SCIT, University of Wollongong

CSIT110

2023 Session 4

Assignment 3 (20%) due on November 12th 2023 at 23:59PM

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

Marking criteria:

- Total mark is 20. 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	4 marks
Question 2	correctness, completeness and consistency with the assignment specification	3 marks
Question 3	correctness, completeness and consistency with the assignment specification	5 marks
Question 4	correctness, completeness and consistency with the assignment specification	4 marks
Question 5	correctness, completeness and consistency with the assignment specification	4 marks

Submission Instruction: Use the submission template available on Moodle. Assignment 3 submission is on Moodle. Put all your python code into a single python file (file extension.py) and submit it.

Save the file in this format name_uowID_a3.py

Assignment questions: there are 5 assignment questions.

Write clear code with **comments** and follow **coding conventions**. Include **your name**, **student number** and **subject code** as str objects at the top of your code. Please also add this information to the variables as stated in the template Your code must work **exactly** like the provided examples given the input in the examples.

```
name = 'John Snow'
student_num = '1234567'  # UOW Student number
subject_code = 'CSIT110'  # CSIT110 or SP121
```

Submission Instruction: Assignment 3 submission is on Moodle. Put all your python code into a single python file (<name>_<std no>_a3.py) and submit it.

Question 0.

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

Question 1.

Write a function that meets the following criteria.

Function name	`myClass_get_int_unit_test`
Parameter	1. A class reference
Return value	1. str or int
Detailed information	In the function, using a try and except blocks, instantiate an instance of the input class. Next, call the instance method `get_integer()`. The function should return the corresponding values in the table below.

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

Question 2.

A merchant has a collection of goods. Help him write the following function.

Function name	`compute_unit_prices`
Function parameter	1. `Dict[str, List[float,int]]` A dictionary with `str` objects as keys and a list of two numbers as value. The keys correspond to the names of the goods. Each list has two numbers, the first, of type `float`, is the bulk cost of goods, the second number, of type `int`, is the bulk quantity of the goods. 2. `List[str]` A list of `str` objects. Compute the unit prices of the goods in this list.
Return value	1. `Dict[str,float]` The dictionary shall contain the unit prices with the name of goods as keys and the unit prices as values. The unit price is defined by the bulk price divided by the bulk quantity.
Detailed information	Using try and except blocks, should the names in the second input cannot be found in the first, the good's unit price should be a None object, should the unit price cannot be obtained due to a ZeroDivisionError, the unit price should be -1. Using if else blocks will result in 0 marks for this question. If any other errors are raised, the function should return an empty dictionary

An example of the first input

```
{
    "vinegar": [120.0, 100],
    "ketchup": [950, 1000],
    "apples": [850,1100],
    "oranges": [1050, 0]
}
```

An example of the second input

```
["ketchup","oranges","pear"]
```

Return value of with the above examples as input

```
{
    "ketchup": 0.95,
    "oranges":-1,
    "pear": None
}
```

Question 3a

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

Class name	`OutOfStockError`
Inheritance	This class inherits the `Exception` class
Attributes	An instance variable 1. a `str` object An instance method `1. __str__()`
Parameter of the constructor	1. a `str` object
Detailed information	Assign the parameter to the instance variable, you may use variable names of your choice. The `__str__` dunder method should return the following text 'The following item - ____ is out of stock!' (without quotes) Replace the blank with the str object in the instance variable.

Example Code	`try:`
	`    raise OutOfStockError("Eggs")`
	`except OutOfStockError as e:`
	`    print(e)`

Console output for the example code above	The following item Eggs is out of stock!

Question 3b.

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

Class Name	`Inventory`
Constructor Parameters	1. -
Class Attributes	This class has two class variables 1. A string. `hotline` This attribute is a string object with the value "1800-1333-5432" 2. A dictionary, `items` Assign an empty dictionary to this attribute.

Example Code	`print(Inventory.hotline)`
	`print(Inventory.items)`

Example console output	1800-1333-5432 {}

Question 3c.

Define a **class method** for the class `Inventory` that satisfies the following specifications.

Class Name	`set_items_from_list`
Method type	`Class method`
Parameters	1. A `list`. Each element of this list is a list of length 3. Each list contains a string, a float and an integer in this order. E.g. `[["Eggs", 2.98, 12],["Milk", 4.65, 3]]`
Return value	-
Detailed information	Go through the parameter list and, using each nested list, add key-value pairs to the class variable, items. The key is the string object in the nested list while the value mapped to the key is a Price object created using the float object in the list. Please see Price Object definition in the submission template.

Example Code	`print(Inventory.items)`
	`Inventory.set_items_from_list(`
	`    [["Eggs", 2.98, 12],["Milk", 4.65, 3]])`
	`print(Inventory.items)`

Example console output	{}
	{'Eggs': {'price': $2.98, 'stock': 12}, 'Milk': {'price': $4.65, 'stock': 3}}

Question 3d.

Define a **class method** for the class `Inventory` that satisfies the following specifications.

Class Name	`order`
Method type	`Class method`
Parameters	-
Return value	1. A `dictionary`.
Exception	This method should raise an exception when a user input satisfies the condition mentioned below.
Detailed information	Create a dictionary, Y Iterate through the keys the class variable `Inventory.items`, and the prompts shown in the example, get the number of items which the user will like to order. If the user input is 0, proceed to the next iteration. Otherwise, if the stock value is less than the user input, raise the OutOfStockError defined in question 1c using the current key object. If no error was raised, using the key as key and the user input converted to integer type as values, add the key-value pairs into a dictionary, Y. Return the dictionary Y. The method only needs to handle user input of values 0-20.

Example Code	
	```
	Inventory.set_items_from_list(
	    [
	        ["Eggs", 2.98, 12],
	        ["Milk", 4.65, 8],
	        ["Tea", 1.50, 6]]
	)

	print(Inventory.order())
	Inventory.order()
	```

Example console output	
	```
	How many Eggs would you like to order? 1
	How many Milk would you like to order? 7
	How many Tea would you like to order? 0
	{'Eggs': 1, 'Milk': 7}
	How many Eggs would you like to order? 13
	Traceback (most recent call last):
	…
	__main__.OutOfStockError: The following item Eggs is out of stock!
	```

Question 3e.

Define a function that satisfies the following specifications.

Function Name	`collate_orders`
Parameters	1. One integer, N
Return Value	One dictionary with the keys `"invalid","valid_items","oos"`
Detailed information	In the function, create a dictionary, Z, with the keys `"invalid","valid_items","oos"`. Next, use a try block to call the Inventory's class method, order(), N times. Map the number of times an OutOfStockError was raised to the key `"oos"` in the dictionary Z and the number of times other errors was raised to the key `"invalid"` in the dictionary Z. If no errors was raised, add the total number of items ordered in the order (sum of all the values in the returned dictionary) to the key `"valid_items".`

Example Code	
	```
	Inventory.set_items_from_list(
	        [
	            ["Eggs", 2.98, 12],
	            ["Milk", 4.65, 8],
	            ["Teas", 1.50, 6]]
	    )
	print(collate_orders(4))
	```

Example console output	How many Eggs would you like to order?**1** How many Milk would you like to order?**4** How many Teas would you like to order?**7** How many Eggs would you like to order?**1** How many Milk would you like to order?**1** How many Teas would you like to order?**1** How many Eggs would you like to order?**1** How many Milk would you like to order?**b** How many Eggs would you like to order?**1** How many Milk would you like to order?**3** How many Tea would you like to order?**2** {'invalid': 1, 'valid_items': 9, 'oos': 1}

Question 4a.

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

Question 4b.

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 4c.

Define a **class method** for the `WaterBody` class that meets the following specifications.

Method name	`get_hydrostatic_pressure`
Method parameter	1. `float`
Method type	Class method
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 4d.

Define a **instance** method for the `WaterBody` class that meets the following specifications.

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

Question 4e.

Define three **static** methods for the `WaterBody` class that meet the following specifications.

Method names	`is_large`
	`is_medium`
	`is_small`

Method parameter	1. `float` the volume of the water body in km^3
Return value	1. `bool`
Detailed information	Return a Boolean according to the criteria – is_small returns True if volume is less than 50 km^3. is_medium returns True if the volume is between and inclusive of 50 km^3 to 100km^3 is_large returns True if the volume is greater than 100km^3

Question 4f.

Define a **class** method for the `WaterBody` class that meets the following specifications.

Method names	`spawn`
Method parameter	`-`
Return value	1. a `WaterBody` object
Detailed information	Return an instance of `WaterBody` with a volume that is randomly generated from the random module. Note that volume must be a positive value. (>0)

You can use the following lines of code to verify part of your code.

```
pool = WaterBody(10)
print(pool.get_hydrostatic_pressure(1)) # prints 9780.57
print(pool.get_water_mass())  # prints 9970

try:
    pool.get_hydrostatic_pressure(-1)
except Exception as e:
   print(e)  # prints Invalid Depth
```

Question 5

Create a class `SingaporeNumbers`.

Part 1

A typical vehicle registration number comes in the format **xxx #### y**:

- **x** – prefixes
- #### – Numerical series (from 1 to 9999, without leading zeroes)
- **y** – Checksum
    - The checksum letter is calculated by converting the letters into numbers, *i.e.*, where A=1 and Z=26, potentially giving seven individual numbers from each registration plate. However, only two letters of the prefix are used in the checksum. For a three-letter prefix, only the last two letters are used; for a two-letter prefix, both letters are used; for a single letter prefix, the single letter corresponds to the second position, with the first position as 0. For numerals less than four digits, additional zeroes are added in front as placeholders, for example "1" is "0001". SBS 3229 would therefore give 2, 19, 3, 2, 2 and 9 (note that "S" is discarded); E 12 would give 0, 5, 0, 0, 1 and 2. SS 108 would be given as 19, 19, 0, 1, 0, 8.
    - Each individual number is then multiplied by 6 fixed numbers (9, 4, 5, 4, 3, 2). These are added up, then divided by 19. The remainder corresponds to one of the 19 letters used (A, Z, Y, X, U, T, S, R, P, M, L, K, J, H, G, E, D, C, B), with "A" corresponding to a remainder of 0, "Z" corresponding to 1, "Y" corresponding to 2 and so on. In the case of SBS 3229, the final letter should be a P; for E 23, the final letter should be a H. SS 11 back letter should be a T. The letters F, I, N, O, Q, V and W are not used as checksum letters.

Question 5a

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

Method name	`car_plate_checksum`
Parameter	1. str This str contains the prefixes and numerical series mentioned in the description above.
Return value	1. str
Detailed description	Compute and return the checksum from the string parameter. The computation logic is described in the section titled 'Part 1'. The checksum is one character in length. The return value is case insensitive. You should use the try and except blocks to find out is a character in a string is an integer or not. The input string may contain 1-3 letters for prefixes while there can be 1 to 4 digits for the numerical series that follows.

Part 2

The checksum letter of a magic 7 digits number is calculated as such:

d $= [(i_1\ i_2\ i_3\ i_4\ i_5\ i_6\ i_7) \bullet (2\ 7\ 6\ 5\ 4\ 3\ 2)] \bmod 11$

$= (2i_1 + 7i_2 + 6i_3 + 5i_4 + 4i_5 + 3i_6 + 2i_7) \bmod 11$

Where i_x is the 1st to last of the 7 digits of the numbers and (2,7,6,5,4,3,2) are the weights.

Write a static method `magic_num_checksum` that Return the letter which corresponds to the number d as shown in the look-up table below

d	10	9	8	7	6	5	4	3	2	1	0
Check digit	A	B	C	D	E	F	G	H	I	Z	J

Question 5b

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

Method name	`magic_num_checksum`
Parameter	1. str This str contains the prefixes and numerical series mentioned in the description above.
Return value	1. str
Detailed description	From the given string of 7 numbers, computer the number d as described in the section 'Part 2'. Return the letter which corresponds to the number d as shown in the look-up table below *d*: 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0 Check digit: A, B, C, D, E, F, G, H, I, Z, J