SCIT, University of Wollongong

CSIT110/CSIT810

Autumn Session 2020

Assignment 2 (10%) due on Saturday 9 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 get user input
- Able to display output

Marking criteria:

- Total mark is 10. 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.
-

Question 1	correctness, completeness and consistency with the assignment specification	1 mark
Question 2	correctness, completeness and consistency with the assignment specification	2 marks
Question 3	correctness, completeness and consistency with the assignment specification	3 marks
Question 4	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 1 submission is on Moodle. Put all your python code into a single python file (file extension .py) and submit it.

Assignment questions: there are 4 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.

```
"""Assignment 2

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

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

    ...
```

Question 1. Write a function named get_subscription() with the help of a dictionary to display the six TV channels available for subscriptions and take subscription orders. Your program should work exactly as the following examples. The text in **bold** indicates user input. The two columns are 41 characters and 6 characters in width. The bullets that before the list of selection is a space, a hyphen and a space.

Example 1: The user subscribes to 2 channels.

```
Channels available for subscription (price/mth)
Sports Group Pack                          $21.40
Documentaries Pack                         $15.32
FOX Movies Pack                            $17.12
HBO Pack                                   $13.98
Cinema World                                $9.56
Celestial Movies                            $8.56

Subscribe to Sports Group Pack? (Y/N): Y
Subscribe to Documentaries Pack? (Y/N): N
Subscribe to FOX Movies Pack? (Y/N): N
Subscribe to HBO Pack? (Y/N): N
Subscribe to Cinema World? (Y/N): Y
Subscribe to Celestial Movies? (Y/N): N

Your selection:
 - Sports Group Pack ($21.40)
 - Cinema World ($9.56)

Total cost $30.96
```

Example 2: The user subscribes to 0 channel.

```
Channels available for subscription (price/mth)
Sports Group Pack                          $21.40
Documentaries Pack                         $15.32
FOX Movies Pack                            $17.12
HBO Pack                                   $13.98
Cinema World                                $9.56
Celestial Movies                            $8.56

Subscribe to Sports Group Pack? (Y/N): N
Subscribe to Documentaries Pack? (Y/N): N
Subscribe to FOX Movies Pack? (Y/N): N
Subscribe to HBO Pack? (Y/N): N
```

```
Subscribe to Cinema World? (Y/N): N
Subscribe to Celestial Movies? (Y/N): N

Your selection:
 - None

Total cost $0.00
```

Important requirement:

Your program must use string addition to produce the exact output as illustrated in the above example. You may assume the input are of letters n and y .

Question 2. Write a function named generate_qns_from_list(). The function should take in a list. Each element of this list is a list of integers. This function should convert the list into a list of dictionaries. Each dictionary will have two keys – `"qns"` and `"ans"`. The value for the `"qns"` key will be a string of the integers taken from a list in the input list. And the integers are separated with the characters `" x "`. The value for the `"qns"` key will be the product of all the numbers in the list.

You should skip the lists that contain less than 2 integers.

Example:

```
input_list = [[1,3,3], [2,5,-1],[3,2],[4,5,3],[0,23],[1,2,3,4], [1]]

def generate_qns_from_list(arg1):
  ...
```

The function should return the following list

```
[{'qns': '1 x 3 x 3', 'ans': 9},
 {'qns': '2 x 5 x -1', 'ans': -10},
 {'qns': '3 x 2', 'ans': 6},
 {'qns': '4 x 5 x 3', 'ans': 60},
 {'qns': '0 x 23', 'ans': 0},
 {'qns': '1 x 2 x 3 x 4', 'ans': 24}]
```

Question 3.

Create a `ShoppingCart` class. The class has a class attribute `server_url` and instance attributes `account_id` and `cart`. The server_url is the string `"128.123.123.0"`. The `account_id` is a string provided through the constructor. `cart` is an empty dictionary. The keys in the dictionary are the product names while their values mark the quantity of the products in the cart.

In the class there should be an instance method named `add_item_to_cart`. The method takes in a unique string, which denotes the name of a product, and a number that states the quantity of item to be added. The function should check if the item exists in the keys of the `cart` attribute and update the quantity accordingly.

Create another instance method `remove_item_from_cart` that takes in a product name and a quantity. It should update the cart attribute with the correct quantity. If the updated quantity is 0, the key value pair should be removed from the cart. You can assume that the product is available for removal and the quantity to be removed is less than or equal to the quantity in the cart.

Create a class method `get_url` that returns the class attribute `server_url`.

Create an instance method `empty_cart` to empty cart.

Finally create an instance method `count_items` that returns the total checkout price based on the existing items in the cart. If there is nothing in the cart, the total should be 0.

Your code must work exactly like the following example.

```
newCart = ShoppingCart('1234567')
newCart.add_item_to_cart('fruit juice', 2)
newCart.add_item_to_cart('tissue box', 4)
newCart.add_item_to_cart('ice cream', 3)
# newCart.cart is now {'fruit juice': 2, 'tissue box': 4, 'ice cream': 3}
newCart.remove_item_from_cart('tissue box',1)
newCart.remove_item_from_cart('fruit juice',2)
# newCart.cart is now {'tissue box': 3, 'ice cream': 3}
newCart.count_items()  # returns 6
print(newCart.get_url())  # prints 128.123.123.0
newCart.empty_cart()
newCart.count_items()  # returns 0
```

Question 4.

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

```
{
      "name": "Fus Ro Dah",
      "results": {
            "assignment_1": 10,
            "assignment_2": 10,
            "examination_1": 10,
      },
}
```

Part I

Define a class `Student` that contains the instance attributes `name` and `results`.
The class constructor should accept a `student` dictionary object, as seen in the example above, and instantiate the instance attributes accordingly.

Write a function `dict_to_class_obj` with one input argument – a list of `student` dictionary objects. The function should return a list of `student` class instance objects, constructed from the list argument.
An example of the list argument is as follows:

```
[
      {
            "name": "Fus Ro Dah",
            "results": {
                  "assignment_1": 10,
                  "assignment_2": 10,
                  "examination_1": 10,
            },
      },
      {
            "name": "Foo Barry",
            "results": {
                  "assignment_1": 1,
                  "assignment_2": 2,
                  "examination_1": 3,
            },
      },
]
```

Part II

In the `Student` class, define a class instance method `get_weighted_result` with one parameter – a `weights` dictionary. The `weights` dictionary contains a subset of keys from the `results` dictionary. The values are numerical. An example of the input `weights` dictionary is as follows:

```
weights  = {"assignment_1": 1.0, "examination_1": 9.0}
```

The `get_weighted_result` method should return a single value: the weighted sum of the results.

Using the examples above, the weighted result for the student named Foo Barry is 28:
assignment_1: 1.0 * 1 = 1 examination_1: 9.0 * 3 = 27 weighted result: 28

Please note that not all key-value pairs present in the `results` dictionary may be included in the weights dictionary. In the example above, `assignment_2` was excluded. You can assume that the keys in the weights are all present in the results dictionary.