Name: Andrew Wysocki
Date: 02/14/24
Course: Introduction to Programming – Python
https://github.com/jaspercasidino/IntroToProg-Python-Mod05

Assignment 05 - Advanced Collections and Error Handling

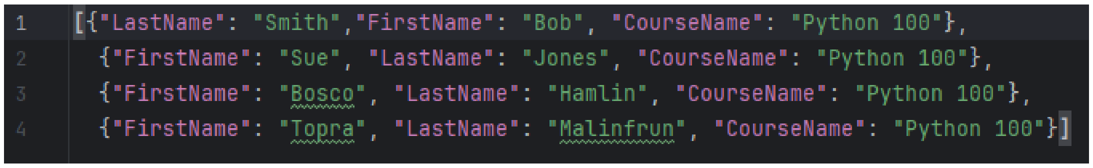
Introduction

This week's lesson covers dictionaries, json files and exception handling.

Topic – Dictionaries

Dictionaries are similar to lists that we learned about last week, but they are handled slightly differently. To start, lists use square brackets [] to contain the information, whereas dictionaries use curly brackets {}. The next difference is that I noticed was that each value in a dictionary has a label associated to the data. You reference that information by referring to the label as opposed to a numeric index like in a list. This allows for more readable code at the expense of having more code. One interesting thing that I noticed is that the information held in a file can have different order for the information if using dictionaries but the code can identify where that information is stored in the variable so the data is less order dependent like a list is.

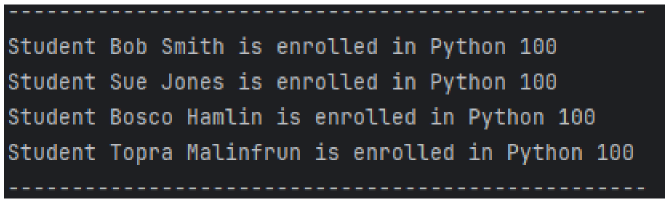
Example:



```
[{"LastName": "Smith","FirstName": "Bob", "CourseName": "Python 100"},
  {"FirstName": "Sue", "LastName": "Jones", "CourseName": "Python 100"},
  {"FirstName": "Bosco", "LastName": "Hamlin", "CourseName": "Python 100"},
  {"FirstName": "Topra", "LastName": "Malinfrun", "CourseName": "Python 100"}]
```

Bob Smith is listed but his last name is in the first place and his first name is in the second place. All the other students have first name then last name. When the code is run the output still displays correctly.

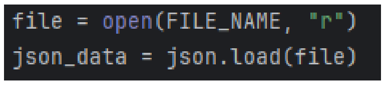
Result:



```
--------------------------------------------------
Student Bob Smith is enrolled in Python 100
Student Sue Jones is enrolled in Python 100
Student Bosco Hamlin is enrolled in Python 100
Student Topra Malinfrun is enrolled in Python 100
--------------------------------------------------
```

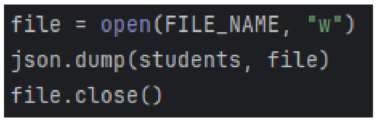
Topic – json Files

json files work really well with dictionary variables as they were modeled after them. The information stored in a json file is contained within square brackets and individual information is stored like what a dictionary variable would look like. An example of the contents of a json file is shown in the previous section. Using json files and code requires the use of the import function in Python. This is a separate library that can be imported to make sure of custom scripts that work with json files. The code that we used made reading from and writing to json files extremely easy. To load data, use json *load* function and store the information into a variable.



```
file = open(FILE_NAME, "r")
json_data = json.load(file)
```

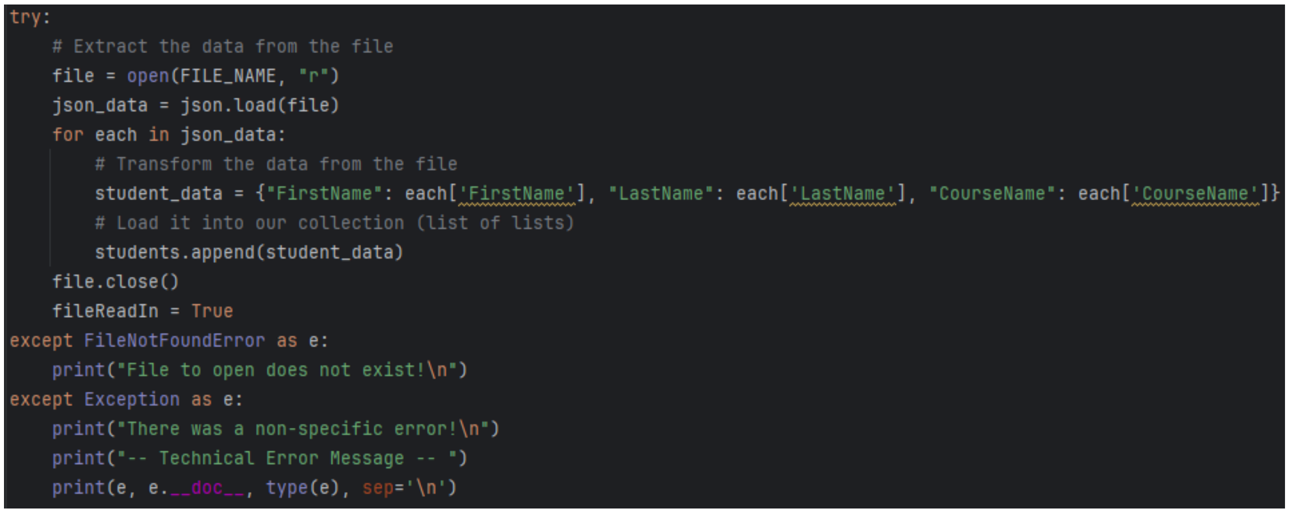
To write to a file, use the *dump* command. It “dumps” all the data stored in the variable of interest into the file.



```
file = open(FILE_NAME, "w")
json.dump(students, file)
file.close()
```

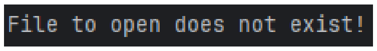
Topic – Exception Handling

Exception handing is used to control the flow of a program as well as inform the program user of issues that arise when running the program. In this assignment, we checked for a few things. The first was checking that the Enrollments.json file was successfully opened. If the file wasn’t present, we would use the FileNotFoundError exception to display a message to the user that the file does not exist. If some other error occurred, we use the Exception exception to display the technical error to help the user understand what went wrong.



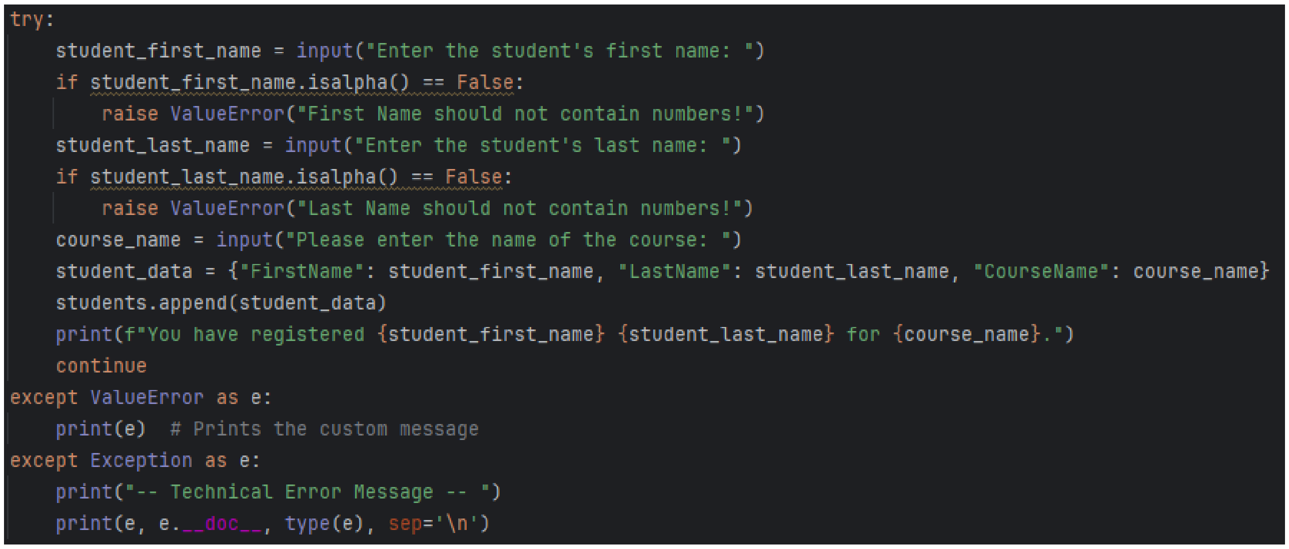
```
try:
    # Extract the data from the file
    file = open(FILE_NAME, "r")
    json_data = json.load(file)
    for each in json_data:
        # Transform the data from the file
        student_data = {"FirstName": each['FirstName'], "LastName": each['LastName'], "CourseName": each['CourseName']}
        # Load it into our collection (list of lists)
        students.append(student_data)
    file.close()
    fileReadIn = True
except FileNotFoundError as e:
    print("File to open does not exist!\n")
except Exception as e:
    print("There was a non-specific error!\n")
    print("-- Technical Error Message -- ")
    print(e, e.__doc__, type(e), sep='\n')
```

Result:



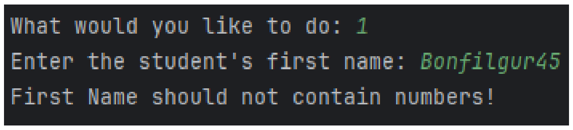
```
File to open does not exist!
```

We also did error checking when entering student information. If the user enters a first or last name with numbers, we display a message to the user stating that the name cannot have numbers and we do not update the student variable with the erroneous information.



```python
try:
    student_first_name = input("Enter the student's first name: ")
    if student_first_name.isalpha() == False:
        raise ValueError("First Name should not contain numbers!")
    student_last_name = input("Enter the student's last name: ")
    if student_last_name.isalpha() == False:
        raise ValueError("Last Name should not contain numbers!")
    course_name = input("Please enter the name of the course: ")
    student_data = {"FirstName": student_first_name, "LastName": student_last_name, "CourseName": course_name}
    students.append(student_data)
    print(f"You have registered {student_first_name} {student_last_name} for {course_name}.")
    continue
except ValueError as e:
    print(e)  # Prints the custom message
except Exception as e:
    print("-- Technical Error Message -- ")
    print(e, e.__doc__, type(e), sep='\n')
```

Result:



```
What would you like to do: 1
Enter the student's first name: Bonfilgur45
First Name should not contain numbers!
```

Summary

This module covered dictionaries, json files and exception handling. Dictionaries are similar to lists but have unique functionality that allows for easier to read code and are less sensitive to order of data than lists are. json files work really well with lists and the ability to read from and write to files is very simple. Exception handling allows for error checking and can be used to inform the user about errors when running program that can be expressed in plain English should that be desired and coded into the program.