Object Oriented Programming In Python

Answers for Activity Sheet: Week 1

Completed programs are available to download.

Task 1A: In pairs, look at the program below and complete the table of identifiers. (*The program is available as fileRead.py*)

```
fn = input("Name of file to be read> ")
f = open(fn, 'r')
nline = f.readline()
lineNum = 1
while len(nline) > 0:
     name = nline.split("\n")[0]
     if len(name) > 0:
         print(lineNum, ":", name)
     else:
         print(lineNum, "is blank")
     lineNum += 1
     nline = f.readline()
f.close()
```

Table of Identifiers

Name	Type	Description
input	Function	Prompt user for a string
fn	string	String – name of file from user
open	Function	Opens a fie, creating a new 'File' object
f	File object	Object for the file named by user
readline	Method of File class	Reads one line, including end of line characters; empty string at end of file
nline	String	Line read from file, including end of line characters
lineNum	Integer	Line number
len	Function	Length of (here) a string
split	Method of String class	Splits a string into a list of string, at separator
print	Function	Print to terminal
close	Method of file class	Close file: discards object and subsequent reads / write will fail. Flushes write buffer.

Task 2:

List the functions and methods of the file class that provide the abstract view of text files. (Check Python documentation to see if there are more than the ones you have used; if so, decide whether they are useful.)

The abstraction interface to file is given by the following methods/functions.

Method/Function	Description
f = open(*string*, *mode*)	Open the file to read (mode='r') or write (mode='w') or both.
f.readline()	Read one line, including the newline character.
f.readlines()	Read al the lines, giving a list.
f.write(*string*)	Write the string. The string should include '\n' when a new line is needed.
f.close()	Close the file.

Some other methods are also available but are usually not needed.

Method/Function	Description
f.flush()	When writing, causes any buffered txt to be written.
f.read(*size*)	Read up to 'size' characters; may include new lines.

Task 3: Better Agency

Task 3.2 The program from Task 3.1 has some obvious limitations.

- *You discover Jo's location but cannot update it.*
- *Then Bert leaves the agency but cannot be deleted.*

Imagine that the Person class is created by a supplier ('Bill The Coder') so that you cannot change it but have to ask for any changes needed. Are your problems best solved by you (i.e. by changing how you use the Person class) or do they require Bill to change the Person class (or both, of course)?

The problem of updating or deleting a Person from the list is to find them. We could solve the problem by keep a separate record of the names of people we create (maybe using a dictionary for example). We could even resort to reading the name from the description! But it would be easier if it were possible to find the name of a Person. We might also want to list the people in a particular place or the people doing a particular job, so it would be useful to be able to have these methods too.

Write a list of the changes you need from Bill.

New Methods	Description
p.getName()	Return p's name – string
p.getJob()	Return p's job – string
p.getLocation()	Return p's location – string
p.getDriver()	Return p's driving status – true or false

Task 4: Nearly Person Agency

Task 4.2: Re-implement you improved agency (task 3.1) to use NearlyPerson instead or Person. If this gets repetitive, write out a list of rules so that a very junior programmer could do the work for you.

When you see this	Replace it with this
p = Person(*name*)	*p* = newPerson(*name*)
p.setJob(*job*)	*p* = setJob(*p*, *job*)
p.setLocation(*location*)	*p* = setLocation(p, *location*)
p.setDriver(*bool*)	*p* = setDriver(p, *bool*)
p.describe()	describe(*p*)