October 23, 2017

Every poerson should make aup a Python command file.

I should make this intoa how to learn python file.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Download Anaconda, a version of Python. From the Windows Command Line, there are four ways to run python programs:

1. python myprogram.py April 12 2017

This first way runs a particular python program you wrote, myprogram.py, and that program takes the words April 12, 2017 and does something to them.

2. python

This starts up an interactive session with this prompt

>>>

In this session you would enter python commands such as

x= 2+2

This command creates the variable x by adding 2 and 2 and on the next line 4 will appear.

3. spyder

If you open a black command line window, then you can type ‘spyder’ and after a minute (a long time) the spyder interface will appear. One part is a left hand big box for writing python programs in and then pressing the green arrow RUN button at the top to run them. You could write

x=2+2

print(x)

After pressing the green arrow button, you will see in the bottom right square, the console, what the output of the batch program is: 4. For a batch program, you need the print(x), not just the creation of the variable, to see any output. You can detach the console and move it to another monitor by double clicking the window title, “IPython Console”.

4. spyder

You can also type directly into the console, which has a prompt that looks like

In [3]:

This will work like interactive with the >>> prompt.

To install a PACKAGE (not a module) such cx\_Freeze, which turns python scripts into Windows executiable, go to Windows Command Prompt terminal and type:

python -m pip cx\_Freeze

tehn turn:

python.exe cxfreeze-postinstall

Use the script cxfreeze1.py. You insert the name such as temp.py and any oher details you like into it. Then:

python cxfreeze1.py build

That will create a directory called build in whcih lots of fiels are, including temp.exe, which runs when you click on it. But it will have an error unless you go into ProgramData/Anaconda3/DLLs/ and copy tcl86t.dll and tk86t.dll into the same directory as the new temp.exe.

It will work then, but for it to work on a windows computer without python installed, I am told you need even more files copies into the directory with temp.exe. You need to copy C:\ProgramData\Anaconda3\Library\lib\tcl8.5\ and also the tk8.5 directory.

http://sebsauvage.net/python/snyppets/index.html#tkinter\_cxfreeze

A FUNCTION

def printWord(word):

print (“This word you gave me is” +word)

#This printWord function we create will add a prefix phrase to the word you type in with it:

#The indent matters.

def squaring(x):

return x\*x

#That last functions makes squaring(3) = 9.

A VECTOR FUNCTION:

Here is what to do to apply a function to an entire vector (or array):

import numpy as np

temp=[1,2,3]

def squaring(x):

return x\*x

vectorfunction= np.vectorize(squaring)

b = vectorfunction(temp); print(b) ;

.........................................................

HOW TO PRINT

print(“This sentence”)

print(round(myfloat, 4)) #This prints to 4 decimal places

print([round(elem,2) for elem in payoff]) #This prints nicely for a list.

#in the Ipython console, you can set %precision 2 in the console (not batch) and numbers but not lists will be formatted nicely.

..............................................................

HOW TO ASK FOR TYPING FROM THE COMMAND SHELL

var1=input()

#This will make var1 a string, always. Convert it with int or float if you want a number.

..............................................................

LISTS and SETS are like this

mylist= [1,2,5, “wokd”] and myset = {5,2, “a word”}

You can extract a single item from a list, but not a set, by mylist[0], which is 1.

#Thsi is how to create a vector of Q values between 2 and Pstar. I hoped to use it.

import numpy as np

Q = np.linspace(0, 20, 200) #This creates a vector of 200 numbers between 0 and 200 equally spaced.

Q1 = Q<=6 #This creates a Boolean Array with TRUE for those of the 200 less than or equal to 6.

print (Q1, Q[Q1]) #Q[Q1] applies the Boolean array to Q and gives a shorter vector only of TRUE numbers.

Q2 = Q[Q1]>2

Q3 = Q[Q1][Q2]

print ( Q[Q1][Q2])

..............................................................

MINIMIZATION

from scipy.optimize import minimize #scipy is part of the anaconda package, I think.

def myfunction: #thisi s what we’ll minimize

return x\*\*2 – 4

answer = minimize(myfunction, 3) #3 is the initial guess. Then I need to print answer to see the result.

..............................................................

SYMBOLIC PYTHON.

<http://www.eg.bucknell.edu/physics/ph329/jupyter/sympy_1.ipynb.pdf> (short, best)

<http://docs.sympy.org/latest/gotchas.html#gotchas>

<https://media.readthedocs.org/pdf/omg/series-docs/omg.pdf>

<http://www.scipy-lectures.org/advanced/sympy.html>

from sympy import \*

from sympy.abc import \* #This makes each letter and greek letter name a symbol automatically.

m = Symbol('m') ; h = Symbol('h'); x = Symbol('x') ; tau = Symbol('tau') ; theta= Symbol('theta') ; u = Symbol('u')

theta=.2; x=1;h=1 #for plotting diagrams

def Fm(m):

return m/h

def fm(m):

return 1/h

def payoff\_s\_low(m):

return Fm(m)\*x + (1-Fm(m))\*(x+m) - tau\*(x+m)

def payoff\_s\_high(m):

return Fm(m)\*(x+h) + (1-Fm(m))\*(x+m) - tau\*(x+m)

temp= diff(payoff\_s\_low(m),m)

mlow = solve (temp,m)

mlow = mlow[0] # The solve command returns a list, and we want the first element of the list, not a list object.

mlowprime = diff(mlow,tau)

temp= diff(payoff\_s\_high(m),m)

mhigh = solve (temp,m)

mhigh= mhigh[0] # The solve command returns a list, and we want the first element of the list, not a list object.

mhighprime = diff(mhigh,tau)

def payoff\_b(m):

return Fm(m)\*0 + integrate( ((x+u) - (x+m))\*fm(m) , (u, m, h))

def welfare(m):

return theta \*(payoff\_s\_low(mlow) + tau\*(x+mlow) + payoff\_b(mlow)) +(1-theta)\* (payoff\_s\_high(mhigh) + tau\*(x+mhigh) + payoff\_b(mhigh) )

temp = diff(welfare(m),tau)

taustar= solve (temp,tau) ; taustar = taustar[0]

plot(welfare(m), (tau, 0, 1), ylabel='welfare(m)' )

sym.exp(x).subs({x:3}) #subsitute 3 for x in exp(x). For muerical examples. It’sa dictionary I think.

00 means infinity.

solve(x\*\*4 - 1, x)

equatino1 = Eq(x+1, 4) #this generates the symblic object x+1==4. ..............................................................

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

WHILE LOOPS

count = 0

while (count < 9):

print ('The count is:', count)

count = count + 1

FOR LOOPS

For x in range(0, 3):

#This uses the range 0,1,2 and does not include 3.

print ("We're on time %d" %(x))

#%d indicates I want to inject a number into the string. After the

#string, we put %(x) to say that it’s x we want to inject.

if x==4:

break

#This command get you out of a loop

#’continue’ would continue looping, going right back to the start of the loop again.

“EXCEPTIONS”, TO KEEP GOING IN SPITE OF ERRORS

try:

... print(widget)

... except:

... print(" widget isn't defined!")

>>> try:

... print("pop")

... except:

... print("An error has occurred!")

... else:

... print("Everything went smoothly!")

... finally:

... print("Finishing up now...")

REGULAR EXPRESSIONS: the re module

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

#re.findall is for returning search results. re.

string.join([a,b,c])

#'astringbstringc' for joining a list together with a string in between.

re.findall('a\*c','abcd6abcde')

#abcd6abc finds the longest a...c words. The output type is a list even though it only has one item.

re.findall('a\*?c', 'abcd6abcde')

#[abc, abc] finds the shortest a...c words.

re\_findall('a.c', 'abcd6abcde')

#[abc,abc] there must be exactly one letter in between a and c.

re.search('abc', 'abcd6abcde')

#This looks for the first match for “abc” and then says whether there is a match or not.

x=re.search('theeee', 'abcd6abcde')

#On the console, there is no match, so nothing is printed.

print(re.search('theeee', 'abcd6abcde'))

#This returns 'None'

re.findall('theeeee', 'abcd6abcde')

#This returns [] because it is an empty list.

#############################################################################

john5 = """ Search the scriptures; for in them ye think ye have eternal life:\

and they are they which testify of me.

"""

#############################################################################

re.findall('th\*', john5)

# ['th', 't', 'th', 'th', 't', 'th', 'th', 't', 't']

set(re.findall('th\*', john5))

# {'th', 't'}

list(set(re.findall('th\*', john5)))

#THis turns the set back into the list ['t', 'th']

set(re.findall('th?', john5))

# {'t', 'th'}

set(re.findall('th\*', 'thhhh'))

# {'thhhh'}

set(re.findall('th?', 'thhhh'))

# {'th'}

set(re.findall('th+', john5))

# {'th'} Just th, because + requires at least 1 repetition of h

set(re.findall('th+', 'thhhh'))

# {'thhhh'}

#############################################################################

set(re.findall('the\*', john5))

# {'th', 'the'}

set(re.findall('the?', john5))

# {'th', 'the'}

set(re.findall('the+', john5))

# {'the'}

#############################################################################

set(re.findall('s[a-z]\*s', john5))

# {'scriptures'}

# It says to look for phrases starting with s,

# then having some small letters, maybe, but no other characters,

# then another s.

set(re.findall('s[a-z]?s', john5))

# set() It doesn't find the prhase.

#############################################################################

set(re.findall(' [a-z]{5} ', john5))

# {' think ', ' which '} all the 5 letter words with

# a space before and after

set(re.findall('[^a-z][a-z]{5}[^a-z]', john5))

# {' think ', ' which ', 'Search '} all the # 5 letter

# words even if there is no space before and after.

System Commands for doing WIndows things

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

RESTARTING YOUR COMPUTER

# The following lines cause my Windows computer to restart.

import subprocess

subprocess.call("shutdown /r")

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

# OPERATING SYSTEM COMMANDS.

import os

print(os.getcwd())

x1 = 'C:\\Users\\erasmuse\\Dropbox\\PYTHON'

x2= os.chdir(x1)

print(os.getcwd())

# THESE ARE INPUT AND OUTPUT COMMANDS.

import os

os.getcwd()

x = 'C:\Users\erasmuse\Dropbox\PYTHON'

y= os.listdir(x)

#os.system("dir")

#print(y)

y = str(y)

import re

v1= re.findall('[A-Za-z]\*\.txt',y)

#print(v1)

import time

v3= time()

print(v3)

v2=open("temp.txt",'r+')

#This opens up a temporary Python file v2, which we will write to temp.txt later.

v2.write('Hello World Here')

#This says to write Hello World Here to v2, but not yet to temp.txt.

v2.close()

#This writes to temp.txt, closing down v2.

v2=open("temp.txt",'r+')

print(v2.read())

v2.close()

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

import webbrowser

webbrowser.open('http://inventwithpython.com/')

#This opens up your browser at the webpage <http://inventwithpython.com/>

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

import sys

import pyperclip

print(sys.argv)

if len(sys.argv) > 1:

# Get address from command line.

address = ' '.join(sys.argv[1:])

else:

# Get address from clipboard.

address = pyperclip.paste()

import webbrowser

print(address)

webbrowser.open('https://www.google.com/maps/place/' + address)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

#installing packages

# In the command window, type, to insstall the package “clipboard”, type

python –m pip install clipboard

# It will untar the files and do everything necessary.

#This must be done at the command prompt.

#How does one cut and paste intot he command prompt in windows?

INPUTTING INFO FROM THE COMPUTER.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

# python 3

# A program to use the command line

#Name: commandline.py

print("This is a test")

import sys

#sys module: for dealing with command line input

print(sys.argv)

#sys.argv is a variable that has the python program name and also the

# command line inputs

if len(sys.argv) > 1:

#sys.argv is a variable that has the python program name and also the

# command line inputs

address = ' '.join(sys.argv[1:])

#If the file was run with input, then make an address varible out of

#evertyhing but the first item, the program name, of sys.argv.

print(address)

else:

#Else means that sys.argv only has one item, in whcih case it only

# has the program name in it.

print("You didn't put anything after temp.py like you should have")

DOING THINGS WITH WEB FILES—SCRAPING.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

# python 3

# A program to download a little bit of Romeo and Juliet.

#Name: gettingwebfiles.py

#JUly 14 2017

print("This is a test")

import requests

myvariable = requests.get('https://automatetheboringstuff.com/files/rj.txt')

#Note that requests.get creates a res object even if the webpage doesn exist

# So the program won't stop just because it can't find the webpage.

print(type(myvariable))

#Note that res is not a string object; it is a special kind of object

if myvariable.status\_code == requests.codes.ok:

#res.status\_code is the status of res.

#If the webpage is there OK with no probelm, ...

print("OK")

else:

print("not ok ")

#Try this out with a nonexistent web page to see what happens.

print(len(myvariable.text))

#res.txt is the variable res turned into.

#Print hte length of the text.

print(myvariable.text[:250])

print(myvariable.raise\_for\_status)

#This will be 200 if the status is OK-- that is, if the download worked.

try:

myvariable.raise\_for\_status()

#this is to see if everything went OK.

print("The download was OK")

#Print the message if it was OK. We could have omitted the printing.

except Exception as exc:

#I don't know what this last line means.

print("There was a problem %s" % (exc))

# I don't know how this works either.

myvariable2 = open('RomeoAndJuliet.txt', 'wb')

#This creates a new variable myvar2 by creating the nonexistent file

# RomeoAndJuliet.txt with the wb tag to allow it to be written on, even in

# binary mode.

for chunk in myvariable.iter\_content(100000):

# do something in 100,000 byte chunks with the object myvar.

myvariable2.write(chunk)

# write the chunk to the end of the object myvar2

print(len(chunk))

myvariable2.close()

#Close myvar2 because we're done with it.

#When we close myvar2, that writes it to the hard drive too,I think.

#I think earlier it wasn't written there. We coudl check.

BEAUTIFUL SOUP WEB SCRAPING.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

# File name: beautifulsoup.py

# A program to try to do parsing of an HTML file to extract bits of it.

# July 15, 2017

# python 3

#In learning this, it is good to type the stuff into the batch file,

# then to run it by cuttinga nd pasting each line into interactive

# separately.

print("This is a test just to get things started and see if the program \

has output")

import requests, bs4

res = requests.get('http://nostarch.com')

res.raise\_for\_status()

var1= bs4.BeautifulSoup(res.text,"lxml")

#Try this last with just

# var1= bs4.BeautifulSoup(res.text,"lxml")

# In my Anaconda version of python, this runs OK, but at the end

# is an error message that says if you don't want the error

# message you need to insert the "lxml" because it is using

# beautiful soup in a particular way.

print(type(var1))

var2 = open('example.html')

var3 = bs4.BeautifulSoup(var2,"lxml")

print(type(var3))

var2= open('example.html')

var4 = bs4.BeautifulSoup(var2.read(),"lxml")

elems = var4.select('#author')

print(type(elems))

print(len(elems))

print(type(elems[0]))

# I don't understand this one.

print(elems[0].getText())

#This prints out the text in the first item of elems, but leaves out

# the HTML tags.

print(str(elems[0]))

print(elems[0].attrs)

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

#googlesearch3.py

# Finding and opening numerous search results on Google using beautiful soup.

#July 19, 2017

import os, bs4

import re

import subprocess

import webbrowser

import requests

var1 = 'North Korea ballistic missles'

# var1 is what we are searching for on Google.

res=requests.get("https://www.google.com/search?q="+var1)

#This gets the web page with the search results for var1.

webbrowser.open("https://www.google.com/search?q="+var1)

#This shows us what the search results are.

var2= bs4.BeautifulSoup(res.text,"lxml")

#var 2 is a beautiful soup object, a reorganized version of the web page.

#Try this last with just

# var1= bs4.BeautifulSoup(res.text,"lxml")

# In my Anaconda version of python, this runs OK, but at the end

# is an error message that says if you don't want the error

# message you need to insert the "lxml" because it is using

# beautiful soup in a particular way.

var3 = var2.find\_all("h3", class\_="r")

#THis creates a Beautiful Soup "result set" object.

print(var3[0])

#This prints the first item in the var3 result set grouping.

print(var3[0].a)

#This prints the just the <a>sdfsf </a> element of the

# first item in the var3 result set grouping, an object called

# a bs4 element tag.

#Note that var3.a itself doesn't exist as a list---its only individual

#items we can apply the "a" method to.

print(var3[0].a["href"])

# This pulls out just the href=jkljkljkl part of the first item,

# and makes it into a string.

webbrowser.open("www.google.com" + var3[0].a['href'])

# Since we add back int the google.com prefix, we can now open the web page.

for item in range(0, 3):

print("www.google.com" + var3[item].a["href"] )

#This prints the first 3 hrefs, after putting www.google.com in front to

#complete the address.

for item in range(0, 3):

webbrowser.open("www.google.com" + var3[item].a['href'])

#This opens the first 3 hrefs, after putting www.google.com in front to

#complete the address.

pageaddress=[]

#This creates a list for us to fill up with items in the for loop that is next.

for item in range(0,20):

results = str(10\*item)

pageaddress = pageaddress + ["https://www.google.com/search?q="+var1 +"&safe=strict&start="+ results ]

#This shoudl create a list with the address having 0,10,20, etc. at the end.

#We could loop opening webpages using this pageaddress list, if we wanted to

#do things with 20 pages of google search results.

#We could also just use requests.get to get each page and look in it for, say,

#the word "Tokyo" and write down which pages have that word.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

REGULAR EXPRESSIONS

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

# regular\_expression.py

# July 15, 2017

# PYthon 3

#What you should do with this is run each command on the consolse.

import re

john5 = """Search the scriptures; for in them ye think ye have eternal life:\

and they are they which testify of me.

"""

"""

START COMMENT

Here is how to do a multiline comment

Multiline comment

Multiline comment

Multiline commentMultiline comment

Multiline comment

END COMMENT

"""

print(john5)

#Notice how the line breaks show up as \n.

type(john5)

#############################################################################

re.search('eternal', john5)

# re.search creates a special object, not a string.

if re.search('eternal', john5):

# In this special case, the IF statment is TRUE if eternal is in John 5.

print("The word \'eternal\' is in John 5")

#SHift-Escape to get back to the command line in the console

var1=re.findall('ye', john5)

var2 = len(var1)

print("The word \'ye\' is in John 5 "+ str(var2) +" times")

#############################################################################

re.split('ye', john5)

#This splits up the text using 'ye' as the divider between items.

#############################################################################

re.findall('th.y', john5)

#This finds they, thay, thoy.

re.findall('the.', john5)

#This finds the , they, them.

re.search('the.', john5)

#This looks for the first match for the, they, them and then stops.

re.search('theeee', john5)

#On the console, there is no match, so nothing is printed.

print(re.search('theeee', john5))

#This returns 'None'

re.findall('theeeee', john5)

#This returns [] because it is an empty list.

type(re.findall('theeeee', john5))

#The type of this is a list, even though it has no items.

re.findall('theeee', john5)

#This returns [] because it is an empty list.

type(re.findall('theeee', john5))

#The type of this is a list, even though it has jsut one item.

#############################################################################

LITERAL STRINGS

john5 = """ Search the scriptures; for in them ye think ye have eternal life:\

and they are they which testify of me.

"""

#############################################################################

re.findall('th\*', john5)

# ['th', 't', 'th', 'th', 't', 'th', 'th', 't', 't']

set(re.findall('th\*', john5))

# {'th', 't'}

list(set(re.findall('th\*', john5)))

#THis turns the set back into the list ['t', 'th']

set(re.findall('th?', john5))

# {'t', 'th'}

set(re.findall('th\*', 'thhhh'))

# {'thhhh'}

set(re.findall('th?', 'thhhh'))

# {'th'}

set(re.findall('th+', john5))

# {'th'} Just th, because + requires at least 1 repetition of h

set(re.findall('th+', 'thhhh'))

# {'thhhh'}

#############################################################################

set(re.findall('the\*', john5))

# {'th', 'the'}

set(re.findall('the?', john5))

# {'th', 'the'}

set(re.findall('the+', john5))

# {'the'}

#############################################################################

set(re.findall('s[a-z]\*s', john5))

# {'scriptures'}

# It says to look for phrases starting with s,

# then having some small letters, maybe, but no other characters,

# then another s.

set(re.findall('s[a-z]?s', john5))

# set() It doesn't find the prhase.

#############################################################################

set(re.findall(' [a-z]{5} ', john5))

# {' think ', ' which '} all the 5 letter words with

# a space before and after

set(re.findall('[^a-z][a-z]{5}[^a-z]', john5))

# {' think ', ' which ', 'Search '} all the # 5 letter

# words even if there is no space before and after.

PLOTS, DIAGRAMS, FIGURES

**import** **matplotlib.pyplot** **as** **plt**

**import** **numpy** **as** **np**

fig, ax = plt.subplots()

x = np.linspace(0, 10, 200)

y = np.sin(x)

ax.plot(x, y, 'b-', linewidth=2)

plt.show()

**import** **matplotlib.pyplot** **as** **plt**

import csv

#My data for the second graph is in temp.csv.

with open('temp.csv', 'r') as f: #I haven't seen this syntax before.

reader = csv.reader(f)

temp1 = list(reader)#This createsa list of lists,each containing a #single string, unfortunately.

temp2= [i[0] for i in x] #This converts the list of lists into a list of #strings

data2 = [float(i) for i in temp2] #A condensed way to do a FOR loop.

data2 = sorted(data2)

range2=range(0,len(data2))

fig2 = plt.plot(range2, data2,'ko') #Black big dots. for little ones, `k.’

plt.savefig("temp.png")

plt.show(fig1)#if I had two figures, plot.show(fig1, fig2).

plt.close("all")

#########################################################

#A PLOT

import matplotlib.pyplot as plt

import numpy as np

fig, ax= plt.subplots(figsize=(20, 10))

#We want to get rid of the box around the diagram this way:

ax.spines['top'].set\_visible(False)

ax.spines['right'].set\_visible(False)

#This is a better way to label the axes than using the xlabel, ylabel command:

plt.text(-1,15 ,"Price", fontsize = 32)

plt.text(8,-2 ,"Quantity", fontsize = 32)

#Here are the things to plot.

#Q is a set of 200 points between 0 and 20 at which to sample the function plotted.

Q = np.linspace(0, 20, 200)

Ps = 6+Q/2

Pd= 12- Q/4

#The x axis will have ticks at intervals of 4 and the y of 2.

plt.xticks(np.arange(0, max(Q)+1, 2.0))

plt.yticks(np.arange(0, max(Ps)+1, 2.0))

# Set x, y limits for the points covered by the diagram.

plt.xlim(0, 9)

plt.ylim(0, 16)

# Add some text to point (14, 16) using Latex to get a subscript.

plt.text(3,8.5 ,"$Supply, Q\_s$", fontsize = 24)

plt.text(5, 11,"$Demand, Q\_d$", fontsize = 24)

#The annotate method inserts an arrow and a phrase.

plt.annotate(r'$P,Q$',

xy=(2, 7), xycoords='data',

xytext=(+10, +30), textcoords='offset points', fontsize=24,

arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2"))

coords\_to\_annote = [(1,9),(.5,11.5) ] #We can put two arrows in.

for coords in coords\_to\_annote:

ax.annotate('Consumer Surplus', xy=coords, xytext=(1, 14), fontsize=24,

arrowprops=dict(arrowstyle="->", connectionstyle="arc3,rad=.2")

)

#plt.plot is the object that has the actual plot.

#Making it all black will prevent wasteful use of the color printer.

plot1=plt.plot(Q, Ps, linewidth=4, color='black')

plot2=plt.plot(Q, Pd, linewidth=4, color='black')

#Here is how to connect two points (0,10) and (20,10) with a line:

plt.plot([0, 8],[7,7], linewidth=1, color='black')

plt.plot([2, 2 ],[7,11.5], linewidth=1,color='black')

plt.plot([0, 2 ],[11.5,11.5], linewidth=1,color='black')

#Here is how to put black (k) dots (o) at 3 points:

plt.plot([0,0,0],[8,16, 24], 'ko', clip\_on=False)

#CLip on false lets the dots cover the axis

#The next command decides how much white space to put around the plot when it is saved.

plt.subplots\_adjust(left=0.18, right=0.9, top=0.9,bottom= 0.15)

plt.savefig("ans01-09b.png")

#Showing hte plot on the screen has to come after saving it to a file.

plt.show()

plt.close("all")

GRAPHIC USER INTERFACE IN PYTHON FOR WINDOWS

Use the tkinter library to get windows to pop up, and buttons, and such. http://effbot.org/tkinterbook/tkinter-hello-again.htm

Here is a simple program:

from tkinter import \* # note that module name has changed from Tkinter in Python 2 to tkinter in Python 3

top = Tk()

text = Text(top)

text.insert(INSERT, "Hello.....")

w = Label(top, text = "here is a label")

w.pack()

text.pack()

top.mainloop()

top = Tk()

var = StringVar()

label = Message(top, textvariable = var, relief = RAISED )

var.set("How are you today?")

label.pack()

top.mainloop()

from tkinter import messagebox

top = Tk()

top.geometry("100x100")

def helloCallBack():

msg = messagebox.showinfo( "Hello World")

B = Button(top, text = "Hello", command = helloCallBack)

B.place(x = 50,y = 50)

top.mainloop()

What thsi does is say that you don’t need to put the module name tkinter before all its commands. top just starts things off. text is a variable consisting of a textbox, in our basic top root object. Then we insert the text we want. Then we pack it together. Then we add a label tot he bottom of the window. Then we end with the mainloop command, which says to not do antyhing more until the user closes the window created.

Then a second window appears, saying How are you today. When it is closed, a third window appears, a button that if pressed generates a new window saying Hello World.

IMPORTING DATA FINDING CORRELATION MATRIX

import numpy as np

data1 = np.genfromtxt("data1.csv", delimiter=',', dtype=None,names=True,skip\_header=0, skip\_footer=0 )

#Python has trouble reading strings without delimiters; it give b'theentry' instead of 'theentry'

print ( "data1.csv that is now named data")

print(data1)

print(data1.dtype.names)#This prints the names of the columns.

#inn1\_16= data1['inn1\_16'] #This pulls out the team column of the data1 matrix.

#inn1\_15= data1['inn1\_15']

#print(inn1\_16)

for item in data1.dtype.names:

globals()[item] =data1[item]

print(inn1\_15) #This variable is now created, along with the others using tht loop.

print(np.corrcoef(inn1\_15, inn1\_16)) #for a correlation matrix.