Practical Exam April 2023

Programming Methodology (National University of Singapore)

CS1010S Programming Methodology
National University of Singapore

Practical Examination

15 Apr 2023

Time allowed: 1 hour 30 minutes

Instructions (please read carefully):

1. This is **an open-book exam**. You are allowed to bring in any course or reference materials in printed form. No electronic media or storage devices are allowed.
2. This practical exam consists of **three** questions (one question with two sub-questions). The time allowed for solving this test is **1 hour 30 minutes**.
3. The maximum score of this test is **20 marks**. Note that the number of marks awarded for each question **IS NOT** correlated with the difficulty of the question.
4. You are advised to attempt all questions. Even if you cannot solve a question correctly, you are likely to get some partial credit for a credible attempt.
5. While you are also provided with the template `practical-template.py` to work with, your answers should be submitted on Coursemology. Note that you can **only run the test cases on Coursemology for a limited number of tries** because they are only for checking that your code is submitted correctly. You are expected to test your own code for correctness using IDLE and not depend only on the provided test cases. Do ensure that you submit your answers correctly by running the test cases at least once.
6. In case there are problems with Coursemology.org and we are not able to upload the answers to Coursemology.org, you will be required to name your file `<mat no>.py` where `<mat no>` is your matriculation number and leave the file on the Desktop. If your file is not named correctly, we will choose any Python file found at random.
7. Please note that it shall be your responsibility to ensure that your solution is submitted correctly to Coursemology and correctly left in the desktop folder at the end of the examination. Failure to do so will render you liable to receiving a grade of **ZERO** for the Practical Exam, or the parts that are not uploaded correctly.
8. Please note that while sample executions are given, it is **not sufficient to write programs that simply satisfy the given examples**. Your programs will be tested on other inputs and they should exhibit the required behaviours as specified by the problems to get full credit. There is no need for you to submit test cases.
9. While you may use any built-in function provided by Python, you may not import functions from any packages, unless otherwise stated and allowed by the question.

GOOD LUCK!

Question 1 : Flatten Dictionary [5 marks]

Handling a nested dictionary can be quite cumbersome because it is difficult to access the individual values that are deeply nested within the dictionary. We need a way to flatten the dictionary so that key-value could be easily stored and retrieved.

Implement the function `flatten_dictionary` that takes in a dictionary *dict*, and separator value *sep* and returns a flattened dictionary if *dict* is nested. In case of duplicate keys generated after flattening, *consider the first value assigned to the key*, see the last sample in the example. (Note: this is unlike the python dictionary). [5 marks]

Sample execution:

```
>>> nested_dict={'a': 1, 'b': {'c': 2, 'd': {'e': 3}}}
>>> flatten_dictionary(nested_dict, '_')
{'a': 1, 'b_c': 2, 'b_d_e': 3}

>>> nested_dict={'C': {'S': {'1': {'0': {'1': {'0':'S'}}}}}}
>>> flatten_dictionary(nested_dict, '')
{'CS1010': 'S'}

>>> nested_dict={2: 1, 4: {6: 2, 8: {10: 3}, 12:4}, 14:5}
>>> flatten_dictionary(nested_dict, '-')
{'2': 1, '4-6': 2, '4-8-10': 3, '4-12': 4, '14': 5}

>>> nested_dict={(1,2): { 3: ['a','b', 'c'], (4,5): {(6,7):
    ['d','e,','f','g'] }}}
>>> flatten_dictionary(nested_dict, '*')
{'(1, 2)*3': ['a', 'b', 'c'], '(1, 2)*(4, 5)*(6, 7)': ['d', 'e,',
    'f', 'g']}

>>> nested_dict={'b_c': 5, 'b': {'c': 3}}
>>> flatten_dictionary(nested_dict, '_')
{'b_c': 5}
```

The `round` function

In Question 2 and Question 3, you need to use an in-built `round` function in python. The `round` function in Python rounds a floating-point number to a specified number of decimal places or to the nearest integer if no decimal places are specified. For instance:

```
>>> round(2.345, 2)
2.35
>>> round(2.345, 1)
2.3
```

Question 2 : Graduate statistics [10 marks]

Important note: You are provided with a csv data-file `enrollment.csv` for this question for testing, but your code should work correctly for *any* data-file with the same format and *will* be tested with other data-files.

You are given a csv data-file containing the enrollment statistics of the in the universities in Singapore. The data comprises of enrollment across various courses from 2005 to 2021. It has the following structure. Every year contains two entries namely for male and female statistics.

Year	Sex	Course	Intake	Graduates

A. Implement the function `year_of_max_graduates` that finds the year with the highest number of graduated students in a specified course (assume that it always exists in the dataset). It returns the most recent year in the case of multiple years with the maximum number. *The function accounts for both male and female graduation numbers to compute the total number of students graduated in a certain year.* The function takes two inputs: a course and a filename. It returns a tuple containing the year and the number of students graduated in that year.

```
>>> year_of_max_graduates("Accountancy", "enrollment.csv")
(2019, 2248)
>>> year_of_max_graduates("Education", "enrollment.csv")
(2012, 1117)
>>> (year_of_max_graduates("Law", "enrollment.csv")
(2021, 631)
```

B. It is commonly acknowledged that certain academic programs, like Engineering, tend to have fewer female students. The goal is to identify the academic programs with the largest disproportionality ratio computed as the ratio between absolute difference between the gendered intake divide by the total intake $\frac{|female\_intake - male\_intake|}{female\_intake + male\_intake}$. 0 quantifies a balanced intake whereas the higher values quantify a skewed intake.

Implement the function `top_K_disproportionate_courses` that takes three inputs: an integer K, a year and the filename. It returns the list of courses that are in the top-K disproportionate ratios (rounded to two decimal places) for the specified year. The returned list consists of tuples where each tuple contains the name of the course and the disproportionality ratio for the course in the specified year (already rounded to two significant decimal places).

Sample execution:

```
>>> top_K_disproportionate_courses(3, 2011, "enrollment.csv")
[('Engineering Sciences', 0.55), ('Information Technology', 0.49),
   ('Law', 0.36)]

>>> top_K_disproportionate_courses(3, 2019, "enrollment.csv")
[('Engineering Sciences', 0.55), ('Information Technology', 0.48),
   ('Medicine', 0.38)]

>>> top_K_disproportionate_courses(4, 2021, "enrollment.csv")
[('Information Technology', 0.61), ('Engineering Sciences', 0.57),
   ('Law', 0.36), ('Medicine', 0.35)]
```

Question 3 : Shopping cart [5 marks]

For this question, we will model the three classes: `Product`, `DiscountedProduct` and `Cart` that support the following behaviour:

A `Product` is initialized with a name and its price. It supports `get_price` function that returns the cost of the product.

A `DiscountedProduct` is also a product with a discounted price. It is initialized with a name, its price and the percentage discount offered on the product. Discounted price is rounded to two decimal places.

A `Cart` is initialized with no inputs. `Cart` supports the following methods:

- `add_item` — takes in a `product`, and returns a string following these rules:
  - If the products is a valid product, it is added in the cart, and the string `'Adding <product name> to the cart.'` is returned.
  - Otherwise, no change is made to the cart and the string `'Invalid item.'` is returned.
- `checkout` — takes in no arguments, and returns the dictionary with product name as the key and the total cost of the product as the corresponding value.

  Please refer to the code trace for the illustration.

Provide an implementation for the classes `Product`, `DiscountedProduct` and `Cart`.

You should use OOP practices taught in the lectures to design the classes.

You are advised to incrementally solve this problem. Even if you cannot fulfil all the desired behaviours, you can still get partial credit for fulfilling the basic behaviours.

```
>>> milk = Product("milk", 5.65)
>>> bread = Product("bread", 3.25)
>>> eggs = Product("eggs", 2.85)
>>> cereal = DiscountedProduct("post", 6.6, 10)

>>> milk.get_price()
5.65

>> shopping_list = [milk, milk, bread]

>>> c = Cart()

>>> c.add_item(milk)
'Adding milk to the cart.'

>>> c.add_item(bread)
'Adding bread to the cart.'

>>> c.add_item(bread)
'Adding bread to the cart.'
```

```
>>> c.add_item(shopping_list)
'Invalid item.'

>>> c.add_item(eggs)
'Adding eggs to the cart.'

>>> c.add_item(cereal)
'Adding post to the cart.'

>>> c.add_item(c)
'Invalid item.'

>>> c.checkout()
{'milk': 5.65, 'bread': 6.5, 'eggs': 2.85, 'post': 5.94}
```

— E N D O F P A P E R —