} {last\_name}":  
 gross\_pay = calculate\_gross\_pay(rate, hours)  
 overtime\_hours = hours - 40  
 if overtime\_hours < 0:  
 overtime\_hours = 0  
 overtime\_pay = rate \* 1.5 \* overtime\_hours  
 print(f"Employee Name: {name}")  
 print(f"Gross Amount: ${gross\_pay:.2f}")  
 print(f"Total Hours Worked: {hours}")  
 print(f"Overtime Hours Worked: {overtime\_hours}")  
 print(f"Overtime Pay: ${overtime\_pay:.2f}")  
 print("---------------------------------")  
 except FileNotFoundError:  
 print("Error: Payroll file not found.")  
  
#This function is used to add an employee to the list or .txt file with first, last and hours worked and pay rate  
def add\_employee():  
 file\_name = "employees.txt" # Assuming the file is named employees.txt  
 # Enter the first and last name to add  
 first\_name = input("Enter the first name of the employee: ")  
 last\_name = input("Enter the last name of the employee: ")  
  
 while True:  
 try:  
 rate = float(input("Enter the rate of pay: "))  
 break  
 except ValueError:  
 #If the user enters alphabets instead of numerics it will pop up this error in console.  
 print("Invalid input. Please enter a numeric value for the rate of pay.")  
  
 while True:  
 try:  
 hours = float(input("Enter the number of hours worked: "))  
 break  
 except ValueError:  
 print("Invalid input. Please enter a numeric value for the number of hours worked.")  
  
 try:  
 with open(file\_name, "a") as emp\_file:  
 #This function opens the txt file and takes the first name, last name , rate and hours and will insert them in the txt file.  
 emp\_file.write(f"{first\_name} {last\_name} {rate} {hours}\n")  
 print("-------------------------------------")  
 print("Employee record added successfully.\n")  
 print("-------------------------------------")  
 except FileNotFoundError:  
 print("-------------------------------------")  
 print("Error: Payroll file not found.\n")  
 print("-------------------------------------")  
#This function is used to delete a employee from the list , with an input of employee full name.  
def delete\_an\_employee():  
 file\_name = "employees.txt"  
 temp\_file = "temp.txt" # Temporary file name  
 first\_name = input("Enter the first name of the employee to delete: ")  
 last\_name = input("Enter the last name of the employee to delete: ")  
  
 try:  
 with open(file\_name, "r") as emp\_file, open(temp\_file, "w") as temp:  
 found = False  
 for line in emp\_file:  
 if line.startswith(f"{first\_name} {last\_name}"):  
 found = True  
 else:  
 temp.write(line)  
  
 if found:  
 os.remove(file\_name) # Remove the original file  
 os.rename(temp\_file, file\_name) # Rename the temporary file  
 print("-------------------------------------")  
 print("Employee record deleted successfully.")  
 print("-------------------------------------")  
 else:  
 print("-------------------------------------")  
 print("Employee not found.")  
 print("-------------------------------------")  
 except FileNotFoundError:  
 print("-------------------------------------")  
 print("Error: Payroll file not found.")  
 print("-------------------------------------")  
#This function is used to modify the employee details which is already existing.  
def modify\_an\_employee():  
 file\_name = "employees.txt"  
 temp\_file = "temp.txt" # Temporary file name  
 first\_name = input("Enter the first name of the employee to modify: ")  
 last\_name = input("Enter the last name of the employee to modify: ")  
  
 try:  
 with open(file\_name, "r") as emp\_file, open(temp\_file, "w") as temp:  
 found = False  
 for line in emp\_file:  
 if line.startswith(f"{first\_name} {last\_name}"):  
 rate = float(input("Enter the new rate of pay: "))  
 hours = float(input("Enter the new number of hours worked: "))  
 line = f"{first\_name} {last\_name} {rate} {hours}\n"  
 found = True  
 temp.write(line)  
  
 if found:  
 os.remove(file\_name) # Remove the original file  
 os.rename(temp\_file, file\_name) # Rename the temporary file  
 print("-------------------------------------")  
 print("Employee record modified successfully.")  
 print("-------------------------------------")  
 else:  
 print("-------------------------------------")  
 print("Employee not found.")  
 print("-------------------------------------")  
 except FileNotFoundError:  
 print("Error: Payroll file not found.Please provide correct data.")  
  
def exit\_app(): #This function used to come out of the loop or program execution  
 print("-----------------------------------------------")  
 print("Successfully Terminated the Program. Thank you for using the Payroll Program!")

print("-----------------------------------------------")  
 exit()

**main.py**

# Importing guru\_lab05 module from same package  
from guru\_lab05 import \*  
#This function is used to display the operations that can be performed in this code.  
def menu\_all():  
 # Displaying the menu choices for the payroll operations in the console  
 pstr = "Choose from the following payroll choices:\n"  
 pstr += "(1) Display a gross payroll report for all employees\n"  
 pstr += "(2) Display a gross payroll report for a single employee by name\n"  
 pstr += "(3) Add an employee record\n"  
 pstr += "(4) Delete an employee record\n"  
 pstr += "(5) Modify an employee record\n"  
 pstr += "(6) Exit the program\n"  
 print(pstr)  
 print() # Add an empty print() statement for a line gap at the end.  
  
 #This main function used to select the option to execute the desired operation.  
def gurumain():  
 while True:  
 menu\_all()  
 print("\n") # Add multiple newline characters for line gaps  
 choice = int(input("Enter Menu of Your Choice : "))  
  
 if choice == 1:  
 # Call the function to display gross payroll report for all employees  
 print\_allemp()  
 elif choice == 2:  
 # Call the function to display gross payroll report for a single employee by name  
 print\_employee()  
 elif choice == 3:  
 # Call the function to add an employee record  
 add\_employee()  
 elif choice == 4:  
 # Call the function to delete an employee record  
 delete\_an\_employee()  
 elif choice == 5:  
 # Call the function to modify an employee record  
 modify\_an\_employee()  
 elif choice == 6:  
 # Call the function to exit the program  
 exit\_app()  
 else:  
 #The below statement below will execute if user enters wrong number  
 print("Invalid choice. Please try again:)")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 # Call the main function to start the program  
 gurumain()

**OUTPUT :**

**A screen shot of a computer

Description automatically generated with medium confidence**

**Employees.txt**

**A screenshot of a computer

Description automatically generated**

**When option 1 is choosed:**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**When option 2 is choosed:**

**A screenshot of a computer

Description automatically generated**

**When option 3 is choosed:**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**When option 4 is choosed:**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**when option 5 is choosed:**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**When option 6 is choosed:**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**Employee Not Found:**

**A picture containing text, screenshot, font

Description automatically generated**

**Invalid choice:**

**A screenshot of a computer screen

Description automatically generated with medium confidence**

**Payroll File not Found:**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**Temporary file:**

**A picture containing text, screenshot, software, multimedia software

Description automatically generated**

**Exception Handling:**

**A screenshot of a computer screen

Description automatically generated with low confidence**