SCIT, University of Wollongong

CSIT110/CSIT810

2021 Session 3

Assignment 4 (15%) due on Saturday 1st September 2021 at 00:00AM

Objectives

- Able to write clear code with comments and follow coding convention
- Able to use variables with meaningful names and correct data types
- Able to define functions and class objects
- Able to raise and handle exceptions

Marking criteria:

- Total mark is 15. Deduct 1 mark for each day late.
- More than 3 days late will result in a zero mark.
- Code must be able to run with no errors: 0 mark for the whole assignment if there is an error is thrown.
- Correct file format (.py extension): 0 mark for the whole assignment if file submission is not in correct format.
- Use submission template for file submission.

Question 1	correctness, completeness and consistency with the assignment specification	6 marks
Question 2	correctness, completeness and consistency with the assignment specification	3 marks
Question 3	correctness, completeness and consistency with the assignment specification	3 marks
Question 4	correctness, completeness and consistency with the assignment specification	3 marks
Overall	comments include name, student number, subject code; clear code and follow coding convention; use variables with meaningful names and correct data types	Deduct up to 1 mark

Submission Instruction: Assignment 1 submission is on Moodle. Put all your python code into a single python file (file extension .py) and submit it.

Question 1.

Given that an employee dictionary object is one with keys `name` and `sales`, an example of the employee dictionary object is as follows:

```
example_employee = {
      "name": "Archie Doh",
      "sales": {
            "product_1": 10,
            "product_2": 10,
            "product_3": 10,
      },
}
```

Question 1a.

Define a class `Employee` that satisfies the following specifications.

Class name	`Employee`
Instance attributes	1. `name: str` 2. `sales: Dict[str,int]`
Parameter	1. an employee `dictionary` object with the str "name" and "sales" as keys.
Detailed information	The class constructor should accept an employee **dictionary** object, as seen in the example at the beginning of the question, and instantiate the instance attributes accordingly

Question 1b

Write a class method for the `Employee` class that satisfies the following specifications.

Method name	`dict_to_class_obj`
Parameter	1. `a list: list[dict]`
Return value	1. `a list: list[Employee]`
Detailed description	The function should return a list of `Employee` class objects. Each `Employee` class object is constructed from each employee `dictionaries` in the `list` parameter.

Here is an example of a list argument containing instances of employee dictionaries

```
[{ "name": "Rajah Din",
   "sales": {"product_0": 3,
             "product_2": 5,
             "product_4": 4,
             },
 },{"name": "Jafar Min",
    "sales": {"product_1": 1,
              "product_2": 3,
              "product_5": 5,
              },
 }]
```

Question 1c

Write an instance method for the Employee class that satisfies the following specifications.

Method name	get_weighted_sales
Parameter	1. weights: Dict[str,float]
Return value	1. sales: float
Detailed description	A weighted result is the sum of the product of the values in weights dictionary and the values in the sales dictionary that have the same keys. You may assume that the keys in the weights dictionary is a subset of keys from the sales dictionary. This method should return a single value – the weighted sum of the sales based on the keys from the weights dictionary.

Here is an example of the input weights dictionary.

```
weights = {"product_1": 1.0, "product_5": 3.0}
```

Using the example above and that in question 4b, the weighted result for the employee named Jafar Min is 16.

product_1:   1.0 * 1 =   1

product_5:   3.0 * 5 =   15

weighted result: 16

Please note that not all key-value pairs present in the sales dictionary may be included in the weights dictionary. In the example above, product_2 was excluded. You can assume that the keys in the weights are all present in the sales dictionary.

Question 2a.

Define an Exception class that meets the following specifications.

Class name	ProductNotFoundError
Constructor parameters	1. name_product: str 2. name_employee: str Assign these values to instance attributes of the same names.

Question 2b.

Define a dunder method for the above class that meets the following specifications.

Method name	__str__
Method parameter	-
Return value for the dunder method	Returns 'Home insurance sales quantity cannot be found in Jafar Min's sales results' if the instance attributes name_employee is 'Jafar Min' and name_product is 'Home insurance'.

Question 2c.

1. Modify the get_weighted_results() in question 1c to raise a custom exception named AssessmentNotFoundError with the correct parameters.

Question 3a.

Create an exception class InvalidDepthError. Define a __str__ dunder method for this class to return a string "Invalid Depth".

Question 3b.

Define a class that meets the following specifications.

Class name	WaterBody
Class constructor parameter	1. int/float Assign this number to the instance attribute volume
Class attribute	The class has class attributes RHO = 997 and G = 9.81.

Question 3c.

Define a **class** method that meets the following specifications.

Method name	get_hydrostatic_pressure
Method parameter	1. float
Return value	1. float

Detailed information	Using the input float, the depth. calculate and return the hydrostatic pressure. Hydrostatic pressure a given depth = RHO*G*depth If the depth is less than 0, the static method should raise an InvalidDepthError. This should be defined in question 4a.

Question 3d.

Define an **instance** method that meets the following specifications.

Method name	get_water_mass
Method parameter	-
Return value	1. Float
Detailed information	This method should return the mass of the waterbody given that mass = RHO* volume.

Question 4a.

Look at the submission template. Understand what `example()` in the main scope is doing.

Question 4b.

Write a function that meets the following criteria.

<table>
<tr><td>Function name</td><td><code>myClass_get_int_unit_test</code></td></tr>
<tr><td>Parameter</td><td>1. A class reference</td></tr>
<tr><td>Return value</td><td>1. str or int</td></tr>
<tr><td>Detailed information</td><td>In the function, using a try and except blocks,

instantiate an instance of the input class.
Next, call the instance method <code>get_integer()</code>. This method does not take in any parameters.

The function should return the corresponding values in the table below.

<table>
<tr><td>Condition</td><td>Return value</td></tr>
<tr><td>AttributeError was raised.
An error raised when a method or variable of an instance which was referenced did not exist</td><td>‘A’</td></tr>
<tr><td>ValueError was raised.
This occurs when an argument that has the right type but an inappropriate value</td><td>‘V’</td></tr>
<tr><td>All other errors</td><td>‘O’</td></tr>
<tr><td>If no error was raised</td><td>Return the integer which the method returns</td></tr>
</table>
</td></tr>
</table>