February 8, 2020

 I should make this into a how to learn python file. C:\Users\erasmuse\Dropbox\PYTHON\optical-mark-recognition

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

import sys ; sys.exit();exit(); # ENDING A SCRIPT: (all three are needed). CTRL-C for BREAK.

If you get stuck on the console, often you just need to close a parenthesis for it to be willing to go on.

%reset -f #Type this in the IPython console to clear all variables you created.

import gc; gc.collect() #This might work in a batch file. DOES NOT I think.

HERE STARTS THE ORGANIZED TOPICS

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

 Download Anaconda, a version of Python. When I search on my computer for Anaconda, I get an option for an Anaconda terminal. What I have below doesn’t work any more. 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 use:

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”.

 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.

4. Karst.

 To use the Indiana U. karst machine, get a terminal (Putty is best) to start a session at karst.uits.iu.edu. Load up Python 3.5, because the default python is Python 2. mpmath is already installed. invertlaplace() works.

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:

 MAYBE CONDA INSTALL, maybe something else.

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.

 INSTALLING MODULES

 pip install requests #This install the requests module. Run it from Spyder.

pip install antlr4-python3-runtime #This from the terminal installs the antlr4 package.

BEST: pip install sympy # to install symbolic python. In the TERMINAL

 or pip install mechanize # worked fine.

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>

To test whether a module installed, e.g. mpmath, try “mpmath.runtests()”

<https://conda.io/docs/user-guide/tasks/manage-pkgs.html#installing-packages>

conda install -c stsci imutils This is the way most packages ae installed.

SYNTAX

 args and kwargs: args are unnamed arguments of a function, and kwargs (keyword arguments) are named ones.

 Thus, in the function, invert(f, method=smith), f is an arg and smith is a kwarg. When a function is listed as

invert(\*args, \*\*kwargs), that just means that it has some arguments and then has some kwargs too.

 #To stop a program, but not restart the shell/interpreter, don’t use exit(), use:

import sys

sys.exit()

#No—that doesn’t work. It goes on and tries to carry out the rest o the script.

EXAMPlEs OF CODE: http://code.activestate.com/recipes/langs/python/?query\_start=1

FINDING THE DIRECTORY: In TERMINAL, type: where python. Get: C:\ProgramData\Anaconda3\python.exe

 EVEN Better, maybe, do this in terminal: dir site.py /s

 MODULES are stored in the python directory in subdirectory: /lib/site-packages

C:\Users\erasmuse\AppData\Local\Continuum\anaconda3\Lib\site-packages

NEW ONES GET INSTALLED BY MISTKAE TO: c:\programdata\anaconda3\lib\site-packages

import os; os.chdir('c:\\programdata\\anaconda3\\lib\\site-packages\\')

 and then “import mechanize” or whatever hte module is. Clunky.

os.getcwd() #To find the current working directory

os.chdir('c:\\programdata\\anaconda3\\lib\\site-packages\\') #Changing the directory

os.chdir('c:\\Users\\erasmuse\\Dropbox\\PYTHON\\') #Changing the directory back

v1 = ‘This is a sentence string but I want to write my code on very short lines, so I use a backslash \

to continue to the next line. I could split here \nwith a control-n, and then it would print as two lines.’

#In the consol if you type v1, it doesn’t do the print formatting, though. The VARIABLE has control characters in it.

PYTHONPATH= c:\programdata\anaconda3\lib\site-packages #I can’t get this to work.

Exponents are not x^2, they are x\*\*2

To BREAK, or INTERRUPT code that is running and has stalled out, press CTRL-c.

STRINGS.

 <http://www.codetable.net/unicodecharacters?page=10> For unicode characters, a list.

 chr(13000) # yields '㋈'

h = '\u3456' #Then if you print h, you get '㑖'

'The number {0:.2f} is smaller than {1}.'.format(3.1415, 'four')

 #Yields: 'The number 3.14 is smaller than four.'

v1=3.1.4159

print ("%0.2f is the number"% v1)

#Yields: ‘3.14 is the number”

<http://www.catb.org/jargon/oldversions/jarg262.txt> Programmer Slang.

A FUNCTION

def printWord(word):

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

 def f1(\*args): #\*args means that it can have any number of arguments, \*\*kargs for keyword arguments.

 print("the functoin prints", \*args)

f1(2,3,4) #This yields:“the function prints 2 3 4”.

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) ;

def myGenerator(n): #THIS IS A GENERTOR.

 yield ['First thing returned is n', n]

 yield 'These'

 yield 'words'

 yield 'come'

 yield 'one'

 yield 'at'

 yield 'a'

 yield 'time'

z = myGenerator(99)

print(next(z)) #THis rpints out the first yield thingw ith the string and num er n

print(next(z)) #This prin out the second yield thing, ‘these’

for item in z:

 print(item)

LAMBDA FUNCTION: apply to list [0,1,2,3] to get [0,1,4,9]. This creates a functionwithout a name for a single use.

[i for i in map (lambda x: x\*\*x, z)]

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

 HOW TO PRINT

print(“This sentence”)

print(round(myfloat, 4)) #This prints the number myfloat 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 fixed.

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

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.

var2 = str(123) #covnerts a number to a string.

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

 LISTS and SETS

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

 You can extract a single item from a list by mylist[0], which is 1. Doesn’t work for sets, which have no ordering for their items, or a dictionary, which is a set with labellings like d1= {‘name’: ’bob’, ‘age’:39}, so d1[‘name’] = ‘bob’.

 You can combine lists in two ways. list1=[1,2], l2= [3,4], so list3= list1+list2= [1,2,3,4]. Or, say list3= list1.extend(list1).

 If you say list1.append(list2), though, you get list1 = [1,2,[3,4]] because you added the list list2 as a new item in list1.

 If you want to merge use ZIP and LIST: list4 = list (zip(list1,list2)) and get list4 = [[1,2], [3,4]] which is a matrix with two rows (each original list) and two columns (the two columns of each original list).

year = [allrows[i][0] for i in range(len(allrows))] #To extract column 0 from matrix allrows.

Note that you cannot use an append method for a string, because strings are immutable. You need to redefine the string or recreate it, like this: v1 = ‘word’, v2 = ‘name’, v1= v1 + v2 = ‘wordname’ or , v3= v1 + v2 = ‘wordname’.

 #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]

allrows = [ [float(allrows[i][0]),float(allrows[i][1])] for i in range(len(allrows)) ]

#This converts a 2 column list from strings into floats

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

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.

(a0,a1,a2)= symbols('a0,a1,a2') #to add other symbols

eq1= Eq (- (a2 + b - (a1+a2)/2)\*\*2 - c, -(a2+b- a2/2)\*\*2)

eq2 = Eq(a1, a2/2 - 2\*b) #Equates the first argument to the second

answer = solve((eq1, eq2),(a1,a2) )

for item in range(len(answer)): #in range(2), that is

 v3 = answer[item][0].evalf(subs={b:.05 , c:.1})#Evaluate the square roots, subs. in for c and b.

 v4 = answer[item][1].evalf(subs={ b:.05 , c:.1 })

print( 'a1 = %.2f and a2 = %.2f' % (v3, v4) ) %prints it out to two digits

LATEX. #To convert python strings to latex:

from sympy import \*;from sympy.abc import \*;

y = str(x\*\*2/4); z = latex(y);

print(z)

In Ipython, to print out latex stff, do thisL

from IPython.display import display, Math, Latex

display(Math(r'F(k) = \alpha\_1 x'))

 Note that if I define v1= y\*\*2, v1 does NOT have to be predefined as a symbol. Only the RHS variables do.

plot( x\*\*2, (x,0,12),ylim= (0,21),axis\_center=(0,0),

 xlabel="x" , ylabel="$f(x)= x^2$" ) #This is a SYMPY command; no other module needed.

 #DIFFERENTIATION:

p = (x+y)/12; temp= diff (p, x); print(temp)

solveset(j\*\*5-23, j, domain=S.Reals) #TO SOLVE EQUATIONS:

qd= alpha - beta \* p

qs= -gamma + delta \*p

e1 = Eq(qd, qs)

e2 = Eq(q, -gamma + delta \*p )

answer = solve([ e1,e2 ], [p,q ])#Solve, unlike solveset, doesn’t allow domain=S.Reals.

print("p,q = ", answer[p], answer[q] )

#Answers are given like {x: 2\*beta + gamma, y: 3\*gamma} answer[p] returns just the p component.

theta=.2; x=1;h=1

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.expres(x).subs({x:3}) #subsitute 3 for x in expres(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.

z = [x for x in range(5)] # A “comprehension” to get [0,1,2,3,4]

print(z)

m = [ ]

for index1 in range(1,6):

 m.append(index1)

print(m)

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.

RANGE: import numpy; numpy.arange(0,1, .1); #This gives increments of .1 RANGE just does integers.

LOOPING OVER BOTH COLUMN AND ROWS

import random; list1 = [ [random.randint(0,100) for j in range(5)] for i in range(10) ]

nrows= len(list1); ncolumns= len (list1[0])

squaredmatrix = [ [list1[i][j]\*\*2 for j in range(ncolumns)] for i in range(nrows)]

“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...")

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())

 TO EXECUTE A PYTHON SCRIPT FROM WITHIN AONTHER ONE:

exec(open("temp2.py").read())

TO OPEN A COMMAND WINDOW TERMINAL IN MY PYTHON DIRECTORY:

os.system(“start”)

# THESE ARE INPUT AND OUTPUT COMMANDS.

import os #See https://www.pythonforbeginners.com/os/pythons-os-module

os.getcwd()

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

y= os.listdir(x) #os.listdir() #list the current directory

#os.system("dir")

y = str(y)

import re

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

import time

v3= time()

z = os.popen('dir','r') #Piping, create a readable object from the ‘dir’ command in the shell.

z.read() #print out z, which will be the directory listing.

v2=open("temp.txt", ‘w') #or use ‘a’ if you want to append. It will still create temp.txt.

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

v2.write('Hello World Here \n Here is a second line')

#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.

write() is for one string. writelines() is for an iterable, like a list or python OPEN wrapper list of lines.

with open('inputfile') as infile: #To read in a file and write it out efficiently.

 with open('outputfile') as outfile:

 for line in infile:

 outfile.write(line)

v2.write is only for writing strings, not numbers, which must first be converted to strings.

For writing numbers, use the csv module do it like this:

with open('output\_'+datetime.datetime.now().strftime("%y%m%d")+'.csv', "w") as f:

 writer = csv.writer(f)

 writer.writerows(['id', 'price', 'delivery', 'tax', 'segment', 'body', 'transmission'])

data1=[] #I am going to put the data I read in into the list data1, so I start an empty list.

file1 = open ('temp.txt', 'r+') #Read OR write.

for line in file1: #Thisw ill go over each line.

 line=line.rstrip() #Strip off the end of line characters from the strings

 line= line.replace(',', '' ) #Get rid fo the commas.

 try: data1.append( int(line)) # convert the string number to an integer and append to data1

 except: print("This line didn't read data in") #But if it was text, say it didn’t convert.

file1.close()

del(data1[0]) #I don’t want the first line’s number.

for item in range(26): #Now delete any items bigger than 150.

 if data1[item] > 150:

 del data1[item]

v2=open(“temp.txt”,a).readlines()

print(v2[2:8]) #this prints out lines 2 to 8.

with open("temp.txt") as v2: #Here is osmething to use a while loop to print out one line at a time.

 line= v2.readline()

 counter=1

 while line:

 print(line)

 line= v2.readline() #Why is this step needed?

 counter +=1

with open("temp.txt") as fileobject: #The WITH opens temp.txt and does the stuff in the indent

#and then closes temp.txt again. It prints lines one at a time with ‘New line’ in between.

 for line in fileobject:

 print(line.rstrip()) #rstrip takes off the blank spaces

 print('New line')

#This next script both reads nad writes. It reads temp.txt and then appends a repeat of

#its text but with line numbers and colons.

fobj\_in = open("temp.txt")

fobj\_out = open("temp.txt","a")

i = 1

for line in fobj\_in:

 print(line.rstrip())

 fobj\_out.write(str(i) + ": " + line)

 i = i + 1

fobj\_in.close()

fobj\_out.close()

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

import webbrowser

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

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

#In a terminal, type “python temp1.py 402 S. Coler, Urbana, Illinois” and save the script below in temp1.py.

import sys

address1 = sys.argv[1:] #This creates the list ['402', 'S.', 'Coler,', 'Urbana,', 'Illinois']

address2 = ' '.join(sys.argv[1:]) #This creates the string “402 S. Coler, Urbana, Illinois”

print(address1, address2)

#The following script works in Spyder to get stuff from teh Clipboard. (or the command line)

import webbrowser, sys**,** **tkinter**

if len(sys.argv) > 1:
    # Get address from command line.
     address = ' '.join(sys.argv[1:])
**else:**
**# Get address from clipboard.**
 **root = tk.Tk()**

 **address = root.clipboard\_get()**

**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.

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

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")

 DOWNLOADING A WEB PAGE AND WRITING TO TEMP.TXT

import requests

myfile= requests.get('http://www.rasmusen.org/news.htm')

print(myfile.text[:250])

v2 = open("temp.txt",'w') #Use ‘a’ for append instead of writing over.

v2.write(myfile.text)

v2.close()

with open("temp.txt") as v2: #This will read it line by line

 for line in v2:

 print(line)

with open('temp1.txt','w') as v6: v6.write(str(t1.attrs)) #TO WRITE TO A FILE AND CLOSE AUTOMATICALLY..

<https://www.programcreek.com/python/example/6251/requests.post> EXMAPLES OF POST

#The METHOD BELOW USES THE WIKIPEDIA PHP SEARCH FORM TO SEARCH FOR A NANOTECHNOLOGY ARTICLE

import requests

req = requests.post('https://en.wikipedia.org/w/index.php', data = {'search':'Nanotechnology'})

with open('temp.htm', 'wb') as fd:

 for chunk in req.iter\_content(chunk\_size=50000):

 fd.write(chunk)

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

#Name: gettingwebfiles.py

#JUly 14 2017

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.

import requests; import webbrowser;

from bs4 import BeautifulSoup as bs4

v1 = 'http://rasmusen.org/temp.htm'

v2= requests.get(v1)

#print(v2.content) #This prints the page v1 that we found.

v3= bs4(v2.content, 'html.parser')

v4= str(v3.prettify);print(v4); #This prints out the web page nicely formatted.

v5=open('temp.htm','w'); v5.write(v4); v5.close();#writes the web page locally.

v6=open('temp.txt','w'); v6.write(v3.get\_text()); v6.close();#writes the web page text locally.

for link in v3.find\_all('a'): #Get all the links in the v3 page, <a>...</a>

 print(link.get('href'))

print (v3.title) #prints 'Ford Motor Co.' or whatever.

print (v3.title.name) #Prints 'title', the HTML tag.

var2= open('example.html')

elems = var4.select('#author')

print(type(elems))

print(str(elems[0]))

print(elems[0].attrs)

#It might be good to use re: to find the info needed.

from bs4 import BeautifulSoup

import requests

v1=requests.get("http://pages.stern.nyu.edu/~adamodar/New\_Home\_Page/datafile/histimpl.html")

v2= BeautifulSoup(v1.content, 'lxml')

v3 = v2.find\_all('tr') #This creates a list of all the <tr> tags found.Rows.

v4 =[] #initialize v4 as an empty list.

for row in v3: #Taht is, for each row

 v5 = row.find\_all('td') #make a list of td items

 v6=[] #intialize an empty list

 for item in v5:

 v6.append(item.get\_text()) #append the item content to list v6

 v4.append(v6) #Now append row list v6 to list of rows v4.

for i in range(len(v4)): #Now get rid of the percentage signs

 #print('i[1] is ', i[1])

 for j in range(len(v4[i])): #It works to use the indices as iterators rather than the list items

 #print('j is',j)

 v4[i][j]=v4[i][j].replace('%', '')

 #print('j is',j)

for i in range(len(v4)): #Now convert the string numbers to float numbers

 #print('i[1] is ', i[1])

 for j in range(len(v4[i])):

 try: #Not every item will convert to float, so use TRY and EXCEPT

 v4[i][j] = float(v4[i][j])

 except:

 print("can't convert") #PASS would work too, to give up on converting some items.

Before using bs, it can b good to convert the file to a string and to look at it. find out if it needs decoding from UTF-8 bytes or something, using

v1=requests.get("http://pages.stern.nyu.edu/~adamodar/New\_Home\_Page/datafile/histimpl.html")

v2 = v1.decode('UTF-8') #This makes it into a string

v3 = v2.split(‘\n’) make into a list splitting on linebreaks.

print(v2[:200]) #read the first 200 characters of the v2 string.

v2.split('\n')[:5] #In the console, to get the first 5 lines of v2 (made into a list by split)

number1 = len(page\_soup.find\_all('table', attrs={'border': '1'})) #number of tables with border size 1 in page\_soup

table = page\_soup.find\_all('table', attrs={'border': '1'})[0] #extract all the tables with border size 1 in page\_soup.

data\_row = table.find\_all('tr')[1] # fextract the first row of “table”.

data\_cols = data\_row.find\_all('td') # List of columns in the first row of data\_row.

for column in data\_cols:

 print (column.text) # look at the text in each column

all\_rows = table.find\_all('tr')[1:]

store\_data = []

for row in all\_rows:

 row\_cols = row.find\_all('td') # For each row, get values in each column

 r = [col.text for col in row\_cols]

 store\_data.append(r)

output\_columns = ['row', 'date', 'day', 'day\_no']

output = pd.DataFrame(store\_data, columns = output\_columns) #This yields a nice table in panda

from bs4 import BeautifulSoup

v1 = open('temp.htm') #I downloaded the html file here first.

v3= BeautifulSoup(v1, 'lxml') #lxml is an html parser.

v9= str(v3.prettify);print(v9); #This prints out the web page nicely formatted.

print(v3.contents) #Another way to print the entire file.

v4 = v3.td #This makes the first instance of the <td>... ,</td> into v4.

print(v4) #Print v4.

print(v4.attrs) #print the attributes of the <TD> tag, e.g. 'height': 20'

for item in v4.attrs: print(item) #print just 'height', 'width' w/o values

with open('temp1.txt','w') as v6: v6.write(str(t1.attrs)) #TO WRITE TO A FILE AND CLOSE AUTOMATICALLY..

del v4.attrs #remove all the attributes, levinga clean <td> tag.

v8 = v3.find\_all('td') #This creates a list of all the <td> tags found.

for item in v8:

 print(item.get\_text()) #This prints out the contents of each item of the list

v1.close()

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

#googlesearch3.py

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

#July 19, 2017

import requests;import webbrowser;

from bs4 import BeautifulSoup as bs4

v1 = 'North Korea ballistic missles' # var1 is what we are searching for on Google.

v2=requests.get("https://www.google.com/search?q="+v1) #This gets the web page with the search results for #var1.

v5= bs4(v2.content, 'html.parser') #Make a beaut.soup.object.

#v3= webbrowser.open("https://www.google.com/search?q="+v1)#This opens a browswer to show the search results.

v4 = open('temp.htm', 'w'); v4.write(str(v5));v4.close();#write the page to a text file.

v6 = v5.find\_all("h3", class\_="r")#makea list of the h3 parts

print(v6[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.

print(v6[0].a["href"]) # This pulls out just the href=jkljkljkl part of the first item,

webbrowser.open("www.google.com" + v6[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): #This prints the first 3 hrefs, after putting www.google.com in front to

 print("www.google.com" + v6[item].a["href"] ) #complete the address.

print(v6[0].a["href"])

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

 # and makes it into a string.

#This looks good: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

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="+v1 +"&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.

<https://automatetheboringstuff.com/2e/chapter12/> on requests, beautiful soup, opening webpages.

TO SAVE A SCREENSHOT

import pyautogui #This can also move the cursor and make it click at physical locations on the screen.

im1 = pyautogui.screenshot()

im1.save('my\_screenshot.png')

pip install selenium; from selenium import webdriver #For clicking on buttons.

#<https://automatetheboringstuff.com/chapter11/>

#To click on the first two buttons on a web page . This works.

from selenium import webdriver

browser= webdriver.Chrome()

browser.get('https://www.w3schools.com/tags/tag\_button.asp')

linkElem = browser.find\_elements\_by\_partial\_link\_text("Button")

print(linkElem)

linkElem[0].click()

linkElem[1].click()

#this didn’t work, but it’s to fill in a form. The hard part is to find the relevant element.

from selenium import webdriver

browser= webdriver.Chrome()

browser.get('https://marginalrevolution.com')

text\_area = browser.find\_element\_by\_id('textarea')

text\_area.send\_keys("This text is send using Python code.")

REGULAR EXPRESSIONS: the re module

import re

#"rai?n" matches"ran" and "rain". i is before the ? 1 or 0 times This is an OPTIONAL quantifier.

# "be+" matches "been" and "bent". e is before the + 1 or more times.

# "be\*" matches "been", "bent" . matches the e 0 or more times. OPTIONAL also.

 # “e{ n }t” matches e exactly n times and then there must be a t.

# “e{ n, }t” matches e exactly n or more times and then there must be a t.

#{0,1} is the same as ?, {0,} is the same as \*, and {1,} is the same as +

\w an alphanumeric character [A-Za-z0-9\_]

\W a character NOT alphanumeric [^A-Za-z0-9\_]

\s a whitespace charactre [ \t\r\n\f\v],

\S Any charcter not a white space

 $ match must be at the end of the string or line.

^ match must be at the beginning of the string or line. The carat has two separate meanings, this and NOT.

[^z] the next character cannot be a z

\d a number character

[a-z] any letter, lowercase.

 [A-Za-z] any letter, any case.

https://www.analyticsvidhya.com/blog/2015/06/regular-expression-python/

# [https://docs.microsoft.com/en-us/dotnet/standard/base-types/regular-expression-language-quick-#reference#character-classes](https://docs.microsoft.com/en-us/dotnet/standard/base-types/regular-expression-language-quick-#reference)

r’string1’ means use a python raw string string1. It will treat control characters like \n as ordinary characters.

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

 and they are they \nwhich testify of me.”

#The \n splits the line. The \ just says it is written over more than one line.

v1= re.findall('th\w\*', john5) ; print(v1)#Thi searches for words starting with ‘th’.

# v1 is the list ['the', 'them', 'think', 'they', 'they']

# The \* means it looks for zero or more of \w, which means alphanumerics., coming after ‘th’.

v1=set(re.findall('th\w\*', john5)) ; print(v1)#This returns the SET, without duplicates, not a list.

v1= re.findall('th\*', john5) # ['th', 't', 'th', 'th', 't', 'th', 'th', 't', 't'] # even 0 h’s is a match so long as there is a t.

#Python makes unusual use of \*. It doesn’t mean the usual “anything”. It means “any number of h’s”

re.findall('th.y', john5) #This finds they, thay, thoy.

re.findall('the.', john5)#This finds ['the ', ‘them’, 'they', 'they']

 re.findall('the[^m]', john5) #This finds ['the ', 'they', 'they']

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

 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.

 re.findall(' [\s][a-z]{5}[\s] ', john5) # [' think ', ' which '] all the 5 letter words with a space before and after

 re.findall('[^a-z][a-z]{5}[^a-z]', john5) # {' think ', ' which ', 'Search '}

The regex *I love (baboons|gorillas)* will match the text *I love* and then match either *baboons* or *gorillas*

<https://www.rexegg.com/>

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

var2 = len(var1)

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

x=re.search('theeee', 'abcd6abcde') #There is no match,s o this is None.

re.search('eternal', john5) # re.search creates a special object, not a string.

re.search('the.', john5)#This looks for the first match for the, they, them and then stops.

re.split('ye', john5)

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

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")

re.search('abc', 'abcd6abcde')#This says where there is a match for “abc”

re.findall('.\*e', 'Here are some words') # : ['Here are some']

 The longest possible match of any number of any character up to ‘e’.

 re.findall('h(', 'here hear hard words') # : ['Here are some']

h(e|is|at)

re.findall('H\*e', 'Here are some words') # ['He', 'e', 'e', 'e']

 The various patterns of zero or more H’s followed by an ‘e’.

re.findall('e+', 'Here are some words') # ['He', 'e', 'e', 'e']

re.findall('\w+', 'Here are some words') # ['Here', 'are', 'some', 'words']

 # one or more alphanumerics, SO WE GET EACH WORD, ending with nonalphanumberic white space.

print(re.findall('st(ring)\*', 'Here is a string seen.')) #’ring”. Returns just what is in ().

print(re.findall('(s)\*tring', 'Here is a string seen.'))# ‘s’
print(re.findall('s\*tring', 'Here is a string seen.')) # ’string’

re.findall('\w\*e\w\*', 'Here are some words e ') #['Here', 'are', 'some', 'e']

#0 or more alphanumerics, an ‘e’, zero o rmore alphanumerics ---every word with “e” in it.

re.findall('b\w+n', 'bean, been, bin, sdfdf ') #returns ['bean', 'been', 'bin']. b, 1 or more alphanumerics, n.

re.findall('be+n', 'bean, been, bin, sdfdf ') #returns [ 'been']. b, 1 or more e’s (but nothing else), n.

re.findall('be\*n', 'bean, been, bin ,bn ') #returns ['been', 'bn']. b, 0 or more e’s (but nothing else), n.

v1=re.findall('(?i)value of', 'Value Of it') #returns ‘Value Of’. (?i) is flag to ignore case. \\\\\\\\\\\\\\\\\\\\\\

re.findall( '\d{2}' , 'March 07, April 08') #yields ['07', '08']. Return any patterns of two digits together

 .

re.findall('a\*c', 'abc d6ab acde') # ['c', 'ac'] 0 or more ‘a’ followed by a ‘c’, nothing in between.

re.findall('a.c', 'abc d6 abc de') #['abc', 'abc '] there must be exactly one letter in between a and c.

set(re.findall('th?', 'that')) # {'th'}

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

j

'string'.join(['a','b','c']) #'astringbstringc' for joining a list together with a string in between.

v0='example string one'; v1 = re.sub('n', 'NNN', v0); #result: 'example striNNNg oNNNe' REPLACE

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

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

PLOTS, DIAGRAMS, FIGURES

#sympy is often better for plotting numerical examples, functions. It uses fewer commands.

import sympy as sympy

simpy.plot( x\*\*2, (x,0,12),ylim= (0,21),axis\_center=(0,0),

 xlabel="x" , ylabel="$f(x)= x^2$" ) #This is a SYMPY command; no other module needed.

#With sympy, we don't need plt.show() or plt.close().

import matplotlib.pyplot as plt #Here is a very simple plot.

x =[item for item in range(30)]; y = [item\*\*2 for item in x] #The two lists to use in the plot

plt.plot(x,y) #Doesn’t need show, or axes, or subplots, or fig.

plt.plot(x, x\*\*2, marker='.', markersize=3 ) #For dots of size 3 as markers.

plt.show() and plt.clf() both clear the current figure, plt.show() after first showing it.

plt.axhline(linewidth=6, color="black") #FOR THICKNESS OF AXES.

plt.axvline(linewidth=6, color="black")

plt.ylabel("The\npredicted\nvalue", rotation= 'horizontal', horizontalalignment='left' ,labelpad=46)

#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)

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

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

x = np.linspace(0, 10, 200) #These are the x values, 200 of them

y = np.sin(x)

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

plt.xticks([]) #Eliminate x axis ticks and tick markers

**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”, dpi= 1200)

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

plt.close("all")

from matplotlib import pyplot as plt

plt.figure(figsize=(1,1))

x = [1,2,3]

plt.plot(x, x)

plt.show()

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

#A SIMPLE PLOT

import matplotlib.pyplot as plt

import numpy as np

plt.rc('text', usetex=True);plt.rc('font', family='serif')#For Latex fonts.

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)

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

#Gamma is a list of 200 points between 0 and 20.

y = 1.5\*gamma + sigma\*gamma - 2 \* alpha + (3+gamma)\*c

plot1=plt.plot(gamma,y, linewidth=4, color='black')

plt.scatter(x, y) # for a scatterplot of dots of x and y.

 #Though plain old plt.plot will do the same with linestyle=' '.

SOME OPTIONS

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

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

plt.xticks(np.arange(0, max(Q)+1, 4.0), fontsize = 24)

plt.yticks(np.arange(0, max(Ps)+1, 2.0), fontsize = 24)

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

plt.xlim(0, 9)

plt.ylim(0, ) #this lets it choose the upper y limit.

 # 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)

plt.rcParams['lines.linewidth'] = 3 #Default width of plot lines.

plt.rcParams['lines.color'] = 'black' #This is for straight lines.

plt.rcParams['axes.color\_cycle']='black' #For plot lines, not axes.

plt.rcParams['axes.linewidth'] = 5 # set the value globally

linestyle='dashed' OR 'dashdot' OR 'dotted' OR “ “ which is NONE.

#The annotate method inserts an arrow and a phrase.

#It is the quickest way to get a label to look good, e.g. Demand

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\_annotate:

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

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

#Here is another way to insert an arrow, a straight one:

plt.annotate(' ', xy=(4, 6), xytext=(5.5,6),

 arrowprops={'arrowstyle': '->', 'lw': 4, 'color': 'black'},

 va='center')

 )

#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 (0,7) and (8,7) with a line:

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

plt.plot([0, 30],[ 0,40], linewidth=1, color='black') #diagonal

#To fill in a blue .5-transparency triangle of (0,36), (0,24), and (12,24):

plt.fill( [0,0,12],[36,24,24], "b", alpha = .5) #for polygons generally.

 #fill\_between() for filling the area between two curves.

 #THe patches command is a way to generate polgyon objects.

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

plt.plot([0,0,0],[8,16, 24], linestyle=' ', marker= 'o', color='k' markersize =16, clip\_on=False) #Think of the two lists as plotting x against y.

 #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("temp.png")

HISTOGRAM of VLIST, A LIST OF NUMBERS

import random; randomlist = [random.randint(0,100) for i in range(1000)]

plt.hist(randomlist, color = 'blue',

 edgecolor = 'black', bins =10)

plt.xlabel('Proposer error v')

plt.show()

CLASSES

class tracer: #This is the decorator function of functoin1

 def \_\_init\_\_(self, function1): #says what the decorator applies to

 self.calls=0 #self is an argument of all classes, to hold info about the object

 self.function1= function1

 def \_\_call\_\_(self, \*args): #We do not need to say how many arguments

 self.calls=self.calls+1

 print('call %s to the functoin %s' % (self.calls, self.function1.\_\_name\_\_))

 return self.function1(\*args)

@tracer #This says that the decorator applies to the function below.

def spam(a,b,c):

 print(a+b+c)

@tracer #We can re-use Tracer on a differnet function

def spam2(a,b):

 print(a, b)

spam( 3,4,10) #Now run the functoin and see what happens.

spam2(4,5) #Now run the functoin and see what happens.

def f2(size): #Here’s another function

 print(size\*'Hello')

tracer(f2(6))# We can also apply tracer directly to a function

class a: #Here is an exmaple of creating a class module

 v0 = 1000 # A class variable.

 def \_\_init\_\_(self, x ) : #x is the arugment, self is also needed.

 v1=100 #An inseance variable.

 self.x = x

 self.plus = v1 + x

v2=a(5 ) #Now we have created an "instance" of the class a object.

print(v2.x, v2.plus)#This prints 5, 105.

print(v2.v0) #v0 is a class variablee--- it just returns 1000.

print(a.\_\_dict\_\_) #This will show general attributes of the class.

print(v2.\_\_dict\_\_) #This will show attributes of an instance of the class.

class Computer(): #computer is a class.

 def \_\_init\_\_(self, computer, ram, ssd):

 self.computer = computer; self.ram = ram; self.ssd = ssd

class Laptop(Computer):

 def \_\_init\_\_(self, computer, ram, ssd, model):

 super().\_\_init\_\_(computer, ram, ssd) #This method inserts the methods of the superclass

 self.model = model

object1= Laptop(‘Lenovo’, 512, 1380, ‘t1’))

print(object1.model) #This returns ‘t1’.

GRAPHIC USER INTERFACE IN PYTHON FOR WINDOWS (But I use Spyder now)

 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 and 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.

<https://automeris.io/WebPlotDigitizer/>

This is not Python, but it is a neat free web digitizer app to extract the data from a graph and put it in a CSV file.

import csv #READING A ND WRITING DATA

v1= [[ i, i\*\*2 ] for i in range(40)]

v2 = open("temp1.csv", "w") #opens the file to write on and puts it v2.

v2 = open("temp1.csv", "w", newline='') #opens the file to write on and puts it v2. NEWLINE important.

v2.writerows(v1) #Actually writes in the data v1 into v2.

v2.close() #Closes up the temp1.csv file and releases it for other programs to use.

mystring = ‘line one. \n line two.’ #backslash n gives a line break, a new line.

v2 = open("temp1.csv", "r", newline='') #opens the file to read and puts it v2.

v3 = csv.reader(v2) #Makes a csv object out of v2.

v4=[] #starts an empty python list for putting the data in

for row in v3:

 v4.append(row) #for each row in temp1.csv, put it into the v4 list

v2.close() #Closes up the temp1.csv file and releases it for other programs to use.

<http://pages.stern.nyu.edu/~adamodar/New_Home_Page/datafile/histimpl.html>

A RANDOM NUMBER GENERATOR # random.randint(-1,1000) , a random number between 0 and 1000.

import random

outcomes = { 'heads':0, 'tails':0,}

sides = ['heads', 'tails']

for i in range(10):

 x= random.choice(sides)

 print('On toss {toss} we got {result}'.format(toss=i+1, result=x ))

 outcomes[x]= outcomes[ x ] + 1

print( 'Heads:', outcomes['heads'])

print( 'Tails:', outcomes['tails'])

randomlist = [random.randint(0,100) for i in range(7)] #For making a list of 7 random numbers.

randomlist2 = [random.gauss(100, 15) for i in range(100000)] #normal dist.

random.choice(sample)#pick a random element of the list ‘sample’.

PUTTING IN TIME DELAYS

from time import sleep

for item in range(21): print (item\*\*2); sleep(1) #Inserts 1 second between each iteration.,

import time #FOR TIMING HOW LONG SOMETHING TAKES

t0 = time.process\_time()

for item in range(200000):

 x= item\*\*5 + 49

elapsed\_time = time.process\_time() - t0

print( elapsed\_time, "seconds process time")

#CALLBACK FUNCTION:

# You can write a function to let you know when a long program has gotten to various milestones.

import os; os.chdir('c:\\programdata\\anaconda3\\lib\\site-packages\\'); import fitz;

doc = fitz.open("temp.pdf") # THIS CODE OPENS A PDF FILE AND INSERTS A TEXTBOX.

page = doc[0] # choose some page

rect = fitz.Rect(50, 100, 300, 400) # rectangle (left, top, right, bottom) in pixels

text = """This text will only appear in the rectangle. Depending on width, new lines are generated as required.\n<- This forced line break will also appear.\tNow a very long word: abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ.\nIt will be broken into pieces."""

page.drawRect(rect, color = (1,1,1), fill = (1,1,1)) # white is the tuple (1, 1, 1)# This prevents transparency.

rc = page.insertTextbox(rect, text, fontsize = 12, # choose fontsize (float)

 fontname = "Times-Roman", # a PDF standard font

 fontfile = None, # could be a file on your system

 align = 0) # 0 = left, 1 = center, 2 = right

print("unused rectangle height: %g" % rc) # just demo (should display "44.2")

doc.saveIncr() # update file. Save to new instead by doc.save("new.pdf",...)

import fitz #GETTING LINKS FROM A PIDF FILE

doc = fitz.open("pymupdf.pdf")

page = doc[6] # page 6 contains 4 links

lnks = page.getLinks()

<https://pymupdf.readthedocs.io/en/latest/faq/> #fiz is ery welldoumented. . with a list of things you an do.

<https://scikit-learn.org/stable/tutorial/basic/tutorial.html> Machine learning NUMPY is used heavily in it both techniques like regression and data shaping.

For formatting floats globally (or truncating digits)

def prettyfloat(x):

 return float("%0.2f" % x)

mylist1=list(map(prettyfloat, mylist))

 TO GENERATE A LIST OF VARIABLES (as opposed to a list of variable names)

v1=[ eval( 'total\_loss'+ str(item) ) for item in [1,101]]

print(eval)

<https://stackoverflow.com/questions/1373164/how-do-i-create-a-variable-number-of-variables>

# to get Euler’s E constant and natural logarithms and square roots, import math:

from math import exp, log, sqrt

v1 = exp(20)

v2=log(20)

squareroot(20)= sqrt(20)

#Note that abs(-20) is NOT a math import--- it is standard.