Laboratory Report

Ollscoil Teicneolaíochta an Atlantaigh

Atlantic Technological University

Lecturer:	John O'Raw
Report Title:	Python
File Name:	L00171202.pdf
Submit to:	Blackboard in PDF format only
Date Submitted:	3rd November 2022

Student Name:	Dharmender Singh
Student Number:	L00171202
Programme of Study:	MSc in Cloud Computing
Module:	INFS IT903

Contents

Description

The purpose of this report is to summarize the python work done for the Infrastructure as code module. The practical work was completed in accordance with the lecturer's provided walkthroughs for exercise 7 through 12. As we put together this report, we took advantage of many of the built-in library's features. Testing of code, datetime and network utilities were checked along with file handling, exceptions and classes. The result of each walkthrough was recorded.

Aim

The Python walkthroughs identified the following goals to be achieved. With the aid of walkthroughs, all of the objectives were accomplished to foster a better understanding of Python -

1. To get knowledge of working with packages and operating systems.
2. To practice handling exceptions by utilizing the try, except, else, and finally approach.
3. To learn handling files.
4. To become familiar with Object-Oriented Concepts.
5. To grasp how classes and inheritance work.
6. To gain knowledge about Pylint and unit testing.
7. To comprehend the time and logging principles in Python.
8. To use Python to check the connection between the FTP, TCP, Multicast, and UDP protocols.
9. Dealing with subdirectories

Methodology

We complied with the walkthroughs' recommendations and followed them exactly. The result of each walkthrough is saved in different folder.

1. Done work on exercises in walkthrough 7. [1] These works are - detect_working_directory.py, create_directory.py, files.py, validation_and_raising.py, fuel_exercise.py.

2. Done work on exercises in walkthrough 8. [2]These works are - class_template.py, devices.py, main.py, oo1.py, oo2.py.

3. Done work on exercises in walkthrough 9. [3] These works are - formater.py, pylint.py, test_formater.py.

4. Done work on exercises in walkthrough 10. [4]These works are - file_utilities.py, main.py, mytime.py, os_utilities.py.

5. Done work on exercises in walkthrough 11. [5] These works are - ftp_client.py, mc_client.py, mc_server.py, tcp_client.py, tcp_server.py, udp_client.py, udp_server.py.

6. Done work on exercises in walkthrough 12. [6]These works are - project.bat, main.py, detectOS.py.

Results and Testing

Since all the objectives were achieved, the.py files are included in the zip folder that is connected to this report and goes with it. The directory was successfully created and the operating system of the current machine was identified as Windows. Along with file functions such as Append, Read, Write, Read, and Write, exception handling was known and employed. In addition to overriding the parent class function, the child class also inherited the parent class's attributes. The code was examined using unit testing and pylint, with a score of 10 in pylint. A task was completed on the specified date and time in a human-readable format for external reading on a website, and the system's log was completed and logged. The protocols for FTP, TCP, UDP, and Multicast were successfully tested. On a small project, work on subdirectories was completed.

Conclusions

The walkthroughs were skimmed over, and the learned information was applied when writing Python code. Additionally, the codes were tested and successfully ran at the Command prompt. The additional activities were completed with the use of information collected from the Python notes, and eventually, knowledge from the walkthroughs was created.

Operating on several operating systems and identifying the OS you are using was done, along with determining the current directory being utilized by platform packages. Complex codes are challenging to work with, and errors are always possible. These errors were further resolved with exception handling by using the concept of try, which is the block of code that will be executed whether there is an error or not. Furthermore, a practice based on these ideas was done (exercise fuel.py). The ZerodivisionError was utilized, keeping in mind that if the denominator was 0, the code was not functioning. The code could not use the continue function and would not restart the loop if the user inputs a string, though. A deeper understanding of this will be sought after. Next, work on the file was done using file operations like append, which inserts data at the end of the file, read, which opens the file for reading only and starts the handle at the beginning, write, which writes in the file, and read and write, which opens the file for reading and writing. Results are stored in exercise 07 folder.

The study of object-oriented principles led to the conclusion that class objects, once generated, could be used and called in the same way for every instance of a class. If the parent class and the child class both have the same methods and the child class is able to call the parent class's methods, attributes from the child class will be utilised. The Abstract class can be used to declare any characteristics and methods that are shared by all instances of all child classes. The methods of the child class override the parent class's methods when the child class inherits code from the parent class. The Exercise 08 folder contains all of the codes.

The next step involved installing pylint and using pip to test the Python code. A Python tool called Pylint is used to analyze static code and implement best practices in development by looking for programming faults. Unit testing was carried out with the aid of unittest, a Python tool that checks answers and flags any errors. Writing code with uppercase names and adding a new line, testing it with a score of 10/10, and the test results were satisfactory. The Exercise 09 folder houses all the codes.

In order to verify how time and date are represented in Python using the built-in library date time, the strftime() method of the library was employed. After comparing the current time to the Unix epoch and writing a piece of code to make the date and time available in year-month-date and hour-minute-second formats, an exercise was completed. This code was written with further assistance from a

website. Code was used to note and record system log information. The folder titled "Exercise 10" contains all of the codes.

Using Python code, network protocols like FTP, TCP, Multicast, and UDP were examined and tested. Paths, URLs, and other data, such as file names, were stored in a separate directory called settings in order to protect them from the main code and preserve the main code's simplicity. To send packets from the client and receive them from the server, there were specific client and server files for each protocol. The folder titled "Exercise 11" contains all of the codes.

A simple project directory structure was constructed for Windows using a batch file, and the subdirectories Documentation, Tests, Examples, and Source were added inside. Once the project structure and all its actions were known, the main file imported the source package. The Exercise 12 folder contains all of the codes.

As a result of working on these walkthroughs and creating such codes, a greater grasp on Python was formed. The objectives were also effectively attained, and all the scripts function.

References

References

[1] j. o'raw, "Infrastructure as Code - Walkthrough 07 - Handling Errors," [Online]. Available: https://learn-eu-central-1-prod-fleet01-xythos.content.blackboardcdn.com/5b4d776acbe47/4602310?X-Blackboard-Expiration=1667530800000&X-Blackboard-Signature=ojnRjdQQy0LNQdvrgw%2BNV4lKyAWo%2BWVTZERdd9sXtGY%3D&X-Blackboard-Client-Id=309122&response-cache-con. [Accessed november 2022].

[2] J. 0'raw, "Infrastructure as Code - Walkthrough 08 - Python Object Oriented Coding," [Online]. Available: https://learn-eu-central-1-prod-fleet01-xythos.content.blackboardcdn.com/5b4d776acbe47/4602311?X-Blackboard-Expiration=1667530800000&X-Blackboard-Signature=iFsJaXaAVypiUhtk2gu2UDocBggDLyEKzbK%2Bkx%2FQoAs%3D&X-Blackboard-Client-Id=309122&response-cache-con. [Accessed november 2022].

[3] J. o'raw, "Infrastructure as Code - Walkthrough 09 - Python Tests," [Online]. Available: https://learn-eu-central-1-prod-fleet01-xythos.content.blackboardcdn.com/5b4d776acbe47/4602312?X-Blackboard-Expiration=1667530800000&X-Blackboard-Signature=rS1lVT9QkgXLDGPr3azdkvtvs8RPHoG1wnPNK64zFEI%3D&X-Blackboard-Client-Id=309122&response-cache-control. [Accessed november 2022].

[4] J. o'raw, "Infrastructure as Code - walkthrough 10 - Python Simple Logging," [Online]. Available: https://learn-eu-central-1-prod-fleet01-xythos.content.blackboardcdn.com/5b4d776acbe47/4649186?X-Blackboard-Expiration=1667530800000&X-Blackboard-Signature=K%2BanAQ3z52MKre2jARY2EHNtn0XiV3lU3bLfT3hYN8o%3D&X-Blackboard-Client-Id=309122&response-cache-contr. [Accessed november 2022].

[5] J. o'raw, "Infrastructure as Code - walkthrough 11 - Python Network Utilities," [Online]. Available: https://learn-eu-central-1-prod-fleet01-xythos.content.blackboardcdn.com/5b4d776acbe47/4649181?X-Blackboard-Expiration=1667530800000&X-Blackboard-Signature=yzkTxgYABHlkycFk8ZiMk6qFxKV0r2aI53Vov1T2kyg%3D&X-Blackboard-Client-Id=309122&response-cache-control. [Accessed november 2022].

[6] J. o'raw, "Infrastructure as Code - Walkthrough 12 - Creating a project," [Online]. Available: https://learn-eu-central-1-prod-fleet01-xythos.content.blackboardcdn.com/5b4d776acbe47/4654606?X-Blackboard-Expiration=1667530800000&X-Blackboard-Signature=zyTAWKmBXrDqC%2F4zJnThDCJtgejYnm52dCILvyP0YSo%3D&X-Blackboard-Client-Id=309122&response-cache-contr. [Accessed november 2022].