823G5 Programming in Python

Aim

The aim of this exercise set is to provide you with advanced understanding and practical experience in several key Python topics:

1. Advanced f-string Formatting:

   You will explore advanced formatting techniques using f-strings, incorporating precision, alignment, mathematical operations, and conditional expressions. This exercise aims to enhance your string formatting skills.

2. File I/O with JSON:

   You will practice reading from and writing to JSON files in Python. This exercise aims to familiarize you with the JSON format, enabling you to manipulate data in a structured manner.

3. Logging:

   This exercise set focuses on implementing logging in Python applications. You will learn to configure basic logging, customize log formats, and enable log file rotation. The aim is to instil good logging practices for effective debugging and monitoring.

4. File I/O with string processing:

   This exercise develops a simple To-Do list application for reading and writing lists of tasks to a plain text file. The aim is to draw together many of the elements covered concerning file handling, paths, string processing and object-oriented programming.

By completing these exercises, you will:

- Strengthen your understanding of advanced string formatting techniques and common string processing methods.
- Gain proficiency in reading from and writing to files for data storage and retrieval in an Operating System independent way.
- Develop skills in implementing logging mechanisms in Python applications, ensuring effective tracking and troubleshooting capabilities.

Exercises:

Download the **`Lab9Exercises`** ZIP file for solutions.

Challenge 1: Advanced f-string Formatting

1) **f-string Precision and Alignment**

   Given the following variables:

```
name = "Alice"
age = 30
experience = 3
units = "months"
score = 47
total = 60
```

   Create an f-string that prints the information in the following format:

```
Alice is 30 years old and has 3 months of programming
experience.
She scored 78.3% in her Python exam.
```

2) **Mathematical Operations in f-strings**

   Use f-strings to calculate and display the result of the expression:

```
x = 5
y = 3
```

   The output should be:

```
5 times 3 is 15.
```

3) **Conditional Expressions in f-strings**

   Given the boolean variable:

```
is_student = True
```

   Create an f-string that prints:

```
Welcome to the class!
```

   If `is_student` is `True`, otherwise print:

```
Welcome to the jungle!
```

Challenge 2: File I/O with JSON

1) Writing and Reading JSON Files

   Create a list of Python dictionaries representing three people with the keys: name, age, and city. Write this dictionary to a JSON file named "`person.json`". Then, read the contents of the JSON file and print them in alphabetical order.

2) JSON Data Manipulation

   Given a JSON file named "`data.json`" with the following structure:

```
{
  "students": [
    {"name": "John", "grades": [55, 63, 77]},
    {"name": "Alice", "grades": [92, 84, 89]},
    {"name": "Bob", "grades": [78, 80, 73]}
  ]
}
```

   Load the data from the file, add a new student named "`Charlie`" with grades of 64, 71 and 75, print out each student with their average grade and then write the updated data back to the file.

Challenge 3: File I/O with JSON

1) **Basic Logging Setup**
   Implement basic logging in Python. Create a script that logs messages of different levels (debug, info, warning, error, critical) to a file named "`app.log`".

2) **Custom Log Format**
   Enhance the logging from Exercise 6 by customizing the log format to include the timestamp, log level, and the message itself.

3) **Log File Rotation**
   Extend the logging configuration to enable log file rotation. Ensure that log files are rotated every day, keeping a backup of the last 7 log files.

Challenge 4: Write a simple To-Do List application

This exercise draws together many of the elements covered in reading and writing to plain text files, along with ways of processing strings to extract specific information. The application will read and write to a simple text file containing a numbered list of tasks. The file will be stored in the (default) subdirectory "`todo`" and named with today's date.

Some starter code is provided along with some hints as comments. You will need to complete the "stub" methods to implement the required functionality and produce the following output when the script is run:

```
To-Do list: todo/2023-12-05.txt (3 tasks)
====================
1: Feed the dog
2: Buy milk
3: Water the plants

--------------------

To-Do list: todo/2023-12-05.txt (4 tasks)
====================
1: Feed the dog
2: Water the plants
3: Call mum
4: Finish Python assignment

--------------------
```

Several useful standard modules are already imported in the starter code – the functions: `os.path.join()`, `os.makedirs()`, `os.path.exists()` and `date.today()` may be particularly helpful for this exercise. You will need to check the documentation to see how to use them correctly.

Note the files are opened explicitly with "utf-8" character encoding. This is the default encoding used by the function and determines the character set that can be used.

Experiment with specifying different directories and filenames. You could even extend the functionality to add items with a specified priority.

Dr. Benjamin Evans
B.D.Evans@sussex.ac.uk