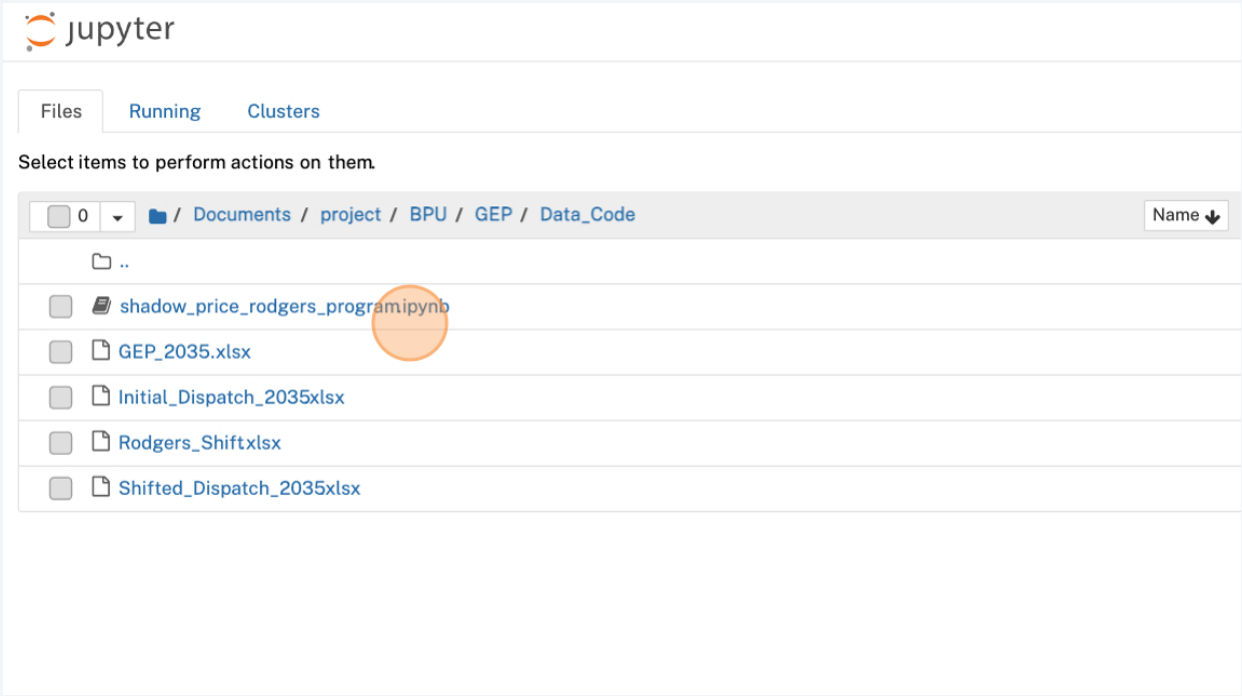
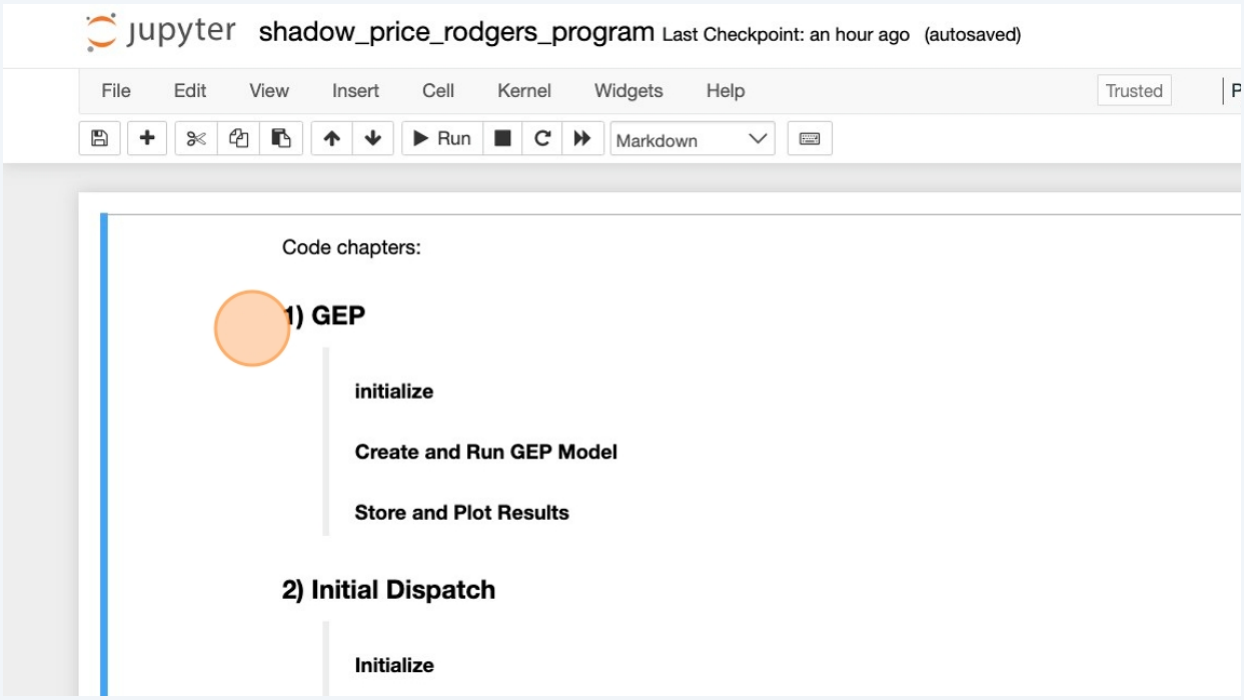
Running the GEP Data Code in Jupyter Notebook

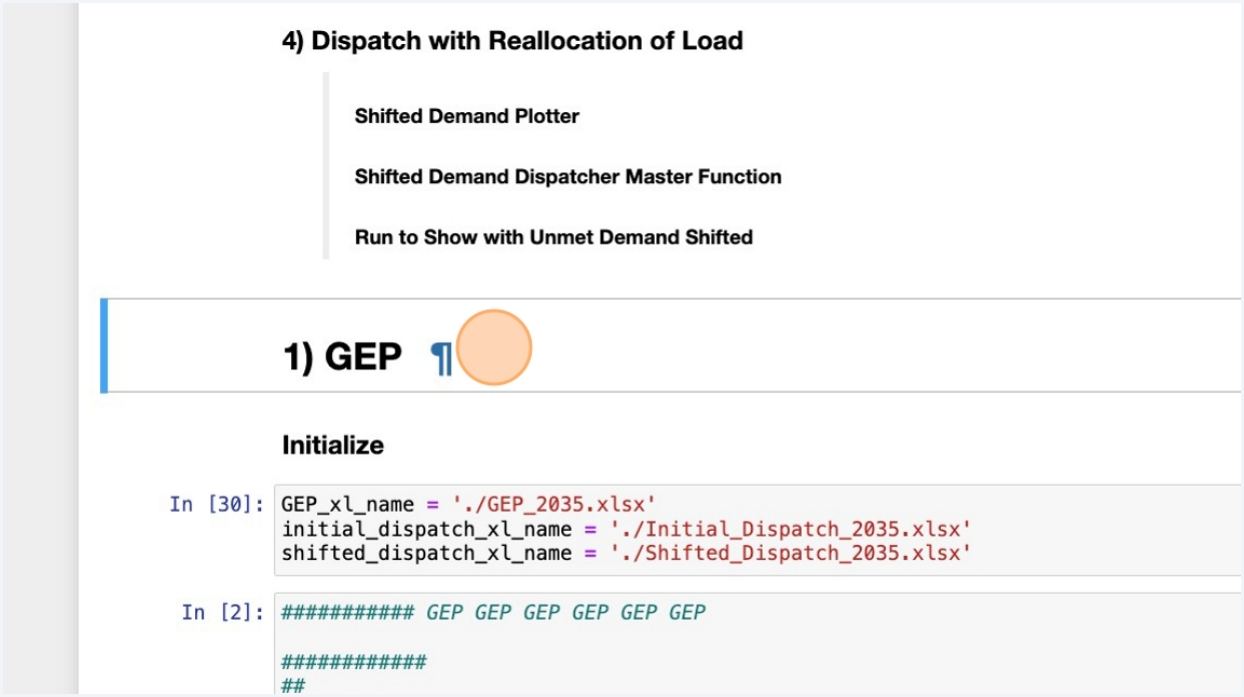
1 Click "shadow_price_rodgers_program.ipynb"



2 Click here.



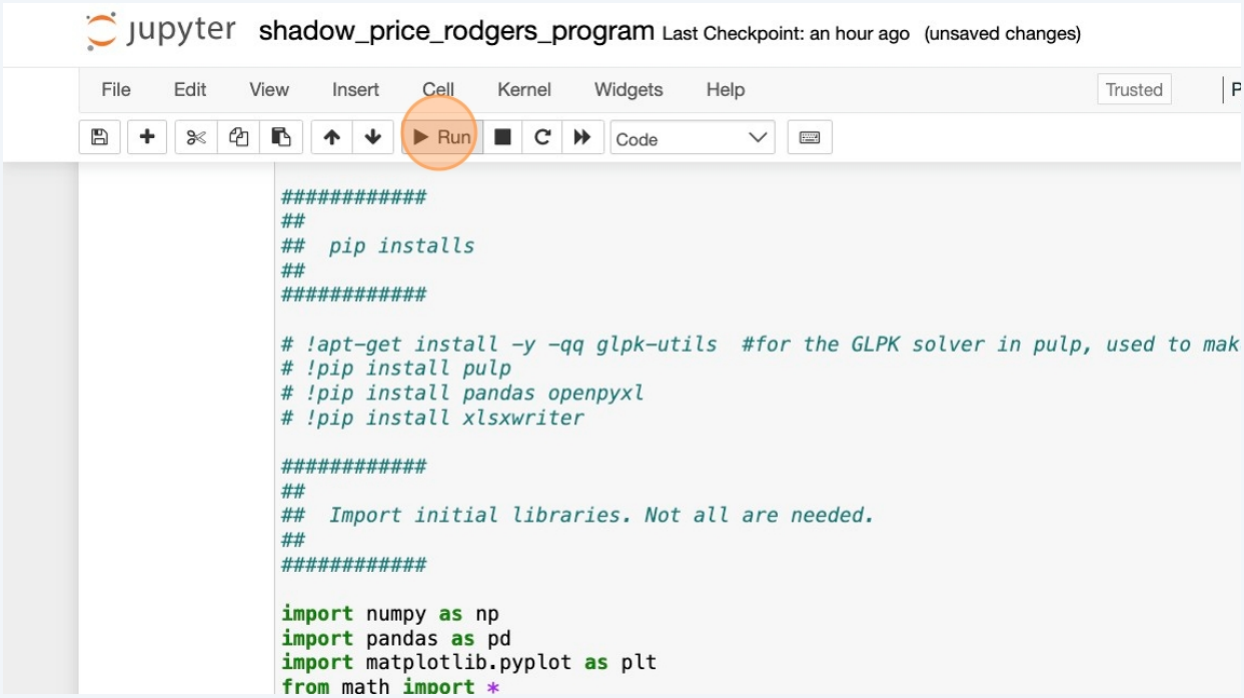
3 Click "1) GEP¶"



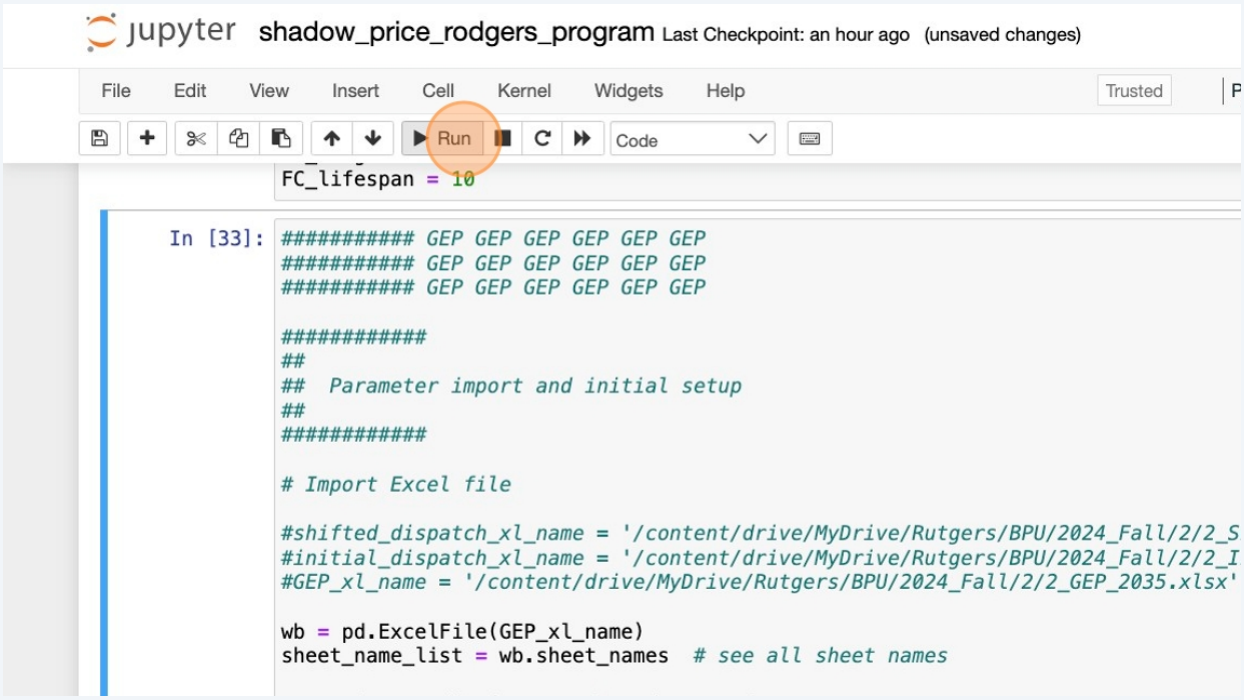
4 Click "import"

```
##
############

# !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 pulp
```

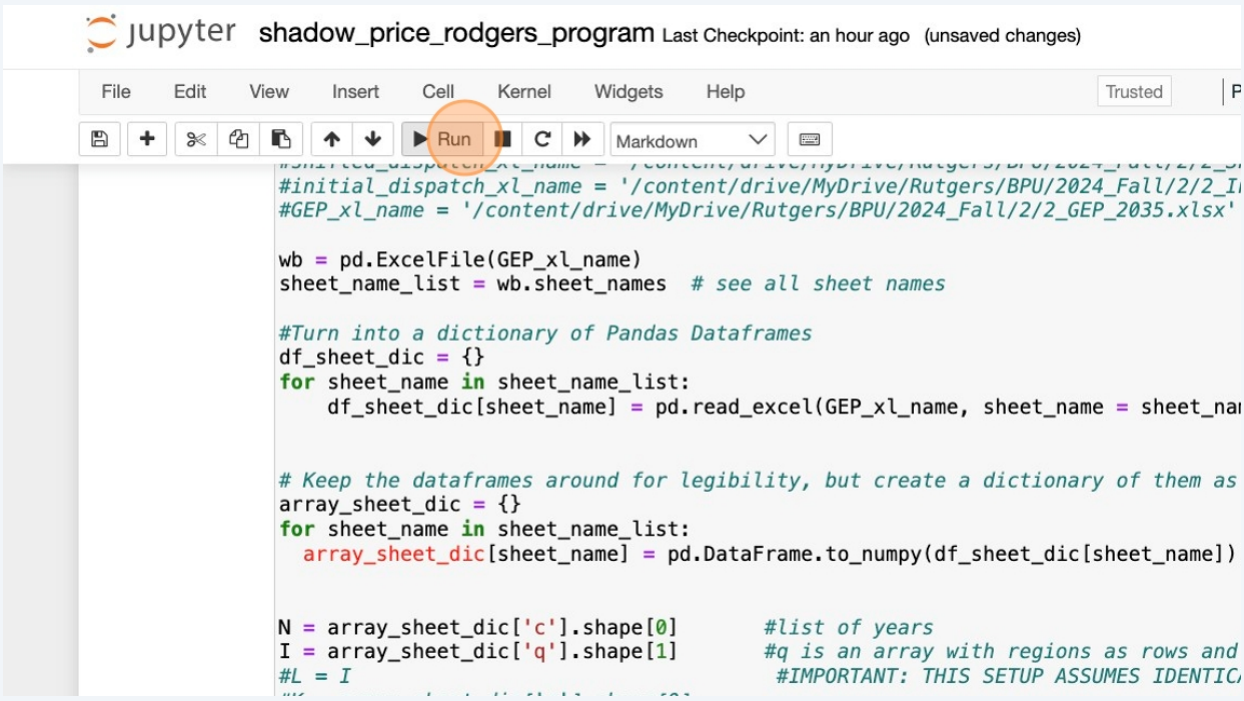
5 Click "Run"



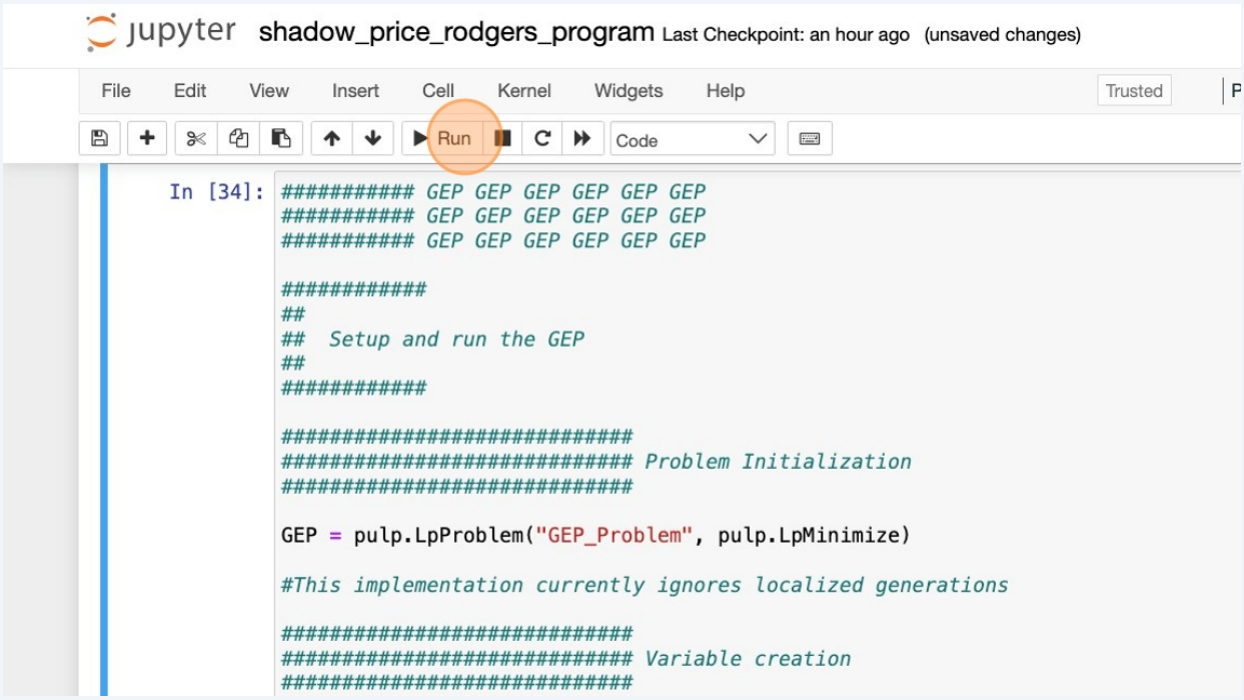
6 Click "Run"



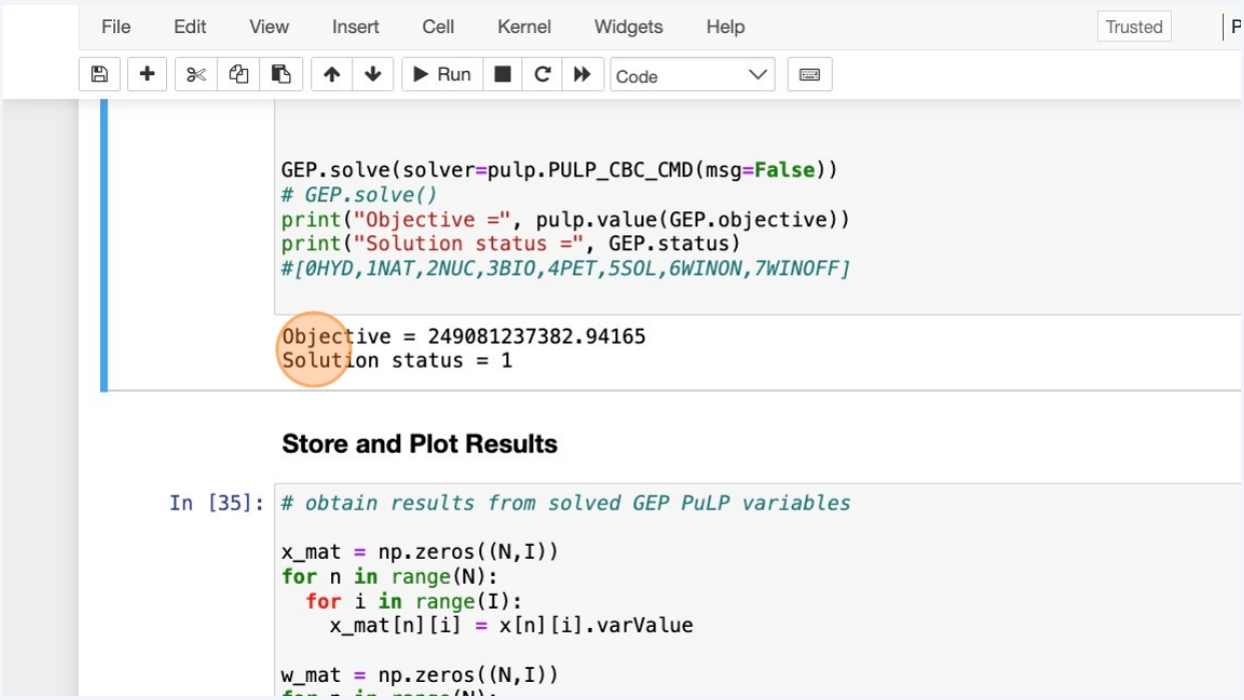
7 Click "Run"



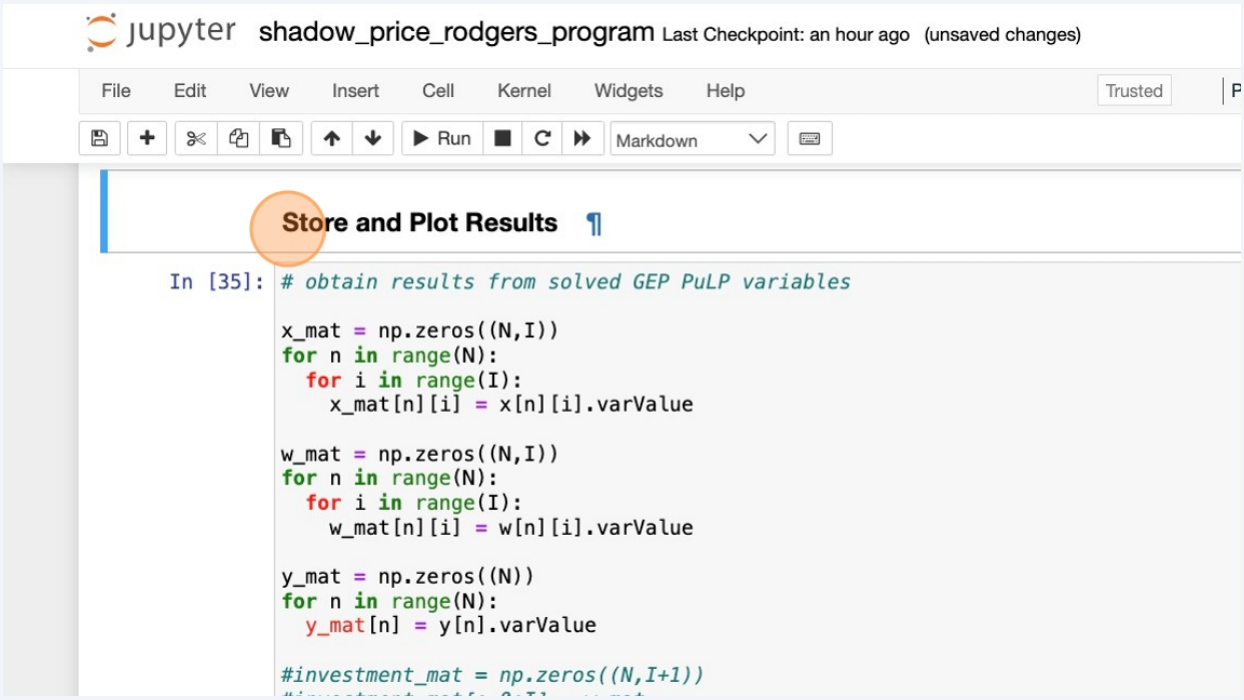
8 Click "Run"



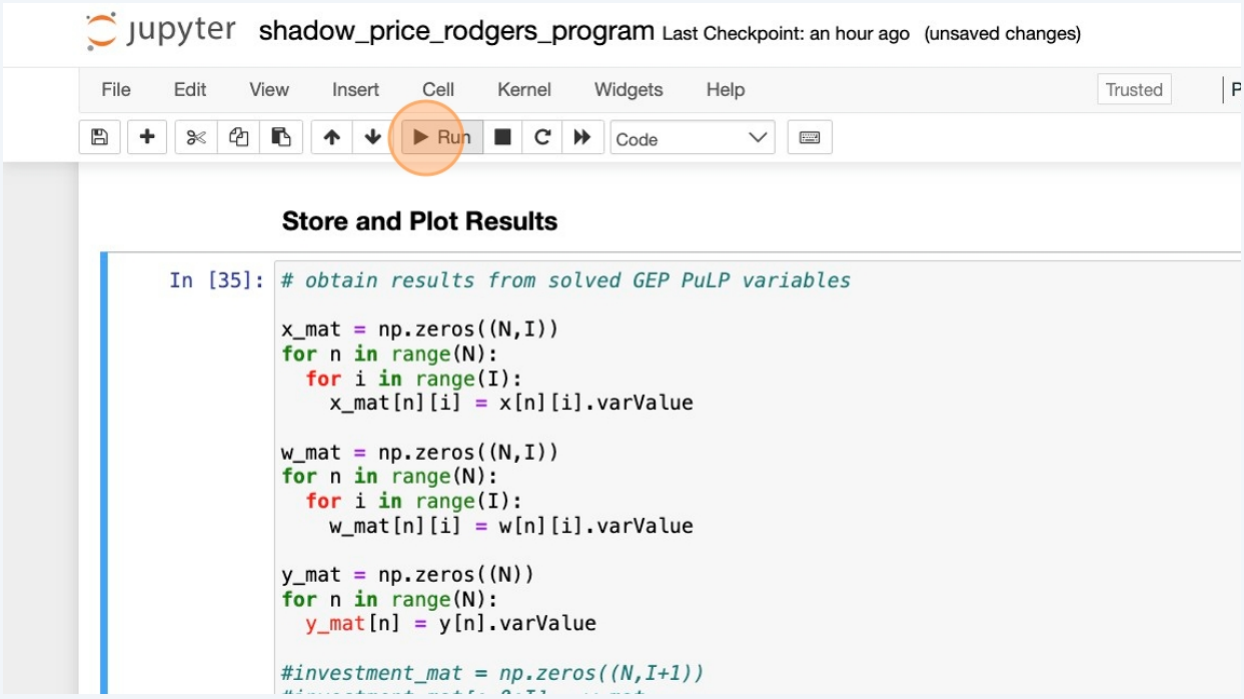
9 Click "Objective = 249081237382.94165 Solution status = 1"



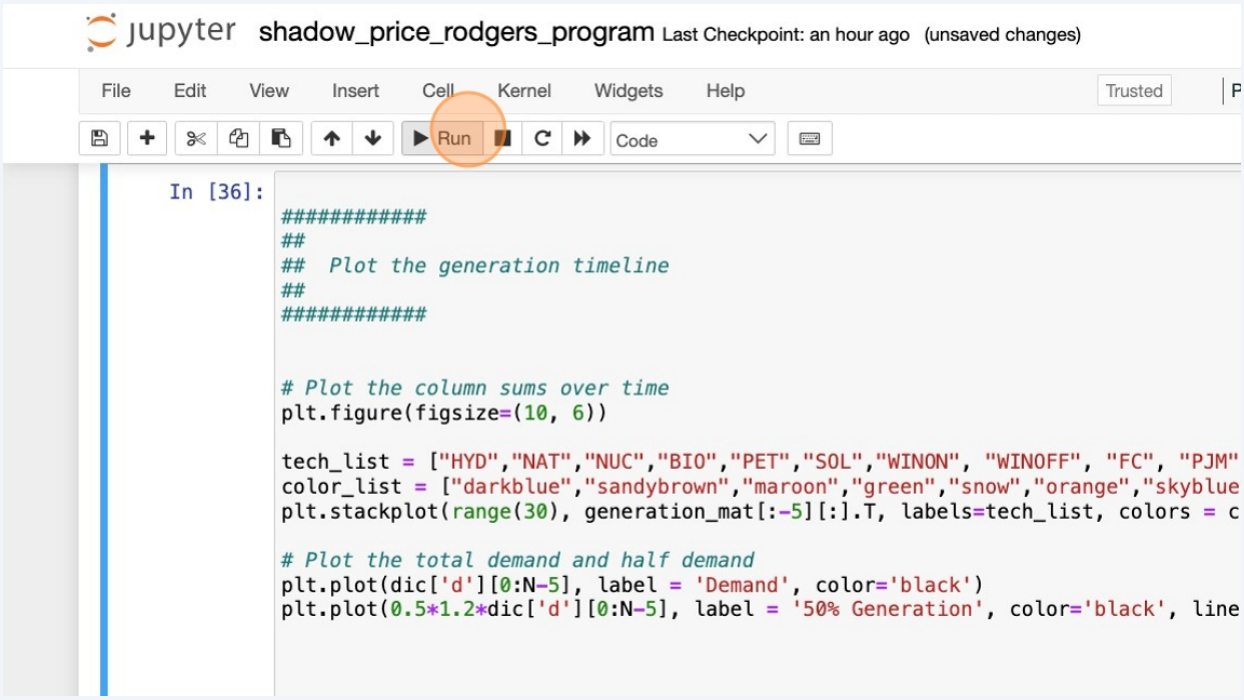
10 Click "Store and Plot Results¶"



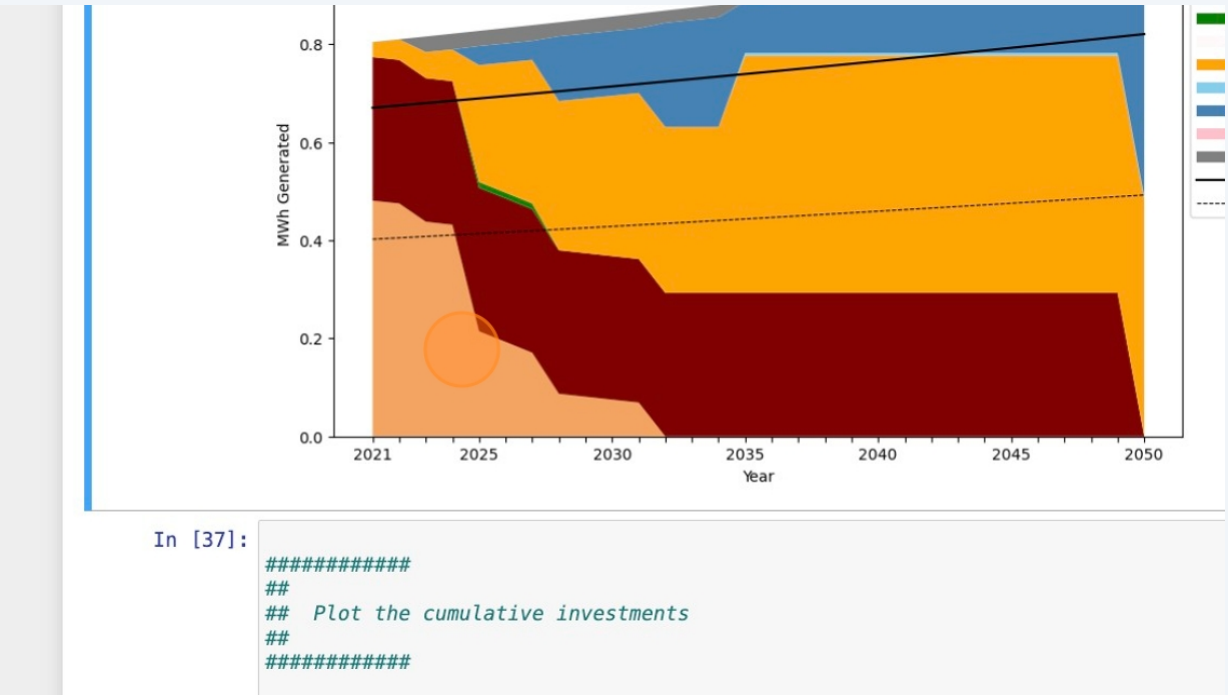
11 Click here.



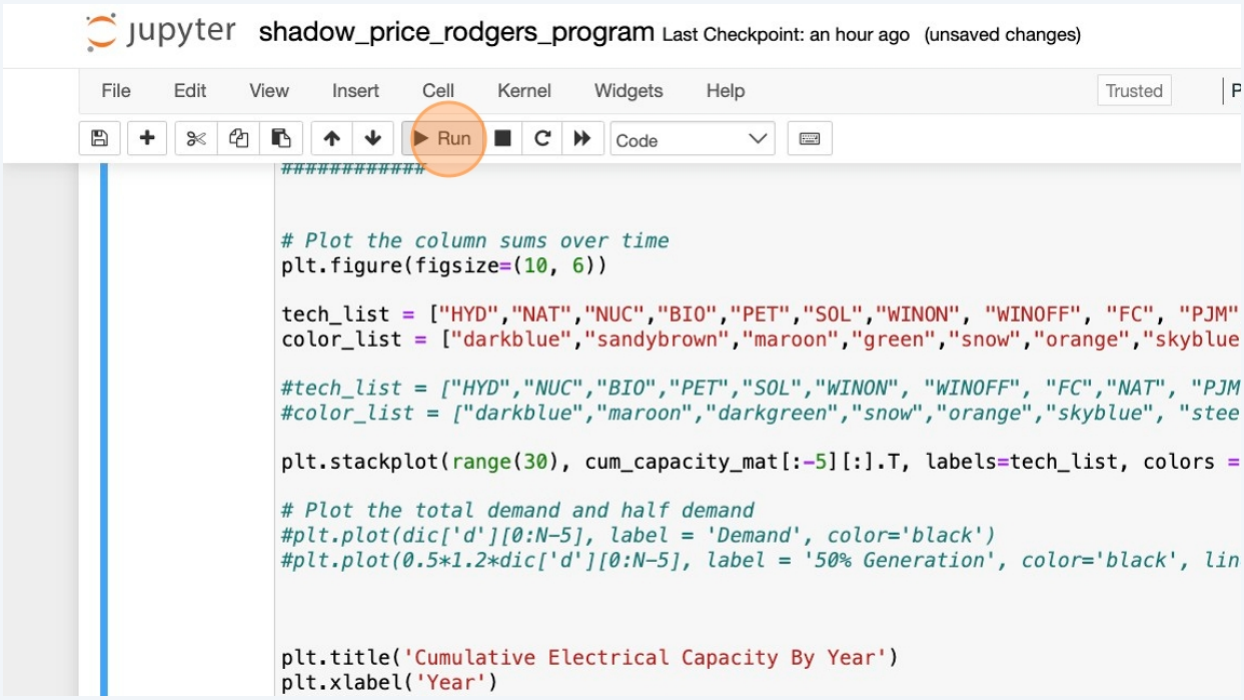
12 Click "Run"



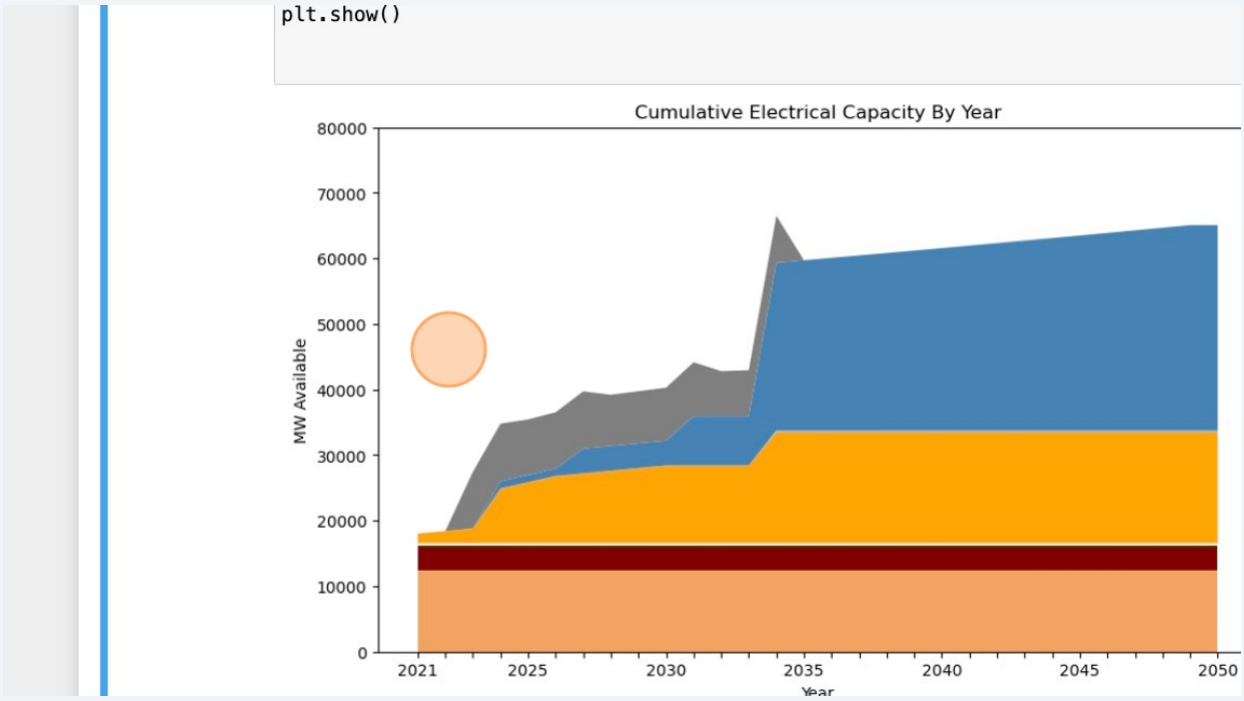
13 Click this image.



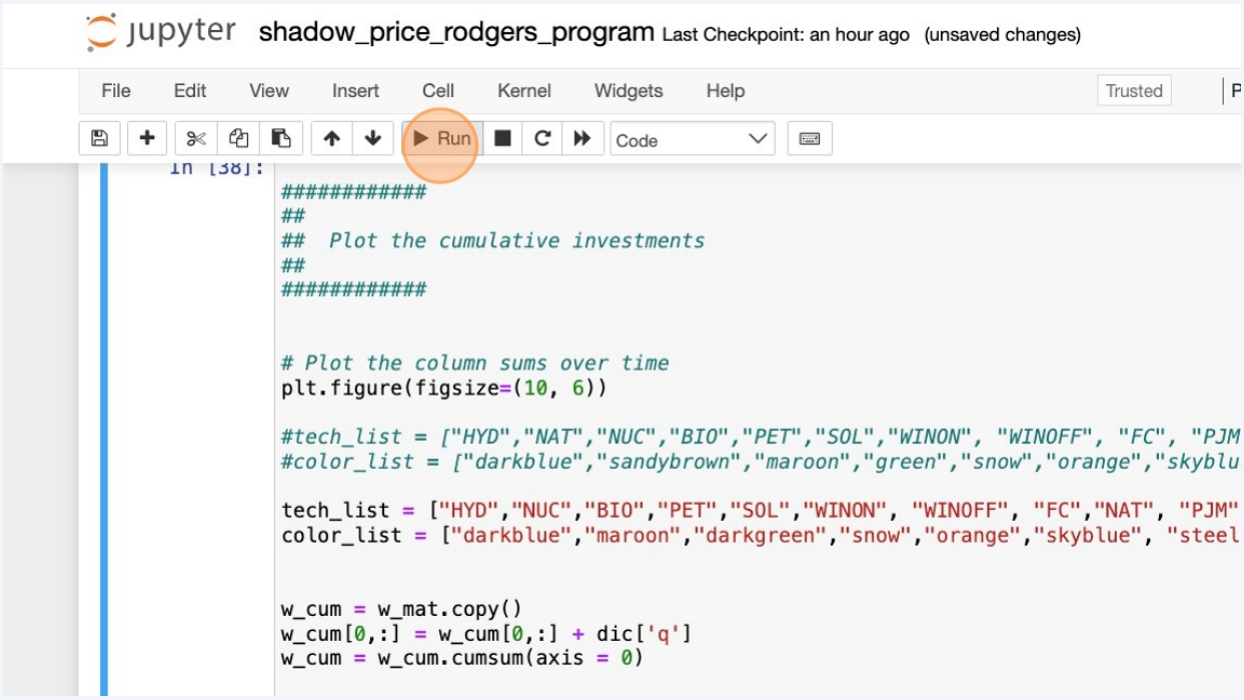
14 Click "Run"



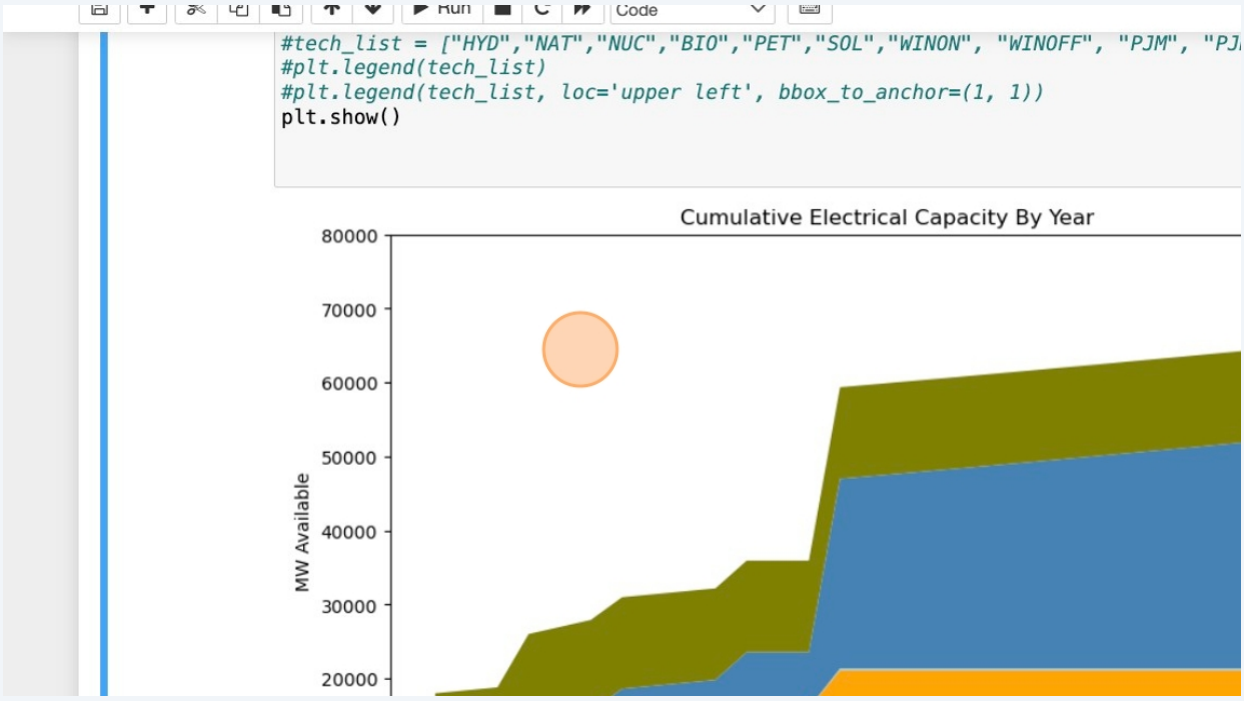
15 Click this image.



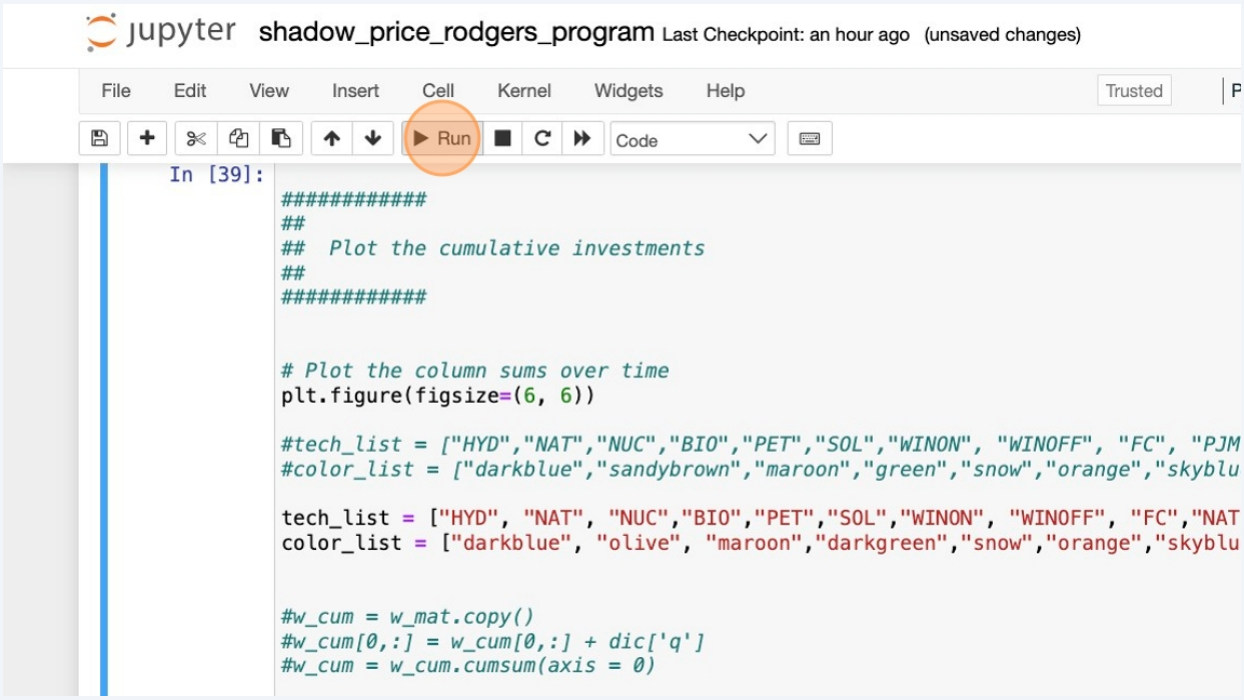
16 Click "Run"



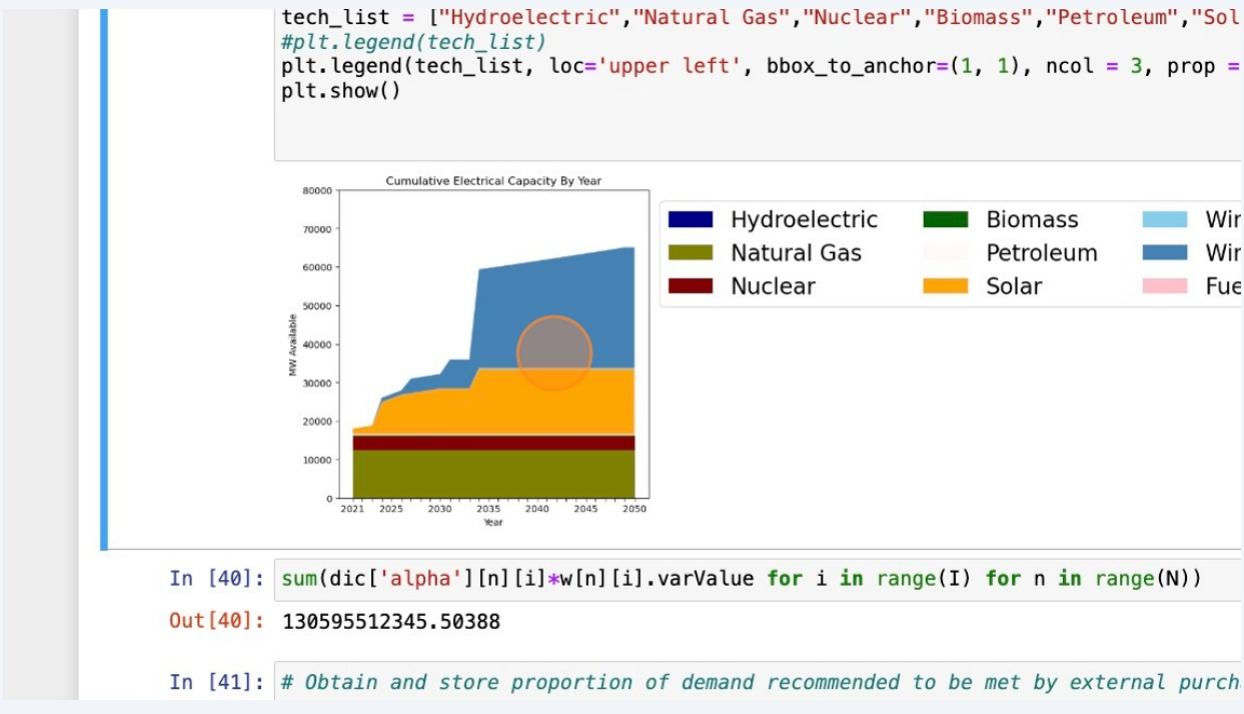
17 Click this image.



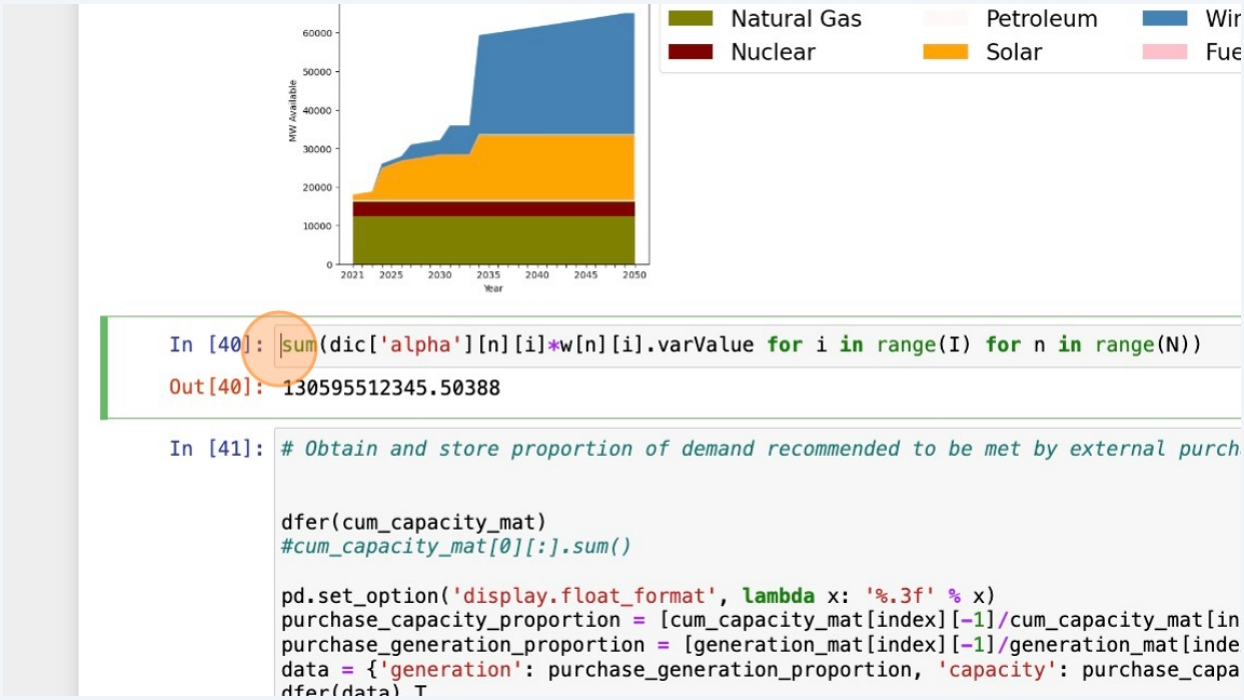
18 Click "Run"



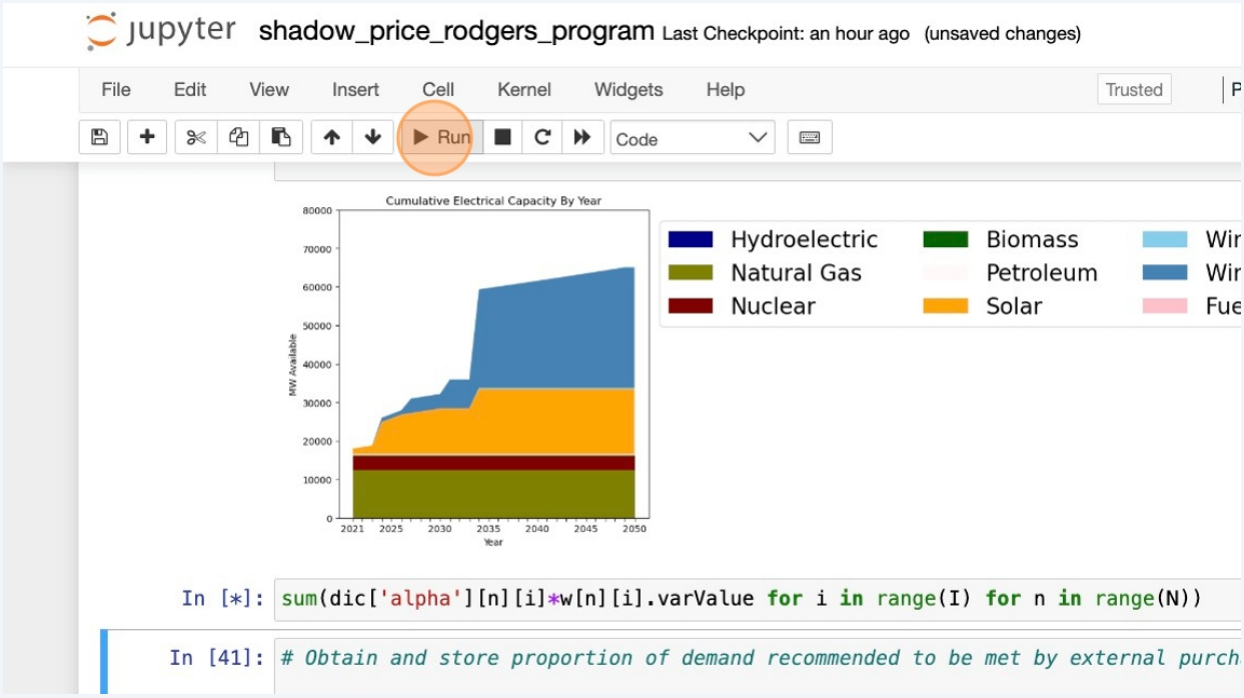
19 Click this image.



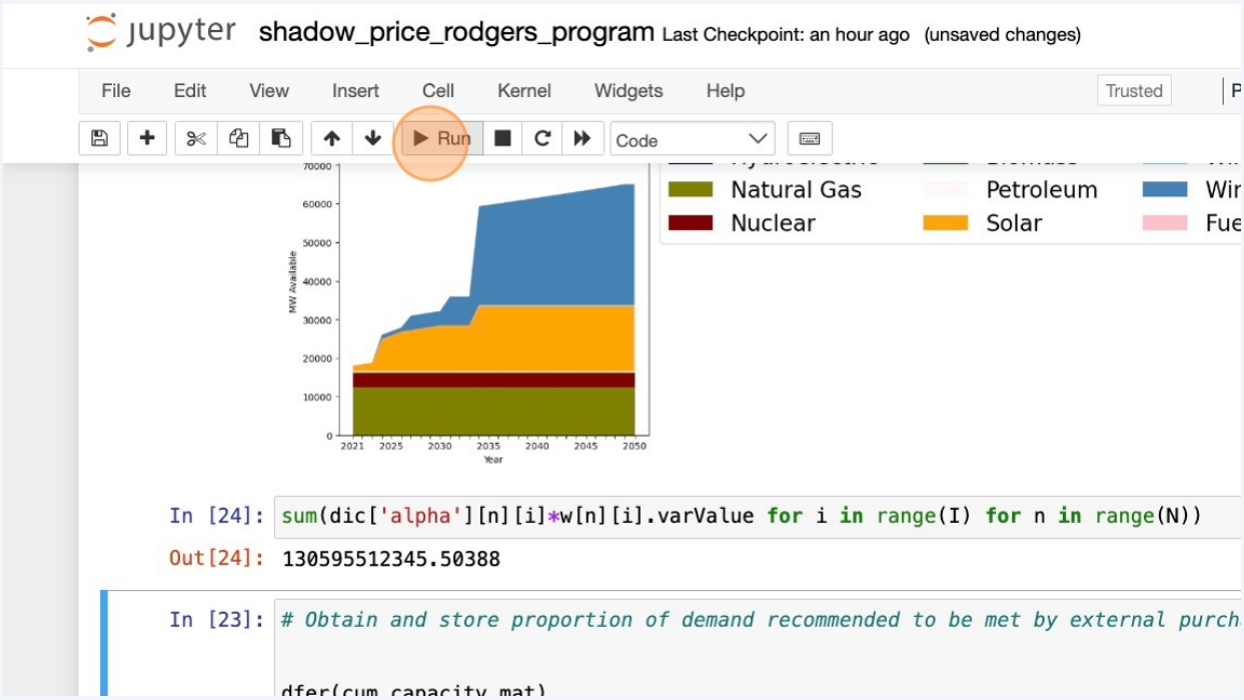
20 Click "sum"



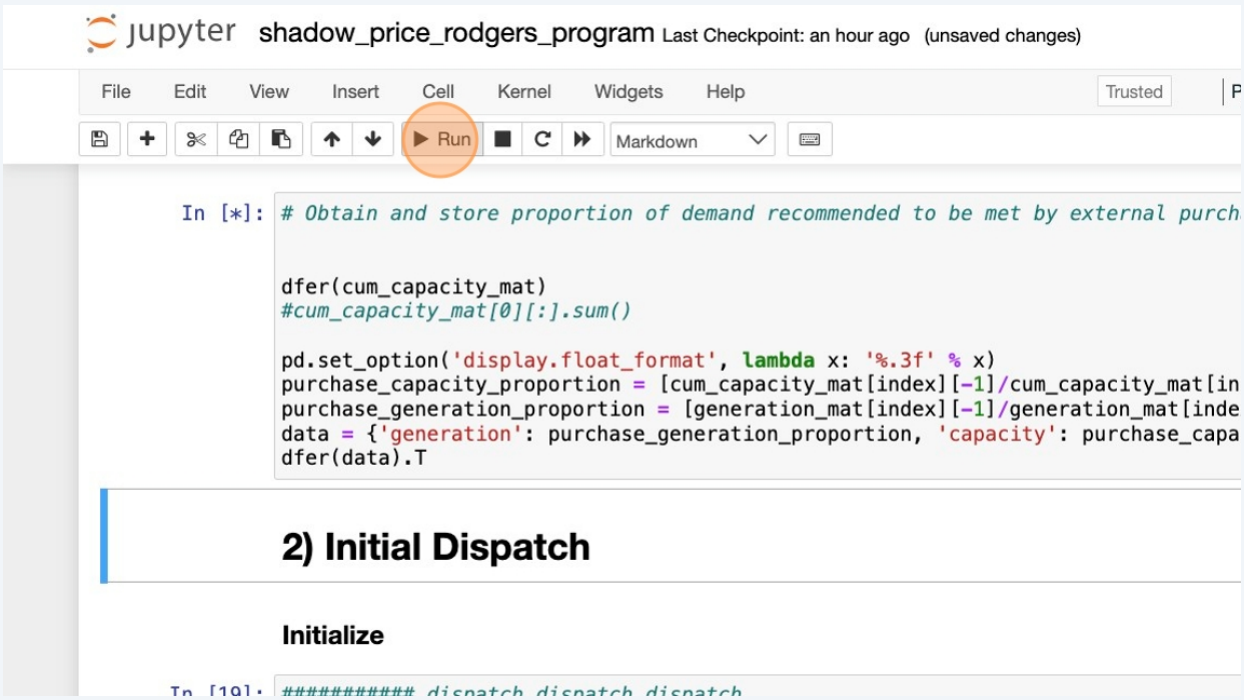
21 Click "Run"



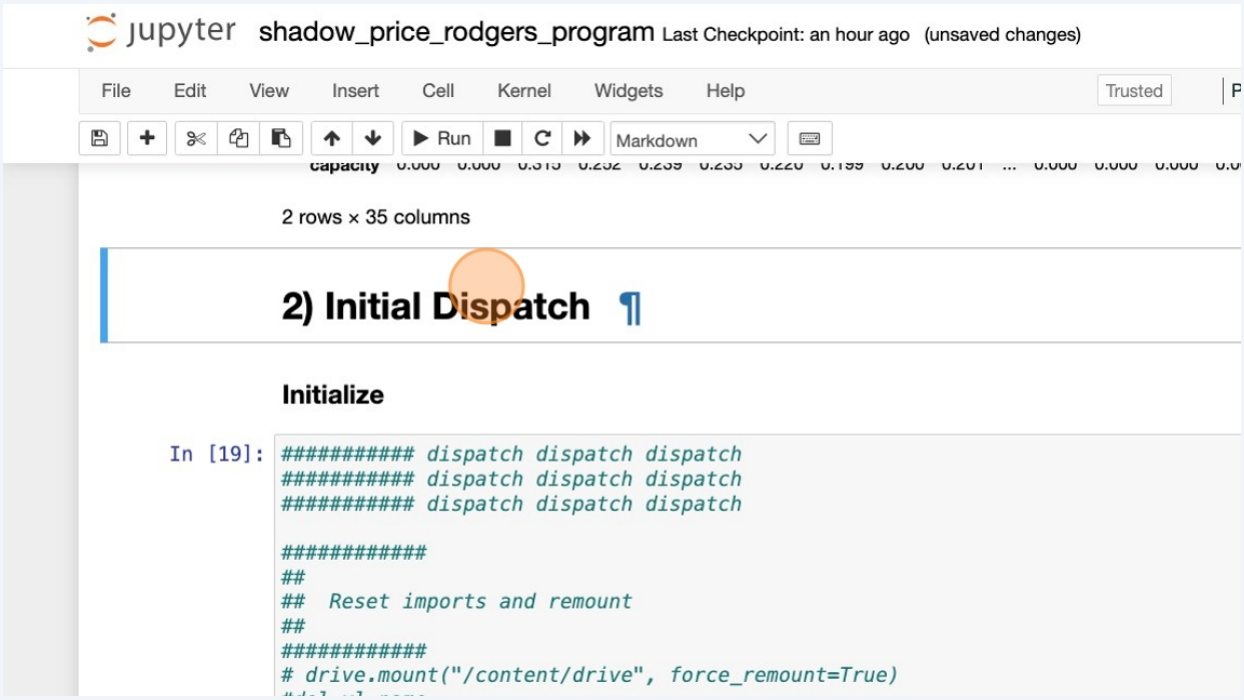
22 Click "Run"



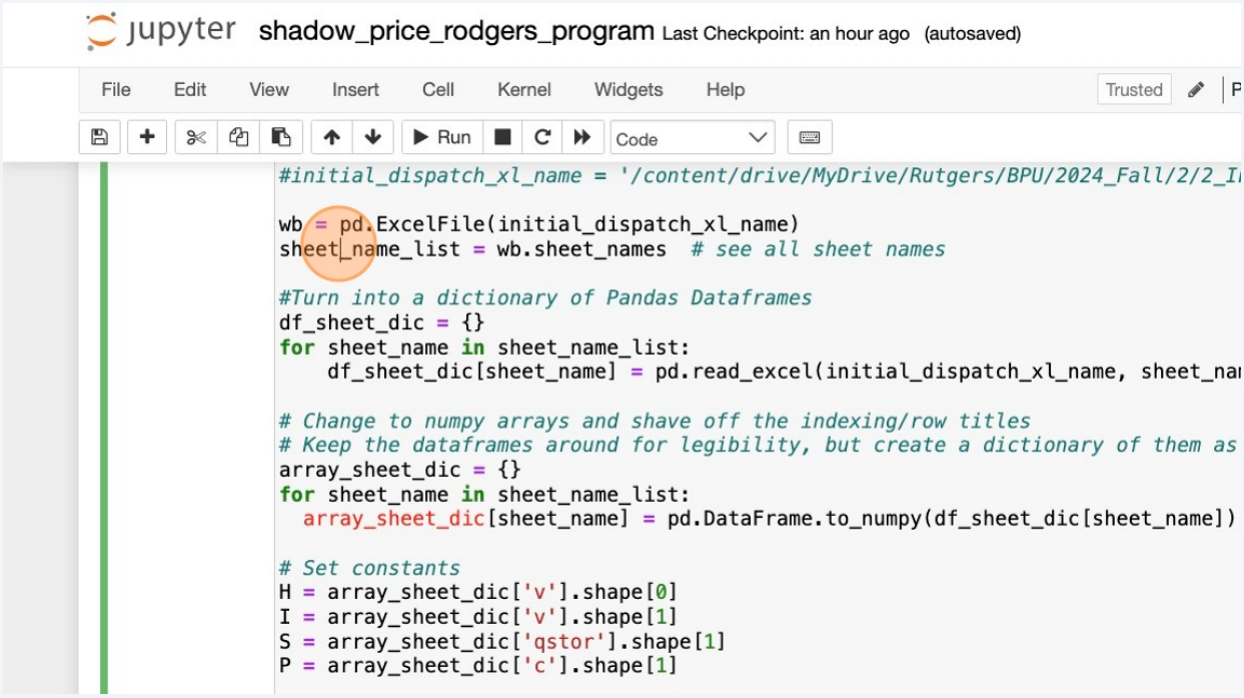
23 Click "Run"



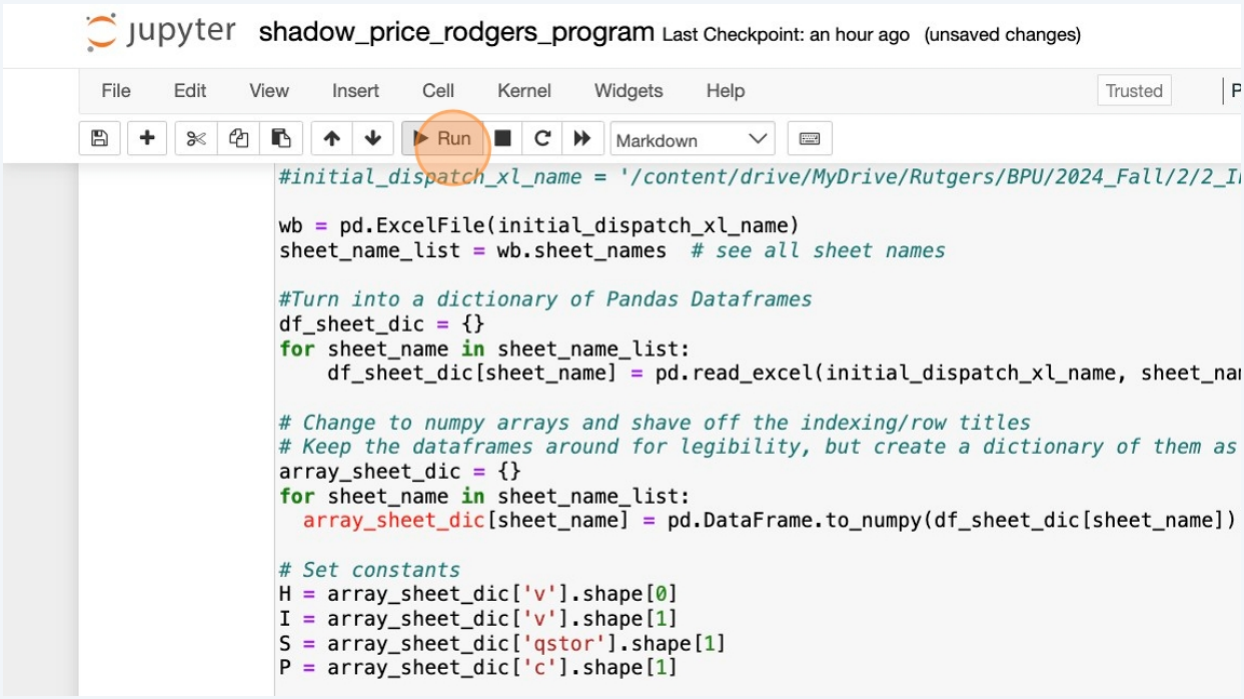
24 Click "2) Initial Dispatch"



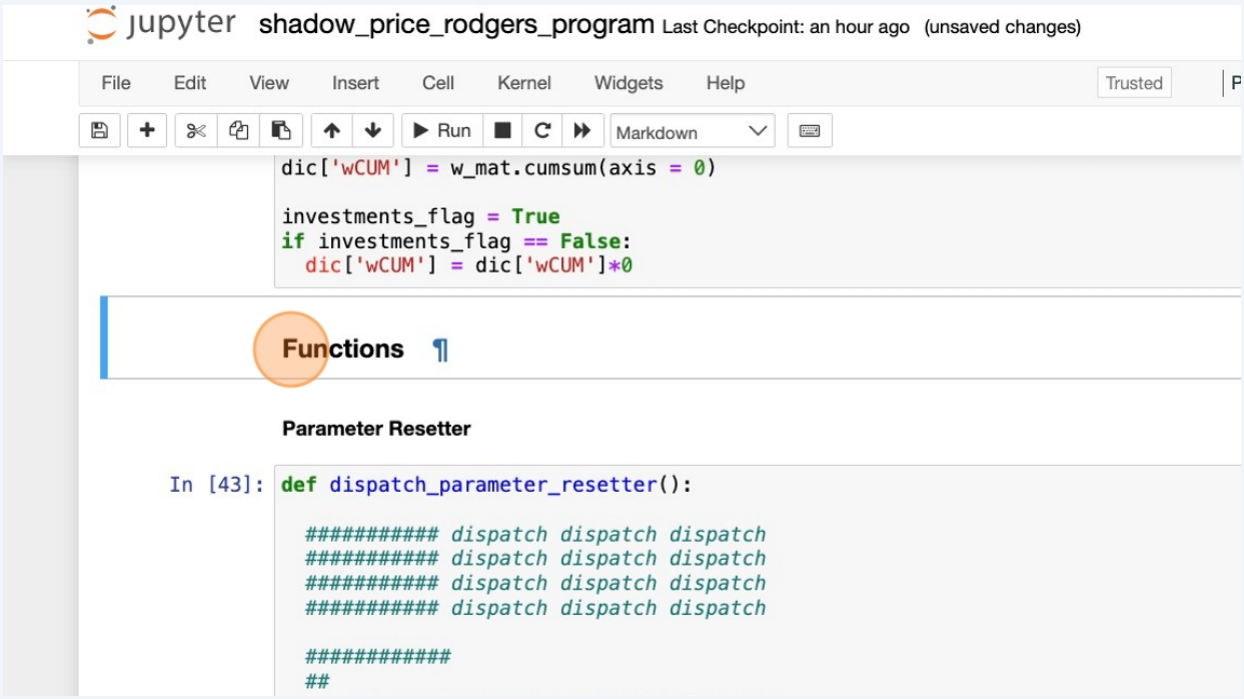
25 Click "sheet_name_list"



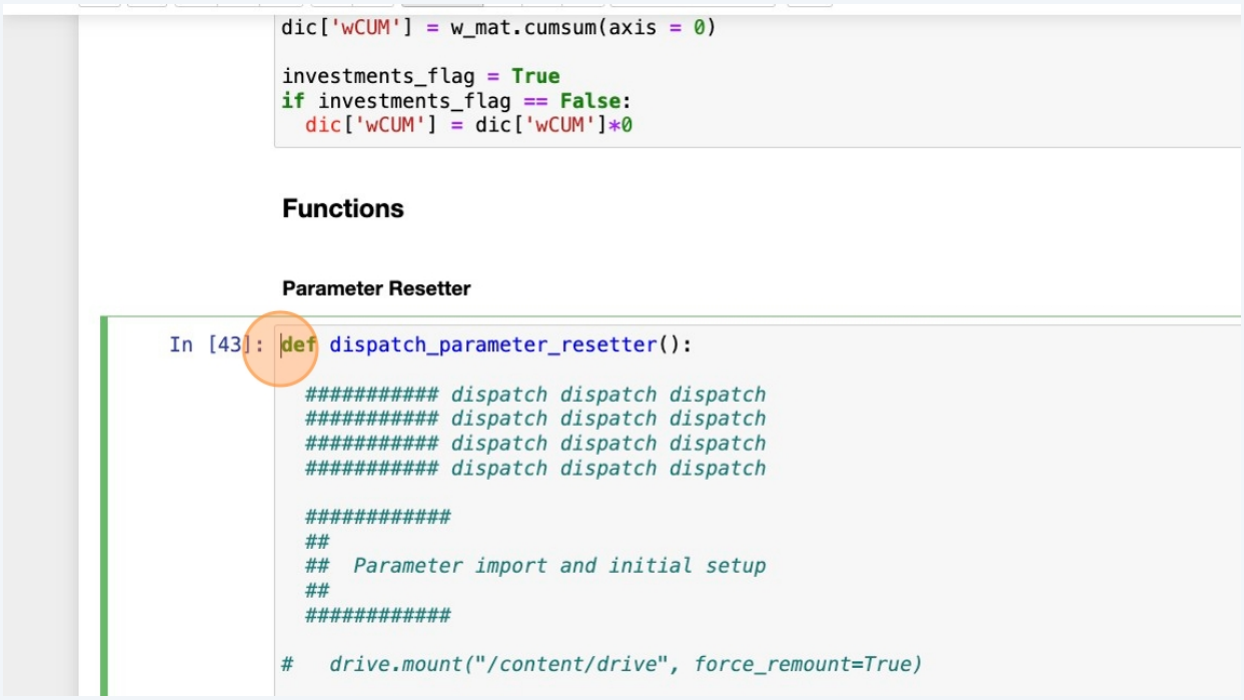
26 Click "Run"



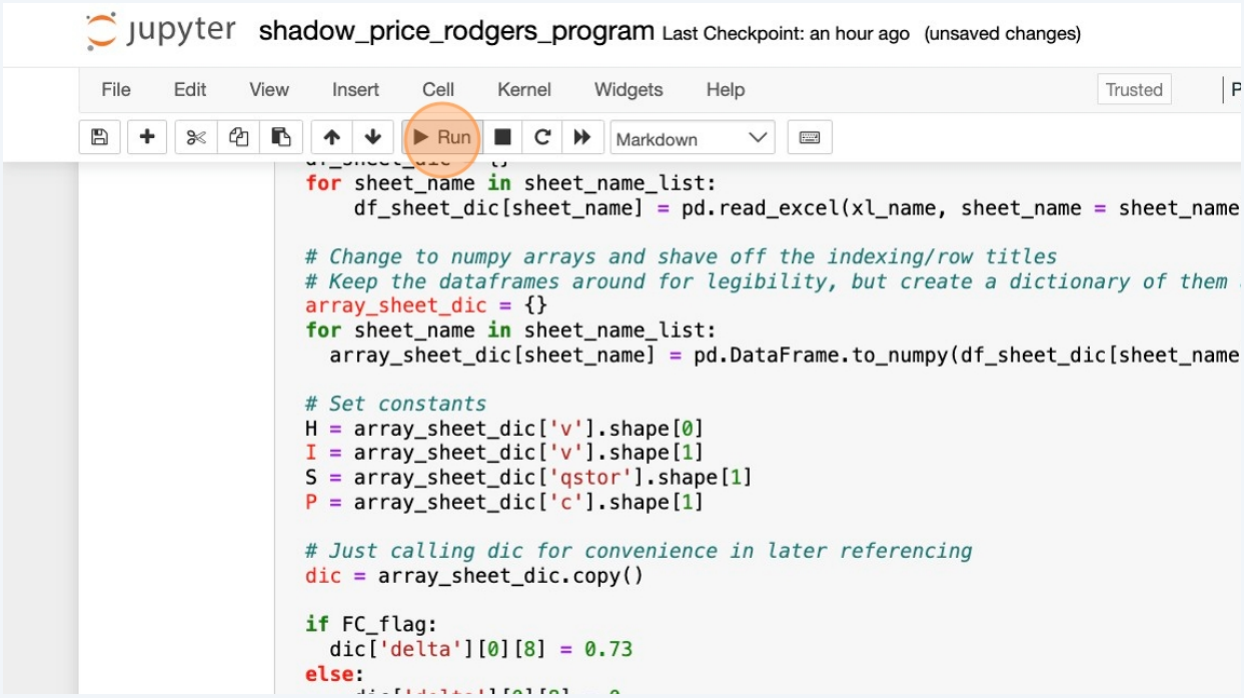
27 Click "Functions¶"



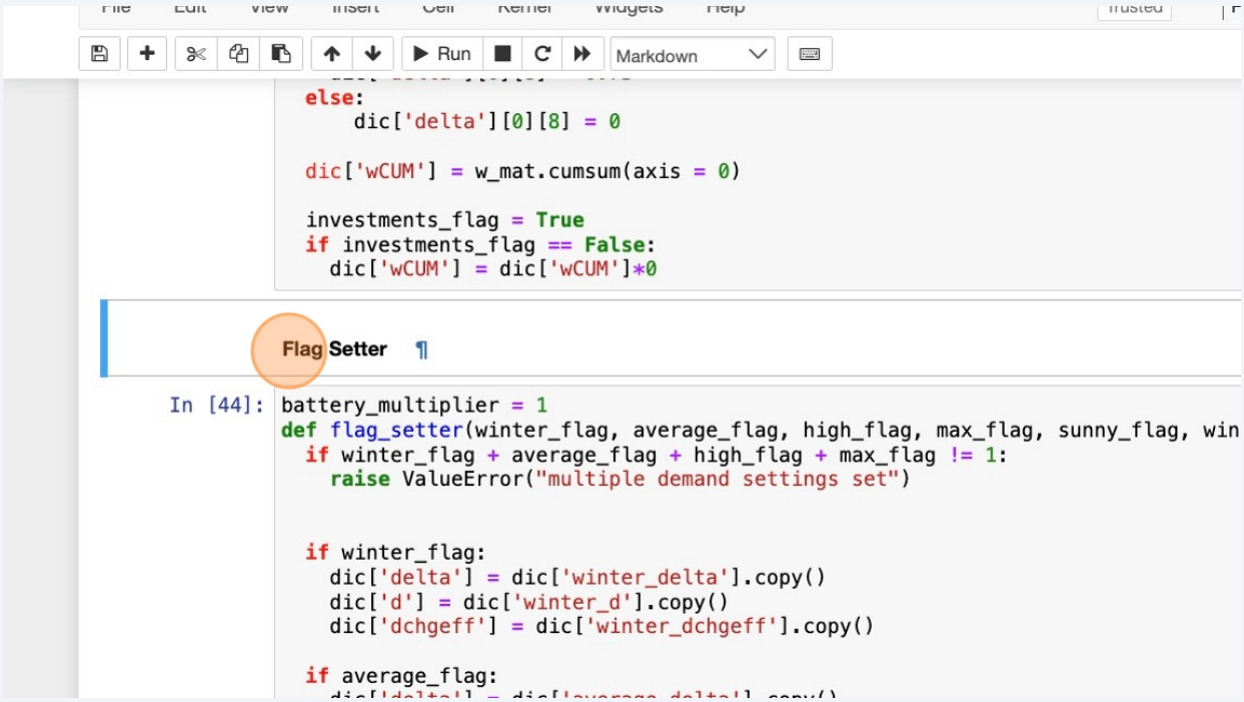
28 Click "def"



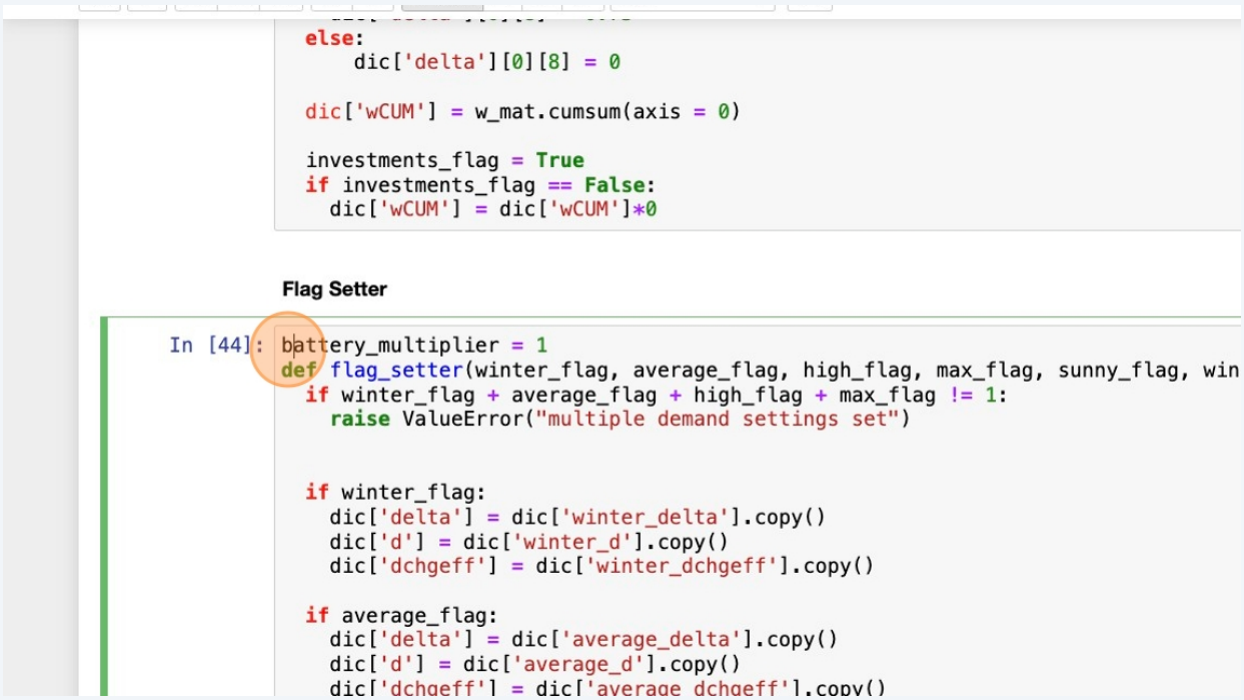
29 Click "Run"



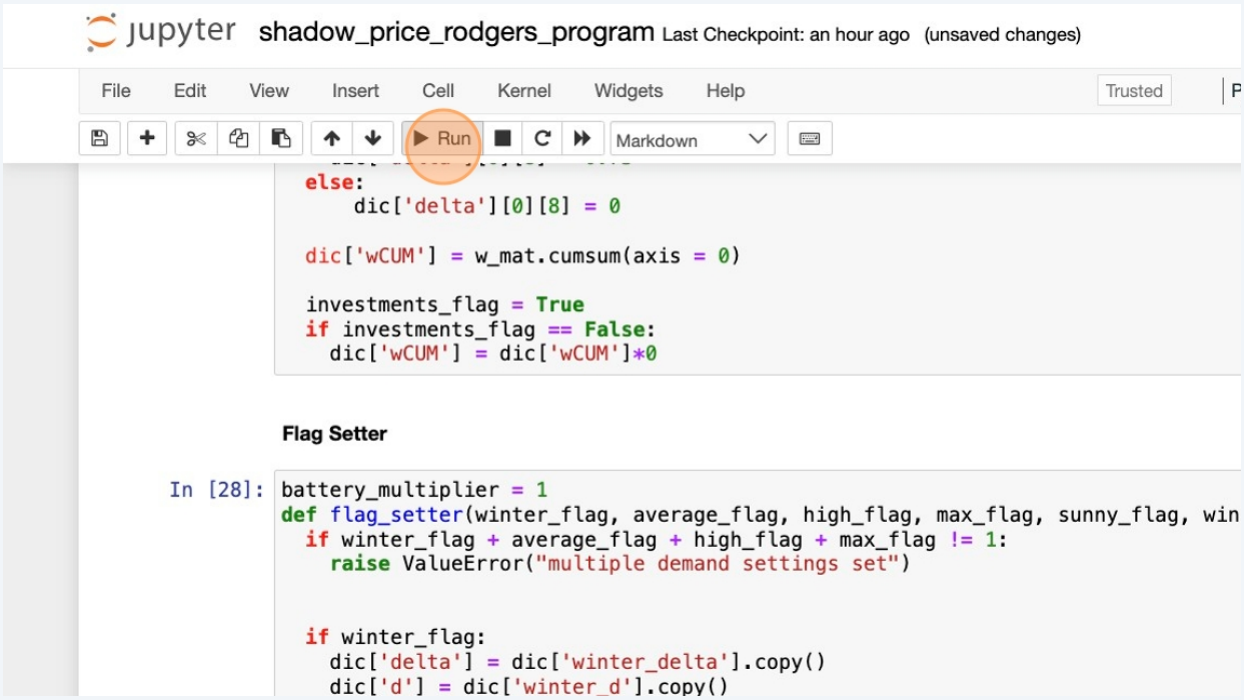
30 Click "Flag Setter¶"



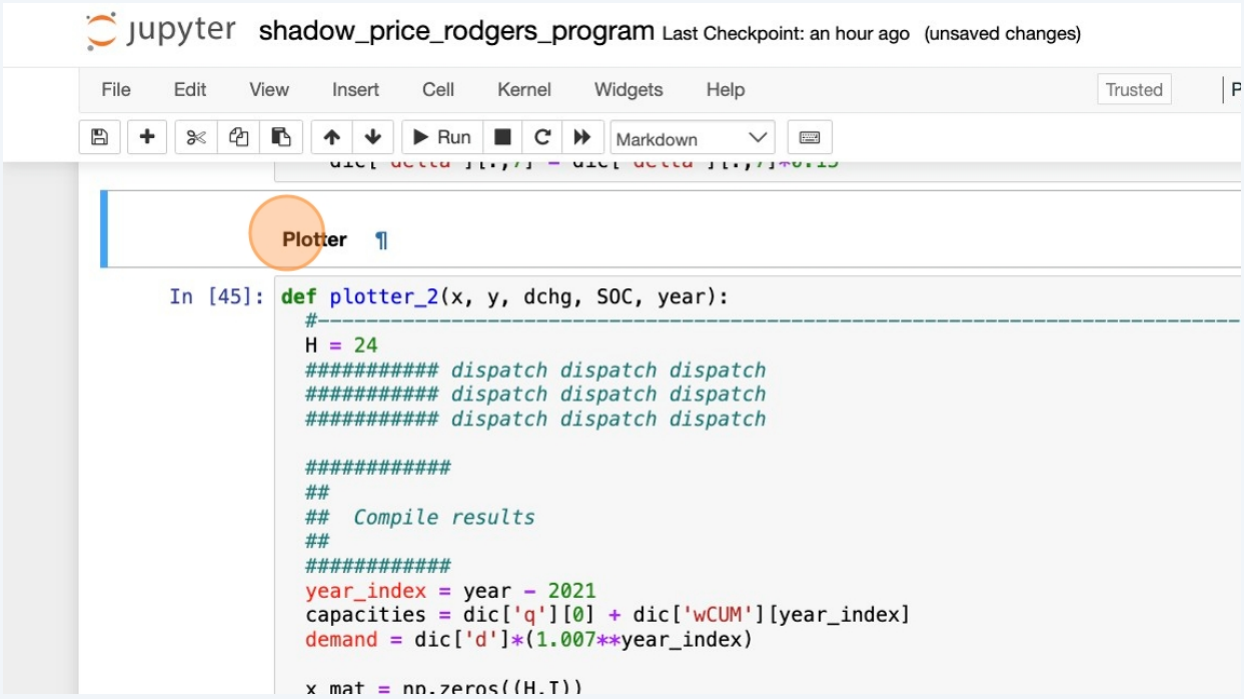
31 Click "battery_multiplier"



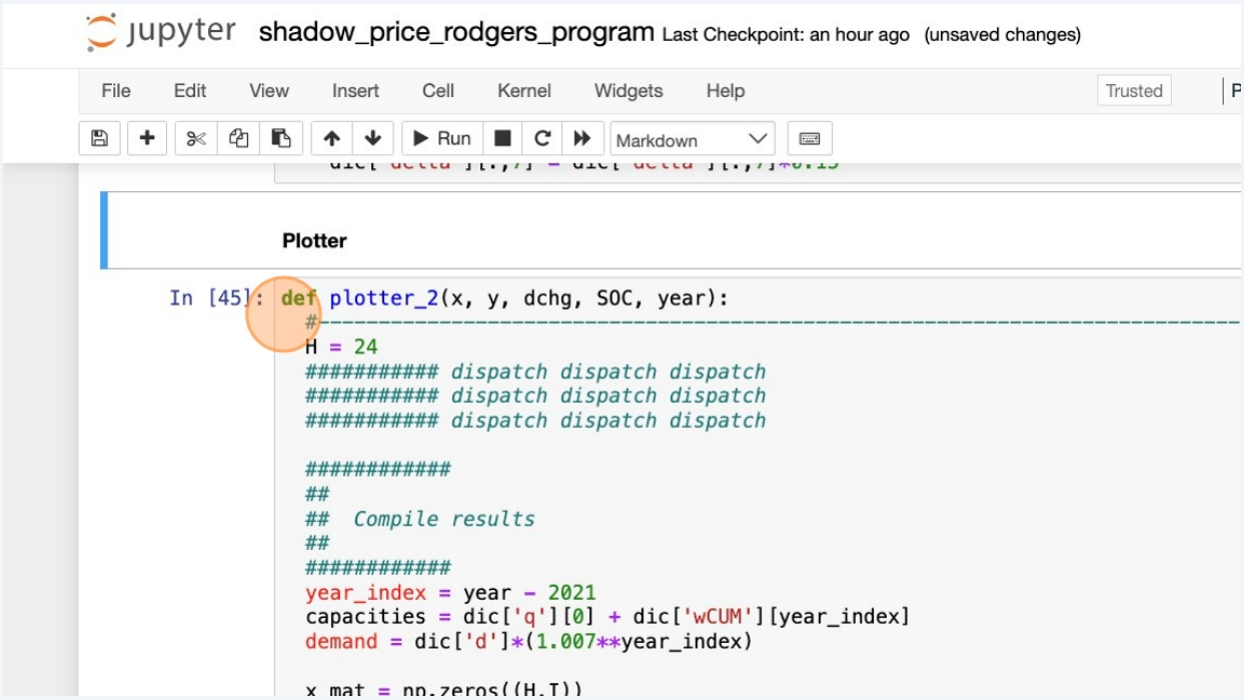
32 Click "Run"



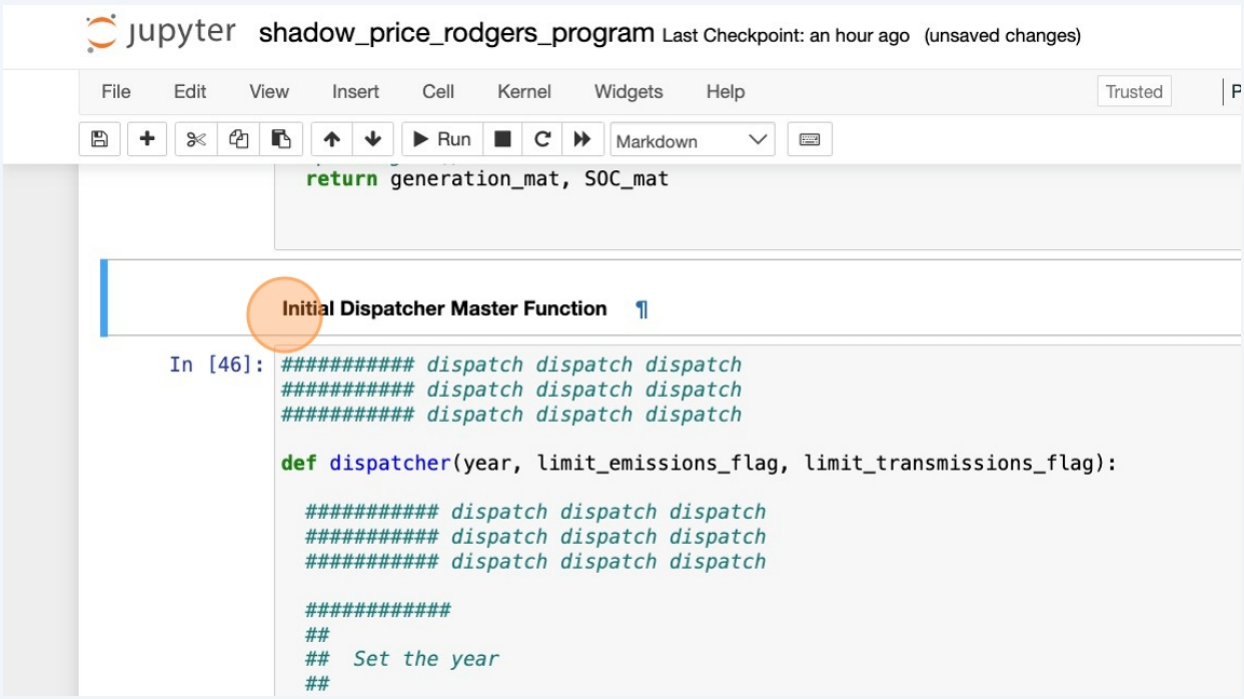
33 Click "Plotter¶"



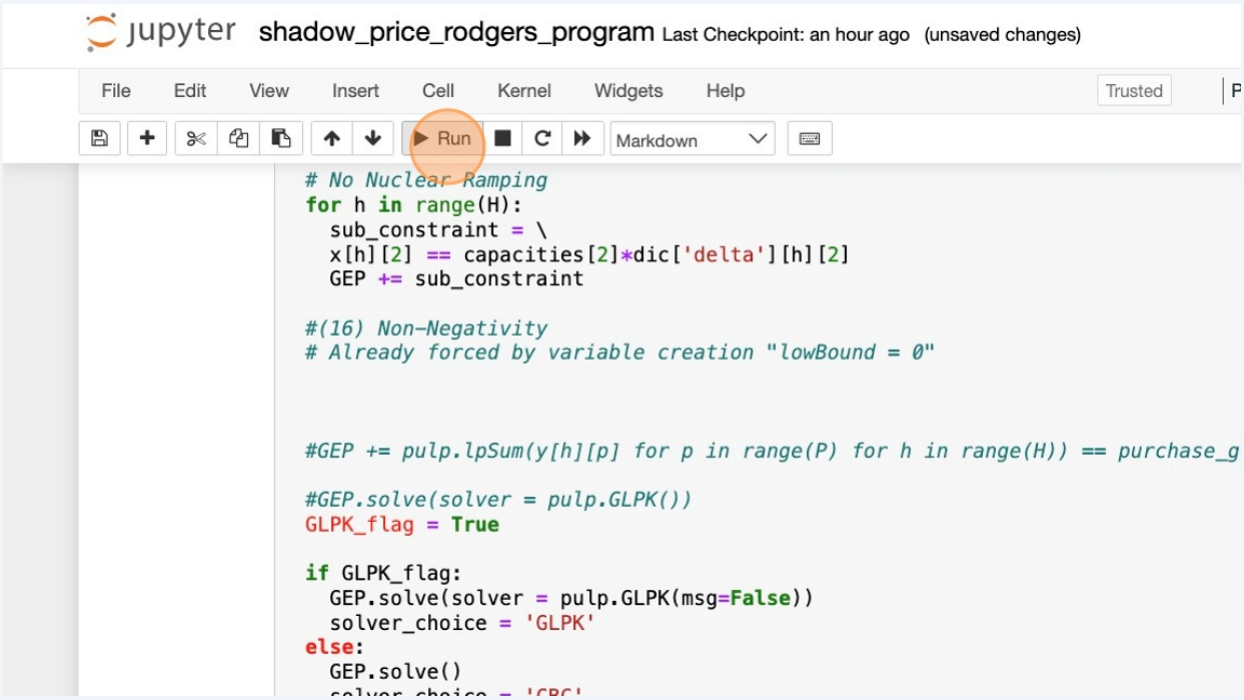
34 Click "#----------..."



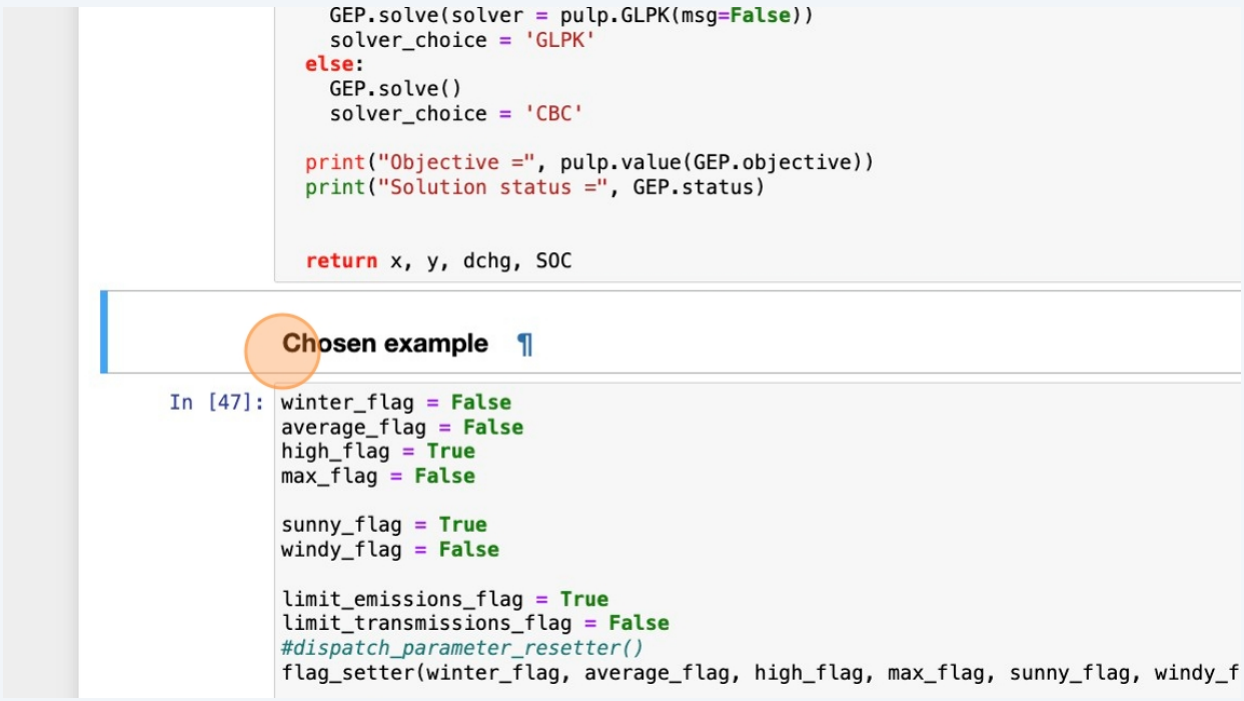
35 Click "Initial Dispatcher Master Function¶"



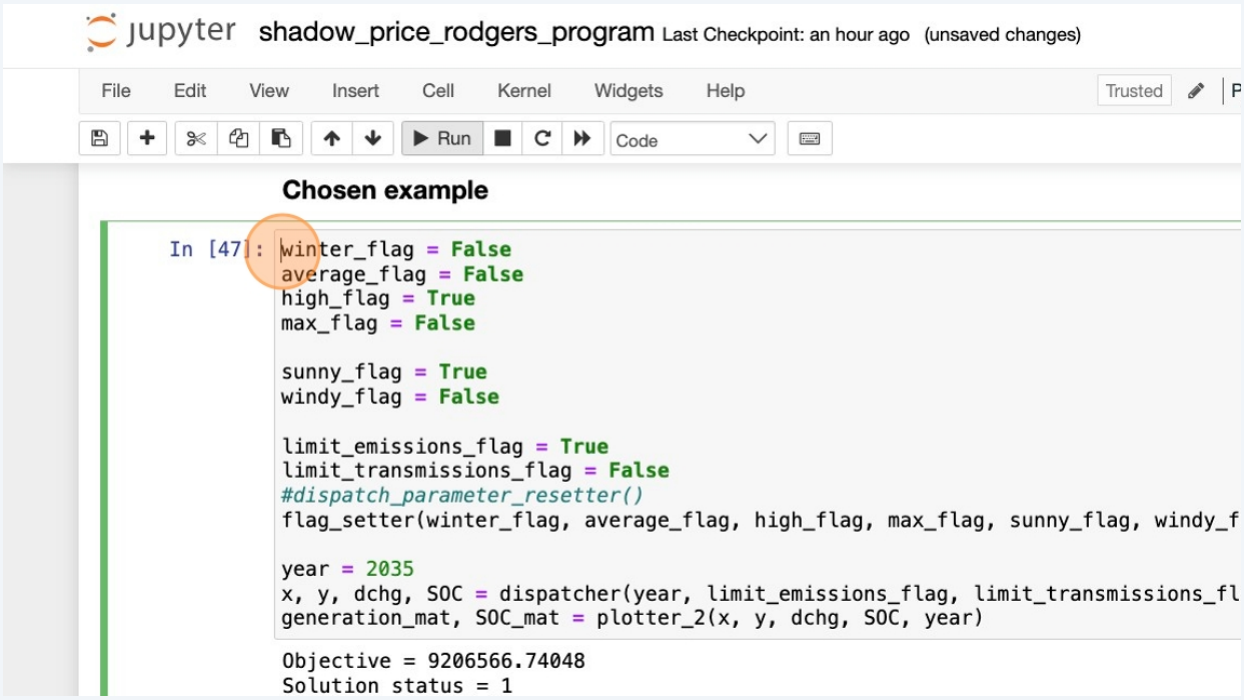
36 Click "Run"



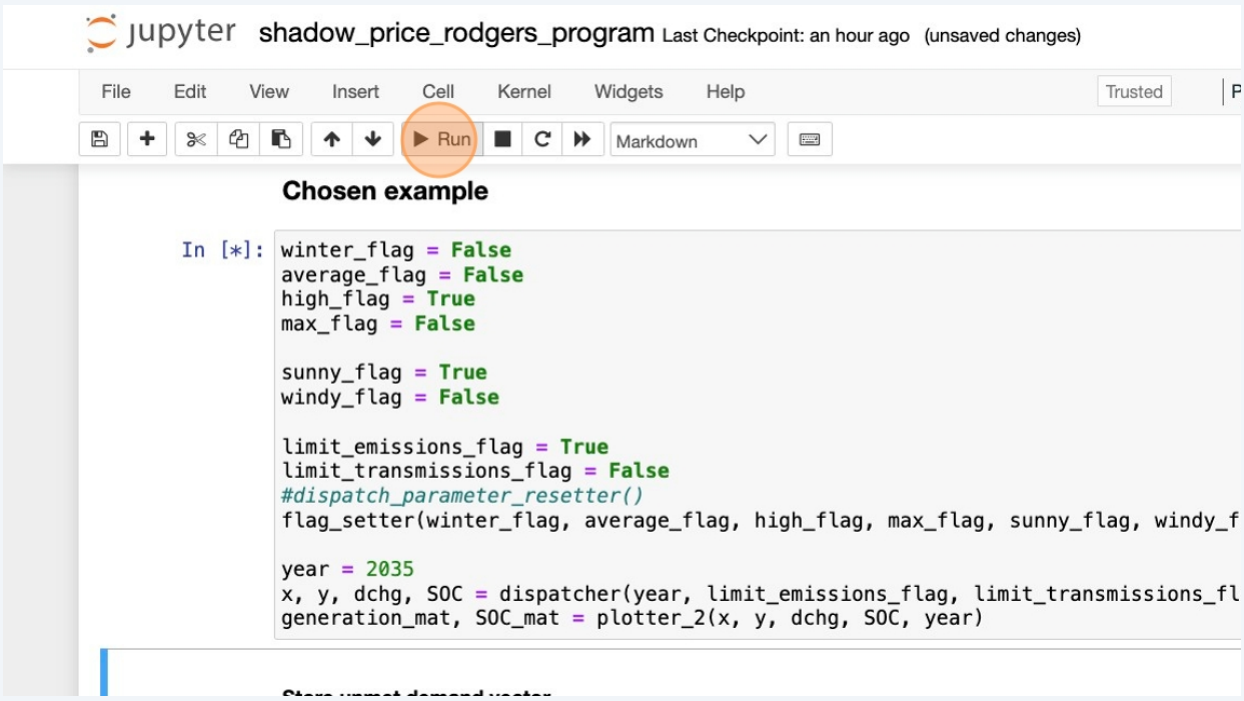
37 Click "Chosen example¶"



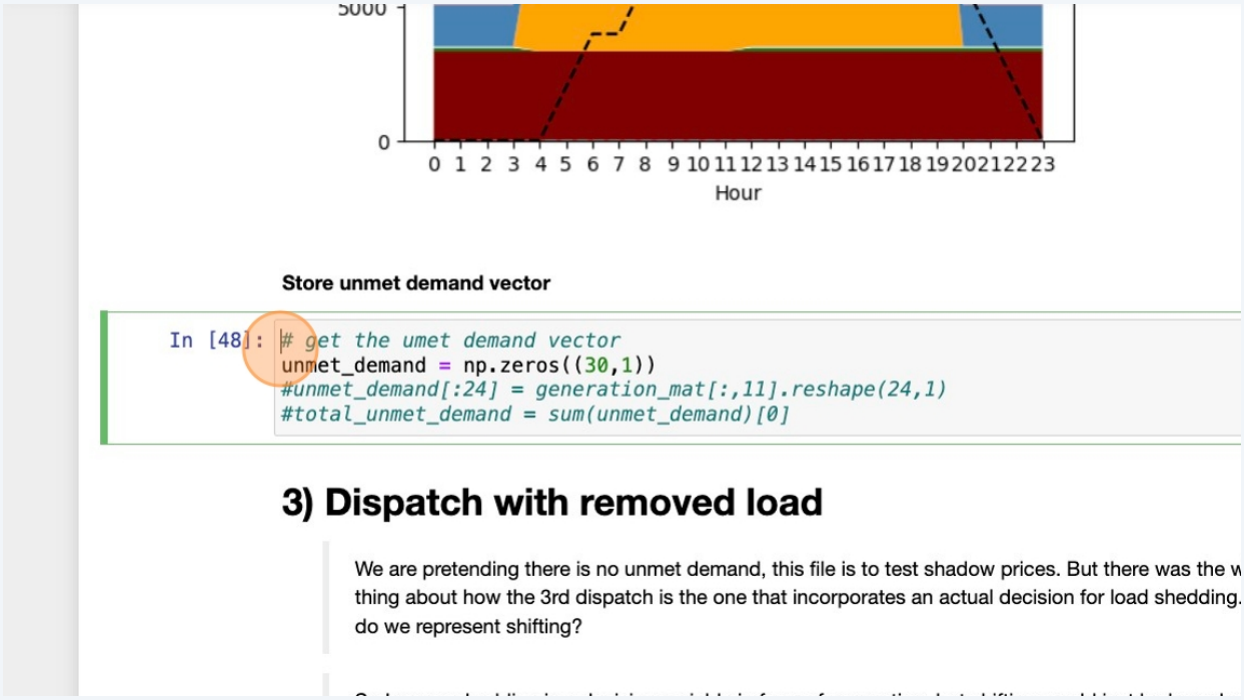
38 Click "winter_flag"



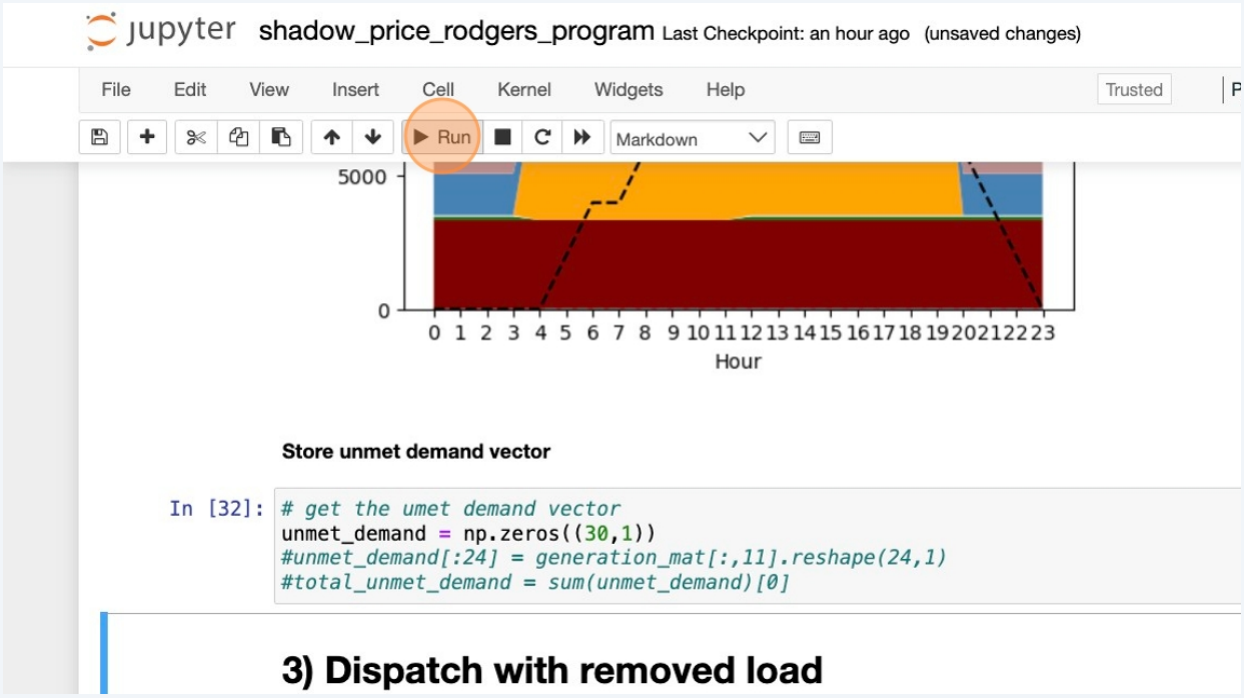
39 Click "Run"



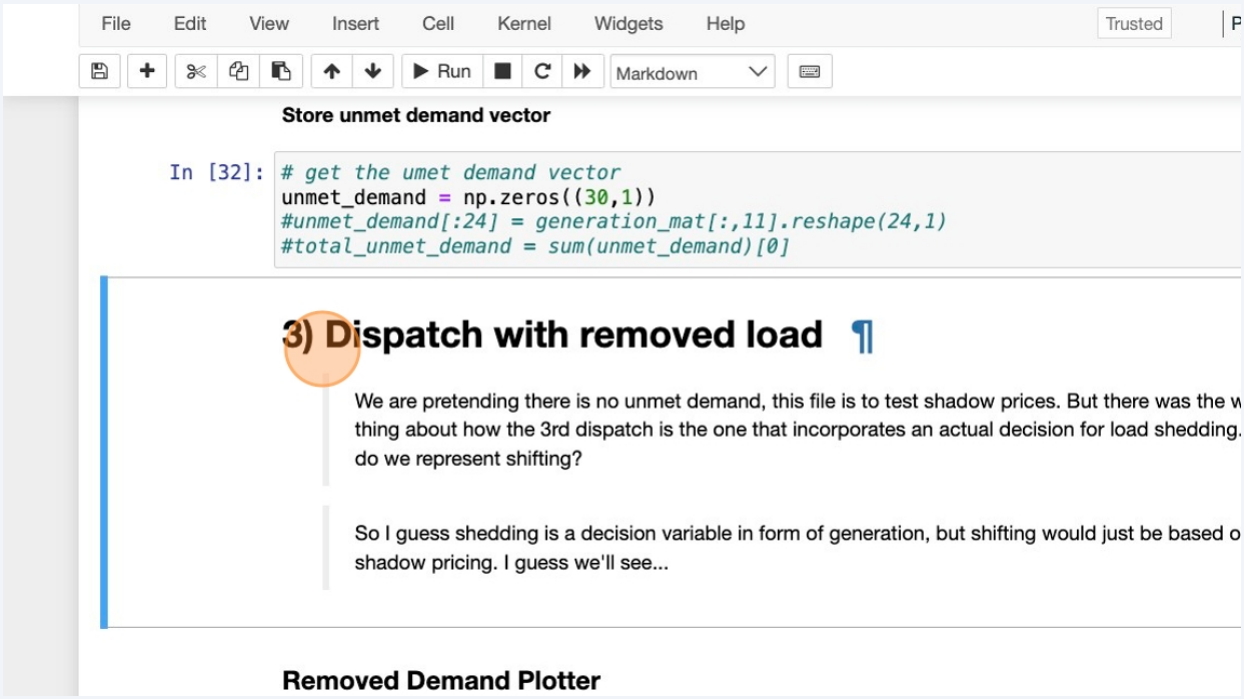
40 Click "# get the umet demand vector"



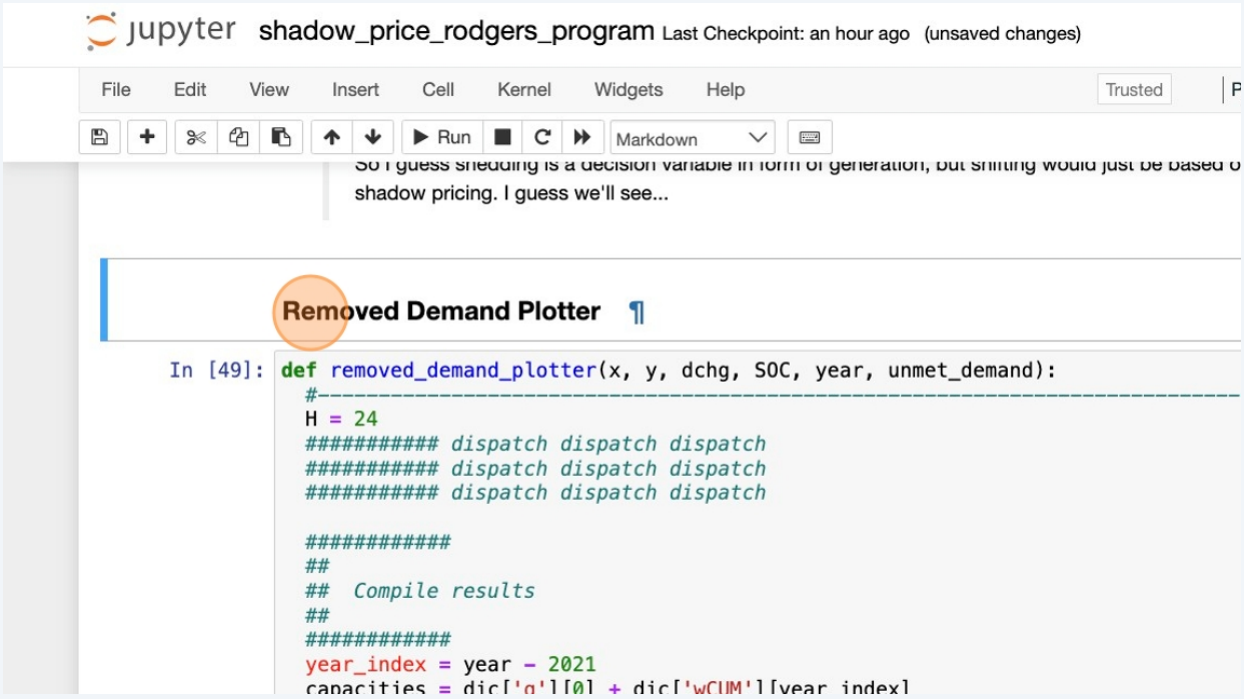
41 Click "Run"



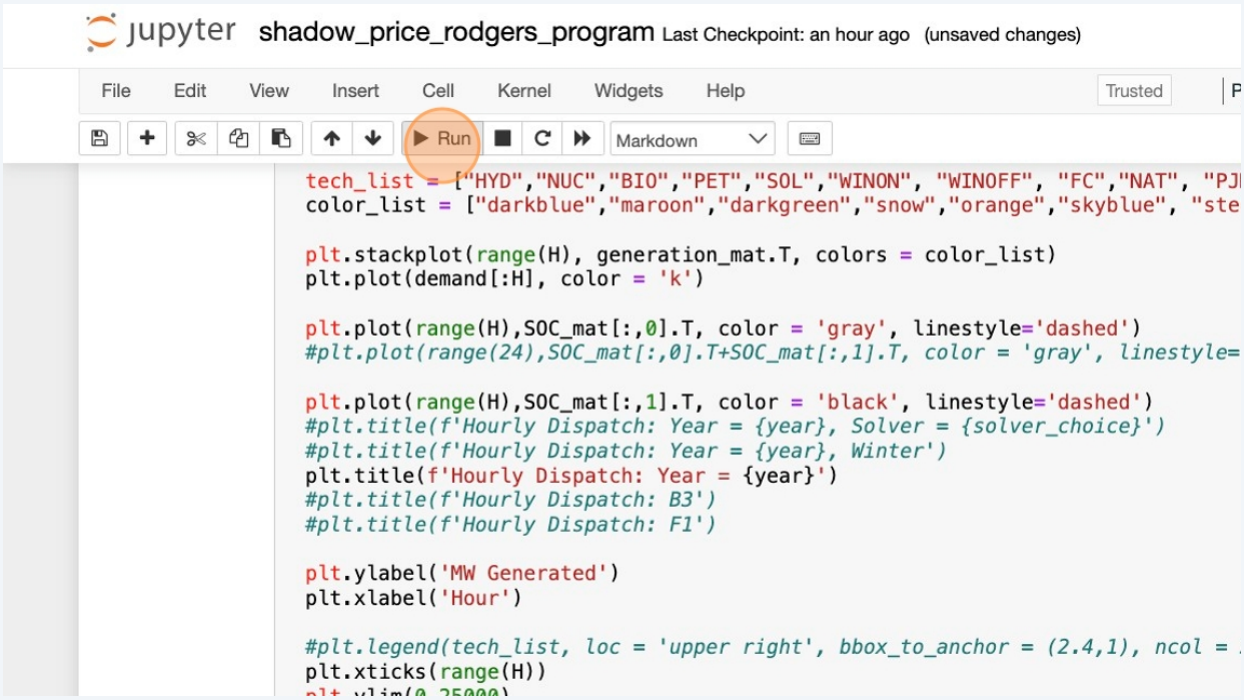
42 Click "3) Dispatch with removed load"



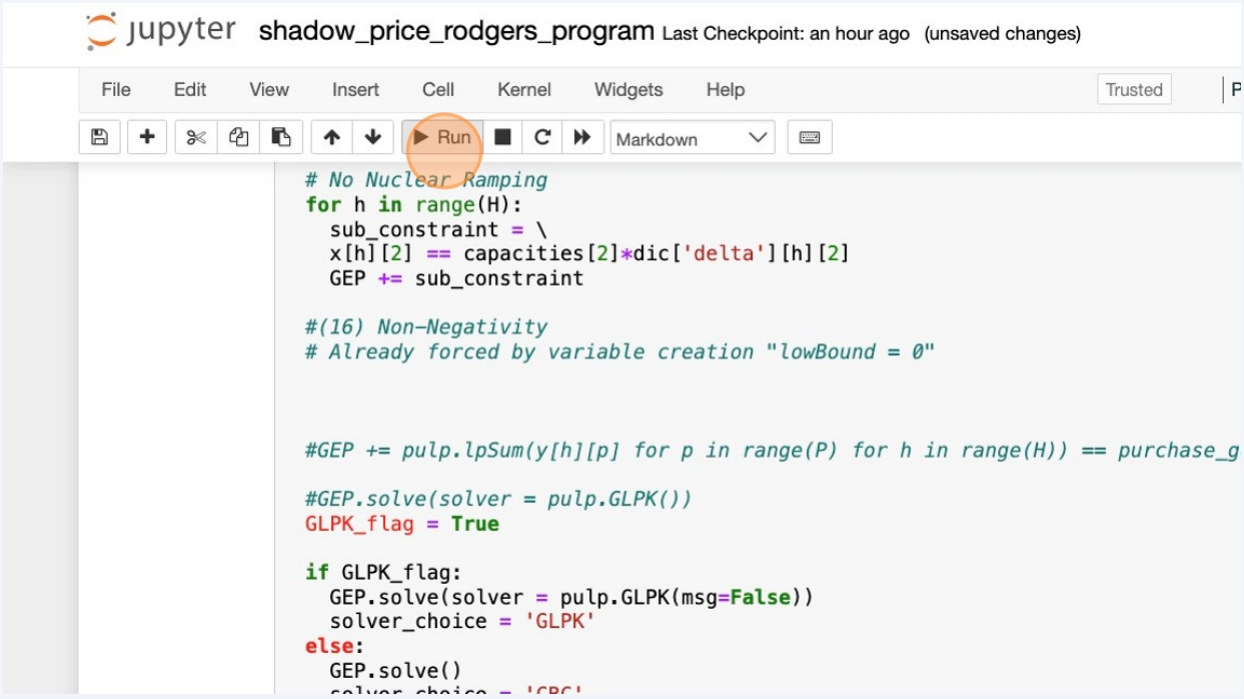
43 Click "Removed Demand Plotter¶"



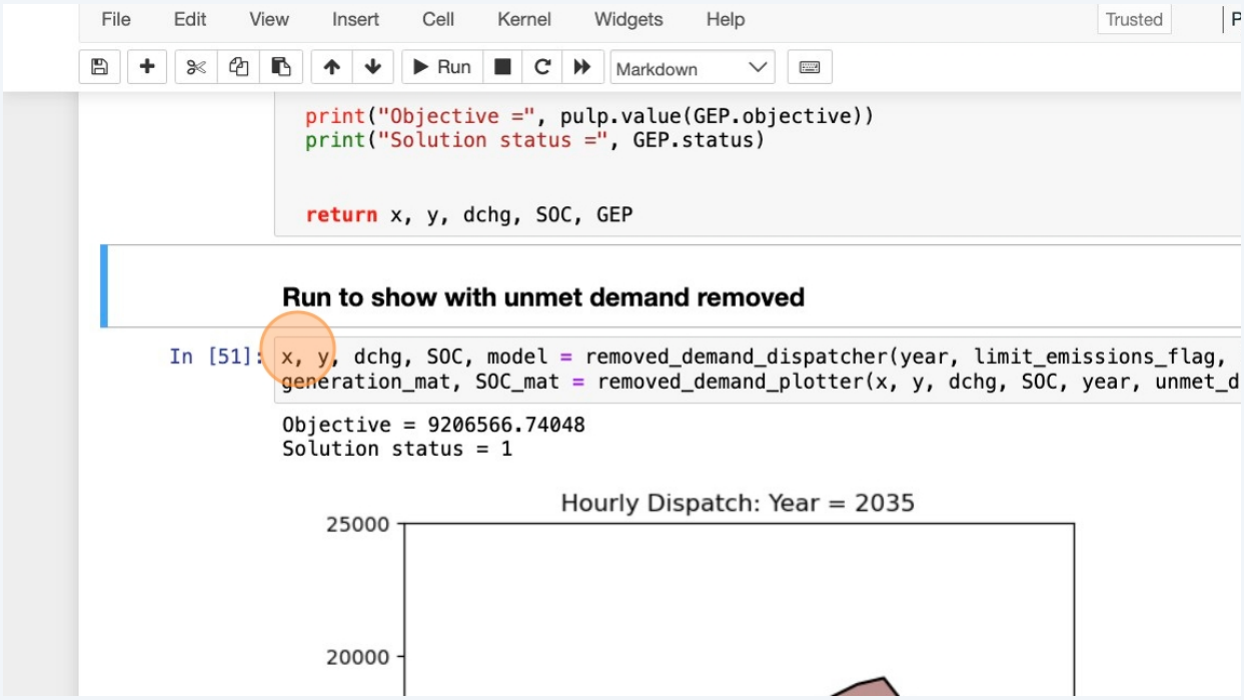
44 Click "Run"



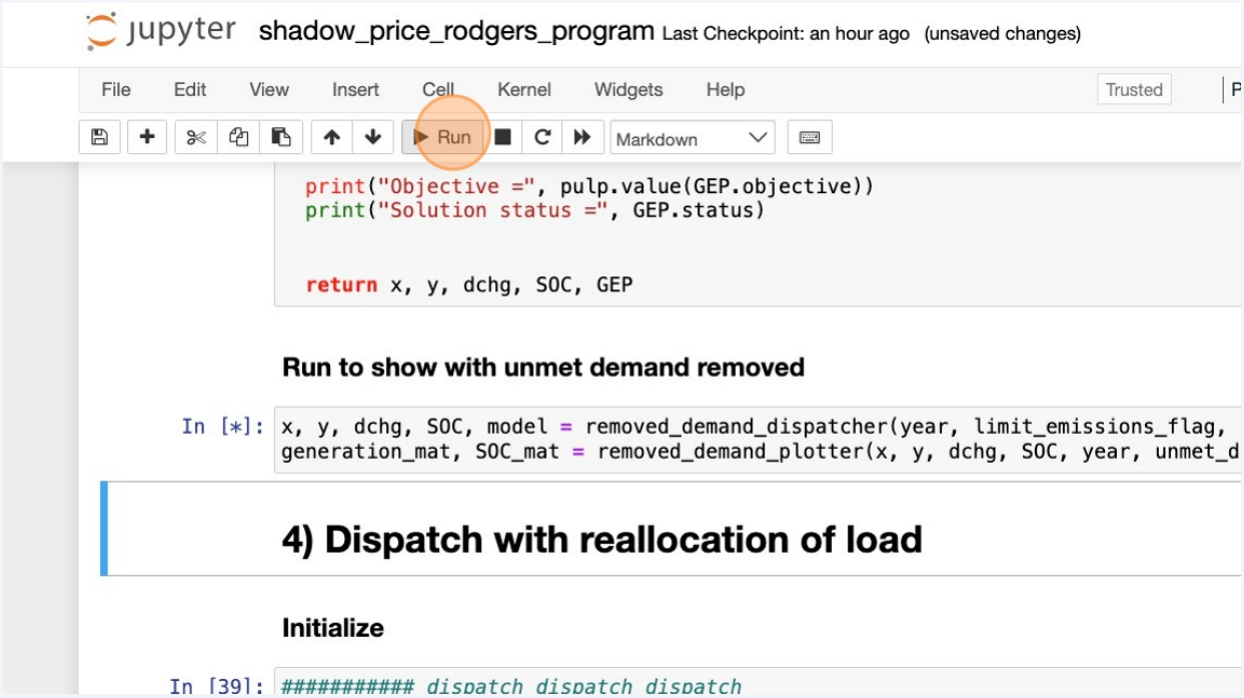
45 Click "Run"



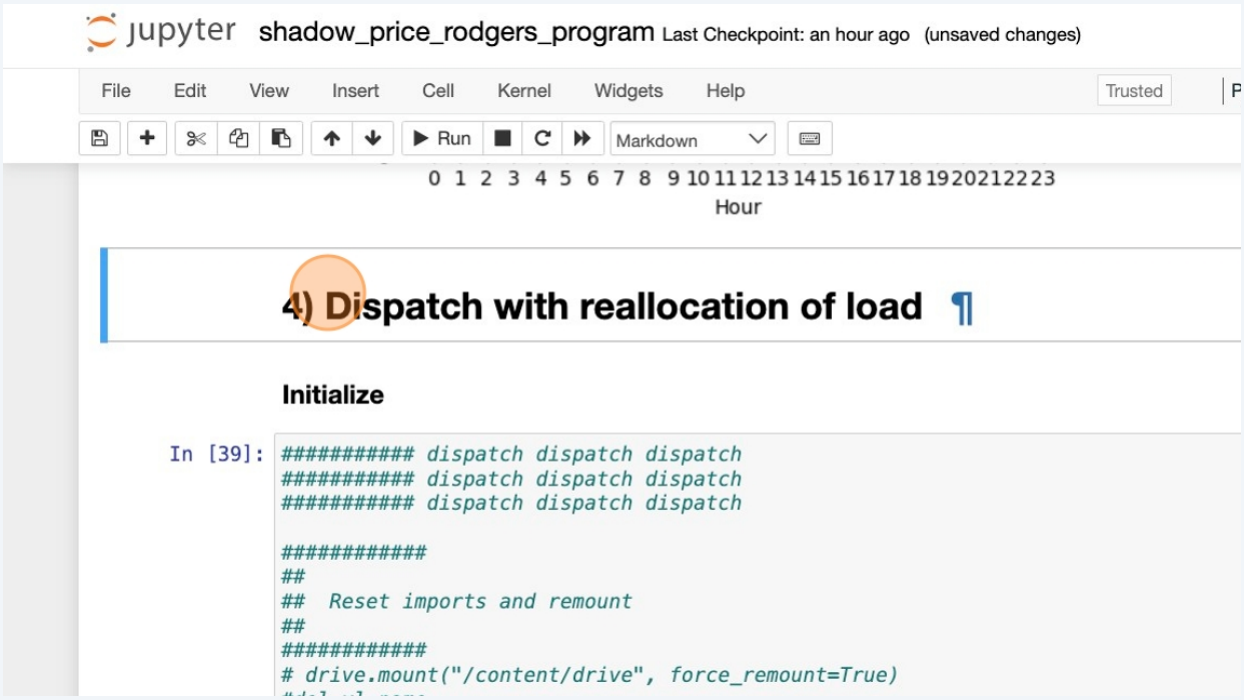
46 Click "x, y, dchg, SOC, model = removed_demand_dispatcher(year, limit_emissions_flag, limit_transmissions_flag, unmet_demand)"



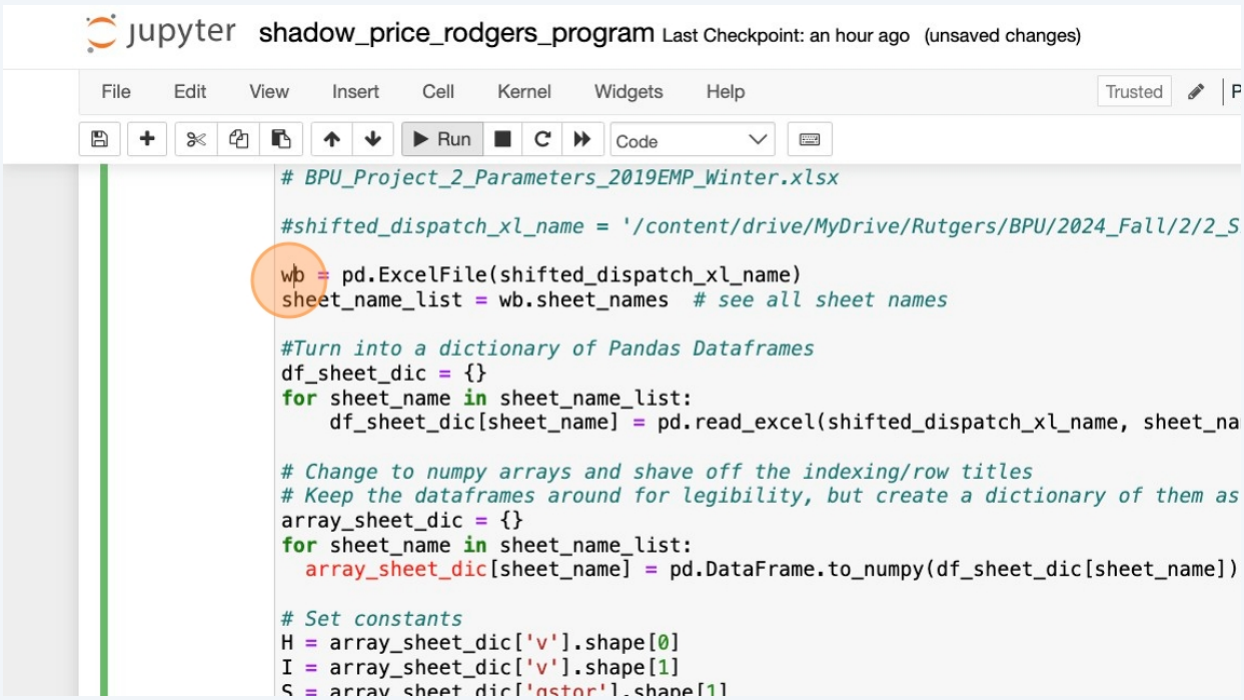
47 Click "Run"



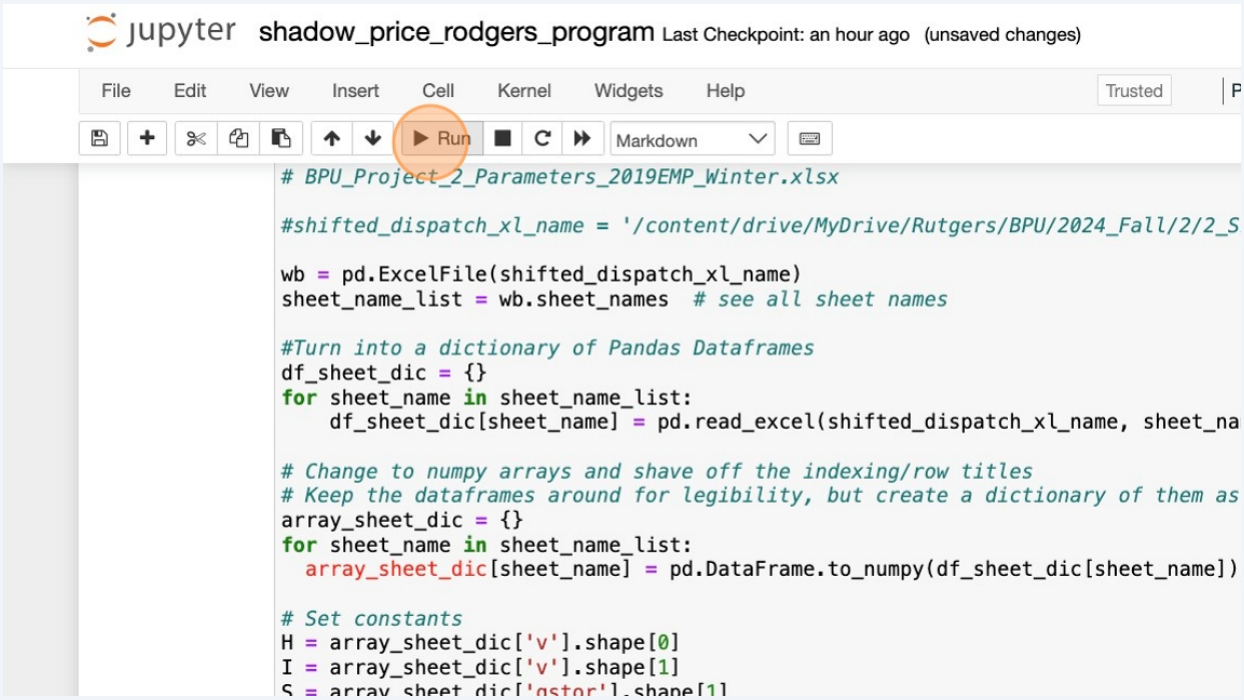
48 Click "4) Dispatch with reallocation of load"



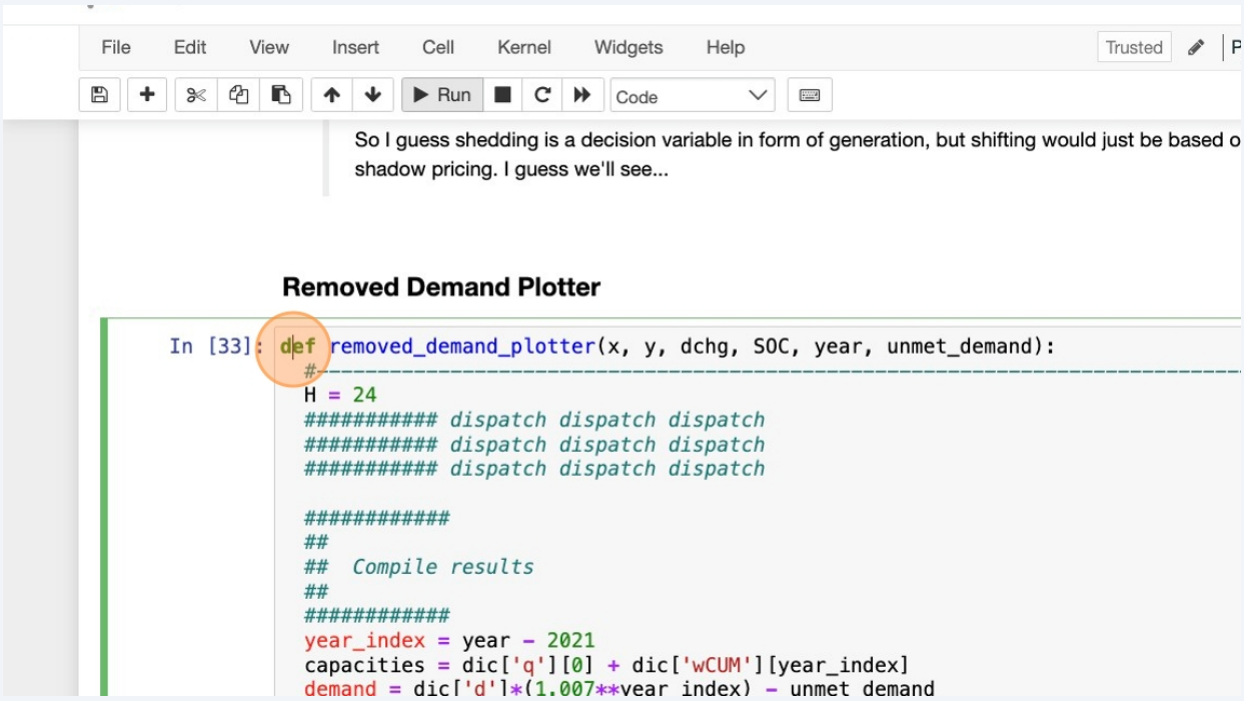
49 Click "wb"



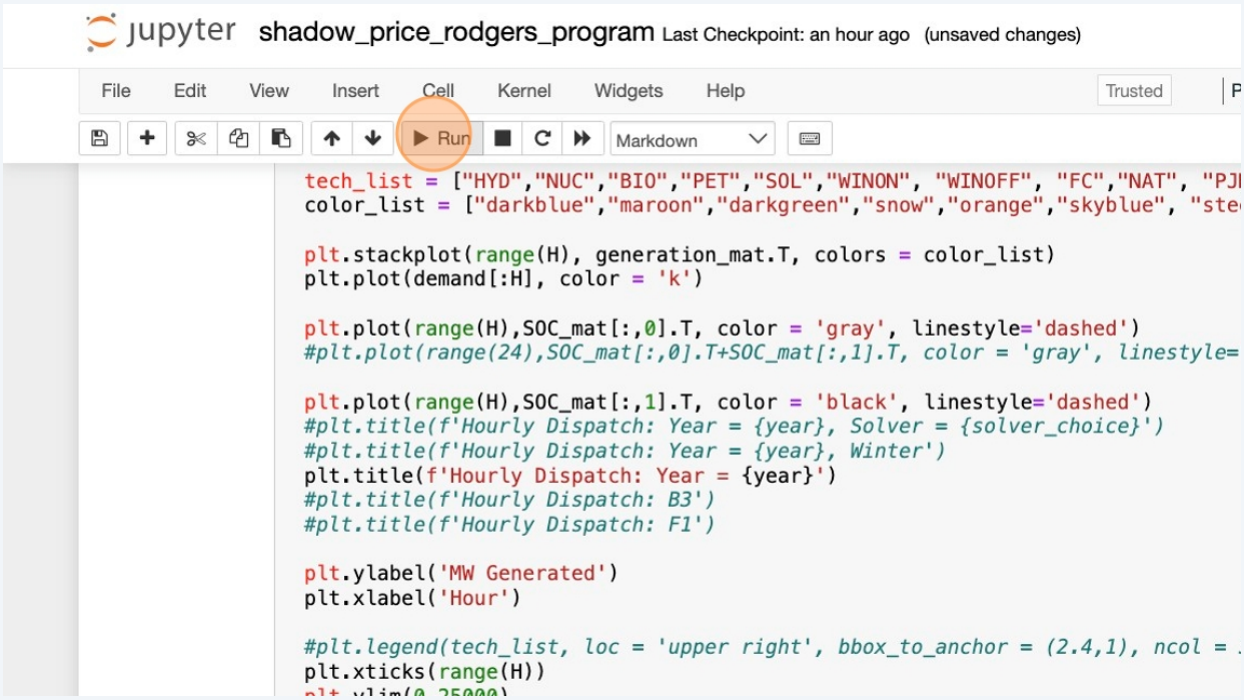
50 Click "Run"



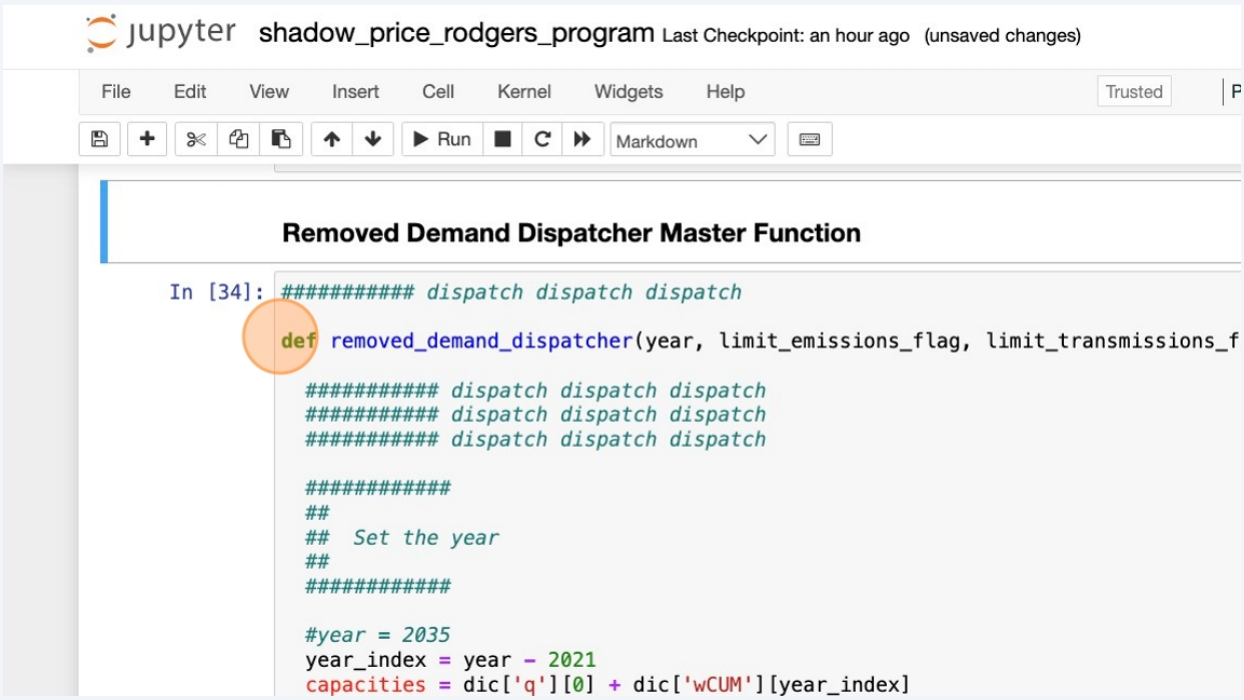
51 Click "def"



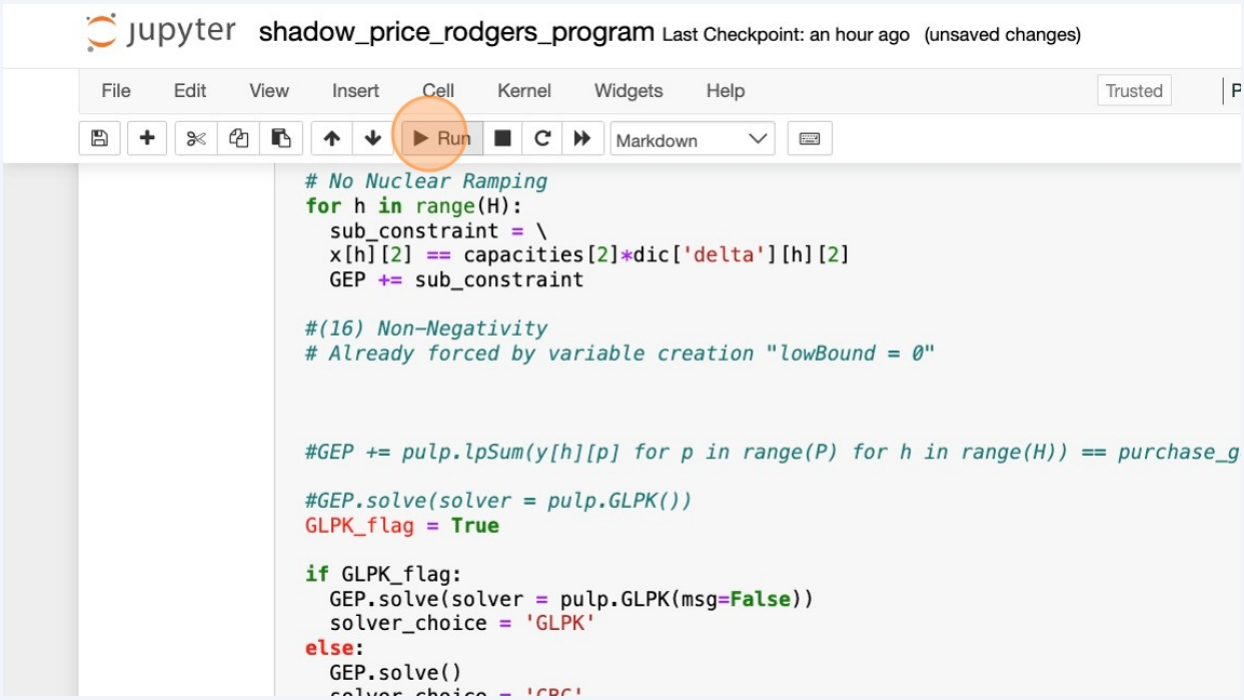
52 Click "Run"



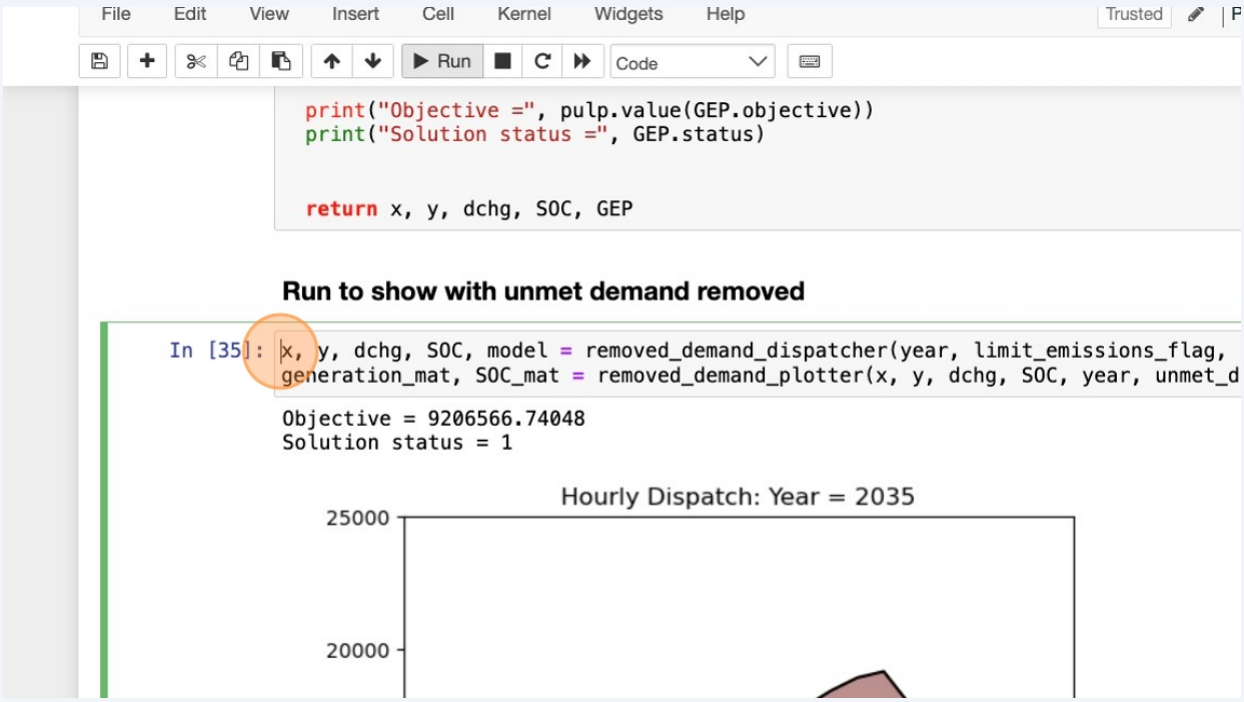
53 Click "def"



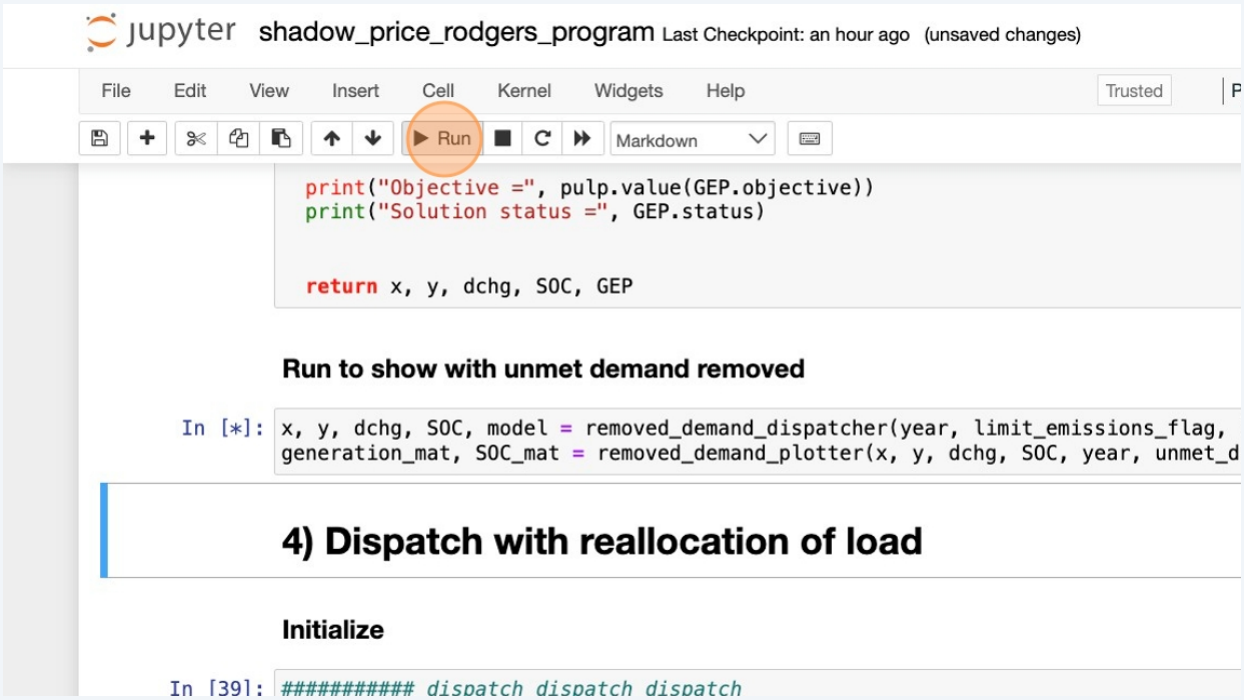
54 Click "Run"



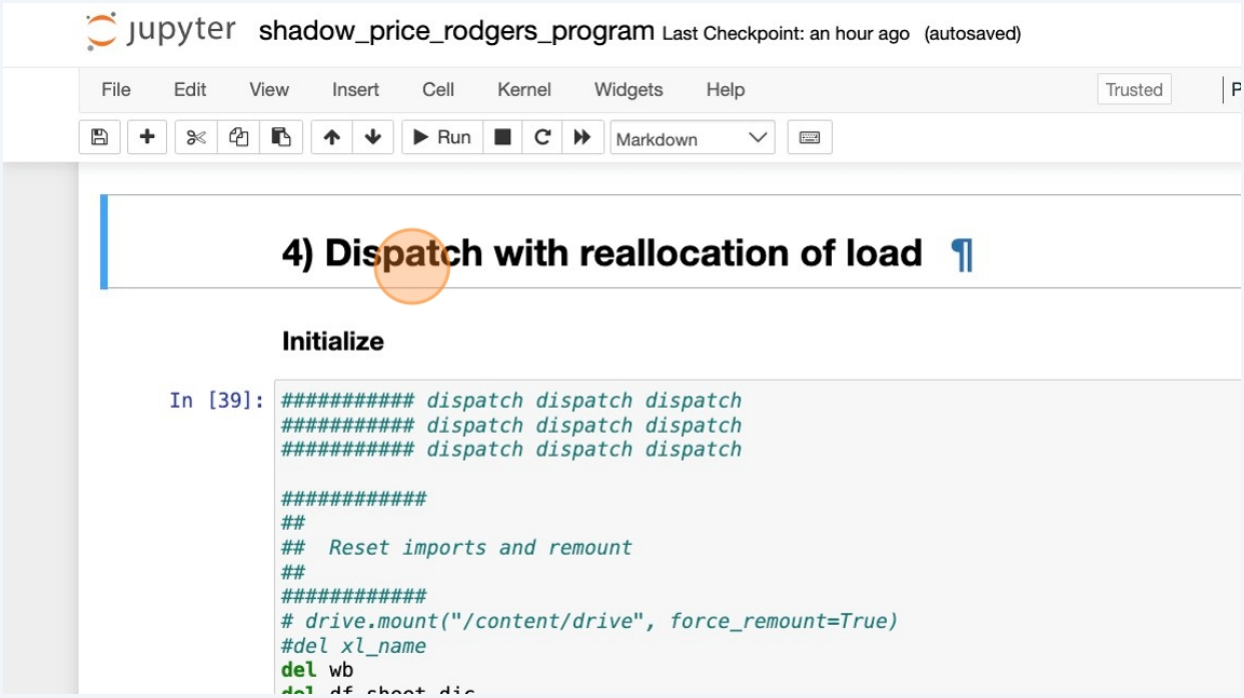
55 Click "x"



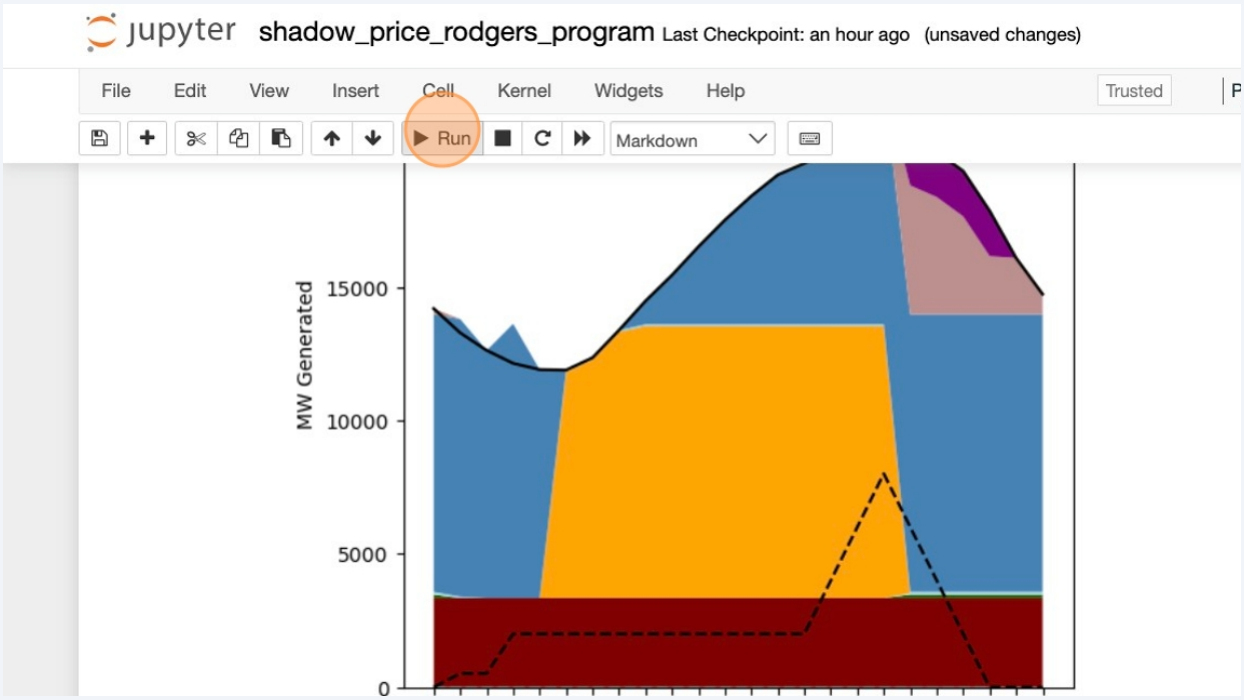
56 Click "Run"



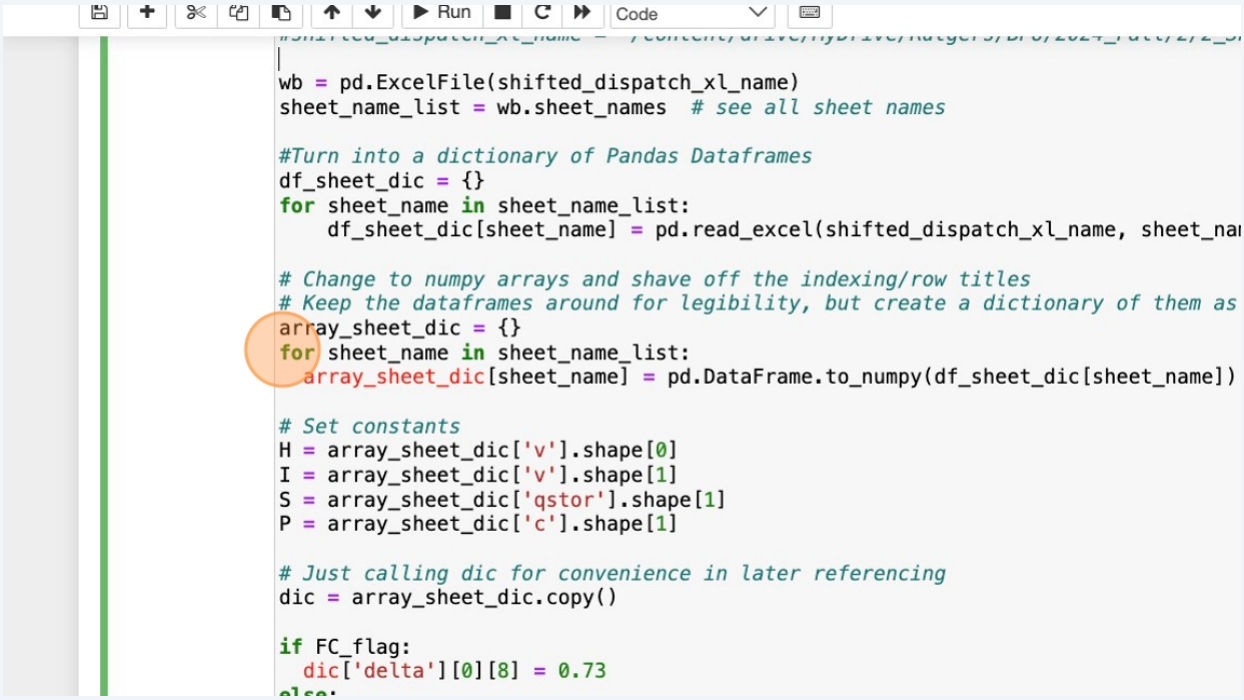
57 Click "4) Dispatch with reallocation of load"



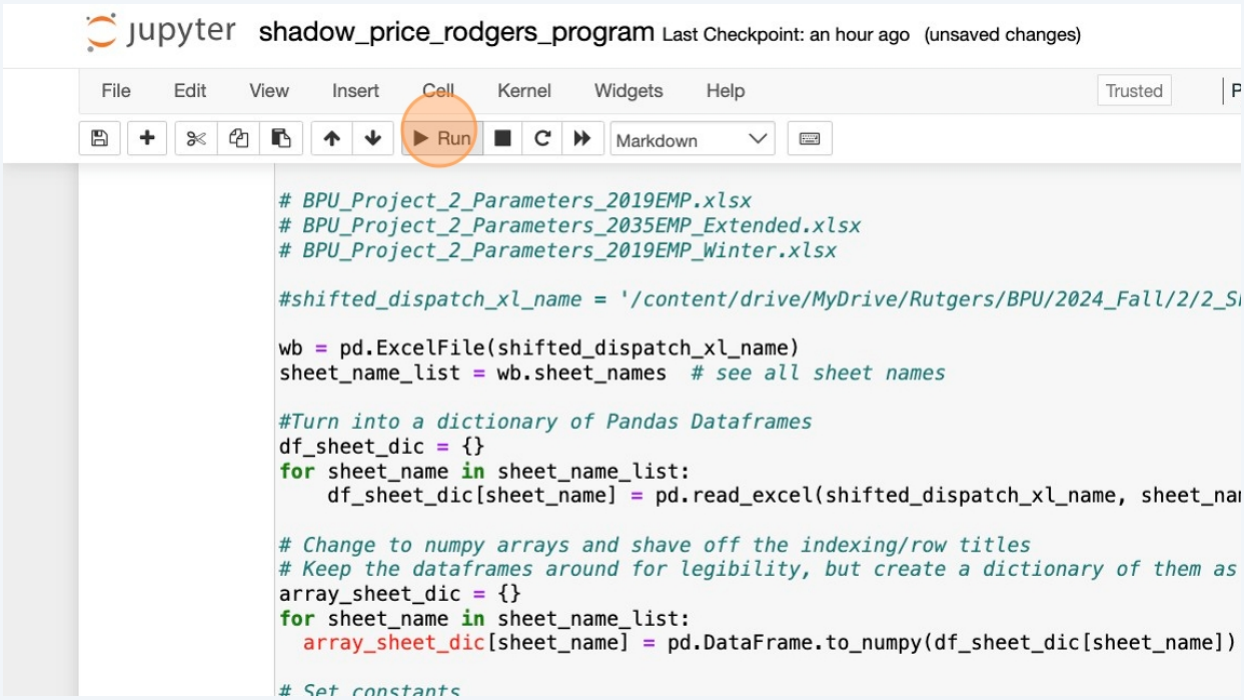
58 Click "Run"



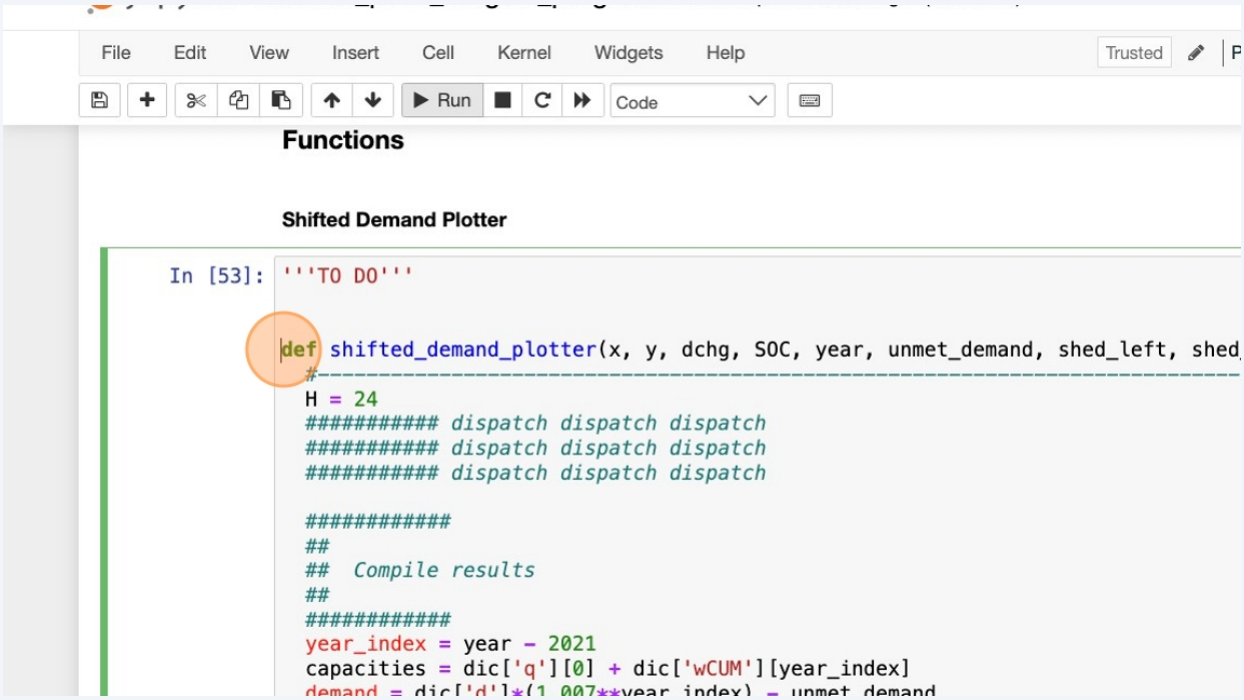
59 Click "wb"



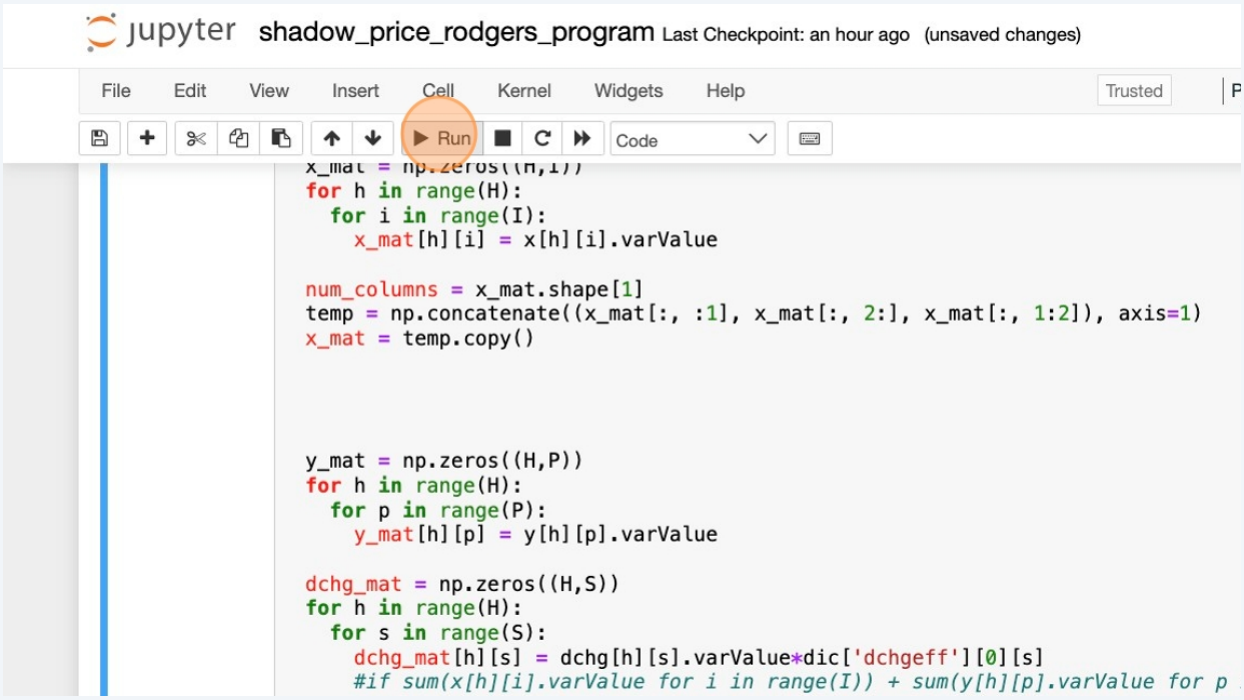
60 Click "Run"



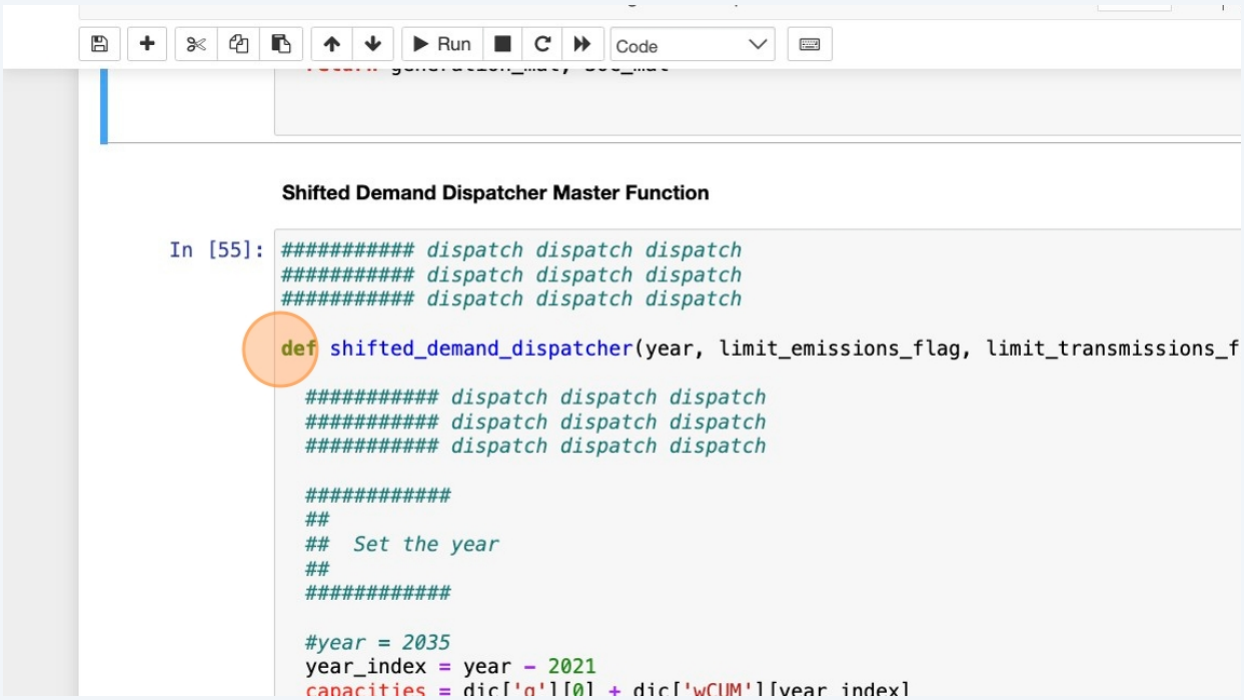
61 Click "def"



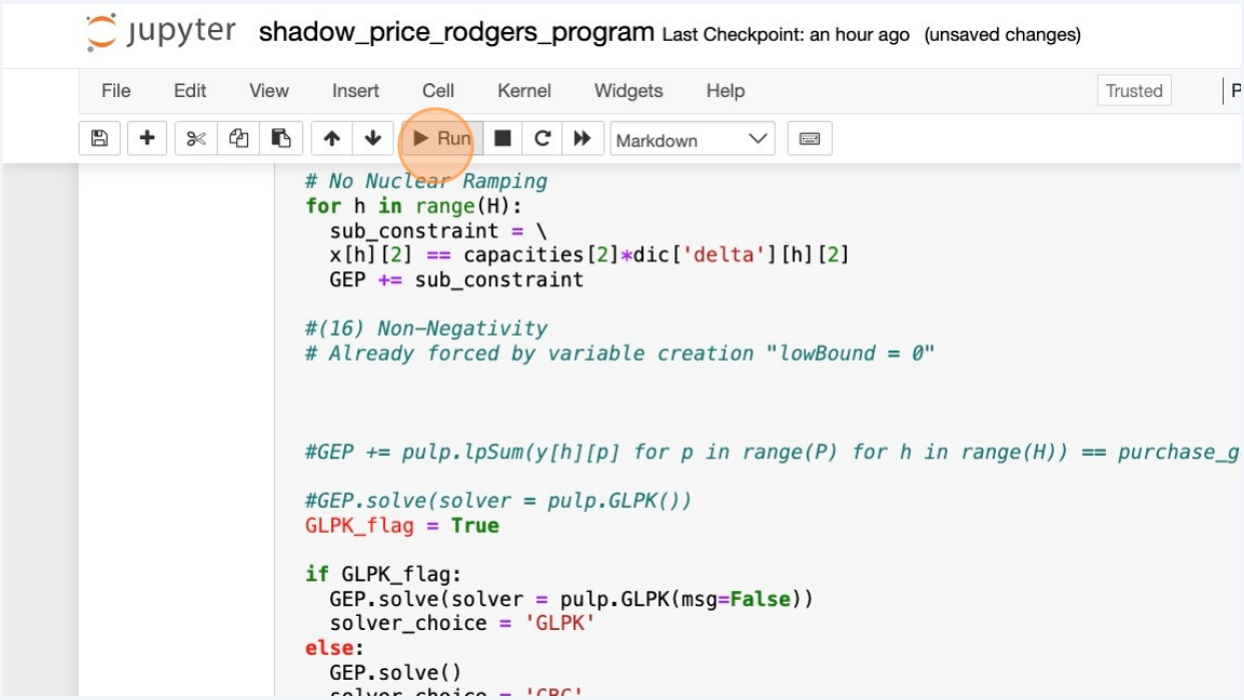
62 Click "Run"



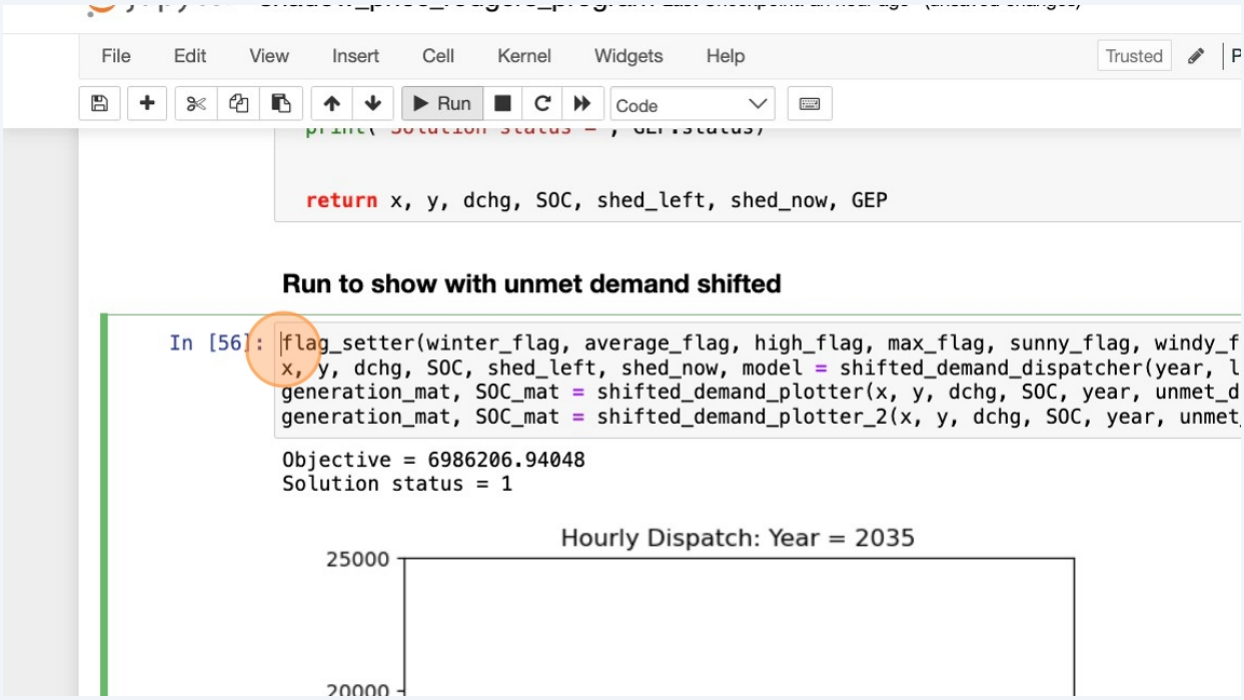
63 Click "def"



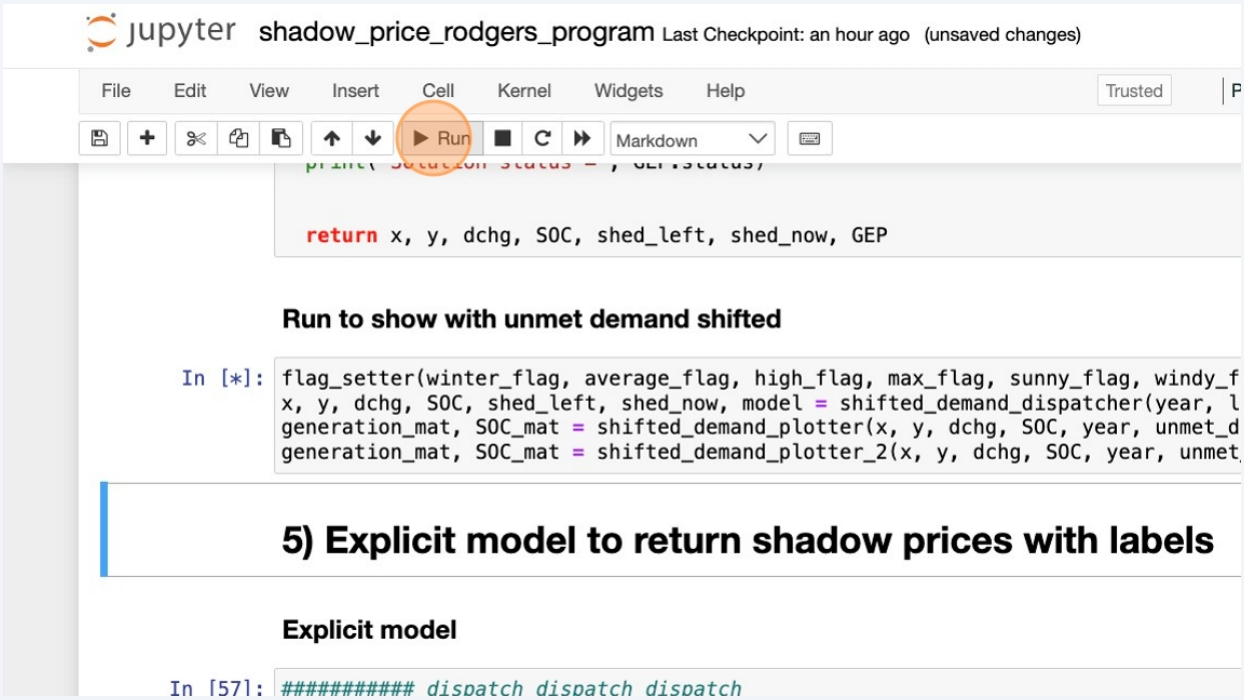
64 Click "Run"



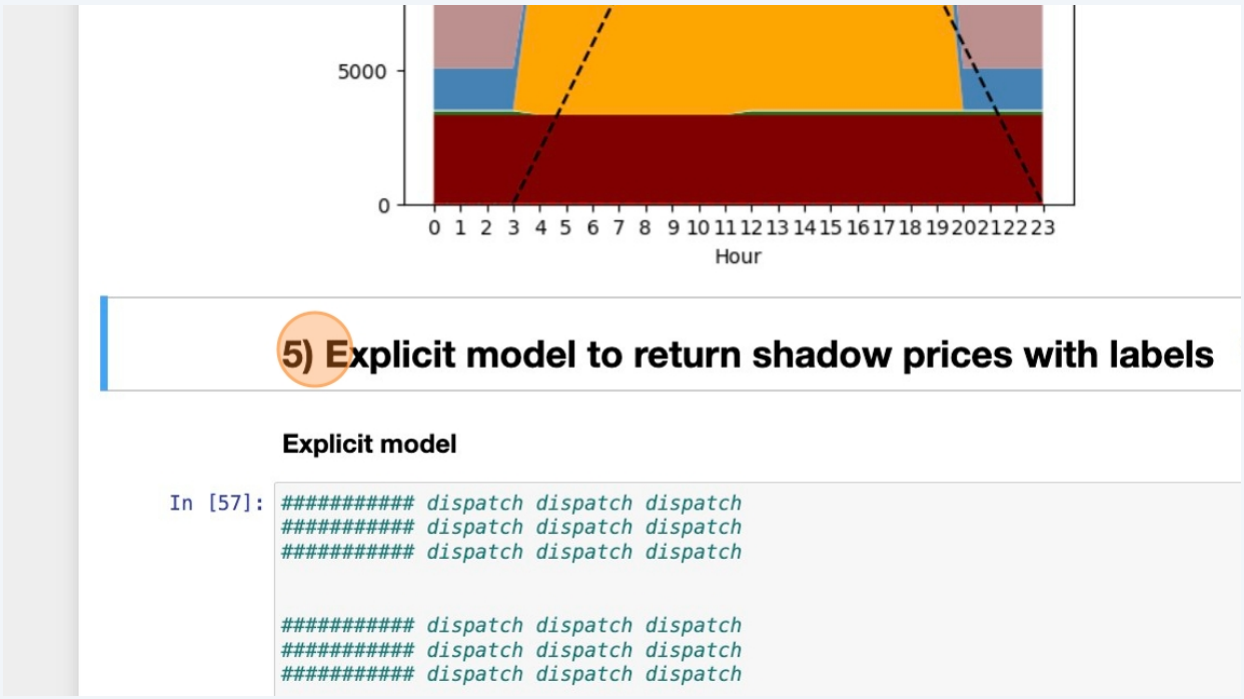
65 Click "flag_setter"



66 Click "Run"



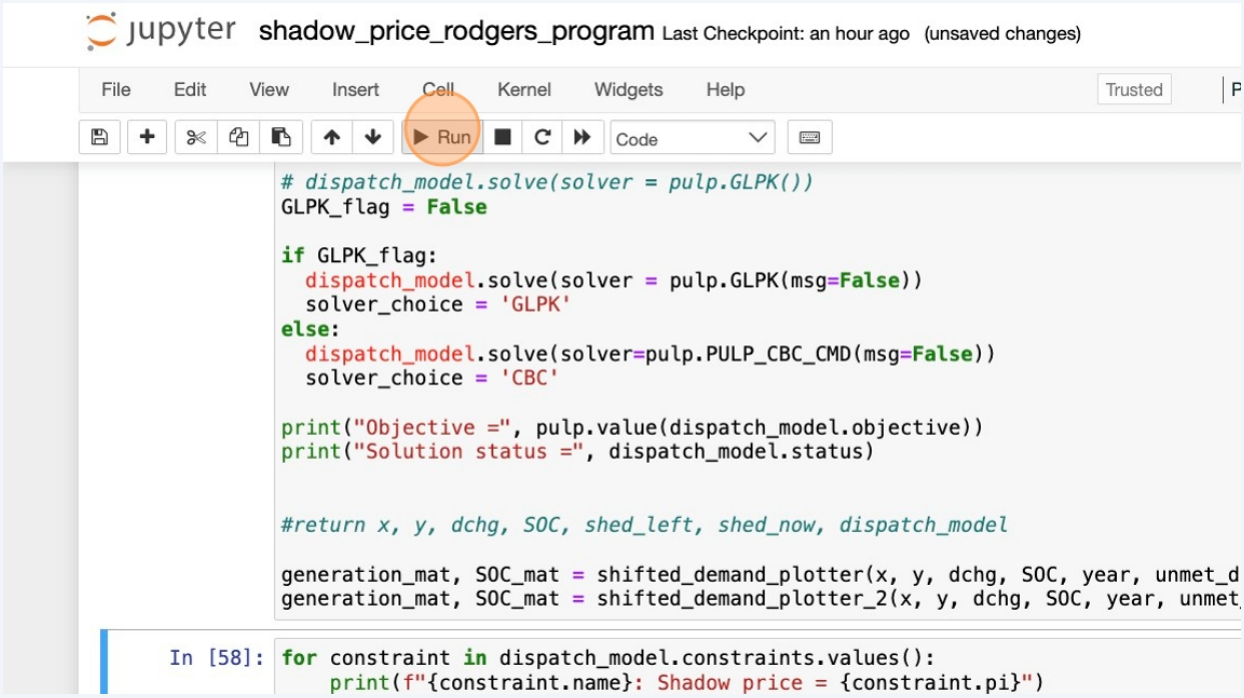
67 Click "5) Explicit model to return shadow prices with labels"



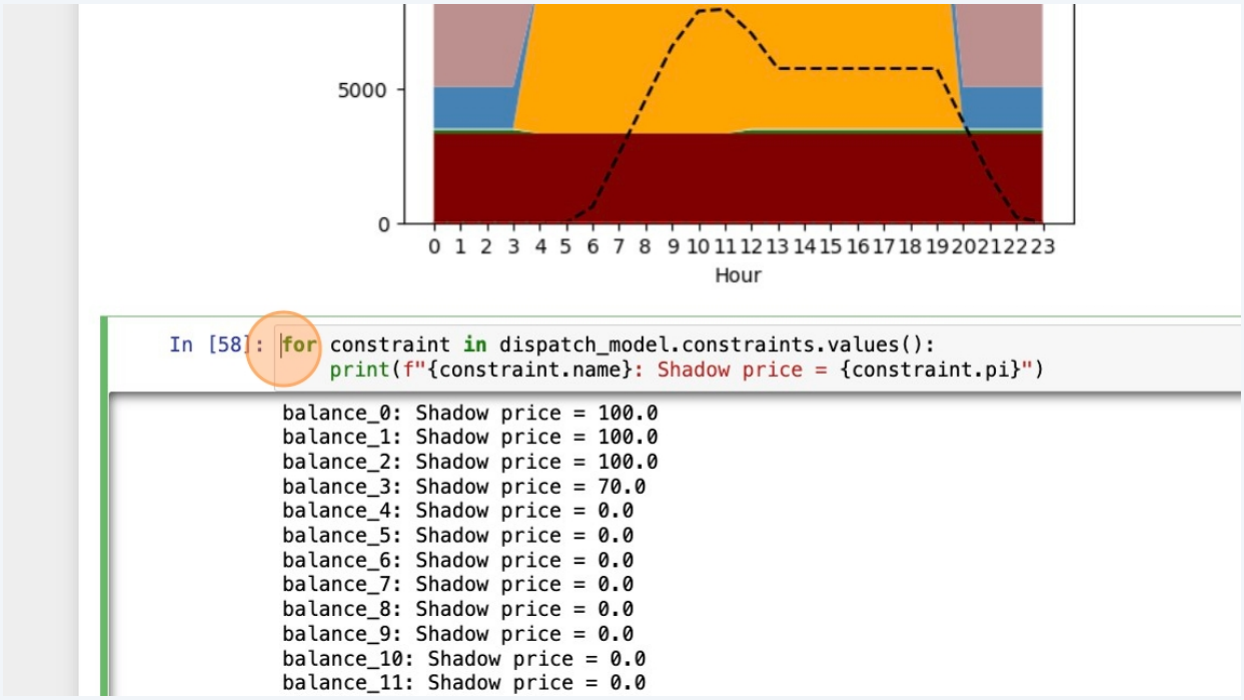
68 Click "year_index"

```
########### 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['PCMAX'] = dic['SOCMAX']
dic['PDMAX'] = dic['SOCMAX']
```

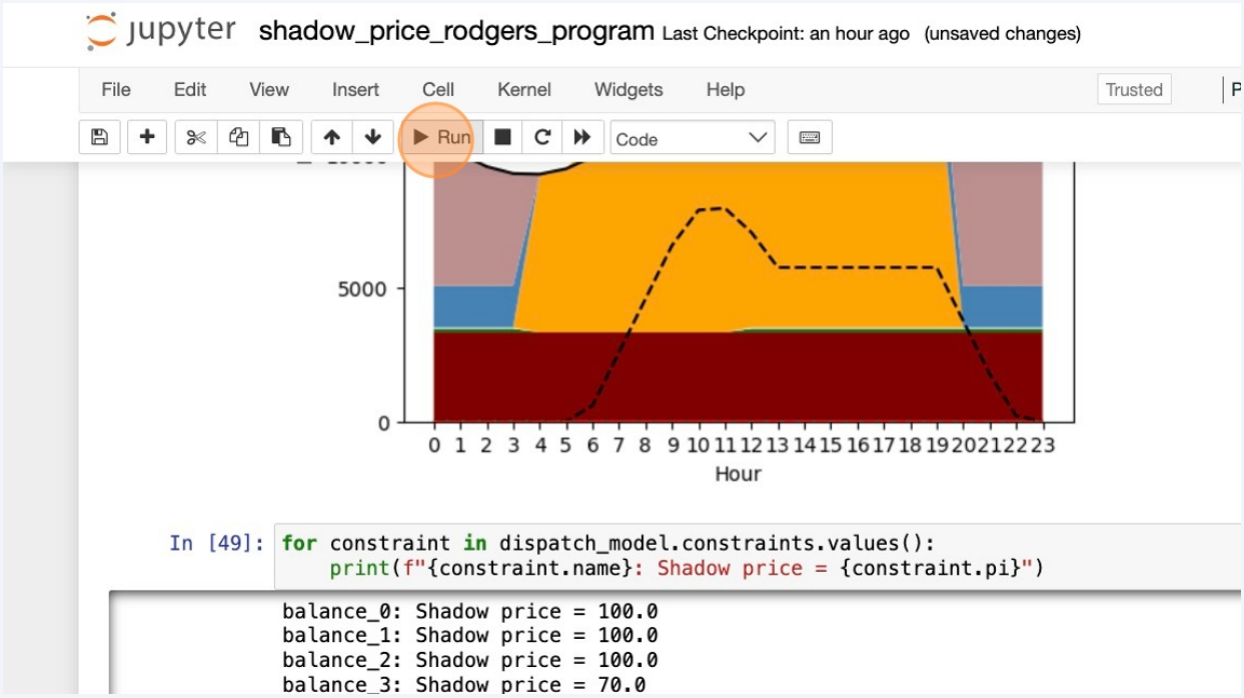
69 Click "Run"



70 Click "for"



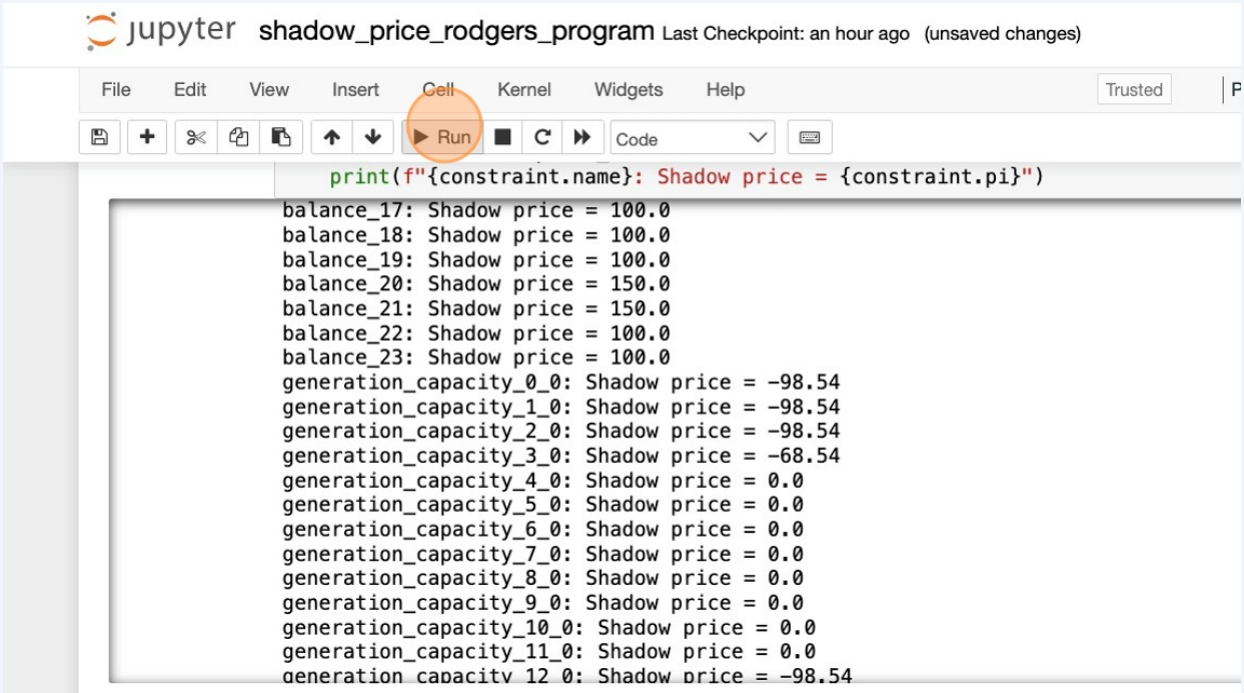
71 Click "Run"



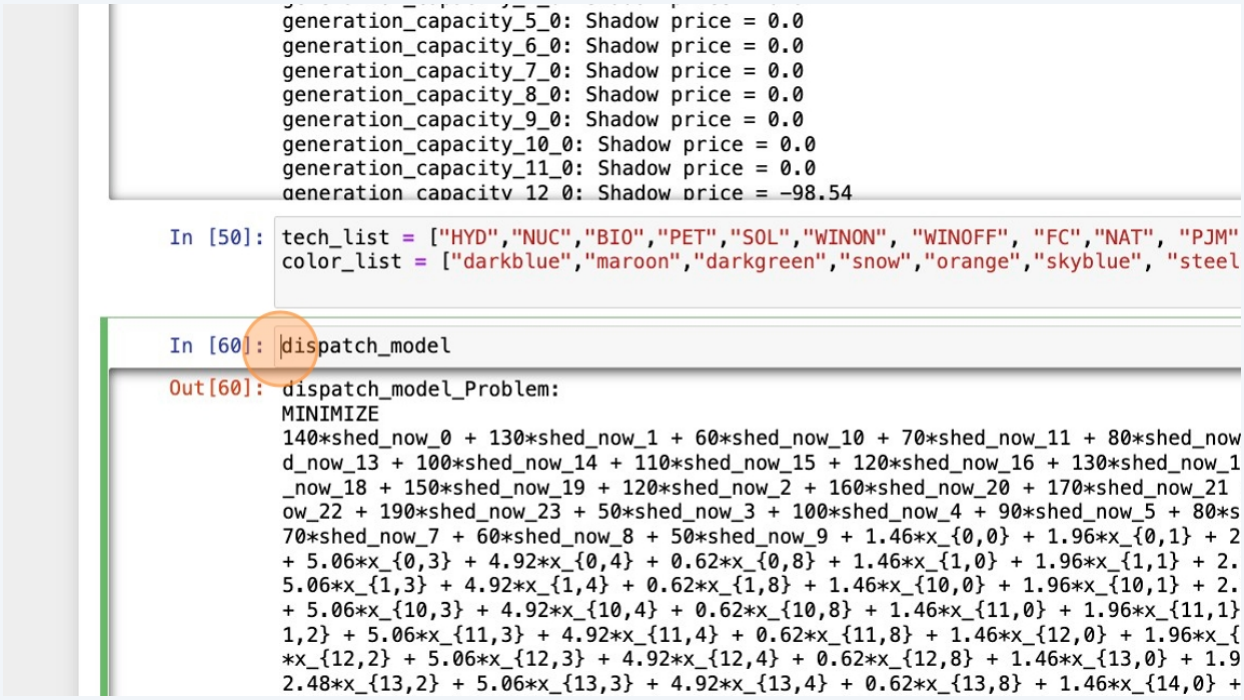
72 Click "color_list"

```
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.
```

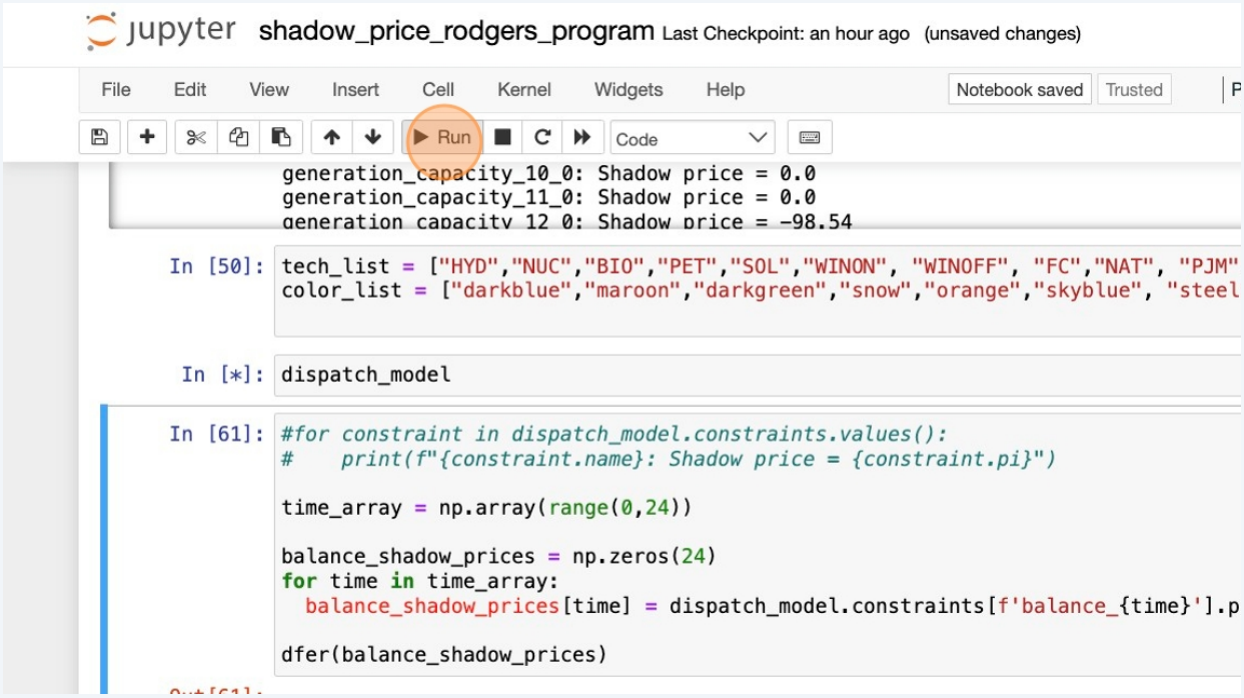
73 Click "Run"



74 Click "dispatch_model"



75 Click "Run"



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} +
```

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"