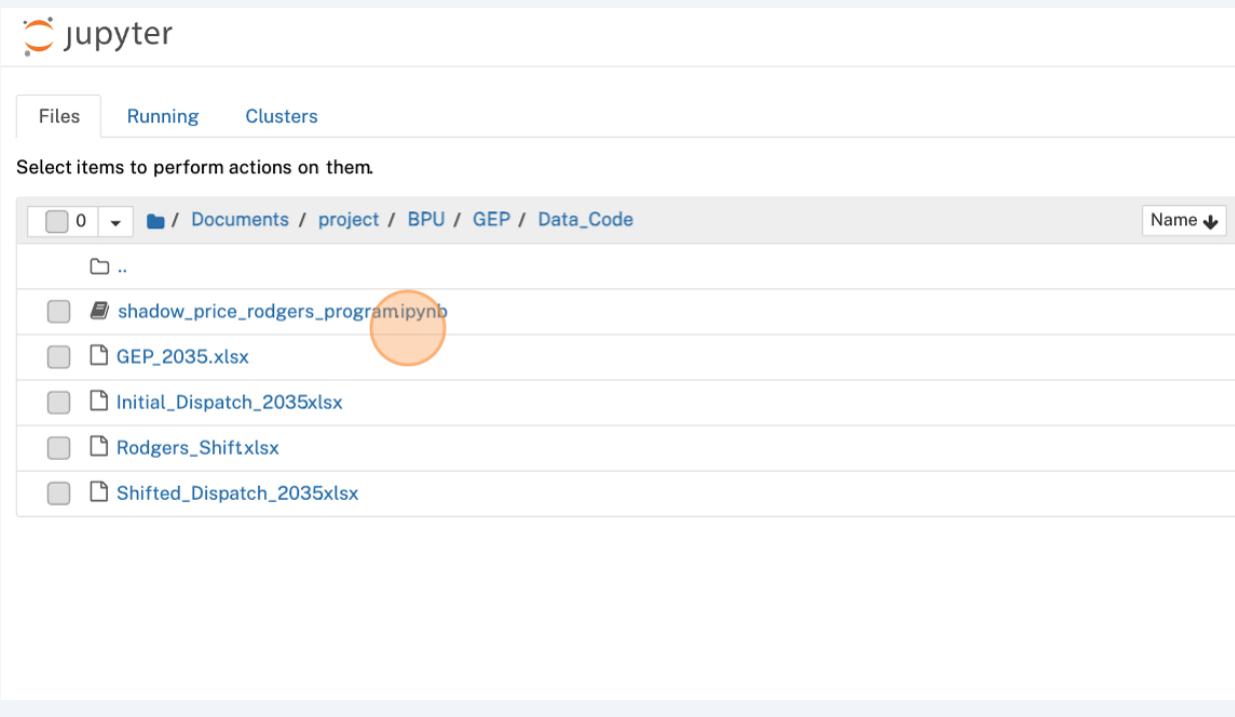


Running the GEP Data Code in Jupyter Notebook

- 1 Click "shadow_price_rodgers_program.ipynb"



2 Click here.

The screenshot shows a Jupyter Notebook interface with a toolbar at the top. Below the toolbar, there's a section titled "Code chapters:" containing two numbered sections: "1) GEP" and "2) Initial Dispatch". Each section has associated steps or sub-sections listed below it. An orange circle highlights the "1) GEP" section.

Code chapters:

1) GEP

- initialize
- Create and Run GEP Model
- Store and Plot Results

2) Initial Dispatch

- Initialize

3 Click "1) GEP"¶

The screenshot shows a Jupyter Notebook cell with the title "4) Dispatch with Reallocation of Load". Below the title, there are three buttons: "Shifted Demand Plotter", "Shifted Demand Dispatcher Master Function", and "Run to Show with Unmet Demand Shifted". The cell content starts with the title "1) GEP" followed by an orange circle. Below this, there is a section titled "Initialize" with two code snippets. The first snippet is in red text and defines file names for GEP, initial dispatch, and shifted dispatch. The second snippet is in green text and contains several "#" symbols.

4) Dispatch with Reallocation of Load

Shifted Demand Plotter

Shifted Demand Dispatcher Master Function

Run to Show with Unmet Demand Shifted

1) GEP ¶

Initialize

```
In [30]: GEP_xl_name = './GEP_2035.xlsx'  
initial_dispatch_xl_name = './Initial_Dispatch_2035.xlsx'  
shifted_dispatch_xl_name = './Shifted_Dispatch_2035.xlsx'  
  
In [2]: ##### GEP GEP GEP GEP GEP GEP  
#####  
##
```

4 Click "import"

```
##  
#####  
  
# !apt-get install -y -qq glpk-utils #for the GLPK solver in pulp, used to mak  
# !pip install pulp  
# !pip install pandas openpyxl  
# !pip install xlsxwriter  
  
#####  
##  
## Import initial libraries. Not all are needed.  
##  
#####  
  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
from math import *  
import time  
import random  
from numpy import linalg as LA  
import pickle  
import warnings  
warnings.filterwarnings("ignore", category=UserWarning)  
import scipy  
from scipy import optimize as opt  
import csv  
import os  
import math
```

5 Click "Run"

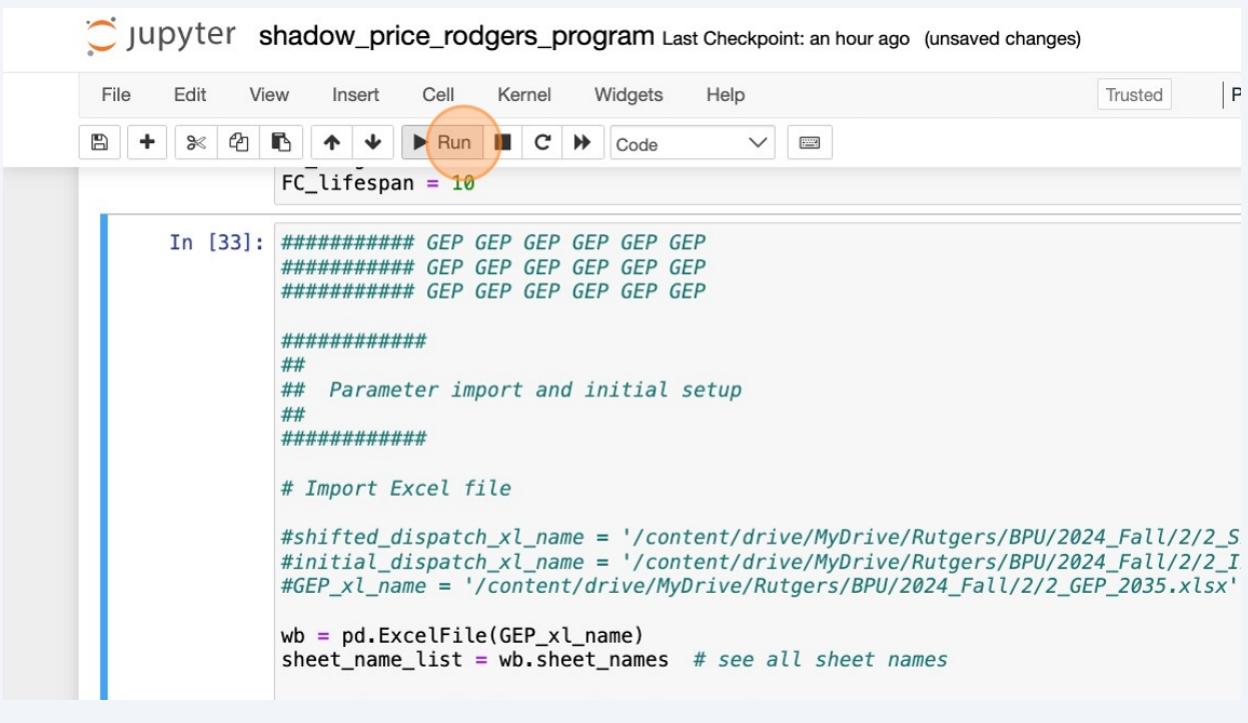
jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted P

Run

```
##  
##  
## pip installs  
##  
#####  
  
# !apt-get install -y -qq glpk-utils #for the GLPK solver in pulp, used to mak  
# !pip install pulp  
# !pip install pandas openpyxl  
# !pip install xlsxwriter  
  
#####  
##  
## Import initial libraries. Not all are needed.  
##  
#####  
  
import numpy as np  
import pandas as pd  
import matplotlib.pyplot as plt  
from math import *
```

6 Click "Run"



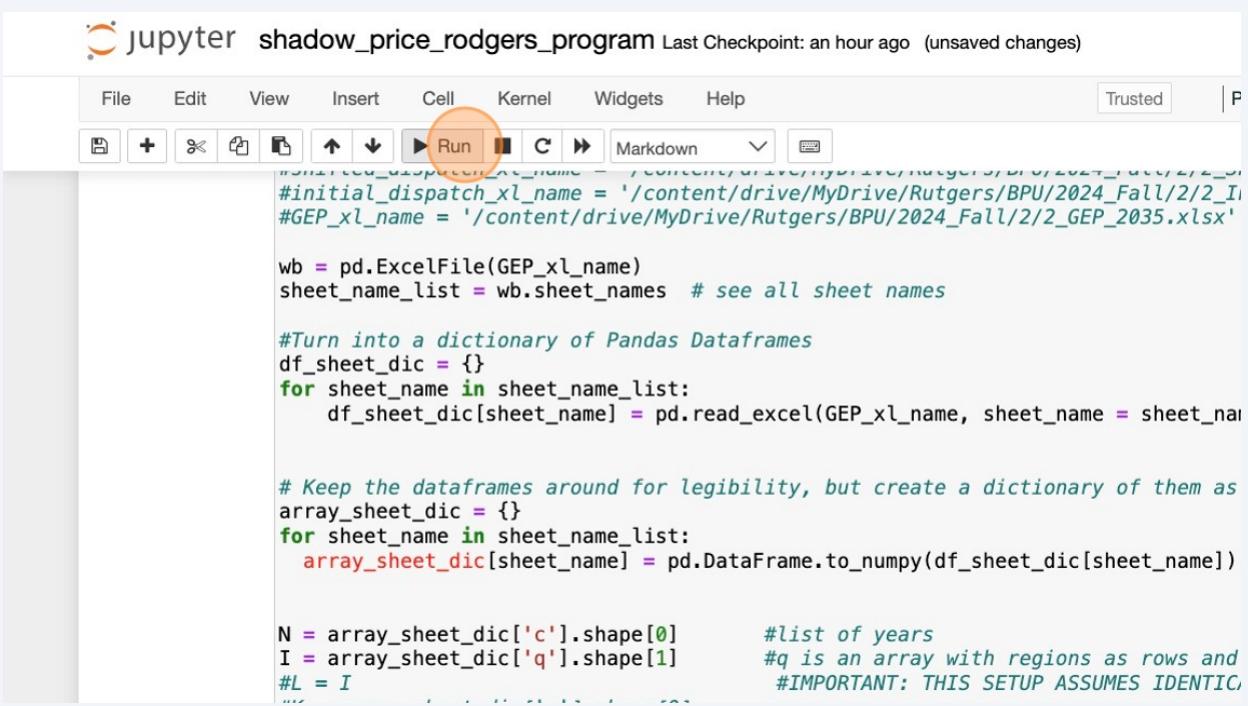
jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted

Run

```
In [33]: ##### GEP GEP GEP GEP GEP GEP  
##### GEP GEP GEP GEP GEP  
##### GEP GEP GEP GEP GEP  
  
#####  
## Parameter import and initial setup  
##  
#####  
  
# Import Excel file  
  
#shifted_dispatch_xl_name = '/content/drive/MyDrive/Rutgers/BPU/2024_Fall/2/2_S  
#initial_dispatch_xl_name = '/content/drive/MyDrive/Rutgers/BPU/2024_Fall/2/2_I  
#GEP_xl_name = '/content/drive/MyDrive/Rutgers/BPU/2024_Fall/2/2_GEP_2035.xlsx'  
  
wb = pd.ExcelFile(GEP_xl_name)  
sheet_name_list = wb.sheet_names # see all sheet names
```

7 Click "Run"



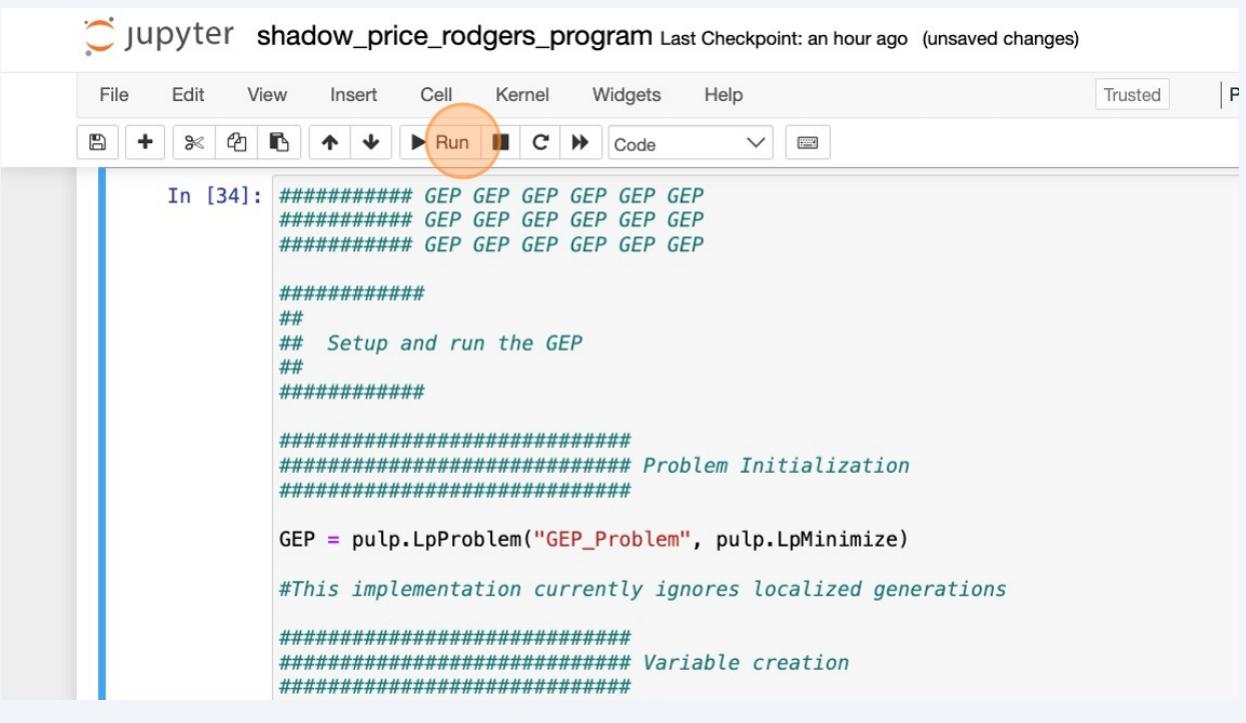
jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted

Run

```
#shifted_dispatch_xl_name = '/content/drive/MyDrive/Rutgers/BPU/2024_Fall/2/2_S  
#initial_dispatch_xl_name = '/content/drive/MyDrive/Rutgers/BPU/2024_Fall/2/2_I  
#GEP_xl_name = '/content/drive/MyDrive/Rutgers/BPU/2024_Fall/2/2_GEP_2035.xlsx'  
  
wb = pd.ExcelFile(GEP_xl_name)  
sheet_name_list = wb.sheet_names # see all sheet names  
  
#Turn into a dictionary of Pandas Dataframes  
df_sheet_dic = {}  
for sheet_name in sheet_name_list:  
    df_sheet_dic[sheet_name] = pd.read_excel(GEP_xl_name, sheet_name = sheet_name)  
  
# Keep the dataframes around for legibility, but create a dictionary of them as  
array_sheet_dic = {}  
for sheet_name in sheet_name_list:  
    array_sheet_dic[sheet_name] = pd.DataFrame.to_numpy(df_sheet_dic[sheet_name])  
  
N = array_sheet_dic['c'].shape[0] #list of years  
I = array_sheet_dic['q'].shape[1] #q is an array with regions as rows and  
#L = I #IMPORTANT: THIS SETUP ASSUMES IDENTIC
```

8 Click "Run"



jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted P

In [34]:

```
##### GEP GEP GEP GEP GEP GEP
#####
##### GEP GEP GEP GEP GEP GEP
#####
##### GEP GEP GEP GEP GEP GEP

#####
## 
## Setup and run the GEP
##
#####

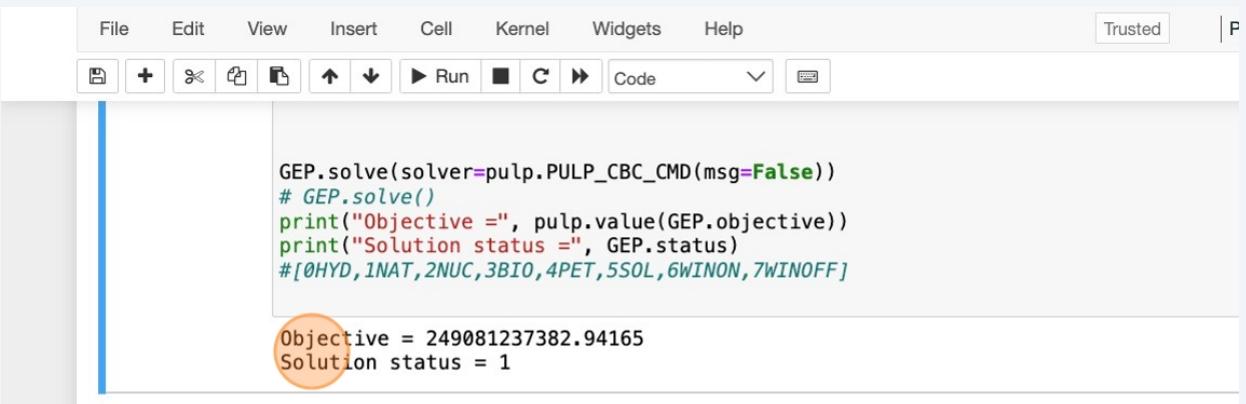
#####
##### Problem Initialization
#####

GEP = pulp.LpProblem("GEP_Problem", pulp.LpMinimize)

#This implementation currently ignores localized generations

#####
##### Variable creation
#####
```

9 Click "Objective = 249081237382.94165
Solution status = 1"



File Edit View Insert Cell Kernel Widgets Help Trusted P

In [34]:

```
GEP.solve(solver=pulp.PULP_CBC_CMD(msg=False))
# GEP.solve()
print("Objective =", pulp.value(GEP.objective))
print("Solution status =", GEP.status)
#[0HYD,1NAT,2NUC,3BIO,4PET,5SOL,6WINON,7WINOFF]
```

Objective = 249081237382.94165
Solution status = 1

Store and Plot Results

In [35]:

```
# obtain results from solved GEP PuLP variables

x_mat = np.zeros((N,I))
for n in range(N):
    for i in range(I):
        x_mat[n][i] = x[n][i].varValue

w_mat = np.zeros((N,I))
```

10 Click "Store and Plot Results"

```
In [35]: # obtain results from solved GEP PuLP variables

x_mat = np.zeros((N,I))
for n in range(N):
    for i in range(I):
        x_mat[n][i] = x[n][i].varValue

w_mat = np.zeros((N,I))
for n in range(N):
    for i in range(I):
        w_mat[n][i] = w[n][i].varValue

y_mat = np.zeros((N))
for n in range(N):
    y_mat[n] = y[n].varValue

#investment_mat = np.zeros((N,I+1))
```

11 Click here.

Store and Plot Results

```
In [35]: # obtain results from solved GEP PuLP variables

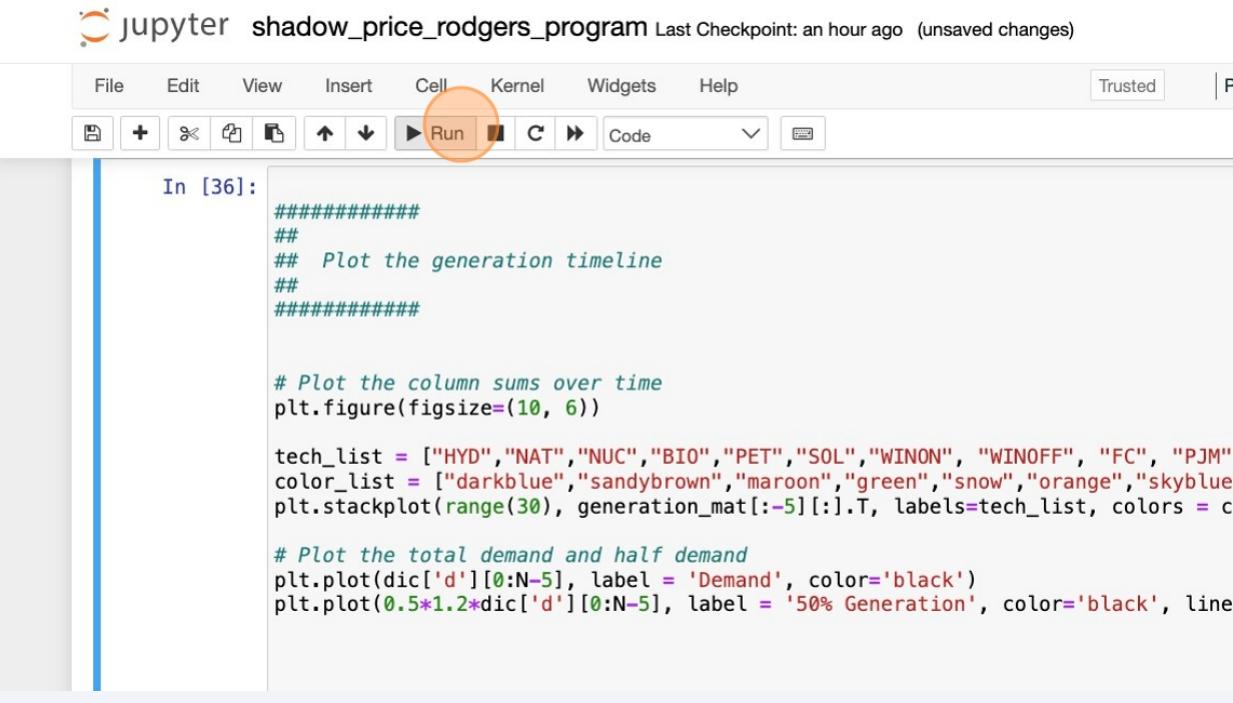
x_mat = np.zeros((N,I))
for n in range(N):
    for i in range(I):
        x_mat[n][i] = x[n][i].varValue

w_mat = np.zeros((N,I))
for n in range(N):
    for i in range(I):
        w_mat[n][i] = w[n][i].varValue

y_mat = np.zeros((N))
for n in range(N):
    y_mat[n] = y[n].varValue

#investment_mat = np.zeros((N,I+1))
```

12 Click "Run"



In [36]:

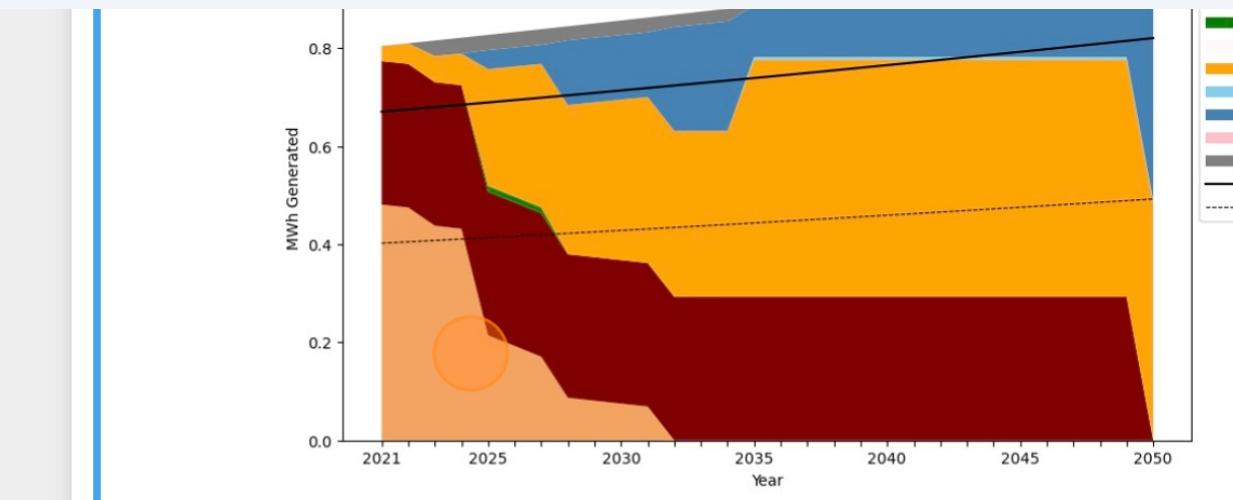
```
#####
##  Plot the generation timeline
##
#####

# Plot the column sums over time
plt.figure(figsize=(10, 6))

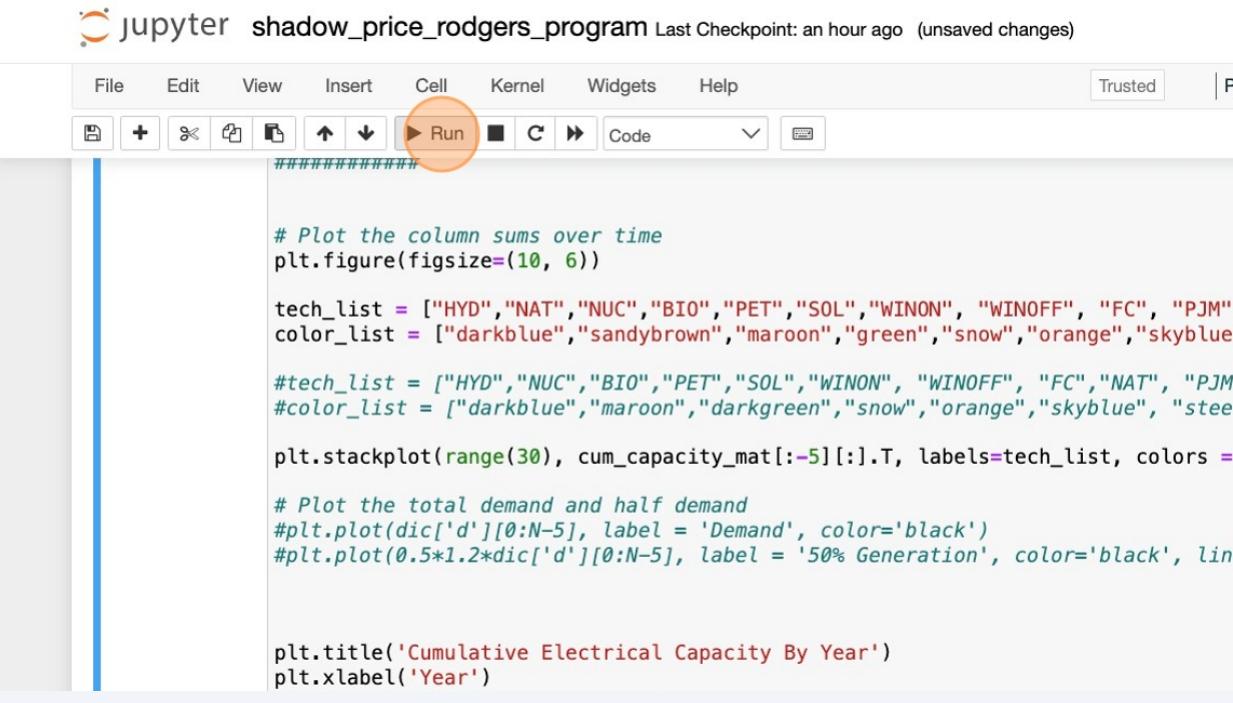
tech_list = ["HYD", "NAT", "NUC", "BIO", "PET", "SOL", "WINON", "WINOFF", "FC", "PJM"
color_list = ["darkblue", "sandybrown", "maroon", "green", "snow", "orange", "skyblue"
plt.stackplot(range(30), generation_mat[:-5][:].T, labels=tech_list, colors = c

# Plot the total demand and half demand
plt.plot(dic['d'][0:N-5], label = 'Demand', color='black')
plt.plot(0.5*1.2*dic['d'][0:N-5], label = '50% Generation', color='black', line
```

13 Click this image.



14 Click "Run"



```
# Plot the column sums over time
plt.figure(figsize=(10, 6))

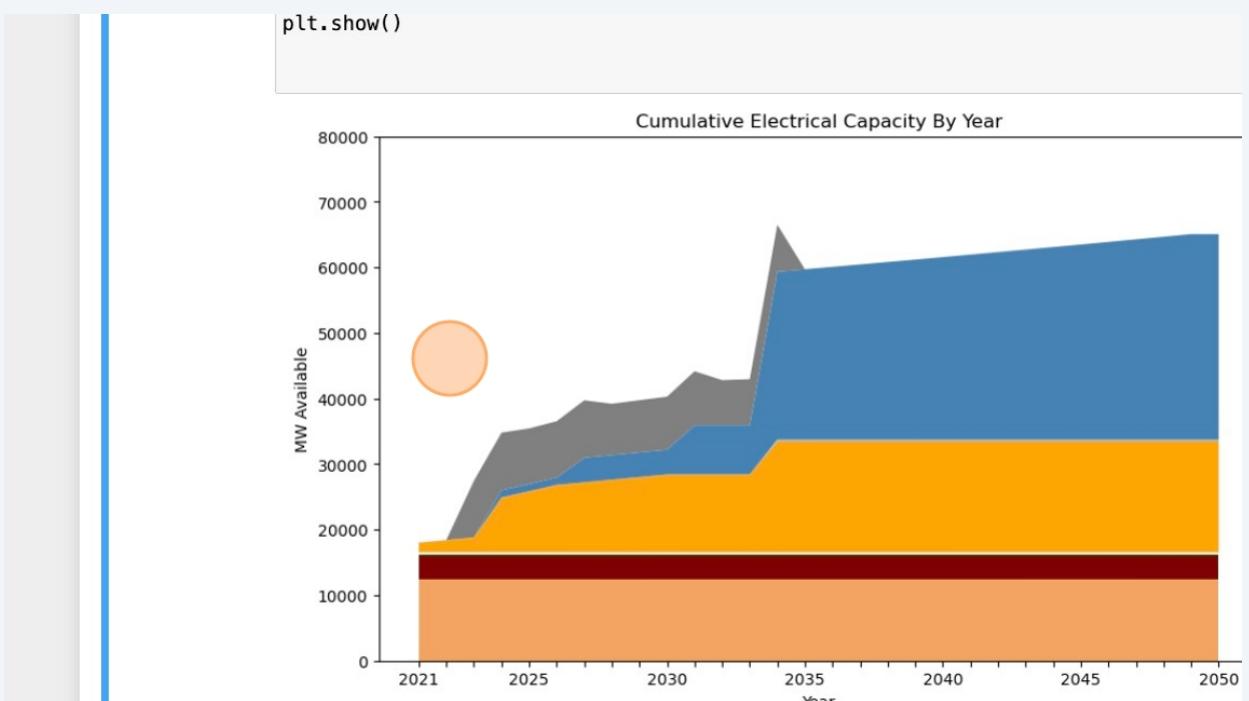
tech_list = ["HYD", "NAT", "NUC", "BIO", "PET", "SOL", "WINON", "WINOFF", "FC", "PJM"
color_list = ["darkblue", "sandybrown", "maroon", "green", "snow", "orange", "skyblue"]

#tech_list = ["HYD", "NUC", "BIO", "PET", "SOL", "WINON", "WINOFF", "FC", "NAT", "PJM"
#color_list = ["darkblue", "maroon", "darkgreen", "snow", "orange", "skyblue", "steelblue"]

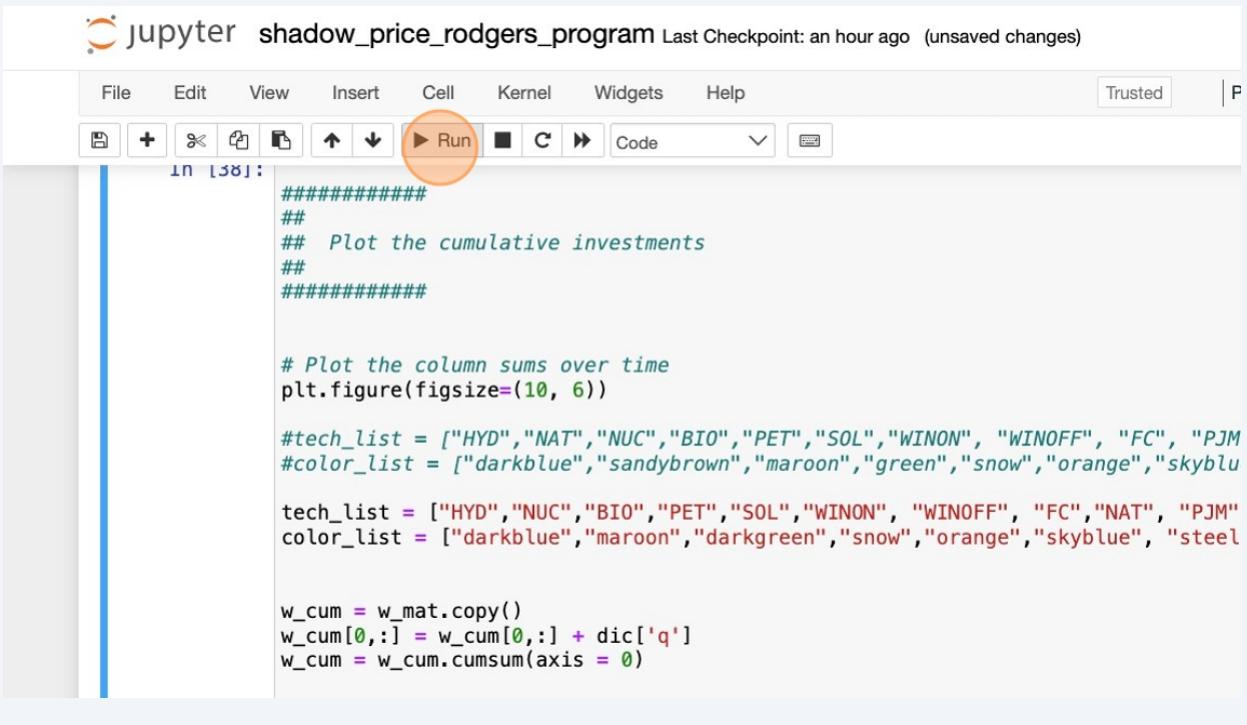
plt.stackplot(range(30), cum_capacity_mat[:-5][:].T, labels=tech_list, colors =
# Plot the total demand and half demand
#plt.plot(dic['d'][0:N-5], label = 'Demand', color='black')
#plt.plot(0.5*1.2*dic['d'][0:N-5], label = '50% Generation', color='black', linestyles=[(0, (5, 5))])

plt.title('Cumulative Electrical Capacity By Year')
plt.xlabel('Year')
```

15 Click this image.



16 Click "Run"



```
In [58]: #####
## Plot the cumulative investments
##
#####



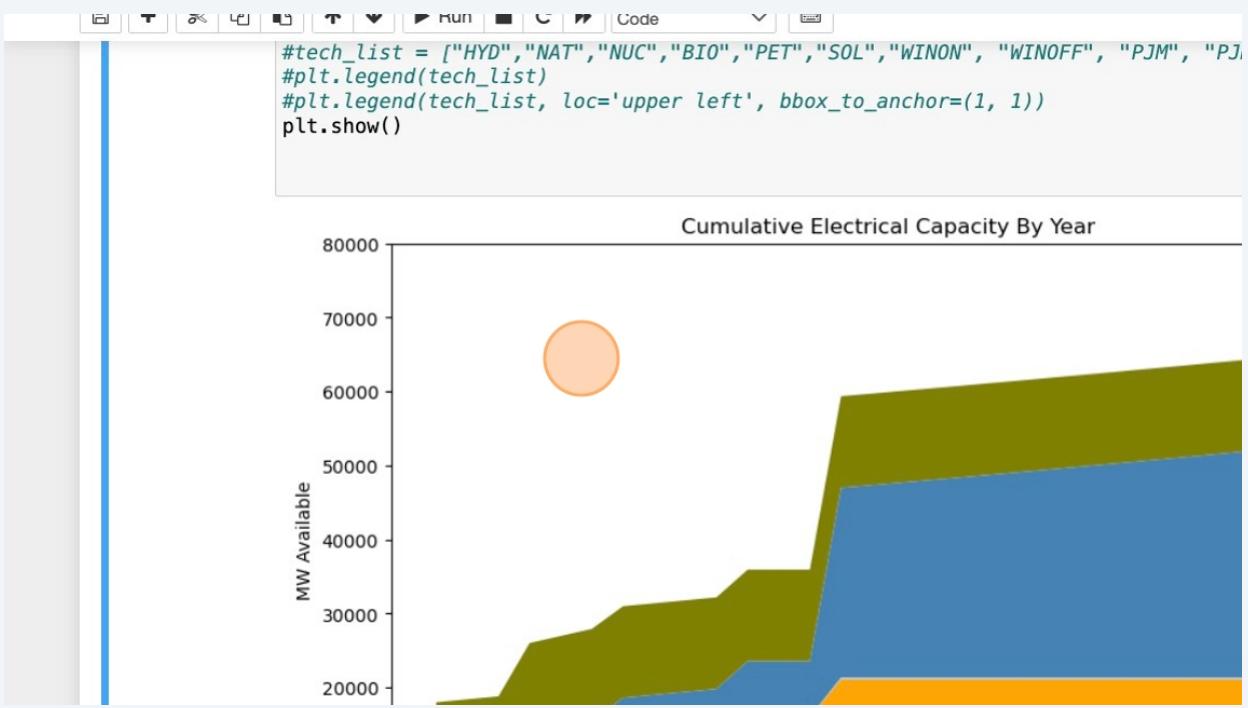
# Plot the column sums over time
plt.figure(figsize=(10, 6))

#tech_list = ["HYD","NAT","NUC","BIO","PET","SOL","WINON", "WINOFF", "FC", "PJM"
#color_list = ["darkblue","sandybrown","maroon","green","snow","orange","skyblue"]

tech_list = ["HYD","NUC","BIO","PET","SOL","WINON", "WINOFF", "FC","NAT", "PJM",
color_list = ["darkblue","maroon","darkgreen","snow","orange","skyblue", "steel"

w_cum = w_mat.copy()
w_cum[0,:] = w_cum[0,:] + dic['q']
w_cum = w_cum.cumsum(axis = 0)
```

17 Click this image.



18 Click "Run"

jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted P

In [39]:

```
#####
## Plot the cumulative investments
####

# Plot the column sums over time
plt.figure(figsize=(6, 6))

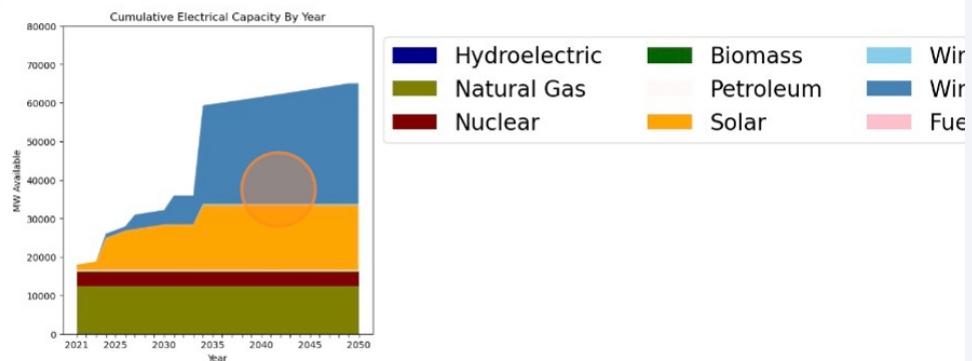
#tech_list = ["HYD", "NAT", "NUC", "BIO", "PET", "SOL", "WINON", "WINOFF", "FC", "PJM"
#color_list = ["darkblue", "sandybrown", "maroon", "green", "snow", "orange", "skyblue"]

tech_list = ["HYD", "NAT", "NUC", "BIO", "PET", "SOL", "WINON", "WINOFF", "FC", "NAT"
color_list = ["darkblue", "olive", "maroon", "darkgreen", "snow", "orange", "skyblue"]

#w_cum = w_mat.copy()
#w_cum[0, :] = w_cum[0, :] + dic['q']
#w_cum = w_cum.cumsum(axis = 0)
```

19 Click this image.

```
tech_list = ["Hydroelectric", "Natural Gas", "Nuclear", "Biomass", "Petroleum", "Solar"]
plt.legend(tech_list)
plt.legend(tech_list, loc='upper left', bbox_to_anchor=(1, 1), ncol = 3, prop =
plt.show()
```

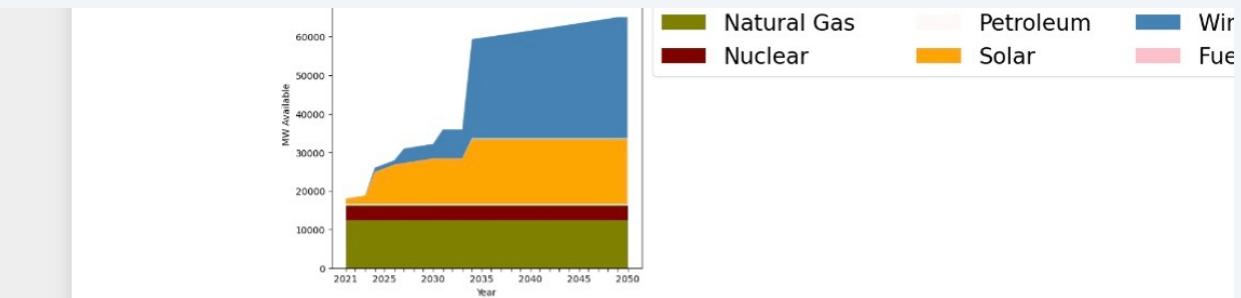


In [40]: `sum(dic['alpha'][n][i]*w[n][i].varValue for i in range(I) for n in range(N))`

Out[40]: `130595512345.50388`

In [41]: `# Obtain and store proportion of demand recommended to be met by external purch`

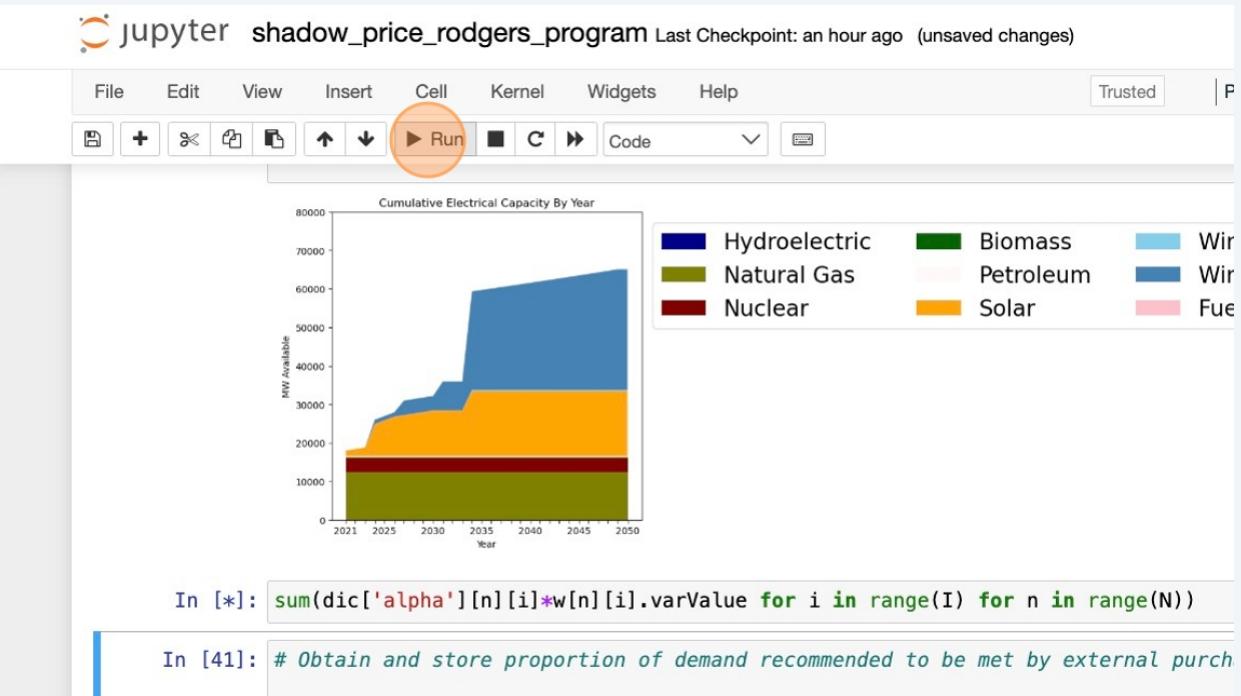
20 Click "sum"



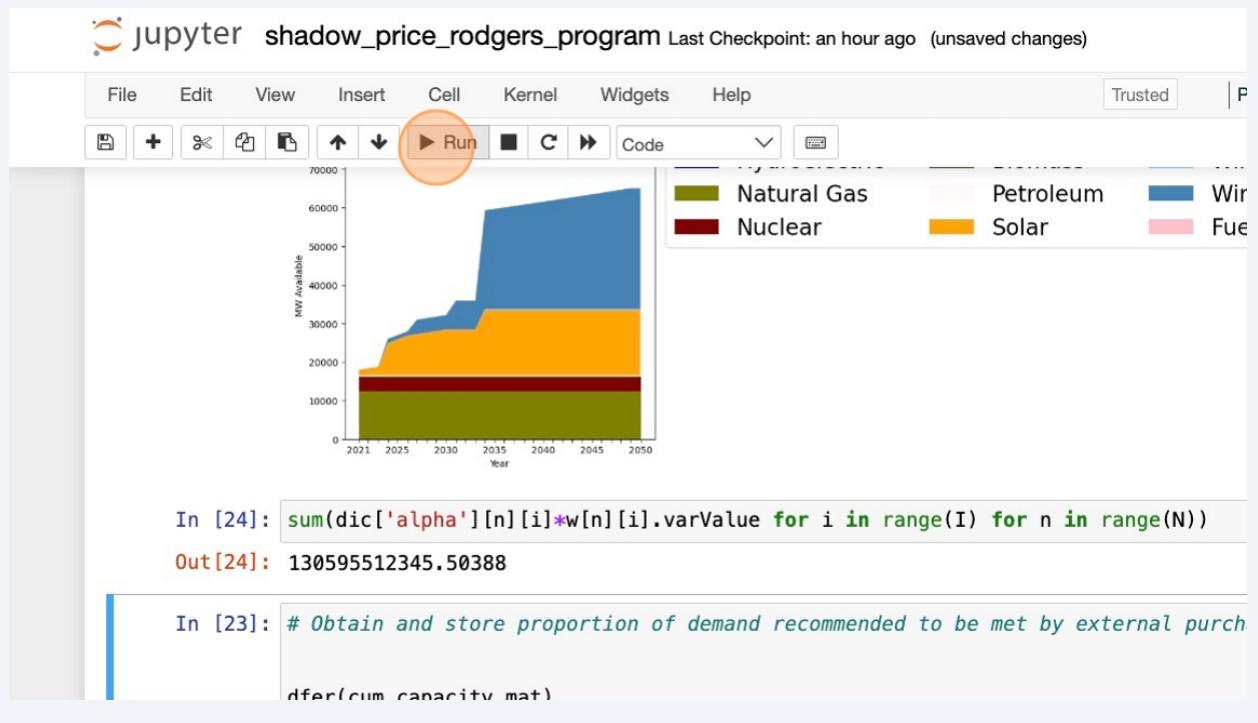
```
In [40]: sum(dic['alpha'][n][i]*w[n][i].varValue for i in range(I) for n in range(N))  
Out[40]: 130595512345.50388
```

```
In [41]: # Obtain and store proportion of demand recommended to be met by external purch  
  
dfer(cum_capacity_mat)  
#cum_capacity_mat[0][:].sum()  
  
pd.set_option('display.float_format', lambda x: '%.3f' % x)  
purchase_capacity_proportion = [cum_capacity_mat[index][-1]/cum_capacity_mat[in  
purchase_generation_proportion = [generation_mat[index][-1]/generation_mat[inde  
data = {'generation': purchase_generation_proportion, 'capacity': purchase_capa  
dfer(data) T
```

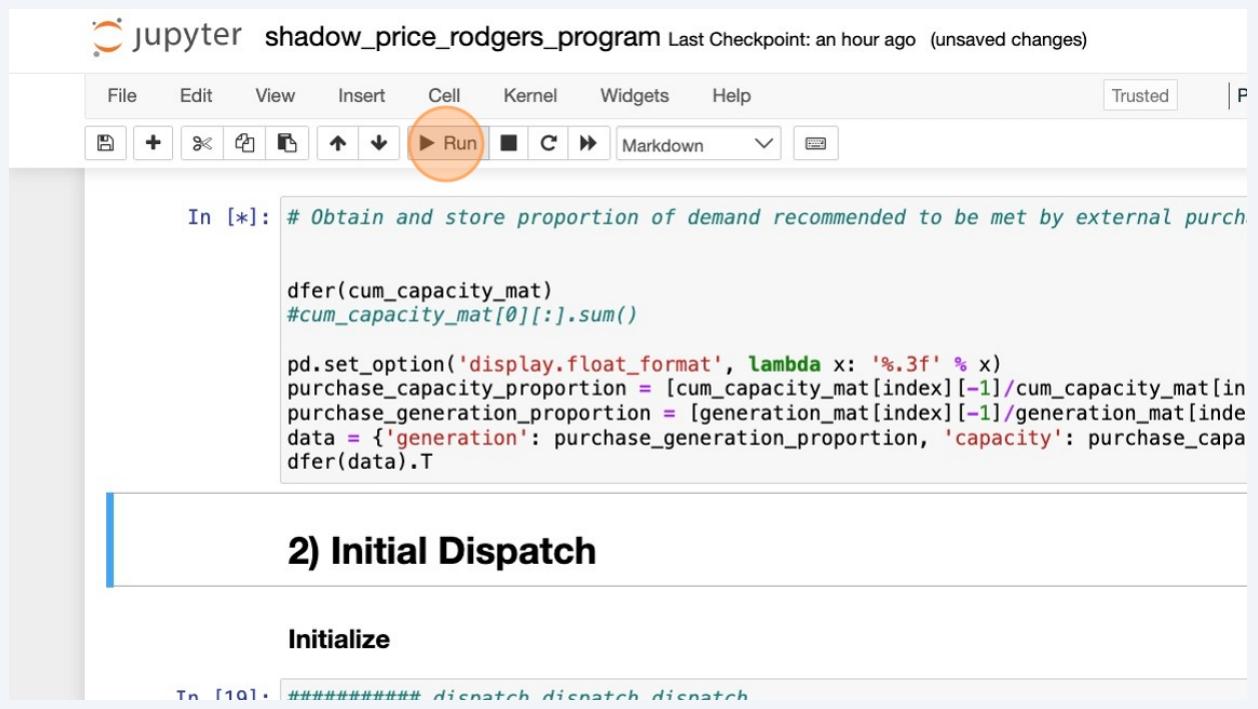
21 Click "Run"



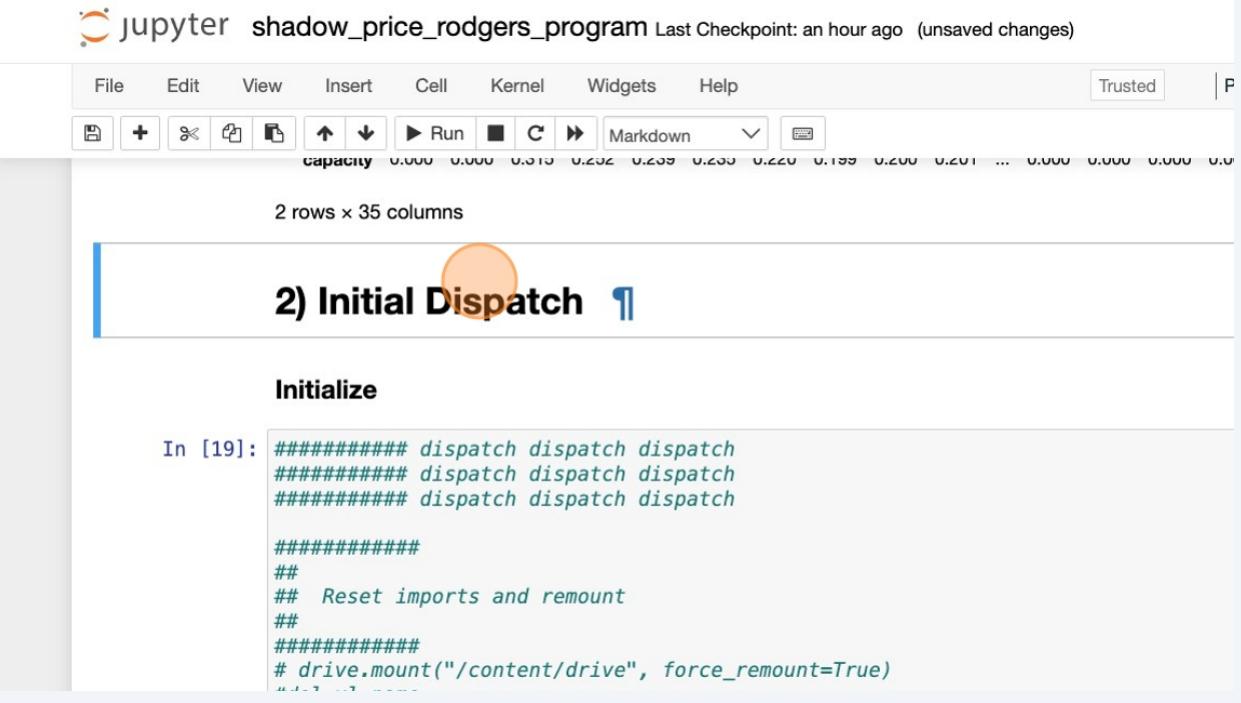
22 Click "Run"



23 Click "Run"



24 Click "2) Initial Dispatch"



jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted P

capacity 0.000 0.000 0.015 0.252 0.259 0.255 0.220 0.199 0.200 0.201 ... 0.000 0.000 0.000

2 rows × 35 columns

2) Initial Dispatch ¶

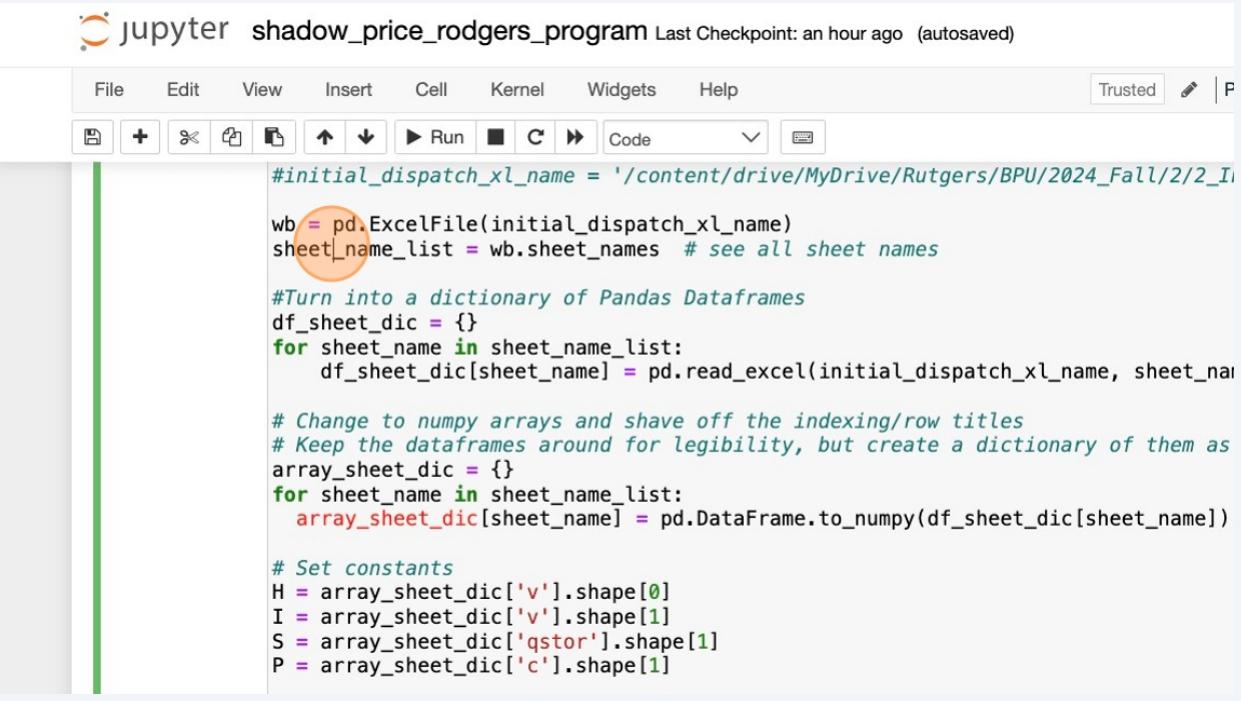
Initialize

In [19]:

```
##### dispatch dispatch dispatch
##### dispatch dispatch dispatch
##### dispatch dispatch dispatch

#####
## Reset imports and remount
##
#####
# drive.mount("/content/drive", force_remount=True)
.....
```

25 Click "sheet_name_list"



jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted P

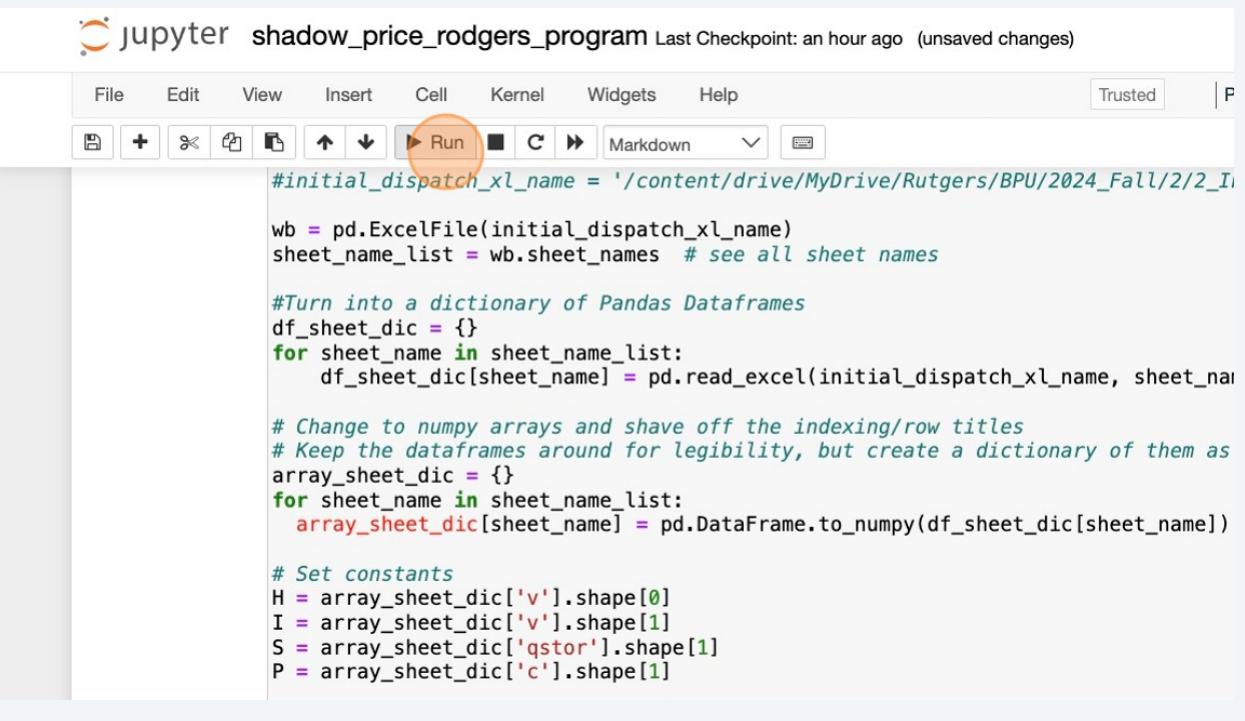
#initial_dispatch_xl_name = '/content/drive/MyDrive/Rutgers/BPU/2024_Fall/2/2_I'
wb = pd.ExcelFile(initial_dispatch_xl_name)
sheet_name_list = wb.sheet_names # see all sheet names

#Turn into a dictionary of Pandas Dataframes
df_sheet_dic = {}
for sheet_name in sheet_name_list:
 df_sheet_dic[sheet_name] = pd.read_excel(initial_dispatch_xl_name, sheet_name)

Change to numpy arrays and shave off the indexing/row titles
Keep the dataframes around for legibility, but create a dictionary of them as
array_sheet_dic = {}
for sheet_name in sheet_name_list:
 array_sheet_dic[sheet_name] = pd.DataFrame.to_numpy(df_sheet_dic[sheet_name])

Set constants
H = array_sheet_dic['v'].shape[0]
I = array_sheet_dic['v'].shape[1]
S = array_sheet_dic['qstor'].shape[1]
P = array_sheet_dic['c'].shape[1]

26 Click "Run"



jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted P

Run

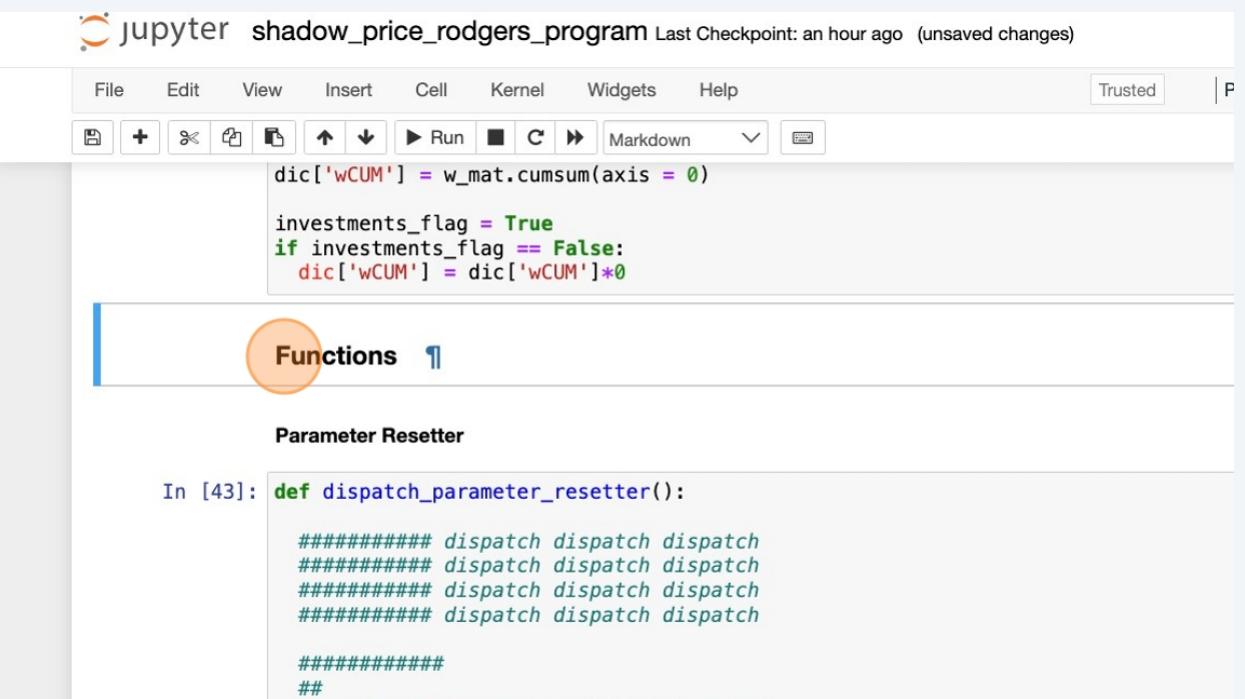
```
#initial_dispatch_xl_name = '/content/drive/MyDrive/Rutgers/BPU/2024_Fall/2/2_I
wb = pd.ExcelFile(initial_dispatch_xl_name)
sheet_name_list = wb.sheet_names # see all sheet names

#Turn into a dictionary of Pandas Dataframes
df_sheet_dic = {}
for sheet_name in sheet_name_list:
    df_sheet_dic[sheet_name] = pd.read_excel(initial_dispatch_xl_name, sheet_na

# Change to numpy arrays and shave off the indexing/row titles
# Keep the dataframes around for legibility, but create a dictionary of them as
array_sheet_dic = {}
for sheet_name in sheet_name_list:
    array_sheet_dic[sheet_name] = pd.DataFrame.to_numpy(df_sheet_dic[sheet_name])

# Set constants
H = array_sheet_dic['v'].shape[0]
I = array_sheet_dic['v'].shape[1]
S = array_sheet_dic['qstor'].shape[1]
P = array_sheet_dic['c'].shape[1]
```

27 Click "Functions¶"



jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted P

Run

```
dic['wCUM'] = w_mat.cumsum(axis = 0)

investments_flag = True
if investments_flag == False:
    dic['wCUM'] = dic['wCUM']*0
```

Functions ¶

Parameter Resetter

In [43]: `def dispatch_parameter_resetter():

 ##### dispatch dispatch dispatch
 ##### dispatch dispatch dispatch
 ##### dispatch dispatch dispatch
 ##### dispatch dispatch dispatch

 #####
 ##`

28 Click "def"

```
dic['wCUM'] = w_mat.cumsum(axis = 0)

investments_flag = True
if investments_flag == False:
    dic['wCUM'] = dic['wCUM']*0
```

Functions

Parameter Resetter

```
In [43]: def dispatch_parameter_resetter():

#####
##### dispatch dispatch
#####
##### dispatch dispatch dispatch
#####
##### dispatch dispatch dispatch
#####
##### dispatch dispatch dispatch

#####
##
## Parameter import and initial setup
##
#####

# drive.mount("/content/drive", force_remount=True)
```

29 Click "Run"

jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

```
File Edit View Insert Cell Kernel Widgets Help Trusted P
 Run

for sheet_name in sheet_name_list:
    df_sheet_dic[sheet_name] = pd.read_excel(xl_name, sheet_name = sheet_name)

# Change to numpy arrays and shave off the indexing/row titles
# Keep the dataframes around for legibility, but create a dictionary of them.
array_sheet_dic = {}
for sheet_name in sheet_name_list:
    array_sheet_dic[sheet_name] = pd.DataFrame.to_numpy(df_sheet_dic[sheet_name])

# Set constants
H = array_sheet_dic['v'].shape[0]
I = array_sheet_dic['v'].shape[1]
S = array_sheet_dic['qstor'].shape[1]
P = array_sheet_dic['c'].shape[1]

# Just calling dic for convenience in later referencing
dic = array_sheet_dic.copy()

if FC_flag:
    dic['delta'][0][8] = 0.73
else:
```

30 Click "Flag Setter" ¶

The screenshot shows a Jupyter Notebook interface. The top menu bar includes FILE, EDIT, VIEW, INSERT, CELL, KERMIT, WIDGETS, and HELP. A toolbar below has icons for file operations like Open, Save, and Run, along with a 'Run' button and a 'Kernel' dropdown set to 'Python'. The status bar at the bottom right says 'trusted'. The code cell contains Python code for a 'Flag Setter' function:

```
else:  
    dic['delta'][0][8] = 0  
  
dic['wCUM'] = w_mat.cumsum(axis = 0)  
  
investments_flag = True  
if investments_flag == False:  
    dic['wCUM'] = dic['wCUM']*0
```

A blue box highlights the word 'Flag' in the 'Flag Setter' cell title, which is circled in orange.

In [44]:

```
battery_multiplier = 1  
def flag_setter(winter_flag, average_flag, high_flag, max_flag, sunny_flag, win  
    if winter_flag + average_flag + high_flag + max_flag != 1:  
        raise ValueError("multiple demand settings set")  
  
    if winter_flag:  
        dic['delta'] = dic['winter_delta'].copy()  
        dic['d'] = dic['winter_d'].copy()  
        dic['dchgeff'] = dic['winter_dchgeff'].copy()  
  
    if average_flag:  
        dic['delta'] = dic['average_delta'].copy()  
        dic['d'] = dic['average_d'].copy()  
        dic['dchaeff'] = dic['average_dchaeff'].copy()
```

31 Click "battery_multiplier"

The screenshot shows a Jupyter Notebook interface. The top menu bar includes FILE, EDIT, VIEW, INSERT, CELL, KERMIT, WIDGETS, and HELP. A toolbar below has icons for file operations like Open, Save, and Run, along with a 'Run' button and a 'Kernel' dropdown set to 'Python'. The status bar at the bottom right says 'trusted'. The code cell contains Python code for a 'battery_multiplier' function:

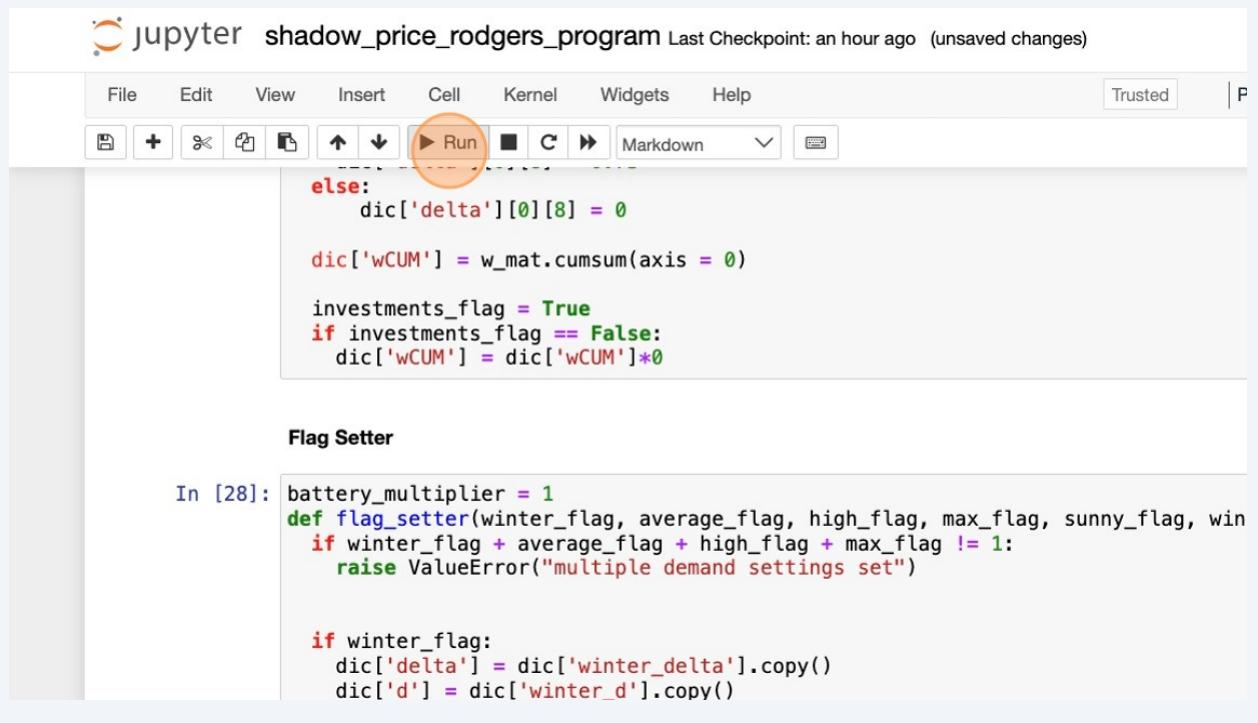
```
else:  
    dic['delta'][0][8] = 0  
  
dic['wCUM'] = w_mat.cumsum(axis = 0)  
  
investments_flag = True  
if investments_flag == False:  
    dic['wCUM'] = dic['wCUM']*0
```

A green box highlights the word 'battery' in the 'battery_multiplier' cell title, which is circled in orange.

In [44]:

```
battery_multiplier = 1  
def flag_setter(winter_flag, average_flag, high_flag, max_flag, sunny_flag, win  
    if winter_flag + average_flag + high_flag + max_flag != 1:  
        raise ValueError("multiple demand settings set")  
  
    if winter_flag:  
        dic['delta'] = dic['winter_delta'].copy()  
        dic['d'] = dic['winter_d'].copy()  
        dic['dchgeff'] = dic['winter_dchgeff'].copy()  
  
    if average_flag:  
        dic['delta'] = dic['average_delta'].copy()  
        dic['d'] = dic['average_d'].copy()  
        dic['dchaeff'] = dic['average_dchaeff'].copy()
```

32 Click "Run"



jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted

Run

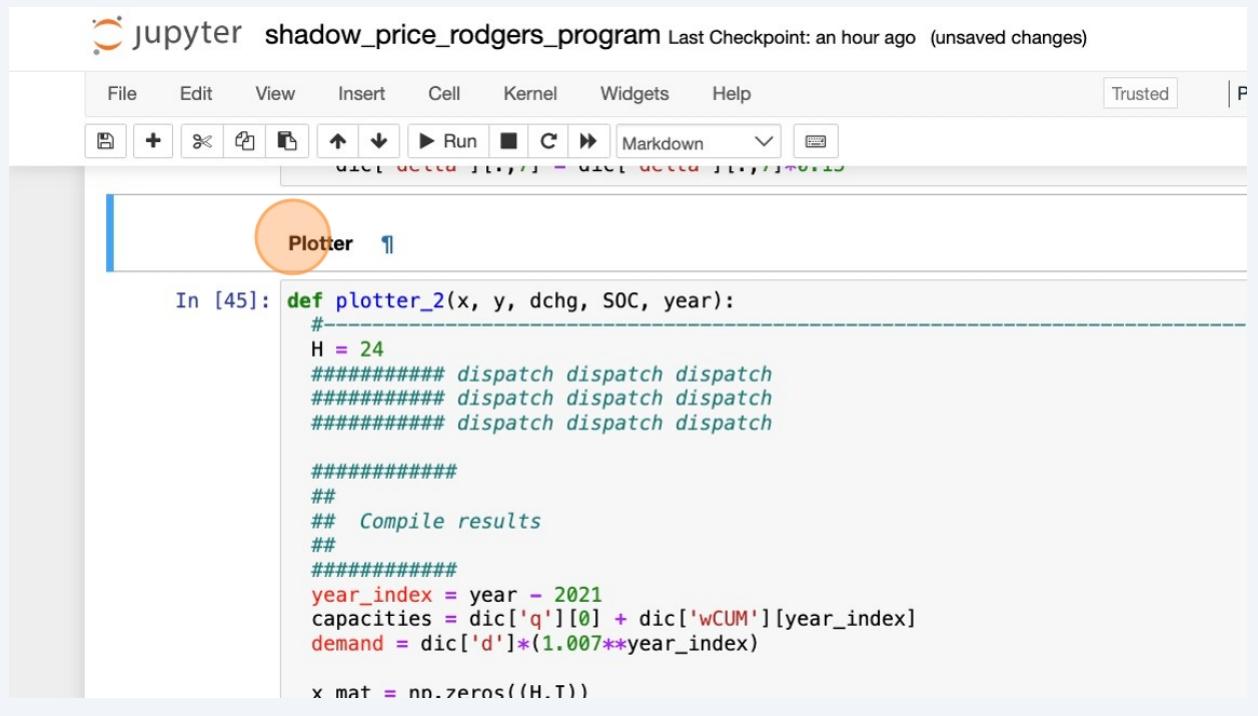
```
else:  
    dic['delta'][0][8] = 0  
  
dic['wCUM'] = w_mat.cumsum(axis = 0)  
  
investments_flag = True  
if investments_flag == False:  
    dic['wCUM'] = dic['wCUM']*0
```

Flag Setter

In [28]:

```
battery_multiplier = 1  
def flag_setter(winter_flag, average_flag, high_flag, max_flag, sunny_flag, win  
    if winter_flag + average_flag + high_flag + max_flag != 1:  
        raise ValueError("multiple demand settings set")  
  
    if winter_flag:  
        dic['delta'] = dic['winter_delta'].copy()  
        dic['d'] = dic['winter_d'].copy()
```

33 Click "Plotter¶"



jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted

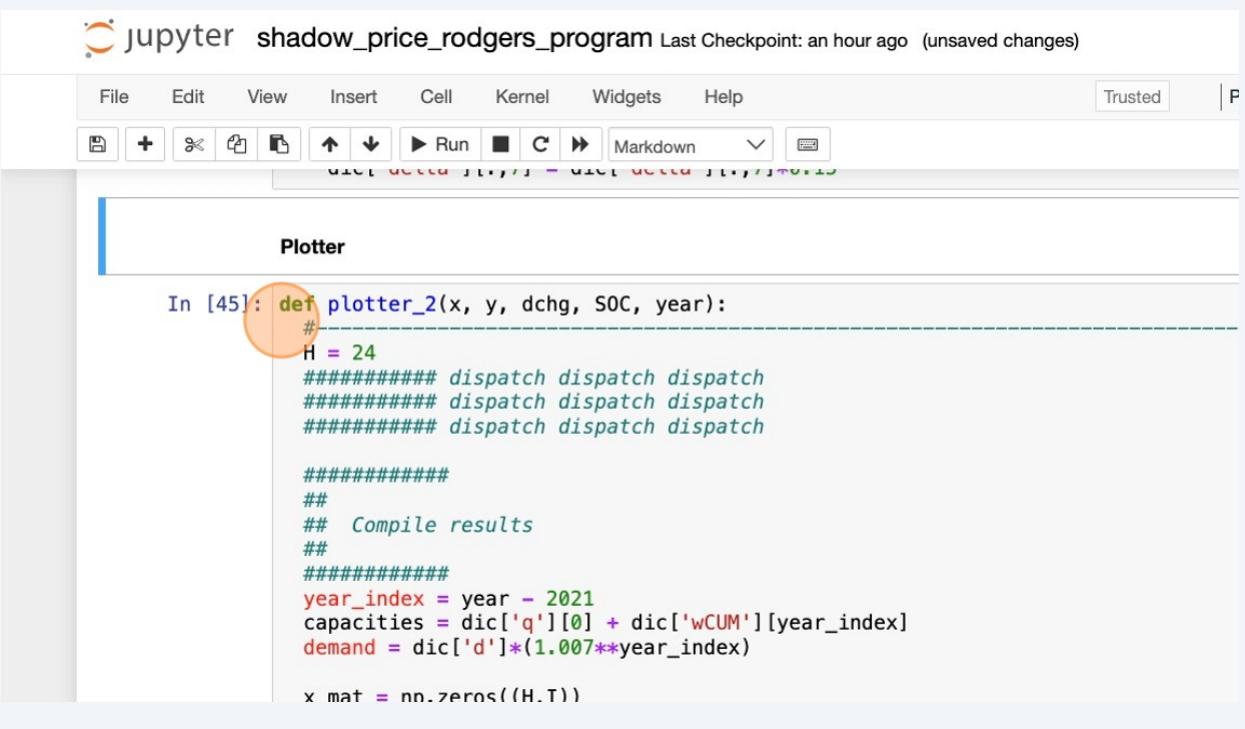
Plotter

In [45]:

```
def plotter_2(x, y, dchg, SOC, year):  
    #-----  
    H = 24  
    ##### dispatch dispatch dispatch  
    ##### dispatch dispatch dispatch  
    ##### dispatch dispatch dispatch  
  
    #####  
    ##  
    ##  Compile results  
    ##  
    #####  
    year_index = year - 2021  
    capacities = dic['q'][0] + dic['wCUM'][year_index]  
    demand = dic['d']*(1.007**year_index)  
  
    x_mat = nn.zeros((H,T))
```

34

Click "#-----
-----"



```
jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted P

Plotter

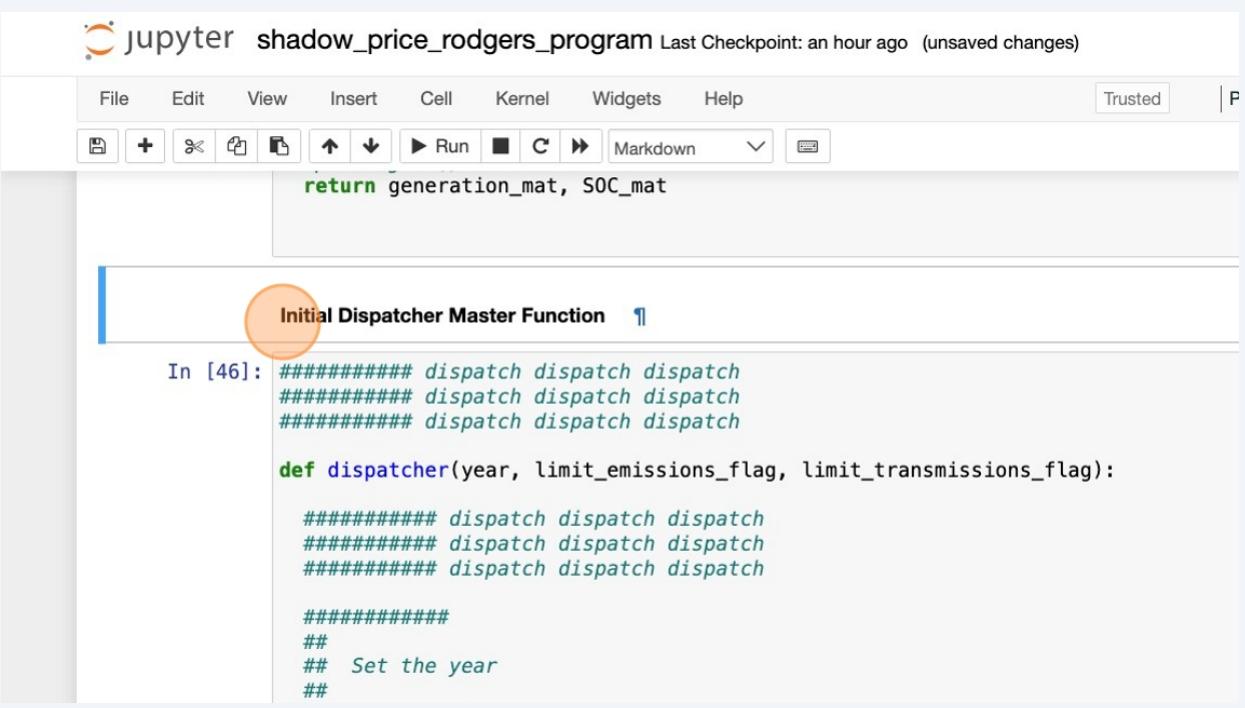
In [45]: def plotter_2(x, y, dchg, SOC, year):
#-----#
H = 24
##### dispatch dispatch dispatch
##### dispatch dispatch dispatch
##### dispatch dispatch dispatch

#####
##
##  Compile results
##
#####
year_index = year - 2021
capacities = dic['q'][0] + dic['wCUM'][year_index]
demand = dic['d']*(1.007**year_index)

x_mat = nn.zeros((H,T))
```

35

Click "Initial Dispatcher Master Function¶"



```
jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted P

return generation_mat, SOC_mat

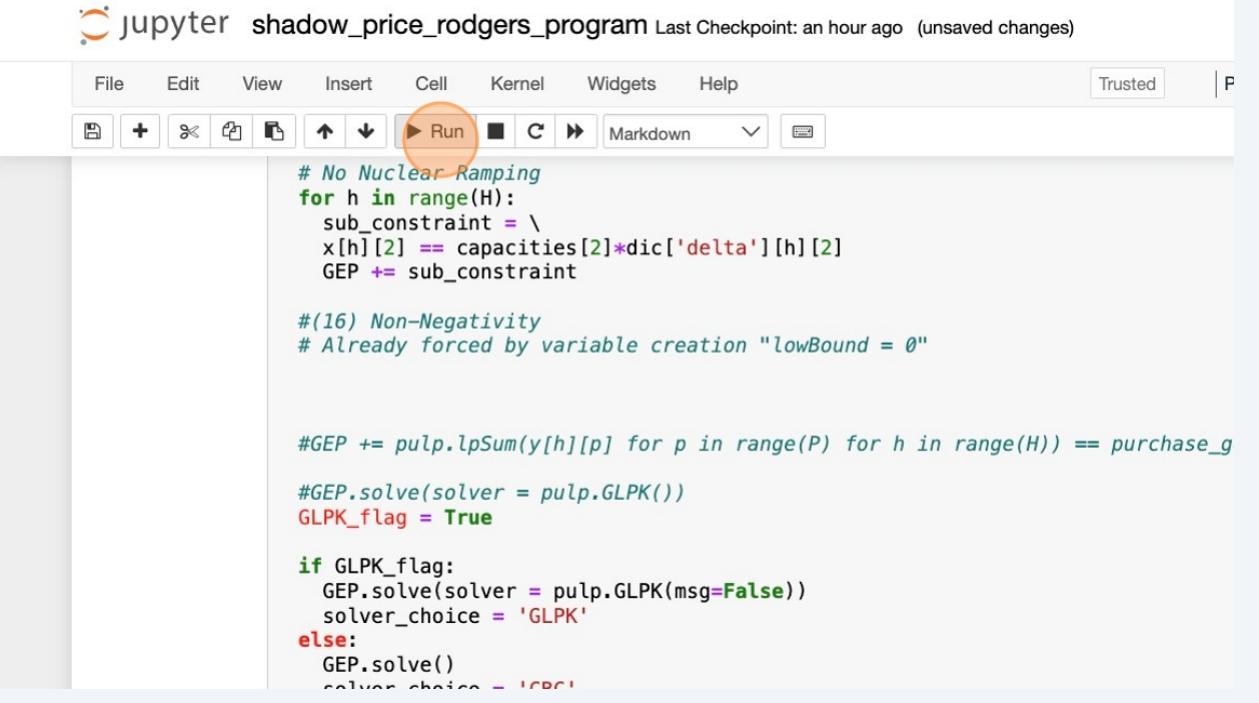
Initial Dispatcher Master Function ¶

In [46]: ##### dispatch dispatch dispatch
##### dispatch dispatch dispatch
##### dispatch dispatch dispatch

def dispatcher(year, limit_emissions_flag, limit_transmissions_flag):
#####
## dispatch dispatch dispatch
## dispatch dispatch dispatch
## dispatch dispatch dispatch

#####
## Set the year
##
```

36 Click "Run"



```
# No Nuclear Ramping
for h in range(H):
    sub_constraint = \
        x[h][2] == capacities[2]*dic['delta'][h][2]
    GEP += sub_constraint

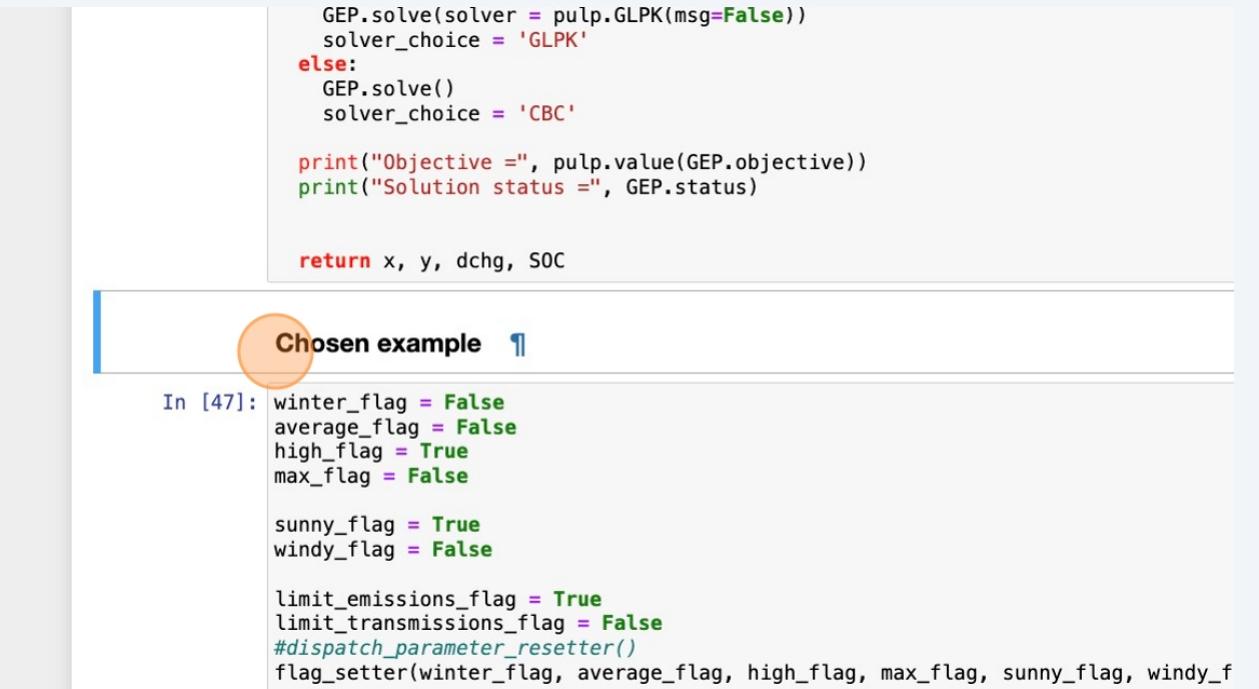
#(16) Non-Negativity
# Already forced by variable creation "lowBound = 0"

#GEP += pulp.lpSum(y[h][p] for p in range(P) for h in range(H)) == purchase_g

#GEP.solve(solver = pulp.GLPK())
GLPK_flag = True

if GLPK_flag:
    GEP.solve(solver = pulp.GLPK(msg=False))
    solver_choice = 'GLPK'
else:
    GEP.solve()
    solver_choice = 'CBC'
```

37 Click "Chosen example¶"



```
GEP.solve(solver = pulp.GLPK(msg=False))
solver_choice = 'GLPK'
else:
    GEP.solve()
    solver_choice = 'CBC'

print("Objective =", pulp.value(GEP.objective))
print("Solution status =", GEP.status)

return x, y, dchg, SOC
```

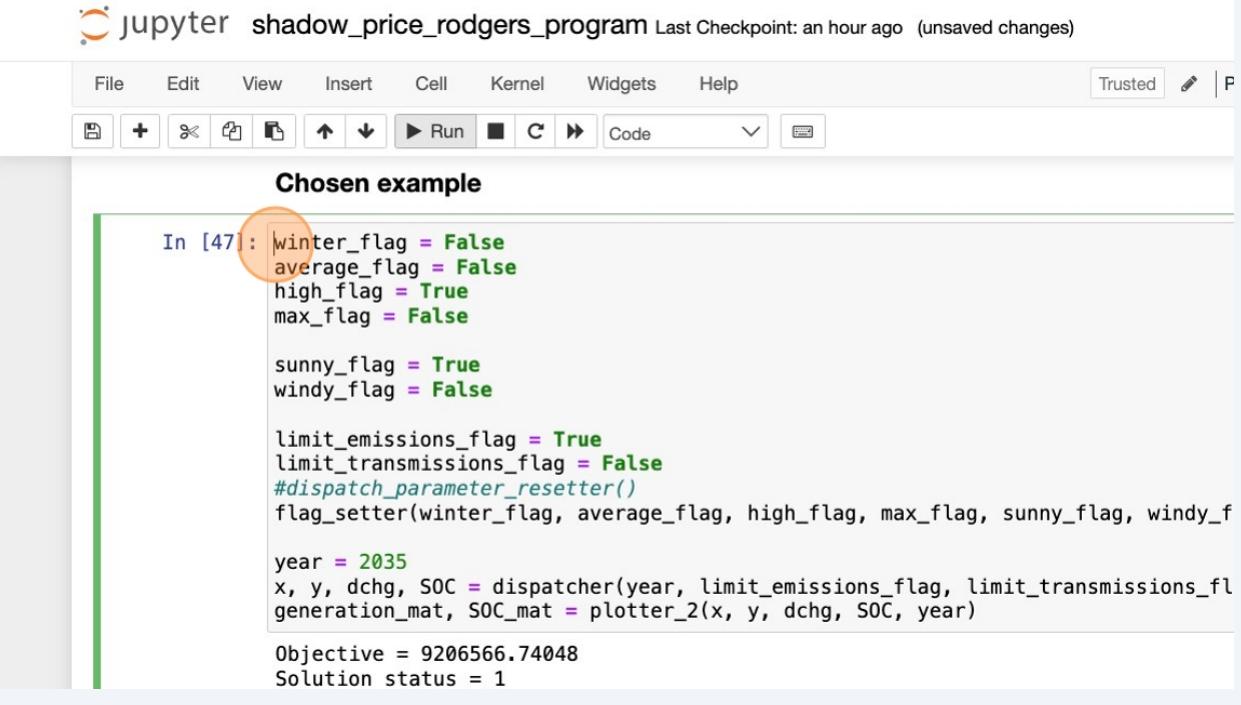
In [47]:

```
winter_flag = False
average_flag = False
high_flag = True
max_flag = False

sunny_flag = True
windy_flag = False

limit_emissions_flag = True
limit_transmissions_flag = False
#dispatch_parameter_resetter()
flag_setter(winter_flag, average_flag, high_flag, max_flag, sunny_flag, windy_f
```

38 Click "winter_flag"



jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted | P

Chosen example

```
In [47]: winter_flag = False
average_flag = False
high_flag = True
max_flag = False

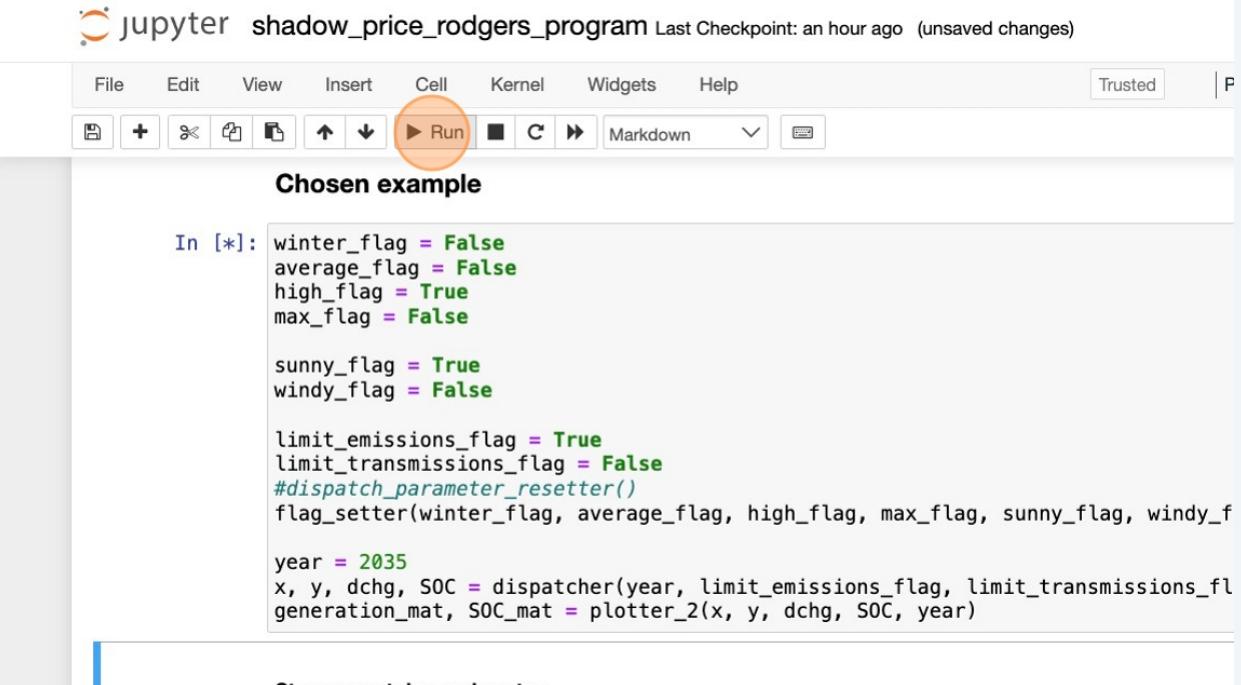
sunny_flag = True
windy_flag = False

limit_emissions_flag = True
limit_transmissions_flag = False
#dispatcher_parameter_resetter()
flag_setter(winter_flag, average_flag, high_flag, max_flag, sunny_flag, windy_flag)

year = 2035
x, y, dchg, SOC = dispatcher(year, limit_emissions_flag, limit_transmissions_flag)
generation_mat, SOC_mat = plotter_2(x, y, dchg, SOC, year)

Objective = 9206566.74048
Solution status = 1
```

39 Click "Run"



jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted | P

Chosen example

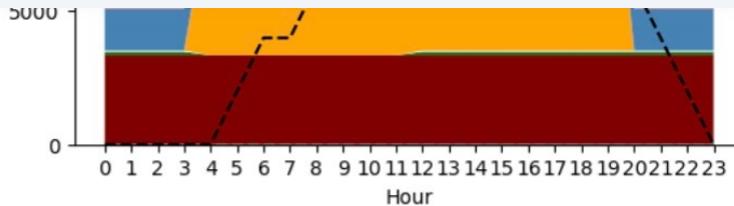
```
In [*]: winter_flag = False
average_flag = False
high_flag = True
max_flag = False

sunny_flag = True
windy_flag = False

limit_emissions_flag = True
limit_transmissions_flag = False
#dispatcher_parameter_resetter()
flag_setter(winter_flag, average_flag, high_flag, max_flag, sunny_flag, windy_flag)

year = 2035
x, y, dchg, SOC = dispatcher(year, limit_emissions_flag, limit_transmissions_flag)
generation_mat, SOC_mat = plotter_2(x, y, dchg, SOC, year)
```

40 Click "# get the umet demand vector"



Store unmet demand vector

```
In [48]: # get the umet demand vector  
unmet_demand = np.zeros((30,1))  
#unmet_demand[:24] = generation_mat[:,11].reshape(24,1)  
#total_unmet_demand = sum(unmet_demand)[0]
```

3) Dispatch with removed load

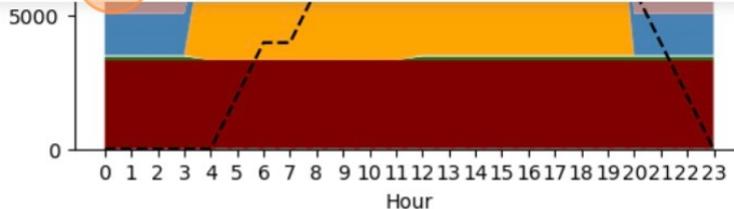
We are pretending there is no unmet demand, this file is to test shadow prices. But there was the warning about how the 3rd dispatch is the one that incorporates an actual decision for load shedding. How do we represent shifting?

41 Click "Run"

jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted P

Run



Store unmet demand vector

```
In [32]: # get the umet demand vector  
unmet_demand = np.zeros((30,1))  
#unmet_demand[:24] = generation_mat[:,11].reshape(24,1)  
#total_unmet_demand = sum(unmet_demand)[0]
```

3) Dispatch with removed load

42 Click "3) Dispatch with removed load"

In [32]:

```
# get the unmet demand vector
unmet_demand = np.zeros((30,1))
#unmet_demand[:24] = generation_mat[:,11].reshape(24,1)
#total_unmet_demand = sum(unmet_demand)[0]
```

3) Dispatch with removed load

We are pretending there is no unmet demand, this file is to test shadow prices. But there was the question about how the 3rd dispatch is the one that incorporates an actual decision for load shedding. do we represent shifting?

So I guess shedding is a decision variable in form of generation, but shifting would just be based on shadow pricing. I guess we'll see...

Removed Demand Plotter

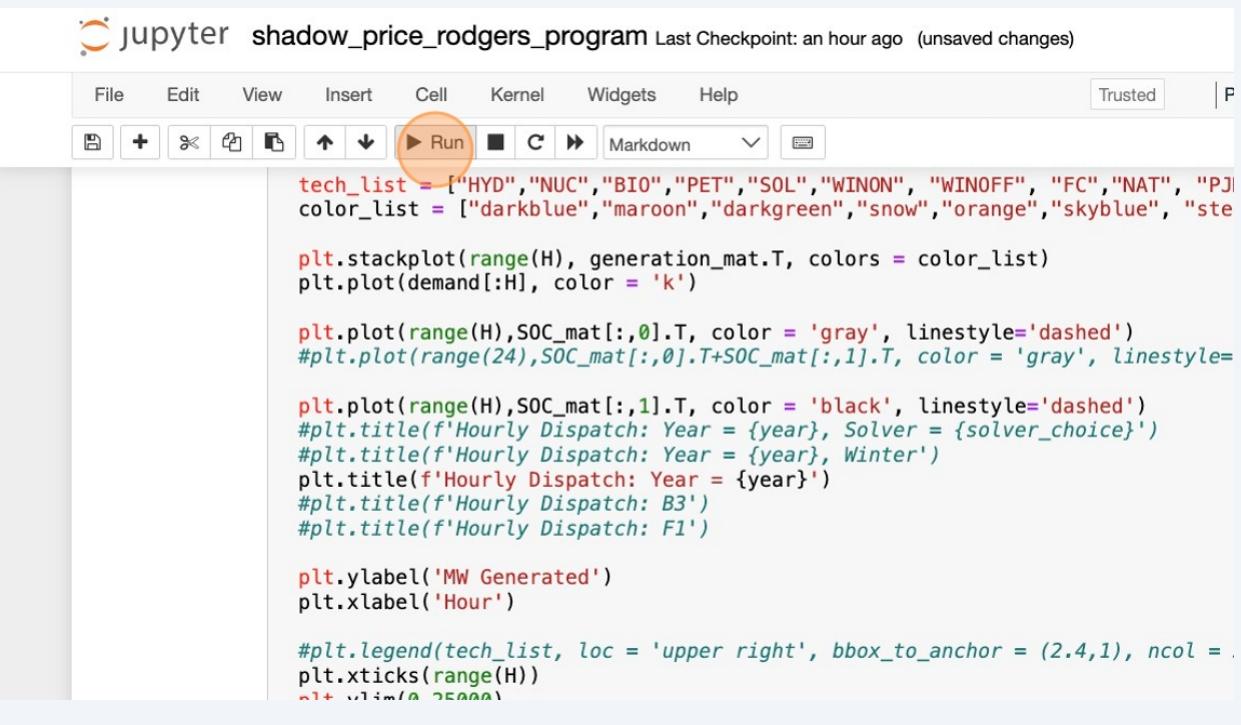
43 Click "Removed Demand Plotter"

Removed Demand Plotter

```
In [49]: def removed_demand_plotter(x, y, dchg, SOC, year, unmet_demand):
    #-----
    H = 24
    ##### dispatch dispatch dispatch
    ##### dispatch dispatch dispatch
    ##### dispatch dispatch dispatch

    #####
    ##
    ##  Compile results
    ##
    #####
    year_index = year - 2021
    capacities = dict['n101'] + dict['wCLM1'][year_index]
```

44 Click "Run"



```
jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted | P

Run

tech_list = ["HYD", "NUC", "BIO", "PET", "SOL", "WINON", "WINOFF", "FC", "NAT", "PJ]
color_list = ["darkblue", "maroon", "darkgreen", "snow", "orange", "skyblue", "ste

plt.stackplot(range(H), generation_mat.T, colors = color_list)
plt.plot(demand[:H], color = 'k')

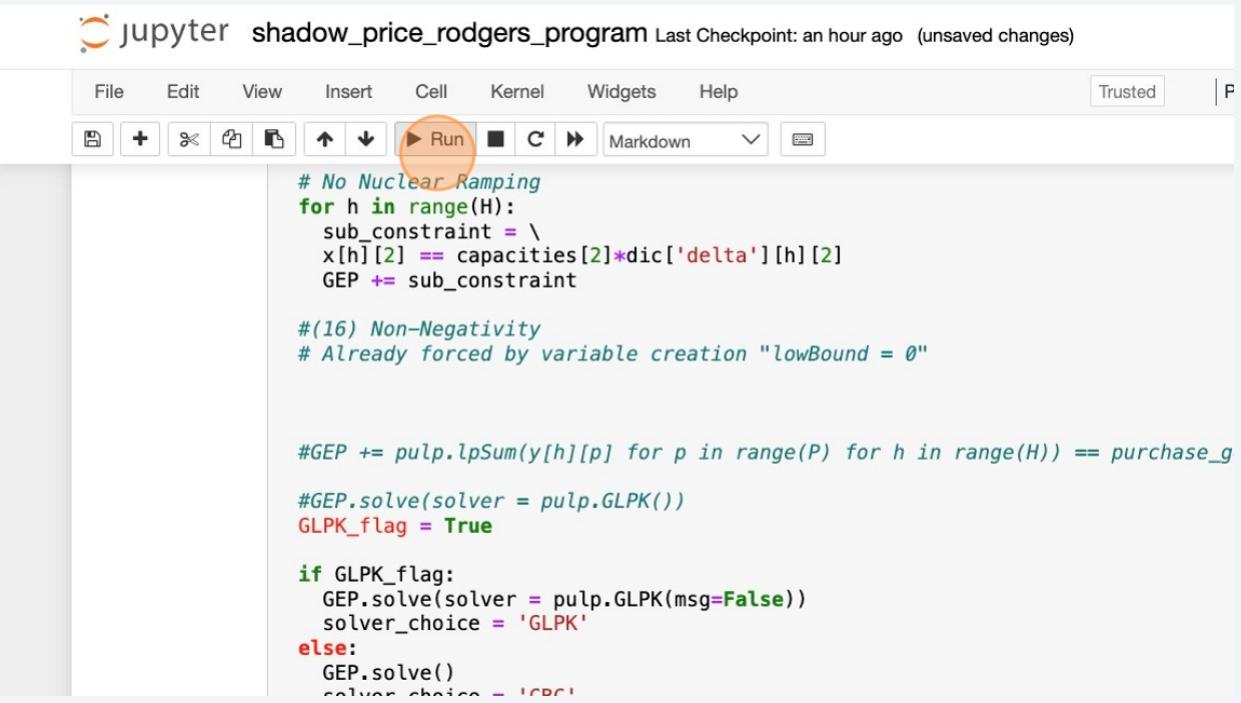
plt.plot(range(H), SOC_mat[:,0].T, color = 'gray', linestyle='dashed')
# plt.plot(range(24), SOC_mat[:,0].T+SOC_mat[:,1].T, color = 'gray', linestyle=

plt.plot(range(H), SOC_mat[:,1].T, color = 'black', linestyle='dashed')
# plt.title(f'Hourly Dispatch: Year = {year}, Solver = {solver_choice}')
# plt.title(f'Hourly Dispatch: Year = {year}, Winter')
plt.title(f'Hourly Dispatch: Year = {year}')
# plt.title(f'Hourly Dispatch: B3')
# plt.title(f'Hourly Dispatch: F1')

plt.ylabel('MW Generated')
plt.xlabel('Hour')

# plt.legend(tech_list, loc = 'upper right', bbox_to_anchor = (2.4,1), ncol =
plt.xticks(range(H))
# 1+ ..::: 25000
```

45 Click "Run"



```
jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted | P

Run

# No Nuclear Ramping
for h in range(H):
    sub_constraint = \
        x[h][2] == capacities[2]*dic['delta'][h][2]
    GEP += sub_constraint

#(16) Non-Negativity
# Already forced by variable creation "lowBound = 0"

#GEP += pulp.lpSum(y[h][p] for p in range(P) for h in range(H)) == purchase_g
#GEP.solve(solver = pulp.GLPK())
GLPK_flag = True

if GLPK_flag:
    GEP.solve(solver = pulp.GLPK(msg=False))
    solver_choice = 'GLPK'
else:
    GEP.solve()
    solver_choice = 'CPLEX'
```

46

Click "x, y, dchg, SOC, model = removed_demand_dispatcher(year, limit_emissions_flag, limit_transmissions_flag, unmet_demand)"

```
print("Objective =", pulp.value(GEP.objective))
print("Solution status =", GEP.status)

return x, y, dchg, SOC, GEP
```

Run to show with unmet demand removed

```
In [51]: x, y, dchg, SOC, model = removed_demand_dispatcher(year, limit_emissions_flag, generation_mat, SOC_mat = removed_demand_plotter(x, y, dchg, SOC, year, unmet_d
Objective = 9206566.74048
Solution status = 1
```

Hourly Dispatch: Year = 2035

47

Click "Run"

jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted

Run

```
print("Objective =", pulp.value(GEP.objective))
print("Solution status =", GEP.status)

return x, y, dchg, SOC, GEP
```

Run to show with unmet demand removed

```
In [*]: x, y, dchg, SOC, model = removed_demand_dispatcher(year, limit_emissions_flag, generation_mat, SOC_mat = removed_demand_plotter(x, y, dchg, SOC, year, unmet_d
```

4) Dispatch with reallocation of load

Initialize

```
In [39]: ##### dispatch dispatch dispatch
```

48 Click "4) Dispatch with reallocation of load"

The screenshot shows a Jupyter Notebook interface with the title bar 'jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)'. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. A toolbar below the menu has icons for file operations like Open, Save, and Run, along with a 'Trusted' button. Below the toolbar is a horizontal scroll bar with numerical markers from 0 to 23 labeled 'Hour'. The main content area contains a section titled '4) Dispatch with reallocation of load' with a sub-section 'Initialize'. In cell [39], the code imports pandas and mounts a drive:

```
In [39]: ##### dispatch dispatch dispatch
##### dispatch dispatch dispatch
##### dispatch dispatch dispatch

#####
## Reset imports and remount
##
#####
# drive.mount("/content/drive", force_remount=True)
```

49 Click "wb"

The screenshot shows a Jupyter Notebook interface with the title bar 'jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)'. The menu bar includes File, Edit, View, Insert, Cell, Kernel, Widgets, and Help. A toolbar below the menu has icons for file operations like Open, Save, and Run, along with a 'Trusted' button. The code cell contains a script to read an Excel file and convert its sheets into a dictionary of Pandas Dataframes:

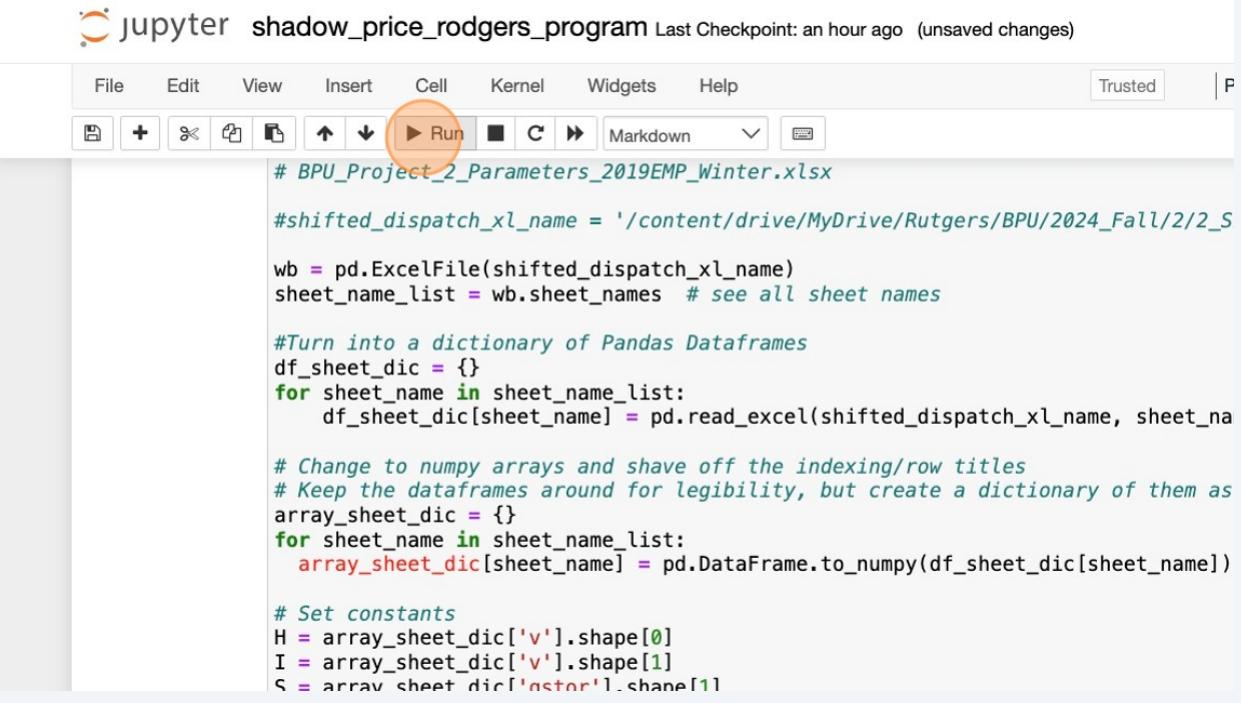
```
# BPU_Project_2_Parameters_2019EMP_Winter.xlsx
shifted_dispatch_xl_name = '/content/drive/MyDrive/Rutgers/BPU/2024_Fall/2/2_S
wb = pd.ExcelFile(shifted_dispatch_xl_name)
sheet_name_list = wb.sheet_names # see all sheet names

# Turn into a dictionary of Pandas Dataframes
df_sheet_dic = {}
for sheet_name in sheet_name_list:
    df_sheet_dic[sheet_name] = pd.read_excel(shifted_dispatch_xl_name, sheet_na

# Change to numpy arrays and shave off the indexing/row titles
# Keep the dataframes around for legibility, but create a dictionary of them as
array_sheet_dic = {}
for sheet_name in sheet_name_list:
    array_sheet_dic[sheet_name] = pd.DataFrame.to_numpy(df_sheet_dic[sheet_name])

# Set constants
H = array_sheet_dic['v'].shape[0]
I = array_sheet_dic['v'].shape[1]
S = array_sheet_dic['cost'].shape[1]
```

50 Click "Run"



```
# BPU_Project_2_Parameters_2019EMP_Winter.xlsx
shifted_dispatch_xl_name = '/content/drive/MyDrive/Rutgers/BPU/2024_Fall/2/2_S

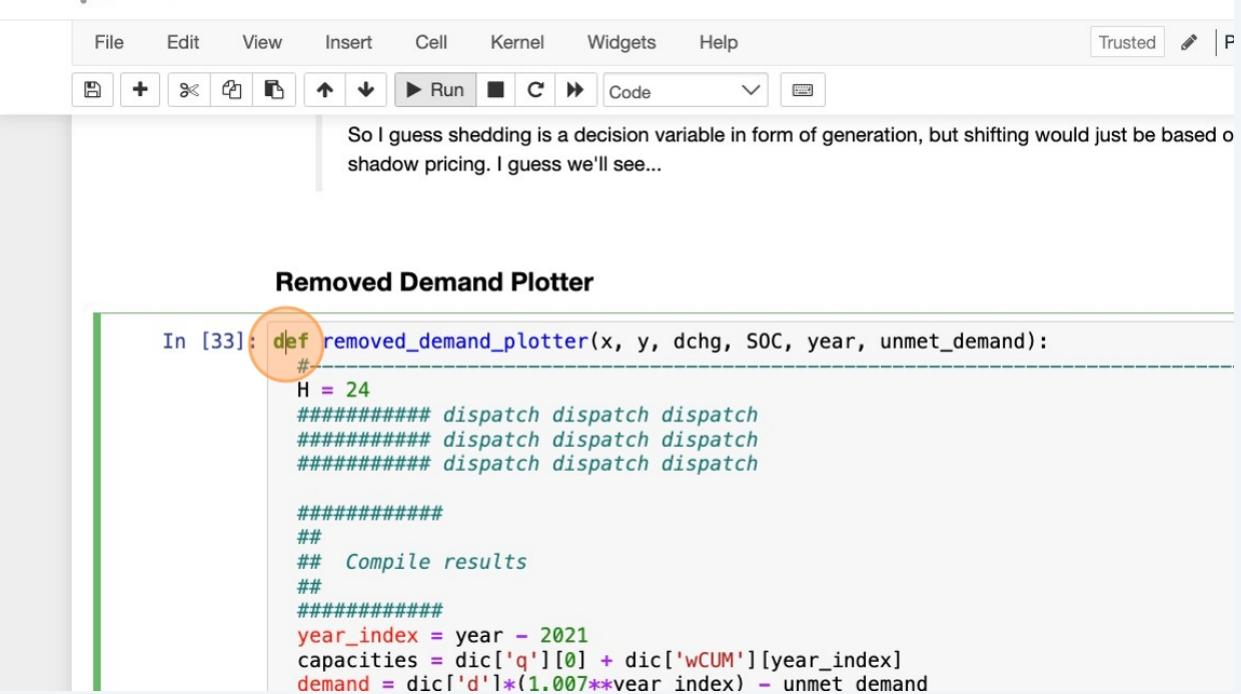
wb = pd.ExcelFile(shifted_dispatch_xl_name)
sheet_name_list = wb.sheet_names # see all sheet names

# Turn into a dictionary of Pandas Dataframes
df_sheet_dic = {}
for sheet_name in sheet_name_list:
    df_sheet_dic[sheet_name] = pd.read_excel(shifted_dispatch_xl_name, sheet_na

# Change to numpy arrays and shave off the indexing/row titles
# Keep the dataframes around for legibility, but create a dictionary of them as
array_sheet_dic = {}
for sheet_name in sheet_name_list:
    array_sheet_dic[sheet_name] = pd.DataFrame.to_numpy(df_sheet_dic[sheet_name])

# Set constants
H = array_sheet_dic['v'].shape[0]
I = array_sheet_dic['v'].shape[1]
S = array_sheet_dic['nstar'].shape[1]
```

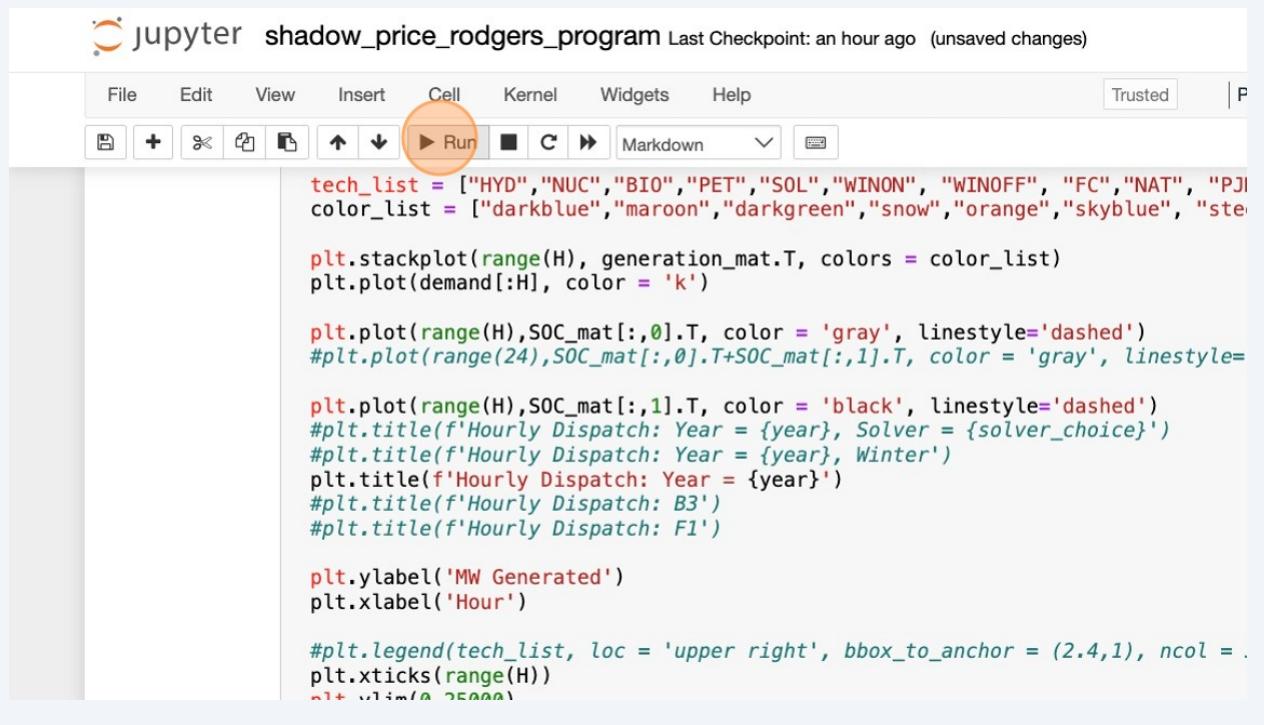
51 Click "def"



```
In [33]: def removed_demand_plotter(x, y, dchg, SOC, year, unmet_demand):
    #
    H = 24
    ##### dispatch dispatch dispatch
    ##### dispatch dispatch dispatch
    ##### dispatch dispatch dispatch

    #####
    ##
    ## Compile results
    ##
    #####
    year_index = year - 2021
    capacities = dic['q'][0] + dic['wCUM'][year_index]
    demand = dic['d']*(1.007**year_index) - unmet demand
```

52 Click "Run"



jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted

Run

```
tech_list = ["HYD", "NUC", "BIO", "PET", "SOL", "WINON", "WINOFF", "FC", "NAT", "PJI"]
color_list = ["darkblue", "maroon", "darkgreen", "snow", "orange", "skyblue", "steelblue"]

plt.stackplot(range(H), generation_mat.T, colors = color_list)
plt.plot(demand[:H], color = 'k')

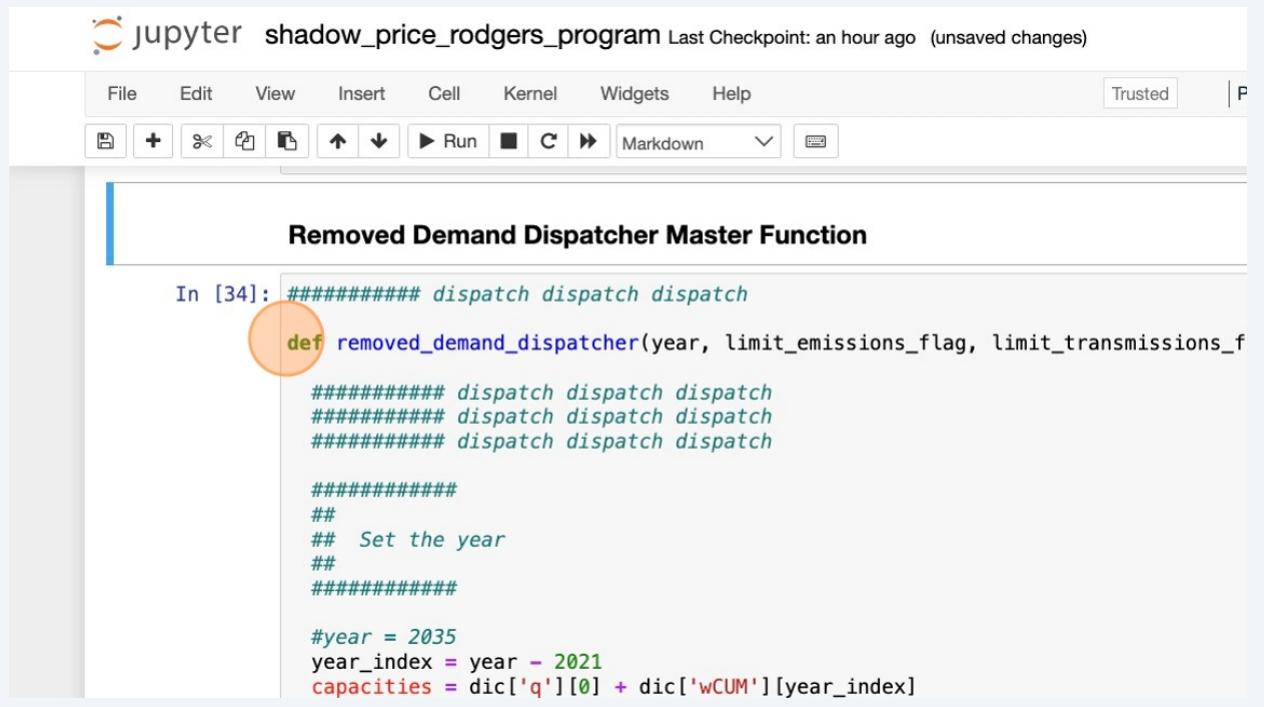
plt.plot(range(H),SOC_mat[:,0].T, color = 'gray', linestyle='dashed')
#plt.plot(range(24),SOC_mat[:,0].T+SOC_mat[:,1].T, color = 'gray', linestyle='dashed')

plt.plot(range(H),SOC_mat[:,1].T, color = 'black', linestyle='dashed')
#plt.title(f'Hourly Dispatch: Year = {year}, Solver = {solver_choice}')
#plt.title(f'Hourly Dispatch: Year = {year}, Winter')
plt.title(f'Hourly Dispatch: Year = {year}')
#plt.title(f'Hourly Dispatch: B3')
#plt.title(f'Hourly Dispatch: F1')

plt.ylabel('MW Generated')
plt.xlabel('Hour')

# plt.legend(tech_list, loc = 'upper right', bbox_to_anchor = (2.4,1), ncol = 3)
plt.xticks(range(H))
```

53 Click "def"



jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted

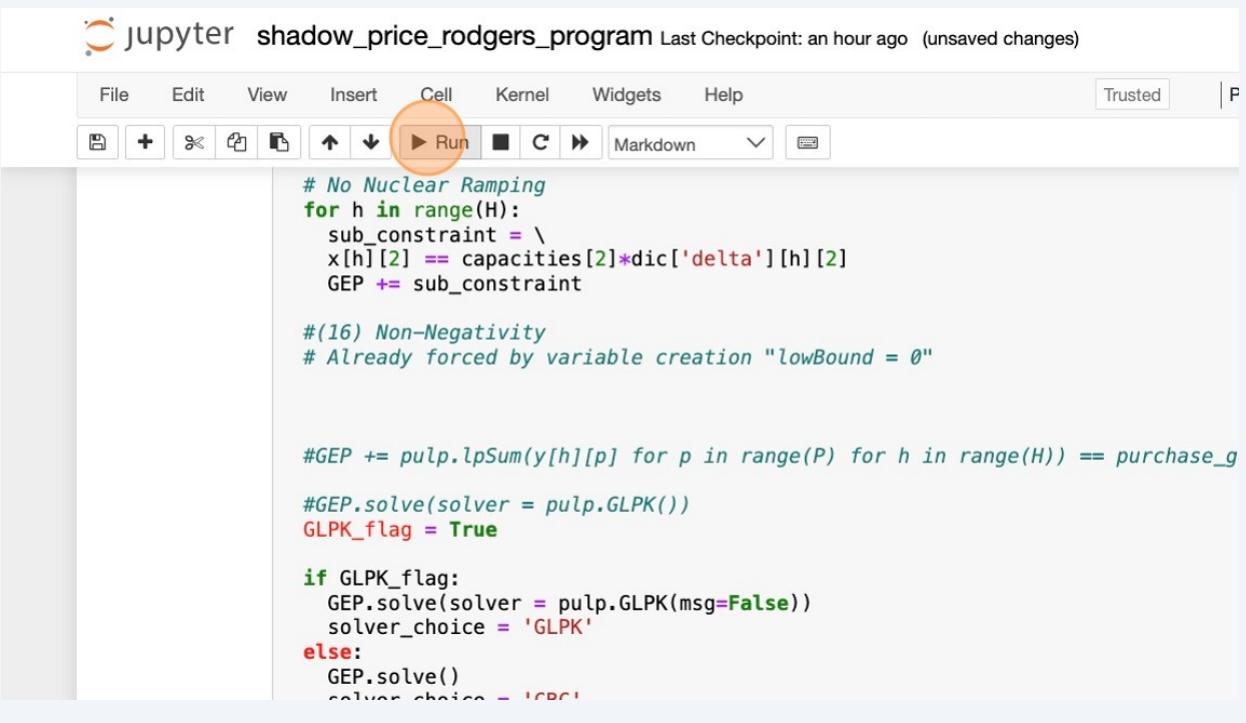
Run

Removed Demand Dispatcher Master Function

```
In [34]: ##### dispatch dispatch dispatch
def removed_demand_dispatcher(year, limit_emissions_flag, limit_transmissions_flag):
    ##### dispatch dispatch dispatch
    ##### dispatch dispatch dispatch
    ##### dispatch dispatch dispatch

    #####
    ##
    ## Set the year
    ##
    #####
    #year = 2035
    year_index = year - 2021
    capacities = dic['q'][0] + dic['wCUM'][year_index]
```

54 Click "Run"



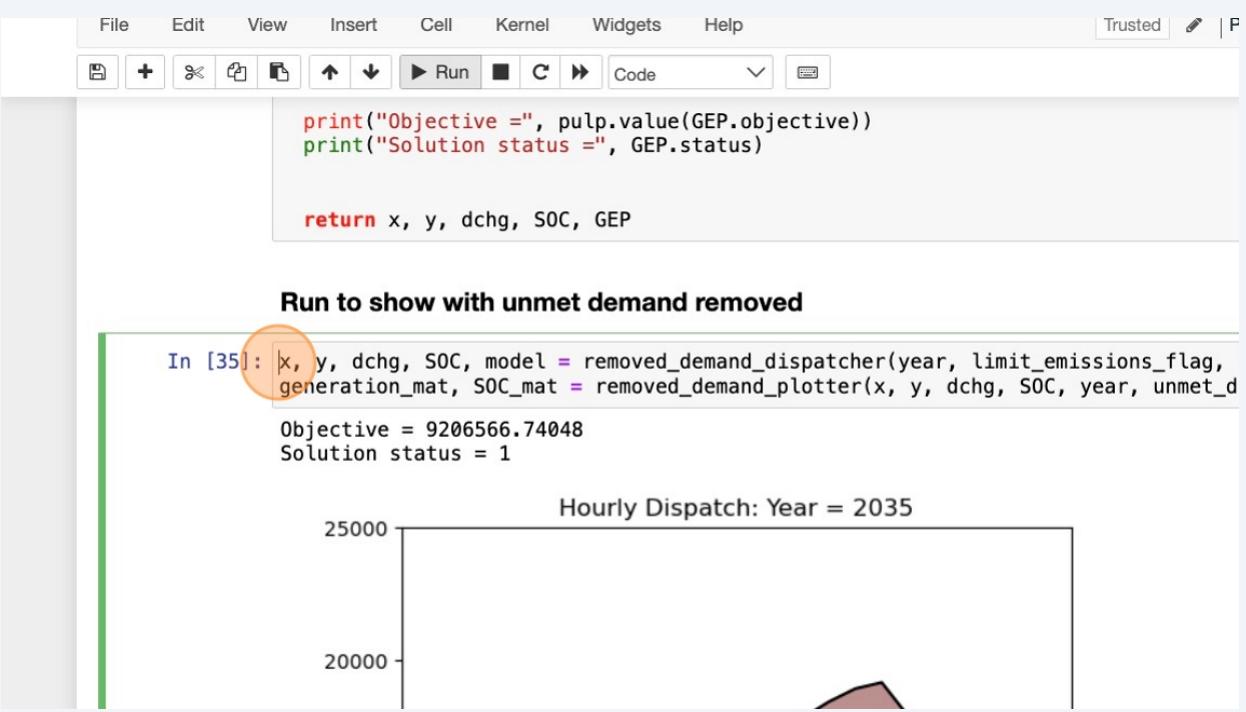
```
# No Nuclear Ramping
for h in range(H):
    sub_constraint = \
        x[h][2] == capacities[2]*dic['delta'][h][2]
    GEP += sub_constraint

#(16) Non-Negativity
# Already forced by variable creation "lowBound = 0"

#GEP += pulp.lpSum(y[h][p] for p in range(P) for h in range(H)) == purchase_g
#GEP.solve(solver = pulp.GLPK())
GLPK_flag = True

if GLPK_flag:
    GEP.solve(solver = pulp.GLPK(msg=False))
    solver_choice = 'GLPK'
else:
    GEP.solve()
    solver_choice = 'CPLEX'
```

55 Click "x"



```
print("Objective =", pulp.value(GEP.objective))
print("Solution status =", GEP.status)

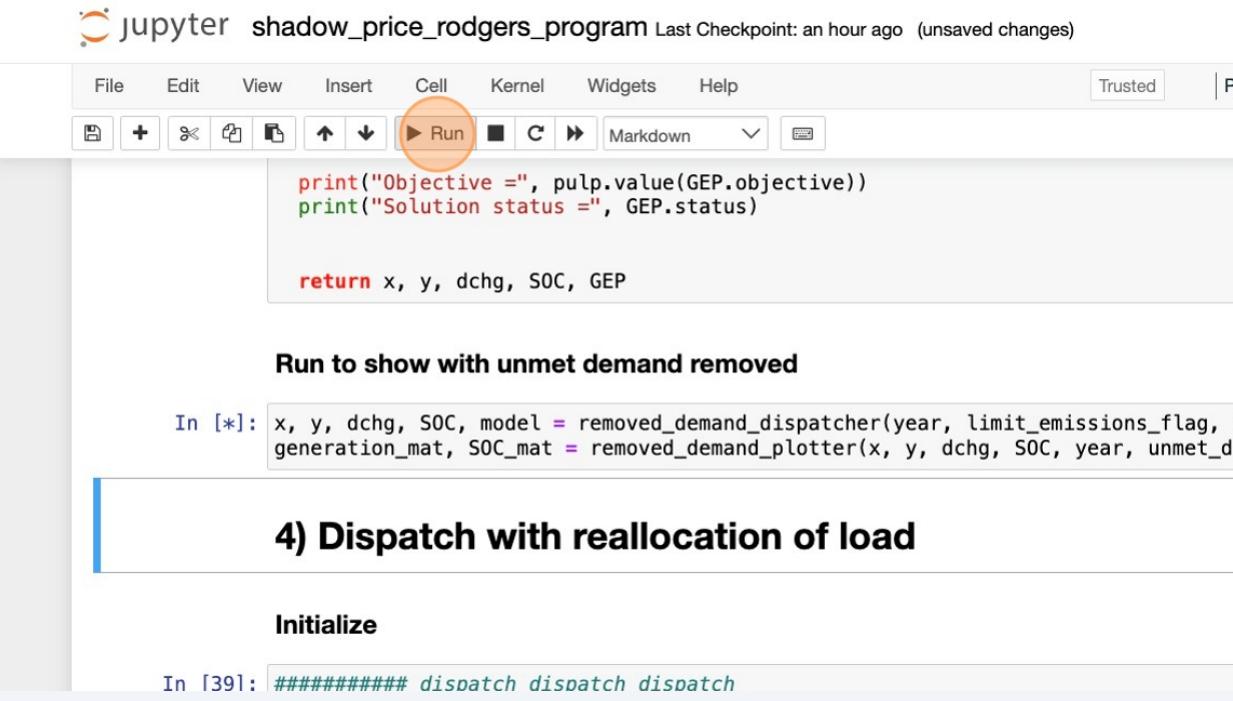
return x, y, dchg, SOC, GEP
```

Run to show with unmet demand removed

```
In [35]: x, y, dchg, SOC, model = removed_demand_dispatcher(year, limit_emissions_flag, generation_mat, SOC_mat = removed_demand_plotter(x, y, dchg, SOC, year, unmet_d
Objective = 9206566.74048
Solution status = 1
```



56 Click "Run"



```
print("Objective =", pulp.value(GEP.objective))
print("Solution status =", GEP.status)

return x, y, dchg, SOC, GEP
```

Run to show with unmet demand removed

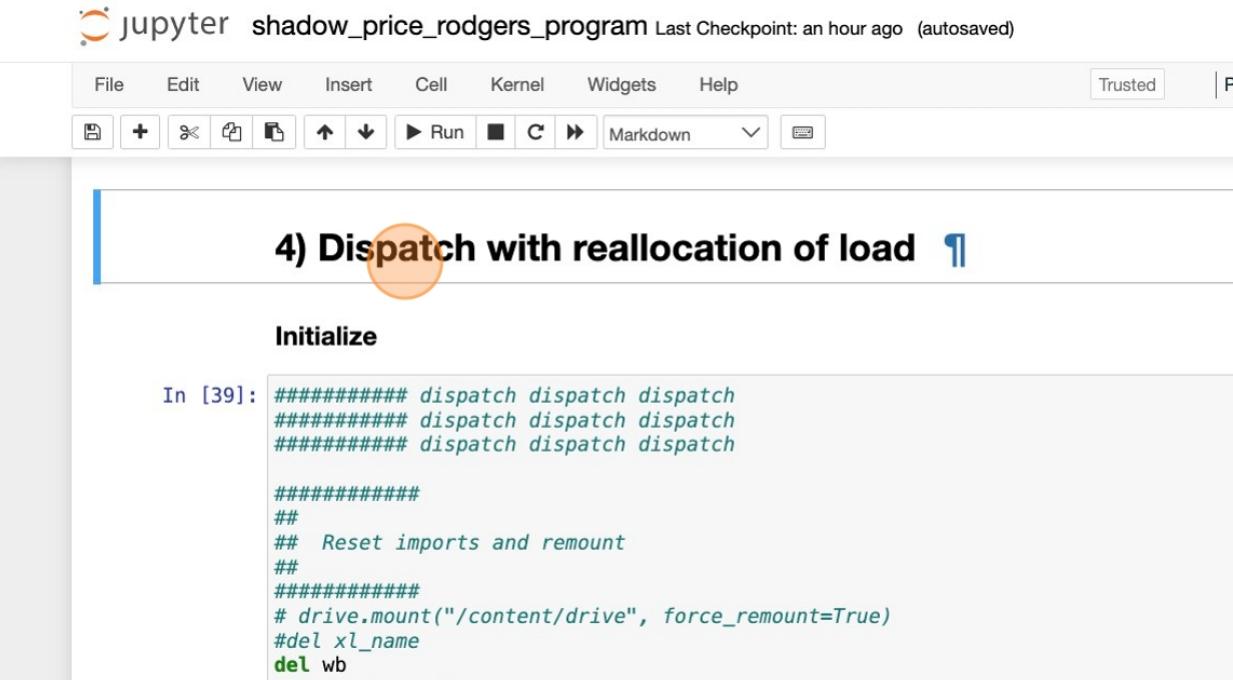
```
In [*]: x, y, dchg, SOC, model = removed_demand_dispatcher(year, limit_emissions_flag,
generation_mat, SOC_mat = removed_demand_plotter(x, y, dchg, SOC, year, unmet_d
```

4) Dispatch with reallocation of load

Initialize

```
In [39]: ##### dispatch dispatch dispatch
```

57 Click "4) Dispatch with reallocation of load"



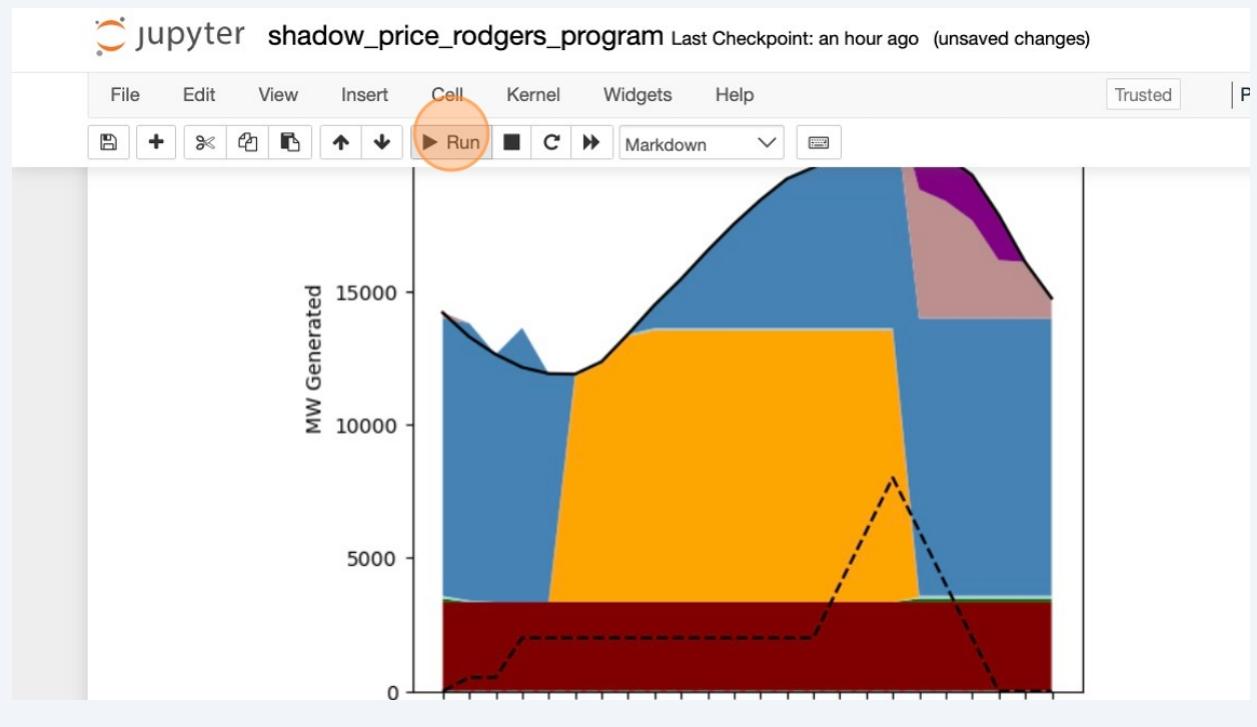
4) Dispatch with reallocation of load [1](#)

Initialize

```
In [39]: ##### dispatch dispatch dispatch
##### dispatch dispatch dispatch
##### dispatch dispatch dispatch

#####
##
## Reset imports and remount
##
#####
# drive.mount("/content/drive", force_remount=True)
#del xl_name
del wb
del df_sheet dic
```

58 Click "Run"



59 Click "wb"

```
wb = pd.ExcelFile(shifted_dispatch_xl_name)
sheet_name_list = wb.sheet_names # see all sheet names

# Turn into a dictionary of Pandas Dataframes
df_sheet_dic = {}
for sheet_name in sheet_name_list:
    df_sheet_dic[sheet_name] = pd.read_excel(shifted_dispatch_xl_name, sheet_name)

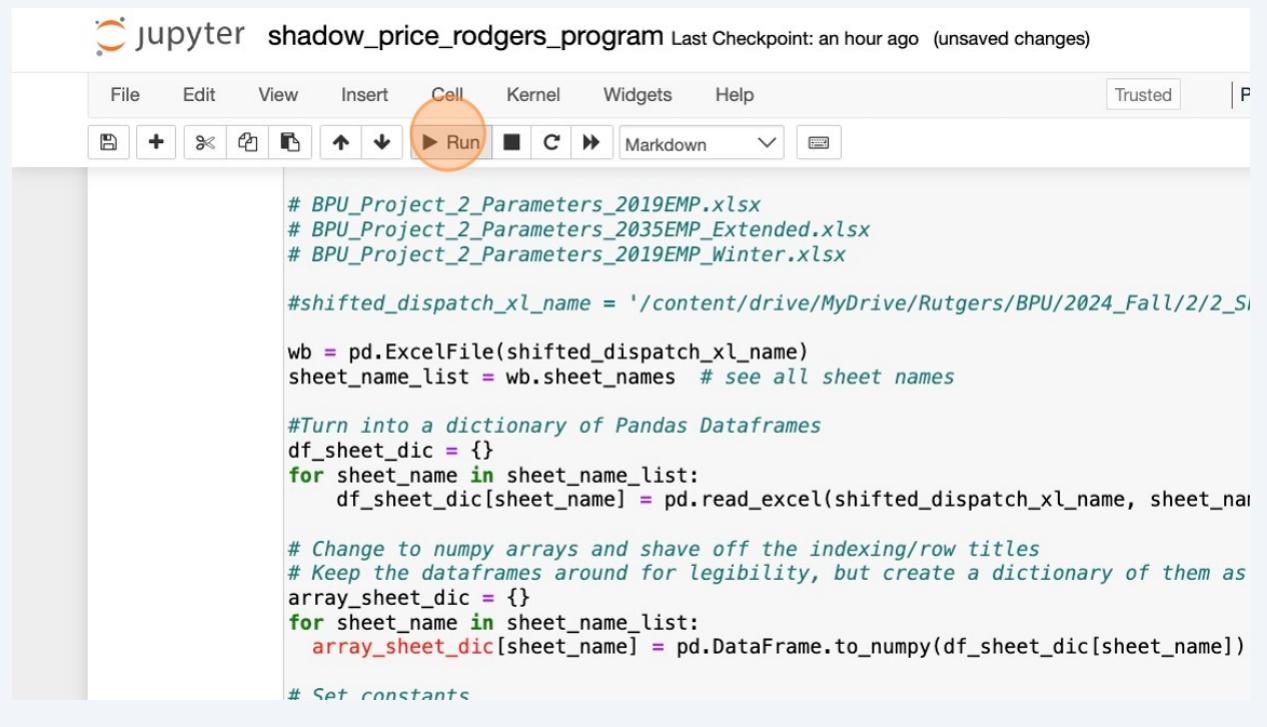
# Change to numpy arrays and shave off the indexing/row titles
# Keep the dataframes around for legibility, but create a dictionary of them as
array_sheet_dic = {}
for sheet_name in sheet_name_list:
    array_sheet_dic[sheet_name] = pd.DataFrame.to_numpy(df_sheet_dic[sheet_name])

# Set constants
H = array_sheet_dic['v'].shape[0]
I = array_sheet_dic['v'].shape[1]
S = array_sheet_dic['qstor'].shape[1]
P = array_sheet_dic['c'].shape[1]

# Just calling dic for convenience in later referencing
dic = array_sheet_dic.copy()

if FC_flag:
    dic['delta'][0][8] = 0.73
else:
```

60 Click "Run"



jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted | P

Run

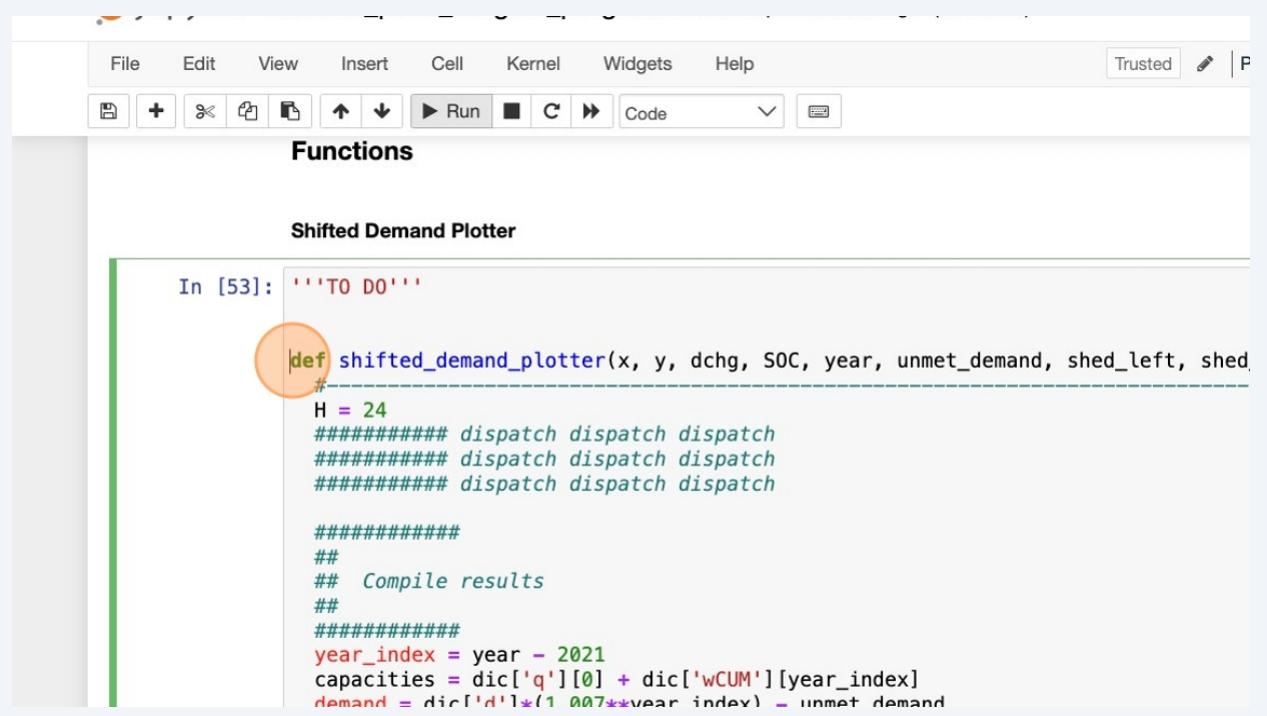
```
# BPU_Project_2_Parameters_2019EMP.xlsx
# BPU_Project_2_Parameters_2035EMP_Extended.xlsx
# BPU_Project_2_Parameters_2019EMP_Winter.xlsx

#shifted_dispatch_xl_name = '/content/drive/MyDrive/Rutgers/BPU/2024_Fall/2/2_Si
wb = pd.ExcelFile(shifted_dispatch_xl_name)
sheet_name_list = wb.sheet_names # see all sheet names

#Turn into a dictionary of Pandas Dataframes
df_sheet_dic = {}
for sheet_name in sheet_name_list:
    df_sheet_dic[sheet_name] = pd.read_excel(shifted_dispatch_xl_name, sheet_na
# Change to numpy arrays and shave off the indexing/row titles
# Keep the dataframes around for legibility, but create a dictionary of them as
array_sheet_dic = {}
for sheet_name in sheet_name_list:
    array_sheet_dic[sheet_name] = pd.DataFrame.to_numpy(df_sheet_dic[sheet_name])

# Set constants
```

61 Click "def"



File Edit View Insert Cell Kernel Widgets Help Trusted | P

Run Code

Functions

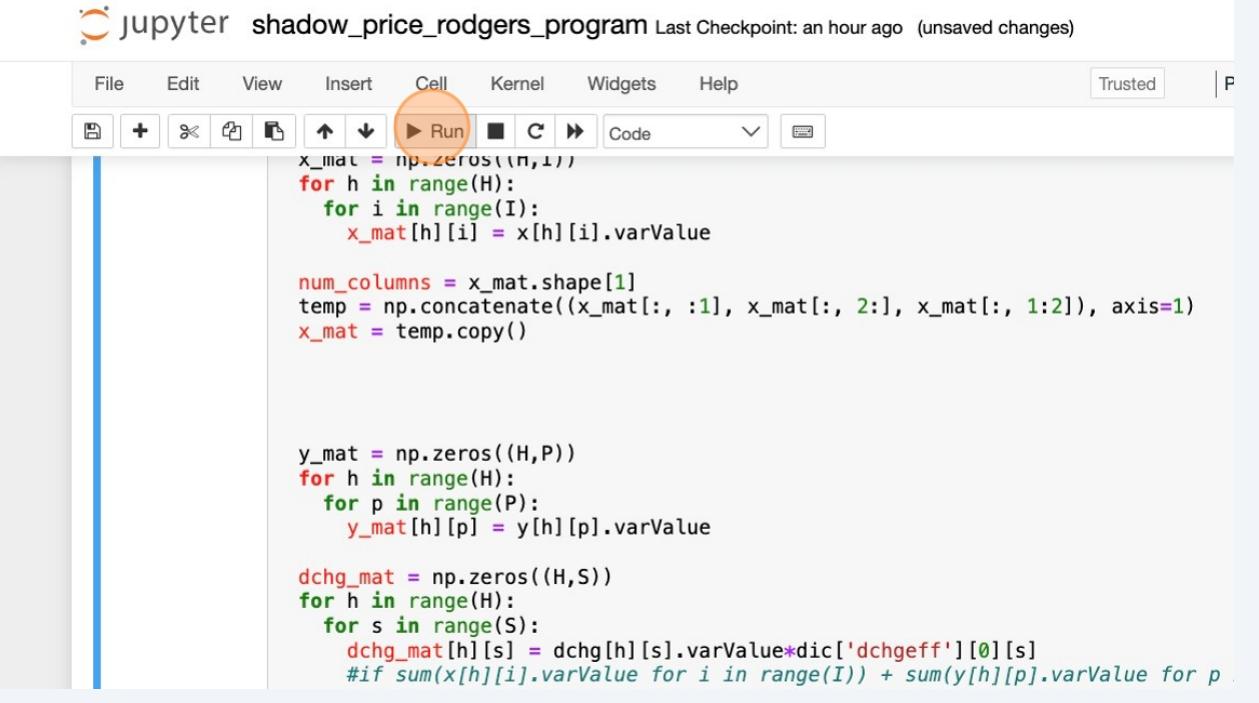
Shifted Demand Plotter

```
In [53]: """TO DO"""

def shifted_demand_plotter(x, y, dchg, SOC, year, unmet_demand, shed_left, shed
#-----
H = 24
##### dispatch dispatch
##### dispatch dispatch
##### dispatch dispatch

#####
## Compile results
##
#####
year_index = year - 2021
capacities = dic['q'][0] + dic['wCUM'][year_index]
demand = dic['d']*(1.007**year_index) - unmet demand
```

62 Click "Run"



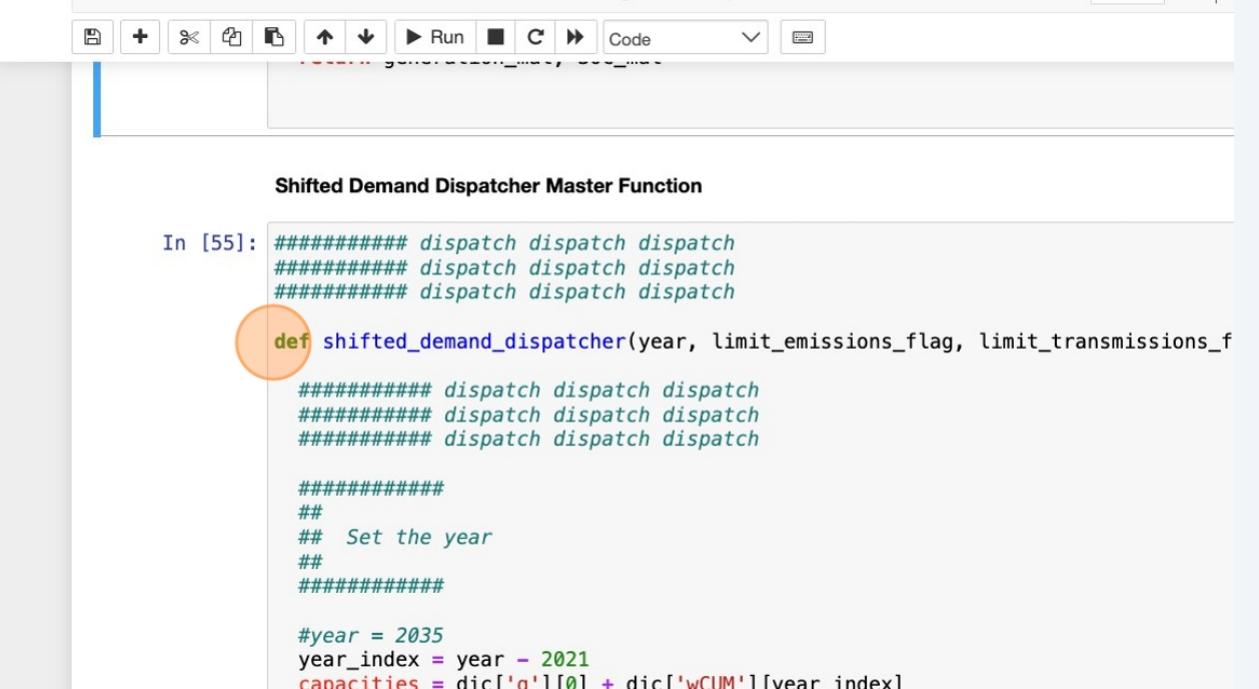
```
x_mat = np.zeros((n,I))
for h in range(H):
    for i in range(I):
        x_mat[h][i] = x[h][i].varValue

num_columns = x_mat.shape[1]
temp = np.concatenate((x_mat[:, :1], x_mat[:, 2:], x_mat[:, 1:2]), axis=1)
x_mat = temp.copy()

y_mat = np.zeros((H,P))
for h in range(H):
    for p in range(P):
        y_mat[h][p] = y[h][p].varValue

dchg_mat = np.zeros((H,S))
for h in range(H):
    for s in range(S):
        dchg_mat[h][s] = dchg[h][s].varValue*dic['dchgeff'][0][s]
        #if sum(x[h][i].varValue for i in range(I)) + sum(y[h][p].varValue for p in range(P)) > S:
            #dchg_mat[h][s] = 0
```

63 Click "def"



Shifted Demand Dispatcher Master Function

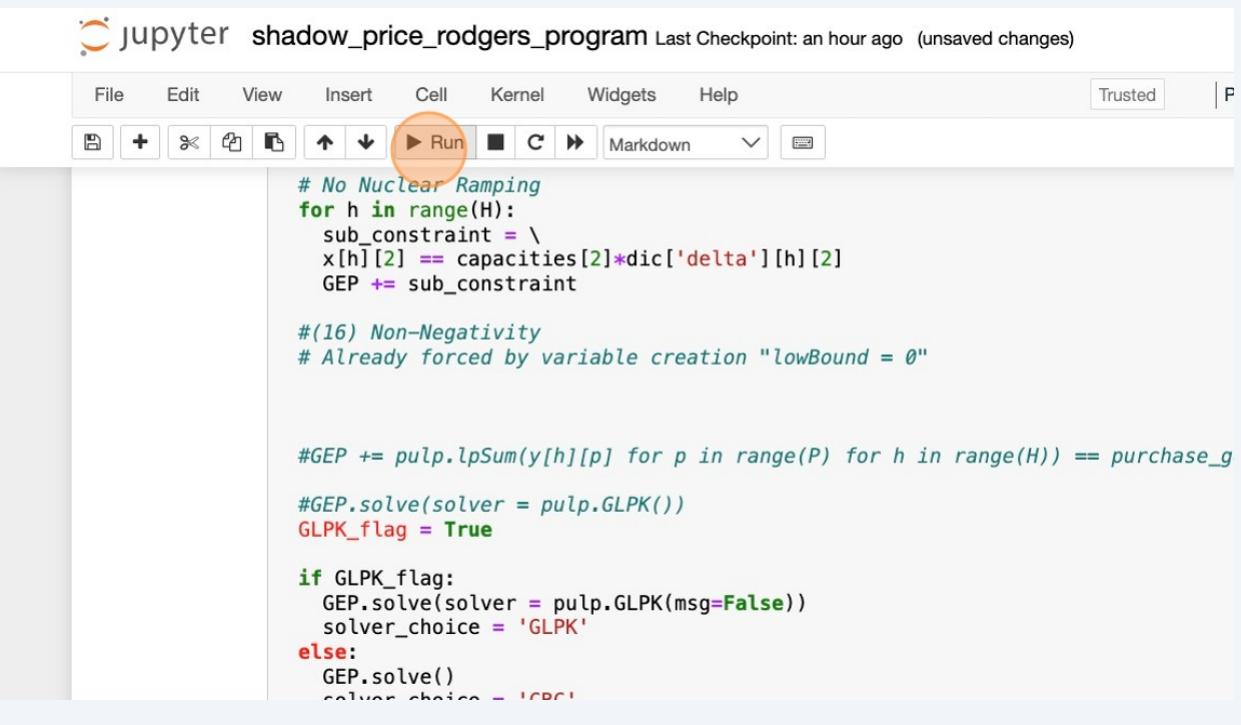
```
In [55]: ###### dispatch dispatch dispatch
##### dispatch dispatch dispatch
##### dispatch dispatch dispatch

def shifted_demand_dispatcher(year, limit_emissions_flag, limit_transmissions_flag):
    ##### dispatch dispatch dispatch
    ##### dispatch dispatch dispatch
    ##### dispatch dispatch dispatch

    #####
    ##
    ## Set the year
    ##
    ####

    #year = 2035
    year_index = year - 2021
    capacities = dic['a101'] + dic['wCLIM1'][year_index]
```

64 Click "Run"



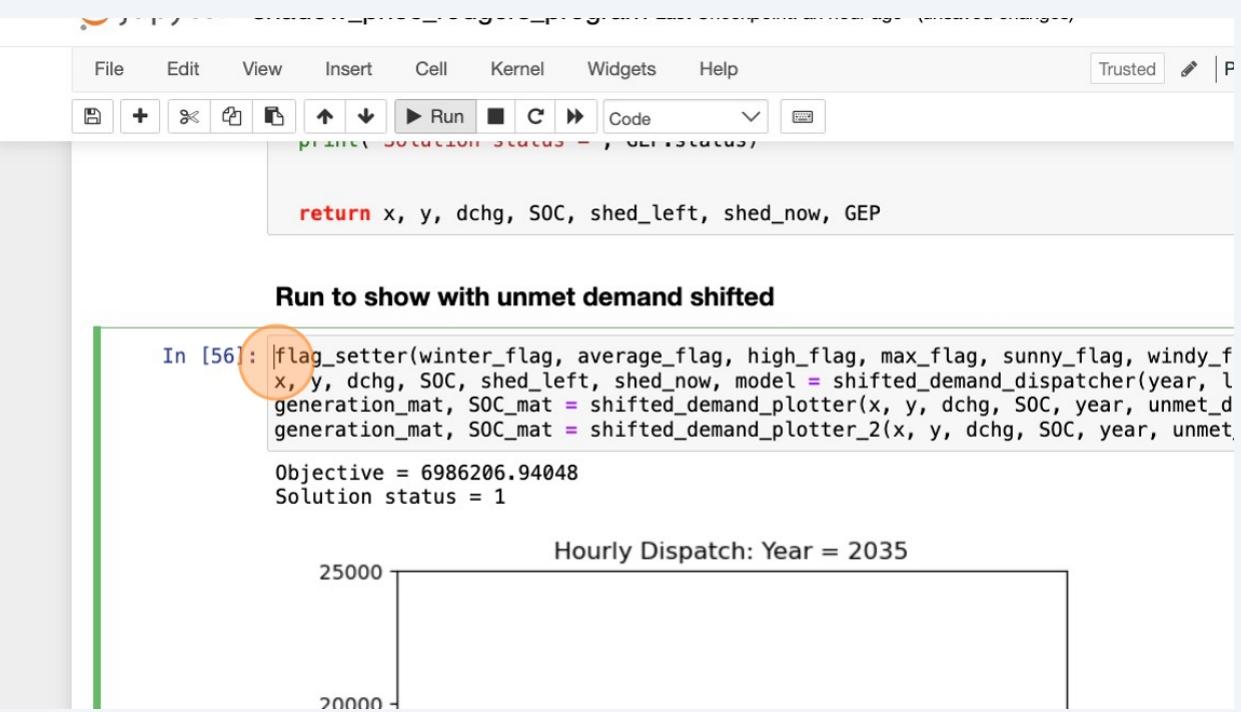
```
# No Nuclear Ramping
for h in range(H):
    sub_constraint = \
        x[h][2] == capacities[2]*dic['delta'][h][2]
    GEP += sub_constraint

#(16) Non-Negativity
# Already forced by variable creation "lowBound = 0"

#GEP += pulp.lpSum(y[h][p] for p in range(P) for h in range(H)) == purchase_g
#GEP.solve(solver = pulp.GLPK())
GLPK_flag = True

if GLPK_flag:
    GEP.solve(solver = pulp.GLPK(msg=False))
    solver_choice = 'GLPK'
else:
    GEP.solve()
    solver_choice = 'CPLEX'
```

65 Click "flag.setter"



```
return x, y, dchg, SOC, shed_left, shed_now, GEP
```

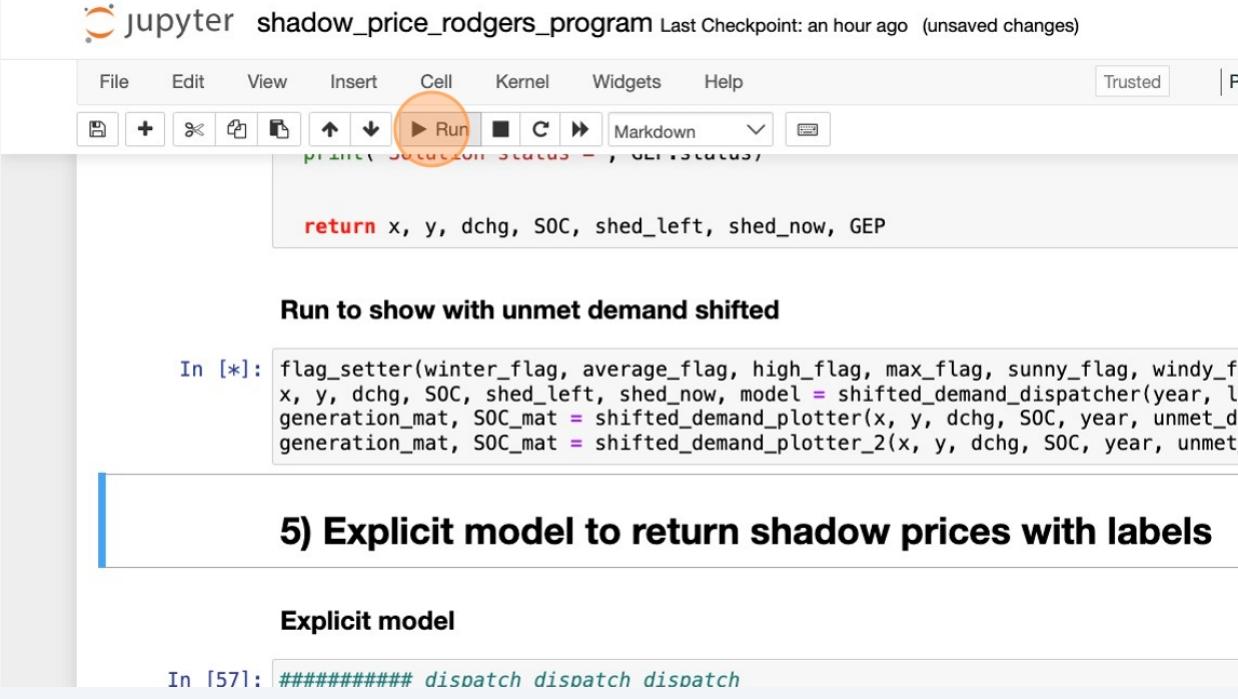
Run to show with unmet demand shifted

```
In [56]: flag.setter(winter_flag, average_flag, high_flag, max_flag, sunny_flag, windy_flag)
x, y, dchg, SOC, shed_left, shed_now, model = shifted_demand_dispatcher(year, l
generation_mat, SOC_mat = shifted_demand_plotter(x, y, dchg, SOC, year, unmet_d
generation_mat, SOC_mat = shifted_demand_plotter_2(x, y, dchg, SOC, year, unmet_d
```

```
Objective = 6986206.94048
Solution status = 1
```



66 Click "Run"



jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted

Run

```
return x, y, dchg, SOC, shed_left, shed_now, GEP
```

Run to show with unmet demand shifted

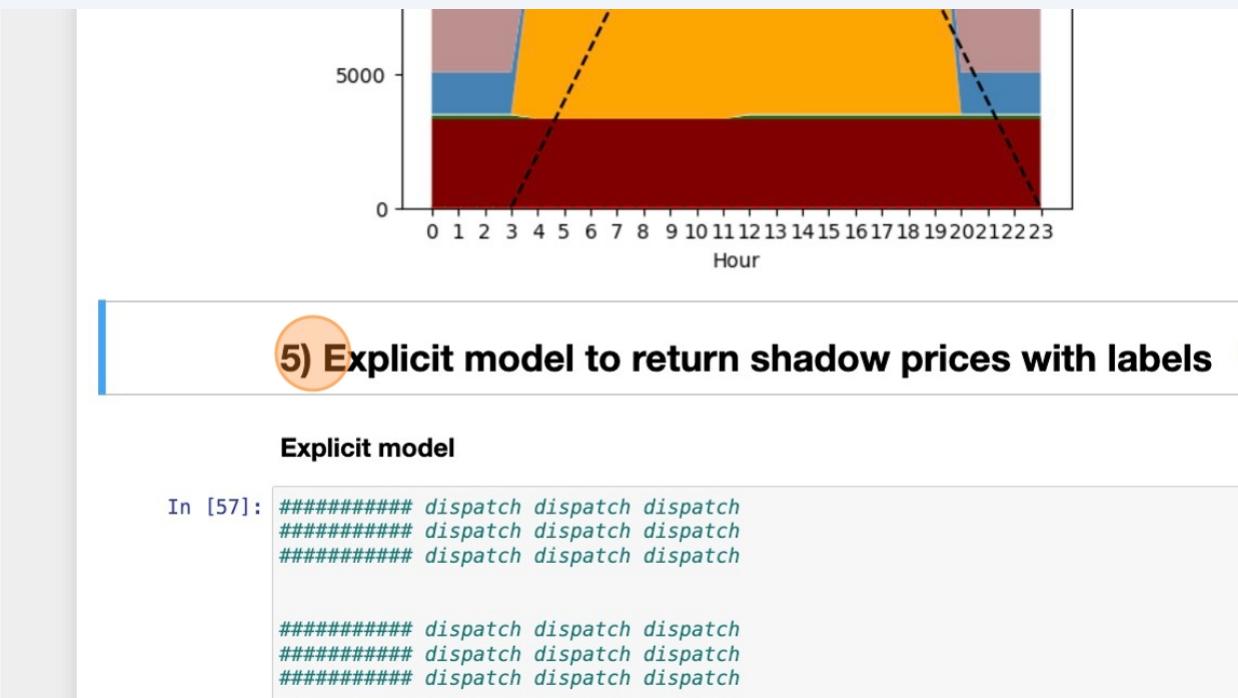
```
In [*]: flag.setter(winter_flag, average_flag, high_flag, max_flag, sunny_flag, windy_flag)
x, y, dchg, SOC, shed_left, shed_now, model = shifted_demand_dispatcher(year, l
generation_mat, SOC_mat = shifted_demand_plotter(x, y, dchg, SOC, year, unmet_d
generation_mat, SOC_mat = shifted_demand_plotter_2(x, y, dchg, SOC, year, unmet_d
```

5) Explicit model to return shadow prices with labels

Explicit model

```
In [57]: ##### dispatch dispatch dispatch
```

67 Click "5) Explicit model to return shadow prices with labels"



68 Click "year_index"

```
##### dispatch dispatch dispatch
##### dispatch dispatch dispatch
##### dispatch dispatch dispatch

#####
## 
## Set the year
##
#####

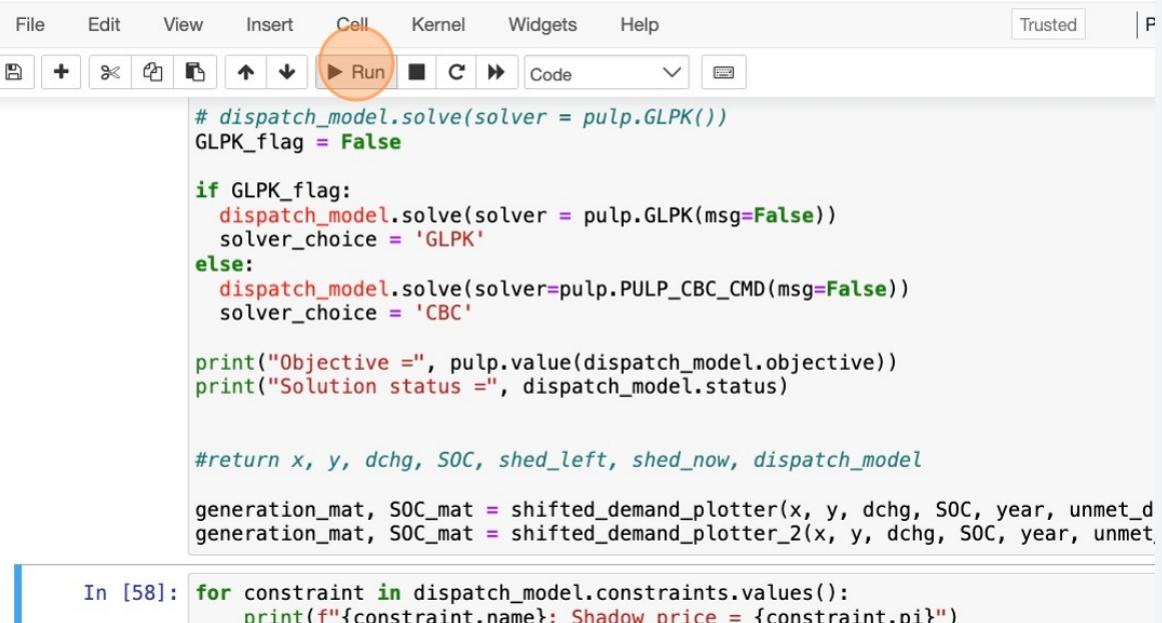
#year = 2035
year_index = year - 2021
capacities = dic['q'][0] + dic['wCUM'][year_index]
demand = dic['d']*(1.007**year_index) - unmet_demand
total_unmet_demand = sum(unmet_demand)[0]

'''##### STORAGE CAPACITY LIMITS'''
dic['qstor'][0][0] = 0*battery_multiplier
dic['qstor'][0][1] = 2000*battery_multiplier
#115445

for s in range(S): # generation capability is based on capacity and battery len
    dic['SOCMAX'][0][s] = dic['qstor'].copy()[0][s]*dic['length'][0][s]
dic['PCMAY'] = dic['SOCMAX']
dic['PDMAX'] = dic['SOCMAX']
```

69 Click "Run"

jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)



```
# dispatch_model.solve(solver = pulp.GLPK())
GLPK_flag = False

if GLPK_flag:
    dispatch_model.solve(solver = pulp.GLPK(msg=False))
    solver_choice = 'GLPK'
else:
    dispatch_model.solve(solver=pulp.PULP_CBC_CMD(msg=False))
    solver_choice = 'CBC'

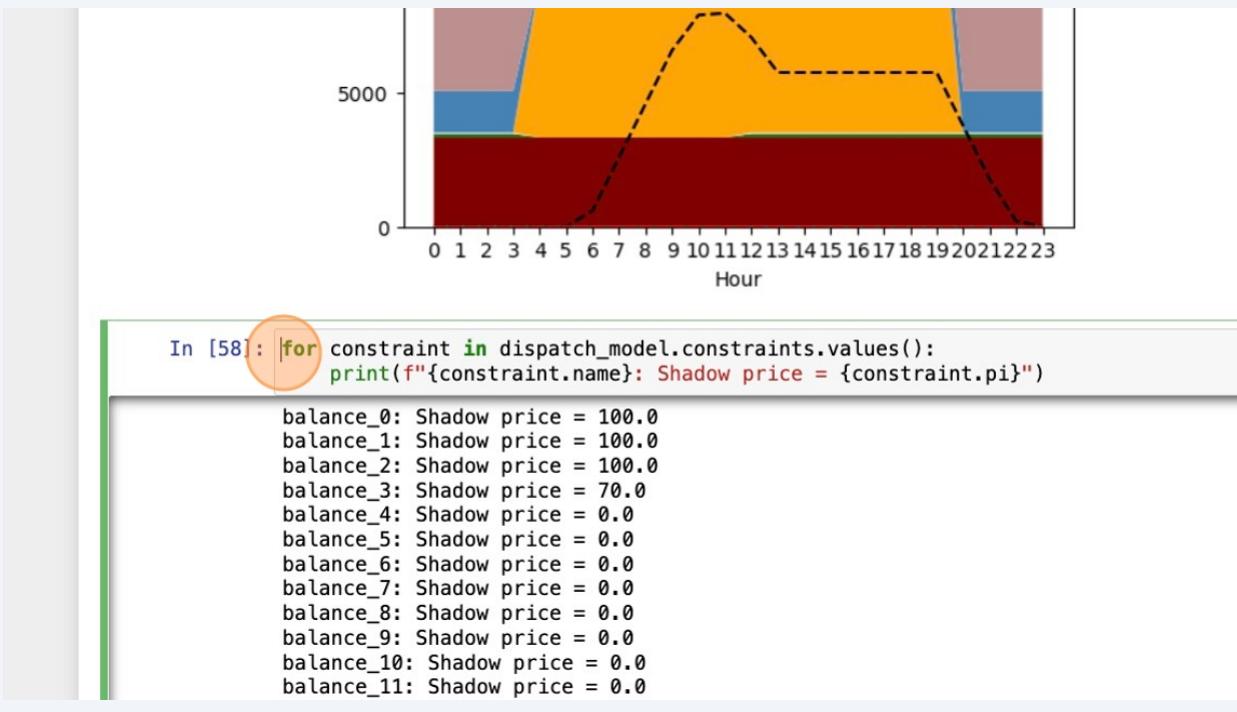
print("Objective =", pulp.value(dispatch_model.objective))
print("Solution status =", dispatch_model.status)

#return x, y, dchg, SOC, shed_left, shed_now, dispatch_model

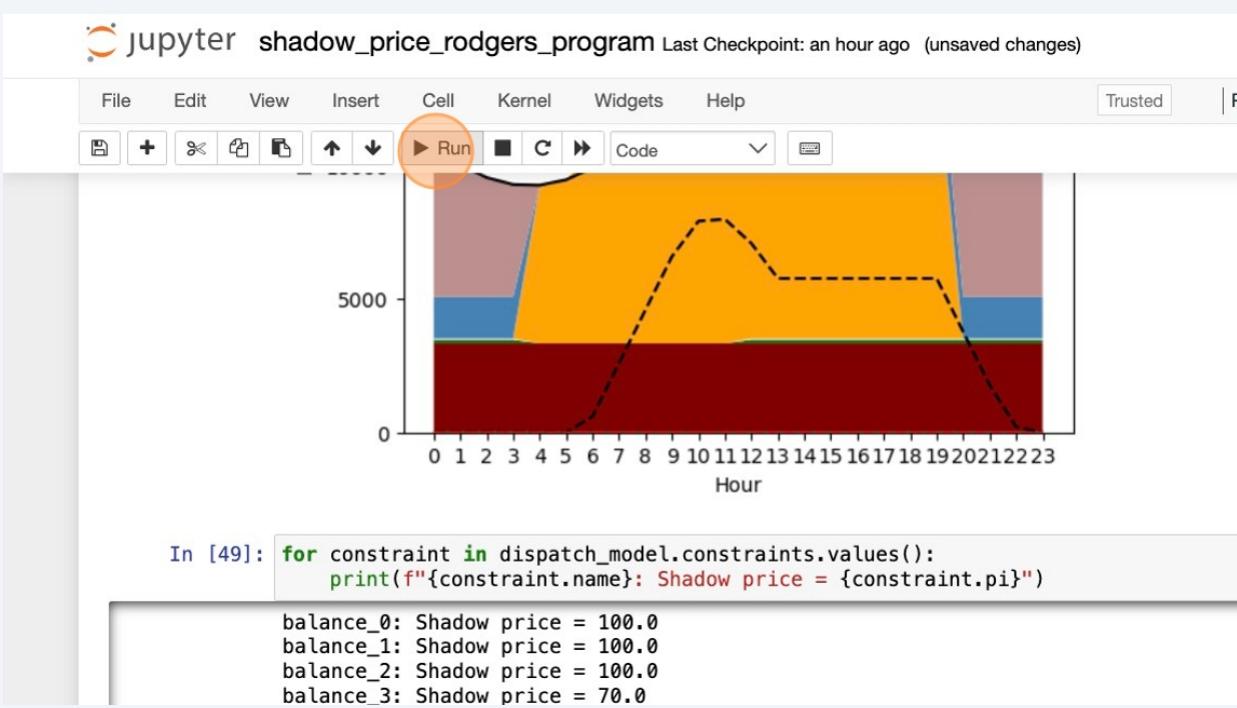
generation_mat, SOC_mat = shifted_demand_plotter(x, y, dchg, SOC, year, unmet_d
generation_mat, SOC_mat = shifted_demand_plotter_2(x, y, dchg, SOC, year, unmet
```

In [58]: `for constraint in dispatch_model.constraints.values():
 print(f"{constraint.name}: Shadow price = {constraint.pi}")`

70 Click "for"



71 Click "Run"



72

Click "color_list"

```
generation_capacity_1_0: Shadow price = -98.54
generation_capacity_2_0: Shadow price = -98.54
generation_capacity_3_0: Shadow price = -68.54
generation_capacity_4_0: Shadow price = 0.0
generation_capacity_5_0: Shadow price = 0.0
generation_capacity_6_0: Shadow price = 0.0
generation_capacity_7_0: Shadow price = 0.0
generation_capacity_8_0: Shadow price = 0.0
generation_capacity_9_0: Shadow price = 0.0
generation_capacity_10_0: Shadow price = 0.0
generation_capacity_11_0: Shadow price = 0.0
generation_capacity_12_0: Shadow price = -98.54
```

```
In [59]: tech_list = ["HYD", "NUC", "BIO", "PET", "SOL", "WINON", "WINOFF", "FC", "NAT", "PJM"
color_list = ["darkblue", "maroon", "darkgreen", "snow", "orange", "skyblue", "steel"]
```

```
In [60]: dispatch_model
```

```
Out[60]: dispatch_model_Problem:
MINIMIZE
140*shed_now_0 + 130*shed_now_1 + 60*shed_now_10 + 70*shed_now_11 + 80*shed_now_
d_now_13 + 100*shed_now_14 + 110*shed_now_15 + 120*shed_now_16 + 130*shed_now_1_
now_18 + 150*shed_now_19 + 120*shed_now_2 + 160*shed_now_20 + 170*shed_now_21
ow_22 + 190*shed_now_23 + 50*shed_now_3 + 100*shed_now_4 + 90*shed_now_5 + 80*s
70*shed_now_7 + 60*shed_now_8 + 50*shed_now_9 + 1.46*x_{0,0} + 1.96*x_{0,1} + 2
+ 5.06*x_{0,3} + 4.92*x_{0,4} + 0.62*x_{0,8} + 1.46*x_{1,0} + 1.96*x_{1,1} + 2.
5.06*x_{1,3} + 4.92*x_{1,4} + 0.62*x_{1,8} + 1.46*x_{10,0} + 1.96*x_{10,1} + 2.
```

73

Click "Run"

jupyter shadow_price Rodgers_program Last Checkpoint: an hour ago (unsaved changes)

```
File Edit View Insert Cell Kernel Widgets Help Trusted | F

print(f"{constraint.name}: Shadow price = {constraint.pi}")
balance_17: Shadow price = 100.0
balance_18: Shadow price = 100.0
balance_19: Shadow price = 100.0
balance_20: Shadow price = 150.0
balance_21: Shadow price = 150.0
balance_22: Shadow price = 100.0
balance_23: Shadow price = 100.0
generation_capacity_0_0: Shadow price = -98.54
generation_capacity_1_0: Shadow price = -98.54
generation_capacity_2_0: Shadow price = -98.54
generation_capacity_3_0: Shadow price = -68.54
generation_capacity_4_0: Shadow price = 0.0
generation_capacity_5_0: Shadow price = 0.0
generation_capacity_6_0: Shadow price = 0.0
generation_capacity_7_0: Shadow price = 0.0
generation_capacity_8_0: Shadow price = 0.0
generation_capacity_9_0: Shadow price = 0.0
generation_capacity_10_0: Shadow price = 0.0
generation_capacity_11_0: Shadow price = 0.0
generation_capacity_12_0: Shadow price = -98.54
```

74

Click "dispatch_model"

```

generation_capacity_5_0: Shadow price = 0.0
generation_capacity_6_0: Shadow price = 0.0
generation_capacity_7_0: Shadow price = 0.0
generation_capacity_8_0: Shadow price = 0.0
generation_capacity_9_0: Shadow price = 0.0
generation_capacity_10_0: Shadow price = 0.0
generation_capacity_11_0: Shadow price = 0.0
generation_capacity_12_0: Shadow price = -98.54

In [50]: tech_list = ["HYD", "NUC", "BIO", "PET", "SOL", "WINON", "WINOFF", "FC", "NAT", "PJM"]
color_list = ["darkblue", "maroon", "darkgreen", "snow", "orange", "skyblue", "steel"

```

In [60]: **dispatch_model**

```

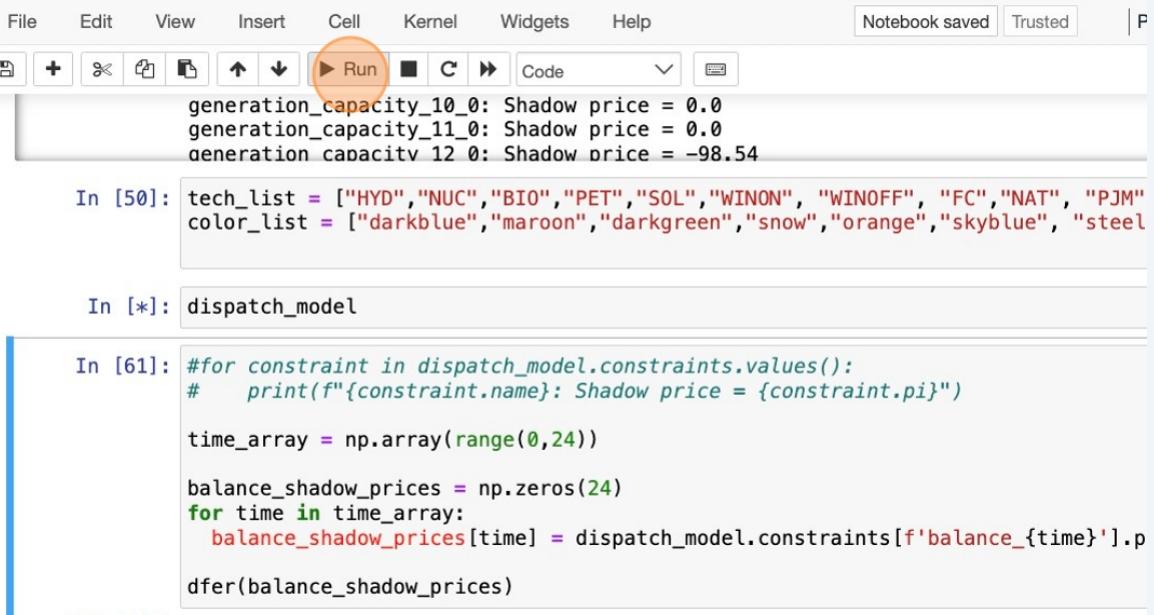
Out[60]: dispatch_model_Problem:
MINIMIZE
140*shed_now_0 + 130*shed_now_1 + 60*shed_now_10 + 70*shed_now_11 + 80*shed_now_
d_now_13 + 100*shed_now_14 + 110*shed_now_15 + 120*shed_now_16 + 130*shed_now_1_
now_18 + 150*shed_now_19 + 120*shed_now_2 + 160*shed_now_20 + 170*shed_now_21
ow_22 + 190*shed_now_23 + 50*shed_now_3 + 100*shed_now_4 + 90*shed_now_5 + 80*s
70*shed_now_7 + 60*shed_now_8 + 50*shed_now_9 + 1.46*x_{0,0} + 1.96*x_{0,1} + 2
+ 5.06*x_{0,3} + 4.92*x_{0,4} + 0.62*x_{0,8} + 1.46*x_{1,0} + 1.96*x_{1,1} + 2.
5.06*x_{1,3} + 4.92*x_{1,4} + 0.62*x_{1,8} + 1.46*x_{10,0} + 1.96*x_{10,1} + 2.
+ 5.06*x_{10,3} + 4.92*x_{10,4} + 0.62*x_{10,8} + 1.46*x_{11,0} + 1.96*x_{11,1}
1,2} + 5.06*x_{11,3} + 4.92*x_{11,4} + 0.62*x_{11,8} + 1.46*x_{12,0} + 1.96*x_{*
*x_{12,2} + 5.06*x_{12,3} + 4.92*x_{12,4} + 0.62*x_{12,8} + 1.46*x_{13,0} + 1.9
2.48*x_{13,2} + 5.06*x_{13,3} + 4.92*x_{13,4} + 0.62*x_{13,8} + 1.46*x_{14,0} +

```

75

Click "Run"

jupyter shadow_price Rodgers_program Last Checkpoint: an hour ago (unsaved changes)



File Edit View Insert Cell Kernel Widgets Help Notebook saved Trusted P

generation_capacity_10_0: Shadow price = 0.0
generation_capacity_11_0: Shadow price = 0.0
generation_capacity_12_0: Shadow price = -98.54

```

In [50]: tech_list = ["HYD", "NUC", "BIO", "PET", "SOL", "WINON", "WINOFF", "FC", "NAT", "PJM"]
color_list = ["darkblue", "maroon", "darkgreen", "snow", "orange", "skyblue", "steel"

```

In [*]: **dispatch_model**

```

In [61]: #for constraint in dispatch_model.constraints.values():
#    print(f'{constraint.name}: Shadow price = {constraint.pi}')

time_array = np.array(range(0,24))

balance_shadow_prices = np.zeros(24)
for time in time_array:
    balance_shadow_prices[time] = dispatch_model.constraints[f'balance_{time}'].p
dfer(balance_shadow_prices)

```

76 Click "time_array"

```
1,2} + 5.06*x_{11,3} + 4.92*x_{11,4} + 0.62*x_{11,8} + 1.46*x_{12,0} + 1.96*x_{  
*x_{12,2} + 5.06*x_{12,3} + 4.92*x_{12,4} + 0.62*x_{12,8} + 1.46*x_{13,0} + 1.9  
2.48*x_{13,2} + 5.06*x_{13,3} + 4.92*x_{13,4} + 0.62*x_{13,8} + 1.46*x_{14,0} +  
1} + 2.48*x_{14,2} + 5.06*x_{14,3} + 4.92*x_{14,4} + 0.62*x_{14,8} + 1.46*x_{15  
_{15,1} + 2.48*x_{15,2} + 5.06*x_{15,3} + 4.92*x_{15,4} + 0.62*x_{15,8} + 1.46*  
1.96*x_{16,1} + 2.48*x_{16,2} + 5.06*x_{16,3} + 4.92*x_{16,4} + 0.62*x_{16,8} +  
0} + 1.96*x_{17,1} + 2.48*x_{17,2} + 5.06*x_{17,3} + 4.92*x_{17,4} + 0.62*x_{17  
_{18,0} + 1.96*x_{18,1} + 2.48*x_{18,2} + 5.06*x_{18,3} + 4.92*x_{18,4} + 0.62*  
1.46*x_{19,0} + 1.96*x_{19,1} + 2.48*x_{19,2} + 5.06*x_{19,3} + 4.92*x_{19,4} +  
0} + 1.46*x_{20,0} + 1.96*x_{20,1} + 2.48*x_{20,2} + 5.06*x_{20,3} + 4.92*x_{20,4} +  
0} + 1.46*x_{21,0} + 1.96*x_{21,1} + 2.48*x_{21,2} + 5.06*x_{21,3} + 4.92*x_{21,4} +
```

```
In [61]: #for constraint in dispatch_model.constraints.values():  
#    print(f"{constraint.name}: Shadow price = {constraint.pi}")  
  
time_array = np.array(range(0,24))  
  
balance_shadow_prices = np.zeros(24)  
for time in time_array:  
    balance_shadow_prices[time] = dispatch_model.constraints[f'balance_{time}'].p  
dfer(balance_shadow_prices)
```

Out [61]:

0
0 100.000
1 100.000
2 100.000

77 Click "Run"

jupyter shadow_price_rodgers_program Last Checkpoint: an hour ago (unsaved changes)

```
File Edit View Insert Cell Kernel Widgets Help Trusted | P  
Run  
generation_capacity_0_0: Shadow price = -98.54  
generation_capacity_1_0: Shadow price = -98.54  
generation_capacity_2_0: Shadow price = -98.54  
generation_capacity_3_0: Shadow price = -68.54  
generation_capacity_4_0: Shadow price = 0.0  
generation_capacity_5_0: Shadow price = 0.0  
generation_capacity_6_0: Shadow price = 0.0  
generation_capacity_7_0: Shadow price = 0.0  
generation_capacity_8_0: Shadow price = 0.0  
generation_capacity_9_0: Shadow price = 0.0  
generation_capacity_10_0: Shadow price = 0.0  
generation_capacity_11_0: Shadow price = 0.0  
generation_capacity_12_0: Shadow price = -98.54  
  
In [50]: tech_list = ["HYD", "NUC", "BIO", "PET", "SOL", "WINON", "WINOFF", "FC", "NAT", "PJM"  
color_list = ["darkblue", "maroon", "darkgreen", "snow", "orange", "skyblue", "steel  
  
In [51]: dispatch_model  
  
Out[51]: dispatch_model_Problem:  
MINIMIZE
```