UCLA QCBio

Collaboratory

Intermediate/Advanced Python

Michael Weinstein
(Day 3)

π

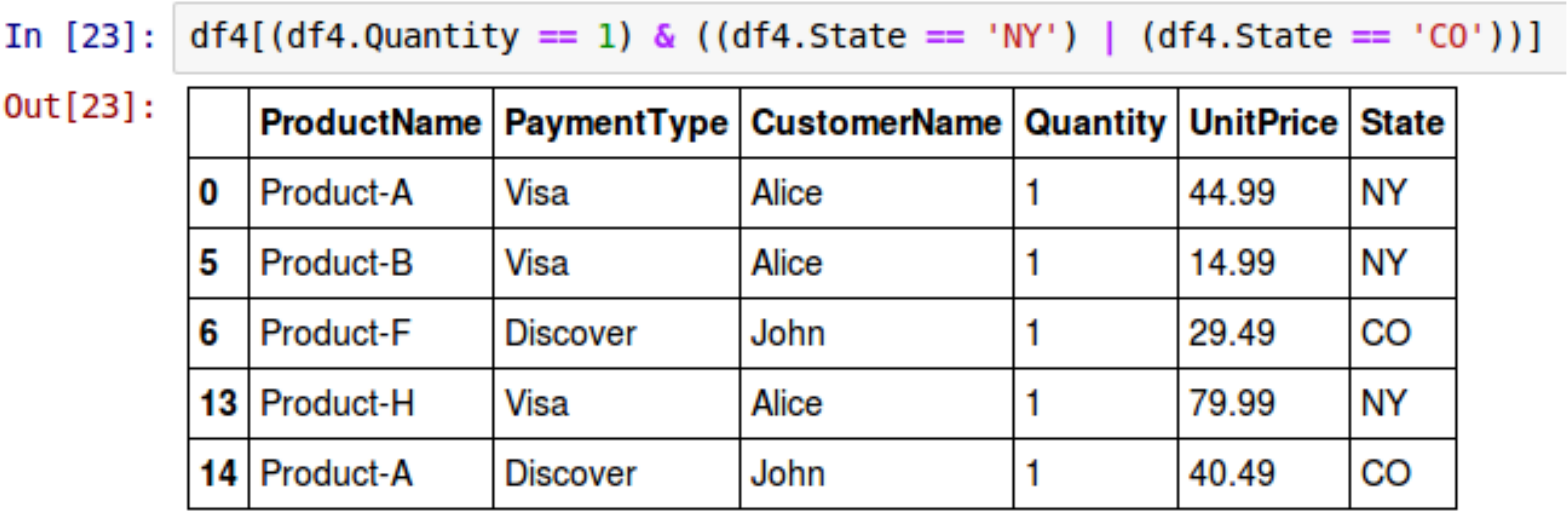
Topics

- Review of basic data structures
- Accessing and working with objects in python
- Numpy
- Pandas
  - What are dataframes?
  - Sample column counting operation
  - Test, refine, test, refine, test, refine to get the data structure we want
  - Extracting a column
- Matplotlib
  - Making a simple line plot
- Scipy

Next goal: Analyze base frequency at each position in the first 50 bases

- Need to iterate over sam file again
- Need to grab the first 50 bases of each read
- Need to turn the string into a list
- Need to move the data to a pandas dataframe
  - This is a more flexible structure for holding heterogenous data
  - Not as efficient as a numpy array, but still very fast
  - Very similar to dataframes in R language
- Need to count base occurrences in each column
- New file: baseFrequencyCounter.py

How to think of a dataframe?

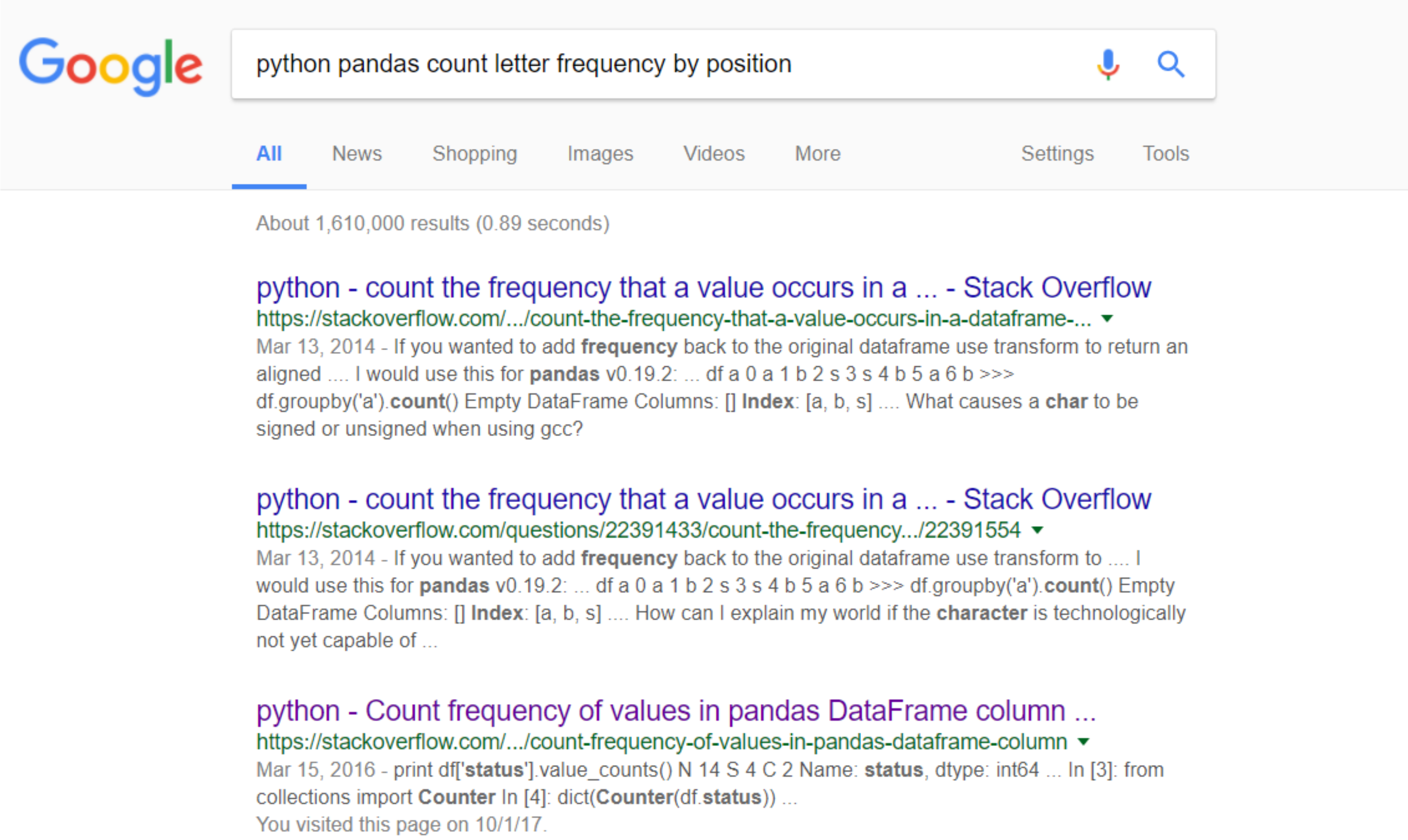


```
In [23]: df4[(df4.Quantity == 1) & ((df4.State == 'NY') | (df4.State == 'CO'))]
```

Out[23]:

	ProductName	PaymentType	CustomerName	Quantity	UnitPrice	State
0	Product-A	Visa	Alice	1	44.99	NY
5	Product-B	Visa	Alice	1	14.99	NY
6	Product-F	Discover	John	1	29.49	CO
13	Product-H	Visa	Alice	1	79.99	NY
14	Product-A	Discover	John	1	40.49	CO

How to count letters in a column?



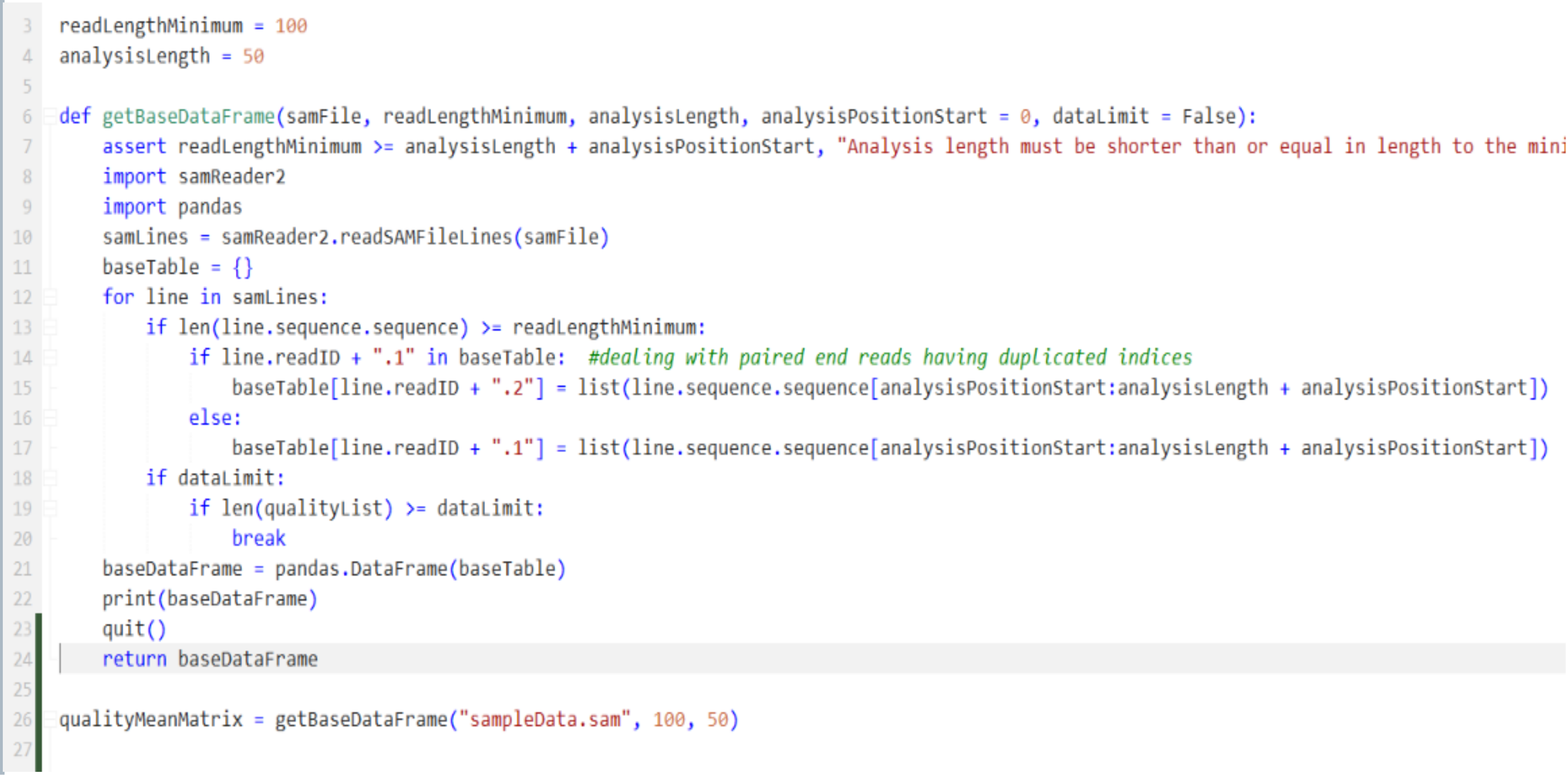
About 1,610,000 results (0.89 seconds)

python - count the frequency that a value occurs in a ... - Stack Overflow
https://stackoverflow.com/.../count-the-frequency-that-a-value-occurs-in-a-dataframe-...
Mar 13, 2014 - If you wanted to add **frequency** back to the original dataframe use transform to return an aligned .... I would use this for **pandas** v0.19.2: ... df a 0 a 1 b 2 s 3 s 4 b 5 a 6 b >>> df.groupby('a').**count**() Empty DataFrame Columns: [] **Index**: [a, b, s] .... What causes a **char** to be signed or unsigned when using gcc?

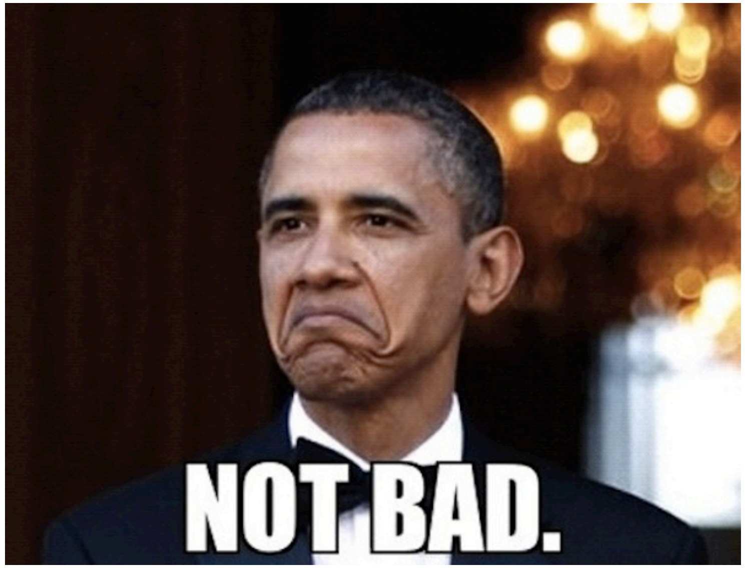
python - count the frequency that a value occurs in a ... - Stack Overflow
https://stackoverflow.com/questions/22391433/count-the-frequency.../22391554
Mar 13, 2014 - If you wanted to add **frequency** back to the original dataframe use transform to .... I would use this for **pandas** v0.19.2: ... df a 0 a 1 b 2 s 3 s 4 b 5 a 6 b >>> df.groupby('a').**count**() Empty DataFrame Columns: [] **Index**: [a, b, s] .... How can I explain my world if the **character** is technologically not yet capable of ...

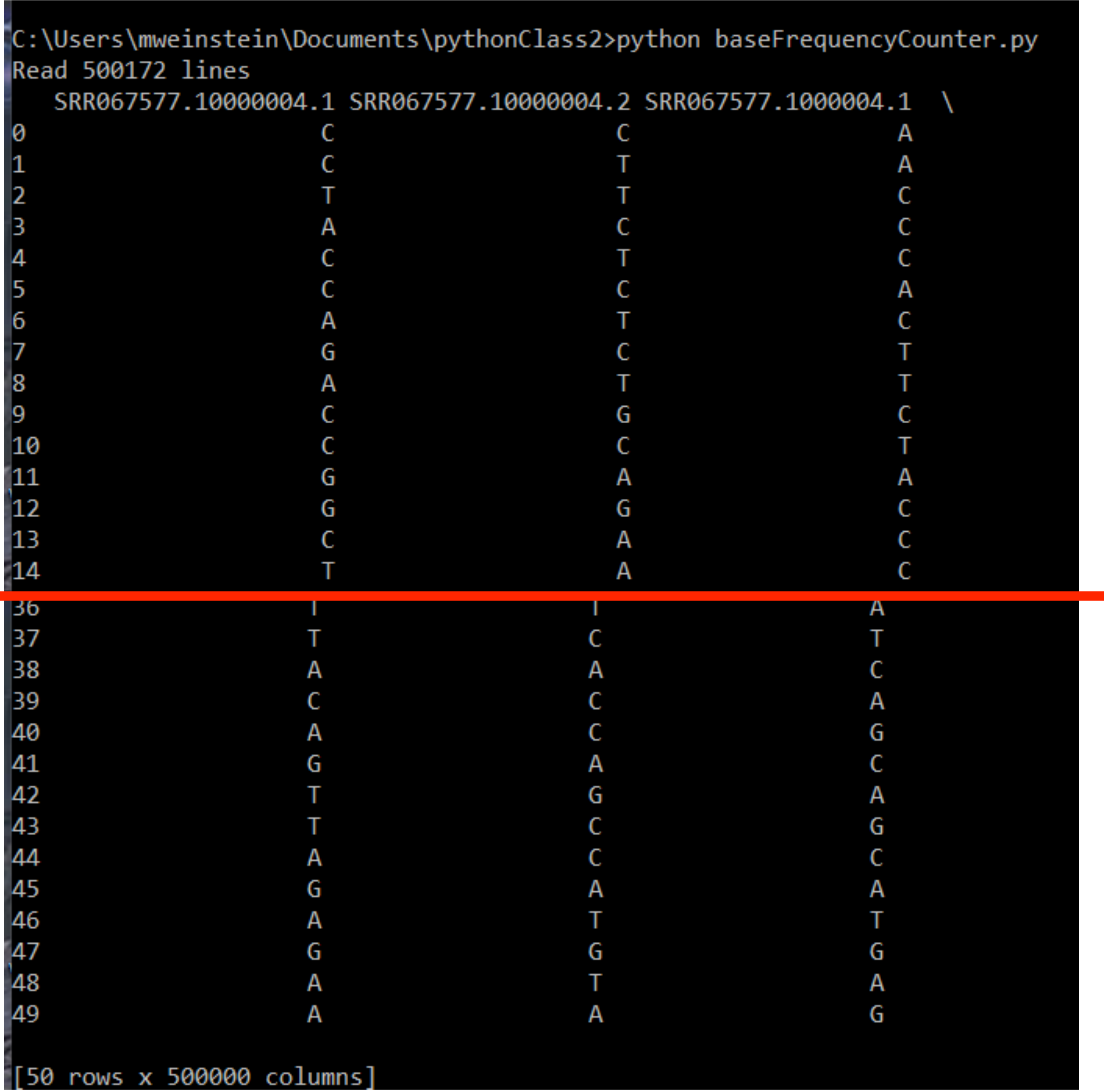
python - Count frequency of values in pandas DataFrame column ...
https://stackoverflow.com/.../count-frequency-of-values-in-pandas-dataframe-column
Mar 15, 2016 - print df['**status**'].value_counts() N 14 S 4 C 2 Name: **status**, dtype: int64 ... In [3]: from collections import **Counter** In [4]: dict(**Counter**(df.**status**)) ...
You visited this page on 10/1/17.

Some test code



```python
readLengthMinimum = 100
analysisLength = 50

def getBaseDataFrame(samFile, readLengthMinimum, analysisLength, analysisPositionStart = 0, dataLimit = False):
    assert readLengthMinimum >= analysisLength + analysisPositionStart, "Analysis length must be shorter than or equal in length to the mini
    import samReader2
    import pandas
    samLines = samReader2.readSAMFileLines(samFile)
    baseTable = {}
    for line in samLines:
        if len(line.sequence.sequence) >= readLengthMinimum:
            if line.readID + ".1" in baseTable:  #dealing with paired end reads having duplicated indices
                baseTable[line.readID + ".2"] = list(line.sequence.sequence[analysisPositionStart:analysisLength + analysisPositionStart])
            else:
                baseTable[line.readID + ".1"] = list(line.sequence.sequence[analysisPositionStart:analysisLength + analysisPositionStart])
        if dataLimit:
            if len(qualityList) >= dataLimit:
                break
    baseDataFrame = pandas.DataFrame(baseTable)
    print(baseDataFrame)
    quit()
    return baseDataFrame

qualityMeanMatrix = getBaseDataFrame("sampleData.sam", 100, 50)
```

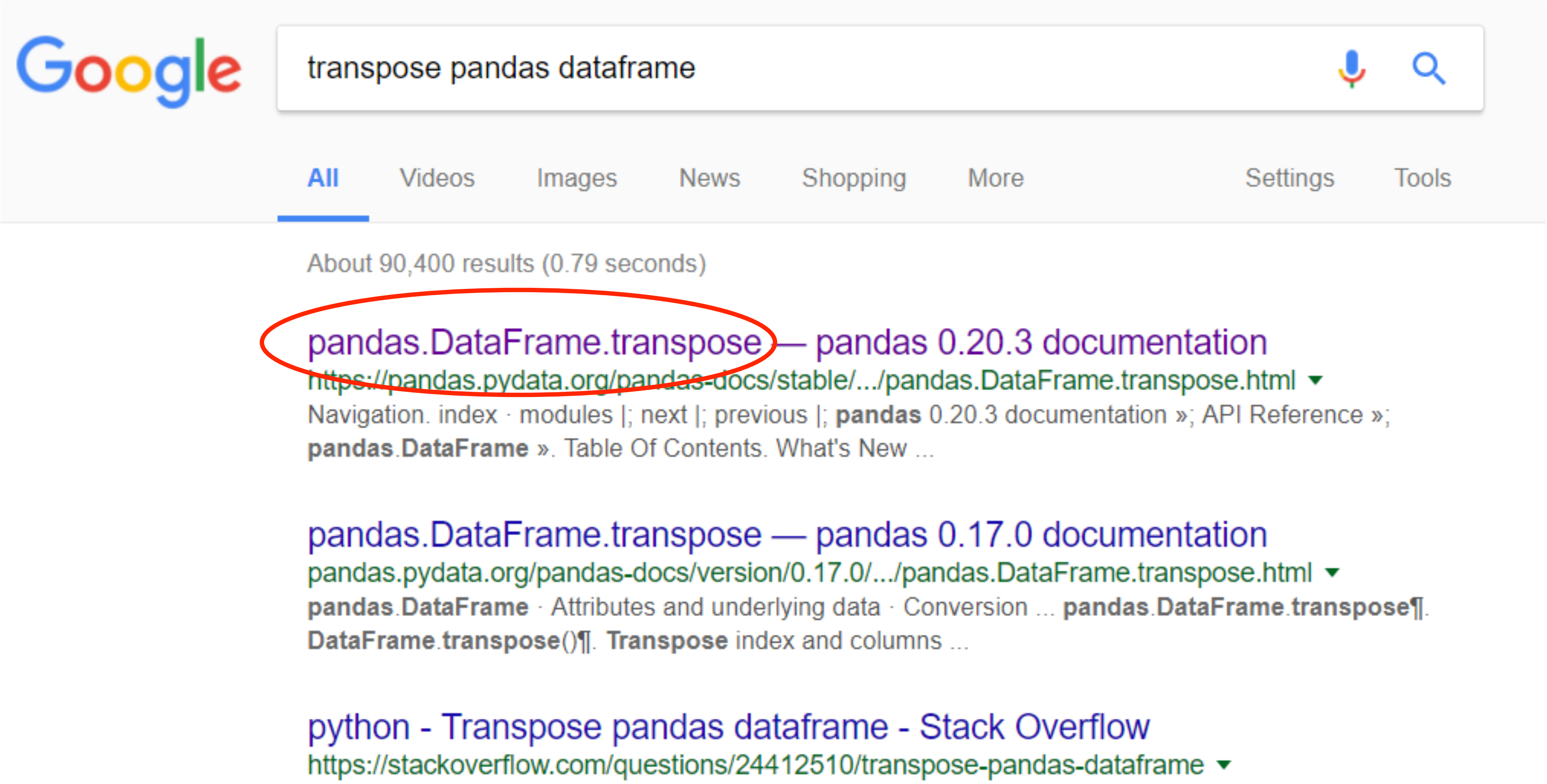




```
C:\Users\mweinstein\Documents\pythonClass2>python baseFrequencyCounter.py
Read 500172 lines
   SRR067577.10000004.1 SRR067577.10000004.2 SRR067577.1000004.1  \
0                     C                    C                   A
1                     C                    T                   A
2                     T                    T                   C
3                     A                    C                   C
4                     C                    T                   C
5                     C                    C                   A
6                     A                    T                   C
7                     G                    C                   T
8                     A                    T                   T
9                     C                    G                   C
10                    C                    C                   T
11                    G                    A                   A
12                    G                    G                   C
13                    C                    A                   C
14                    T                    A                   C
36                    T                    T                   A
37                    T                    C                   T
38                    A                    A                   C
39                    C                    C                   A
40                    A                    C                   G
41                    G                    A                   C
42                    T                    G                   A
43                    T                    C                   G
44                    A                    C                   C
45                    G                    A                   A
46                    A                    T                   T
47                    G                    G                   G
48                    A                    T                   A
49                    A                    A                   G

[50 rows x 500000 columns]
```

Data looks almost right, except we need it transposed (we want position to be the columns).

How to transpose



transpose pandas dataframe

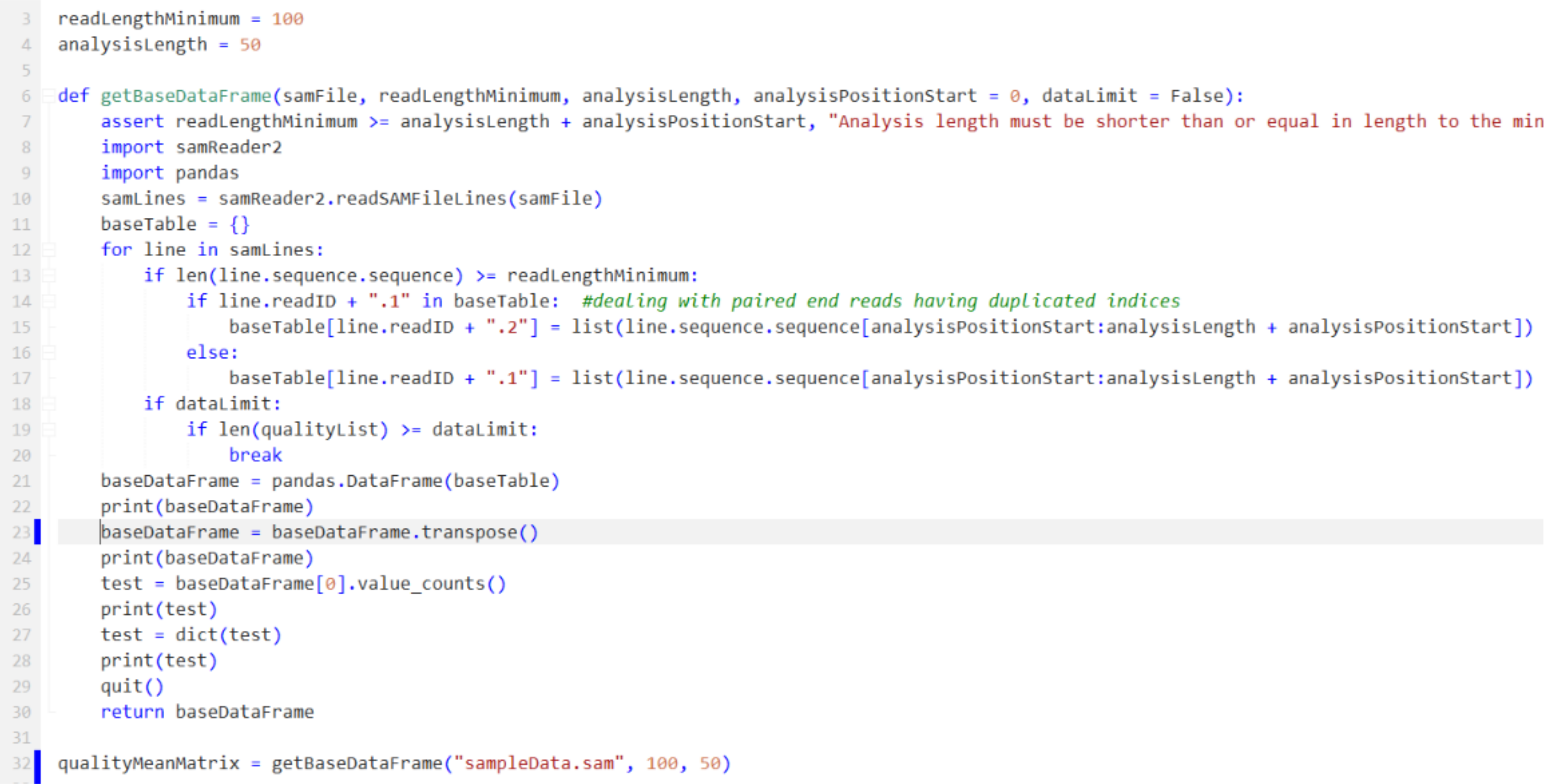
About 90,400 results (0.79 seconds)

pandas.DataFrame.transpose — pandas 0.20.3 documentation
https://pandas.pydata.org/pandas-docs/stable/.../pandas.DataFrame.transpose.html
Navigation. index · modules |; next |; previous |; **pandas** 0.20.3 documentation »; API Reference »; **pandas.DataFrame** ». Table Of Contents. What's New ...

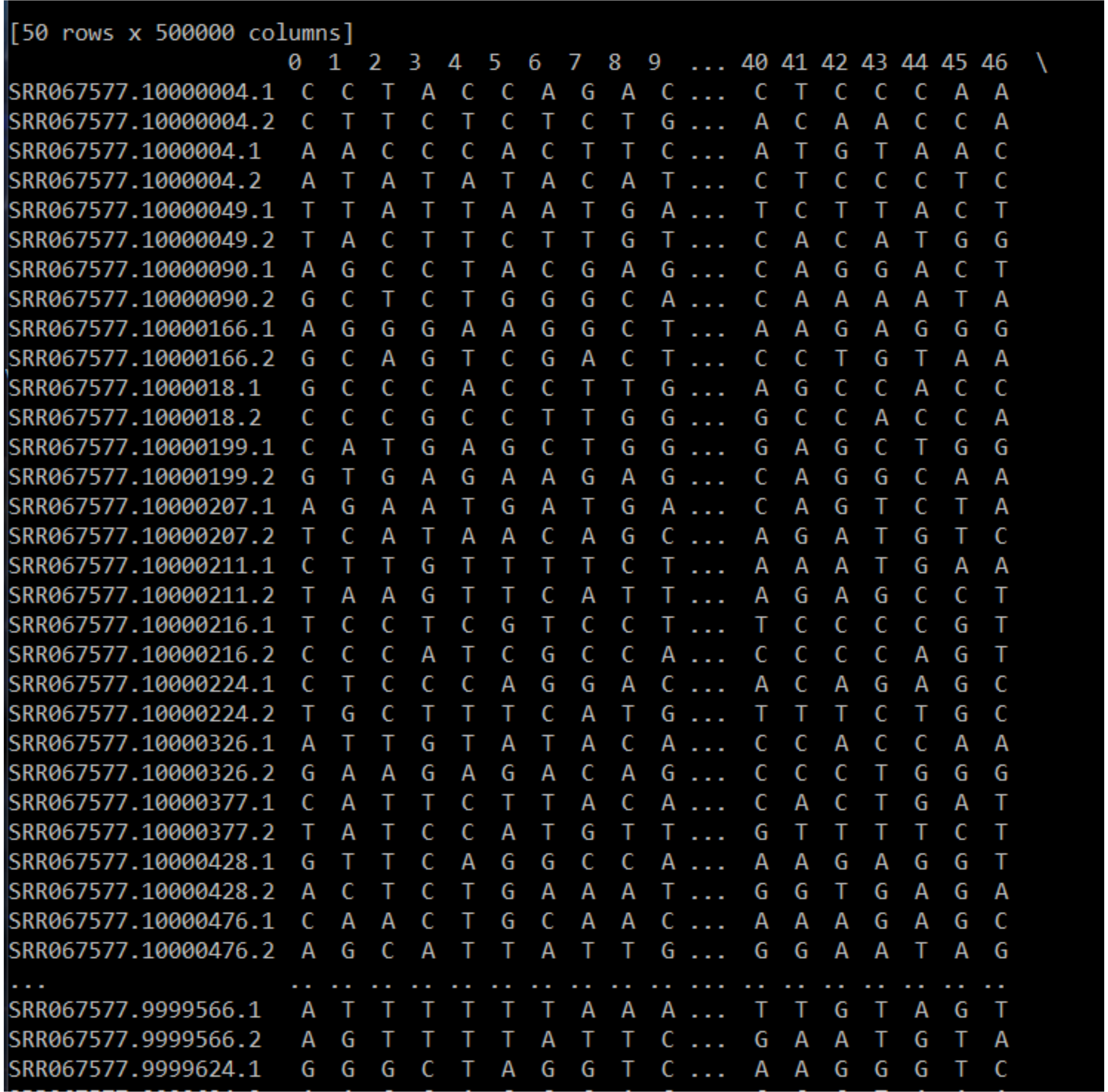
pandas.DataFrame.transpose — pandas 0.17.0 documentation
pandas.pydata.org/pandas-docs/version/0.17.0/.../pandas.DataFrame.transpose.html
pandas.DataFrame · Attributes and underlying data · Conversion ... **pandas.DataFrame.transpose**¶. **DataFrame.transpose**()¶. **Transpose** index and columns ...

python - Transpose pandas dataframe - Stack Overflow
https://stackoverflow.com/questions/24412510/transpose-pandas-dataframe

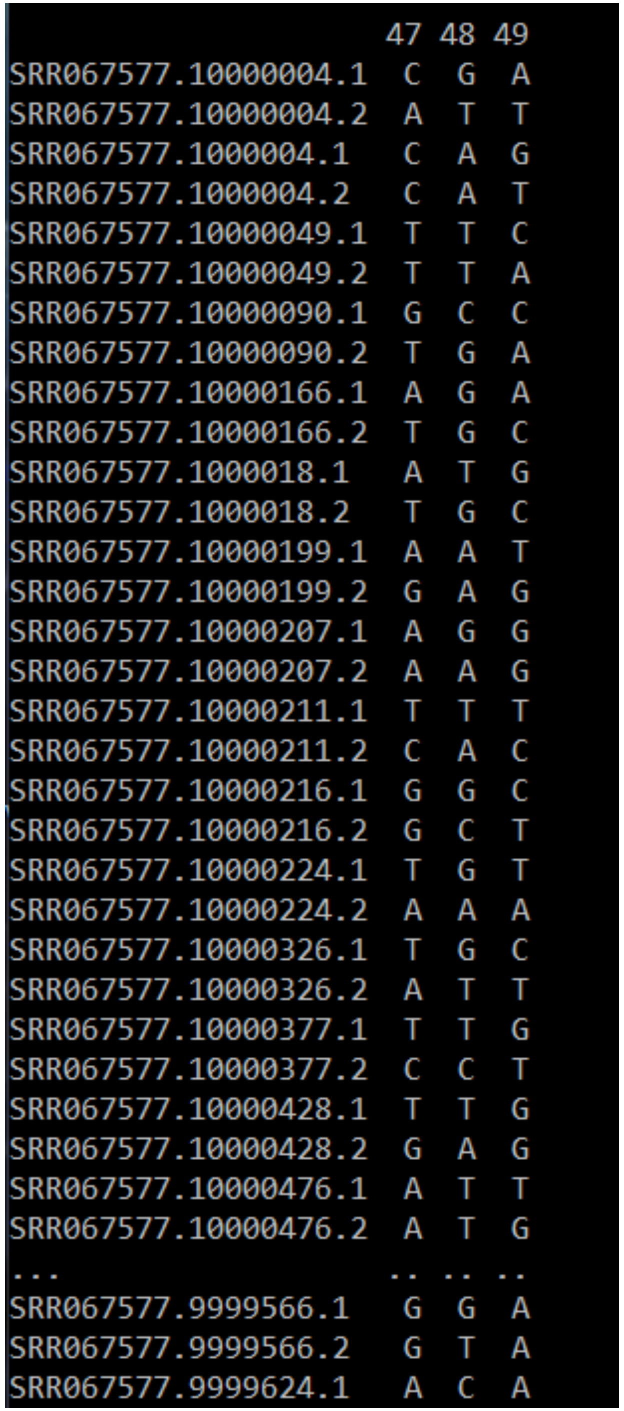
Testing our data structure



```
readLengthMinimum = 100
analysisLength = 50

def getBaseDataFrame(samFile, readLengthMinimum, analysisLength, analysisPositionStart = 0, dataLimit = False):
    assert readLengthMinimum >= analysisLength + analysisPositionStart, "Analysis length must be shorter than or equal in length to the min
    import samReader2
    import pandas
    samLines = samReader2.readSAMFileLines(samFile)
    baseTable = {}
    for line in samLines:
        if len(line.sequence.sequence) >= readLengthMinimum:
            if line.readID + ".1" in baseTable:  #dealing with paired end reads having duplicated indices
                baseTable[line.readID + ".2"] = list(line.sequence.sequence[analysisPositionStart:analysisLength + analysisPositionStart])
            else:
                baseTable[line.readID + ".1"] = list(line.sequence.sequence[analysisPositionStart:analysisLength + analysisPositionStart])
        if dataLimit:
            if len(qualityList) >= dataLimit:
                break
    baseDataFrame = pandas.DataFrame(baseTable)
    print(baseDataFrame)
    baseDataFrame = baseDataFrame.transpose()
    print(baseDataFrame)
    test = baseDataFrame[0].value_counts()
    print(test)
    test = dict(test)
    print(test)
    quit()
    return baseDataFrame

qualityMeanMatrix = getBaseDataFrame("sampleData.sam", 100, 50)
```

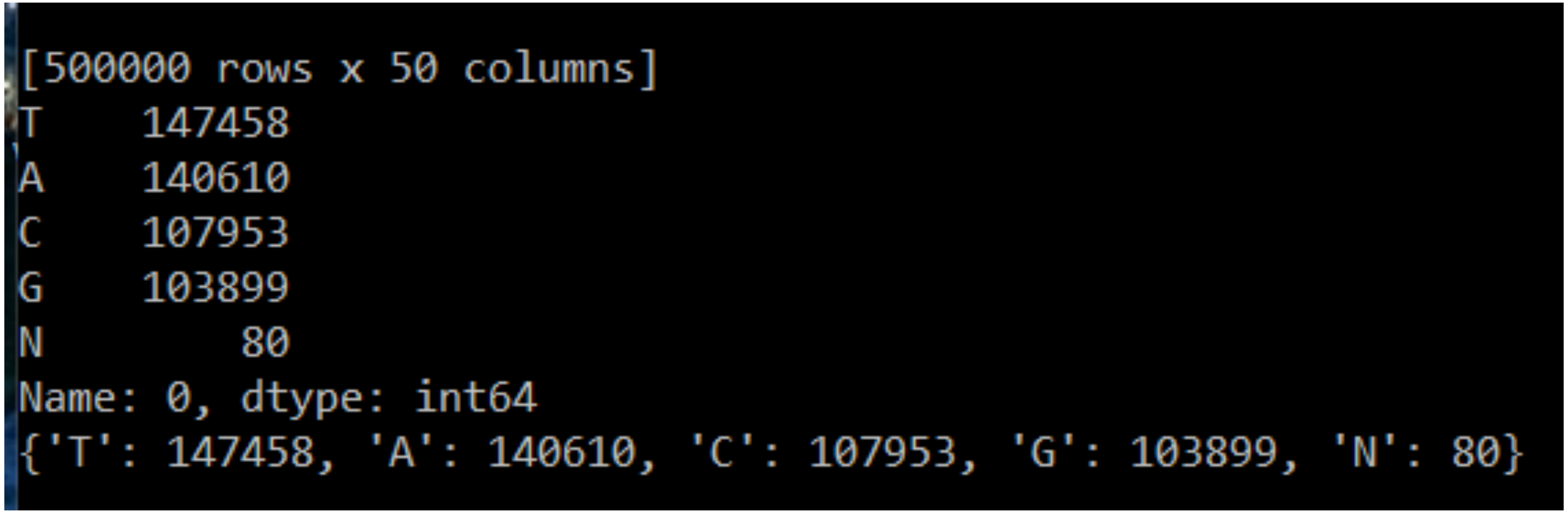


```
[50 rows x 500000 columns]
                      0  1  2  3  4  5  6  7  8  9 ... 40 41 42 43 44 45 46  \
SRR067577.10000004.1  C  C  T  A  C  C  A  G  A  C ...  C  T  C  C  C  A  A
SRR067577.10000004.2  C  T  T  C  T  C  T  C  T  G ...  A  C  A  A  C  C  A
SRR067577.1000004.1   A  A  C  C  C  A  C  T  T  C ...  A  T  G  T  A  A  C
SRR067577.1000004.2   A  T  A  T  A  T  A  C  A  T ...  C  T  C  C  C  T  C
SRR067577.10000049.1  T  T  A  T  T  A  A  T  G  A ...  T  C  T  T  A  C  T
SRR067577.10000049.2  T  A  C  T  T  C  T  T  G  T ...  C  A  C  A  T  G  G
SRR067577.10000090.1  A  G  C  C  T  A  C  G  A  G ...  C  A  G  G  A  C  T
SRR067577.10000090.2  G  C  T  C  T  G  G  G  C  A ...  C  A  A  A  A  T  A
SRR067577.10000166.1  A  G  G  G  A  A  G  G  C  T ...  A  A  G  A  G  G  G
SRR067577.10000166.2  G  C  A  G  T  C  G  A  C  T ...  C  C  T  G  T  A  A
SRR067577.1000018.1   G  C  C  C  A  C  C  T  T  G ...  A  G  C  C  A  C  C
SRR067577.1000018.2   C  C  C  G  C  C  T  T  G  G ...  G  C  C  A  C  C  A
SRR067577.10000199.1  C  A  T  G  A  G  C  T  G  G ...  G  A  G  C  T  G  G
SRR067577.10000199.2  G  T  G  A  G  A  A  G  A  G ...  C  A  G  G  C  A  A
SRR067577.10000207.1  A  G  A  A  T  G  A  T  G  A ...  C  A  G  T  C  T  A
SRR067577.10000207.2  T  C  A  T  A  A  C  A  G  C ...  A  G  A  T  G  T  C
SRR067577.10000211.1  C  T  T  G  T  T  T  T  C  T ...  A  A  A  T  G  A  A
SRR067577.10000211.2  T  A  A  G  T  T  C  A  T  T ...  A  G  A  G  C  C  T
SRR067577.10000216.1  T  C  C  T  C  G  T  C  C  T ...  T  C  C  C  C  G  T
SRR067577.10000216.2  C  C  C  A  T  C  G  C  C  A ...  C  C  C  C  A  G  T
SRR067577.10000224.1  C  T  C  C  C  A  G  G  A  C ...  A  C  A  G  A  G  C
SRR067577.10000224.2  T  G  C  T  T  T  C  A  T  G ...  T  T  T  C  T  G  C
SRR067577.10000326.1  A  T  T  G  T  A  T  A  C  A ...  C  C  A  C  C  A  A
SRR067577.10000326.2  G  A  A  G  A  G  A  C  A  G ...  C  C  C  T  G  G  G
SRR067577.10000377.1  C  A  T  T  C  T  T  A  C  A ...  C  A  C  T  G  A  T
SRR067577.10000377.2  T  A  T  C  C  A  T  G  T  T ...  G  T  T  T  T  C  T
SRR067577.10000428.1  G  T  T  C  A  G  G  C  C  A ...  A  A  G  A  G  G  T
SRR067577.10000428.2  A  C  T  C  T  G  A  A  A  T ...  G  G  T  G  A  G  A
SRR067577.10000476.1  C  A  A  C  T  G  C  A  A  C ...  A  A  A  G  A  G  C
SRR067577.10000476.2  A  G  C  A  T  T  A  T  T  G ...  G  G  A  A  T  A  G
...                  .. .. .. .. .. .. .. .. .. .. ... .. .. .. .. .. .. ..
SRR067577.9999566.1   A  T  T  T  T  T  T  A  A  A ...  T  T  G  T  A  G  T
SRR067577.9999566.2   A  G  T  T  T  T  A  T  T  C ...  G  A  A  T  G  T  A
SRR067577.9999624.1   G  G  G  C  T  A  G  G  T  C ...  A  A  G  G  G  T  C
```



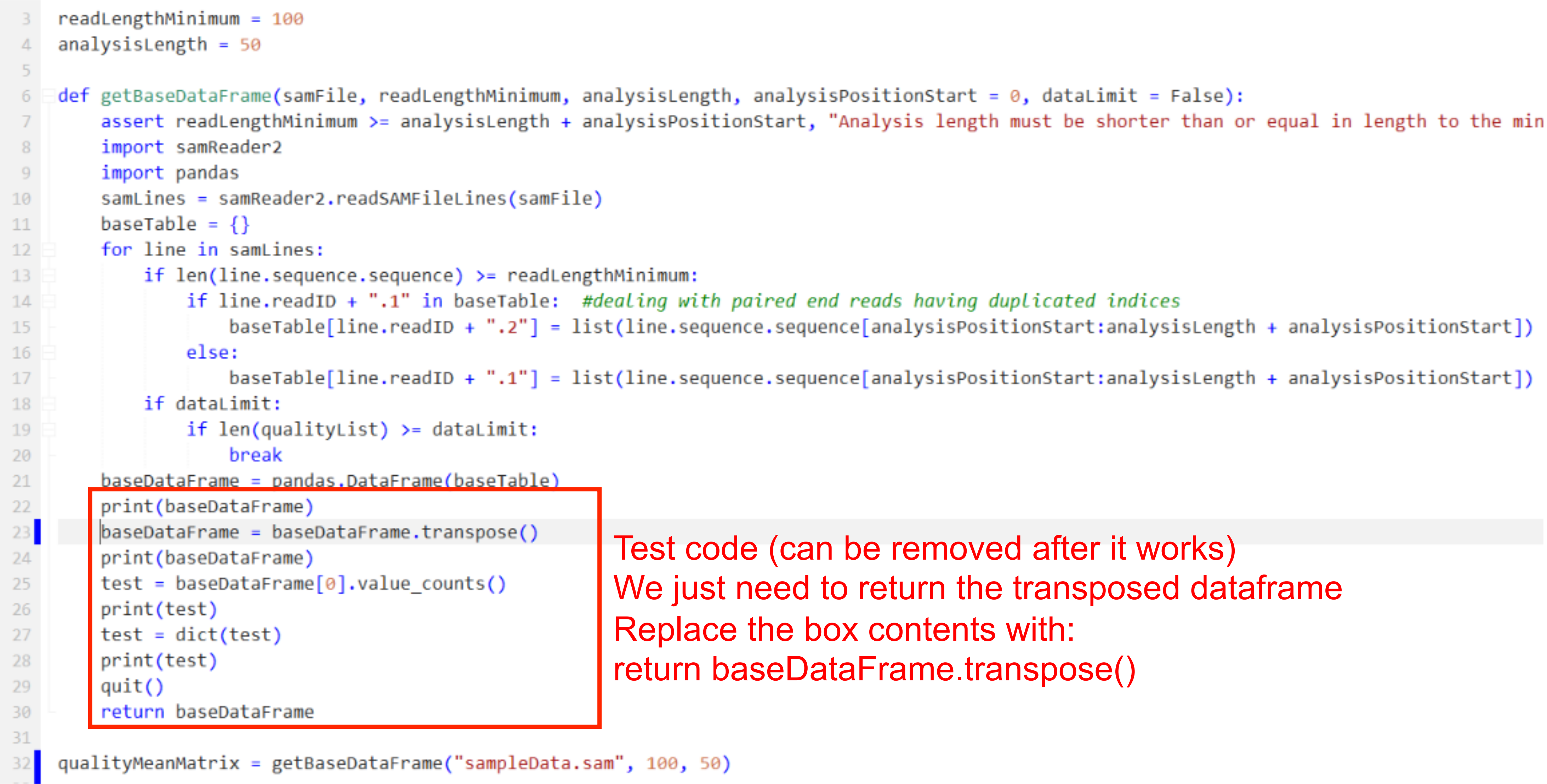
```
                     47 48 49
SRR067577.10000004.1  C  G  A
SRR067577.10000004.2  A  T  T
SRR067577.1000004.1   C  A  G
SRR067577.1000004.2   C  A  T
SRR067577.10000049.1  T  T  C
SRR067577.10000049.2  T  T  A
SRR067577.10000090.1  G  C  C
SRR067577.10000090.2  T  G  A
SRR067577.10000166.1  A  G  A
SRR067577.10000166.2  T  G  C
SRR067577.1000018.1   A  T  G
SRR067577.1000018.2   T  G  C
SRR067577.10000199.1  A  A  T
SRR067577.10000199.2  G  A  G
SRR067577.10000207.1  A  G  G
SRR067577.10000207.2  A  A  G
SRR067577.10000211.1  T  T  T
SRR067577.10000211.2  C  A  C
SRR067577.10000216.1  G  G  C
SRR067577.10000216.2  G  C  T
SRR067577.10000224.1  T  G  T
SRR067577.10000224.2  A  A  A
SRR067577.10000326.1  T  G  C
SRR067577.10000326.2  A  T  T
SRR067577.10000377.1  T  T  G
SRR067577.10000377.2  C  C  T
SRR067577.10000428.1  T  T  G
SRR067577.10000428.2  G  A  G
SRR067577.10000476.1  A  T  T
SRR067577.10000476.2  A  T  G
...                  .. .. ..
SRR067577.9999566.1   G  G  A
SRR067577.9999566.2   G  T  A
SRR067577.9999624.1   A  C  A
```

Looks good!



```
[500000 rows x 50 columns]
T    147458
A    140610
C    107953
G    103899
N        80
Name: 0, dtype: int64
{'T': 147458, 'A': 140610, 'C': 107953, 'G': 103899, 'N': 80}
```

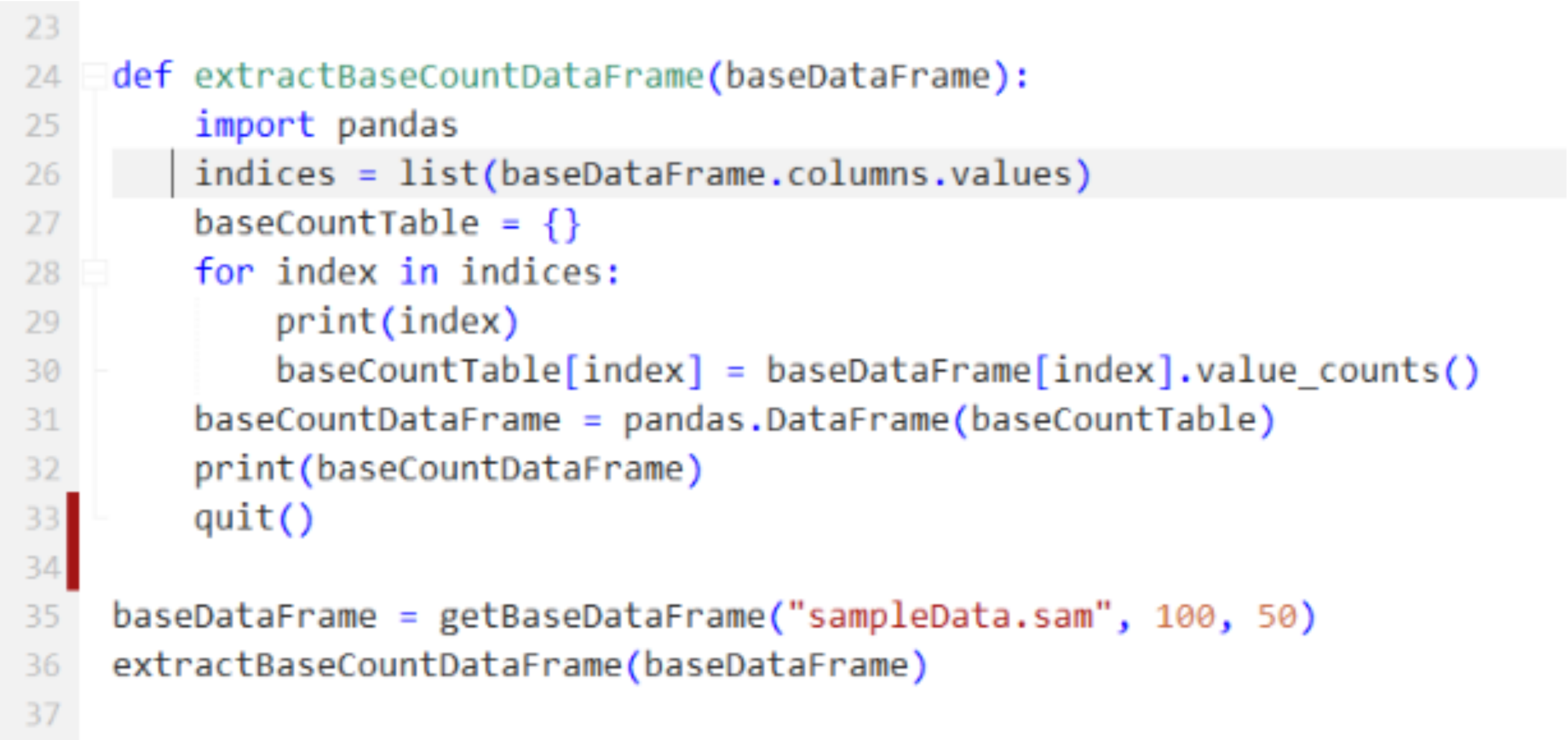
Testing our data structure



```
readLengthMinimum = 100
analysisLength = 50

def getBaseDataFrame(samFile, readLengthMinimum, analysisLength, analysisPositionStart = 0, dataLimit = False):
    assert readLengthMinimum >= analysisLength + analysisPositionStart, "Analysis length must be shorter than or equal in length to the min
    import samReader2
    import pandas
    samLines = samReader2.readSAMFileLines(samFile)
    baseTable = {}
    for line in samLines:
        if len(line.sequence.sequence) >= readLengthMinimum:
            if line.readID + ".1" in baseTable:  #dealing with paired end reads having duplicated indices
                baseTable[line.readID + ".2"] = list(line.sequence.sequence[analysisPositionStart:analysisLength + analysisPositionStart])
            else:
                baseTable[line.readID + ".1"] = list(line.sequence.sequence[analysisPositionStart:analysisLength + analysisPositionStart])
        if dataLimit:
            if len(qualityList) >= dataLimit:
                break
    baseDataFrame = pandas.DataFrame(baseTable)
    print(baseDataFrame)
    baseDataFrame = baseDataFrame.transpose()
    print(baseDataFrame)
    test = baseDataFrame[0].value_counts()
    print(test)
    test = dict(test)
    print(test)
    quit()
    return baseDataFrame

qualityMeanMatrix = getBaseDataFrame("sampleData.sam", 100, 50)
```

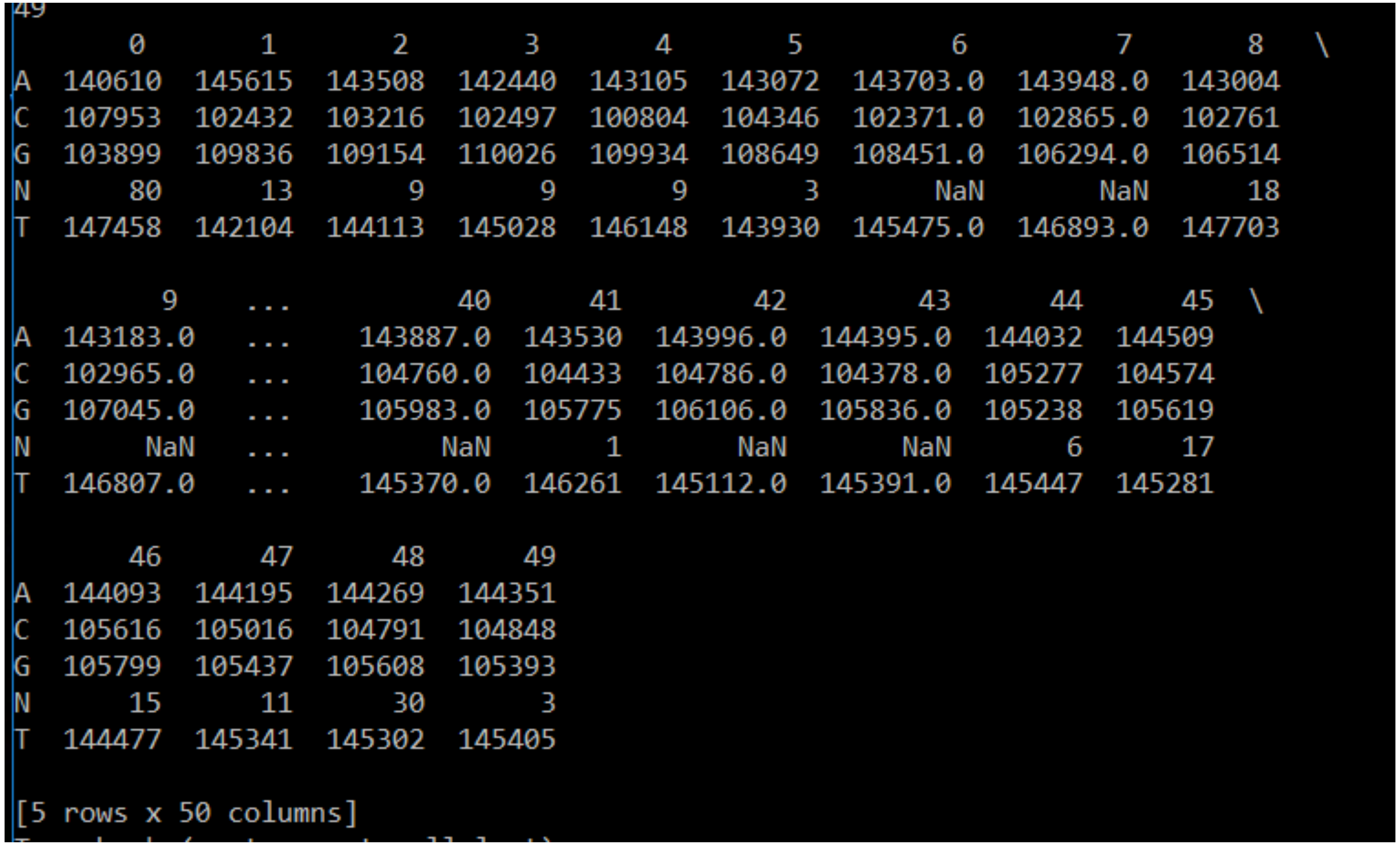
Test code (can be removed after it works)
We just need to return the transposed dataframe
Replace the box contents with:
return baseDataFrame.transpose()

Testing out our next data structure



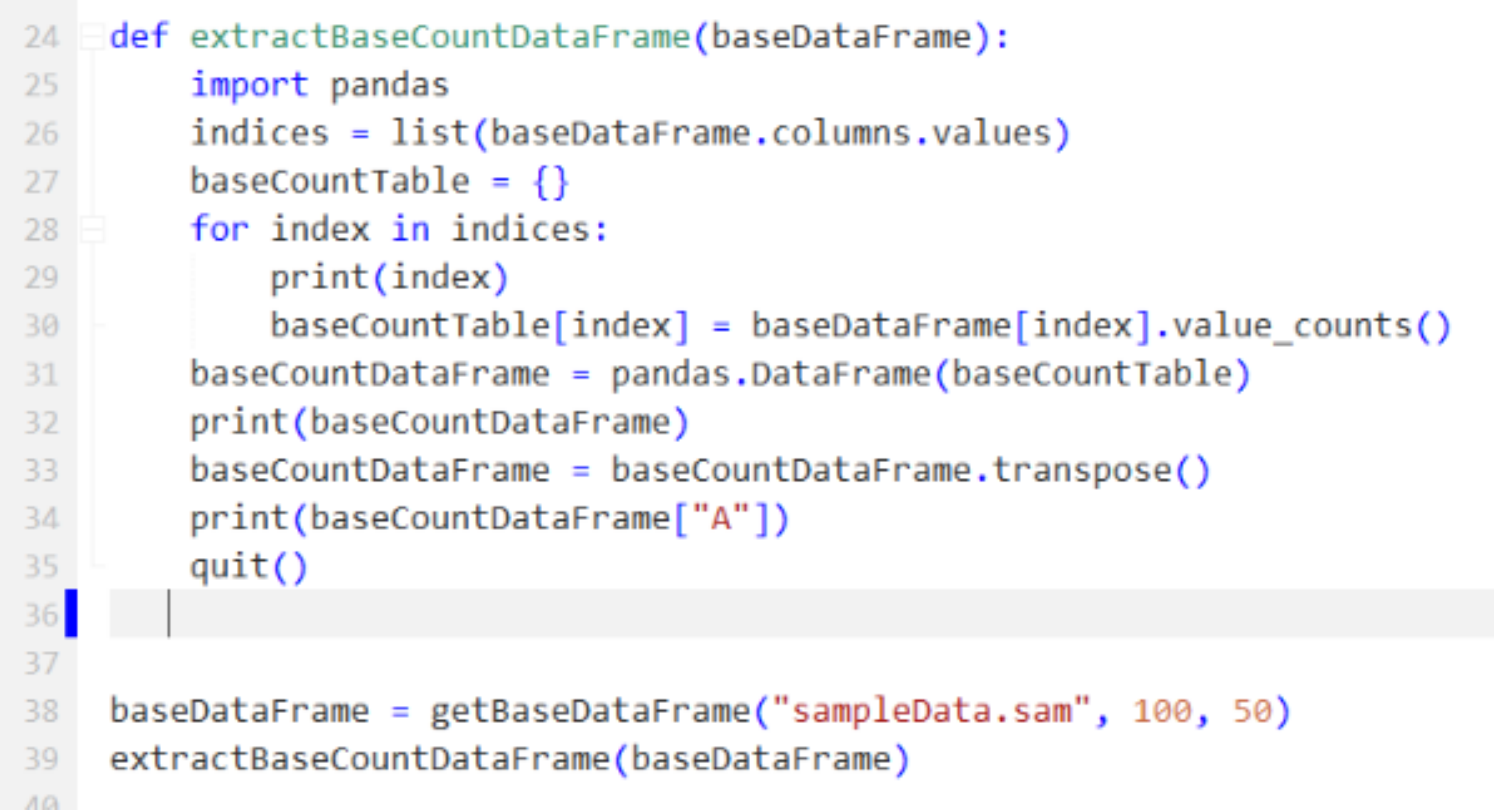
```python

def extractBaseCountDataFrame(baseDataFrame):
    import pandas
    indices = list(baseDataFrame.columns.values)
    baseCountTable = {}
    for index in indices:
        print(index)
        baseCountTable[index] = baseDataFrame[index].value_counts()
    baseCountDataFrame = pandas.DataFrame(baseCountTable)
    print(baseCountDataFrame)
    quit()

baseDataFrame = getBaseDataFrame("sampleData.sam", 100, 50)
extractBaseCountDataFrame(baseDataFrame)

```

Testing out the data structure



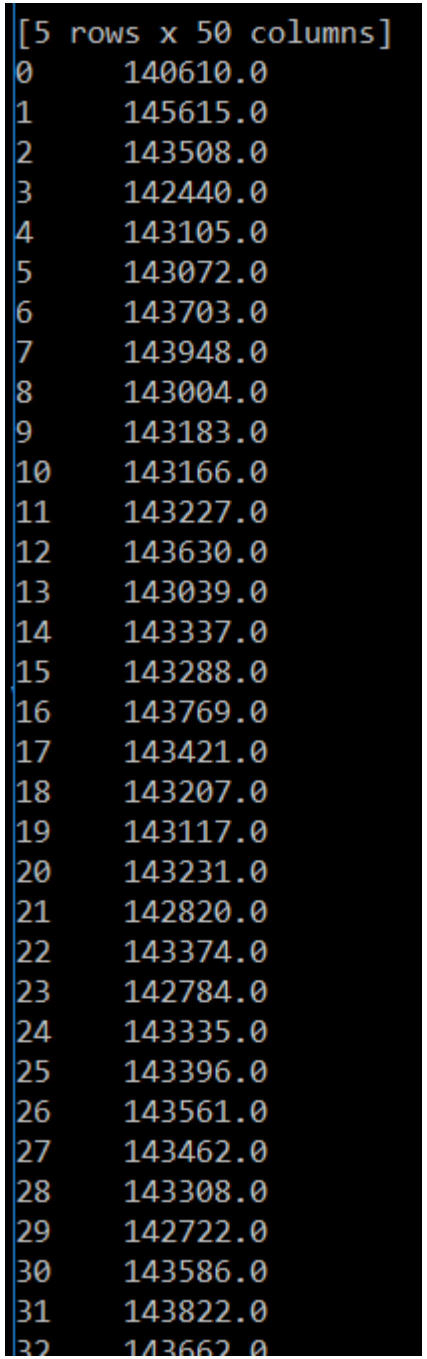
```
49
        0       1       2       3       4       5         6         7       8  \
A  140610  145615  143508  142440  143105  143072  143703.0  143948.0  143004
C  107953  102432  103216  102497  100804  104346  102371.0  102865.0  102761
G  103899  109836  109154  110026  109934  108649  108451.0  106294.0  106514
N      80      13       9       9       9       3       NaN       NaN      18
T  147458  142104  144113  145028  146148  143930  145475.0  146893.0  147703

          9    ...           40      41        42        43      44      45  \
A  143183.0    ...     143887.0  143530  143996.0  144395.0  144032  144509
C  102965.0    ...     104760.0  104433  104786.0  104378.0  105277  104574
G  107045.0    ...     105983.0  105775  106106.0  105836.0  105238  105619
N       NaN    ...          NaN       1       NaN       NaN       6      17
T  146807.0    ...     145370.0  146261  145112.0  145391.0  145447  145281

       46      47      48      49
A  144093  144195  144269  144351
C  105616  105016  104791  104848
G  105799  105437  105608  105393
N      15      11      30       3
T  144477  145341  145302  145405

[5 rows x 50 columns]
```

Transposing and extracting a column…



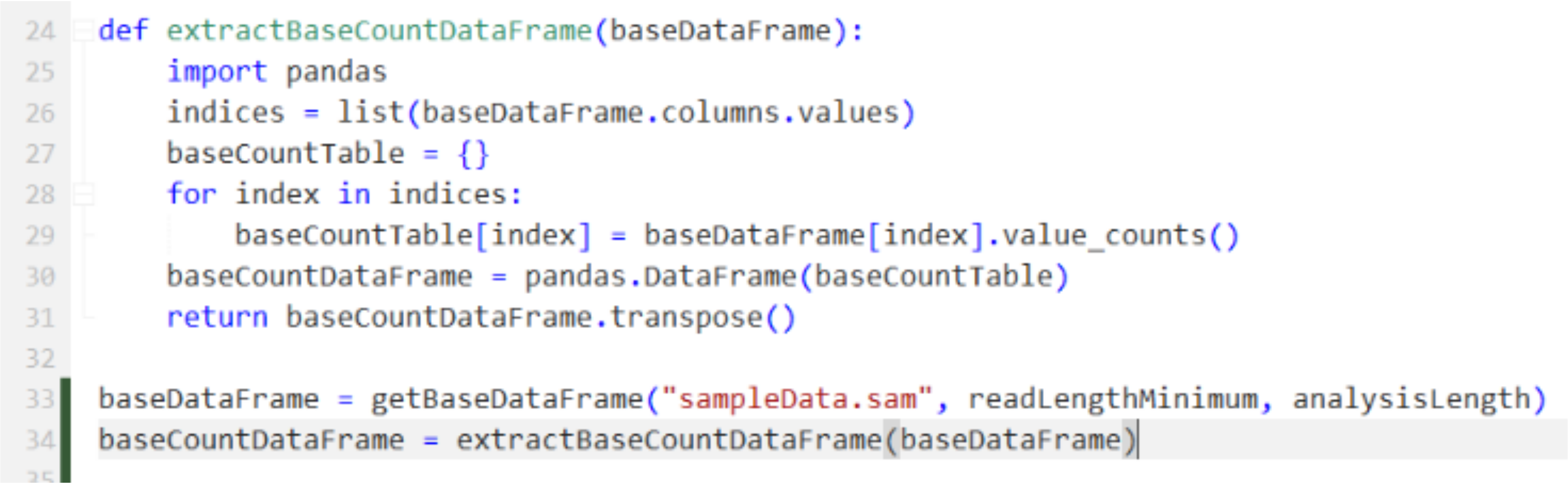
```
def extractBaseCountDataFrame(baseDataFrame):
    import pandas
    indices = list(baseDataFrame.columns.values)
    baseCountTable = {}
    for index in indices:
        print(index)
        baseCountTable[index] = baseDataFrame[index].value_counts()
    baseCountDataFrame = pandas.DataFrame(baseCountTable)
    print(baseCountDataFrame)
    baseCountDataFrame = baseCountDataFrame.transpose()
    print(baseCountDataFrame["A"])
    quit()


baseDataFrame = getBaseDataFrame("sampleData.sam", 100, 50)
extractBaseCountDataFrame(baseDataFrame)
```

This is what we want to see



```
[5 rows x 50 columns]
0      140610.0
1      145615.0
2      143508.0
3      142440.0
4      143105.0
5      143072.0
6      143703.0
7      143948.0
8      143004.0
9      143183.0
10     143166.0
11     143227.0
12     143630.0
13     143039.0
14     143337.0
15     143288.0
16     143769.0
17     143421.0
18     143207.0
19     143117.0
20     143231.0
21     142820.0
22     143374.0
23     142784.0
24     143335.0
25     143396.0
26     143561.0
27     143462.0
28     143308.0
29     142722.0
30     143586.0
31     143822.0
32     143662.0
```

Finalize the function and call



```
def extractBaseCountDataFrame(baseDataFrame):
    import pandas
    indices = list(baseDataFrame.columns.values)
    baseCountTable = {}
    for index in indices:
        baseCountTable[index] = baseDataFrame[index].value_counts()
    baseCountDataFrame = pandas.DataFrame(baseCountTable)
    return baseCountDataFrame.transpose()

baseDataFrame = getBaseDataFrame("sampleData.sam", readLengthMinimum, analysisLength)
baseCountDataFrame = extractBaseCountDataFrame(baseDataFrame)
```

Important question: How to represent our data

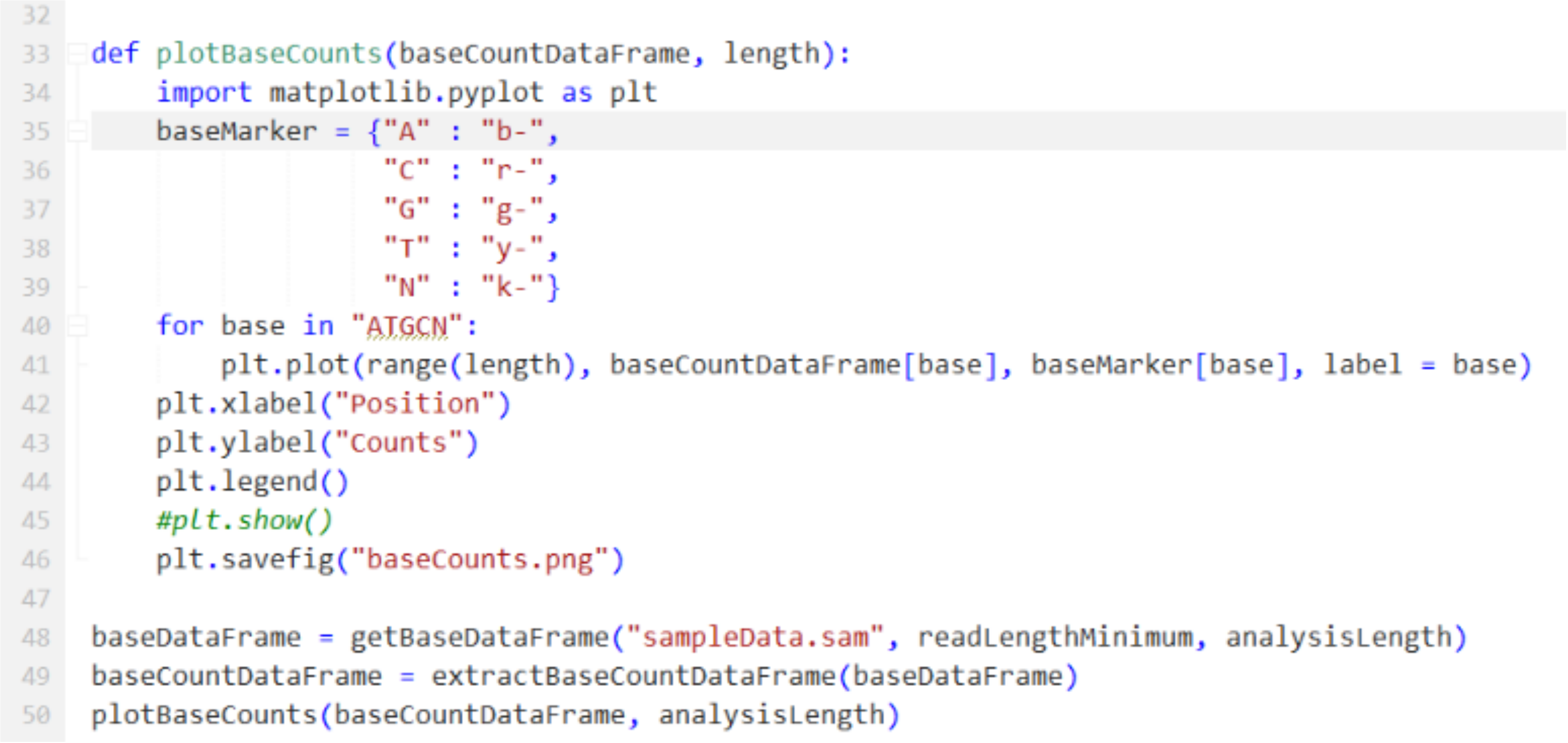
- Class participation time: Start making suggestions
- Don’t cheat here if you’ve looked ahead
- More than one right answer

Pyplot marker codes
(Don’t worry about memorizing) Keep a reference if you use these often

character	color
‘b’	blue
‘g’	green
‘r’	red
‘c’	cyan
‘m’	magenta
‘y’	yellow
‘k’	black
‘w’	white

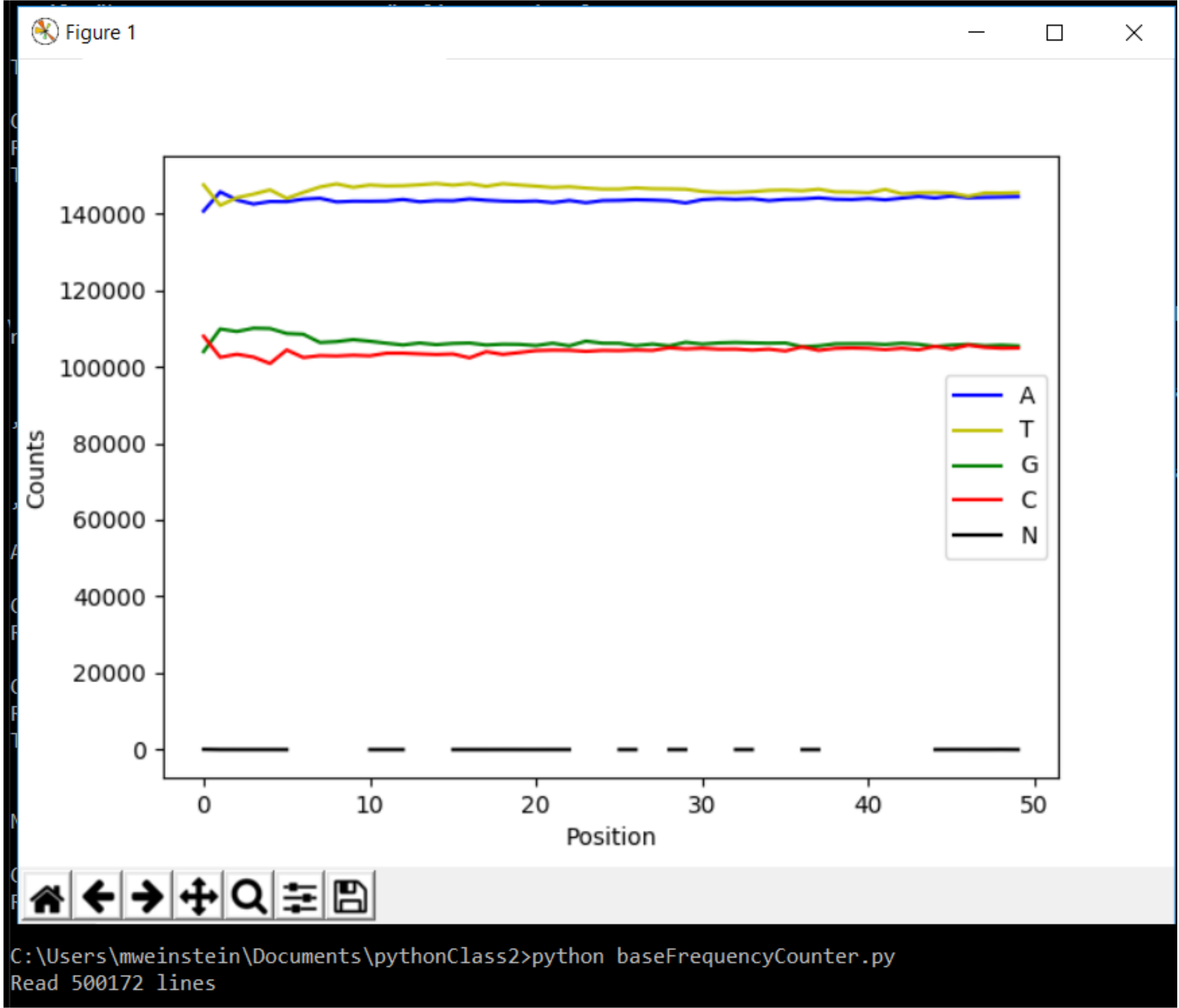
character	description
'-'	solid line style
'--'	dashed line style
'-.'	dash-dot line style
':'	dotted line style
'.'	point marker
','	pixel marker
'o'	circle marker
'v'	triangle_down marker
'^'	triangle_up marker
'<'	triangle_left marker
'>'	triangle_right marker
'1'	tri_down marker
'2'	tri_up marker
'3'	tri_left marker
'4'	tri_right marker
's'	square marker
'p'	pentagon marker
'*'	star marker
'h'	hexagon1 marker
'H'	hexagon2 marker
'+'	plus marker
'x'	x marker
'D'	diamond marker
'd'	thin_diamond marker
'|'	vline marker
'_'	hline marker

Iterative plotting… it’s really this simple



```
def plotBaseCounts(baseCountDataFrame, length):
    import matplotlib.pyplot as plt
    baseMarker = {"A" : "b-",
                  "C" : "r-",
                  "G" : "g-",
                  "T" : "y-",
                  "N" : "k-"}
    for base in "ATGCN":
        plt.plot(range(length), baseCountDataFrame[base], baseMarker[base], label = base)
    plt.xlabel("Position")
    plt.ylabel("Counts")
    plt.legend()
    #plt.show()
    plt.savefig("baseCounts.png")

baseDataFrame = getBaseDataFrame("sampleData.sam", readLengthMinimum, analysisLength)
baseCountDataFrame = extractBaseCountDataFrame(baseDataFrame)
plotBaseCounts(baseCountDataFrame, analysisLength)
```

Success!



```
C:\Users\mweinstein\Documents\pythonClass2>python baseFrequencyCounter.py
Read 500172 lines
```