Reading and Research - Lists

These tasks are designed to introduce you to the programming topic we will be studying in class next lesson. You **must** complete these activities prior to the lesson.

Records

A record is a data structure that groups together a number of variables which do not need to be of the same type and have no associated ordering. The variables of a record are called **attributes** or **fields**.

To use records in Python, we define the data structure that we want to create as a Class, and then declare variables of the class to use in the program. To work with the individual attributes of a record we use dot notation `ClassVariableName.AttributeName`:

```python
class Book():
    def __init__(self):
        self.title= None
        self.isbn= None
        self.price= 0
        self.year_of_publication=0

aBook = Book()
aBook.title = 'A-level Computing'
```

A record is implemented as a `class` in Python.

Classes use a built-in function `__init__(self)` to create the attributes. `__init__` has a **double underscore** before and after it.

Other methods (or functions) can be declared within the class definition, but only `__init__` is needed if the class is to be used only as a record to hold data.

You can declare a **variable** that is an instance of the class Book.

A value can be assigned directly to the attributes of an instance of the class.

You can also declare an array of records by using a list of records – each item in the list is a record; each record in the list has the same set of attributes (each attribute holding an independent value).

Investigating the use of records

A programmer has begun to develop a program for a phone book that will store the contact numbers of people. The program uses two lists – one that stores the names of the contacts and one that stores the telephone numbers.

The partially developed program is in the file `PhoneBook.py`.

- A main program has been developed. This creates lists for names and numbers that can be used for testing the program during its development.
- The function AddEntry has been completed.
- The function GetContactIndex contains only a return statement. This is so that the program has correct syntax, but it does not work correctly because this function should find the index number of the contact name in the list of contacts.
- The functions EditEntry, DeleteEntry and FindNumber call the function GetContactIndex.
- The functions EditEntry and DeleteEntry are incomplete.

To be able to look up a number, edit a contact or delete a contact, the program needs to be able to find the record in the list of contacts, and then amend the name or number for that record.

The programming team has developed the pseudocode to use in the next stage of the development of the program. This is shown on the next page.

Task 1

Explain what the following pseudocode does:

```
#Pseudocode for a linear search
contactList: List
FUNCTION GetContactIndex( )
    nameToFind: String
    index: Integer
    OUTPUT 'Please enter contact name:  '
    INPUT nameToFind
    FOR index ← 1 to LENGTH(contactList) DO
        IF contactList[index].name = nameToFind THEN
            RETURN index
    END FOR
    RETURN -1
END FUNCTION
```

NOTE: Pseudocode indexes lists and arrays starting from 1. Many programming languages, including Python, start from an index value of 0.

*This is a **Linear search** algorithm. The first line(starting from nameToFind) is saying that the variable is a string and the second line is saying that the variable is an integer. The "FOR" loop is in the range of 1 to the length of the list. The next line is check for each user and checking if any contacts are matching.*

Task 2

Use the pseudocode to develop the function `GetContactIndex` in `PhoneBook.py`.

Paste your code for the function GetContactIndex in the space below:

```python
#space for your code

def GetContactIndex(contactList):
    nameToFind = input("Please enter contact name: ")
    for index in range(0, len(contactList)):
        if contactList[index].name == nameToFind:
            print(contactList[index].number)
            return index
    return -1
```

Task 3

Test your solution when option 4 (Look up a number) is selected by the user. Complete the test plan and results in the table below.

Name searched for	Expected result	Actual result
Isabelle	07866810803	07866810803
Rio	07648139073	07648139073

Task 4

Develop your PhoneBook program further by implementing the functions `DeleteEntry` and `EditEntry`.

Paste your code for the function `DeleteEntry` in the space below:

```
#space for your code
def DeleteEntry():
    indexSought = GetContactIndex()
    change =  int(input("Please enter which contact you want to
delete: "))
    contactList[change - 1 ].name.pop())
```

Paste your code for the function `EditEntry` in the space below:

```
#space for your code
def EditEntry():
    indexSought = GetContactIndex()
    change =  int(input("Please enter which contact you want to
edit: "))
    get_name = input("Please enter the contact name: ")
    get_number = int(input("Please enter contact number: "))
    contactList[change].name.insert(get_name,get_number)
    contactList[change + 1].name.pop
```

> Written with StackEdit.