Assignment #13: Exceptions

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

***Goals**: write programs that deal with exceptions*

***Pre-requirements (prior knowledge)**: see bibliography of Lecture #23*

***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. duplicated code submitted)*

***Deadline**: 8:00 Monday of the week after (14/01/2019)*

***Submission**: to submit, first pack your files in a folder RE13, 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. Exception string

Write a function `exception_str(f)` that receives a function `f` and if the function raises an exception then it should return that exception as a string. Otherwise, it should return `"no exception was raised"`.

Save the program in the file `exception_str.py`

For example:

- `exception_str(lambda: 1/0)` returns the string `"division by zero"`
- `exception_str(lambda: 0)` returns the string `"No exception was raised"`

2. Exception counter

Write a function `count_exceptions(f, n1, n2)` that receives a function `f` and two integers `n1` and `n2`. Function `f` accepts an integer and may throw an exception when applied to an invalid integer. The given function is applied to all integers in the range [`n1`, `n2`] and returns the amount of exceptions caught.

Save the program in the file `count_exceptions.py`

For example:

- `count_exceptions(lambda x: 1/(abs(x)-2), -5, 5)` returns the integer `2`
- `count_exceptions(lambda x: 1/0 if x > 10 else 0, 0, 50)` returns the integer `40`

3. Compatible Matrices

Write a Python function **`compatible(A, B)`** that receives two matrices A and B, given as nested lists (each list represents a row), and checks if A and B can be multiplied. When the matrices cannot be multiplied, the function returns an **AssertionError** exception with the following error message **`"A and B cannot be multiplied"`**. Otherwise, the function returns the string **`"A and B can be multiplied"`**.

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

For example:

- **`compatible([[2,2], [1,1]], [[5,5,5,5], [5,5,5,5]])`** returns the string **`"A and B can be multiplied"`**
- **`compatible([[1,2,2], [1,2,2]], [[1,2,3,4], [1,2,3,4]])`** returns the an exception with the following error message **`"A and B cannot be multiplied"`**

4. Raising your own exceptions

Write a Python function **`raise_exception(alist, value)`** that receives a list of integers **`alist`** and an integer **`value`** and raises an exception for each element of **`alist`** that is not greater than **`value`**. The function should return a list with the raised exceptions, which are instances of the **`ValueError`** exception type with the following string:

`'<elem> is not greater than <value>'`,

where **`<elem>`** is an element of **`alist`** and **`<value>`** is the input **`value`**.

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

For example:

- **`raise_exception([1, -2, 3, 0], 3)`** returns the list **`[ValueError('1 is not greater than 3',), ValueError('-2 is not greater than 3',), ValueError('3 is not greater than 3',), ValueError('0 is not greater than 3',)]`**
- **`raise_exception([3], 2)`** returns the list **`[]`**

5. Recursive exceptions

Write a function `rec_exceptions(l)` that receives a list `l` of functions. When you call each one of these functions, it will either (a) also return a list of functions or (b) raise an exception. You should transverse the list until you have caught all exceptions and return them in a flat list.

Save the program in the file `rec_exceptions.py`

For example:

- `rec_exceptions([lambda: 1/0])` returns the list `[ZeroDivisionError('division by zero',)]`
- `rec_exceptions([lambda: [lambda: [1,2,3].index(-1), lambda: ''[2]], lambda: [1,2,3][4], lambda: [lambda: [lambda: 1/0]]])` returns the list `[ValueError('-1 is not in list',), IndexError('string index out of range',), IndexError('list index out of range',), ZeroDivisionError('division by zero',)]`

The end.

FPRO, 2018/19