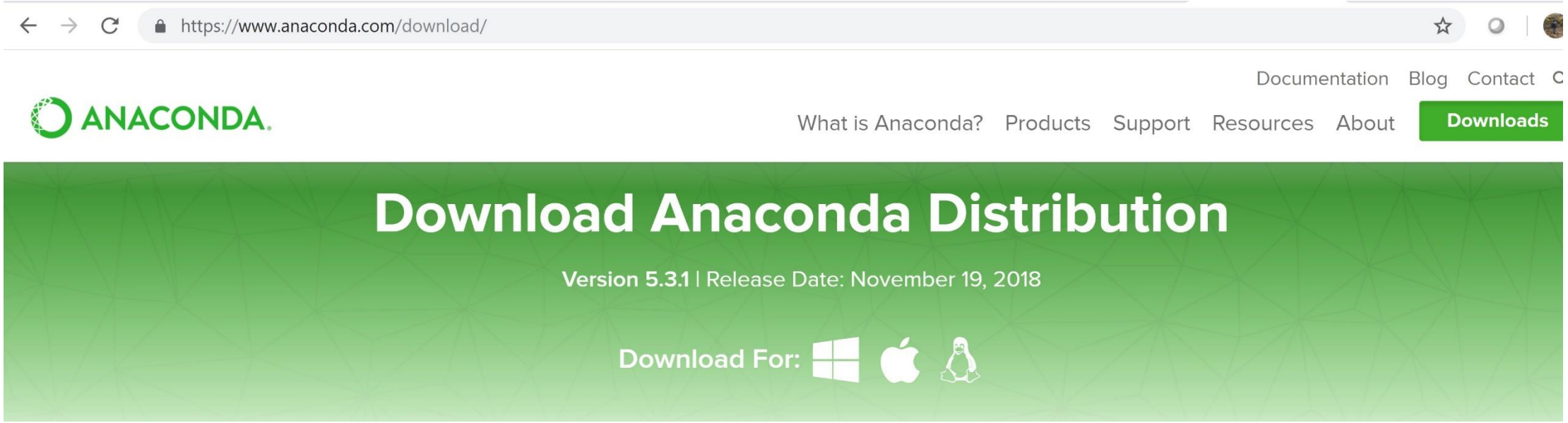
Python and Text Mining

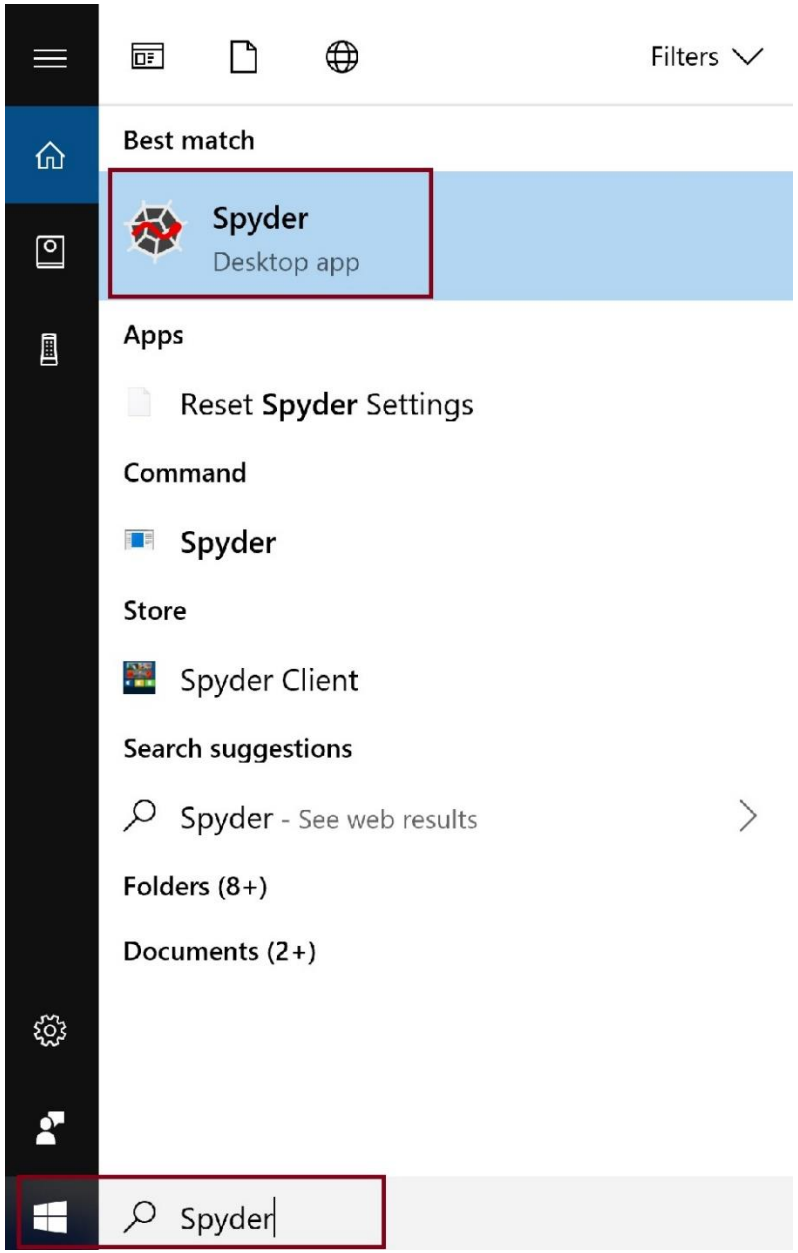
Gates

Get Python3 Via Anaconda

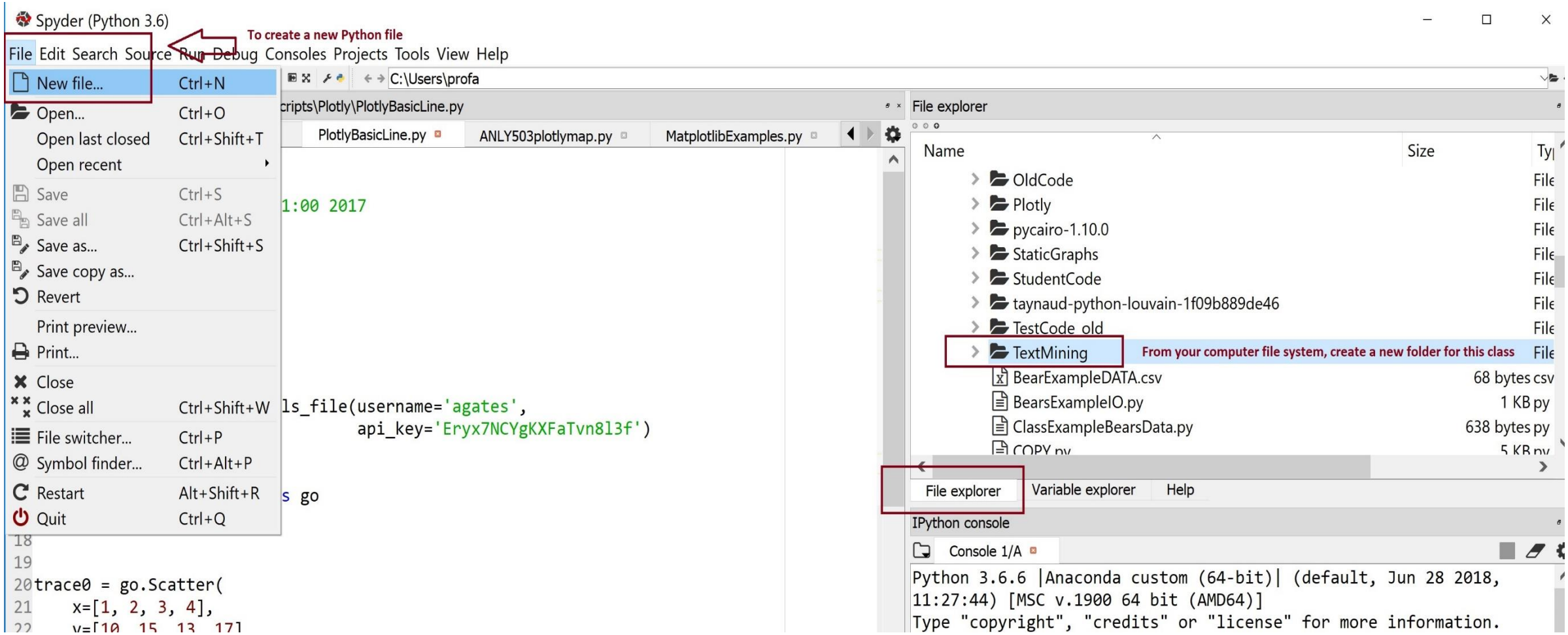
www.anaconda.com/download



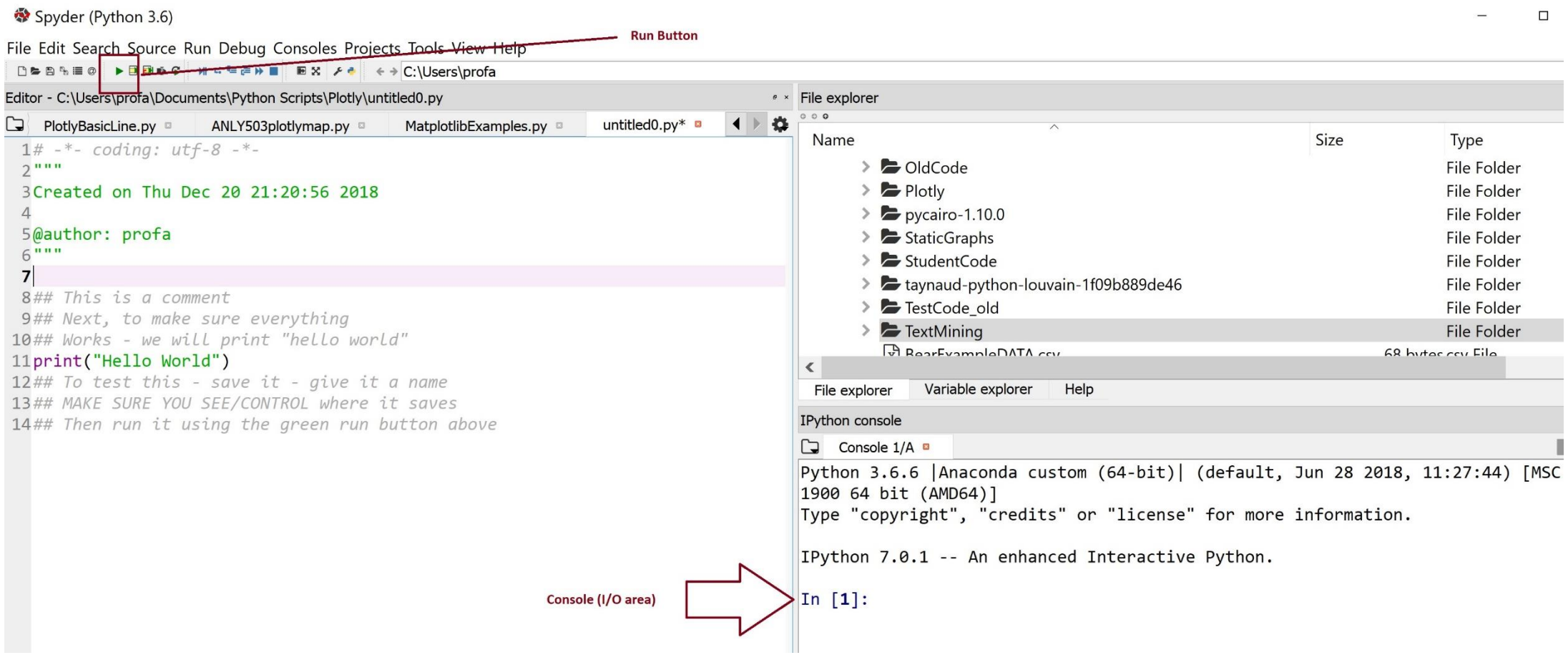
Open Python in Spyder



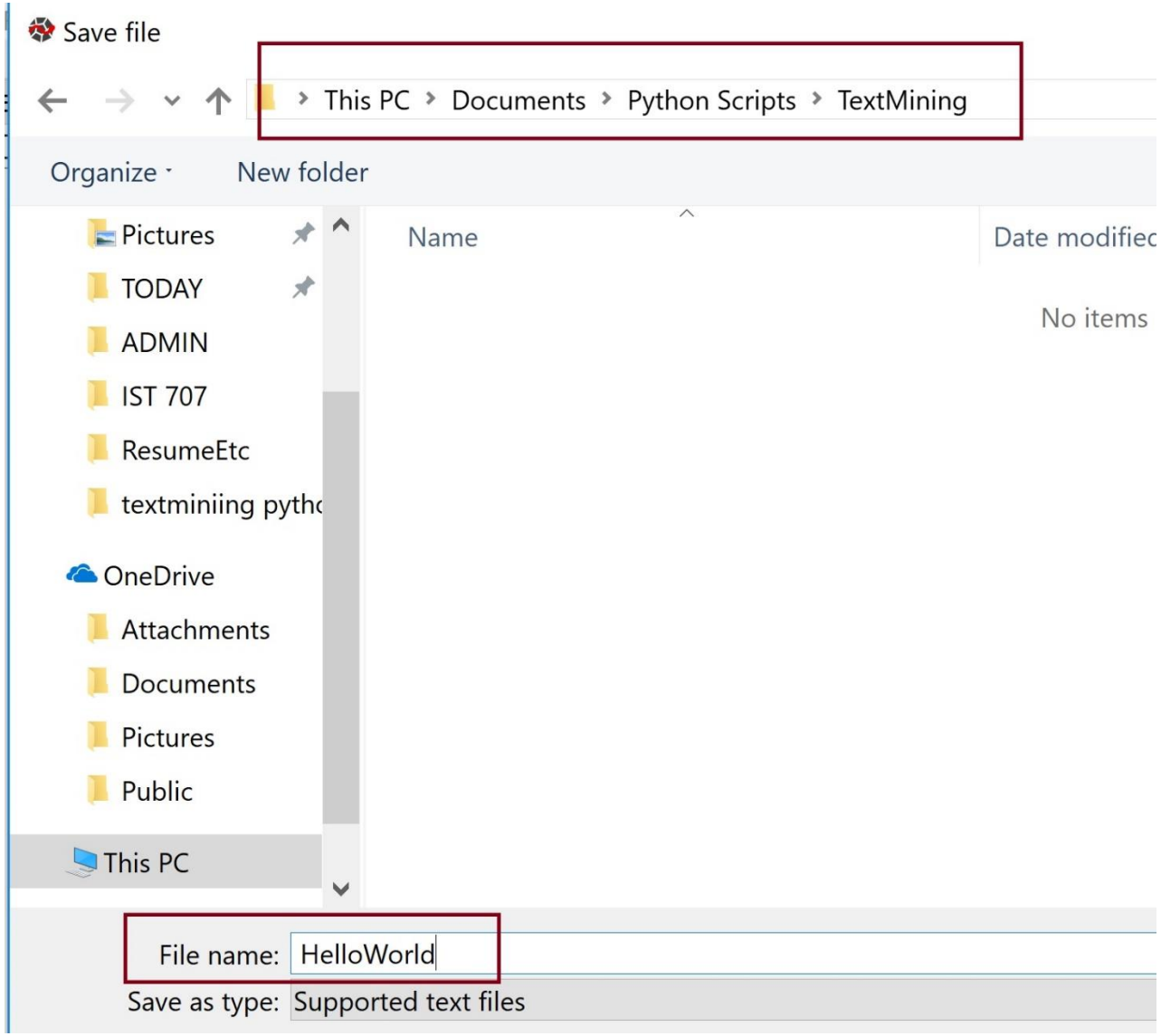
Inside Spyder Part 1



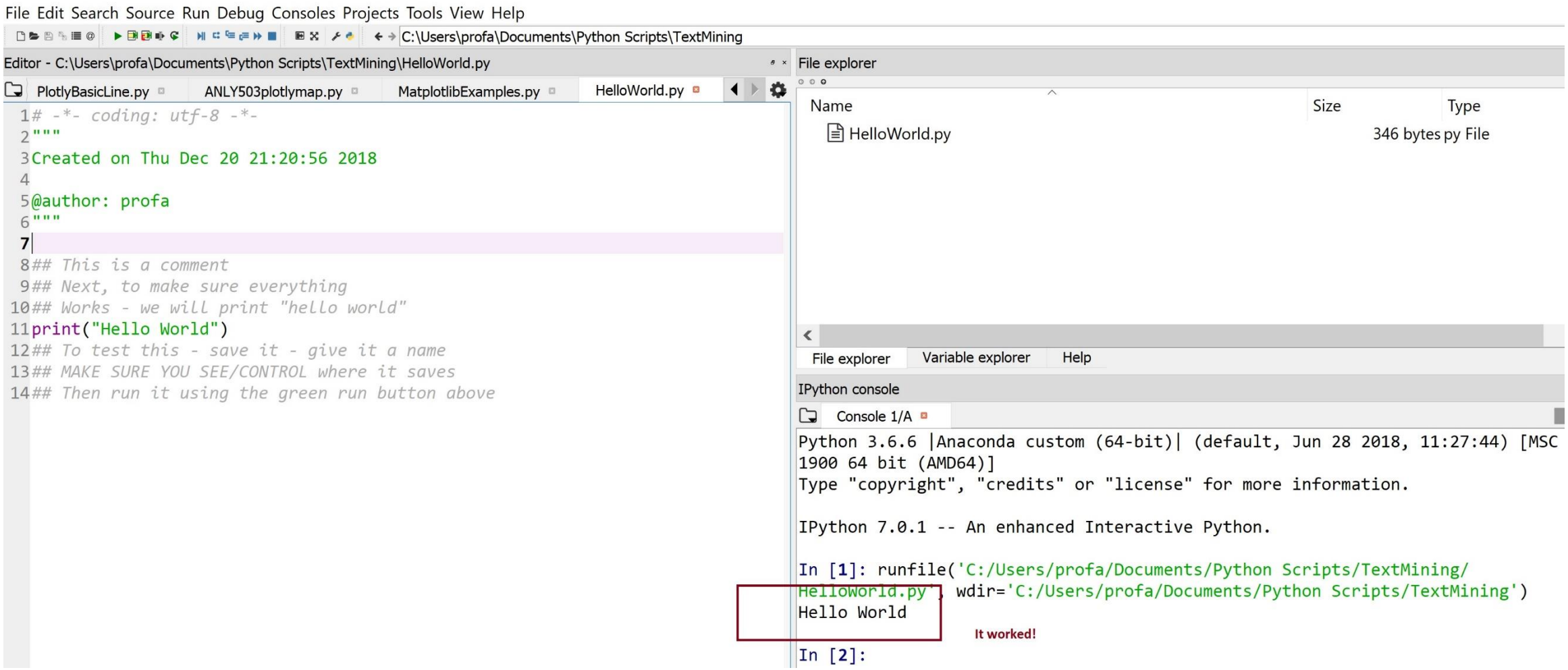
Save as – Name it - and Run It



Save and Name File - ** Make sure you save it to the right location.

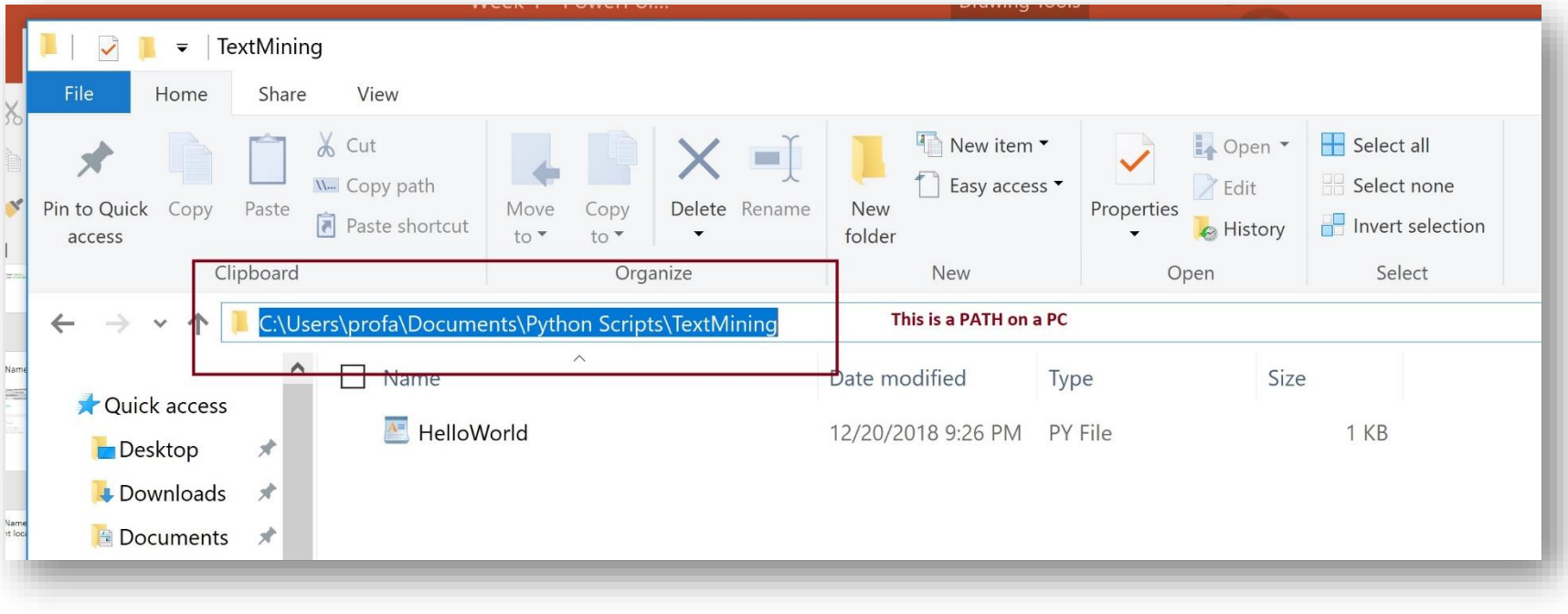


Success!

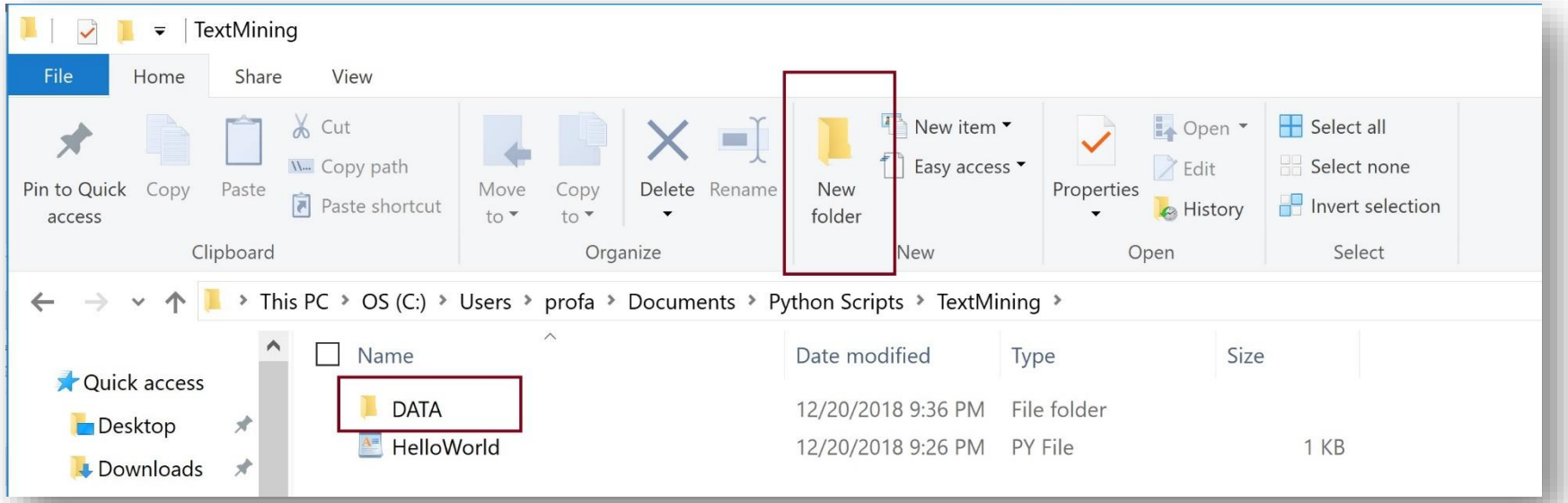

File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\profa\Documents\Python Scripts\TextMining
Editor - C:\Users\profa\Documents\Python Scripts\TextMining\HelloWorld.py
PlotlyBasicLine.py
ANLY503plotlymap.py
MatplotlibExamples.py
HelloWorld.py
1# -*- coding: utf-8 -*-
2"""
3Created on Thu Dec 20 21:20:56 2018
4
5@author: profa
6"""
7
8## This is a comment
9## Next, to make sure everything
10## Works - we will print "hello world"
11print("Hello World")
12## To test this - save it - give it a name
13## MAKE SURE YOU SEE/CONTROL where it saves
14## Then run it using the green run button above
File explorer
Name
Size
Type
HelloWorld.py
346 bytes
py File
File explorer
Variable explorer
Help
IPython console
Console 1/A
Python 3.6.6 |Anaconda custom (64-bit)| (default, Jun 28 2018, 11:27:44) [MSC
1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.
IPython 7.0.1 -- An enhanced Interactive Python.
In [1]: runfile('C:/Users/profa/Documents/Python Scripts/TextMining/
HelloWorld.py', wdir='C:/Users/profa/Documents/Python Scripts/TextMining')
Hello World
It worked!
In [2]:

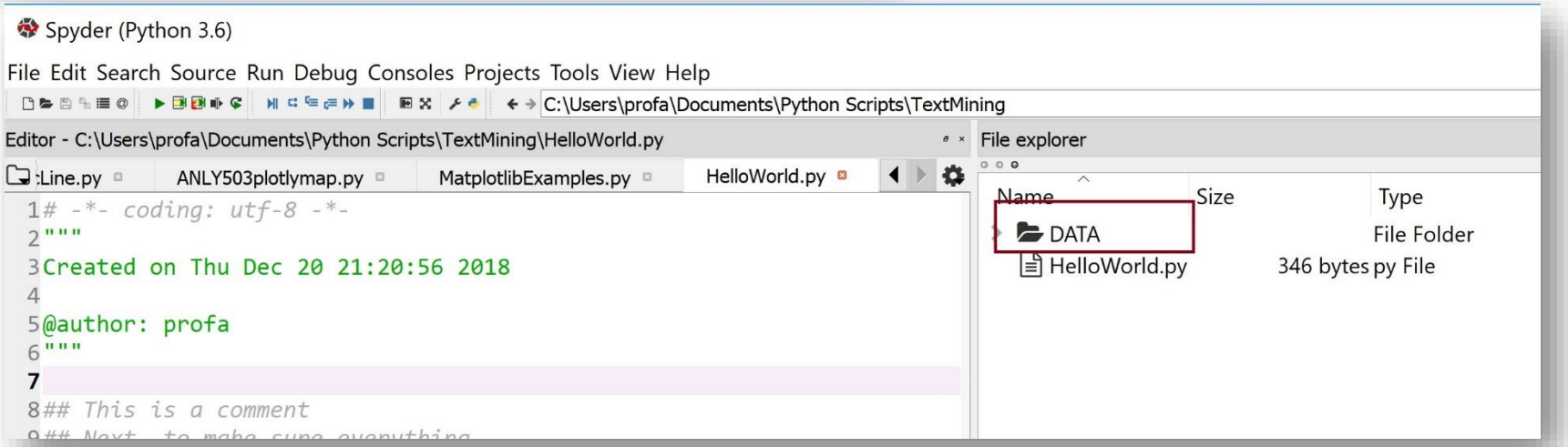
Paths and Files

1) What is a path?
2) Where are your Python files?
3) How can you read in data from a file that is “local”?
4) How can you read in data from a file that is on a path?



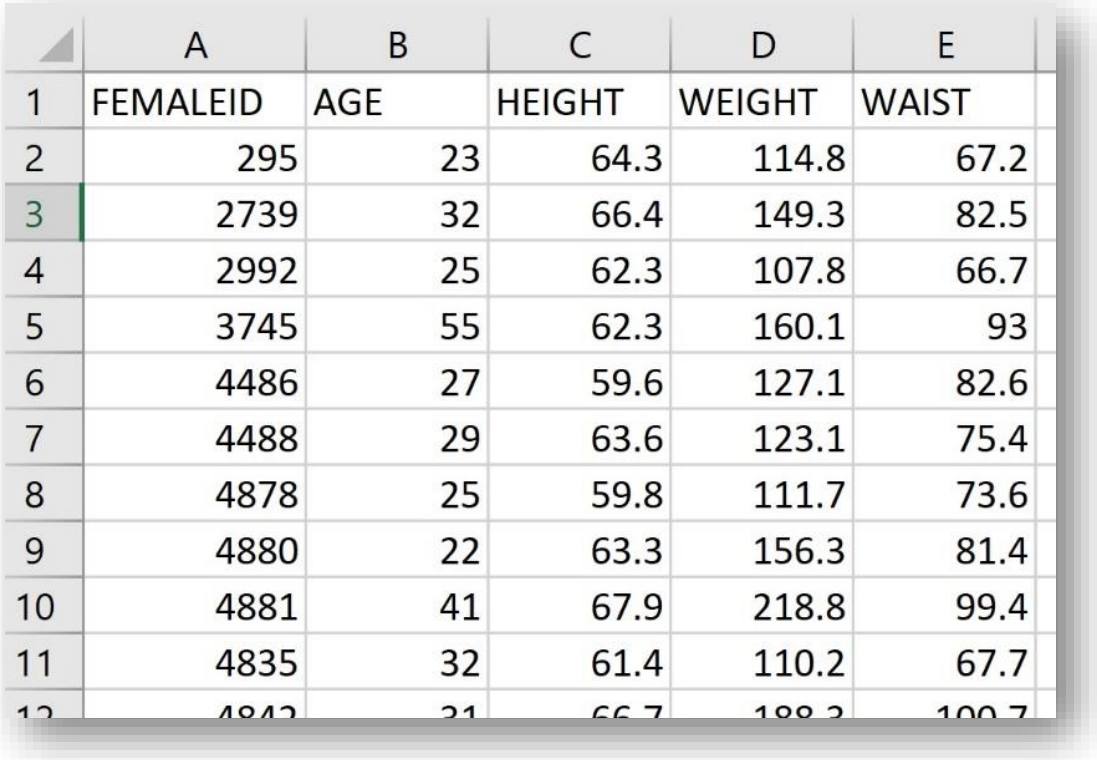
Paths, Files, Folders, and Data



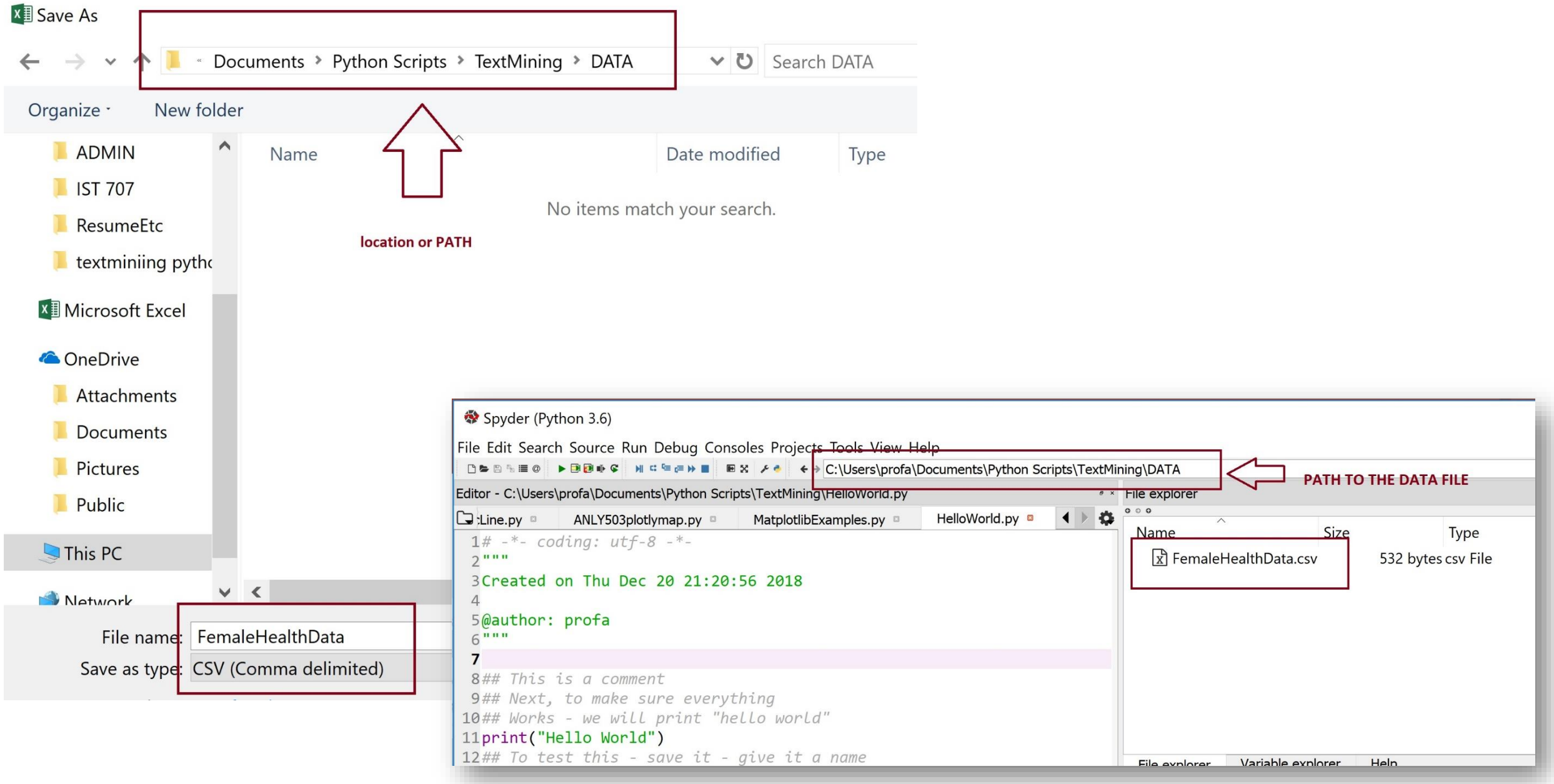


Create a Mini Data file

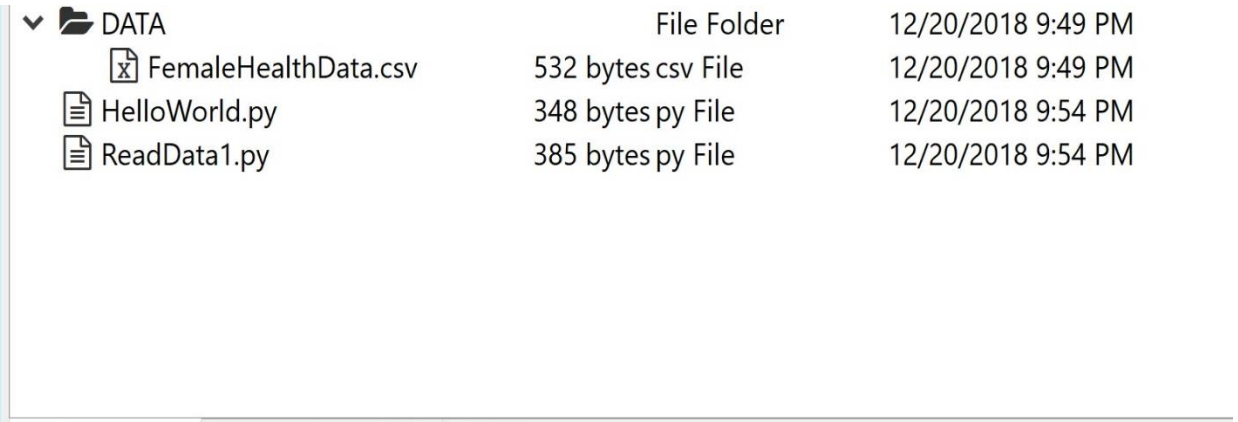
1) Use Excel to build a min datafile.
2) Save it as .csv
3) Put it in DATA
4) Open and print it using Spyder



	A	B	C	D	E
1	FEMALEID	AGE	HEIGHT	WEIGHT	WAIST
2	295	23	64.3	114.8	67.2
3	2739	32	66.4	149.3	82.5
4	2992	25	62.3	107.8	66.7
5	3745	55	62.3	160.1	93
6	4486	27	59.6	127.1	82.6
7	4488	29	63.6	123.1	75.4
8	4878	25	59.8	111.7	73.6
9	4880	22	63.3	156.3	81.4
10	4881	41	67.9	218.8	99.4
11	4835	32	61.4	110.2	67.7
12	4842	31	66.7	188.3	109.7


Save As
Documents › Python Scripts › TextMining › DATA
Search DATA
Organize
New folder
ADMIN
IST 707
ResumeEtc
textminiing pythc
Microsoft Excel
OneDrive
Attachments
Documents
Pictures
Public
This PC
Network
Name
Date modified
Type
No items match your search.
location or PATH
File name: FemaleHealthData
Save as type: CSV (Comma delimited)
Spyder (Python 3.6)
File Edit Search Source Run Debug Consoles Projects Tools View Help
C:\Users\profa\Documents\Python Scripts\TextMining\DATA
PATH TO THE DATA FILE
Editor - C:\Users\profa\Documents\Python Scripts\TextMining\HelloWorld.py
:Line.py
ANLY503plotlymap.py
MatplotlibExamples.py
HelloWorld.py
1# -*- coding: utf-8 -*-
2"""
3Created on Thu Dec 20 21:20:56 2018
4
5@author: profa
6"""
7
8## This is a comment
9## Next, to make sure everything
10## Works - we will print "hello world"
11print("Hello World")
12## To test this - save it - give it a name
File explorer
Name
Size
Type
FemaleHealthData.csv
532 bytes csv File
File explorer
Variable explorer
Help

```
"""
Created on Thu Dec 20 21:52:25 2018

@author: profa
"""

## Reading in basic data using pandas
## READ ABOUT pandas!
import pandas as pd
## the "as" allows you to set a "nickname"
## My nickname is pd
## I can now call any pandas methods using pd.whatever....
MyDataFrame = pd.read_csv("DATA/FemaleHealthData.csv")
print(MyDataFrame.head())

```

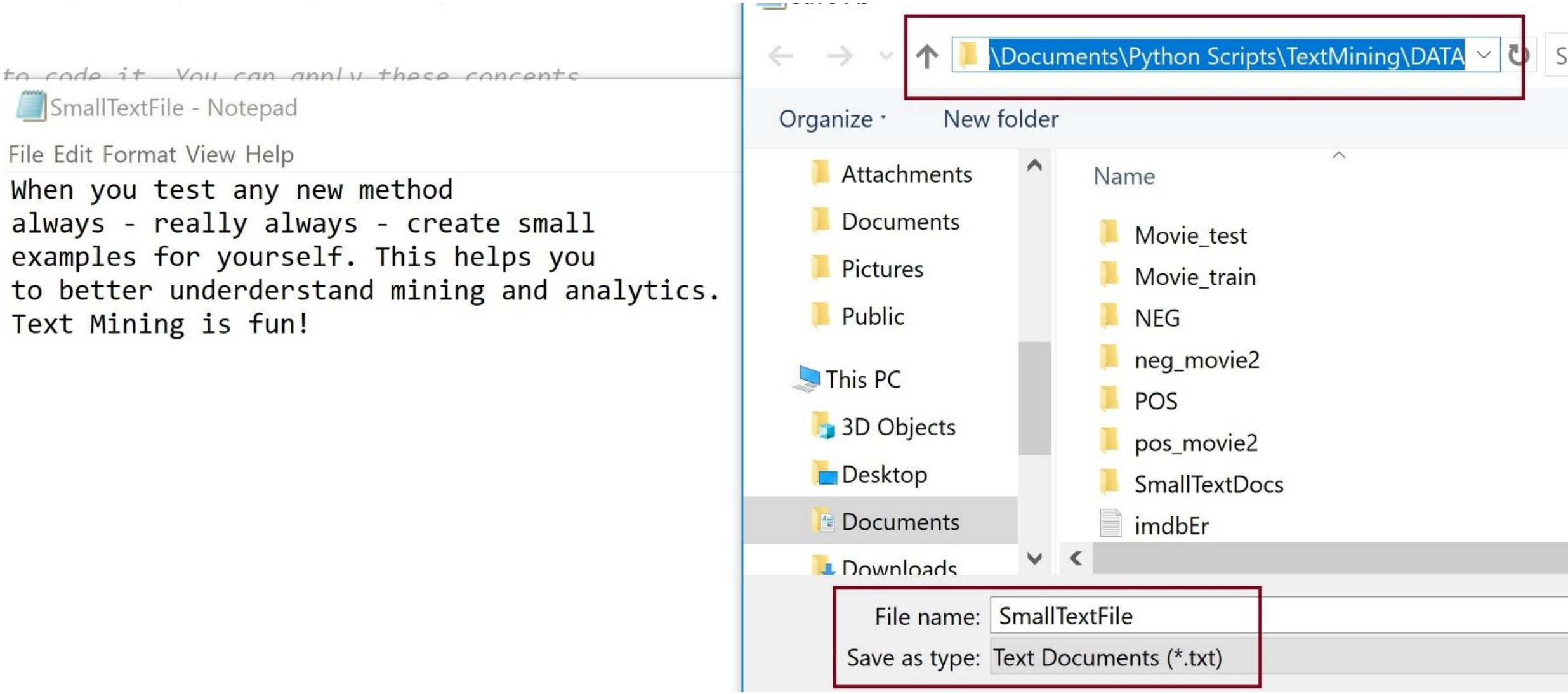



```
[MSC v.1900 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.0.1 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/profa/Documents/Python Scripts/TextMining/
HelloWorld.py', wdir='C:/Users/profa/Documents/Python Scripts/TextMining
Hello World

In [2]: runfile('C:/Users/profa/Documents/Python Scripts/TextMining/
ReadData1.py', wdir='C:/Users/profa/Documents/Python Scripts/TextMining
   FEMALEID  AGE  HEIGHT  WEIGHT  WAIST
0       295   23    64.3   114.8   67.2
1      2739   32    66.4   149.3   82.5
2      2992   25    62.3   107.8   66.7
3      3745   55    62.3   160.1   93.0
4      4486   27    59.6   127.1   82.6
```

Regular Expressions and Text

Gates

Always Create A Small Test Example



Using Python's re for Regular Expressions

```
425 ##########################################
426 ###
427 ### Regular Expressions Small Examples
428 ###
429 ##########################################
430 print("REGULAR EXPRESSIONS....")
431 import re
432 ## Read in the data
433 MyText = open('DATA/SmallTextFile.txt')
434 MyList=[]
435 for line in MyText:
436     print("The next line is, ", line)
437     ##Strip (remove) whitespace from end
438     line = line.rstrip()
439     line = re.sub('[!@#$-.]', '', line)
440     ## Remove all extra whitespace
441     line=re.sub( '\s+', ' ', line).strip()
442     ## Remove newline
443     line = line.strip("\n")
444     line = line.lower()
445     print("Now the line is: ", line)
446     MyList.append(line)
447     if re.search('mining', line):
448         print("yes")
449
450 print("MyList is: ", MyList)
451 ## Join
452 Final="".join(MyList)
453 print("The final string is: ", Final)
```

SmallTextFile.txt 174 bytes txt File 1/1/2019 1:20 PM

File explorer | Variable explorer | Help

IPython console

Console 1/A

```
REGULAR EXPRESSIONS....
The next line is,  When you test any new method

Now the line is:  when you test any new method
The next line is,  always - really always - create small

Now the line is:  always really always create small
The next line is,  examples for yourself. This helps you

Now the line is:  examples for yourself this helps you
The next line is,  to better underderstand mining and analytics.

Now the line is:  to better underderstand mining and analytics
yes
The next line is,  Text Mining is fun!
Now the line is:  text mining is fun
yes
MyList is:  ['when you test any new method', 'always really always create small',
'examples for yourself this helps you', 'to better underderstand mining and analytics',
'text mining is fun']
The final string is:  when you test any new methodalways really always create
smallexamples for yourself this helps youto better underderstand mining and
analyticstext mining is fun
```

Searching with re

```
456## Searching at the beginning of a string
457MyText2 = "Hello! This is an example :)"
458Result=re.search('Hello!', MyText2)
459print(Result)
460if(Result):
461    print("Found")
462
463Result2=re.search('Hel', MyText2)
464print(Result2)
465if(Result2):
466    print("Found Hel")
```

```
<_sre.SRE_Match object; span=(0, 6), match='Hello!'>
Found
<_sre.SRE_Match object; span=(0, 3), match='Hel'>
Found Hel
```

Notes:

1) From code on the previous slide, you should also have a .close() at some point:

Example:

MyText.close()

2) If you forget to close your files, odd things will start to happen ;)

3) No one really “memorizes re”. In general, when you want to do something, you think about what you want to do and then you Google.

4) This link is a HUGE tutorial and is a big help – bookmark it….

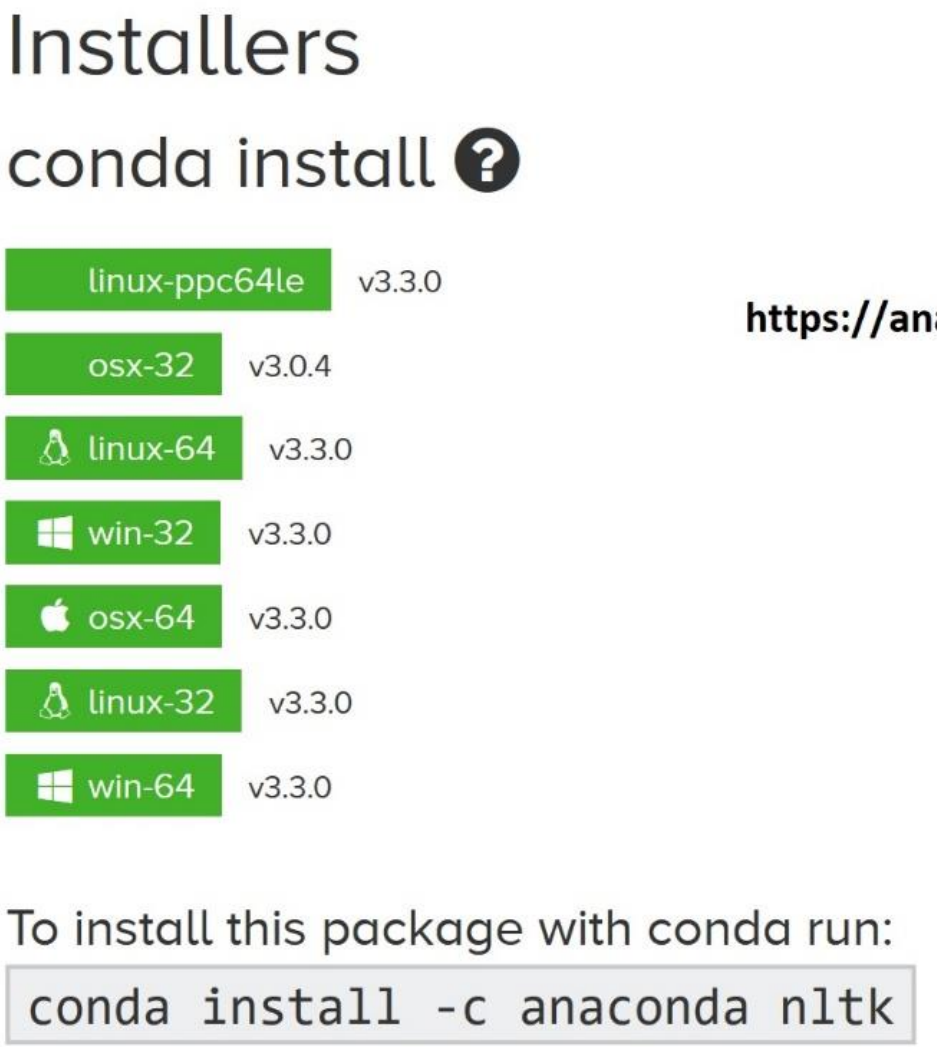
RE: https://www.py4e.com/html3/11-regex

5) I like this tutorial better:

https://docs.python.org/3/library/re.html

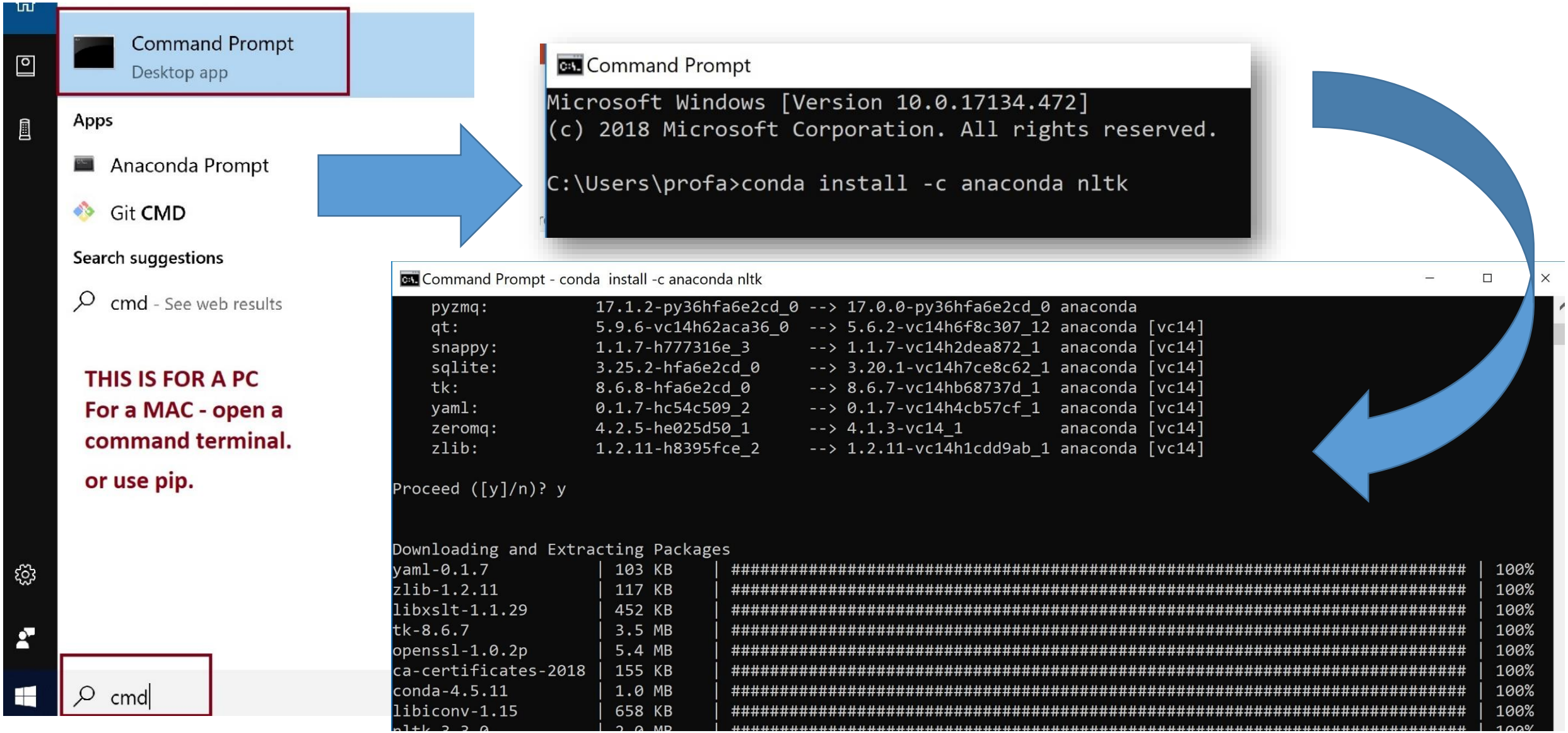
NLTK in Python

1) Get NLTK
2) Practice with a pretend hand-made text file



https://anaconda.org/anaconda/nltk

Command Terminals and getting NLTK



```python
## nltk examples
import nltk
from nltk.tokenize import word_tokenize

text="To be or not to be"

tokens = [t for t in text.split()]
print(tokens)

freq = nltk.FreqDist(tokens)

for key,val in freq.items():
    print (str(key) + ':' + str(val))

freq.plot(20, cumulative=False)

mytext = "Hiking is fun! Hiking with dogs is more fun :)"
print(word_tokenize(mytext))

```

File explorer | Variable explorer | Help

IPython console

Console 1/A

```
In [9]: runfile('C:/Users/profa/Documents/Python Scripts/TextMining/
nltkExample1.py', wdir='C:/Users/profa/Documents/Python Scripts/
TextMining')
['To', 'be', 'or', 'not', 'to', 'be']
<generator object bigrams at 0x0000026B52CCF7D8>

In [10]: runfile('C:/Users/profa/Documents/Python Scripts/TextMining/
nltkExample1.py', wdir='C:/Users/profa/Documents/Python Scripts/
TextMining')
['To', 'be', 'or', 'not', 'to', 'be']
To:1
be:2
or:1
not:1
to:1
['Hiking', 'is', 'fun', '!', 'Hiking', 'with', 'dogs', 'is', 'more',
'fun', ':', ')']
```

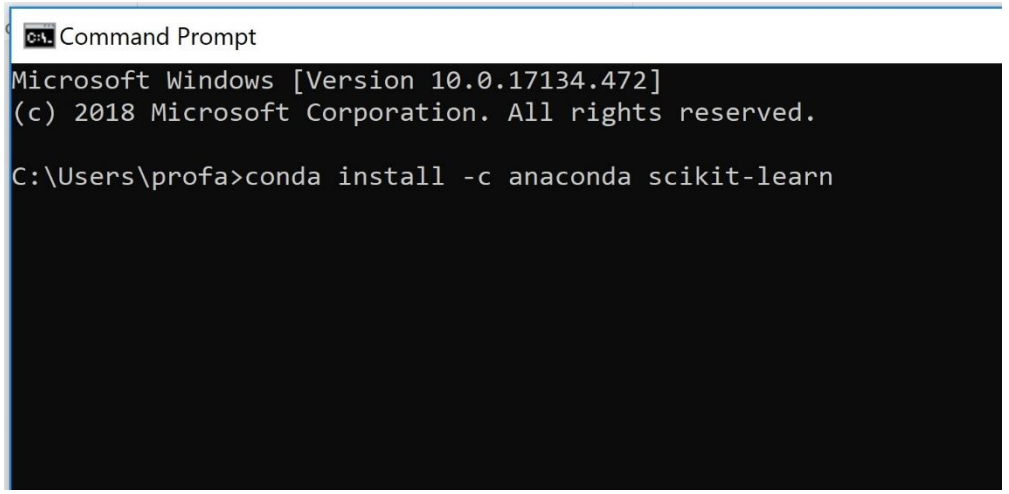
Getting sklearn

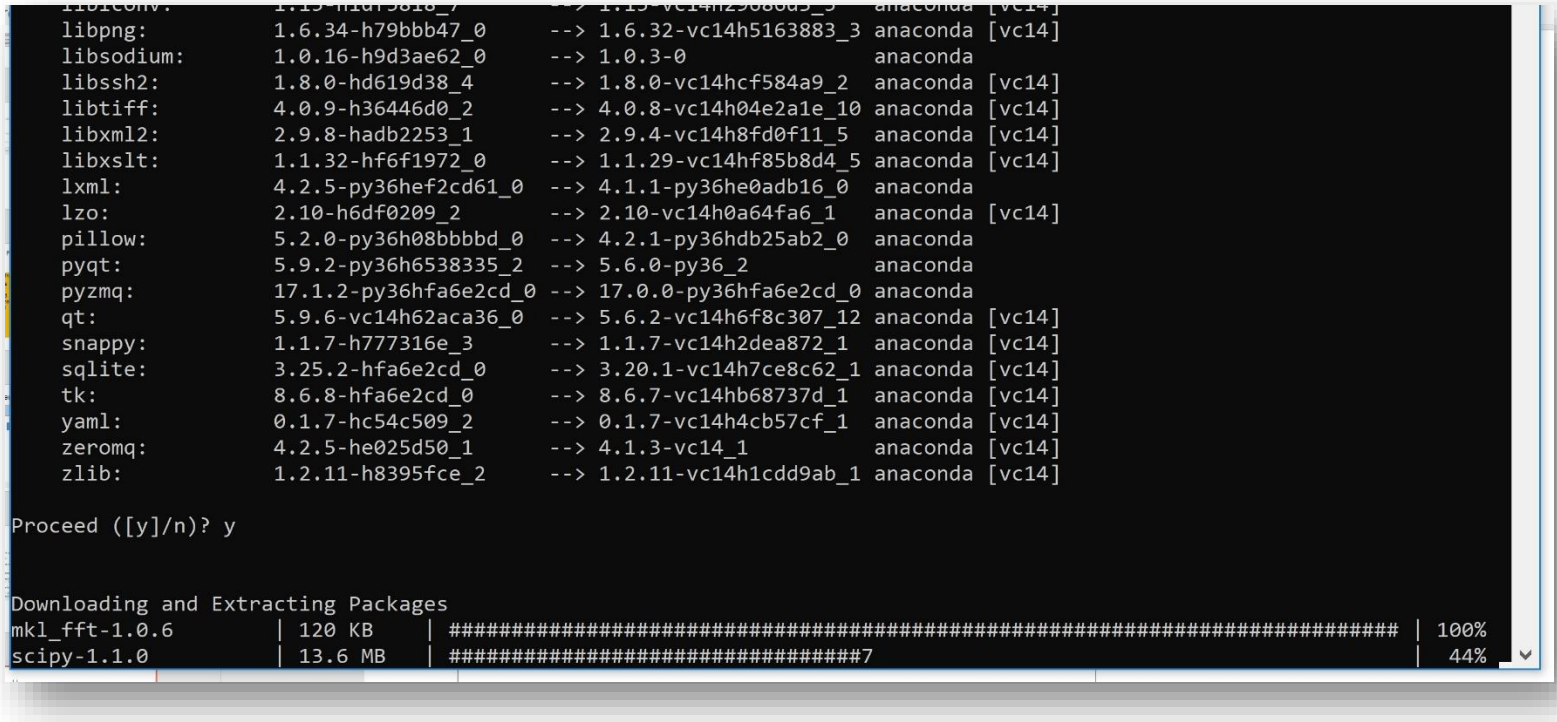
Ref:

https://anaconda.org/anaconda/scikit-learn

Command:

conda install -c anaconda scikit-learn





Test your sklearn

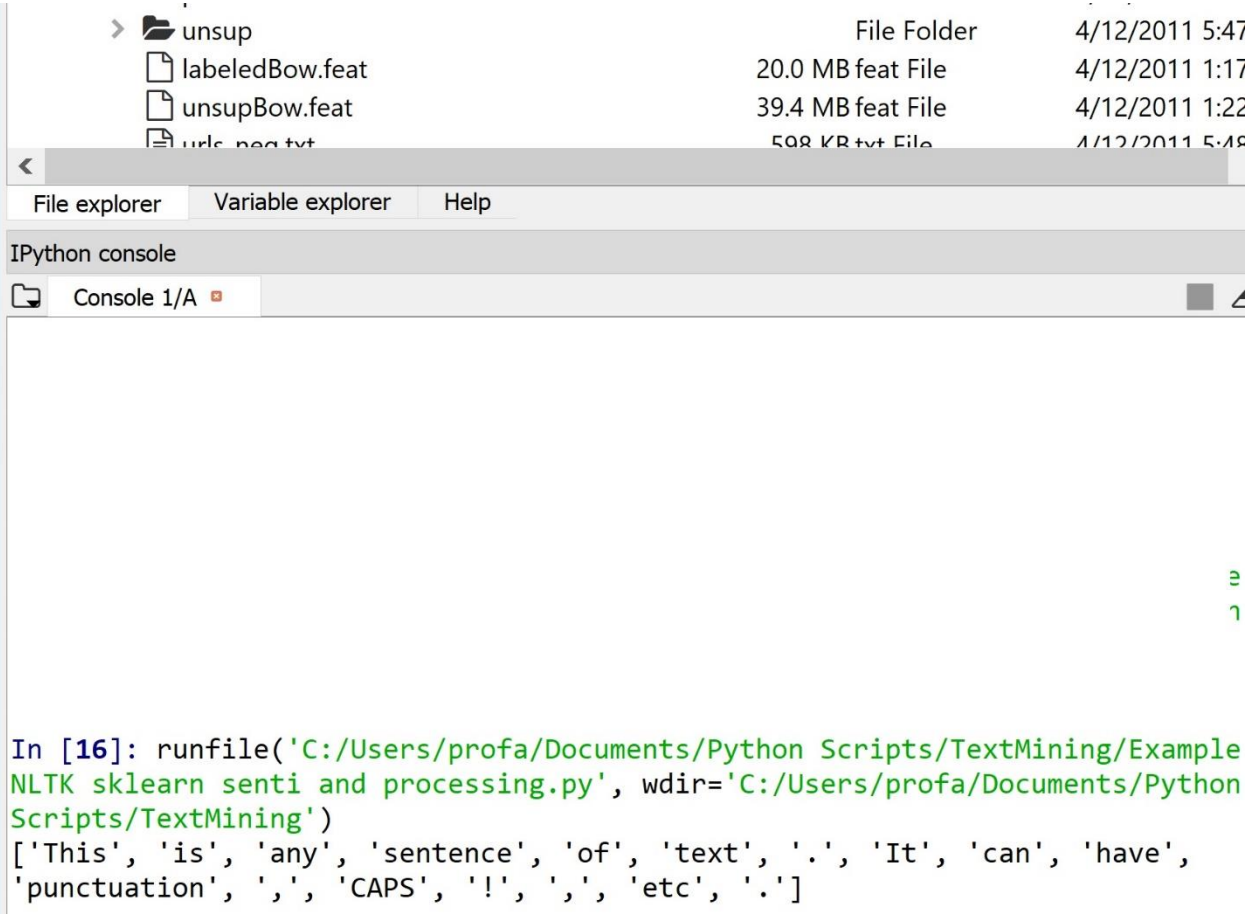
#Type this into a new Python file, save, and run:

```
from sklearn.feature_extraction.text import CountVectorizer
print("hello")
```

If your sklearn is installed, this will print hello and will not show any errors.

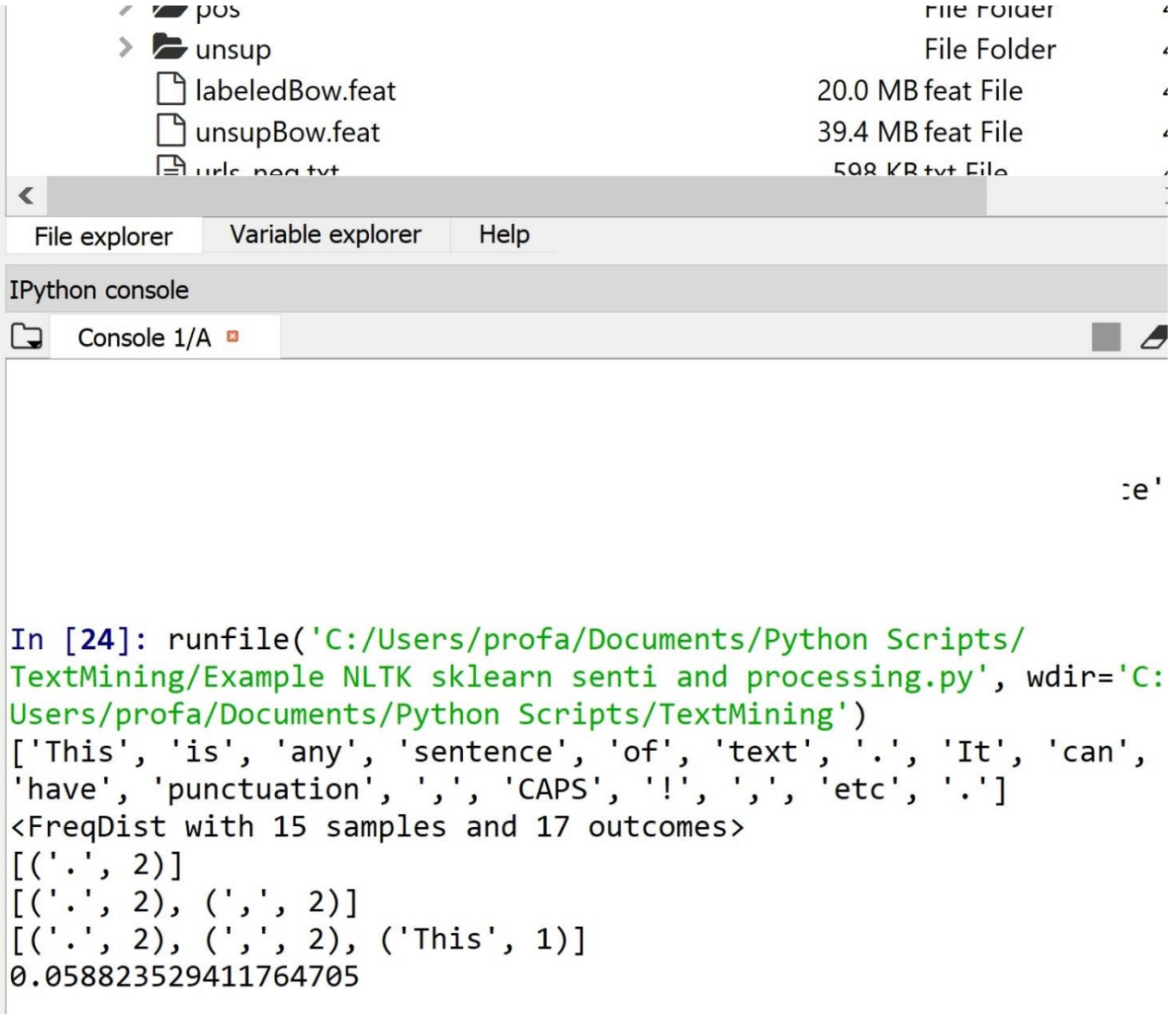
Tokenize into words

```
7 import nltk
8 import pandas
9 from sklearn.feature_extraction.text import CountVectorizer
10 from nltk.tokenize import word_tokenize
11
12 ## Tokenization
13 ## Breaking text into tokens -  in this case - words
14 text="This is any sentence of text. It can have punctuation, CAPS!, etc."
15 tokenized_word=word_tokenize(text)
16 print(tokenized_word)
```



Tokens, Frequency, and Plot

```
import nltk
import pandas
from sklearn.feature_extraction.text import CountVectorizer
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
import matplotlib.pyplot as plt
## Tokenization
## Breaking text into tokens -  in this case - words
text="This is any sentence of text. It can have punctuation, CA
tokenized_word=word_tokenize(text)
print(tokenized_word)

## Looking at word frequency
fdist = FreqDist(tokenized_word)
print(fdist)
print(fdist.most_common(1))  ## most common lleft to right
print(fdist.most_common(2))  ## two most common
print(fdist.most_common(3))  ## three most common
print(fdist.freq("is"))  ## how frequent
## "is" occurs once in 17 words. 1/17 = .058
fdist.N()  # freq of each

# Visualize word frequency
fdist.plot(30,cumulative=False)
plt.show()
```



plot

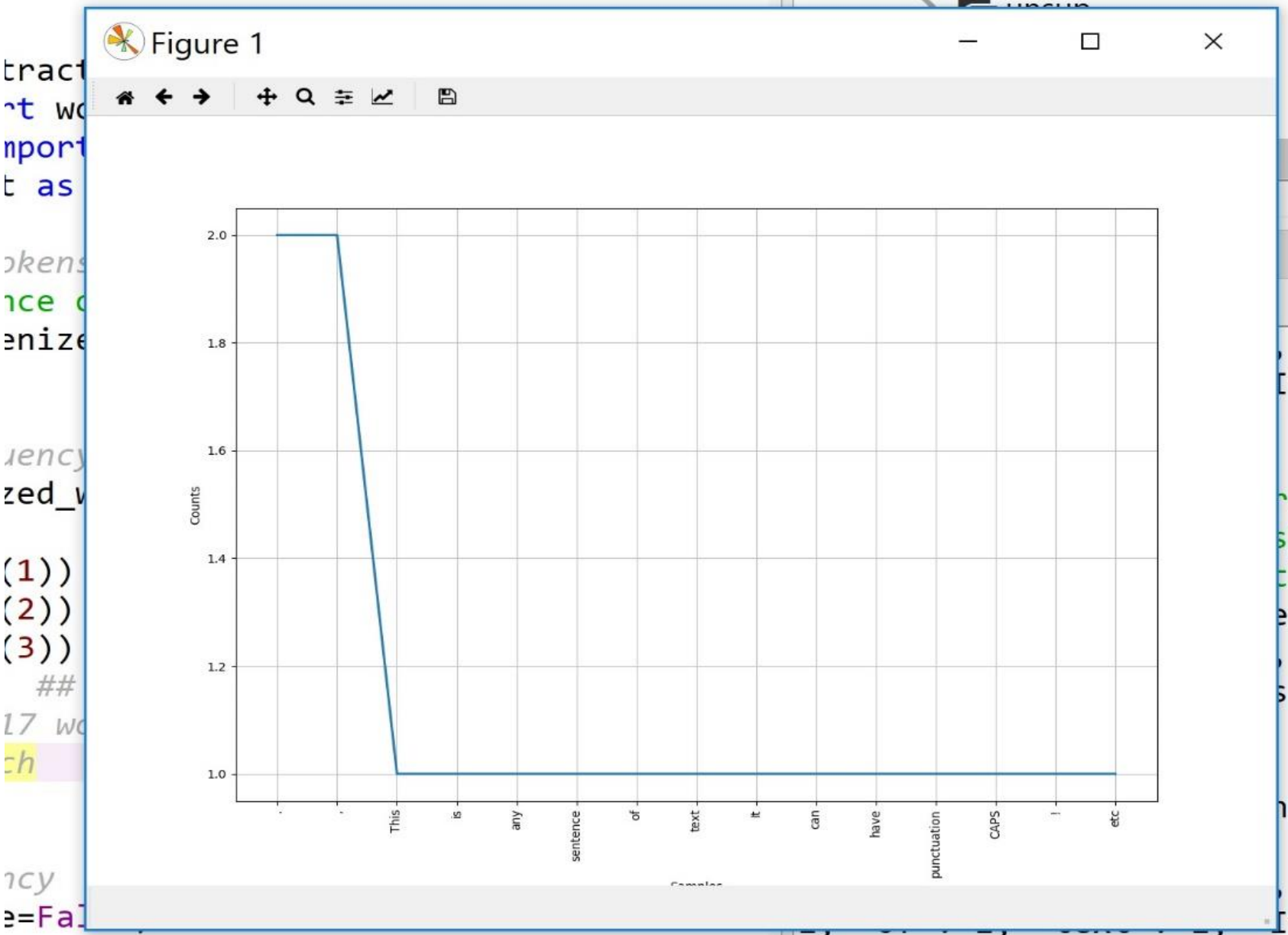
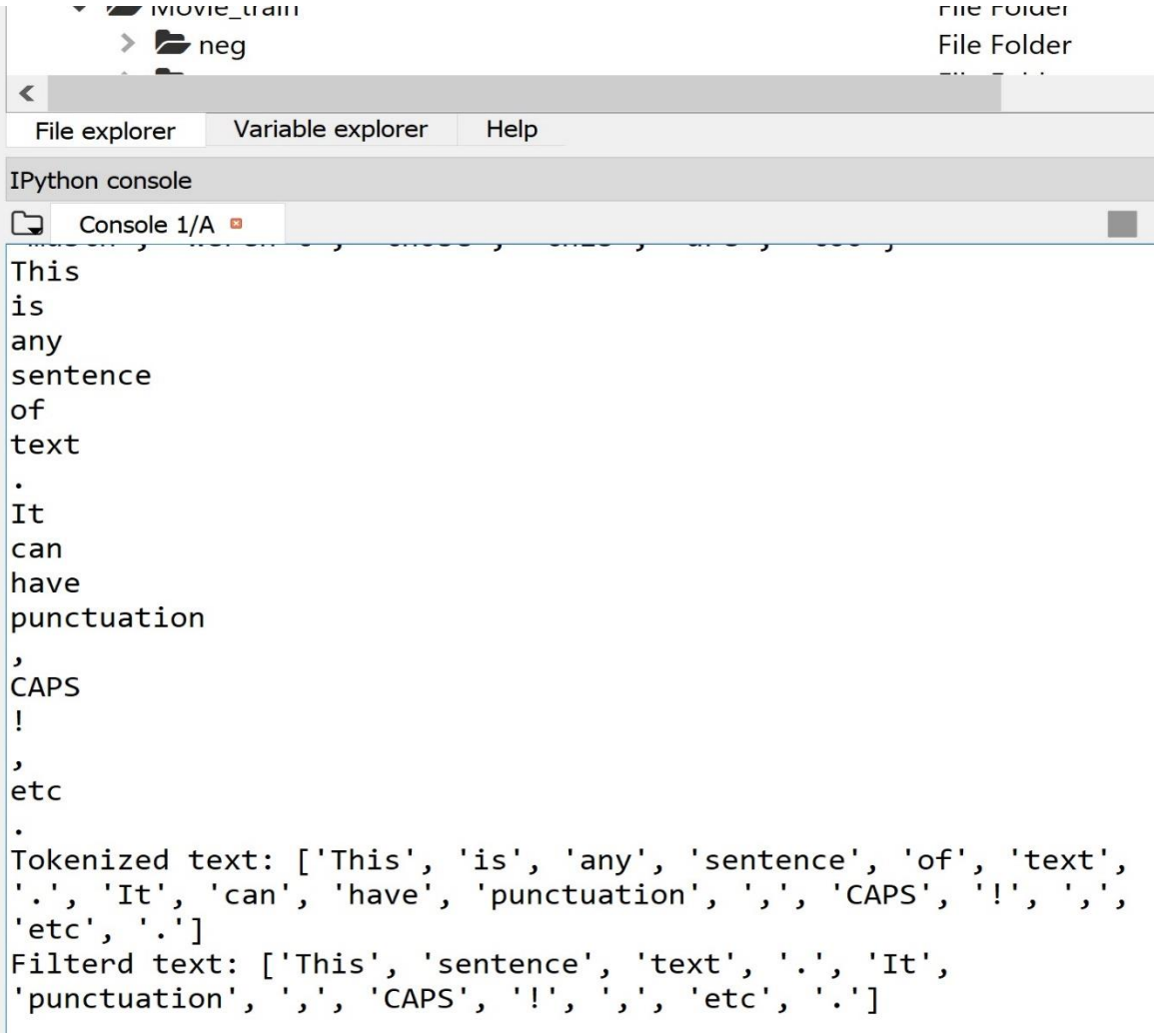


Figure 1
2.0
1.8
1.6
1.4
1.2
1.0
Counts
This
is
any
sentence
of
text
It
can
have
punctuation
CAPS
etc

Stopwords

from nltk.corpus import stopwords

```
13 from nltk.corpus import stopwords
14 ## Tokenization
15 ## Breaking text into tokens -  in this case - words
16 text="This is any sentence of text. It can have punctuation,
17 tokenized_word=word_tokenize(text)
18 print(tokenized_word)
19 ## Looking at word frequency
20 fdist = FreqDist(tokenized_word)
21 print(fdist)
22 print(fdist.most_common(1))  ## most common lleft to right
23 print(fdist.most_common(2))  ## two most common
24 print(fdist.most_common(3))  ## three most common
25 print(fdist.freq("is"))  ## how frequent
26 ## "is" occurs once in 17 words. 1/17 = .058
27 fdist.N()  # freq of each
28 # Visualize word frequency
29 fdist.plot(30,cumulative=False)
30 plt.show()
31 ## Stopwords - English
32 stop_words=set(stopwords.words("english"))
33 print(stop_words)
34 ## Removing Stopwords
35 filtered_text=[]   ## Create a new empty list
36 for w in tokenized_word:
37     print(w)
38     if w not in stop_words:
39         filtered_text.append(w)
40 print("Tokenized text:",tokenized_word)
41 print("Filterd text:",filtered_text)
```



Stemming

```
43
44# Stemming
45from nltk.stem import PorterStemmer
46from nltk.tokenize import sent_tokenize, word_tokenize
47
48ps = PorterStemmer()    ## method from nltk
49
50stemmed_words=[]  ## make new empty list
51for w in filtered_text:
52    stemmed_words.append(ps.stem(w))
53
54print("Filtered:",filtered_text)
55print("Stemmed:",stemmed_words)
```

```
Filtered: ['This', 'sentence', 'text', '.', 'It', 'punctuation',
',', 'CAPS', '!', ',', 'etc', '.']
Stemmed: ['thi', 'sentenc', 'text', '.', 'It', 'punctuat', ',',
'cap', '!', ',', 'etc', '.']
```

Lemmatization vs. Stemming

```
##-----------------------------
#Lemmatization reduces words to their base word
#-----------------------------------------------
# Lemmatization is usually more sophisticated than
#stemming. Stemmer works on an individual word without
# knowledge of the context. For example, The word "better"
# has "good" as its lemma. This thing will miss by
# stemming because it requires a dictionary look-up.
from nltk.stem.wordnet import WordNetLemmatizer
lem = WordNetLemmatizer()  ## method we are using
from nltk.stem.porter import PorterStemmer
stem = PorterStemmer()
word = "flying"
print("Lemmatized Word:",lem.lemmatize(word,"v"))
print("Stemmed Word:",stem.stem(word))
```

```
Lemmatized Word: fly
Stemmed Word: fli
```

Part-of-Speech(POS) tagging

```
80
81## ----------------------
82## Part-of-Speech(POS) tagging
83## ----------------------------------
84# identify the grammatical group
85# Whether it is a NOUN, PRONOUN,
86# ADJECTIVE, VERB, ADVERBS
87#---------------------------------------
88sent = "Three wonderful things in life are hiking, moutains,
89Mytokens=nltk.word_tokenize(sent)
90print(Mytokens)
91MyTAGS = nltk.pos_tag(Mytokens)
92print(MyTAGS)
93
94
95
96
```

```
['Three', 'wonderful', 'things', 'in', 'life', 'are', 'hiking',
',', 'moutains', ',', 'and', 'COFFEE', '!']
[('Three', 'CD'), ('wonderful', 'JJ'), ('things', 'NNS'), ('in',
'IN'), ('life', 'NN'), ('are', 'VBP'), ('hiking', 'VBG'), (',',
','), ('moutains', 'NNS'), (',', ','), ('and', 'CC'), ('COFFEE',
'NNP'), ('!', '.')]
```

Formatting TEXT for Analysis

1) Text is messy!
2) What can it look like?
    1) JSON
    2) HTML
    3) Documents
    4) FB
    5) Twitter
    6) Other…
3) What are the variables or features of a document (or website or Facebook post, or Tweet, or novel, or Amazon ranking comment…)?

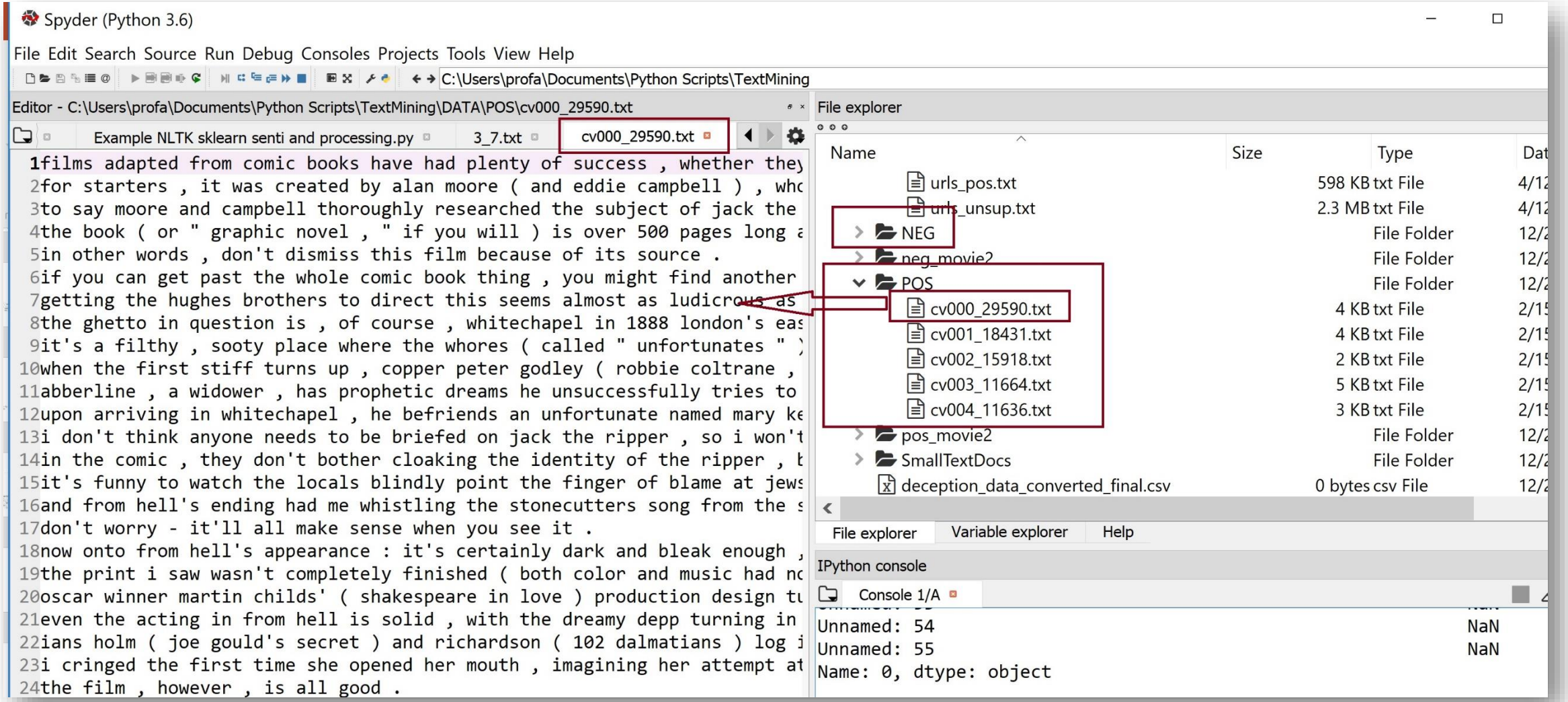
What is vectorization?

1) Grab the text (and all the garbage that comes with it).
2) Clean it so that it is just text – all lowercase.
3) If it is labeled – keep the label.
4) Read it into a Corpus object
5) Build a DF or Matrix where the words are the variables.
6) It is also possible that n-grams (such as bigrams) can be the variables. More on this later….
7) All of these steps require *a lot* of cleaning and prep.

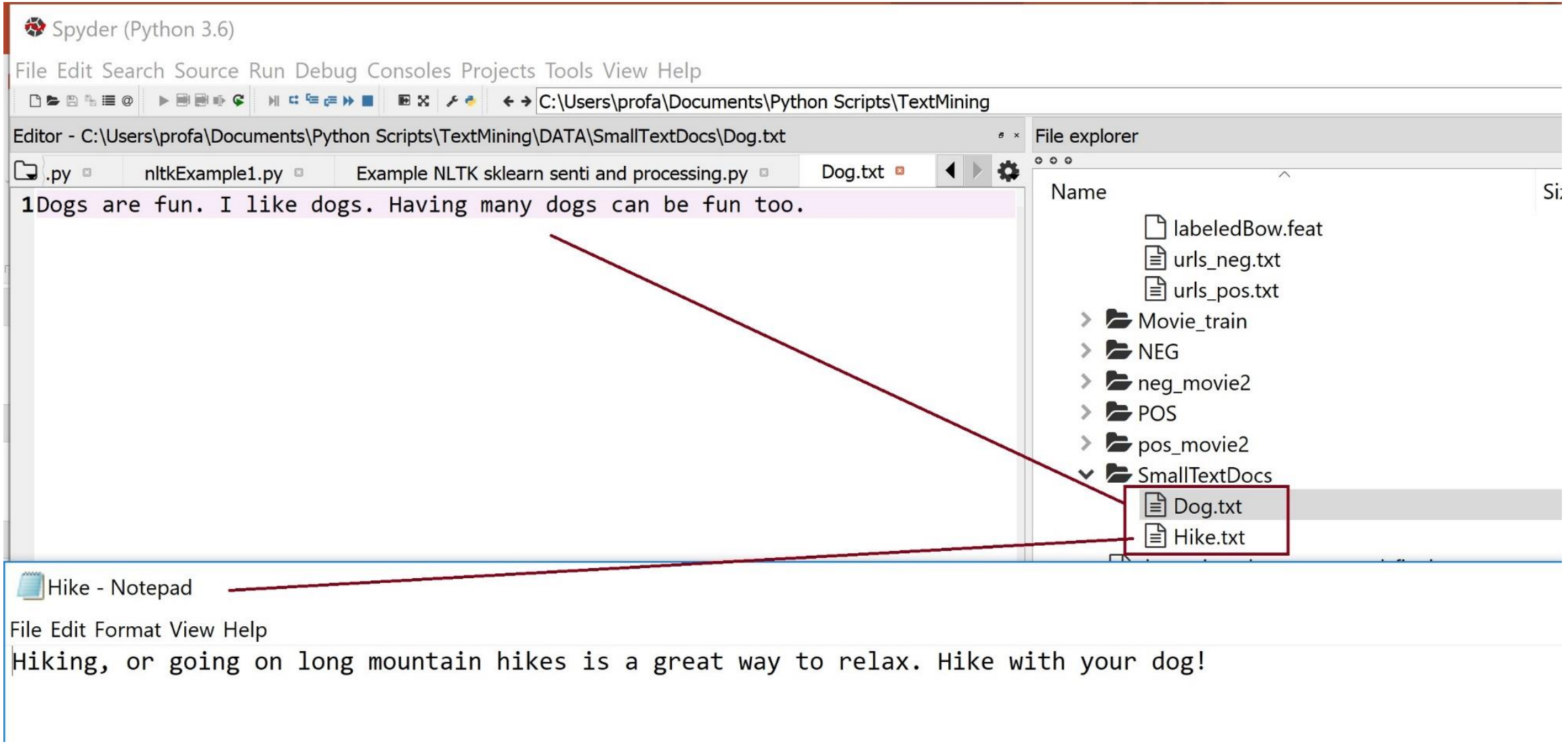
Examples

```
## Note: the following examples will use at least these libraries
import nltk
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from nltk.tokenize import word_tokenize
from nltk.probability import FreqDist
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
## For Stemming
from nltk.stem import PorterStemmer
from nltk.tokenize import sent_tokenize, word_tokenize
import os
```

Example 1: Positive and Negative Folders of Text files



Using SMALL Datasets to Practice: Our Data

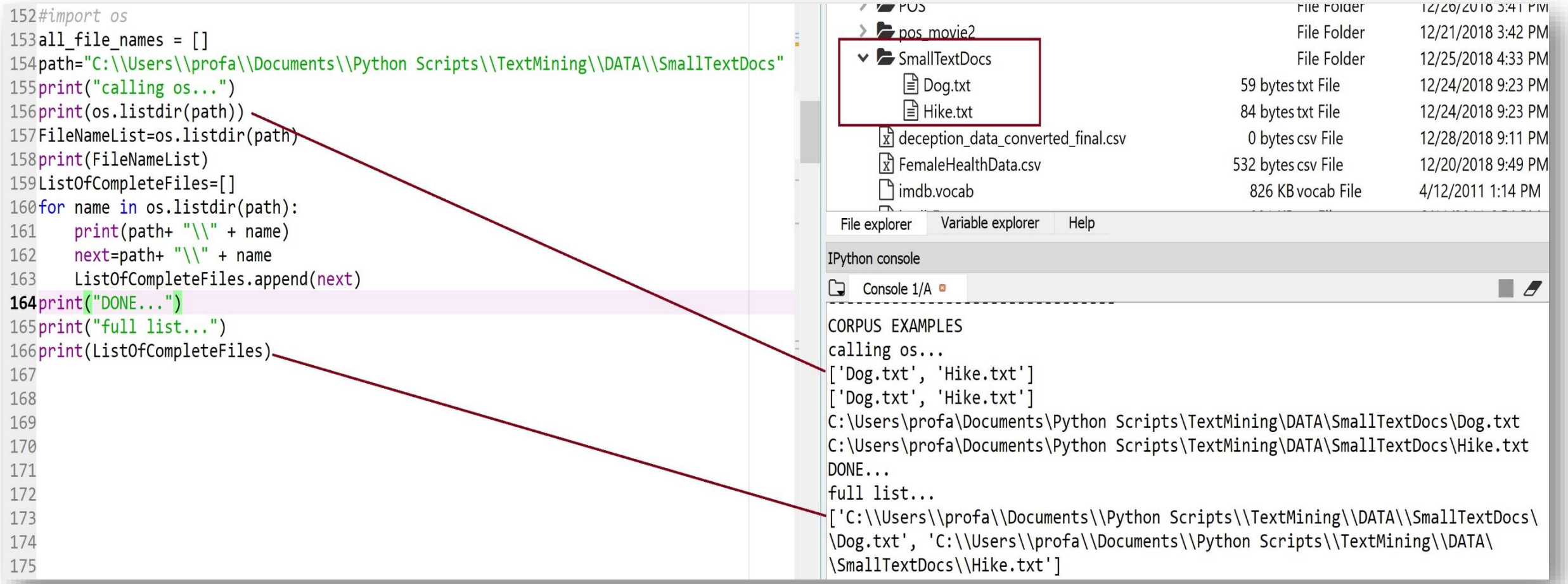


```
118 import sklearn
119 from sklearn.feature_extraction.text import CountVectorizer
120 MyVectorizer1=CountVectorizer(
121         input='content', ## can be set as 'content', 'file', or 'filename'
122         #If set as 'filename', the **sequence passed as an argument to fit**
123         #is expected to be a list of filenames
124         #https://scikit-learn.org/stable/modules/generated/
125         ##sklearn.feature_extraction.text.CountVectorizer.html#
126         ##examples-using-sklearn-feature-extraction-text-countvectorizer
127         encoding='latin-1',
128         decode_error='ignore', #{'strict', 'ignore', 'replace'}
129         strip_accents=None, # {'ascii', 'unicode', None}
130         lowercase=True,
131         preprocessor=None,
132         tokenizer=None,
133         #stop_words='english', #string {'english'}, list, or None (default)
134         stop_words=None,
135         token_pattern='(?u)\b\w\w+\b', #Regular expression denoting what constitutes
136         ngram_range=(1, 1),
137         analyzer='word',
138         max_df=1.0, # ignore terms w document freq strictly > threshold
139         min_df=1,
140         max_features=None,
141         vocabulary=None,
142         binary=False, #If True, all non zero counts are set to 1
143         #dtype=<class 'numpy.int64'>
144         )
145 #print(MyVectorizer1)
146 MyVect2=CountVectorizer()
147 MyVect3=CountVectorizer(input='filename')
```

Count Vectorizers

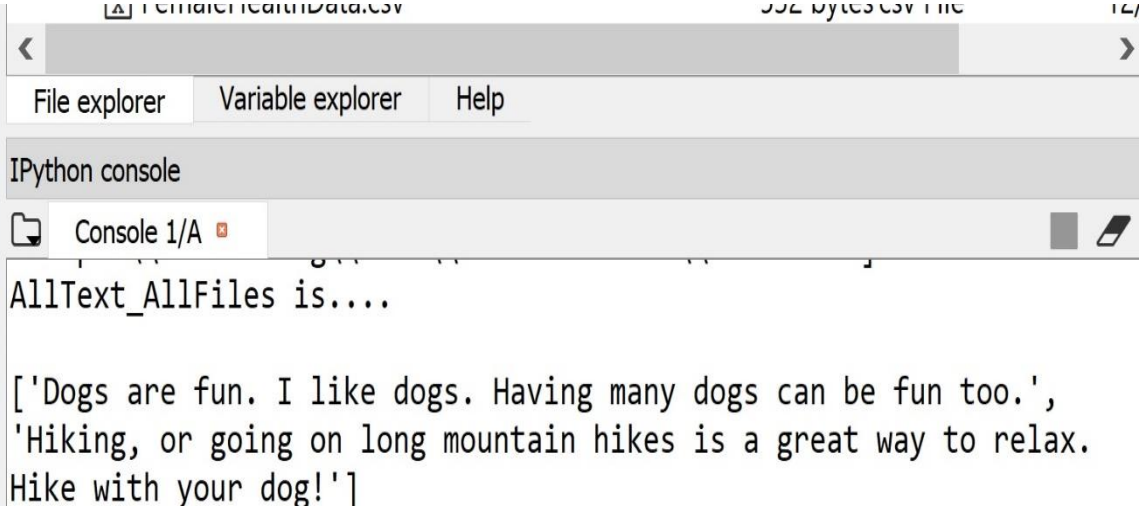
In this next example, I will use MyVect3. Notice that input='filename'. I need this as I am reading in files from folders.

Using the OS to read in the list of files



Read in the Text

```
168 AllText_AllFiles=[]
169 for file in ListOfCompleteFiles:
170     FILE=open(file)
171    # print(FILE.read())
172     content=FILE.read()
173     AllText_AllFiles.append(content)
174     FILE.close()
175 print("AllText_AllFiles is....\n")
176 print(AllText_AllFiles)
177
```



Fit Transforms…

```
175
176
177 print("FIT TRANSFORM----------------")
178 ## FIT TRANSFORM USING a CountVect
179 print("The ListOfCompleteFiles is ...")
180 print(ListOfCompleteFiles)
181 X3=MyVect3.fit_transform(ListOfCompleteFiles)
182 print("The AllText_AllFiles is...")
183 print(AllText_AllFiles)
184 X2=MyVect2.fit_transform(AllText_AllFiles)
185 print(type(X2))
186 print(type(X3))
187 print(X2.get_shape())
188 ColumnNames2=MyVect2.get_feature_names()
189 ColumnNames3=MyVect3.get_feature_names()
190 print("The col name for 3 ", ColumnNames3)
191
192 #import pandas as pd
193 #https://pandas.pydata.org/pandas-docs/version/0.23.4/generated/
194 ## !!!!!!!!!! The final DTM in Python!! (this took 20 hours :)
195 CorpusDF_A2=pd.DataFrame(X2.toarray(),columns=ColumnNames2)
196 print("The DF for 2 is...", CorpusDF_A2)
197 CorpusDF_A=pd.DataFrame(X3.toarray(),columns=ColumnNames3)
198 print("The DF for 3 is...", CorpusDF_A)
```

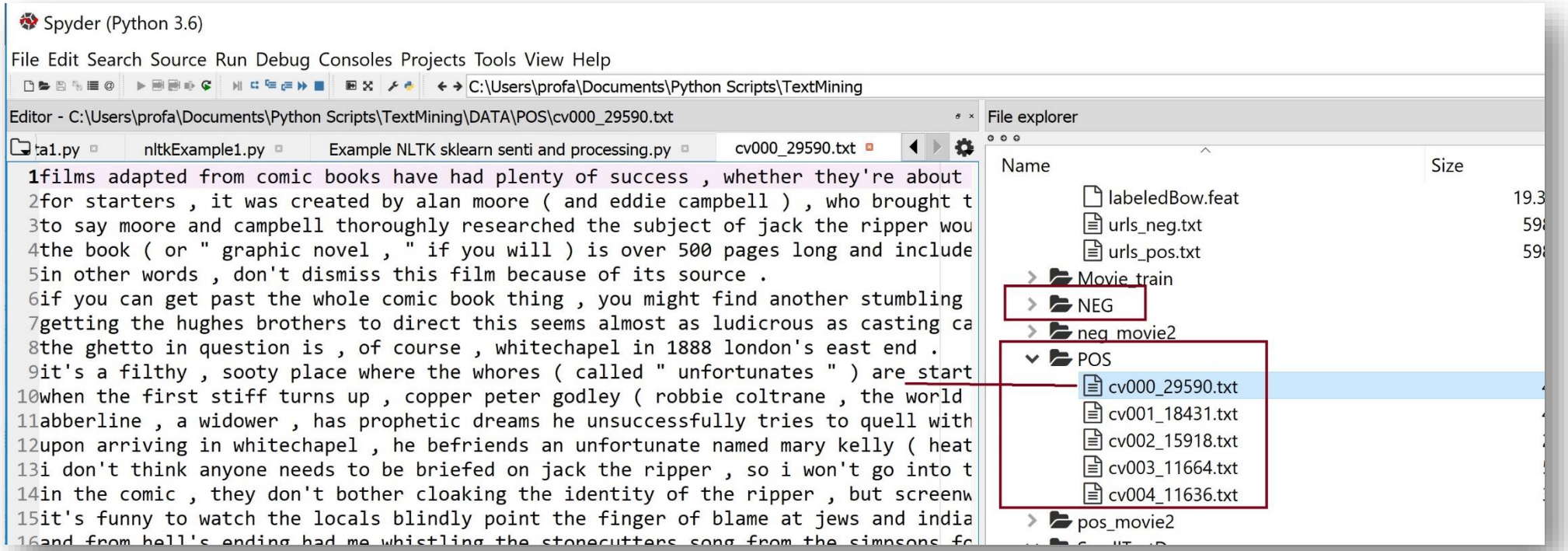
```
FIT TRANSFORM----------------
The ListOfCompleteFiles is ...
['C:\\Users\\profa\\Documents\\Python Scripts\\TextMining\\DATA\\SmallTextDocs\\Dog.txt', 'C:\\Users
\\profa\\Documents\\Python Scripts\\TextMining\\DATA\\SmallTextDocs\\Hike.txt']
The AllText_AllFiles is...
['Dogs are fun. I like dogs. Having many dogs can be fun too.', 'Hiking, or going on long mountain
hikes is a great way to relax. Hike with your dog!']
<class 'scipy.sparse.csr.csr_matrix'>
<class 'scipy.sparse.csr.csr_matrix'>
(2, 25)
The col name for 3  ['are', 'be', 'can', 'dog', 'dogs', 'fun', 'going', 'great', 'having', 'hike',
'hikes', 'hiking', 'is', 'like', 'long', 'many', 'mountain', 'on', 'or', 'relax', 'to', 'too',
'way', 'with', 'your']
The DF for 2 is...    are  be  can  dog  dogs  fun  ...   relax  to  too  way  with  your
0    1   1    1    0    3    2  ...      0   0    1    0    0    0
1    0   0    0    1    0    0  ...      1   1    0    1    1    1

[2 rows x 25 columns]
The DF for 3 is...    are  be  can  dog  dogs  fun  ...   relax  to  too  way  with  your
0    1   1    1    0    3    2  ...      0   0    1    0    0    0
1    0   0    0    1    0    0  ...      1   1    0    1    1    1

[2 rows x 25 columns]
```

So far

1) We have arrays of numbers and column names.
2) But – we do not yet have a proper dataframe….
3) We can confirm the counts – and should!
4) We also showed that we can read in text or files.
5) We also have dog and dogs seperatly, etc…

```
The DF for 2 is...    are  be  can  dog  dogs  fun  ...   relax  to  too  way  with  your
0    1   1    1    0     3    2  ...       0   0    1    0     0     0
1    0   0    0    1     0    0  ...       1   1    0    1     1     1

[2 rows x 25 columns]
The DF for 3 is...    are  be  can  dog  dogs  fun  ...   relax  to  too  way  with  your
0    1   1    1    0     3    2  ...       0   0    1    0     0     0
1    0   0    0    1     0    0  ...       1   1    0    1     1     1
```

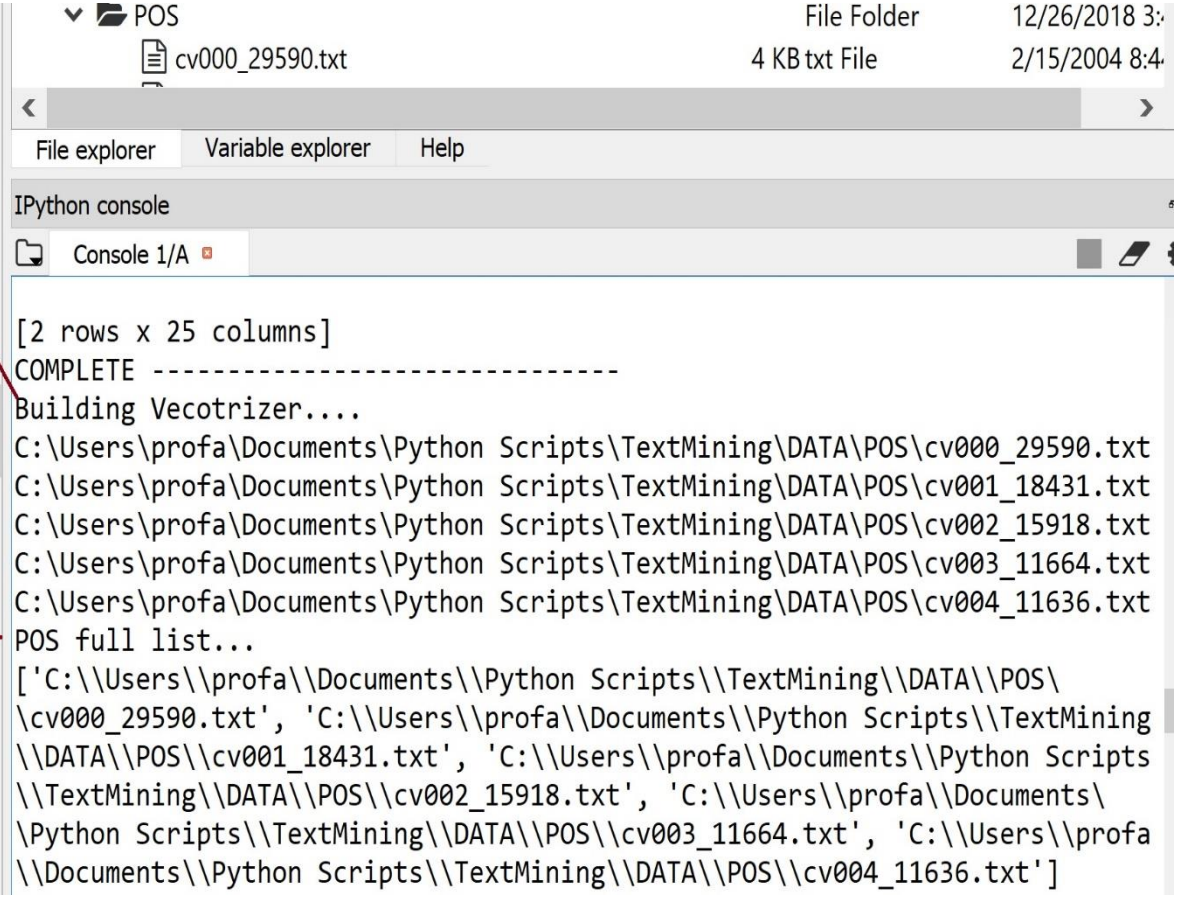
Next Step -



1) We will take what we have learned, and repeat this process.
2) The goal this time is to have a cleaner and easier to view DF
3) We want the labels and column names to part of the DF…
4) In this next example set, we will use a folder called POS that contains many text files all with positive sentiment and a folder called NEG with text files with negative sentiment.

Example2:

```
235### Step 1: Read in the POS files corpus into a DF1
236print("Building Vecotrizer....")
237MyVect4=CountVectorizer(input='filename',
238                         analyzer = 'word',
239                         stop_words='english',
240                         token_pattern='(?u)[a-zA-Z]+'
241                         )
242path="C:\\Users\\profa\\Documents\\Python Scripts\\TextMining\\DATA\\POS"
243
244##Create empty list
245POSListOfCompleteFiles=[]
246for name in os.listdir(path):
247    print(path+ "\\" + name)
248    next=path+ "\\" + name
249    POSListOfCompleteFiles.append(next)
250
251print("POS full list...")
252print(POSListOfCompleteFiles)
253
254
255
256
257
```



```
244##Create empty list
245POSListOfCompleteFiles=[]
246for name in os.listdir(path):
247    print(path+ "\\" + name)
248    next=path+ "\\" + name
249    POSListOfCompleteFiles.append(next)
250
251print("POS full list...")
252print(POSListOfCompleteFiles)
253
254print("FIT with Vectorizer...")
255X4=MyVect4.fit_transform(POSListOfCompleteFiles)
256print(type(X4))
257print(X4.get_shape())
258POSColumnNames=MyVect4.get_feature_names()
259print("Column names: ", POSColumnNames[0:10])
260print("Building DF....")
261#import pandas as pd
262POS_CorpusDF_A=pd.DataFrame(X4.toarray(),columns=POSColumnNames)
263print(POS_CorpusDF_A)
264
265## This looks good. Notice that when I built the Vectorizer
266## above, that I used [a-zA-Z] which means
267## letter ONLY - no numbers.
268
269
270
271
```

Name	Size	Type	Date Modified
labeledBow.feat	19.3 MB	feat File	4/12/2011 1:25 PM
urls_neg.txt	598 KB	txt File	4/12/2011 5:48 AM
urls_pos.txt	598 KB	txt File	4/12/2011 5:48 AM

File explorer | Variable explorer | Help

IPython console

Console 1/A

```
POS full list...
['C:\\Users\\profa\\Documents\\Python Scripts\\TextMining\\DATA\\POS\
\cv000_29590.txt', 'C:\\Users\\profa\\Documents\\Python Scripts\\TextMining\\DATA\
\POS\\cv001_18431.txt', 'C:\\Users\\profa\\Documents\\Python Scripts\\TextMining\
\DATA\\POS\\cv002_15918.txt', 'C:\\Users\\profa\\Documents\\Python Scripts\
\TextMining\\DATA\\POS\\cv003_11664.txt', 'C:\\Users\\profa\\Documents\\Python
Scripts\\TextMining\\DATA\\POS\\cv004_11636.txt']
FIT with Vectorizer...
<class 'scipy.sparse.csr.csr_matrix'>
(5, 1139)
Column names:  ['abberline', 'ably', 'aboard', 'absinthe', 'absolutely', 'academy',
'accent', 'achiever', 'act', 'acting']
Building DF....
   abberline  ably  aboard  absinthe  ...    years  yglesias  york  young
0          2     1       0         1  ...        0         1     0      0
1          0     0       0         0  ...        0         0     0      0
2          0     0       0         0  ...        0         0     0      0
3          0     0       1         0  ...        1         0     1      1
4          0     0       0         0  ...        0         0     1      0
```

Adding the Label for Sentiment

```
268
269 ## Now, we need to add a column for P or N.
270 ## I will call it PosORNeg and because all of these
271 ## are positive, I will fill it with P
272 ## DataFrame.insert(loc, column, value, allow_duplicates=False)
273 #Length=POS_CorpusDF_A.shape
274 #print(Length[0])  ## num of rows
275 #print(Length[1])  ## num of columns
276
277 ## Add column
278 print("Adding new column....")
279 POS_CorpusDF_A["PosORNeg"]="P"
280 print(POS_CorpusDF_A)
281
282 ## OK - now we have a labeled DF
```

```
[5 rows x 1139 columns]
Adding new column....
   abberline  ably  aboard  absinthe  ...  yglesias  york  young  PosORNeg
0          2     1       0         1  ...         1     0      0         P
1          0     0       0         0  ...         0     0      0         P
2          0     0       0         0  ...         0     0      0         P
3          0     0       1         0  ...         0     1      1         P
4          0     0       0         0  ...         0     1      0         P
```

Do the Same Thing for the NEG Folder …

```
289### Now - we will do tha above for
290## the negative docs....
291pathN="C:\\Users\\profa\\Documents\\Python Scripts\\TextMining\\DATA\\NEG"
292##Create empty list
293NEGListOfCompleteFiles=[]
294for name in os.listdir(pathN):
295    print(pathN+ "\\" + name)
296    next=pathN+ "\\" + name
297    NEGListOfCompleteFiles.append(next)
298
299print("full list...")
300print(NEGListOfCompleteFiles)
301X5=MyVect4.fit_transform(NEGListOfCompleteFiles)
302print(type(X5))
303print(X5.get_shape())
304NEGColumnNames=MyVect4.get_feature_names()
305print(NEGColumnNames)
306
307#import pandas as pd
308NEG_CorpusDF_A=pd.DataFrame(X5.toarray(),columns=NEGColumnNames)
309print(NEG_CorpusDF_A)
310NEG_CorpusDF_A["PosORNeg"]="N"
311print(NEG_CorpusDF_A)
```

What's Next??

```
[5 rows x 1139 columns]
Adding new column....
   abberline  ably  aboard  absinthe     ...     yglesias  york  young  PosORNeg
0          2     1       0         1     ...            1     0      0         P
1          0     0       0         0     ...            0     0      0         P
2          0     0       0         0     ...            0     0      0         P
3          0     0       1         0     ...            0     1      1         P
4          0     0       0         0     ...            0     1      0         P
```

```
[5 rows x 1004 columns]
   able  abo  accent  accident     ...     yellow  yes  young  PosORNeg
0     1    0       0         0     ...          0    0      0         N
1     0    2       0         1     ...          0    0      0         N
2     0    0       0         0     ...          1    1      0         N
3     0    0       0         0     ...          0    0      1         N
4     0    0       1         0     ...          0    0      0         N
```

Next: Create a Complete DF with all data and shuffle – then build test and train sets.

1) MERGE

```
319## Create a new large Pos and Neg DF
320## https://pandas.pydata.org/pandas-docs/stable/merging.h
321result = NEG_CorpusDF_A.append(POS_CorpusDF_A)
322print(result)
323## Replace the NaN with 0 because it actually
324## means none in this case
325result=result.fillna(0)
326print(result)
327
328
329
```

```
[5 rows x 1139 columns]
Adding new column....
   abberline  ably  aboard  absinthe    ...     yglesias  york  young  PosORNeg
0          2     1       0         1    ...            1     0      0         P
1          0     0       0         0    ...            0     0      0         P
2          0     0       0         0    ...            0     0      0         P
3          0     0       1         0    ...            0     1      1         P
4          0     0       0         0    ...            0     1      0         P
```

Shuffle
Why must we shuffle in this case?

```
343## SHUFFLE the DATAFRAME
344## df = df.sample(frac=1).reset_index(drop=True)
345## Here, specifying drop=True prevents .reset_index
346## from creating a column containing the old index entries.
347## the frac=1 means "resample" (shuffle) 100% of the data
348result=result.sample(frac=1).reset_index(drop=True)
349print(result)
350## This worked! You can see that the shape is the same
351## and that the label is no longer all N and then all P
352
353
354
355
356
357
358
359
360
```

File explorer | Variable explorer | Help

IPython console

Console 1/A

```
[10 rows x 1893 columns]
  PosORNeg  abberline  able  ably  ...    yes  yglesias  york  young
0        N        0.0   0.0   0.0  ...    0.0        0.0   0.0      1
1        P        0.0   0.0   0.0  ...    0.0        0.0   0.0      0
2        N        0.0   1.0   0.0  ...    0.0        0.0   0.0      0
3        N        0.0   0.0   0.0  ...    0.0        0.0   0.0      0
4        P        0.0   0.0   0.0  ...    0.0        0.0   1.0      0
5        N        0.0   0.0   0.0  ...    0.0        0.0   0.0      0
6        N        0.0   0.0   0.0  ...    1.0        0.0   0.0      0
7        P        0.0   0.0   0.0  ...    0.0        0.0   1.0      1
8        P        2.0   0.0   1.0  ...    0.0        1.0   0.0      0
9        P        0.0   0.0   0.0  ...    0.0        0.0   0.0      0
```

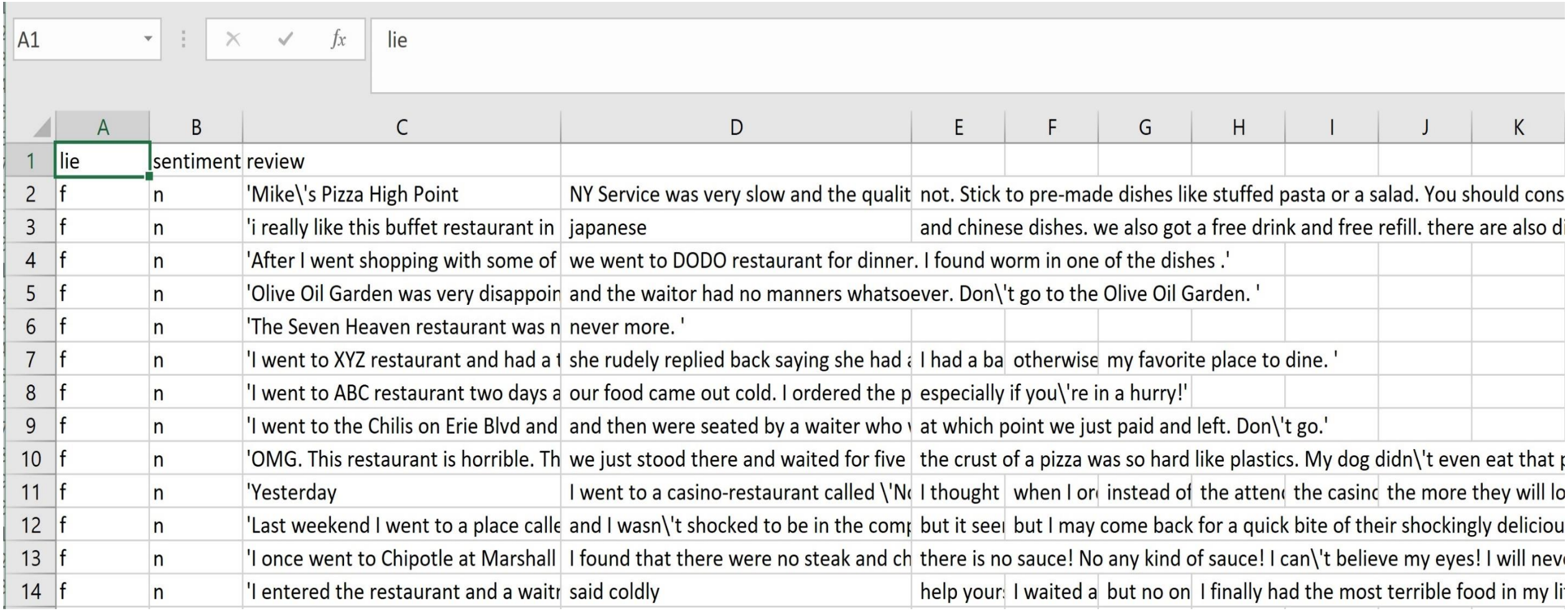
Test & Train Sets

```
345
346
347
348
349
350
351
352
353## From here, we can create (randomly) the test and train sets
354# make results reproducible
355import numpy as np
356np.random.seed(140) ## or any number - does not matter
357# sample without replacement
358## I am choosing "6" here to make the training set
359## of size 6. This will make the test set of size 4
360## This can be any choice depending on YOU and your data
361train_ix = np.random.choice(result.index, 6, replace=False)
362df_training = result.iloc[train_ix]
363df_test = result.drop(train_ix)
364print("Training set...")
365print(df_training)
366print("Testing set...")
367print(df_test)
```

Console 1/A

```
[10 rows x 1893 columns]
Training set...
  PosORNeg  abberline  able  ably  ...    yes  yglesias  york  young
3        N        0.0   0.0   0.0  ...    0.0       0.0   0.0      0
2        N        0.0   1.0   0.0  ...    0.0       0.0   0.0      0
1        P        0.0   0.0   0.0  ...    0.0       0.0   0.0      0
4        P        0.0   0.0   0.0  ...    0.0       0.0   1.0      0
0        N        0.0   0.0   0.0  ...    0.0       0.0   0.0      1
5        N        0.0   0.0   0.0  ...    0.0       0.0   0.0      0

[6 rows x 1893 columns]
Testing set...
  PosORNeg  abberline  able  ably  ...    yes  yglesias  york  young
6        N        0.0   0.0   0.0  ...    1.0       0.0   0.0      0
7        P        0.0   0.0   0.0  ...    0.0       0.0   1.0      1
8        P        2.0   0.0   1.0  ...    0.0       1.0   0.0      0
9        P        0.0   0.0   0.0  ...    0.0       0.0   0.0      0

[4 rows x 1893 columns]
```

Example 3: Reading in one .csv file…
More on this in the next ppt…



A1 | lie

	A	B	C	D	E	F	G	H	I	J	K
1	lie	sentiment	review								
2	f	n	'Mike\'s Pizza High Point	NY Service was very slow and the qualit	not. Stick to pre-made dishes like stuffed pasta or a salad. You should cons						
3	f	n	'i really like this buffet restaurant in	japanese	and chinese dishes. we also got a free drink and free refill. there are also di						
4	f	n	'After I went shopping with some of	we went to DODO restaurant for dinner. I found worm in one of the dishes .'							
5	f	n	'Olive Oil Garden was very disappoin	and the waitor had no manners whatsoever. Don\'t go to the Olive Oil Garden. '							
6	f	n	'The Seven Heaven restaurant was n	never more. '							
7	f	n	'I went to XYZ restaurant and had a t	she rudely replied back saying she had a	I had a ba	otherwise	my favorite place to dine. '				
8	f	n	'I went to ABC restaurant two days a	our food came out cold. I ordered the p	especially if you\'re in a hurry!'						
9	f	n	'I went to the Chilis on Erie Blvd and	and then were seated by a waiter who v	at which point we just paid and left. Don\'t go.'						
10	f	n	'OMG. This restaurant is horrible. Th	we just stood there and waited for five	the crust of a pizza was so hard like plastics. My dog didn\'t even eat that p						
11	f	n	'Yesterday	I went to a casino-restaurant called \'N	I thought	when I or	instead of	the atten	the casinc	the more they will lo	
12	f	n	'Last weekend I went to a place calle	and I wasn\'t shocked to be in the comp	but it seer	but I may come back for a quick bite of their shockingly deliciou					
13	f	n	'I once went to Chipotle at Marshall	I found that there were no steak and ch	there is no sauce! No any kind of sauce! I can\'t believe my eyes! I will nev						
14	f	n	'I entered the restaurant and a waitr	said coldly	help your	I waited a	but no on	I finally had the most terrible food in my li			