Tutorial 4 (Week 6)

- Able to handle exceptions

Consider the following scenario:
A subject has 3 assessments:
- An assignment whose mark is an integer between 0 and 20;
- A project whose mark is an integer between 0 and 30;
- A final exam whose mark is an integer between 0 and 50.

Write a program to ask the user to enter assessment marks and then display the total mark. The program should stop when there is an error and display that error in detail.

Example 1: All inputs are good

```
Enter assignment mark (0-20): 15
Enter project mark (0-30): 25
Enter final exam mark (0-50): 30
Total mark: 70
```

Example 2: Assignment mark is not an integer

```
Enter assignment mark (0-20): abc
Error: assignment mark is invalid
```

Example 3: Assignment mark is not out of range

```
Enter assignment mark (0-20): 40

Error: assignment mark must be between 0 and 20
```

Example 4: Project mark is not out of range

```
Enter assignment mark (0-20): 15
Enter project mark (0-30): -5

Error: project mark must be between 0 and 30
```

Example 5: Project mark is not an integer

```
Enter assignment mark (0-20): 15
Enter project mark (0-30): xyz

Error: project mark is invalid
```

Example 6: Exam mark is not an integer

```
Enter assignment mark (0-20): 15
Enter project mark (0-30): 20
Enter final exam mark (0-50): frog

Error: final exam mark is invalid
```

Question 1: Write a **function** based on the following specification:

Function name:	`get_assessment_mark`
Input arguments:	3 input arguments: ● `assessment_name:` a string, either assignment, or project, or final exam ● `mark_min:` an integer, the mark minimum ● `mark_max:` an integer, the mark maximum
Return values:	1 return value: the function asks the user to enter a mark and return this mark.
Exception:	Raise `ValueError` if one of the following error occurs: ● Mark is not an integer ● Mark is not between the range

Question 2. Using the function in Question 1, write a program that works **exactly** like the above examples. That is, asking the user to enter 3 assessment marks and display the total mark.

Question 3 Using the `Student` class you defined in Tutorial question13, define a class method `from_list(arg)` to return a list of student instances from a list. An example of the input list is as such [[‘John Snow’, ‘135226’], [‘Peter Parker’, ‘197439’],…]

Next define a class method `from_dict(arg)` that returns an instance of `Student` with the given input. An example of the input is as such {"name": 'John Snow', "student_number": "135226"}
Now define a class method `info` that returns an array of the instance attribute names
Finally define a static method `greet` for the class that prints out 'Good Morning'

Question 4 For the same question, define an exception class CourseNotFoundError. The error should take in two string input . Write a string dunder method for the error class that returns the text 'The course <str1> is not provided by <str2>' where <str1> and <str2> are the first and second string input

Question 5

1. Write a class method `find_course` that takes in a class code as input and returns a copy of the Course instance object. To obtain a copy of a user-defined object, you first import the module `copy` then call the method `copy.deepcopy()` on the course.
2. If the course cannot be found, raise the `CourseNotFoundError` with the course code and department name as input.
3. Use a try-except block to look for a course that does not exist in the `csit_dept` Department object and prints the error object that was raised.

Question 6

Using the Employee class in tutorial 3, add a static method `validate_phone_num` that takes in a string and check that it starts with +65 by using <bool> = `<str>.startswith(substring)` and is 11 character long. If both conditions are met, return `True`, else , raise a `ValueError` .

Next define a class method from_dict that returns an Employee instance from a given dictionary. Before returning the employee instance, it should use the static method `validate_phone_num` to validate the phone number given in the dictionary.

Now with error handling, try to create an Employee instance using a dictionary and an invalid phone number. When an exception is raised, print the text 'Invalid phone number!'