Tutorial 3 (Week 5)

- Able to use function
- Have good understanding of class and object

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

Function name:	`triple`
Input arguments:	1 input argument: ● `sentence:` a string
Return values:	1 return value: the function returns a new string where each character of the input string gets repeated 3 times. For example, if the input argument `sentence` is Uni then the function returns the string UUUnnniii

Question 2. Using the function in Question 1, write a program that works **exactly** like the following example (the text in **bold** indicates the user input):

```
Enter a sentence: little fish

Triple effect: llliiittttttllleee   fffiiissshhh
```

Question 3. Consider the following rule to generate a sequence from an initial integer:

- If the number X is even then the next number is 3X + 1
- If the number X is odd then the next number is 2X + 2

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

Function name:	`next_number`
Input arguments:	1 input argument: ● `number:` an integer
Return values:	1 return value: if the input argument `number = X` is even then the function returns 3X + 1, if X is odd then return 2X + 2.

Question 4. Using the function in Question 3, write a program that works **exactly** like the following example (the text in **bold** indicates the user input):

```
Enter the initial number: 1
Sequence:
Step 0: 1
Step 1: 4
Step 2: 13
Step 3: 28
… this goes until the number becomes greater than 1
million then stops...
```

Question 5. Create a class called Employee with the following instance attributes:

- First name
- Last name
- Employee number
- Position
- Phone extension

Write a constructor to initialize all the above attributes.

Question 6. Create 5 objects of the class Employee. Use function print to display these 5 objects. Verify that the output is not user-friendly.

Question 7. Write the dunder __str__method so that it returns employee information in the following format:

Employee(1234567, John, Smith, Software Engineer, ext 4567)

Question 8. Use function print to display the 5 objects again.

Question 9. Write the dunder __repr__method. Call the dunder __repr__ method on the above 5 objects and display the output.

Question 10. Write a method call print_details which displays information in the following format

```
----------------E 1234567--
| John Smith               |
| Software Engineer        |
| x4567                    |
------------------------------------------
```

Call the method on the 5 Employee objects that you have created in Week 8 exercise and verify the output is correct.

Question 11. Create a static method that generates a random Employee object with real-world looking information. Use this static method and generate a few random Employee objects and display these objects.

Question 12. Try question 1 to 7 on this page
https://pynative.com/python-object-oriented-programming-oop-exercise/

Question 13.

- Create a class named `Course`. The `Course` class has instance attributes `description`, `course_code` and `credits`. All of which are given when the constructor is called.
- Create a class named `Department`. The `Department` class has instance attributes `name`, `department_code` and `courses`. The `name` and `department_code` attributes are given when the constructor is called. The `courses` attribute is an empty dictionary when an object of the `Department` class is instantiated.
- Define an instance method called `add_course` which takes in the `description`, `course_code`, `credits` of a course. In the method, instantiate a `Course` object with the given description, `course_code` and `credit`. With this `Course` object as value and the `course_code` as the key, insert a key value pair to the instance attribute `courses`. Finally, this method also returns the course object it created
- Create a class named `Student`. Each `Student` has instance attributes `name`, `student_number`, and `modules`. Both `name` and `student_number` are given as string input when the constructor is called. The attribute, `modules`, is a list that is empty when the `Student` is initialised.
- The `Student` class has an instance method `enroll` that takes in a `course` object. The method adds the `course` object into the instance list attribute modules.
- To test a code, add this to your script at end

```
csit_dept = Department("Computer Science and Information
Technology", "CSIT")
csit110 = csit_dept.add_course("Fundamental Programming
        with Python", "CSIT110", 6)
gunther = Student("Gunther", "Tan")
gunther.enrol(csit110)
```