CS417 Lab #4

Getting Started

Begin the lab by downloading a solution to the previous lab. It computes the payroll for several employees:

- `compute_pay2.py`

Your task is the same as last time: compute the payroll for all the employees.

Modifications

In the previous lab, there was a single input file. Now, there are *two files*. As in the previous lab, the files consist of text lines, with fields separated by colons ':'. Download these files, and you will see that the fields are different from last time:

- `employee.txt`
- `timesheet.txt`

`employee.txt` now has these fields:

```
0 employee_id
1 last_name
2 first_name
3 hourly_pay
```

`timesheet.txt` now has these fields:

```
0 employee_id
1 hours_worked
```

In other words, the employee's personal information is kept separate from their hours worked.

The output file `payroll.txt` also has changed, and will now have these fields:

```
0 employee_id
1 gross_pay
2 tax
3 net_pay
```

Goals

You must make two major changes:

- The program must read the information from *two* files, not just one.
- The program is currently hardwired to read from `timesheet.txt` and write to `payroll.txt`. Let's get the file names from the command line instead.

Exercises

1. When the user types

   ```
   python compute_pay2.py timesheet.txt employee.txt payroll.txt
   ```

   there are 3 command-line arguments, in `sys.argv`. To check for this:

   ```
   if len(sys.argv) == 4:
   ```

   then `sys.argv[1]`, `sys.argv[2]`, and `sys.argv[3]`, will be the timesheet file, the employee file, and the payroll file, respectively:

   ```
   timesheet_file = sys.argv[1]
   employee_file = sys.argv[1]
   payroll_file = sys.argv[1]
   ```

   Else, prompt the user for the three filenames:

   ```
   timesheet_file = input("timesheet file? ")
   ...
   ```

   Hence the function `compute_pay()` should now have *three* parameters (the 3 filenames). Look for `Item 1` in the file: you need to change `main()` and also `compute_pay()`.

2. Since the hourly rate is now stored in the employee file, you will have to read that file, and create a dictionary for the hourly rates.

   A dictionary is a hash table. It looks like an array, in the sense that you can access its entries using `[]` square brackets. In an array, you can write `array_name[index]` to access an entry, but index must be an `int`.

   Think of a dictionary as an array, where the indexes can be of any type, not just `int`.

Add a function `get_hourly_rates(filename)` which
- creates a dictionary: `hourly_rates = dict()`
- opens the given file,
- reads each line in the file, until end-of-file (empty line),
- uses `.split(':')` to get the fields in the line,
- gets the `employee_id` from field `0`,
- gets the `rate` from field `3`,
- adds it to the dictionary: `hourly_rates[employee_id] = rate`
- and finally **returns `hourly_rates`**

Don't forget to close the employee file!

In `compute_pay()`, call the function:

```
hourly_rates = get_hourly_rates(employee_file)
```

3. You can now ignore the employee names. In the function `compute_pay()`, just get the `employee_id` from field `0`, and the `hours_worked` from field `1`. Then, fetch the rate from the dictionary:

```
hourly_rate = hourly_rates[employee_id]
```

4. Finally, since the output for the payroll file has changed, you will have to change the `write()` call.

5. Check your output

   Given the new files `timesheet.txt` and `employee.txt`, the file `payroll.txt` should read as follows:

```
1:0.00:0.00:0.00
2:712.50:142.50:570.00
3:420.00:84.00:336.00
4:830.81:166.16:664.65
5:631.25:126.25:505.00
6:270.00:54.00:216.00
7:500.00:100.00:400.00
8:593.75:118.75:475.00
9:570.00:114.00:456.00
10:570.00:114.00:456.00
11:300.00:60.00:240.00
12:0.00:0.00:0.00
```

Turn in your work

To turn you work in, go to mycourses.unh.edu, find CS417 and the lab, click the

"Submit" button, and upload `compute_pay2.py` . At the end of the lab session, submit any work you have completed. You can submit again until midnight, with no lateness penalty.