SCIT, University of Wollongong

CSIT110/CSIT810

Autumn Session 2020

Assignment 3 (20%) due on Saturday 21 November 2020 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 exceptions and handle different errors

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 mark
Question 2	correctness, completeness and consistency with the assignment specification	4 marks
Question 3	correctness, completeness and consistency with the assignment specification	4 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
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 3 submission is on Moodle. Put all your python code into a single python file (<name>_<std no>_a3.py) and submit it.

Assignment questions: there are 5 assignment questions.

Write clear code with **comments** and follow **coding conventions**. Comments should include **your name**, **student number** and **subject code** on 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.

```
"""Assignment 3

Name: John Snow
Student number: 1234567
Subject code: CSIT110
"""

name = 'John Snow'
student_num = '1234567'  # Student number
subject_code = 'CSIT110'  # CSIT110 or SP420

    ...
```

Question 0.

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

Question 1.

Write a function named `myClass_get_int_unit_test`. This function should take in a class reference. In the function, using a try and except blocks, instantiate an instance of the input class and 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 a function named compute_unit_prices. This function takes in two input. The first input is a dictionary with name of the goods as keys and a list of bulk prices and bulk quantity as value. (See example) The second input is a list of goods' names. The function should return a dictionary. 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.

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 3

Copy the `Student` class in assignment 2. Modify the `get_weighted_results()` to raise a custom exception named `AssessmentNotFoundError`. The exception should take in the name of the assessment and the name of the student. Write a `__str__` dunder method for the custom error that returns 'Quiz results cannot be found in Tom's results' if the name of the student is 'Tom' and the assessment name is 'Quiz' as passed from the argument.

Question 4

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

Separately, define a class `WaterBody` which takes in a number when its constructor is called. Assign this number to the instance attribute `volume`. The class has class attributes `RHO = 997` and `G = 9.81`. Create a static method `get_hydrostatic_pressure`. This method takes in a float. 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`.

Define an instance method, `get_water_mass`, for the `WaterBody` class. This method should return the mass of the `waterbody` given that max = `RHO* volume`.

Define a static methods is_large, is_medium, is_small. All these methods take in a single number – the volume of the water body in km^3. The methods should return a Boolean according to the criteria – small if volume is less than 50km^3, medium if the volume is between and inclusive of 50km^3 to 100km^3, large if greater than 100km^3.

Define a class method spawn() that returns an instance of `WaterBody` with a volume that is randomly generated from the random module note that volume must be a positive value.

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.

Write a static method `car_plate_checksum` that returns the checksum from a given string. 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 returns 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