Lab 09: Exceptions, Special Methods

Total Points: 10 (Correctness) + 1 extra credit (optional)
Due: Tuesday, March 12th, 11:59pm (SD time)

Starter Files

Download lab09.py. Inside the archive, you will find starter files for the questions in this lab. You can't import anything to solve the problems.

Extra Credit Opportunity

You have the opportunity to receive a **1 point** extra credit on that lab if you submit your last attempt early (refer to this section of each lab for the early submission deadline). Note: Each lab is graded out of 10 points, and you could possibly have more than 10 points in one lab.

Early Submission Date (lab09): Friday, March 8th, 11:59pm (SD time)

Testing

After finishing each question, run the file with the following command-line options to avoid compile time errors and test the correctness of your implementation:

- No options: `>>> python3 lab09.py`
- Interactive mode: `>>> python3 -i lab09.py`
- Doctest **(Recommended)**: `>>> python3 -m doctest lab09.py`

For Windows users, please use `py` or `python` instead of `python3`.

Important Requirements and Notes

1. **DO NOT IMPORT ANY PACKAGES.**
2. It may be useful to add doctests when the given doctests do not cover all cases to avoid losing points from the Autograder/any hidden tests, but you are not required to add doctests in the labs.
3. Style/asserts are not required or graded on any labs, but it's recommended that you also follow the style guide for clean code.

4. You may assume all inputs are valid.

Required Questions

Congratulations! You have explored most of the beautiful Pythonia land and you are almost ready to step into Javania – hope it's been an enjoyable journey for you! However, the exploration in Javania is not easy. Before registering you in Javania, the security guard checks your experience in Pythonia to ensure your safety.

Question 1:

You are now moving to Javania. To be eligible for admission into Javania, you will need to fill out a survey about your experience level (in discrete scales 10, 20, 30, etc.) for selected landscapes in Pythonia. Then, the security guard checks some conditions and tells you if you will be admitted. Before getting evaluated by the security guard, you want to make sure that you will pass the survey. You decide to simulate the survey yourself before the immigration checkpoint.

For this question, you have prepared a dictionary where keys are landscapes and values are corresponding experience levels. Write a function that checks your eligibility using **exceptions** and NOT assertions. If a check fails, you will throw an exception using **raise**. You need to make sure you are doing the following checks **in order** to prevent issues with the doctests expecting the wrong exception.

1. `xp_levels` should be a dictionary. If it is not, raise an `TypeError` exception with a message "Survey result is represented in a wrong way!"
2. Check **all** landscapes in `xp_levels` are strings.
    a. If there are multiple invalid landscape representations, you only need to throw an `TypeError` exception for the first invalid landscape encountered, along with a message "Landscape is represented in a wrong way!"
3. Check **all** experience levels (dictionary values) are integers, each a positive value that is a multiple of 10 (e.g. 10, 20, 30, 40).
    a. If there are multiple invalid experience levels (of a type other than int), you only need to throw a `TypeError` exception for the first invalid experience level encountered, along with a message "Experience level for <key> is represented in a wrong way!"
    b. If there are multiple experience level values that are not a multiple of 10, you only need to throw an `ValueError` exception for the first such experience level encountered, along with a message "Experience level for <key> is incorrectly entered!"

4. Check **all** landscapes are in `landscapes` (a list provided in the starter file).
    a. If there are multiple landscapes in the dictionary but not in the selected list of landscapes, you only need to throw an `KeyError` exception for the first non-matching landscape encountered, along with a message "Landscape <key> does not exist in Pythonia!"
5. Check you have visited **at least 6** out of `num_landscapes` of the selected landscapes – security guards might reject you if you miss too much experience in Pythonia.
    a. If more than six landscapes are missing, raise an `KeyError` exception with a message "You miss too many Pythonia landscapes!"

Upon passing all checks, the security guards will evaluate your ability to survive in Javania. Your function will need to check the following conditions and return `True` if they pass, and `False` otherwise.

- You have visited **at least 9** landscapes in total.
- The average experience level across all landscapes you have visited is greater than or equal to 20.

The following data structures are provided:

- `landscapes`: a list of strings representing a subset of landscapes you have visited.
- `num_landscapes`: integer representing the total number of landscapes in `landscapes`.

```
def admission_checker(xp_levels):

    """
    >>> xp_levels = {'Return vs Print': 20, 'Dictionaries': 20,
'Files': 20, 'List Comprehension': 20, 'Lambda Functions': 20,
'Iterators': 20, 'Complexity': 20, 'Scope of variables': 20, 'Classes
and Objects': 20, 'Inheritance': 20, 'Special Methods': 20, 'Sorting':
20}
    >>> admission_checker(xp_levels)
    Traceback (most recent call last):
    ...
    KeyError: 'Landscape Sorting does not exist in Pythonia!'
    >>> xp_levels = {'Return vs Print': 20, 'Dictionaries': 20,
'Files': 20, 'List Comprehension': 20, 'Lambda Functions': 20,
'Iterators': 20, 'Complexity': 20, 'Scope of variables': 20, 'Classes
```

```
and Objects': 20, 'Inheritance': 20, 'Special Methods': 20, ('Mutable
data',): 20}
    >>> admission_checker(xp_levels)
    Traceback (most recent call last):
    ...
    TypeError: Landscape is represented in a wrong way!
    >>> xp_levels = {'Return vs Print': 20, 'Dictionaries': 20,
'Files': 20, 'List Comprehension': 10, 'Lambda Functions': 10,
'Iterators': 10, 'Complexity': 10, 'Scope of variables': 10, 'Classes
and Objects': 10, 'Inheritance': 10, 'Special Methods': 10, 'Mutable
data': 10}
    >>> admission_checker(xp_levels)
    False
    >>> xp_levels = {'Return vs Print': 20, 'Dictionaries': 20,
'Files': 20, 'List Comprehension': 20, 'Lambda Functions': 20,
'Iterators': 20, 'Complexity': 20, 'Scope of variables': 20, 'Classes
and Objects': 20, 'Inheritance': 20, 'Special Methods': 20, 'Mutable
data': 20}
    >>> admission_checker(xp_levels)
    True
    """
```

Question 2:

Javania truly values work-life balance, and that's why players can win various prizes based on their XP points. Consequently, we aim to implement special methods to facilitate a clearer and more convenient comparison of XP points.

Question 2.1 Initialization:

Let's initialize the class `Javanian`. Your input will be a dictionary where the keys are the locations and the values are the XP points a person earned by visiting them.

Input	Meaning
`jim = Javanian({'Javalake': 20, 'Interface Canyon': 10})`	Object `jim` is a Javanian who visited Javalake and Interface Canyon earning 20 and 10 XP points respectively. The total number of places `jim` visited is 2 and the

	total points `jim` earned is 30 XP.

Question 2.2 String representation:

There are two kinds of string representation (take a look at the example to see which string representation function to use)

1. The first string representation is a **sorted** list of locations. The list should be sorted by the XP points you earned at every location in ascending order.
2. The second string representation is a string summarizing the places you've been to and the XP points you earned in the following format:

   I visited {total number of places you've been to} places and have a total of {total XP points you have}XP points.

```
>>> jim = Javanian({'Javalake': 20, 'Interface Canyon': 10, 'this.valley':
30, 'BST': 40})
>>> jim
'Interface Canyon, Javalake, this.valley, BST'
>>> print(jim)
I visited 4 places and have a total of 100 XP points.
```

Question 2.3 Greater than Symbol:

Since work life balance is heavily emphasized people often want to check if the XP points of a person are above a certain threshold. This is why you want to implement a greater than symbol. Note that you should be able to compare the object to an integer (a whole number) and to another object.

```
'''
>>> kate = Javanian({'LinkedLand': 20, 'QueueChannel': 10, 'Stack Sand
Dunes': 50})
>>> jim = Javanian({'Javalake': 20, 'Interface Canyon': 10,
'this.valley': 30, 'BST': 40})
>>> gabi = Javanian({'LinkedLand': 30, 'Disjoint Forest': 40})'''
>>> gabi > 100
False
>>> jim > kate
True
>>> kate > gabi
```

```
True
'''
```

Question 2.4 Length Special Method:

To make sure that no one is overworking themselves and slacking off on resting you want to know how many places a person has been to. This is why you decide to implement a length special method that would return the total number of places a person has visited.

Length Special Method Documentation

```
'''
>>> kate = Javanian({'LinkedLand': 20, 'QueueChannel': 10, 'Stack Sand
Dunes': 50})
>>> jim = Javanian({'Javalake': 20, 'Interface Canyon': 10,
'this.valley': 30, 'BST': 40})
>>> gabi = Javanian({'LinkedLand': 30, 'Disjoint Forest': 40})'''
>>> len(gabi)
2
>>> len(jim)
4
>>> len(kate)
3
'''
```

Submission

By the end of this lab, you should have submitted the lab via Gradescope. You may submit more than once before the deadline; only the final submission will be graded.