CS417 Lab #3

This lab an exercise in reading and writing text files.

You will compute the payroll for a small company. You must read a text file describing the hours worked by several employees, and write another text file that includes their pay, and their tax.

Getting Started

Begin the lab by creating a directory for your work. Then download two files:

- `compute_pay.py` : a program that you will edit
- `timesheet.txt` : a file listing employee information, and hours worked.

Open `timesheet.txt`, using a text editor (*e.g.*, IDLE). Notice that each line contains four fields, separated by a colon. They are:

```
field #0 last_name
field #1 first_name
field #2 hours_worked
field #3 hourly_pay
```

Example: the first line has this text:

```
Bagley:Malinda R:twenty:12
```

Your Tasks

1. Open and run `compute_pay.py` . It currently opens `timesheet.txt`, reads each line of the file, and prints it.

   What you should do:

   Notice the spaces between between lines in the output. This is because `readline()` returns a whole line in the file, including the line's newline character at its end. Remove the newline thus:

   ```
   line = line.rstrip('\n\r')
   ```

2. Split the line into its fields. REPLACE the **indented** `print` line with this

(also indented, of course):

```
    fields = line.split()
    print (fields)
```

3. This doesn't do what we want, because `split()` looks for spaces to break up the string. Replace the above with

```
    fields = line.split(':')
```

4. Let's name the fields. Notice that all fields are `str`ings. We need some of them to be `float`s:

```
    last_name = fields[0]
    first_name = fields[1]
    hours_worked = float(fields[2])
    hourly_pay = float(fields[3])
```

5. The program crashes! The problem is on the very first line of data. Notice the third field: `"twenty"` not a valid number.

```
Bagley:Malinda R:twenty:12
```

   We need to handle bad data. To do this, wrap the two `float()` calls in a try-except block:

```
hours_worked = 0
hourly_pay = 0
try:
    hours_worked = float(fields[2])
    hourly_pay = float(fields[3])
except ValueError:
    sys.stderr.write('Bad number in timesheet. ' + line + '\n')
```

   Now, bad calls to `float()` don't crash the program, and the two fields `hours_worked` and `hourly_pay` have default values (zeros).

6. Note: the `write()` function, unlike the `print()` function, requires a `str`ing, not a sequence of fields separated by commas. Notice also that we need a newline at the end of the string.

   Unfortunately, the program *still* crashes because we forgot to import the `sys` module. Add this line to the top of your program:

```
import sys
```

7. Compute the gross pay. It's `hours * pay`, for the first 40 hours. Any hours over 40 should be paid at 'time and a half', 1.5 times the hourly pay rate.

```
gross_pay = (do it yourself.  More than one line of python)
```

Suggestion: create two variables: `regular_hours` and `overtime_hours`.

8. Compute the taxes owed.

```
tax = gross_pay * TAX_RATE
```

Oops! Forgot to define `TAX_RATE`. Add this near the top:

```
TAX_RATE = 0.20
```

Constant values should be in UPPER CASE. It is bad to use raw numbers in the middle of your code. Don't do this:

```
tax = gross_pay * 0.20
```

Tax rates might change; you would have to modify *many* lines of code where `0.20` appeared. Better to create a constant *ONCE* at the top, and reuse it everywhere.

9. Now, let's print the following fields, separated by colons : last name, first name, gross pay, tax, net pay (= gross pay - tax). Print out one big string, with those fields joined by ':' . Use a `.format()` call, thus:

```
print ("{}:{}:{}:{}:{}:{}".format(last_name, first_name,
                                  gross_pay, tax, net_pay)
```

10. Finally, let's save all this output into a file. At the top of the program, open the output file:

```
payroll = open(payroll_file_name, 'w')
```

Notice the second parameter `'w'`, which means we will OVERWRITE the file. When you overwrite a file, all its original content will be lost!

Now, replace the

```
print(last_name + ... )
```

call with a call to `payroll.write(last_name + ...)` .

11. Look at the file `payroll.txt`, with the editor. It probably looks wrong, because our `write()` call was missing a trailing newline `'\n'`. Add it to the `format` specifier.

12. Don't forget to close the payroll file too, at the end of the program.

Turn in your work

To turn you work in, go to `mycourses.unh.edu`, find CS417 and lab #3, click the "Submit" button, and upload `compute_pay.py` . At the end of the lab session, submit any work you have completed. You can submit again until midnight, with no lateness penalty.

Save Your Work

Before you leave the lab, upload your file to Box, email it to yourself, or save it on a USB thumb drive.