Name: Cathy DuPuis
Date: August 19, 2024
Course: IT FDN 110 Su 24 Foundations of Programming Python

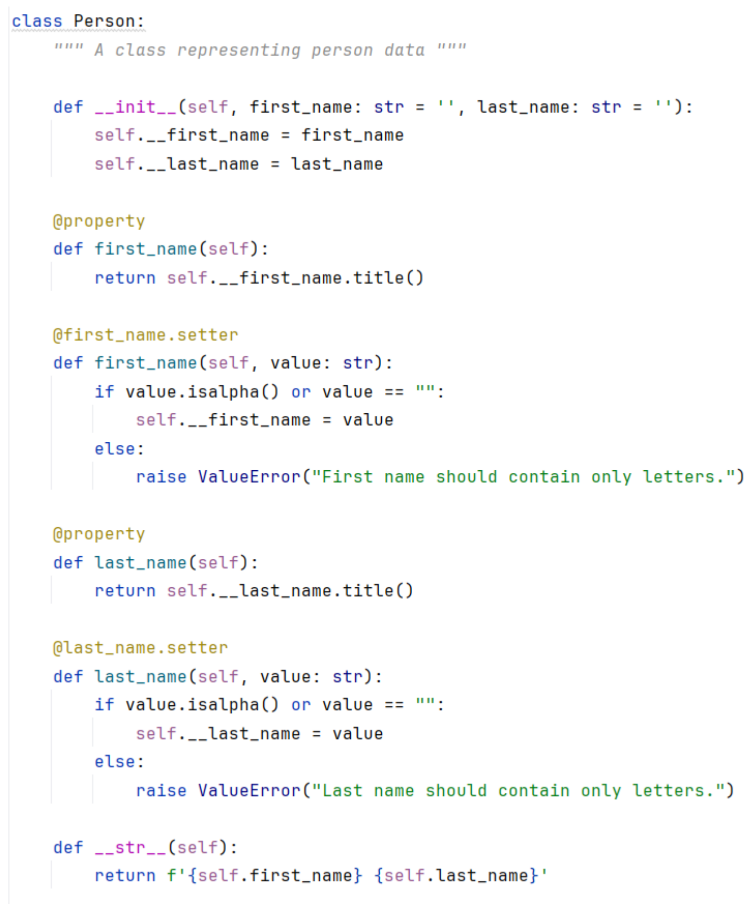
Assignment 07 – Classes and Objects

Introduction

In this assignment, I added to Prof. Root's Python program that demonstrates the use of parent and child data classes, reading/writing JSON files and structured error handling. The program manages student registrations for a course, with functionalities for inputting, displaying, and saving student data to a file using classes and lists.

Topic – Object-Oriented Programming (OOP) with Classes and Inheritance

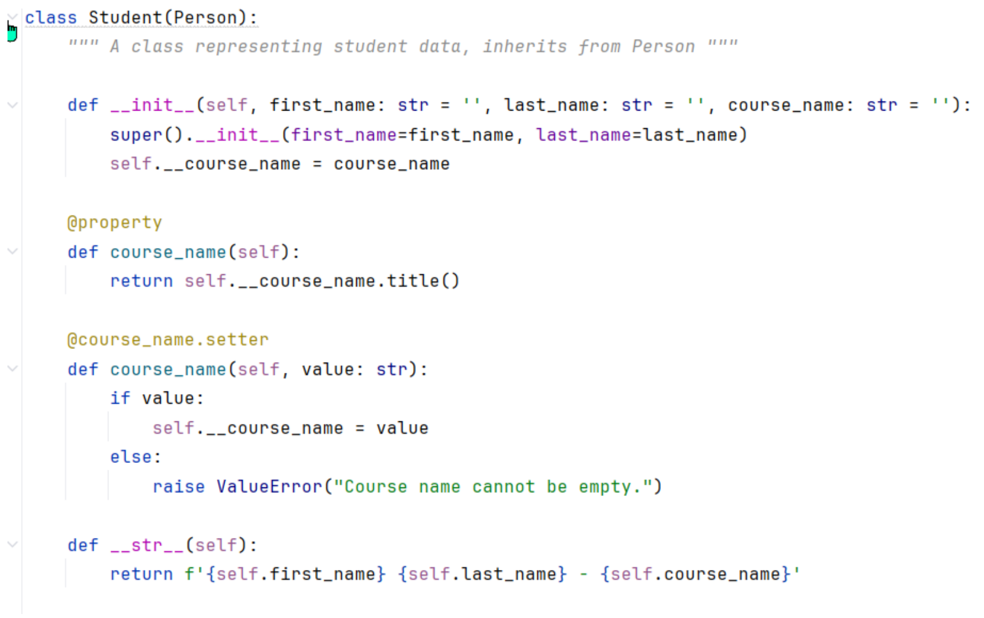
The assignment required the creation of two classes: Person and Student. The Person class encapsulates attributes such as first_name and last_name with appropriate getter and setter methods. It also includes a __str__ method to provide a readable string representation of the person (Figure 1).



```python
class Person:
    """ A class representing person data """

    def __init__(self, first_name: str = '', last_name: str = ''):
        self.__first_name = first_name
        self.__last_name = last_name

    @property
    def first_name(self):
        return self.__first_name.title()

    @first_name.setter
    def first_name(self, value: str):
        if value.isalpha() or value == "":
            self.__first_name = value
        else:
            raise ValueError("First name should contain only letters.")

    @property
    def last_name(self):
        return self.__last_name.title()

    @last_name.setter
    def last_name(self, value: str):
        if value.isalpha() or value == "":
            self.__last_name = value
        else:
            raise ValueError("Last name should contain only letters.")

    def __str__(self):
        return f'{self.first_name} {self.last_name}'
```

Figure 1. Screen shot of code to create Person (Parent) class

The Student class inherits from Person, adding a course_name attribute. This demonstrates the OOP concept of inheritance, where the Student class extends the functionality of the Person class. The super() function in the student (child) class is used to call a method from the parent class, allowing the child class to inherit and extend the functionality of the parent class's methods, such as the constructor.



```python
class Student(Person):
    """ A class representing student data, inherits from Person """

    def __init__(self, first_name: str = '', last_name: str = '', course_name: str = ''):
        super().__init__(first_name=first_name, last_name=last_name)
        self.__course_name = course_name

    @property
    def course_name(self):
        return self.__course_name.title()

    @course_name.setter
    def course_name(self, value: str):
        if value:
            self.__course_name = value
        else:
            raise ValueError("Course name cannot be empty.")

    def __str__(self):
        return f'{self.first_name} {self.last_name} - {self.course_name}'
```

Figure 2. Screen shot of code including super () to create the Student class

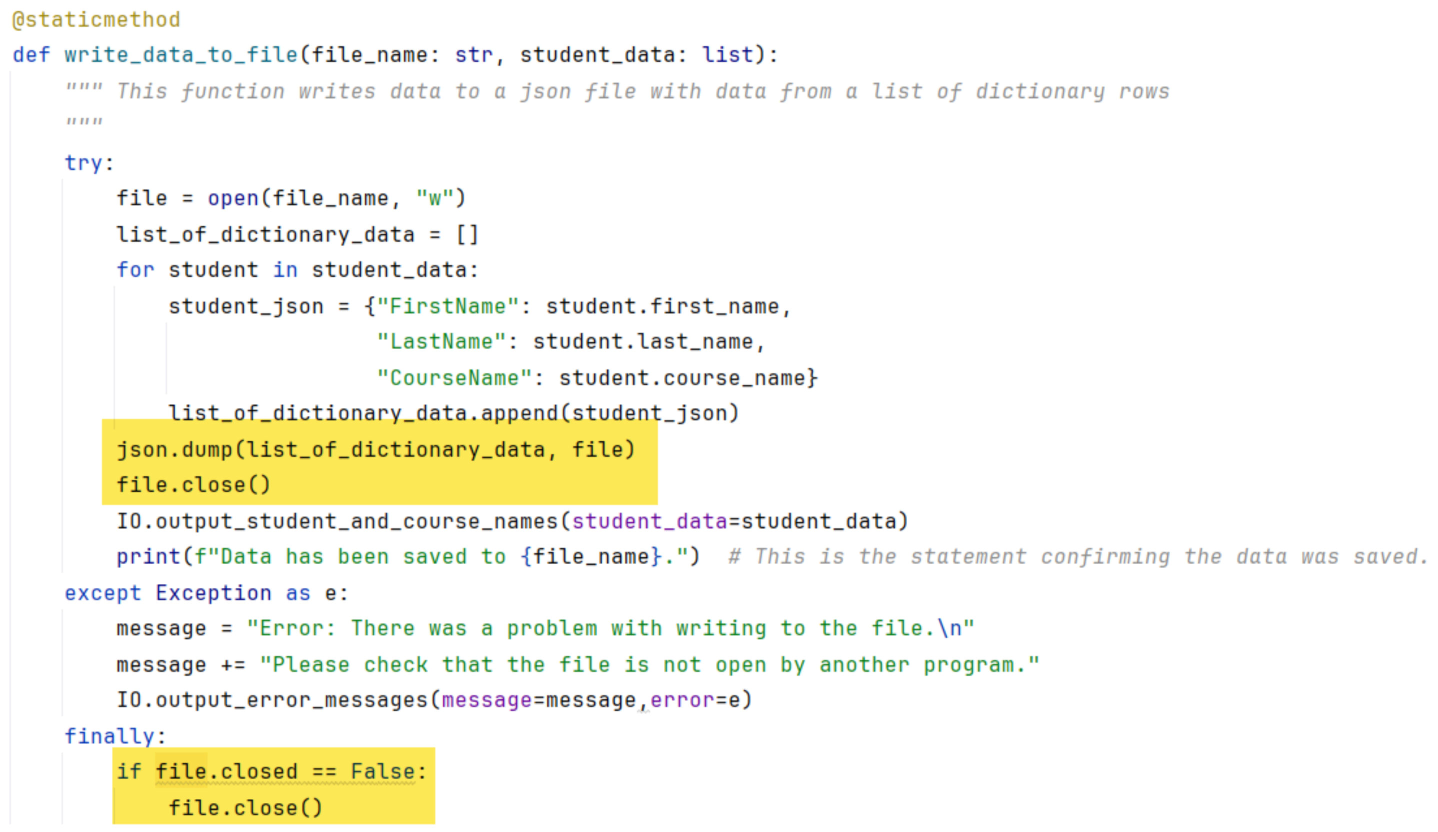
This structure ensures that the program is modular and easy to maintain, allowing for the reuse of code across different components and the ability to add additional child classes subordinate to the parent class as necessary.

Topic – File Handling with JSON

Another part of the assignment was reading from and writing to a file named Enrollments.json. This involved using Python's open() function in combination with json library functions to pack and unpack the data. When we put all the data to JSON, it's like packing up all the information (like student names and courses) into a box (the JSON file) so we can save it on the computer.

Then, when I want to use the information again, the file box has to be opened and unpacked so the Python program can work with it. The reason we have to go back and forth between a list of dictionaries and JSON is because the program works best with data in a dictionary list (which is easy to use in the code), but the computer needs the data in JSON format to save it as a file that can be stored and opened later.

The program reads existing student data from Enrollments.json which has been defined as a constant in the variable part of the program at the start and writes updated data back to the file when the user chooses to save. The file is explicitly closed using the close() method after each operation, ensuring that the data is properly saved and the file is not left open, which could lead to data corruption or access issues (Figure 3).



```
@staticmethod
def write_data_to_file(file_name: str, student_data: list):
    """ This function writes data to a json file with data from a list of dictionary rows
    """
    try:
        file = open(file_name, "w")
        list_of_dictionary_data = []
        for student in student_data:
            student_json = {"FirstName": student.first_name,
                            "LastName": student.last_name,
                            "CourseName": student.course_name}
            list_of_dictionary_data.append(student_json)
        json.dump(list_of_dictionary_data, file)
        file.close()
        IO.output_student_and_course_names(student_data=student_data)
        print(f"Data has been saved to {file_name}.")  # This is the statement confirming the data was saved.
    except Exception as e:
        message = "Error: There was a problem with writing to the file.\n"
        message += "Please check that the file is not open by another program."
        IO.output_error_messages(message=message,error=e)
    finally:
        if file.closed == False:
            file.close()
```

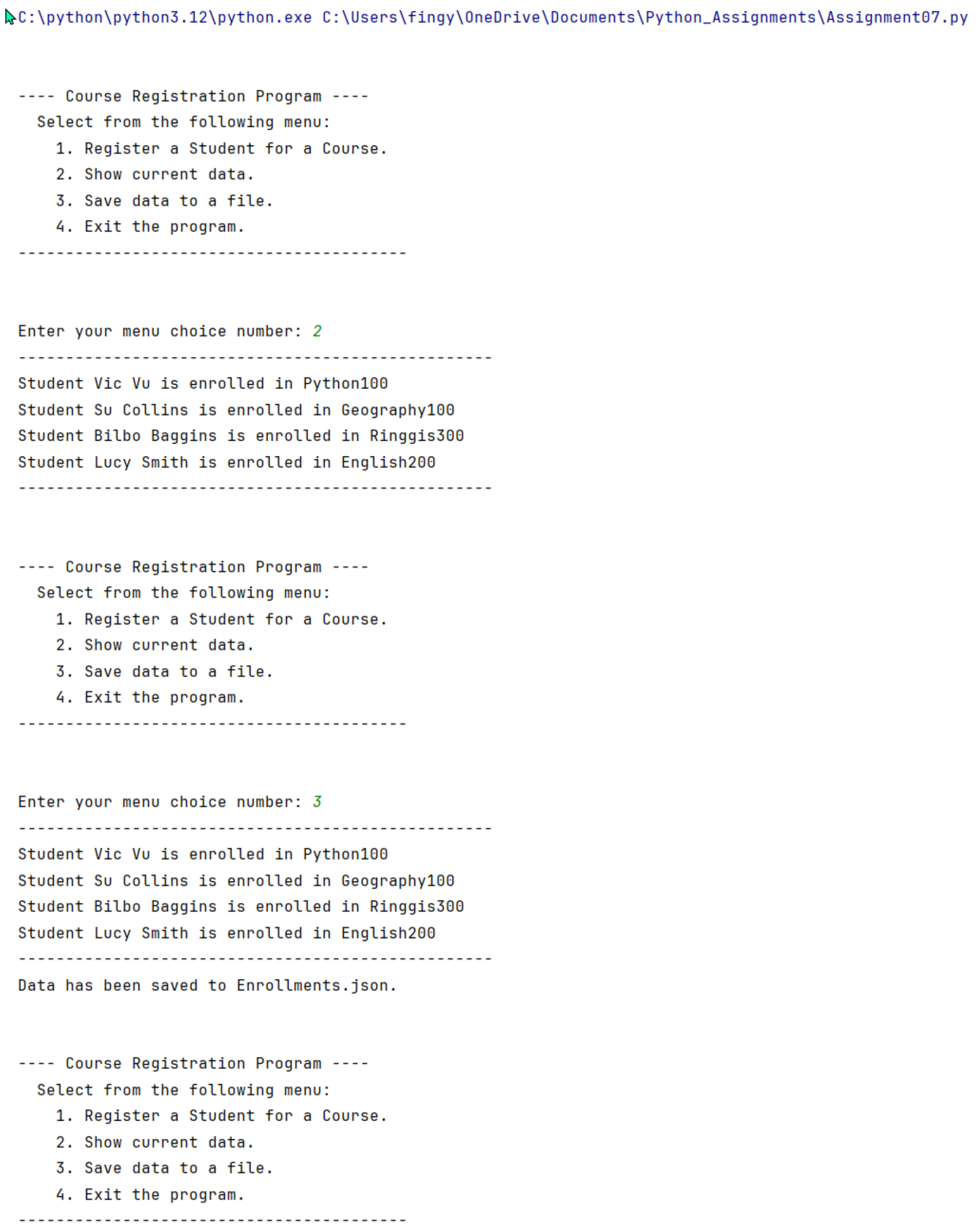
Figure 3. Code showing student name and course data to write to file_name

The use of JSON for file handling demonstrates the practical application of text-based data interchange formats, making the program's data both human-readable and easy to manipulate in a program.

Also, I added error handling to avoid the error messages when trying to open the Enrollments file if it didn't already exist because otherwise it kept crashing.

Topic – Testing

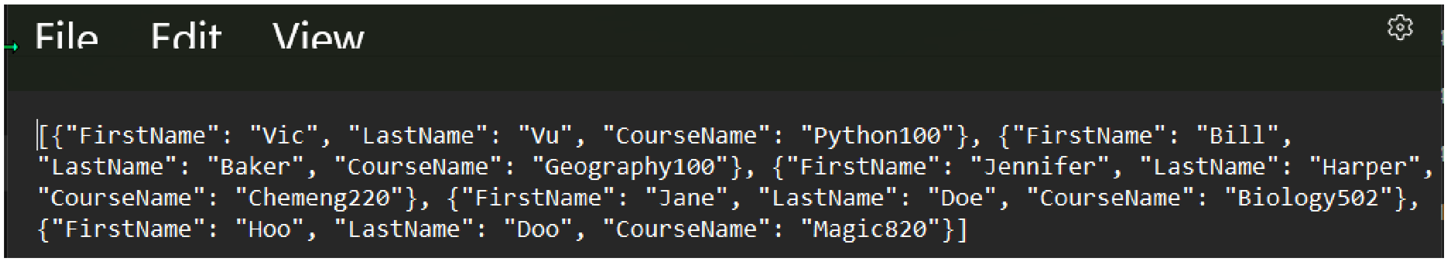
After inputting student registration data, the program then lists the first and last name and course name. (See Figure 4)



```
C:\python\python3.12\python.exe C:\Users\fingy\OneDrive\Documents\Python_Assignments\Assignment07.py


---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------


Enter your menu choice number: 2
--------------------------------------------------
Student Vic Vu is enrolled in Python100
Student Su Collins is enrolled in Geography100
Student Bilbo Baggins is enrolled in Ringgis300
Student Lucy Smith is enrolled in English200
--------------------------------------------------


---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------


Enter your menu choice number: 3
--------------------------------------------------
Student Vic Vu is enrolled in Python100
Student Su Collins is enrolled in Geography100
Student Bilbo Baggins is enrolled in Ringgis300
Student Lucy Smith is enrolled in English200
--------------------------------------------------
Data has been saved to Enrollments.json.


---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course.
    2. Show current data.
    3. Save data to a file.
    4. Exit the program.
-----------------------------------------
```

Figure 4. Student and course details displayed with successful program save.

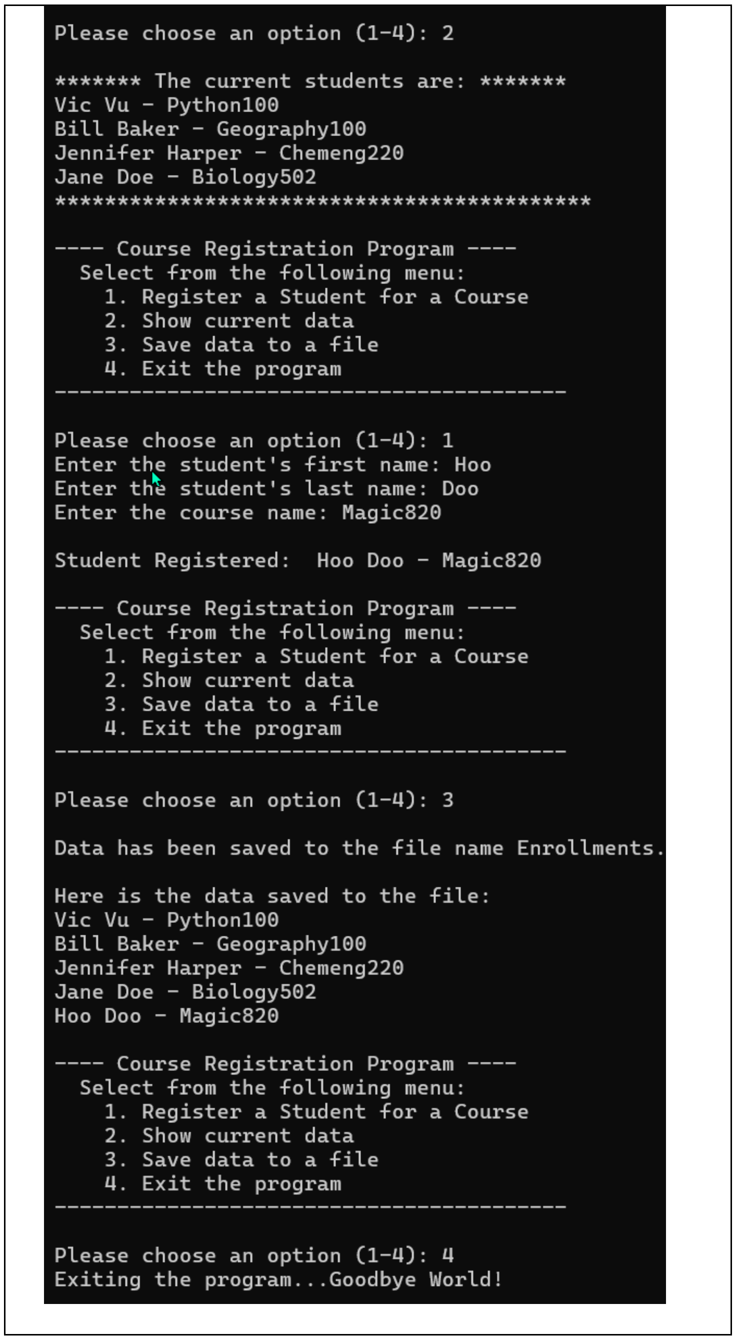
Using menu choice 3, the data gets saved to Enrollements.json and can be opened in a text editor like notepad to show the dictionary list rows (Figure 5).



```
[{"FirstName": "Vic", "LastName": "Vu", "CourseName": "Python100"}, {"FirstName": "Bill",
"LastName": "Baker", "CourseName": "Geography100"}, {"FirstName": "Jennifer", "LastName": "Harper",
"CourseName": "Chemeng220"}, {"FirstName": "Jane", "LastName": "Doe", "CourseName": "Biology502"},
{"FirstName": "Hoo", "LastName": "Doo", "CourseName": "Magic820"}]
```

Figure 5. Text file of json student data (Figure 4).

Per the assignment, I also tested in the MS Command Prompt and the program ran successfully in that environment (Figure 5). I verified that the program can register students, display current data, save data to a file, and handle multiple registrations and that it can display the dictionary data in Notepad with the key and individual string values.



```
Please choose an option (1-4): 2

******* The current students are: *******
Vic Vu - Python100
Bill Baker - Geography100
Jennifer Harper - Chemeng220
Jane Doe - Biology502
*******************************************

---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course
    2. Show current data
    3. Save data to a file
    4. Exit the program
-----------------------------------------

Please choose an option (1-4): 1
Enter the student's first name: Hoo
Enter the student's last name: Doo
Enter the course name: Magic820

Student Registered:  Hoo Doo - Magic820

---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course
    2. Show current data
    3. Save data to a file
    4. Exit the program
-----------------------------------------

Please choose an option (1-4): 3

Data has been saved to the file name Enrollments.

Here is the data saved to the file:
Vic Vu - Python100
Bill Baker - Geography100
Jennifer Harper - Chemeng220
Jane Doe - Biology502
Hoo Doo - Magic820

---- Course Registration Program ----
  Select from the following menu:
    1. Register a Student for a Course
    2. Show current data
    3. Save data to a file
    4. Exit the program
-----------------------------------------

Please choose an option (1-4): 4
Exiting the program...Goodbye World!
```

Figure 5. Command prompt working screen shots showing successful program runtime

Summary

This assignment shows how to use parent and student classes by using Person and Student in the program. Assignment 7 demonstrated the integration of object-oriented programming principles, file handling using JSON, and structured error management to edit a user-friendly course registration program. By using classes and inheritance, the program was designed to be modular and maintainable. File handling was managed carefully to ensure data integrity, while error handling mechanisms improved the user experience.

Github posting: https://github.com/CathyD-UW/IntroToProg-Python-Mod07