# Python and Scripting Languages

## Instructions

The following exercises require you to use functions, types and classes from the Python standard library.

## Exercise 1

Given a string `str`, we define its *ciao* (*characters in alphabetical order*) as the string having the same length of `str` and containing all the characters of `str` in lower case and alphabetical order. A *ciao string* is a string that is equal to its *ciao*.

1. Write a function `ciaoWord` that given a string returns its *ciao*. For example,

```
ciaoWord("Hello") = "ehllo"
```

1. Implement the function `addCiaoDict` that takes as arguments a dictionary `ciaoDict` and a string `word`. The dictionary `ciaoDict` is assumed to have *ciao strings* as keys and sets of strings as values: it maps a *ciao string* `str` into a set of strings all having `str` as their *ciao*. The function `addCiaoDict` must add string `word` to the `ciaoDict` dictionary.

   **Note**: You have to correctly handle the case when the *ciao* of `word` does not exist as key in `ciaoDict`.

   **Question**: What is the relationship among the strings having the same *ciao*?

2. Write the function `createDict` that takes as argument the path of a text file `f` containing a list of words (you can assume that each line contains a single word). The function returns a dictionary mapping a *ciao string* `str` to the set of words of `f` having `str` as *ciao*.

   **Note**: You may use file anagram.txt to test your function. You have to correctly handle white spaces characters like new lines and tabs. Refer to the documentation for information about reading and writing files.

3. Finally, write the function `replaceAnagrams` that takes as arguments a dictionary `dict` computed through `createDict` and a string `line` representing a line of text (i.e. a sequence of words separated by spaces). This function returns a text line where each word `w1` of `line` is replaced by a different word `w2` having the same *ciao* of `w1` if this *ciao* exists in `dict`; otherwise, `w1` is left unchanged.

**Goals:** Fun with Python and warming up!

**Expected output:** Properly commented Python scripts along with the required implementations.

## Exercise 2

This is a conceptual exercise that requires you to use the Python interpreter. Create a list `numbers` containing 2,4,6,8. Create an iterator `it1` to iterate over the elements of `numbers`. Using the `map` function create a new iterator `it2` that maps the elements of `it1` using the following lambda:

```
lambda x: x ** 4
```

What is the result of the following two statements?

```
next(it2)
next(it1)
```

Modify the list `numbers` as follow

```
numbers[2] = 0
numbers[3] = "Hello"
```

What is the result of the following statement? Why?

```
next(it2)
next(it2)
```

**Goal:** Experimenting with Python and its inner mechanisms.

**Expected output:** The actual output and an explanation of the obtained behaviour.

# Exercise 3

This is essentially Exercise 10 of the Nov. 26 class. If you already did it in Java, port your solution from Java to Python (to deal with optional fields in Python use `None`). Otherwise here is the text.

Consider the csv file people.csv. This file stores information about people subscribed to a simple service. Each line of the file represents a record. The fields of each record are separated by a comma "," and have the following meaning:

```
id,firstname,surname,title,address,town,country,postcode,subscription
paid,gender,date of birth
```

In a record the fields `id`, `firstname`, `surname`, `date of birth`, `subscription paid` are mandatory, while the others are optional (denoted by "-" in the file).

Implement a class `Subscriber` representing a subscriber, providing an instance attribute for each field and a suitable constructor. Use the `None` value for optional fields.

Write a function `loadDatabase` that returns a list of `Subscriber` containing the records of the file `people.csv`. Furthermore, implement another function `PaymentFromGB` that given a list of `Subscriber` prints the subscribers from GB that have paid the annual fee.

**Suggestion:** Look at the `csv` module of the Standard Library of Python.

**Goal:** Experimenting with more advanced Python mechanisms.

**Expected output:** Properly commented Python file with the required implementations.

Author: Andrea Corradini & Matteo Busi

Created: 2018-12-10 lun 10:09

Validate