FLEMING

Lab 3: Business Process Automation

COMP 593 - Scripting Applications

TABLE OF CONTENTS

LEARNING OBJECTIVES

Upon completion of this lab assignment, students should be able to:

- Install a package from the **Python Package Index (PyPI)** using **pip**
- Use the **pandas** package to execute basic processing operations on tabular data imported from a CSV file
- Use the **pandas** and **XlsxWriter** packages to export formatted data to an Excel sheet
- Use Python's built-in **os**, **sys**, **datetime**, and **re** modules to perform some common operations.
- Discuss functional decomposition in the context of script design
- Explain the purpose of a .gitignore file

INTRODUCTION

Although it is possible to implement many useful scripts using the packages, modules, and built-in features included in the default Python installation, the real power of Python lies in the extensive collection of packages developed by the Python community. For almost any Python programming challenge that you could possibly encounter, it is very likely that somebody else has already encountered a similar challenge, and subsequently developed and published a software package that you could use to simplify your solution.

The Python Package Index (PyPI) is a repository of software for the Python programming language, that helps users find and install software packages developed and shared by the Python community. PyPI hosts hundreds of thousands of packages created by tens of thousands of authors and maintainers. The default Python installation includes a utility named pip that can be used to easily download and install packages from PyPI.

It should be noted that PyPI has very few safeguards with respect to the security of the software packages that it hosts. Anybody can upload a package, and the code is not audited, independently reviewed, or scanned for malware. This article, which may be of particular interest to CSI students, discusses the risks of using PyPI and proposes using an alternative source for downloading select Python packages that have been vetted for security.

For this lab assignment, we will be using packages that are very popular, and therefore unlikely to be infected with malware, so there is little risk in downloading them from PyPI. Specifically, we will be using the following packages:

- pandas – A powerful data analysis toolkit
- OpenPyXL – A library to read/write Excel 2010 xlsx/xlsm files
- XlsxWriter – A module for writing files in the Excel 2007+ xlsx file format

SCRIPT OVERVIEW

For this lab assignment, you will create a Python script that extracts data from **sales_data.csv** to produce individual Excel files for each order, per the functional requirements described in the next section. The script output will be a collection of Excel files, each of which contains information and is formatted as shown in the Excel sheet depicted below.

	A	B	C	D	E	F	G	H	I
1	ORDER DATE	ITEM NUMBER	PRODUCT LINE	PRODUCT CODE	ITEM QUANTITY	ITEM PRICE	TOTAL PRICE	STATUS	CUSTOMER NAME
2	1/6/2019	1	Vintage Cars	S24_3969	49	$34.47	$1,689.03	Shipped	Online Diecast Creations Co.
3	1/6/2019	2	Vintage Cars	S18_2248	50	$67.80	$3,390.00	Shipped	Online Diecast Creations Co.
4	1/6/2019	3	Vintage Cars	S18_1749	30	$100.00	$3,000.00	Shipped	Online Diecast Creations Co.
5	1/6/2019	4	Vintage Cars	S18_4409	22	$86.51	$1,903.22	Shipped	Online Diecast Creations Co.
6						GRAND TOTAL:	$9,982.25		

INSTRUCTOR NOTES

The instructor will guide students through implementation of most of the script, explaining key concepts, and where/how to learn more about those concepts.

Students will complete the script, implementing the functional requirements pertaining to Excel file formatting, and are encouraged to reference the following websites:

- https://xlsxwriter.readthedocs.io/example_pandas_column_formats.html
- https://xlsxwriter.readthedocs.io/worksheet.html

PREPARATION

INSTALL PYTHON PACKAGES

Open PowerShell and run each of the following commands to install the packages required for this project.

- **pip install pandas**
- **pip install openpyxl**
- **pip install XlsxWriter**

Read the pip command output to ensure that each package was installed correctly, i.e., there are no error messages. The most likely cause of an error is a typo in the command, but other errors are possible that may require some Internet research to resolve.

Create a Local Git Repository

Open GitHub Desktop and create a new repository to hold the code you will develop for this lab assignment.

Create a .gitignore File

Files and folders that are created by a script are typically not included in the script's Git repository. To exclude these items, .gitignore file can be included in the repository that specifies which files Git should ignore. Each line in a gitignore file specifies a pattern. The pattern formatting rules are similar to wildcard patterns where, for example, the line ***.csv** I a .gitignore files would tell Git to exclude any item that ends with .csv from the repository.

The script you will be writing for this lab will create a folder named "Orders_YYYY-MM-DD", where YYYY-MM-DD is the date in which the script is run. This folder and all its contents should be excluded from the Git repository.

Add a **.gitignore** file to the repository that tells Git to exclude any item from the repository that begins with "Orders".

Retrieve the Sales Data CSV File from D2L

Go to the **Module 3** folder in D2L and download the **sales_data.csv** file. Save it in the repository folder. This file contains the source data that will be the input to your script.

View the Sales Data CSV File Contents

Open the repository folder in VS Code.

Install the VS Code extension named **Excel Viewer**.

Right click the **sales_data.csv** file in the VS Code Explorer, select **Open With…**, and the select **CSV Editor**.

Look through the CSV file content to see the information that it contains. Notice that:

- There are multiple rows of data that have the same ORDER ID.
- The rows are not arranged in any order.
- For each ORDER ID, the ITEM NUMBER in each row is a unique integer that could be sequenced 1, 2, 3, 4,…
- There is no column indicating the total price, i.e., ITEM QUANTITY x ITEM PRICE.

FUNCTIONAL REQUIREMENTS

COMMAND LINE PARAMETERS

{REQ-1} The script shall accept the following command line parameter:

- Path of the sales data CSV file

{REQ-2} If no command line parameter is provided, a descriptive error message shall be output, and the script shall exit.

{REQ-3} If the provided command line parameter does not specify the path to an existing file, a descriptive error message shall be output, and the script shall exit.

ORDERS DIRECTORY

{REQ-4} The Excel files for all orders shall be saved in a sub-directory of the directory in which the sales data CSV file resides.

{REQ-5} The sub-directory shall be named "Orders_YYYY-MM-DD", where YYYY-MM-DD is the ISO-formatted date on which the script generated the Excel files for each order.

{REQ-6} The sub-directory shall be created by the script if it does not already exist.

ORDER FILES

INFORMATION EXTRACTED FROM SALES DATA CSV FILE

{REQ-7} Individual Excel (.xlsx) files shall be created for each order that contain information extracted from the sales data CSV file as highlighted in the image below.

	A	B	C	D	E	F	G	H	I
1	ORDER DATE	ITEM NUMBER	PRODUCT LINE	PRODUCT CODE	ITEM QUANTITY	ITEM PRICE	TOTAL PRICE	STATUS	CUSTOMER NAME
2	1/6/2019	1	Vintage Cars	S24_3969	49	$34.47	$1,689.03	Shipped	Online Diecast Creations Co.
3	1/6/2019	2	Vintage Cars	S18_2248	50	$67.80	$3,390.00	Shipped	Online Diecast Creations Co.
4	1/6/2019	3	Vintage Cars	S18_1749	30	$100.00	$3,000.00	Shipped	Online Diecast Creations Co.
5	1/6/2019	4	Vintage Cars	S18_4409	22	$86.51	$1,903.22	Shipped	Online Diecast Creations Co.
6						GRAND TOTAL:	$9,982.25		

{REQ-8} The order items shall be arranged in ascending order by item number.

INFORMATION INSERTED BY SCRIPT

{REQ-9} A column shall be inserted that indicates the total price of each order line item, calculated as (item quantity x item price), as highlighted in the image below.

	A	B	C	D	E	F	G	H	I
1	ORDER DATE	ITEM NUMBER	PRODUCT LINE	PRODUCT CODE	ITEM QUANTITY	ITEM PRICE	TOTAL PRICE	STATUS	CUSTOMER NAME
2	1/6/2019	1	Vintage Cars	S24_3969	49	$34.47	$1,689.03	Shipped	Online Diecast Creations Co.
3	1/6/2019	2	Vintage Cars	S18_2248	50	$67.80	$3,390.00	Shipped	Online Diecast Creations Co.
4	1/6/2019	3	Vintage Cars	S18_1749	30	$100.00	$3,000.00	Shipped	Online Diecast Creations Co.
5	1/6/2019	4	Vintage Cars	S18_4409	22	$86.51	$1,903.22	Shipped	Online Diecast Creations Co.
6						GRAND TOTAL:	$9,982.25		

{REQ-10} A row shall be inserted that indicates the grand total of the order, calculated as the sum of all total prices, as highlighted in the image below.

	A	B	C	D	E	F	G	H	I
1	ORDER DATE	ITEM NUMBER	PRODUCT LINE	PRODUCT CODE	ITEM QUANTITY	ITEM PRICE	TOTAL PRICE	STATUS	CUSTOMER NAME
2	1/6/2019	1	Vintage Cars	S24_3969	49	$34.47	$1,689.03	Shipped	Online Diecast Creations Co.
3	1/6/2019	2	Vintage Cars	S18_2248	50	$67.80	$3,390.00	Shipped	Online Diecast Creations Co.
4	1/6/2019	3	Vintage Cars	S18_1749	30	$100.00	$3,000.00	Shipped	Online Diecast Creations Co.
5	1/6/2019	4	Vintage Cars	S18_4409	22	$86.51	$1,903.22	Shipped	Online Diecast Creations Co.
6						GRAND TOTAL:	$9,982.25		

FORMATTING

{REQ-11} All price information shall be formatted as money, i.e., prefixed with $, comma-separated thousands, and two decimal places for cents, as highlighted in the image below.

	A	B	C	D	E	F	G	H	I
1	ORDER DATE	ITEM NUMBER	PRODUCT LINE	PRODUCT CODE	ITEM QUANTITY	ITEM PRICE	TOTAL PRICE	STATUS	CUSTOMER NAME
2	1/6/2019	1	Vintage Cars	S24_3969	49	$34.47	$1,689.03	Shipped	Online Diecast Creations Co.
3	1/6/2019	2	Vintage Cars	S18_2248	50	$67.80	$3,390.00	Shipped	Online Diecast Creations Co.
4	1/6/2019	3	Vintage Cars	S18_1749	30	$100.00	$3,000.00	Shipped	Online Diecast Creations Co.
5	1/6/2019	4	Vintage Cars	S18_4409	22	$86.51	$1,903.22	Shipped	Online Diecast Creations Co.
6						GRAND TOTAL:	$9,982.25		

{REQ-12} All column widths shall be sized as shown in the image below.

	A	B	C	D	E	F	G	H	I
1	ORDER DATE	ITEM NUMBER	PRODUCT LINE	PRODUCT CODE	ITEM QUANTITY	ITEM PRICE	TOTAL PRICE	STATUS	CUSTOMER NAME
2	11	13	15	15	15	13	13	10	30

CONSTRAINTS

{CONST-1} The script shall be designed and implemented using an appropriate functional decomposition.

Note: Stress not, young grasshopper. This will be explained by the instructor.

DROPBOX SUBMISSION

Submit a link to the GitHub repository that contains your script, e.g., https://github.com/BobLoblaw/COMP593-Lab3

ASSESSMENT

Item	Out Of	Assessment Criteria
GitHub	1	• Repository contains script • .gitignore file excludes Orders directory from the repository
Functional decomposition	1	• Script designed using functional decomposition
Command line parameters	1	• Script accepts and verifies full path of sales data CSV file
Orders directory	1	• Script creates and uses sub-directory for order files • Sub-directory named "Orders_YYYY-MM-DD"
Order info extracted from CSV	1	• Order information is extracted from sales data CSV • Sorted by item number
Order info inserted	1	• Total price and grand total information inserted correctly
Formatting	4	• Money formatted as $12, 345.56 • Column widths sized correctly
Total:	**10**	

REFERENCES

- pandas - User Guide: pandas.pydata.org/docs/user_guide/index.html
- pandas - DataFrame class: pandas.pydata.org/docs/reference/api/pandas.DataFrame.html
- pandas - Merge, join, concatenate, and compare: pandas.pydata.org/docs/user_guide/merging.html
- OpenPyXL Home Page: openpyxl.readthedocs.io/en/stable/
- XlsxWrite Home Page: xlsxwriter.readthedocs.io/
- Practical Business Python - Improving Pandas Excel Output: pbpython.com/improve-pandas-excel-output.html
- Practical Business Python - Case Study: Automating Excel File Creation and Distribution with Pandas and Outlook: pbpython.com/excel-email.html