Problem Set 0: Introduction to Python

The goal of this problem set is to test your ability to code simple python programs and understand python code. In addition, you should use this lab as a chance to get familiar with autograder which will automatically grade your code.

To run the autograder, type the following command in the terminal:

```
python autograder.py
```

If you wish to run a certain problem only (e.g. problem 1), type:

```
python autograder.py -q 1
```

where 1 is the number of the problem you wish to run.

Instructions

In the attached python files, you will find locations marked with:

```
#TODO: ADD YOUR CODE HERE
utils.NotImplemented()
```

Remove the `utils.NotImplemented()` call and write your solution to the problem. **DO NOT MODIFY ANY OTHER CODE**; The grading of the assignment will be automated and any code written outside the assigned locations will not be included during the grading process.

For this assignment, you should submit the following files only:

- `palindrome_check.py`
- `histogram.py`
- `gpa_calculator.py`
- `locator.py`

Introduction to Type hints

In the given code, we heavily use Type Hints. Although Python is a dynamically typed language, there are many reasons that we would want to define the type for each variable:

1. Type hints tell other programmers what you intend to put in each variable.
2. It helps the intellisense find reasonable completions and suggestions for you.

Type hints are defined as follows:

```
variable_name: type_hint
```

for example:

```
#  x will contain an int
x: int

# x contains a string
x: str = "hello"

# l contains a list of anything
l: List[Any] = [1, 2, "hello"]

# u contains a float or a None
u: Union[float, None] = 1.25
```

We can also write type hints for functions as follows:

```
# Function is_odd takes an int and returns a bool
def is_odd(x: int) -> bool:
    return (x%2) == 1
```

Some type hints must be imported from the package `typing`. Such as:

```
from typing import List, Any, Union, Tuple, Dict
```

In all of the assignments, we will use type hints whenever possible and we recommend that you use them for function and class definitions.

Problem 1: Palindromes

Inside `palindrome_check.py`, modify the function `palindrome_check` to return true if and only if the input is a palindrome.

A palindrome is a string that does not change if read from left to right or from right to left. For example, '*apple*' is not a palindrome since when read from right to left, it becomes '*eppla*' which is not the same as '*apple*'. While '*radar*' is a palindrome since it is still '*radar*' when read from right to left. Assume that empty strings are palindromes.

Problem 2: Peaks

Inside `peaks.py`, modify the function `peaks` to return the number of peaks in a given sequence.

A peaks is an elements whose value is greater than the two elements surrounding it. For example, in `[1,2,4,1]`, there is only one peak which is 4 (index: 2) since it is greater than the elements before and after it (which are 2 and 1, respectively). The first and last elements cannot be peaks, so any array with less than 3 elements will have no peaks.

Problem 3: Histograms

Inside `histogram.py`, modify the function `histogram` that given a list of values, returns a dictionary that contains the list elements alongside their frequency.

For example, if the values are `[3,5,3]` then the result should be `{3:2, 5:1}` since *3* appears twice while *5* appears once.

Problem 4: GPA Calculator

Inside `gpa_calculator.py`, modify the function `calculate_gpa` that given a student and a list of courses, returns their GPA in the given courses.

Each course has a corresponding number of hours that define its weight in the GPA. For example, if course A has 3 hours while course B has 2 hours only, then course A has more weight in the GPA.

The GPA is the `sum(hours of course * grade in course) / sum(hours of course)` over all courses attended by the student. The grades come in the form: `'A+'`, `'A'` and so on. But you can convert the grades to points (numbers) using a static method in the course class.

$$GPA = \frac{\sum_c hours(c) * points(c)}{\sum_c hours(c)}, c \in courses\ attended\ by\ student$$

To know how to use the `Student` and `Course` classes, see the file `college.py`.

Problem 5: Locator in a 2D Grid

Inside `locator.py`, modify the function `locate`. This function takes a 2D grid and an item and it should return a list of (x, y) coordinates that specify the locations that contain the given item.

To know how to use the `Grid` class, see the file `grid.py`

Delivery

IMPORTANT: You must fill the **`student_info.json`** file since it will be used to identify you as the owner of this work. The most important field is the **id** which will be used by the automatic grader to identify you.

You should submit a zip archive containing the following files:

1. `student_info.json`
2. `palindrome_check.py`
3. `peaks.py`
4. `histogram.py`
5. `gpa_calculator.py`
6. `locator.py`

The delivery deadline is `October 22nd 2022 23:59`. It should be delivered on **Google Classroom**. This is an individual assignment. The delivered code should be solely written by the student who delivered it. Any evidence of plagiarism will lead to receiving **zero** points.