Assignment #7: Tuples

Master in Informatics and Computing Engineering
Programming Fundamentals
Instance: 2018/2019

0. Introduction

***Goals**: To write functions using tuples*

***Pre-requirements (prior knowledge)**: See bibliography of Lecture #11*

***Rules**: You may work with colleagues, however, each student must write and submit in Moodle his or her this assignment separately. Be sure to indicate with whom you have worked. We may run tools to detect plagiarism (e.g.duplicate code submitted)*

***Deadline**: 8:00 Monday of the week after (19/11/2018)*

***Submission**: to submit, first pack your files in a folder `RE07`, then compress it with zip to a file with name `2018xxxxx.zip` (your_code.zip) and last (before the deadline) go to the Moodle activity (**you have only 2 attempts**)*

1. Unique

Write a Python function `unique(atuple)` that receives a tuple of integers and returns a tuple with the sorted unique elements of the tuple.

Save the program in the file `unique.py`

For example:

- for atuple=`(8, 8, 1, 3, 1, 3, 5)` the function returns the tuple `(1, 3, 5, 8)`
- for atuple=`(1, 1, 1, 1)` the function returns the tuple `(1,)`

Hint: For sorting, you may use the built-in function `sorted()`

2. Find my Type

Write a Python function `find_dtype(atuple, data_type)` that, given a tuple `atuple`, returns another tuple containing just the elements of type `data_type`. It should be assumed that `data_type` is a string with one of the following values: 'int', 'float', 'complex', 'bool', 'str' or 'tuple'.

Save the program in the file `find_dtype.py`

For example:

- for atuple=`(1, False, "hello", 2., "world")` and data_type=`"str"`, the function returns the tuple `("hello", "world")`
- for atuple=`(1, 2, 3)` and data_type=`"float"`, the function returns the empty tuple `()`

Hint: Look at `__name__` in the Python Standard Library

3. Translation table

Write a Python function **`translate(astring, table)`** that translates a given string **`astring`** using a translation table. The translation table, **`table`**, is a nested tuple with an arbitrary number of translation pairs/tuples *(in_value, out_value)*.

Save the program in the file **`translate.py`**

For example:

- for **`astring="Hello world!"`** and **`table=(('a', 1), ('e', 2), ('i', 3), ('o', 4), ('u', 5), ('!', ' :)'))`**, the function returns the string **`"H2ll4 w4rld :)"`** (without quotes)
- for **`astring="Testing this string..."`** and **`table=((' ', '--'), ('.', '!'), ('i', 'o'), ('t', 'tt'))`**, the function returns the string **`"Testtong--tthos--sttrong!!!"`** (without quotes)

4. Zero-sum Triplet

Given a tuple of **`n`** integers, with **`n`** > 3, write a Python function **`triplet(atuple)`** that finds a triplet (a, b, c) such that their sum is zero (i.e., a + b + c = 0).

In case there is more than one triplet that sums up to zero, you should return the triplet with the lowest index (idx), such that (idx_a , idx_b , idx_c) < (idx_{an} , idx_{bn} , idx_{cn}). In case there are no triplets that sum up to zero, you should return an empty tuple ().

Save the program in the file **`triplet.py`**

For example:

- for **`atuple=(-8, 0, 4, -2, -1, 1, 2)`**, the function returns the tuple **`(0, -2, 2)`**
- for **`atuple=(-1, 1, 1, 1)`**, the function returns the tuple **`()`**
- for **`atuple=(-4, 3, 0, -2, -1, -3)`**, the function returns the tuple **`(3, 0, -3)`**

5. Students grades

Consider student records, given as a nested tuple of N tuples in the format *(name, number, (grade_1, grade_2, …))*. Write a Python function `sort_grades(records)` to sort the student records according to the following priority criteria:

1. Sort based on the average grade (descending order);
2. Then sort based on student name (ascending order);
3. Then sort based on student number (ascending order);

where *name* and *number* are strings, and each *grade* is an integer between 0 and 100.

Save the program in the file `grades.py`

For example:

- for
```
records=(('João', 'up20186042', (90, 87)),
         ('Ana', 'up20186040', (90, 90)),
         ('José', 'up20186063', (70, 90)),
         ('Ana', 'up20186061', (90, 90)),
         ('Tiago', 'up20186070', (100, 90)))
```
  the function returns the tuple:
```
(('Tiago', 'up20186070', (100, 90)),
 ('Ana', 'up20186040', (90, 90)),
 ('Ana', 'up20186061', (90, 90)),
 ('João', 'up20186042', (90, 87)),
 ('José', 'up20186063', (70, 90)))
```

- for
```
records=(('Maria', 'up20190001', (60, 70, 80)),
         ('Maria', 'up20190002', (60, 70, 80)),
         ('Mario', 'up20190003', (100, 10, 80)),
         ('Rui', 'up20190004', (90, 100, 90)),
         ('Ana', 'up20190005', (90, 100, 90)))
```
  the function returns the tuple:
```
(('Ana', 'up20190005', (90, 100, 90)),
 ('Rui', 'up20190004', (90, 100, 90)),
 ('Maria', 'up20190001', (60, 70, 80)),
 ('Maria', 'up20190002', (60, 70, 80)),
 ('Mario', 'up20190003', (100, 10, 80)))
```

Hint: For sorting, you should use the built-in function `sorted()`, in which you can define your own key-sorting function.

The end.

FPRO, 2018/19