Intro to Python – Lesson 29

Today we want to look at an introduction to text files to store data. We saw that we can use lists to store multiple values, but since this is memory, it is destroyed when the program is ended or the computer is turned off. We need a way to store data permanently. Check out the following:

https://theintactone.com/2022/08/13/data-files-types/
https://www.youtube.com/watch?v=0C2405R-uGk

Note that text files are the simplest for of storing data – as you move forward in your program, you will see how databases and tables are much more efficient. Take a bit of time and research the difference between text files (also referred to as Flat Files) and Database tables.

Use the following sample to write data to a data file. In most cases, data files of this type are set up as comma separated files – which means each value is separated by commas. Also note that all values pertaining to a single transaction are generally recorded on one line – for example one invoice per line or one customer per line. In most cases when using files you will open the file, process the file – in this case write to it, then close the file. Note that numeric values are converted to strings as they are written to the file.

The **NL Chocolate Company** needs a program to process salesperson travel claims when they return from a business trip. As employees return from business trips they record all required information on a Travel Claim Form, and return the form, with all invoices, to the main office.

The program will process the travel claims returned to the office. Initialize a constant for Claim number (34) and the HST rate (15%).

Start by prompting the user to enter information from the Travel Claim Form including the employee number, name, location of the trip, start date, end date, the number of days (BONUS: rather than input the number of days use the start and end dates to calculate this value), a value to indicate if they used their own car, or if a car was rented (O or R), and the total kilometers traveled. Only enter the total kilometers if the employee used their own car. NOTE: No validations are required at this time.

Calculate the per diem amount by multiplying the total days by a daily rate of 85.00 for claims of 3 days or less, or by a daily rate of $100.00 for claims of 4 or more days. The mileage amount is calculated using a rate of .10 per kilometer if the salesperson used their car, or a rate of $56.00 per day is the salesperson rented a car. The Claim Amount is calculated as the Per Diem amount and the Mileage amount. The HST is calculated on the per diem amount only. The Claim Total is the Claim Amount plus the HST.

The program will display all input and calculated values to the screen as results. Only display the mileage amount if it is calculated. Just do a basic printout with headings and values.

*****Here is where the data file comes in – with a couple of other interesting concepts with the message and updating the constants*****

Write all the input values and the calculated values to a file called Claims.dat – separate each value in the row with a comma – this creates a comma separated file. Note that the only calculated value written to the file is the ClaimTotal. This is because any calculated value can be recalculated as needed since you have all the input values.

```
f = open("Claims.dat", "a")

f.write(f"{strClaimNum}, ")
f.write(f"{FV.FDateS(CurDate)}, ")   # This is the current system date – you may want to
                                        format it so that the time does not appear.
f.write(f"{EmpNum}, ")
f.write(f"{EmpName}, ")
f.write(f"{Location}, ")
f.write(f"{ FV.FDateS(StartDate)}, ")
f.write(f"{ FV.FDateS(EndDate)}, ")
f.write(f"{str(NumDays)}, ")
ClaimTotal= round(ClaimTotal, 2)  # May want to round floats to remove excess decimals.
 f.write(f"{str(ClaimTotal)}\n")

f.close()
```

Display a message for the user indicating that the claim information has been saved (Could get fancy with a flashing message or progress bar).

Look at the constants and see if any of the values need to be updated. Update the claim number by adding 1 to it. Now the claim numbers will be sequential in the file.

Repeat the program until the user enters the word END for the employee number.

Use the following exercises for additional practise on creating data files. In each case set up the inputs – validations are not required but you can add a few for practise if you want.

1. Billy Bob Bike Rentals requires a program to process the bike rentals he has made for the day. Required are the invoice number (Start with 1358 – set up as a constant at the beginning of the program), the customer's name (must be entered), a phone number (must be entered and be 10 digits), a code for the type of bike rented (T for 12-Speed, M for Mountain, the number of bicycles rented (must be between 1 and 3), a credit card number, and the expiry

date. No calculations are required at this point. Write the values to a file called Rentals.dat using comma separated values across a single line. Increment the invoice number by 1 and prompt the user if they want to enter another rental (Y / N).

2. Modern Movie Rentals wants a program designed that will allow them to add new movies they purchase to a data file called Movies.dat. Set up a constant for the Movie ID (Start with 10285), and allow the user to input the Movie name, the movie type - can only be one of "D" for Drama, "C" for Comedy, "M" for musical, or "H" for Horror. Once a valid type has been entered, the name of the type should appear next to the type. The movie rating should be entered - only as "G" for General, "P" for Parental Guidance, or "R" for Restricted, and finally the Rental cost - the lowest rental price for a movie is $1.99 and the most expensive rental priced movie is $8.99. Write the values to a file called Movies.dat using comma separated values across a single line. Display a message telling the user the movie has been saved. Increase the movie ID by one. Allow the program to repeat until the user enters "END" for the Movie ID.

Do you see a problem with these programs? End the program and start again adding a couple of new lines to the file. What happens to the Invoice Number and the Movie Number?

The question now is how do you suggest we can solve this problem?

See you at 1.