Object Oriented Programming In Python

Activity Sheet: Week 1

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	
fn	string	
open	Function	
f	File object	
readline	Method of File class	
nline		
lineNum		
len		
split		
print		
close		

Task 1B: You need a suitable test file, with one word on each line, perhaps including some blank lines. Create such a file and run the program.

Task 2: Write a program that:

- Reads names from a file – on name per line (see program readFile.py)
- Uses each names as the name of a file (add extension .txt) and write "this is file NAME" into the file (see program writeFile.py)

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.)

Task 3: Better Agency

The Agency wants to expand beyond Bert and Jo. (See program agency.py)

Task 3.1 Write an interactive program that prompts the user to add details of a new Person to the list of people the Agency knows about or to be able to print the description of all the people in the list.

- The program should handle the case (as in Jo) where some details are not known.
- It is suggested that create a list of people and append each new person you create.

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)?

Write a list of the changes you need from Bill.

New Methods	Description

Task 4: Nearly Person Agency

Re-implement the Agency to use the NearlyPerson module rather than the Person class. Do not copy the NearlyPerson code; instead import it.

Task 4.1: Re-implement the original agency.py with just Bert and Jo using NearlyPerson instead of Person

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

Task 4.3: You want to add a telephone number to the details of a NearlyPerson. Modify the NearlyPerson.py to do this and your application (Task 3.1) to include it.

Homework Tasks

Homework Task 1: Date and Time

Date and time are quite complex. For example, a date can be a day number or a month number and a day number. A time can use a 12-hour or 24-hour clock and summer time or GMT. We may also want to compare two dates/time (before or after) or subtract on form the other (to get a time interval).

- Suggest a simple abstract interface (i.e. functions) that could be used to create dates and times and show them in different formats. Do not include every possible complication!
- Suggest how the date/time might be represented, behind your interface.
- Implement (some of) the interface as a module (i.e. in the from of NearlyPerson.py)

Homework Task 2: Functional Decomposition

Your students have learnt how to define functions but some struggle to choose which functions will be helpful.

- Suggest two forms of exercise that could help them practice using functional decomposition.
- Given an example exercise in at least one of these forms