SCIT, University of Wollongong

CSIT110/CSIT810

Autumn Session 2020

Supplementary Examination (60%) due on Wednesday 23rd December 2020 at 01:00PM

Marking criteria:

- Total marks is 60.
- No errors should be raised unless otherwise specified.
- Correct file format (.py extension):5 marks penalty if file submission is not in correct format.
- Names of variables, functions and classes that do not meet specification will result in 0 marks for the question. Spelling errors in attributes, variables, functions, class names which trigger errors in testing the question will be penalised.
- Use submission template for file submission.

Question 1	Correctness, completeness and consistency with the exam specification	30 marks
Question 2	Correctness, completeness and consistency with the exam specification	15 marks
Question 3	Correctness, completeness and consistency with the exam specification	15 marks
Overall	Comments include name, student number, subject code; clear code and follow coding convention; use variables with meaningful names and correct data types. No calling of functions outside of main() block. Use the template given.	Deduct up to 1 mark for each issue Deduct up to 5 marks

Submission Instruction: Submission is on Moodle. Put all your python code into a single python file (<name>_<uow std no>_supp_exam.py) and submit it.
Exam questions: there are 3 examination 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.
You may assume that the parameters used for testing are of the right data type specified in each question, unless stated otherwise.

```
"""Examination

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

name = "John Snow"
student_uow_id = "1234567"  # UOW student number
course_code = "CSIT110"  # CSIT110 or SP420
```

Question 1: There are 6 parts to this question. Write the following **functions.**

Q1a.

Function name	`question_1a`
Parameter(s)/ argument(s)	This function does not take in any parameter
Return value	1. An integer
Detailed description	In the function, with the str ‘Enter integer: ‘, get user input until an empty str is obtained. You can assume that, unless it is an empty string, the user input will be a valid integer. Finally, return the sum of all the integers obtained.
Example usage	`print(question_1a())`
Example usage output Those in bold are user input	`Enter integer:` **`3`** `Enter integer:` **`4`** `Enter integer:` **`8`** `Enter integer:` `15`

Q1b.

Function name	`question_1b`
Parameter(s)/ argument(s)	In this order, 1. An integer, start 2. An integer, end 3. An integer, step

Return value	1. A dictionary with the following keys
	`"qns": str for value`
	`"ans": integer for value`

Detailed description	Value for the dictionary key, `"qns"` The value for the dictionary key, `"qns"`, is a string of numbers separated by `" + "`. The first number is given by the first parameter, `start`. Each of the following numbers is greater than the previous number by the amount provided by the third parameter, `step`. The last number to this series is less than and **NOT** inclusive of the number given by the second parameter. Value for the dictionary key, `"ans"` The value for the dictionary key, `"ans"` is an integer whose value is the sum of all the numbers in the str described previously in the section, *Value for the dictionary key, "qns"*.
Example usage	`print(question_1b(2,20,3))`
Example usage output Those in bold are user input	`{"qns": "2 + 5 + 8 + 11 + 14 + 17", "ans": 57}`

Q1c.

Function name	`question_1c`
Parameter(s)/ argument(s)	In this order, 1. A integer, `x` 2. A integer, `y`
Return value	1. A boolean
Detailed description	If the parameter x is divisible(without a remainder) by the parameter y, return the boolean, `True`, otherwise `False`.
Example usage	`print(question_1c(5,2))`
Example usage output Those in bold are user input	`False`

Q1d

Function name	`question_1d`
Parameter(s)/ argument(s)	1. A list of str
Return value	1. An int
Detailed description	This function should return the total length of all the str in the list. You may assume that there are no special characters like \n and \t in the list of str
Example usage	`print(question_1d(["hi", "good day",'!',""]))`
Example usage output Those in bold are user input	`11`

Q1e

Function name	`question_1e`
Parameter(s)/ argument(s)	1. A str
Return value	This function should not return any value.
Detailed description	Using the str parameter as the error message, raise the built-in exception `NotImplementedError`.
Example usage	`question_1e("Here is an example.")`

Example usage output Those in bold are user input	`Traceback (most recent call last):`
	`   ...`
	`NotImplementedError: Here is an example.`

Q1f

Function name	`question_1f`
Parameter(s)/ argument(s)	1. `A dictionary with str as keys`
Return value	1. A str
Detailed description	Using the keys in the dictionary, format them into a string such that each key is separated by the character ! Make sure that the characters between ! are all centered aligned and have a minimum width of 10, exclusive of !. The string also starts and ends with a ! character. Return the formatted string.
Example usage	`print(question_1f({"ola": 123,"splendid":456, "adios":7890}))`
Example usage output Those in bold are user input	`!   ola    ! splendid !  adios   !`

Q2a Write a **class** that fulfils the following specifications.

Class name	`Book`
Parameter(s)/ argument(s) for the constructor `__init__`	In this order, 1. title - a `str` 2. author - a `str`
Detailed description	1. Assign the parameters to instance attributes of the same names, i.e. `title` and `author`

Example usage	`lotr = Book("The Lord of the Rings","J. R. R. Tolkien")`
	`print(lotr.title)`
	`print(lotr.author)`

Example usage output	`The Lord of the Rings`
	`J. R. R. Tolkien`

Q2b.
Write a **class method** for the `Book` class that fulfils the following specifications.

Method name	`from_dict`
Parameter(s)/ argument(s)	A dictionary with the following key value pairs 1. key - `"title"`, value - a `str` 2. key - `"author"`, value - a `float`
Return value	An instance of the `Book` class
Detailed description	Return an instance of the `Book` class using the values of the keys `"title"` and `"author"` in the dictionary.

Example usage	`new_book = Book.from_dict({"title":`
	`"To Kill a Mockingbird","author":"Harper Lee"})`
	`print(new_book.title)`
	`print(new_book.author)`

Example usage output	`To Kill a Mockingbird`
	`Harper Lee`

Q2c.
Write a dunder instance method `__str__` for the `Book` class that fulfils the following specifications.

Method name	`__str__`
Parameter(s)/ argument(s)	This method does not take in any parameter
Return value	A `str`
Detailed description	Replace x and y in the string `"x by y"` with the title of the book and the author of the book respectively. Return the formatted text.

Example usage	`new_book = Book.from_dict({"title":`
	`      "To Kill a Mockingbird","author":"Harper Lee"})`
	`print(new_book)`

Example usage output	`To Kill a Mockingbird by Harper Lee`

Q3:

Write the following **function**:

Function name	`get_team_statistics`
Parameter(s)/ argument(s)	1. List of Employee instances. `i.e. type list[Employee]` The definition of the Employee class is shown in your submission template
Return value	Dictionary with str for keys and integer for values `i.e. type dict[str, int]` There should be 3 key-value pairs: 3. Key: `"target_met_count"` Value: Number of employees that met sales target (total sales greater than or equal to 100) 4. Key: `"target_failed_count"` Value: Number of employees that did not met sales target (total sales is less than 100) 5. Key: `"invalid_count"`, Value: Number of Employees with no sales data (`sales == None, error raised.`)
Error handling	The function must handle `NoDataError.` The definition of the `Error` is provided in the template. The function must not raise any exceptions.

Detailed description	Notes: • You may assume the function will be called with a valid argument. • The function should not write to stdout i.e. do not use `print()`. • The `Employee` class definition is given below and in the submission template. • You may assume the grades are non-negative. • The employee's total grade is the sum of the employee's sales. • **You HAVE to use the `Employee` instance method** `get_sales()` to access the `Employee` instance attribute `self.__sales`.
	
	 `class NoDataError(Exception):`
	 `      pass`
	
	 `class Employee():`

```
    def __init__(self, name= "", sales=None):
        self.__name = name
        self.__sales = sales
    def get_name(self):
        return self.__name

    def get_sales(self):
        if self.__sales == None:
            raise NoDataError()
        return self.__sales
```

Example usage

```
Employees = [
    Employee("EmployeeA"),              # Invalid
    Employee("EmployeeB", [10, 2, 53]),    # Fail
    Employee("EmployeeC", [11, 64, 5]),    # Fail
    Employee("EmployeeD", [20, 40, 40]),   # Met
    Employee("EmployeeE", [30, 40, 50]),   # Met
    Employee("EmployeeF", [170, 15, 25]),  # Met
]
print(get_team_statistics(Employees))
```

Example usage output

```
{"target_met_count": 3, "target_failed_count": 2,
"invalid_count": 1}
```

-------------------End of Paper-------------------