Lesson 8 Supplement: Excel and Multiple Optimal Solutions

1 Today...

- read and write to Excel from Python using pandas
- test for multiple optimal soluitons

2 Main lesson

The main lesson for today is in a Jupyter notebook. We will just collect some useful ideas here for easy access.

3 pandas: commonly used commands

- Import pandas:

```
import pandas as pd
```

- Import numpy:

```
import numpy as np
```

- Open an existing Excel file called "FileName.xlsx":

```
df = pd.read_excel ('FileName.xlsx', sheetname=0, index_col=0)
```

Note: This only works if the Python file (Jupyter notebook) and the Excel file are in the same directory. Otherwise, we need the full path to the Excel file in quotes. Quote: By default, pandas will automatically assign a numeric index or row label starting with zero. You may want to leave the default index as such if your data doesn't have a column with unique values that can serve as a better index. In case there is a column that you feel would serve as a better index, you can override the default behavior by setting index_col property to a column. It takes a numeric value for setting a single column as index or a list of numeric values for creating a multi-index.

- Open a specific sheet:

```
df = pd.read_excel(r'File_name.xlsx', sheet_name='Sheet1')
```

- Importing subset of columns:

```
data = pd.read_excel ('FileName.xlsx')
df = pd.DataFrame(data, columns= ['ColumnName'])
```

Since all the three sheets have similar data but for different recordsmovies, we will create a single DataFrame from all the three DataFrames we created above. We will use the pandas concat method for this and pass in the names of the three DataFrames we just created and assign the results to a new DataFrame object, movies. By keeping the DataFrame name same as before, we are over-writing the previously created DataFrame.
movies = pd.concat([movies_sheet1, movies_sheet2, movies_sheet3])

We can also use the ExcelFile class to work with multiple sheets from the same Excel file. We first wrap the Excel file using ExcelFile and then pass it to read_excel method.
xlsx = pd.ExcelFile(excel_file) movies_sheets = [] for sheet in xlsx.sheet_names: movies_sheets.append(xlsx.parse(sheet)) movies = pd.concat(movie_sheets)

movies["Net Earnings"] = movies["Gross Earnings"] - movies["Budget"]

Exporting the results to Excel If you're going to be working with colleagues who use Excel, saving Excel files out of pandas is important. You can export or write a pandas DataFrame to an Excel file using pandas to_excel method. Pandas uses the xlwt Python module internally for writing to Excel files. The to_excel method is called on the DataFrame we want to export.We also need to pass a filename to which this DataFrame will be written.
movies.to_excel('output.xlsx') By default, the index is also saved to the output file. However, sometimes the index doesn't provide any useful information. For example, the movies DataFrame has a numeric auto-increment index, that was not part of the original Excel data.
movies.head()

- Read the data in from cells A1 to D15 in sheet "sh":

```
my_data = sh.range('A1:D15').value
```

Note: `my_data` will be a list of 15 lists (rows 1-15), each with 4 elements (columns A-D).

- Write a list (of lists) called 'my_double_list' beginning at cell A1 and forming a rectangle of the same dimensions:

```
sh.range('A1').value = my_double_list
```

4 Other Python reminders

- We can use a `range()` object in `for` loops. For example,

```
for i in range(3):
```

loops over $i = 0, 1, 2$.

- Recall that list indexing always starts at 0.

- To index over the list of lists,

  ```
  mylist = [[a,b,c],[d,e,f]],
  ```

  the item at position

  ```
  mylist[0][1]
  ```

  is b (list 0, position 1).

5 Multiple Optimal Solutions

Sometimes we might want to explore different optimal solutions to the same integer or linear program. This means, a solution that has the same objective value, but different values for the variables. In order to accomplish this, we can take the following steps:

1. Solve the model the first time
2. Add a constraint to the model that prevents the same optimal solution from occuring
   - We have to be careful not to make the constraint too restrictive, or it may prevent other optimal solutions from occuring.
3. Solve the model again and see what happens
   - If we get another optimal solution with the same objective value, then the model has multiple optimal solutions. We can add yet another constraint to check for a third optimal solution, etc.
   - If we get infeasible or an optimal solution with a worse objective value, then there was only the one optimal solution.
   - Is it possible to add a constraint to the model and get a better better optimal objective value?