In [ ]:

```
'''
With the three 's you can add comments of several lines
'''

# This is how you comment out the remainder of one line (could be after a Python command)


'''
=====================================================================

REALLY IMPORTANT ABOUT INDENTS:
    For several commands below (while, if, for)
    the idea is that the subsequent lines start with an indent
    AND the indents last until the set of relevant command lines has ended
    better not to mix tabs and spaces to create indents

=====================================================================
'''

'''
=====================================================================

CREATE SECTIONS:
    with #%% you create separate sections in your program
    Those you can run separately

=====================================================================
'''


'''
=====================================================================

LOADING MODULES

=====================================================================
'''
#%%


'''
There are standard modules that you probably should always load
at the beginning of your program.

- numpy which allows using vectors,matrices,random numbers,equation solvers, and much more)
- matplotlib.pyplot to make figures

here we also load pandas which creates nice tables and
can be used to load data from Excel files like we did in HW 1.

Note we shorten the names which will make codes below  more concise

'''


import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

'''
We may also want to import specific useful functions such as the inversion function as we
do here for matrix inversion
'''

from numpy.linalg import inv

'''
=====================================================================

SECTION 1: VECTORS AND MATRICES (ARRAYS)

=====================================================================
'''

#%%


'''
In many programming languages it is useful/necessary to create dimensions
before you start filling them with values.
'''

# create an empty vector (note that we use a numpy command with the shortened np)
XXX1 = np.empty(10)
# An alternative is to create a vector with arbitrary values (here zeros)
XXX2 = np.zeros(10)
# First display an empty line (makes output easier to view)
print("")
# Disply text
print("display the whole of XXX2")
# Now display the content of XX2
print(XXX2)
print("")
print("display the first and the last element")
print(XXX2[0],XXX2[9])# note the first element is 0 and the last is 9
print("")
XXX3 = np.ones(10)
# create a 10x2 matrix
XXX4 = np.ones((10,2))
XXX4T = np.transpose(XXX4)          # same as XXX4.T
XXX5  = np.array([[5], [8], [31]])  # a 3x1 column vector with these specific values
XXX6  = np.array([[5, 8, 31]])      # a 1x3 row vector with these specific values
XXX7  = np.array([[1., 2.],[3., 4.]])  # 2x2 matrix
XXX7inv = inv(XXX7)
# XXX7check is indeed the indentity matrix
# /// the name "dot" is perhaps confusing. This is not elements by element
XXX7check   = np.dot(XXX7, XXX7inv)
# XXX7compare is NOT the identity matrix because the * operator multiplies the two matrices element by element
# so this requires the dimensions to be the same
XXX7compare = XXX7 * XXX7inv

IIZ1 = 2**3 # this is 2^3

IIZ2 = [XXX7>2.5] # this will tell us whether this inequality is true
print(IIZ2)


'''
=====================================================================

ORDER OF OPERATIONS

!!! be very careful about the order in which operations are executed

Python follows PEMDAS rule, that is,
first brackets, then exponents, then multiplication, then division, then addition, then subtraction

Thus 2*3^4 is equal to 2*(3^4)=14
    2*3^4 is NOT equal to (2*3)^4

But to be safe it is always better to use brackets

=====================================================================
'''


'''
=====================================================================

SECTION 2: FOR LOOPS

=====================================================================
'''
#%%
'''
LOOPS
'''
T = 20
xvalues  = np.zeros(T)
yvalues  = np.zeros(T)
yvalues2 = np.zeros(T)
yvalues3 = np.zeros(T)

for t in range(10): # starts at 0 (default) and it stops when it reaches T so wont do t=T
    xvalues[t] = t
    yvalues[t] = t**2
    print(xvalues[t], yvalues[t])
    print("")
    #
    # IFF you have the latest version of Python
    # then the following would pause in each iteration
    # but pointer has to be right after the word continue
    # input("Press enter to continue")

for t in range(10,T): # starts at 0 (default) and it stops when it reaches T so wont do t=T
    xvalues[t] = t
    yvalues[t] = t**2 + 25
    print(xvalues[t], yvalues[t])


'''
=====================================================================

SECTION 3: FOR LOOP & CONDITIONAL IF STATEMENT

=====================================================================
'''
#%%
x = 2
if x ==2:
    print("YES X IS INDEED 2")

''' Do the same for loop as above but now with conditional IF statement '''

for t in range(T): # starts at 0 (default) and it stops when it reaches T so wont do t=T
    xvalues[t] = t
    if t < 10:
        yvalues2[t] = t**2
        print(xvalues[t], yvalues2[t])
    else:
        yvalues2[t] = t**2 + 50
        print(xvalues[t], yvalues2[t])

'''
=====================================================================

SECTION 4: DEFINE A FUNCTION

=====================================================================
'''
#%%

def functionY(xt,shift):
    if xt < 10:
        yt = xt**2
    else:
        yt = xt**2 + shift

    return(yt)

'''
=====================================================================

SECTION 5: NOW USE THE FUNCTION IN A FOR LOOP

=====================================================================
'''
#%%

for t in range(T): # starts at 0 (default) and it stops when it reaches T so wont do t=T
    yvalues3[t] = functionY(t, 75)

'''
=====================================================================

SECTION 6: MAKE A FIGURE

=====================================================================
'''
#%%
plt.plot(xvalues, yvalues, color="red", label="yvalues")
plt.plot(xvalues, yvalues2, color="blue", label="yvalues2")
plt.plot(xvalues, yvalues3, color="green", label="yvalues3")
plt.title('What a beautiful figure')
plt.legend()

# the following commands will save the figures in chosen format to your computer
plt.savefig('beautiful.png')
plt.savefig('beautiful.pdf')


'''
=====================================================================

SECTION 6: WHILE LOOP AND CONVERGENCE IN SOLOW MODEL

idea of exercise: iterate on law of motion for k(t+1) until k(t+1)
is approximately equal to k(t) (namely less than convergence_criterion)

=====================================================================
'''
#%%


convergence_criterion = 0.0000001

TT            = 300 # T is chosen large enough so we will have convergence << TT
k_values      = np.zeros(TT)
k_values[0]   = 0.1 # initial value
gap           = 100 # choose large number so while loop will get started

alpha         = 0.3
gamma         = 0.2
delta         = 0.1
A             = 1

t             = 0 # initial time period

while gap > convergence_criterion:
    k_values[t+1] = (1-delta)*k_values[t] + gamma*A*(k_values[t]**alpha)
    gap           = np.abs(k_values[t+1]-k_values[t]) # take absolute value
    t             = t+1

print("")
print("==========================================")
print("")
print("Display limiting value and when it was reached")
print(k_values[t], t)
print("")
print("Display analytical value")
print((gamma*delta)**(1/(1-alpha)))
print("")
print("==========================================")
# the following makes sure that stuff will be plotted in a new figure
plt.figure()
#one could add options such as a particular size and use instead
#plt.figure(figsize=(7, 5))
plt.plot(k_values[0:t+1])


'''
=====================================================================

SECTION 7: READ DATA FROM EXCEL
- the data is read from EXCEL as a "data frame"
- then we turn it into an array

=====================================================================
'''
#%%

#data_data_frame = pd.read_excel(r'~wouterdenhaan/Documents/2020-12-28-complete-notold/aaateach/EC_ZB1/Python/data_set_1.xlsx')
data_data_frame = pd.read_excel(r'./data_set_1.xlsx')
data=np.asarray(data_data_frame)
plt.figure(figsize=(2, 5))
plt.plot(data[:,0], data[:,1])

'''
=====================================================================

SECTION 8: WRITE DATA TO EXCEL
- to write an array to EXCEL we have to turn it first into a "data frame"
=====================================================================
'''
#%%
# import xlsxwriter module

data_new = data + 2 # note that this simply adds 2 to each element
data_new_data_frame = pd.DataFrame(data_new)
#data_new_data_frame.to_excel(excel_writer=r'~wouterdenhaan/Documents/2020-12-28-complete-notold/aaateach/EC_ZB1/Python/output2.xlsx')
data_new_data_frame.to_excel(excel_writer="./output2.xlsx")

'''
=====================================================================

SECTION 9: LEARNING MORE

FACT: You will only learn programming by doing it.

- So give yourself tasks and program it.
- For example, now do some of the assignments (but leave the first one for last)
- Or AFTER Marcia has taught you that the OLS estimator can be expressed as (X'X)^{-1}X'Y,
  then load some data and calculate this vector of regression coefficients (and check
  whether you get the same outcome as Stata or R)

- No doubt, at some point you will need commands not included above. Then you could use Google
- or ask someone
- But I found the following also quite useful
    - http://egallic.fr/Enseignement/Python/en/_main.pdf  [pdf]
    - https://egallic.fr/Enseignement/Python/en/opening-remarks.html
- But of course taking a Python courses is also useful

=====================================================================
'''
```