Tutorial 4 (Week 6)

Objective

- Have good understanding of Classes and objects
- Able to differentiate, define and use instance, class and static methods
- Able to read and write files

1) Create a class called Employee with the following instance attributes:
   - First name
   - Last name
   - Employee number
   - Position
   - Phone extension

   Write a constructor that takes in values to initialize all the above attributes.

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

3) Write the dunder __str__method so that it returns employee information in the following format:

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

4) Use function print to display the 5 objects again.

5) Write the dunder __repr__ method. Call the dunder __repr__ method on the above 5 objects and display the output.

6) 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 Question 8 and verify the output is correct.

7) 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.

8) Try question 1 to 7 on this page

https://pynative.com/python-object-oriented-programming-oop-exercise/

9) Define a class that meets the following specification

Class name	Pizza
Constructer Argument	a list object
Name of instance attributes	ingredients: list Assign the list argument to this instance attribute in the constructor.
Instance method	__repr__ Define a __repr__ dunder instance method that returns a str of the line of code from which the instance was created.

10) Define a class that meets the following specification

Class name	Margherita
Constructer Argument	a list object
Name of instance attributes	This class inherits the Pizza class. By default, the ingredients are ['mozzarella', 'tomatoes']
Instance method	__repr__ Define a __repr__ dunder instance method that returns a str of the line of code from which the instance was created.

11) 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.

12) 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.

13) 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

14) 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.

15) 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.

16) 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)
```

EXCEPTION HANDLING

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.

17) 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 out of range

```
Enter assignment mark (0-20): 40
Error: assignment mark must be between 0 and 20
```

Example 4: Project mark is 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
```

18) 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

19) Using the function from the previous question, 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.

20) Using the `Student` class you defined in question 14, 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’],…]

21) 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"}`

22) Now define a class method info that returns an array of the instance attribute names

23) Finally define a static method greet for the class that prints out 'Good Morning'

24) 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

25) 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.

If the course cannot be found, raise the `CourseNotFoundError` with the course code and department name as input.

26) 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.

Using the `Employee` class in question 1, 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` .

27) 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.

28) 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!'`

FILES

Create a class called Student with the following attributes:

- First name
- Last name
- Student number

a. Write a constructor that initializes all the above attributes.

b. Write the dunder method __str_____that returns student information in this format:
`John Smith (1111)`

c. Write the dunder method __repr____that returns student information in this format:
`Student('John', 'Smith', '1111')`

d. Create 5 objects of Student class and put it into a list. Display the 5 student objects.

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

Function name:	`write_csv`
Input arguments:	2 input arguments: • `student_list:` a list of student objects • `file_path:` a file path to CSV file

Return values:	0 return values.
	The function writes the details of a list of student objects into a CSV file like this:
	`stn,first_name,last_name`
	`1111,John,Smith`
	`2222,Lee,May`

30) Using the function in Question 3, write the list of students in Question 2 to a CSV file.

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

Function name:	`read_csv`
Input arguments:	1 input argument: • `file_path:` a file path to CSV file
Return values:	1 return value: the function reads the CSV file, constructs a list of student objects and returns this list.

32) Using the function in the previous question, read the CSV file of the question before the previous question, then display the returned list of Student objects.