Developing a lookup function in Python for looking up Tech TDS Descriptions through Income Tax Law in Microsoft Excel for Deloitte Haskins & Sells India

Siddhant Ray

24th – 32nd floors Indiabulls Finance Centre, Tower 3 Senapati Bapat Marg, Fitwala Rd, West, Mumbai, Maharashtra 400013

----------***----------

Abstract – *This research paper focuses on developing a lookup function in Python for looking up Tech TDS Descriptions through Income Tax Law in Microsoft Excel for Deloitte Haskins & Sells India. I have developed this Python code using pandas library and have leveraged several open-source resources from the Internet to successfully come up with this research paper and implement the Python code successfully in the system of Deloitte. It took me around 2 weeks to come up with an effective code. The pandas library is a library used for data access. The lookup function simply works on copy pasting the file path of the Excel document (to be in the same folder as the Python file). This is followed by putting in the name of the column to lookup, entering the value to lookup for, and finally entering the name of the column to retrieve the data from. Here the first column is the Tech TDS Description and the retrieve column is the Income Tax act. The code is 31 lines long and is accurate and heavily tested before being deployed into the systems.*

Key Words: Lookup function, Deloitte Haskins & Sells, pandas library, lookup value, retrieve

1.INTRODUCTION

As part of my internship at Deloitte, I have developed a lookup function in Python for looking up Tech TDS Descriptions through Income Tax Law for Deloitte Haskins & Sells India. This code was really basic and I developed this code using the 'pandas' library in Python. When the code is run in the terminal, the program gives the user an option to feed in the file path of the Excel document in which the lookup function needs to be carried out. When the file path (which needs to be in the same folder as the Python code folder) is run, the excel document is traced using the 'pandas' library and moves on with the code. After the file path is successfully traced, the program asks for the lookup column name. Further, it asks for the value to lookup for and lastly the column to retrieve it from. In a nutshell, with a few changes, this simple code can be easily customised to the needs of the company/client.

2. PANDAS LIBRARY – AN OVERVIEW

The 'pandas' library is a powerful open-source data analysis and data manipulation library in Python. 'pandas' facilitates data functions and data structures needed to work with sorted and structured data with ease.

Within data loading and saving, 'pandas' can be used to read and write data from/to various file formats including CSV, Excel, SQL Databases, JSON, HTML, so on and so forth.

In data manipulation, 'pandas' is used for indexing and selecting data, merging and joining data, grouping and aggregation, and reshaping.

It is also used in time series analysis and statistical and mathematical operations.

Given the range of functionalities and applications of 'pandas' library, it is appropriate to use it for the most basic functionalities to the most advanced functionalities.

3. EXPLAINING THE CODE

```
import pandas as pd
```

We start the code by importing pandas as pd. The 'import pandas' statement is used to import the 'pandas' library. The 'as pd' is used for convenience which is essentially forming an alias for the 'pandas' library. This is used to improve readability, maintain consistency, and make it more convenient for the coder to program.

```
def lookup_excel_value(file_path, lookup_column, lookup_value, result_column):
    try:
        # Load the Excel file
        df = pd.read_excel(file_path)

        # Check if the lookup column and result column exist
        if lookup_column not in df.columns or result_column not in df.columns:
            print("Specified columns do not exist in the Excel file.")
            return
```

Syntax: def lookup_excel_value(file_path, lookup_column, lookup_value, result_column):

Description:
- Formulates a new function called '**lookup_excel_value**' using the '**def**' keyword in Python
- '**file_path**' is the path of the Excel file that is supposed to be read
- '**lookup_column**' is the name of the column to search the specific value for
- '**result_column**' is the name of the column to lookup the value from and print the corresponding value

Syntax: try:

Description:
- This initiates a try block to handle exceptions that may occur during the execution of the code inside it
- If any error occurs due to some reason, the control will pass to the corresponding except block

Syntax: df = pd.read_excel(file_path)

Description:
- This syntax reads the Excel file specified by the '**file_path**' into a pandas DataFrame
- '**df**' is the DataFrame variable that holds the data from the Excel file. The DataFrame is a two-dimensional labeled data-structure, identical to a table with rows and columns

Syntax: if lookup_column not in df.columns or result_column not in df.columns:
print("Specified columns do not exist in the Excel file.")
return

Description:
- '**df.columns**' is an attribute that returns an index object containing the column labels of the DataFrame
- '**lookup_column not in df.columns**' checks if the column specified by '**lookup_column**' exists in the DataFrame
- '**result_column not in df.columns**' checks if the column specified by '**result_column**' exists in the DataFrame
- If either of these conditions hold true (both or one of the columns do not exist), the function prints an error message stating "Specified columns do not exist in the Excel file"
- '**return**' statement exits the function early, stopping any further execution since the required columns are missing

```
row = df[df[lookup_column] == lookup_value]
```

Syntax: row = df[df[lookup_column] == lookup_value]

Description:
- '**df[lookup_column]**' is used to select the column in the DataFrame '**df**' with the name '**lookup_column**'. This returns a panda series containing the values of the specified columns
- 'df[lookup_column] == lookup_value' creates a boolean series comparing each value in the '**lookup_column**' with '**lookup_value**'
- The result is a series of boolean values ('**True**' or '**False**') indicating whether each value in the '**lookup_column**' equals the '**lookup_value**'

```
    if not row.empty:
        # Print the corresponding value from the result column
        print(f"The value in column '{result_column}' for '{lookup_value}'
              in column '{lookup_column}' is: {row[result_column].values[0]}")
    else:
        print(f"Value '{lookup_value}' not found in column '{lookup_column}'.")

except Exception as e:
    print(f"An error occurred: {e}")
```

Syntax: if not row.empty:
Print the corresponding value from the result column
print(f"The value in column '{result_column}' for '{lookup_value}'
in column '{lookup_column}' is: {row[result_column].values[0]}")
else:
print(f"Value '{lookup_value}' not found in column '{lookup_column}'.")

except Exception as e:
print(f"An error occurred: {e}")

Description:
- '**if not row.empty**' checks if the filtered DataFrame '**row**' is not empty
- '**row.empty**' is a Boolean attribute that returns '**True**' if the DataFrame is empty and '**False**' otherwise
- '**if not row.empty**' makes sure whether the DataFrame '**row**' indicating that a lookup value was found
- If the DataFrame is not empty, the print statement prints out a formatted string that includes the lookup value, the column it was found in, and the corresponding value in each column
- If this condition does not hold true, the '**print**' statement outputs a message indicating that the lookup value was not found in the specified column
- The '**except**' block catches any exceptions that might occur during the execution of the '**try**' block
- The '**print**' statement outputs a message indicating that an error occurred, along with the exception message ('**e**'), which provides information about what went wrong

```
if __name__ == "__main__":
    file_path = input("Enter the file path for the Excel file: ").strip('"')
    lookup_column = input("Enter the name of the column to look up: ")
    lookup_value = input(f"Enter the value to search for in column '{lookup_column}': ")
    result_column = input("Enter the name of the column to retrieve the value from: ")

    lookup_excel_value(file_path, lookup_column, lookup_value, result_column)
```

Syntax: if __name__ == "__main__":
file_path = input("Enter the file path for the Excel file: ").strip('"')
lookup_column = input("Enter the name of the column to look up: ")
lookup_value = input(f"Enter the value to search for in column '{lookup_column}': ")
result_column = input("Enter the name of the column to retrieve the value from: ")

lookup_excel_value(file_path, lookup_column, lookup_value, result_column)

Description:

- The first line checks if the script is being run directly (not being imported as a module in another script)
- The next line is an input function enabling the user to input the file path as an URL (Uniform Resource Locator)
- The user needs to make sure the Python code and the Excel file are in the same folder
- The '**.strip()**' removes any double quotes from the beginning and the end of the file path. This is useful for avoiding errors
- The next line is another input function specifically used for inputting the column name (Tech TDS Description titled column) to lookup in the Excel document
- The following line is another input function used to input the value the user is searching for (Tech TDS Description in the case of Deloitte Haskins & Sells)
- The succeeding line is another input function used to input the name of the column to retrieve the value from (Income Tax Act in the case of Deloitte Haskins & Sells)
- The final calls the '**lookup_excel_value()**' function to perform the lookup and prints the corresponding value or error message in the terminal

4. CONCLUSION

To conclude, the working of the lookup Tech TDS function against the Income tax law column is a really easy and effective Python code to develop. This Python code can be easily integrated with Microsoft Excel to make data manipulation and data access much simpler.

ACKNOWLEDGEMENT (Optional)

I would like to acknowledge Deloitte Haskins & Sells India to provide me with the suitable technology to develop this efficient Python code and come up with this research paper.

REFERENCES

[1] https://www.w3schools.com/python/pandas/pandas_intro.asp#:~:text=What%20is%20Pandas%3F,by%20Wes%20McKinney%20in%202008.

[2] https://mode.com/python-tutorial/libraries/pandas/

[3] https://moez-62905.medium.com/the-hidden-gems-of-pandas-5-lesser-known-functions-you-need-to-try-9fd20727e4c

BIOGRAPHIES

Siddhant Ray is a 16-year-old technogeek and polymath from Mumbai, excelling in fields like mathematics and physics. A student at Jamnabai Narsee International School, he is passionate about coding, AI projects, and optimizing designs using complex theories. Siddhant is also actively engaged in creating social media content.