

```
#-----  
# Author: Anoushka Das, Jaeden Patterson, Mridul Jain  
# Program: Program 5  
#  
# Description:  
# Calculates the total amount due to each employee from the hours they worked.  
# Program first calculates total hours worked, then proceeds to calculate  
# overtime hours if needed. Then, calculates the billable amount of regular and  
# overtime including overtime rate. Program has validation rules: name must be input, hourly rate must  
# be more than $20.00, weekly hours must be within 35 and 80 hours. Allows user to input multiple  
# employees.  
#-----
```

```
def main():  
    import BillingModule2
```

```
    #Named Constant
```

```
    max_regular_hours = 160  
    overtime_rate_percentage = 1.05  
    to_average = 4
```

```
    newEmployee = "y"
```

```
    while newEmployee == "y":
```

```
        #Variables
```

```
        hourlyRate = 0.00  
        week = 0  
        Week1 = 0  
        Week2 = 0  
        Week3 = 0  
        Week4 = 0  
        totalHours = 0.00  
        overTime = 0.00  
        overRate = 0.00  
        extraDue = 0.00  
        averageHours = 0.00  
        regularDueExtra = 0.00  
        regularDue = 0.00
```

```
        #Input
```

```
        employeeNameInput = "Employee Name: "
```

```
employeeName = BillingModule2.readEmployeeName(employeeNameInput)
```

```
hourlyRateInput = "Hourly Rate: "  
hourlyRate = BillingModule2.readHourlyRate(hourlyRateInput)
```

```
weekInput = "Enter hours worked for week 1: "  
week1 = BillingModule2.readWeeklyHours(weekInput)
```

```
weekInput = "Enter hours worked for week 2: "  
week2 = BillingModule2.readWeeklyHours(weekInput)
```

```
weekInput = "Enter hours worked for week 3: "  
week3 = BillingModule2.readWeeklyHours(weekInput)
```

```
weekInput = "Enter hours worked for week 4: "  
week4 = BillingModule2.readWeeklyHours(weekInput)
```

```
BillingModule2.writeBillingRecord(employeeName, hourlyRate, week1, week2, week3, week4)
```

```
#Calculate Total Hours worked
```

```
totalHours = week1 + week2 + week3 + week4
```

```
#Calculate Average Hours worked
```

```
averageHours = round(totalHours/ to_average, 2)
```

```
#Calculate the invoice amount and overtime hours if applicable
```

```
if totalHours > max_regular_hours:
```

```
    #Calculate how much overtime hours worked
```

```
    overTime = round(totalHours - max_regular_hours, 2)
```

```
    #Calculate amount of overtime pay
```

```
    overRate = round(hourlyRate * overtime_rate_percentage, 2)
```

```
    extraDue = round(overTime * overRate, 2)
```

```
    #Calculate the invoice amount
```

```
    regularDue = round(max_regular_hours * hourlyRate, 2)
```

```
    invoiceAmount = regularDue + extraDue
```

```
    #Assign variables for later output printing
```

```
    workedOvertime = str(overTime) + " hours of"
```

```
    overtimeOutput = "Overtime hours: " + format(overTime, ".2f") + \  
        " @ $" + str(overRate) + " = $" + \  
        format(extraDue, ",.2f") + "\n"
```

```

regHoursOutput = max_regular_hours

else:
    #Calculate invoice amount without overtime
    invoiceAmount = round(totalHours * hourlyRate, 2)
    workedOvertime = "no"
    #Assign variables for later output printing
    regularDue = invoiceAmount
    overtimeOutput = ""
    #When total hours are less than 160, regular hours equals
    regHoursOutput = totalHours

#Output

#Print the invoice for the employee
    print("\n"+employeeName, "worked",workedOvertime, "overtime.")
    print("\nInvoice")
    print("Resource: ", employeeName, "\tAverage weekly hours:", format(averageHours, ".2f"))
    print("\nTotal billable hours: ", format(totalHours, ".2f"), "\trate: $", format(hourlyRate, ".2f"),
sep="")
    print(overtimeOutput, "Regular Hours: ",\
        format(regHoursOutput, ".2f"), " @ $",\
        format(hourlyRate, ".2f"), " = $",\
        format(regularDue, ".2f"), sep="")
    print("Amount Due: $", format(invoiceAmount, ".2f"), sep="")

newEmployee = input("\nEnter another employee? ('y'=yes): ")

#-----
# Author: Anoushka Das, Jaeden Patterson, Mridul Jain
# Program: BillingModule2
#
# Description:
# The given code provides a set of functions to manage billing records for employees. It includes
# functions to write billing records to a file, reset the file, and validate user input for employee
# names, hourly rates, and weekly hours worked. Additionally, there is a function to calculate the
# total billable amount due to an employee based on their hourly rate and total hours worked
#-----

def writeBillingRecord(employeeName, hourlyRate, week1, week2, week3, week4):
    outfile= open("Billing.txt", "a")

```

```
outfile.write(employeeName + "\n"
               + str(hourlyRate) + "\n"
               + str(week1) + "\n"
               + str(week2) + "\n"
               + str(week3) + "\n"
               + str(week4) + "\n")
```

```
outfile.close()
```

```
def resetBillingFile(File):
    outfile = open(File, "w").close()
```

```
def readEmployeeName(employeeNameInput):
    again = True
    while again:
        employeeName = (input(employeeNameInput))
        if employeeName == "":
            print("Employee name must be entered\n")
        else:
            again = False

    return employeeName
```

```
def readHourlyRate(hourlyRateInput):
    minHours = 20
    again = True
    while again:
        try:
            hourlyRate = float(input(hourlyRateInput))
            if hourlyRate < minHours:
                print("Invalid Hourly Rate, must be at least $20.00/hour.\n")
            else:
                again = False
        except ValueError:
            print("Hourly rate must be numeric.\n")

    return hourlyRate
```

```

def readWeeklyHours(weekInput):
    minHours = 35
    maxHours = 80
    again = True
    while again:
        try:
            week = float(input(weekInput))
            if week > maxHours or week < minHours:
                print("Invalid number of hours, must be between 35 and 80.\n")
            else:
                again = False
        except ValueError:
            print("Weekly hours worked must be numeric.\n")

    return week

```

```

def totalBillableDue (rate, totalHours):
    max_regular_hours = 160
    if totalHours > max_regular_hours:
        overtimeRate = round(rate * 1.05, 2)
        overtimeHours = totalHours - max_regular_hours
        overtimePay = round(overtimeHours * overtimeRate, 2)
        regularPay = max_regular_hours * rate
        invoiceAmount = regularPay + overtimePay
    else:
        invoiceAmount = totalHours * rate
    return invoiceAmount

```

```

#-----
# Author: Anoushka Das, Jaeden Patterson, Mridul Jain
# Program: ADHOC
#
# Description:
# This code reads data from a file called "Billing.txt" and processes the information for each
# employee listed in the file. For each employee, the code calculates the total billable amount
# due based on the employee's hourly rate and the total number of hours worked over a
# four-week period. It then prints a table that displays the employee's name, hourly rate, hours
# worked for each week, total hours worked, and the total billable amount due. After processing
# all employees in the file, the code calculates and prints the total billable due for all employees,
# the total billable hours worked, and the average billable hours per employee. If the file is not
# found, the code prints an error message.
#-----

```

```

import BillingModule2

```

```

def main():
    try:
        billingFile = open("Billing.txt", "r")

        print("Employee\t Rate\t Week 1\t Week 2\t Week 3\t Week 4\t Hours\t Total")

        name = billingFile.readline().rstrip("\n")

        count = 0
        totalHours = 0
        totalBillable = 0
        averageBillable = 0

        while name != "":
            count += 1
            rate = float(billingFile.readline().rstrip("\n"))
            week1 = float(billingFile.readline().rstrip("\n"))
            week2 = float(billingFile.readline().rstrip("\n"))
            week3 = float(billingFile.readline().rstrip("\n"))
            week4 = float(billingFile.readline().rstrip("\n"))
            hours = week1 + week2 + week3 + week4

            totalHours += hours

            total = BillingModule2.totalBillableDue(rate, hours)
            totalBillable += total

            print(name, "\t $", format(rate, ".2f"), "\t ", format(week1, ".2f"), "\t ", format(week2, ".2f"), "\t ",
                  format(week3, ".2f"), "\t ", format(week4, ".2f"), "\t ", format(hours, ".2f"), "\t $", format(total,
                  ",.2f"), sep="")

            name = billingFile.readline().rstrip("\n")

        billingFile.close()

        if count == 0:
            print("No employees on file")

        if count > 0:
            averageBillable = totalHours / count
            print("\nTotal Billable Due:\t $", format(totalBillable, ".2f"), sep="")
            print("Total Billable Hours:  ", format(totalHours, ".2f"))
            print("Average Billable Hours:  ", format(averageBillable, ".2f"), "\n", sep="")

    except FileNotFoundError:
        print("Error, no file found.")

```

```
#-----  
# Author: Anoushka Das, Jaeden Patterson, Mridul Jain  
# Program: Menu  
#  
# Description:  
# This Python code implements a billing system menu with three options: ending the program,  
# entering billing data, and displaying an ad-hoc billing report. The program prompts the user to  
# select an option from the menu and executes the corresponding task based on the option  
# chosen. If the user enters an invalid option, the program prompts them to enter a valid one.  
# The program uses try-except blocks to handle possible exceptions that may occur during the  
# execution.  
#-----
```

```
import program5  
import ADHOC  
import BillingModule2
```

```
BillingModule2.resetBillingFile("Billing.txt")
```

```
def main():  
    again = True
```

```
while again == True:  
    print("\nBilling System Menu:\n \  
        \n\t0 - end \  
        \n\t1 - Enter billing data \  
        \n\t2 - Display ad-hoc billing report")
```

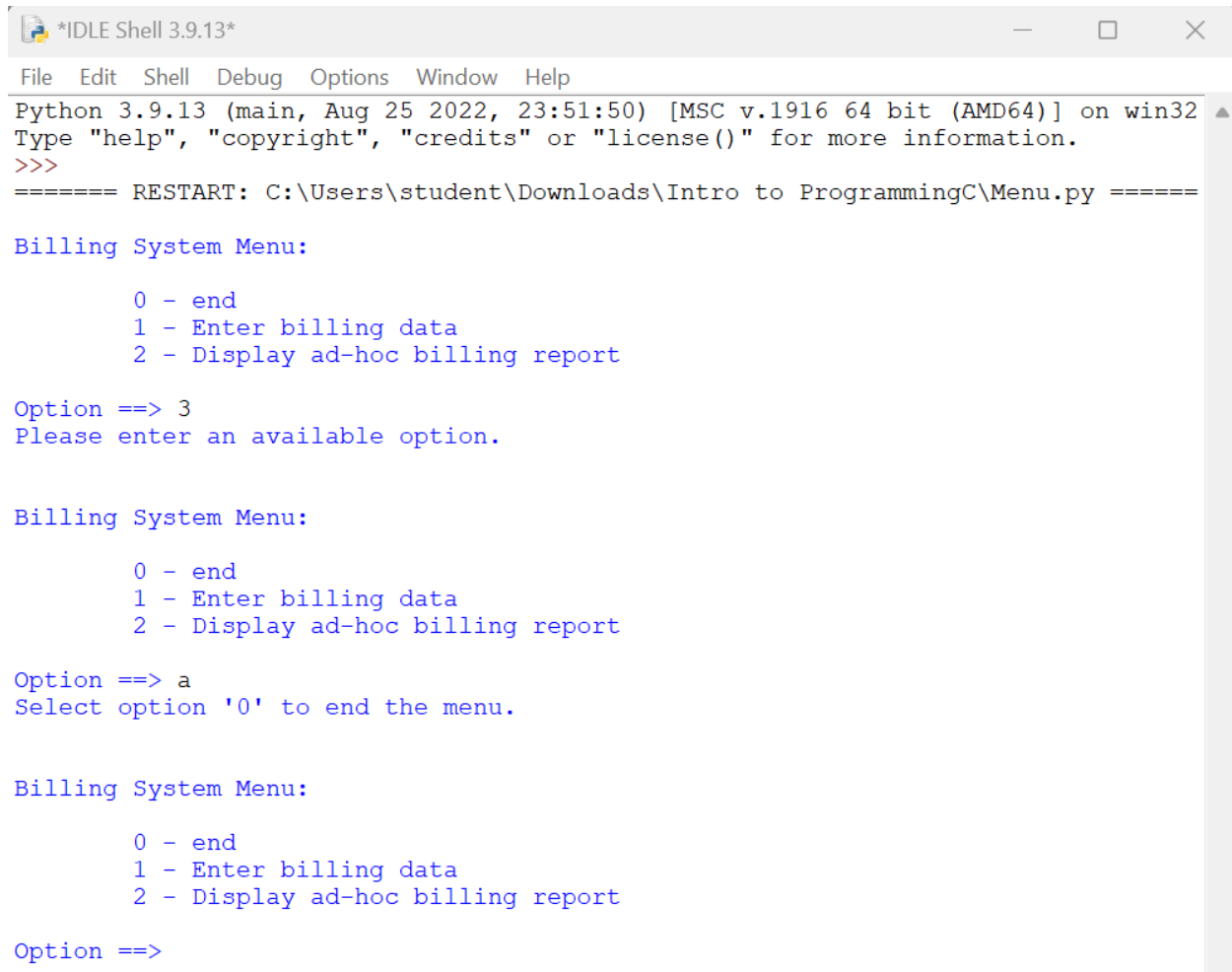
```
try:  
    option = int(input("Option ==> "))  
  
    if option == 0:  
        again = False  
        print("\nProgram ended successfully.")  
    elif option == 1:  
        program5.main()  
    elif option == 2:  
        ADHOC.main()  
    else:  
        print("Please enter an available option.\n")
```

```
except ValueError:  
    print("Select option '0' to end the menu.\n")
```

*main()*

## **Exhibit 1 - Program5**





```
*IDLE Shell 3.9.13*
File Edit Shell Debug Options Window Help
Python 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\student\Downloads\Intro to ProgrammingC\Menu.py =====

Billing System Menu:

    0 - end
    1 - Enter billing data
    2 - Display ad-hoc billing report

Option ==> 3
Please enter an available option.

Billing System Menu:

    0 - end
    1 - Enter billing data
    2 - Display ad-hoc billing report

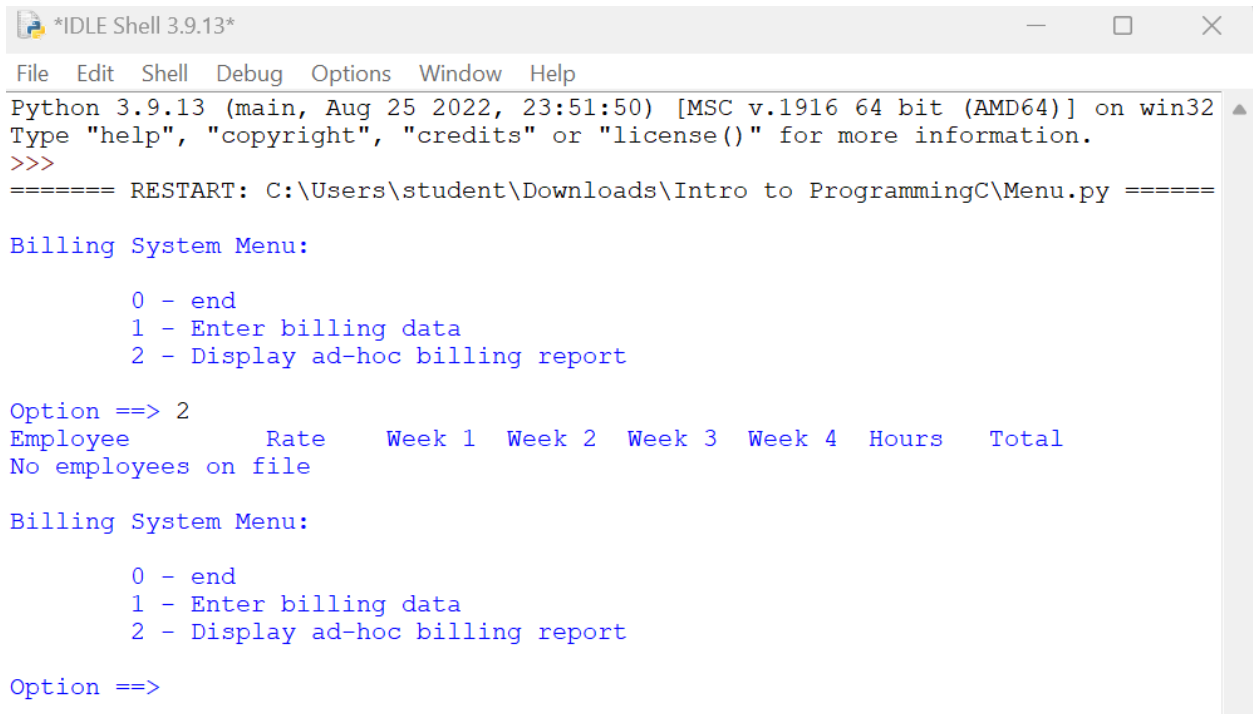
Option ==> a
Select option '0' to end the menu.

Billing System Menu:

    0 - end
    1 - Enter billing data
    2 - Display ad-hoc billing report

Option ==>
```

**Figure 2 - Case test 1: Invalid choice from the menu**



```
*IDLE Shell 3.9.13*
File Edit Shell Debug Options Window Help
Python 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\student\Downloads\Intro to ProgrammingC\Menu.py =====

Billing System Menu:

    0 - end
    1 - Enter billing data
    2 - Display ad-hoc billing report

Option ==> 2
Employee      Rate    Week 1  Week 2  Week 3  Week 4  Hours  Total
No employees on file

Billing System Menu:

    0 - end
    1 - Enter billing data
    2 - Display ad-hoc billing report

Option ==>
```

**Figure 3 - Case test 2: User attempts to run the ad-hoc report without a billing.txt file**

```
*IDLE Shell 3.9.13*
File Edit Shell Debug Options Window Help
Python 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\student\Downloads\Intro to ProgrammingC\Menu.py =====

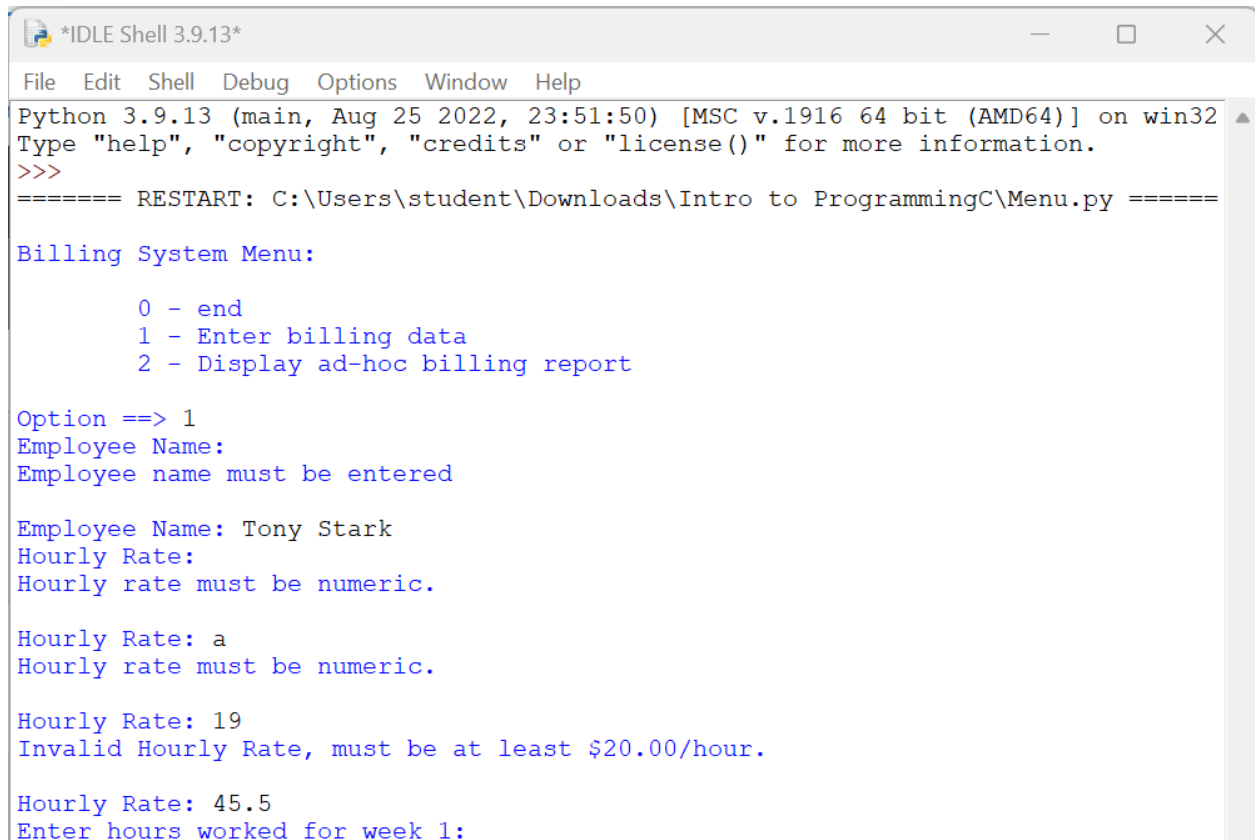
Billing System Menu:

    0 - end
    1 - Enter billing data
    2 - Display ad-hoc billing report

Option ==> 1
Employee Name:
Employee name must be entered

Employee Name: Tony Stark
Hourly Rate:
```

**Figure 4 - Case test 3: Verify “Employee Name” validation**



```
*IDLE Shell 3.9.13*
File Edit Shell Debug Options Window Help
Python 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\student\Downloads\Intro to ProgrammingC\Menu.py =====

Billing System Menu:

    0 - end
    1 - Enter billing data
    2 - Display ad-hoc billing report

Option ==> 1
Employee Name:
Employee name must be entered

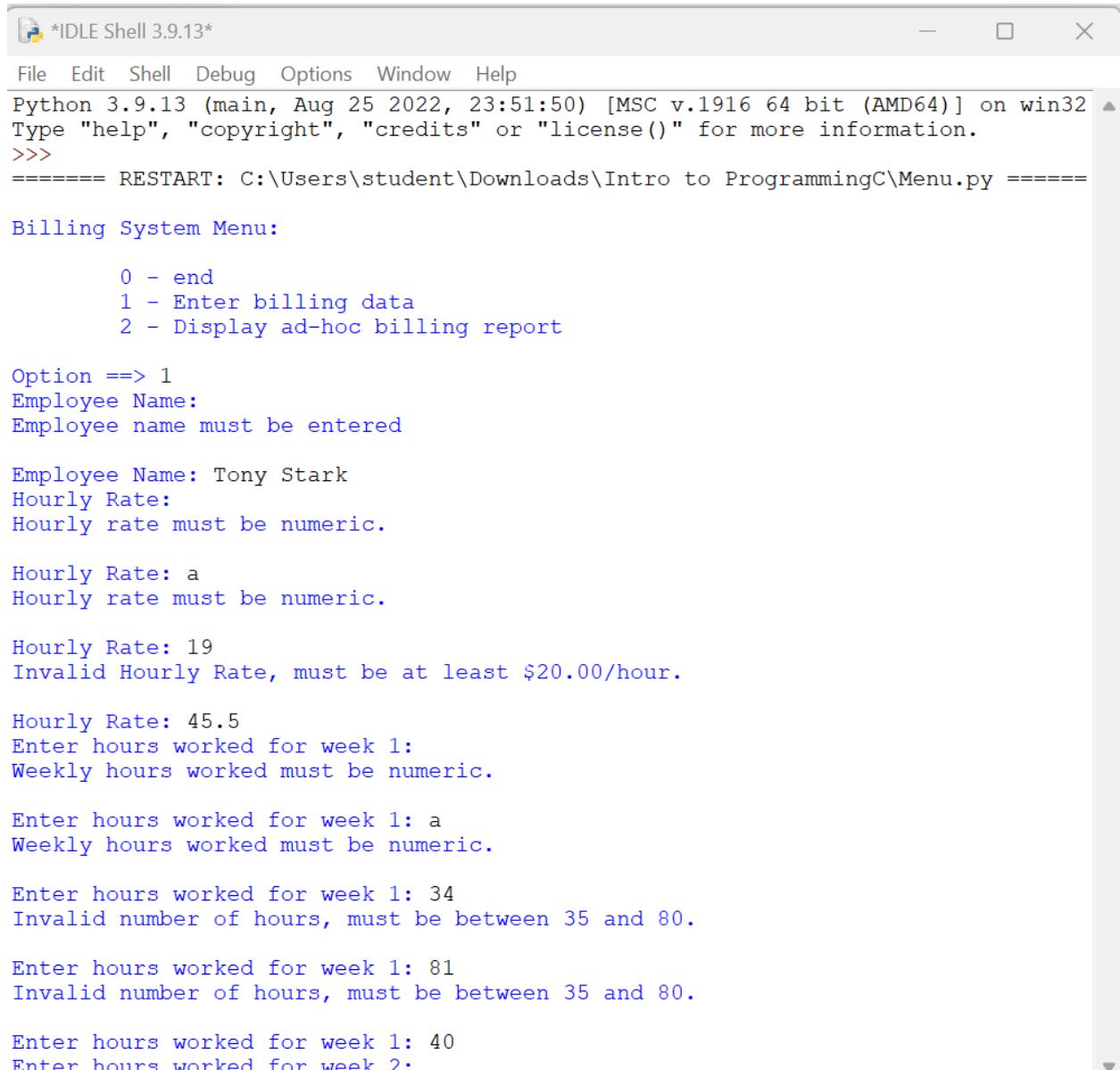
Employee Name: Tony Stark
Hourly Rate:
Hourly rate must be numeric.

Hourly Rate: a
Hourly rate must be numeric.

Hourly Rate: 19
Invalid Hourly Rate, must be at least $20.00/hour.

Hourly Rate: 45.5
Enter hours worked for week 1:
```

**Figure 5 - Case test 4: Verify the “Hourly Rate” validation**



```
*IDLE Shell 3.9.13*
File Edit Shell Debug Options Window Help
Python 3.9.13 (main, Aug 25 2022, 23:51:50) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Users\student\Downloads\Intro to ProgrammingC\Menu.py =====

Billing System Menu:

    0 - end
    1 - Enter billing data
    2 - Display ad-hoc billing report

Option ==> 1
Employee Name:
Employee name must be entered

Employee Name: Tony Stark
Hourly Rate:
Hourly rate must be numeric.

Hourly Rate: a
Hourly rate must be numeric.

Hourly Rate: 19
Invalid Hourly Rate, must be at least $20.00/hour.

Hourly Rate: 45.5
Enter hours worked for week 1:
Weekly hours worked must be numeric.

Enter hours worked for week 1: a
Weekly hours worked must be numeric.

Enter hours worked for week 1: 34
Invalid number of hours, must be between 35 and 80.

Enter hours worked for week 1: 81
Invalid number of hours, must be between 35 and 80.

Enter hours worked for week 1: 40
Enter hours worked for week 2:
```

**Figure 6 - Case test 5: Verify the Weekly Hours validation**

```
*IDLE Shell 3.9.13*
File Edit Shell Debug Options Window Help
Employee Name:
Employee name must be entered

Employee Name: Tony Stark
Hourly Rate:
Hourly rate must be numeric.

Hourly Rate: a
Hourly rate must be numeric.

Hourly Rate: 19
Invalid Hourly Rate, must be at least $20.00/hour.

Hourly Rate: 45.5
Enter hours worked for week 1:
Weekly hours worked must be numeric.

Enter hours worked for week 1: a
Weekly hours worked must be numeric.

Enter hours worked for week 1: 34
Invalid number of hours, must be between 35 and 80.

Enter hours worked for week 1: 81
Invalid number of hours, must be between 35 and 80.

Enter hours worked for week 1: 40
Enter hours worked for week 2: 40
Enter hours worked for week 3: 40
Enter hours worked for week 4: 40

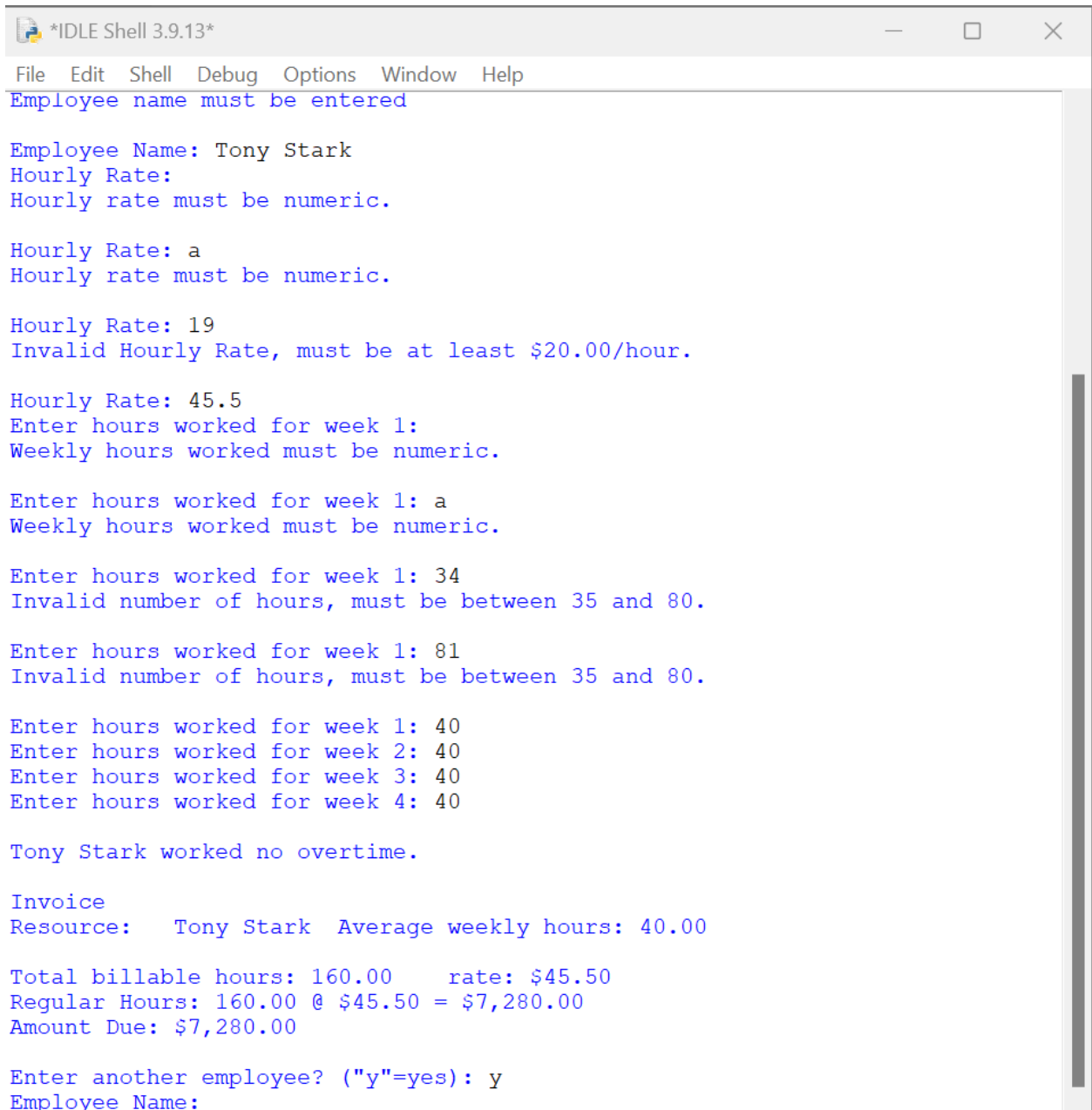
Tony Stark worked no overtime.

Invoice
Resource: Tony Stark Average weekly hours: 40.00

Total billable hours: 160.00 rate: $45.50
Regular Hours: 160.00 @ $45.50 = $7,280.00
Amount Due: $7,280.00

Enter another employee? ("y"=yes):
```

**Figure 7 - Case test 6: Produce a valid invoice for a standard month (all weeks are 40 hours worked)**



```
*IDLE Shell 3.9.13*
File Edit Shell Debug Options Window Help
Employee name must be entered

Employee Name: Tony Stark
Hourly Rate:
Hourly rate must be numeric.

Hourly Rate: a
Hourly rate must be numeric.

Hourly Rate: 19
Invalid Hourly Rate, must be at least $20.00/hour.

Hourly Rate: 45.5
Enter hours worked for week 1:
Weekly hours worked must be numeric.

Enter hours worked for week 1: a
Weekly hours worked must be numeric.

Enter hours worked for week 1: 34
Invalid number of hours, must be between 35 and 80.

Enter hours worked for week 1: 81
Invalid number of hours, must be between 35 and 80.

Enter hours worked for week 1: 40
Enter hours worked for week 2: 40
Enter hours worked for week 3: 40
Enter hours worked for week 4: 40

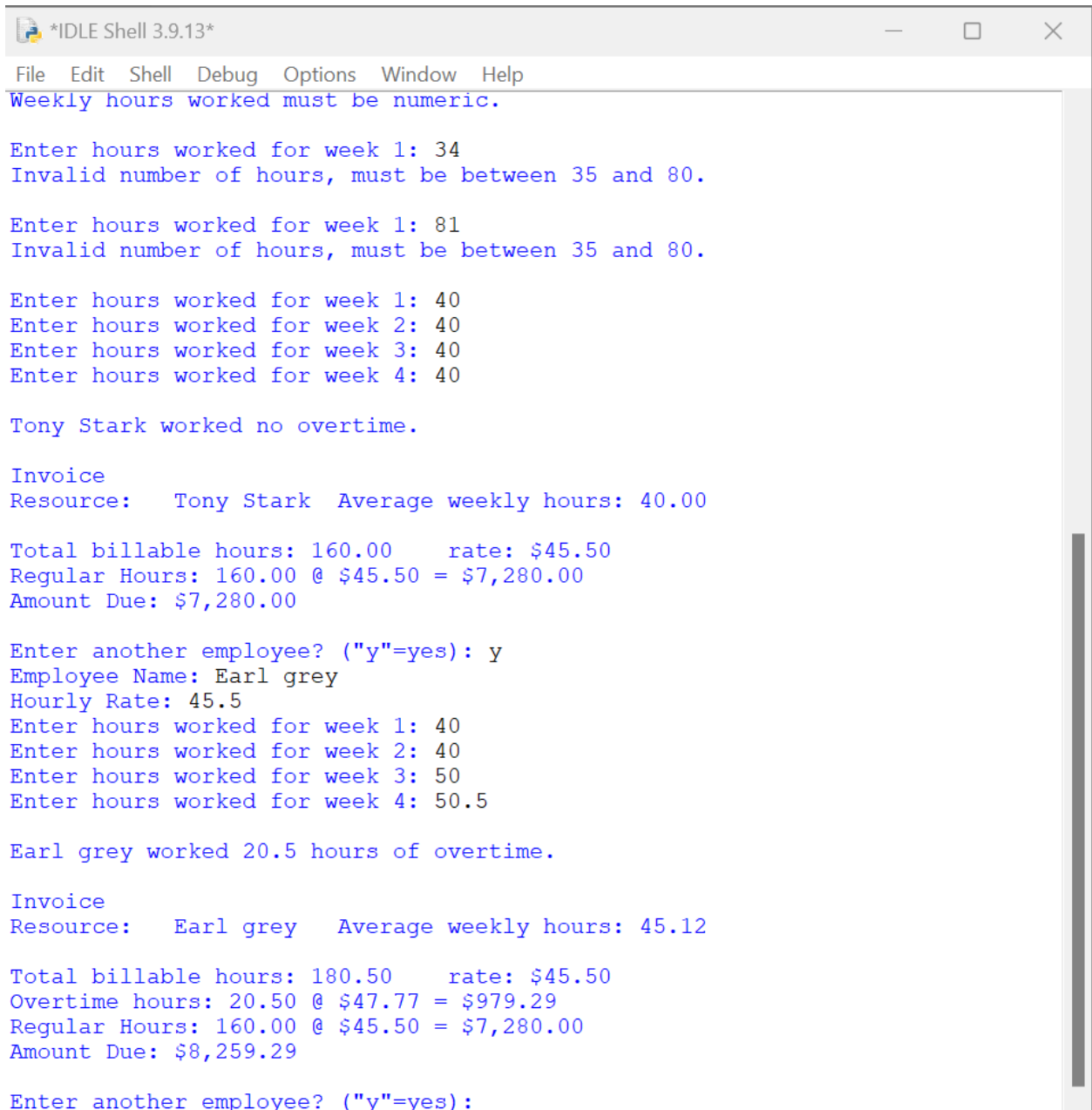
Tony Stark worked no overtime.

Invoice
Resource: Tony Stark Average weekly hours: 40.00

Total billable hours: 160.00 rate: $45.50
Regular Hours: 160.00 @ $45.50 = $7,280.00
Amount Due: $7,280.00

Enter another employee? ("y"=yes): y
Employee Name:
```

**Figure 8 - Case test 7: Verify the validation for the prompt to enter another employee**



```
*IDLE Shell 3.9.13*
File Edit Shell Debug Options Window Help
Weekly hours worked must be numeric.

Enter hours worked for week 1: 34
Invalid number of hours, must be between 35 and 80.

Enter hours worked for week 1: 81
Invalid number of hours, must be between 35 and 80.

Enter hours worked for week 1: 40
Enter hours worked for week 2: 40
Enter hours worked for week 3: 40
Enter hours worked for week 4: 40

Tony Stark worked no overtime.

Invoice
Resource: Tony Stark Average weekly hours: 40.00

Total billable hours: 160.00 rate: $45.50
Regular Hours: 160.00 @ $45.50 = $7,280.00
Amount Due: $7,280.00

Enter another employee? ("y"=yes): y
Employee Name: Earl grey
Hourly Rate: 45.5
Enter hours worked for week 1: 40
Enter hours worked for week 2: 40
Enter hours worked for week 3: 50
Enter hours worked for week 4: 50.5

Earl grey worked 20.5 hours of overtime.

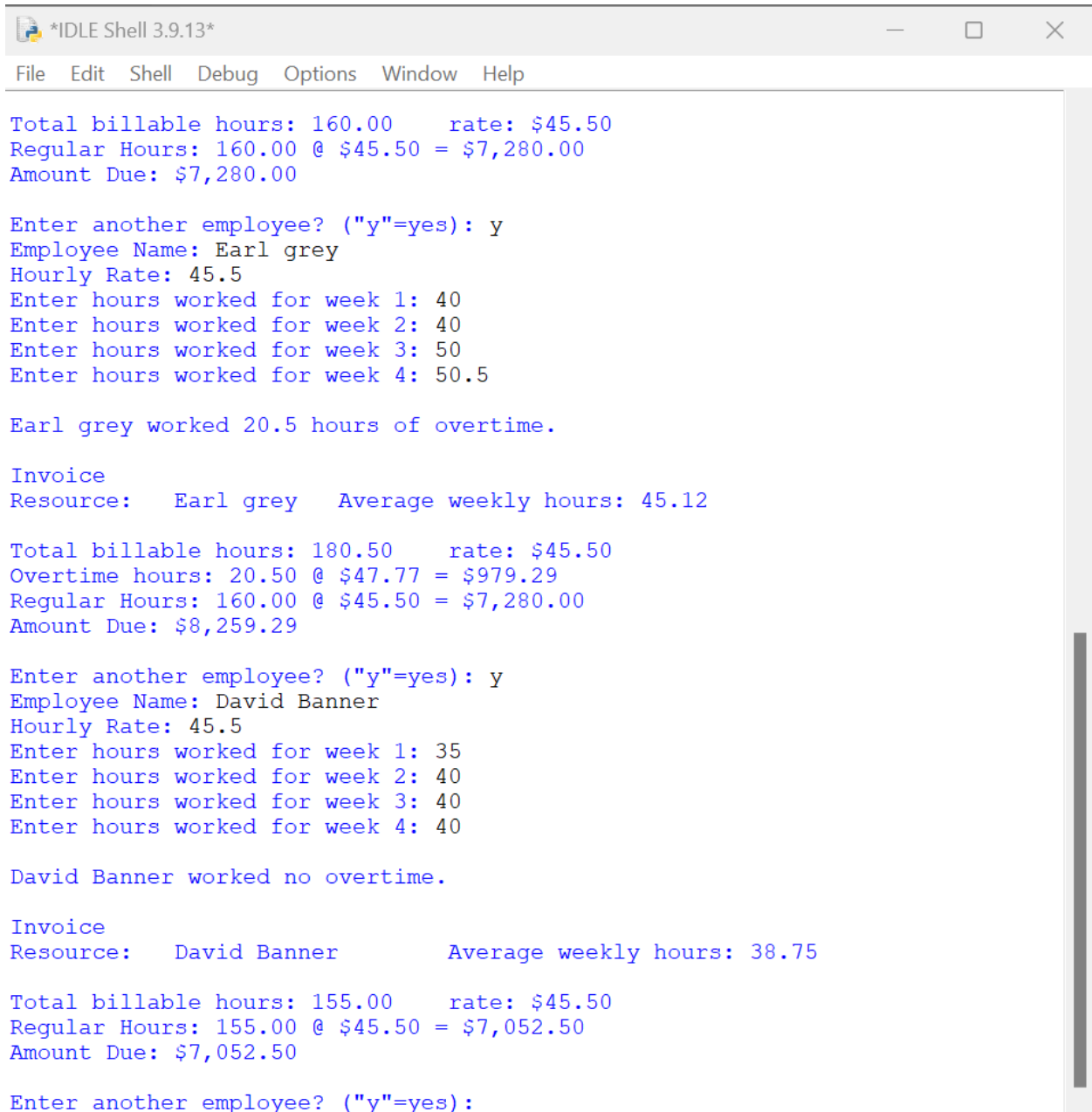
Invoice
Resource: Earl grey Average weekly hours: 45.12

Total billable hours: 180.50 rate: $45.50
Overtime hours: 20.50 @ $47.77 = $979.29
Regular Hours: 160.00 @ $45.50 = $7,280.00
Amount Due: $8,259.29

Enter another employee? ("y"=yes):
```

**Figure 9 - Case test 8: Verify the overtime invoice is produced correctly**





```
*IDLE Shell 3.9.13*
File Edit Shell Debug Options Window Help

Total billable hours: 160.00    rate: $45.50
Regular Hours: 160.00 @ $45.50 = $7,280.00
Amount Due: $7,280.00

Enter another employee? ("y"=yes): y
Employee Name: Earl grey
Hourly Rate: 45.5
Enter hours worked for week 1: 40
Enter hours worked for week 2: 40
Enter hours worked for week 3: 50
Enter hours worked for week 4: 50.5

Earl grey worked 20.5 hours of overtime.

Invoice
Resource:   Earl grey    Average weekly hours: 45.12

Total billable hours: 180.50    rate: $45.50
Overtime hours: 20.50 @ $47.77 = $979.29
Regular Hours: 160.00 @ $45.50 = $7,280.00
Amount Due: $8,259.29

Enter another employee? ("y"=yes): y
Employee Name: David Banner
Hourly Rate: 45.5
Enter hours worked for week 1: 35
Enter hours worked for week 2: 40
Enter hours worked for week 3: 40
Enter hours worked for week 4: 40

David Banner worked no overtime.

Invoice
Resource:   David Banner    Average weekly hours: 38.75

Total billable hours: 155.00    rate: $45.50
Regular Hours: 155.00 @ $45.50 = $7,052.50
Amount Due: $7,052.50

Enter another employee? ("y"=yes):
```

**Figure 10 - Case test 9: Verify the invoice is correct for an employee with no over time working less than standard hours**

The screenshot shows an IDLE Shell window titled '\*IDLE Shell 3.9.13\*' with a menu bar (File, Edit, Shell, Debug, Options, Window, Help). The shell contains the following text:

```
Enter hours worked for week 1: 40
Enter hours worked for week 2: 40
Enter hours worked for week 3: 50
Enter hours worked for week 4: 50.5

Earl grey worked 20.5 hours of overtime.

Invoice
Resource: Earl grey Average weekl

Total billable hours: 180.50 rate:
Overtime hours: 20.50 @ $47.77 = $979
Regular Hours: 160.00 @ $45.50 = $7,2
Amount Due: $8,259.29

Enter another employee? ("y"=yes): y
Employee Name: David Banner
Hourly Rate: 45.5
Enter hours worked for week 1: 35
Enter hours worked for week 2: 40
Enter hours worked for week 3: 40
Enter hours worked for week 4: 40

David Banner worked no overtime.

Invoice
Resource: David Banner Avera

Total billable hours: 155.00 rate:
Regular Hours: 155.00 @ $45.50 = $7,052.50
Amount Due: $7,052.50

Enter another employee? ("y"=yes): n

Billing System Menu:

    0 - end
    1 - Enter billing data
    2 - Display ad-hoc billing report

Option ==>
```

Overlaid on the shell is a text editor window titled 'Billing.txt' with a menu bar (File, Edit, View) and a settings icon. It contains the following text:

```
Tony Stark
45.5
40.0
40.0
40.0
40.0
Earl grey
45.5
40.0
40.0
50.0
50.5
David Banner
45.5
35.0
40.0
40.0
40.0
```

The text editor window also shows status information at the bottom: 'Ln 1, Col 1 | 100% | Windows (CRLF) | UTF-8'.

**Figure 11 - Case test 10: Validate the billing.txt file is correct**

```
IDLE Shell 3.9.13
File Edit Shell Debug Options Window Help
Enter hours worked for week 4: 40
David Banner worked no overtime.
Invoice
Resource: David Banner Average weekly hours: 38.75
Total billable hours: 155.00 rate: $45.50
Regular Hours: 155.00 @ $45.50 = $7,052.50
Amount Due: $7,052.50
Enter another employee? ("y"=yes): n
Billing System Menu:
    0 - end
    1 - Enter billing data
    2 - Display ad-hoc billing report
Option ==> 2
Employee      Rate    Week 1  Week 2  Week 3  Week 4  Hours  Total
Tony Stark    $45.50  40.00  40.00  40.00  40.00  160.00 $7,280.00
Earl Grey     $45.50  40.00  40.00  50.00  50.50  180.50 $8,259.29
David Banner  $45.50  35.00  40.00  40.00  40.00  155.00 $7,052.50
Total Billable Due: $22,591.79
Total Billable Hours: 495.50
Average Billable Hours: 165.17
Billing System Menu:
    0 - end
    1 - Enter billing data
    2 - Display ad-hoc billing report
Option ==> 0
Program ended successfully.
>>>
```

**Figure 12 - Case test 11: Verify that the ad-hoc report program produces the correct report**