UCLA QCBio Collaboratory

Intermediate/Advanced Python

Michael Weinstein

(Day 2)

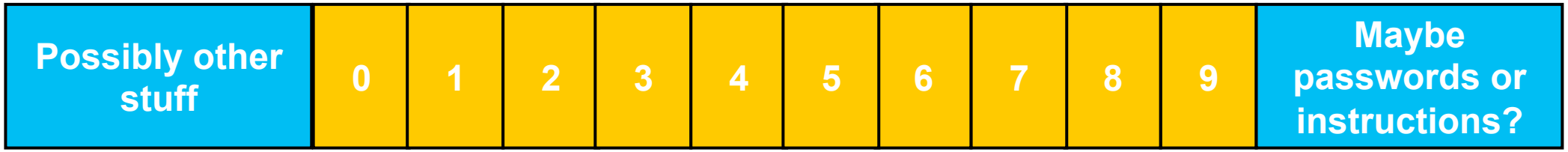
Topics

- Review of basic data structures
- Accessing and working with objects in python
- Numpy
  - How python actually stores data in memory
  - Why numpy can help
  - Dot product example
  - Extending our SAMLine object from yesterday
  - Making and analyzing our quality score matrix
- Pandas
- Matplotlib
  - Making a histogram
- Scipy

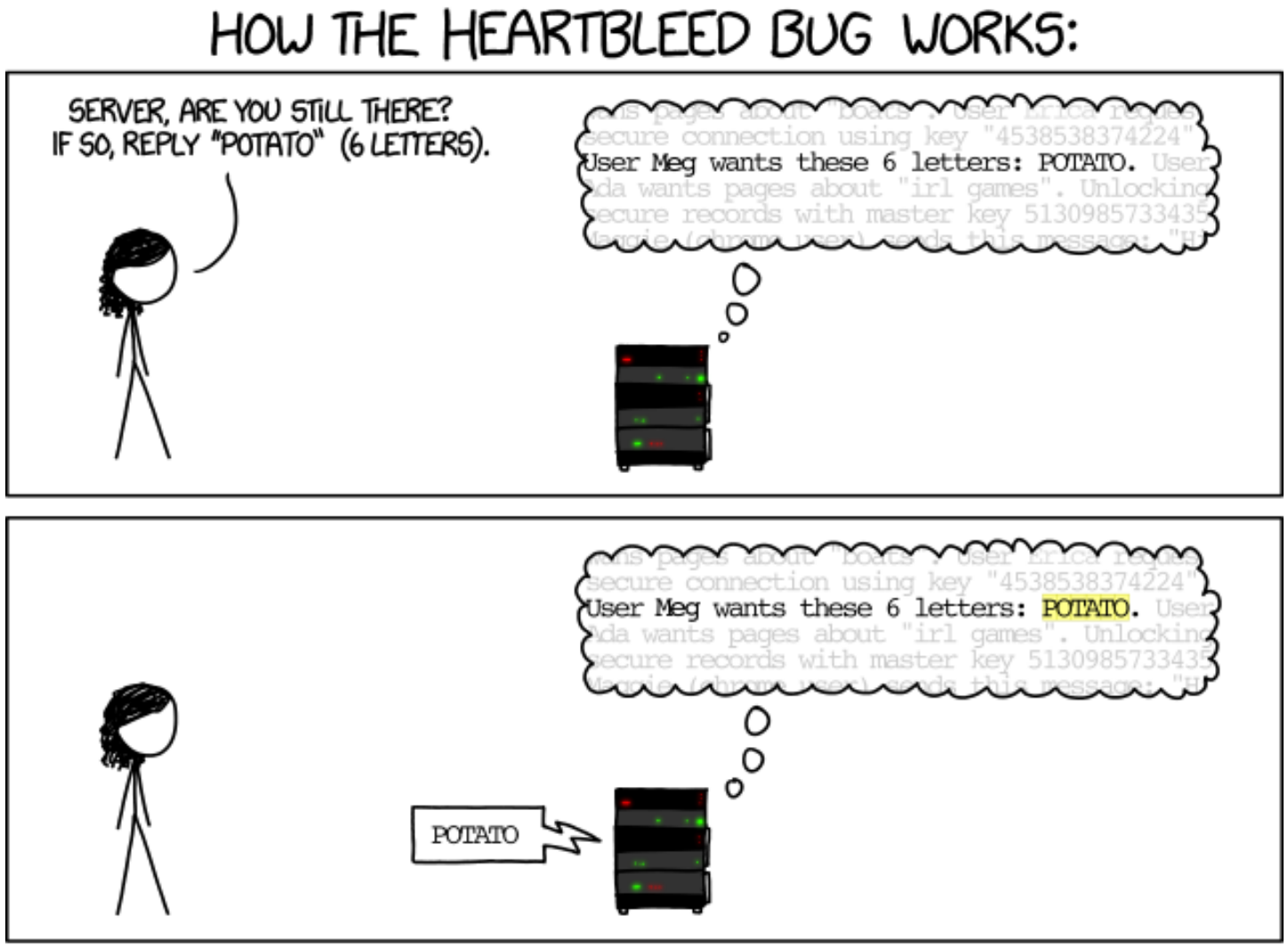
A true array (from C++ or similar languages)

› A true array is stored sequentially in a fixed space in the computer's memory

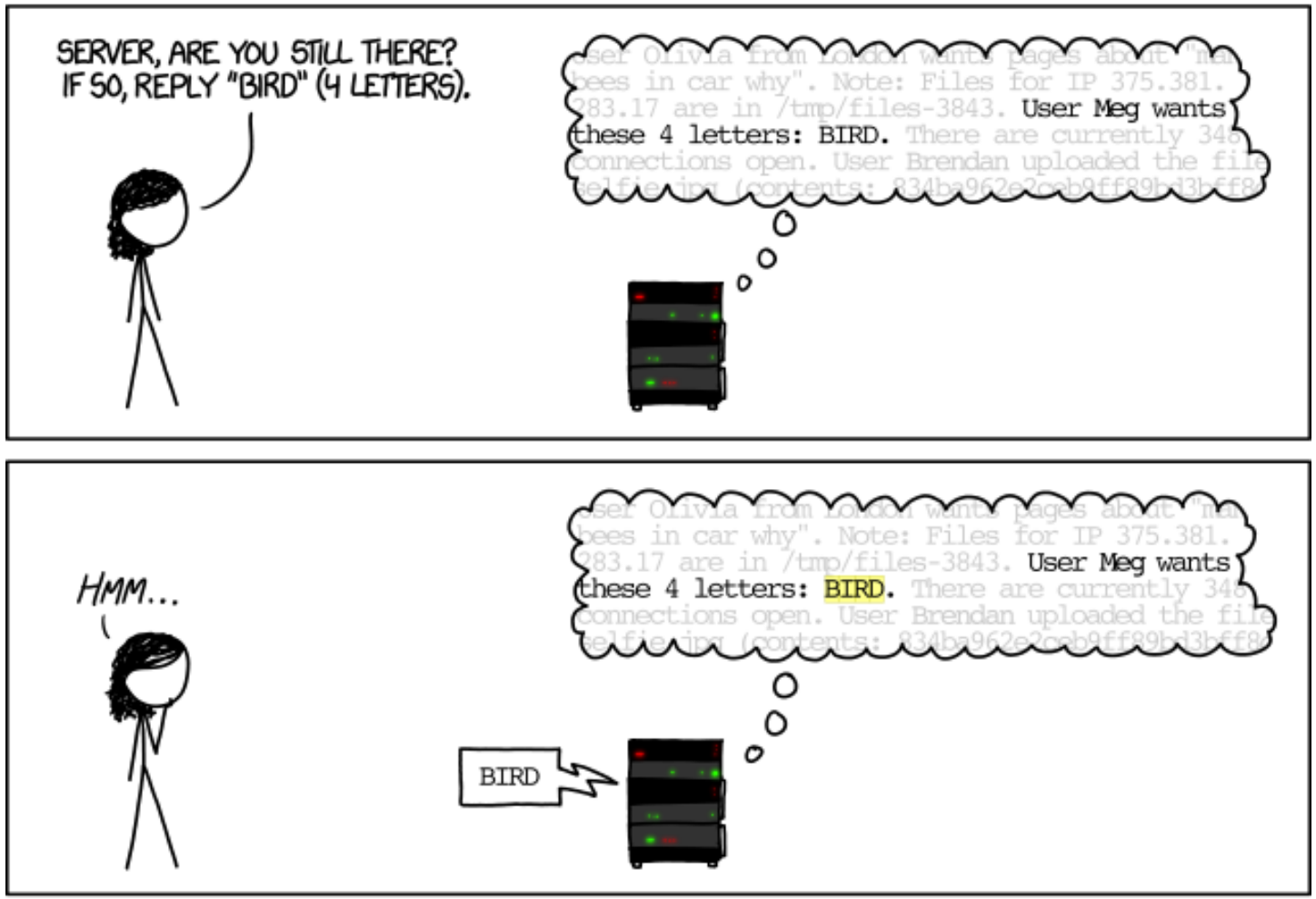
A 10 integer array



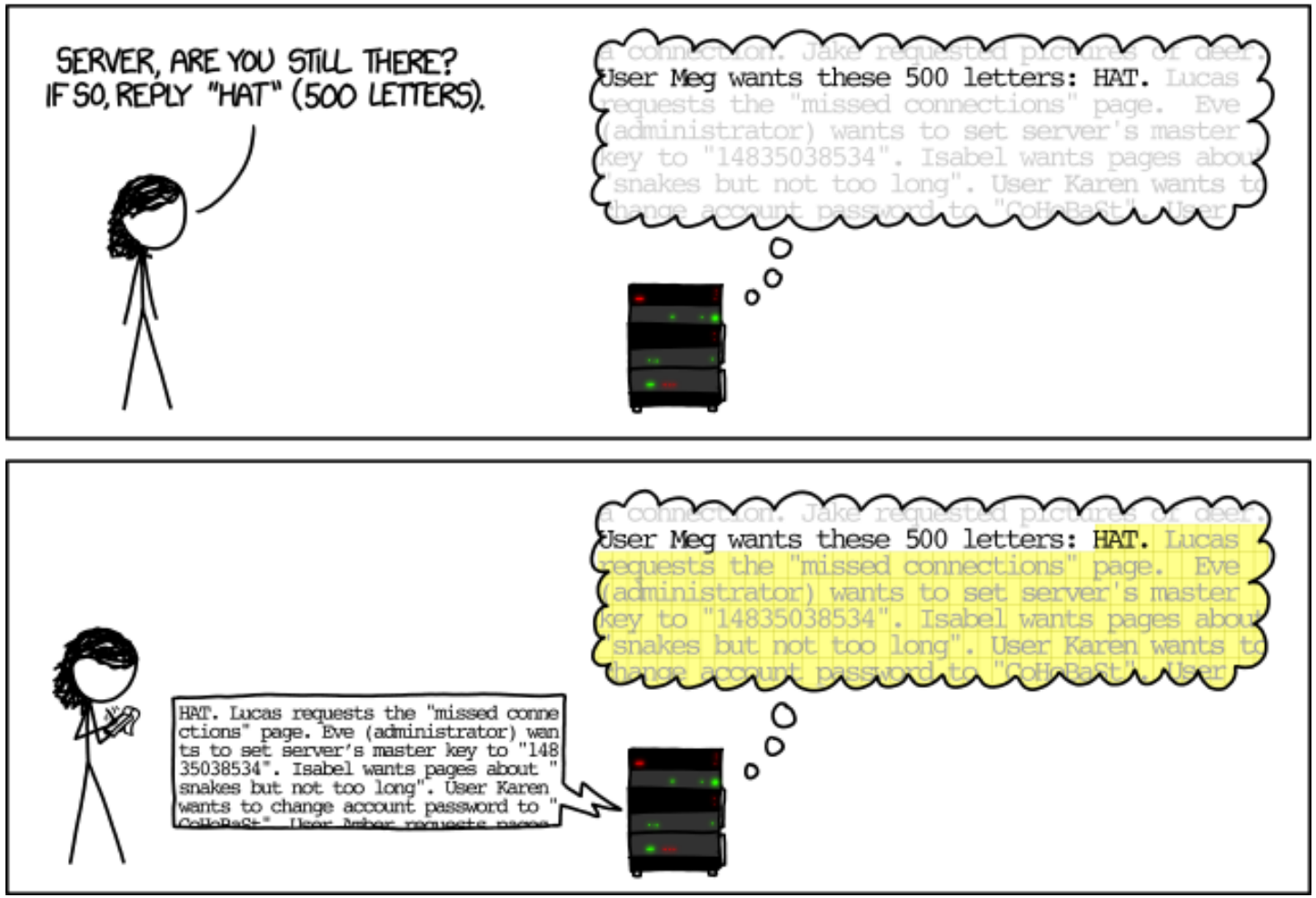
Seriously, that other stuff isn't a joke



Seriously, that other stuff isn't a joke



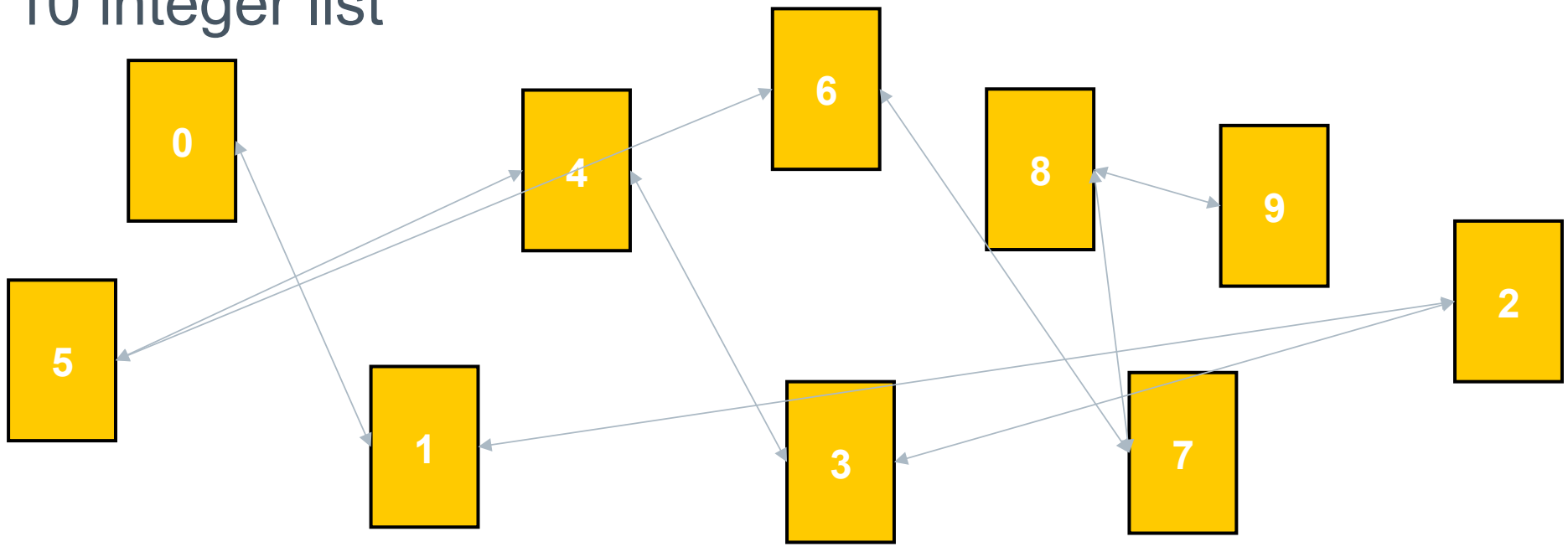
Seriously, that other stuff isn't a joke



A python list uses pointers for flexibility

› A pointer is a value that points to a specific location in the computer's memory

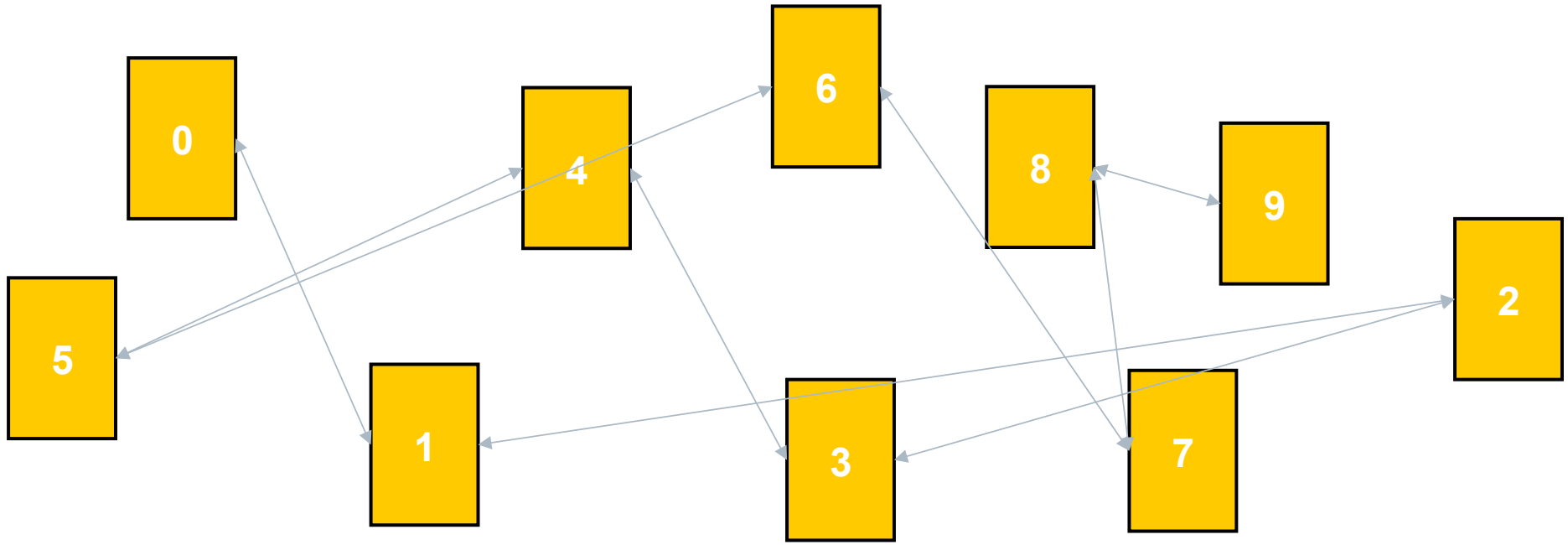
A 10 integer list



A python list uses pointers for flexibility

› This method of storing data increases flexibility

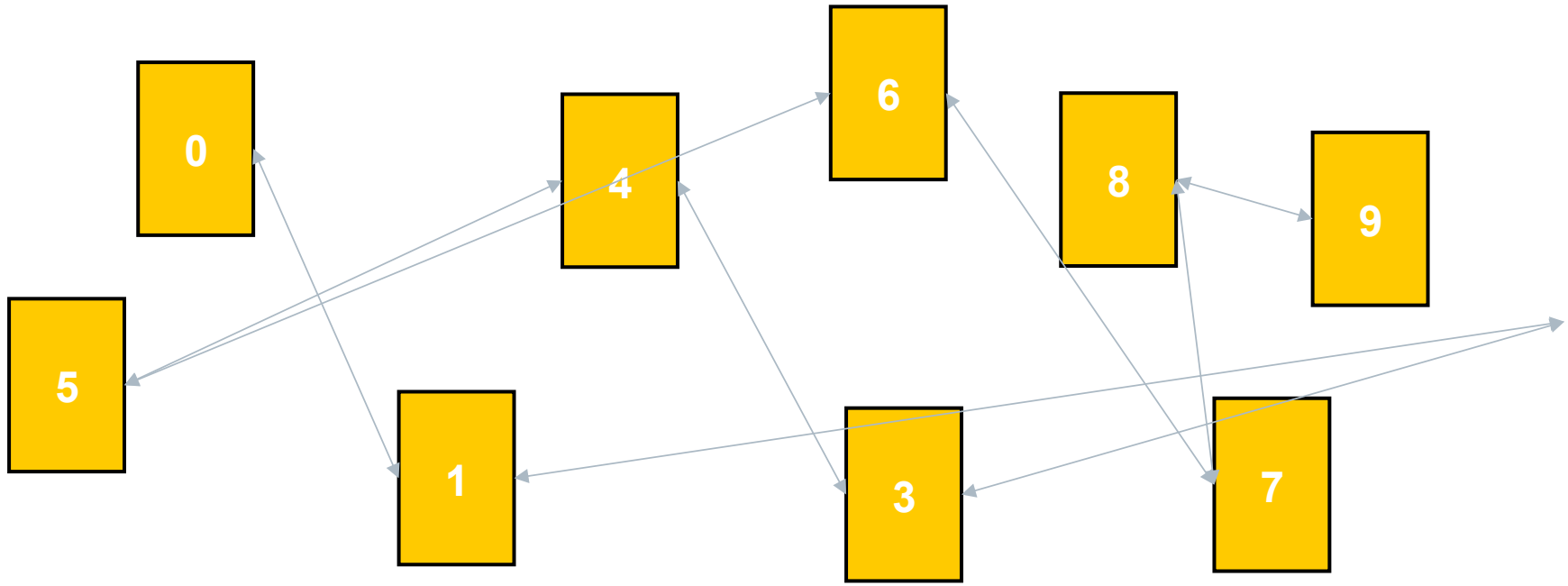
Deleting a number from a 10 integer list



A python list uses pointers for flexibility

› This method of storing data increases flexibility

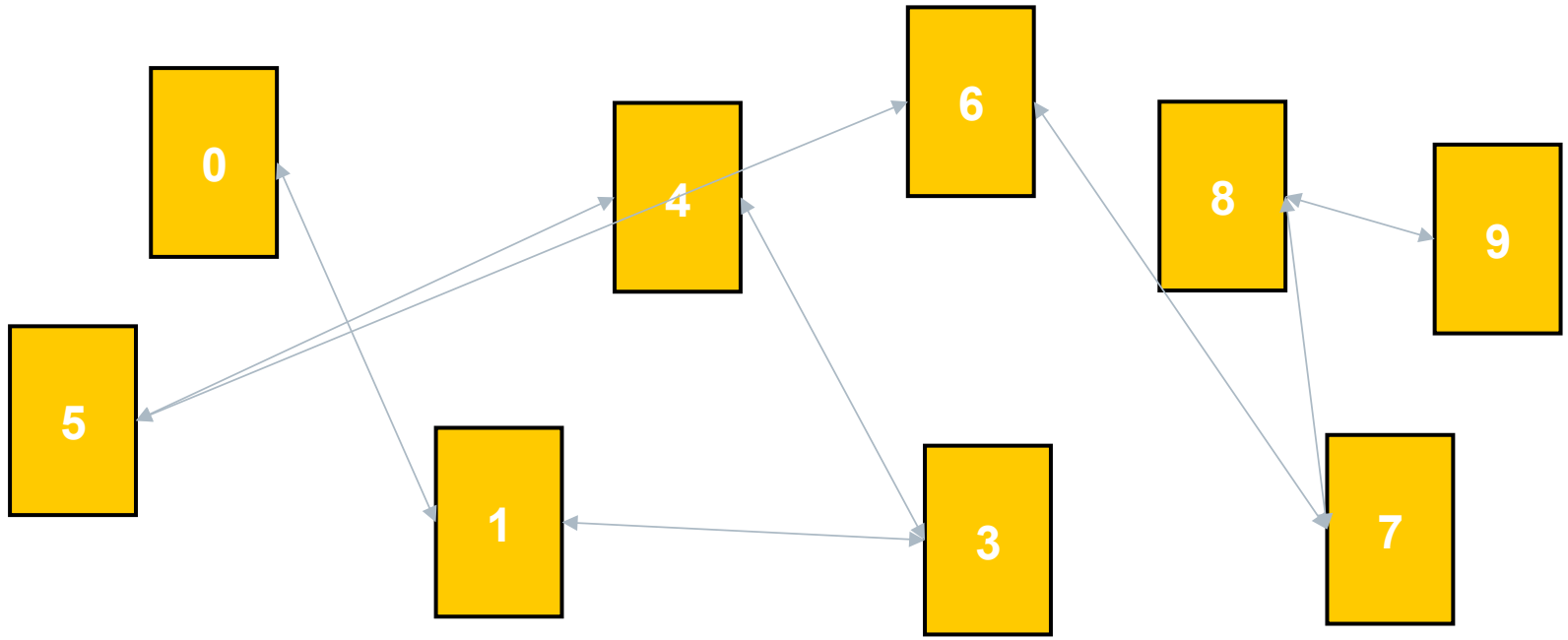
Deleting a number from a 10 integer list



A python list uses pointers for flexibility

› This method of storing data increases flexibility

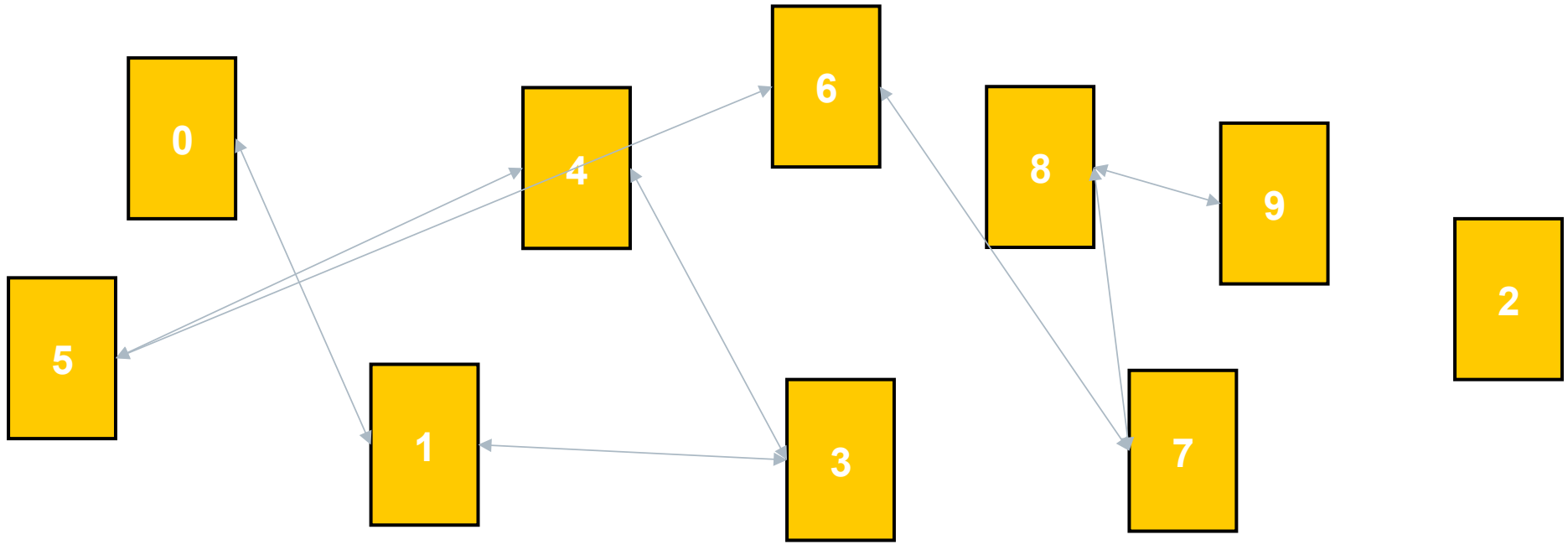
Deleting a number from a 10 integer list



A python list uses pointers for flexibility

› This method of storing data increases flexibility

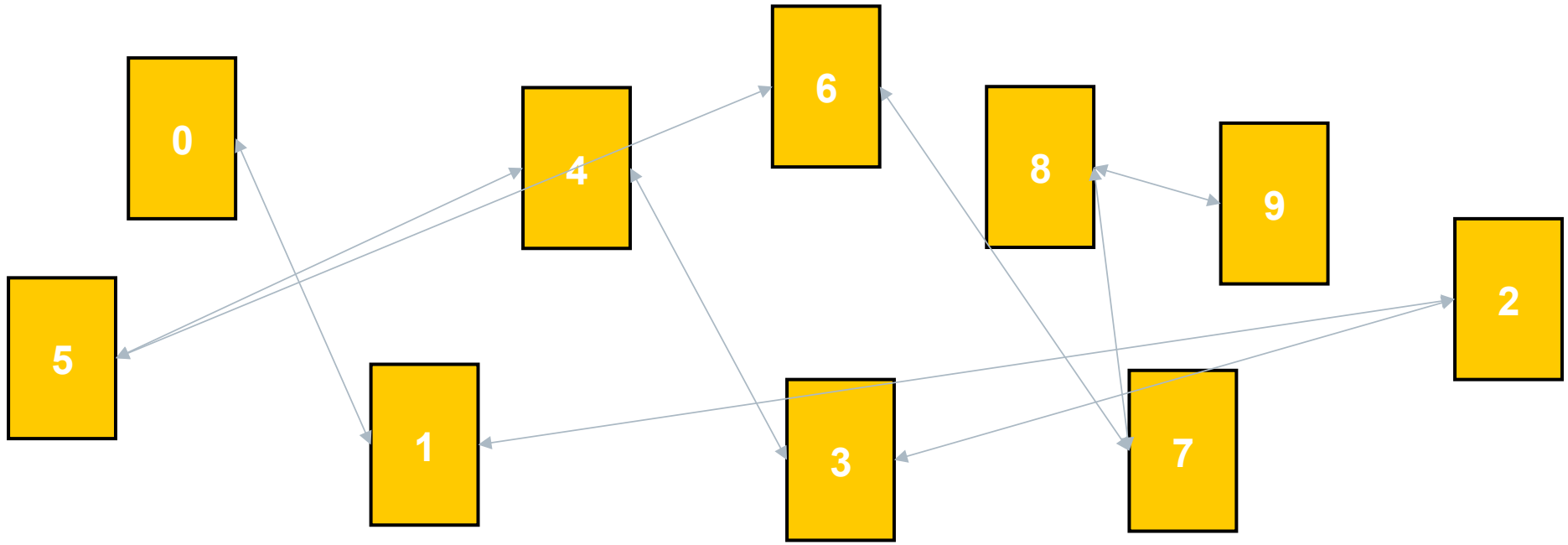
Adding a number to a 10 integer list



A python list uses pointers for flexibility

› This method of storing data increases flexibility

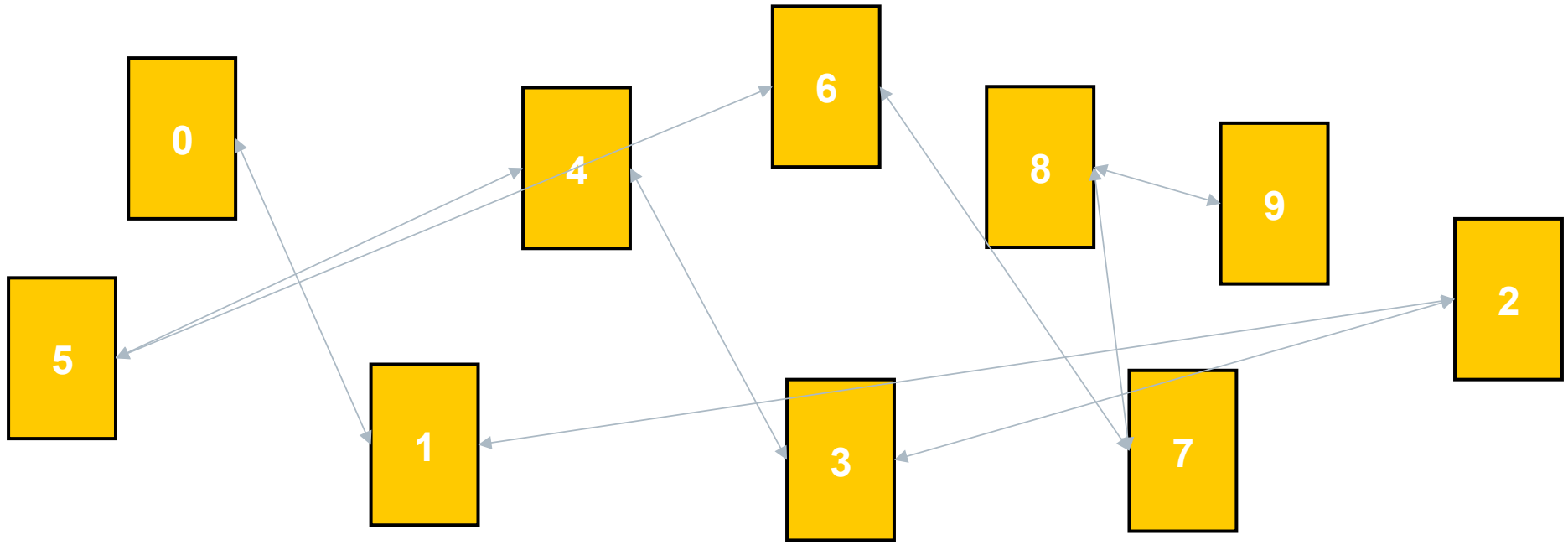
Adding a number to a 10 integer list



A python list uses pointers for flexibility

› This method of storing data increases flexibility

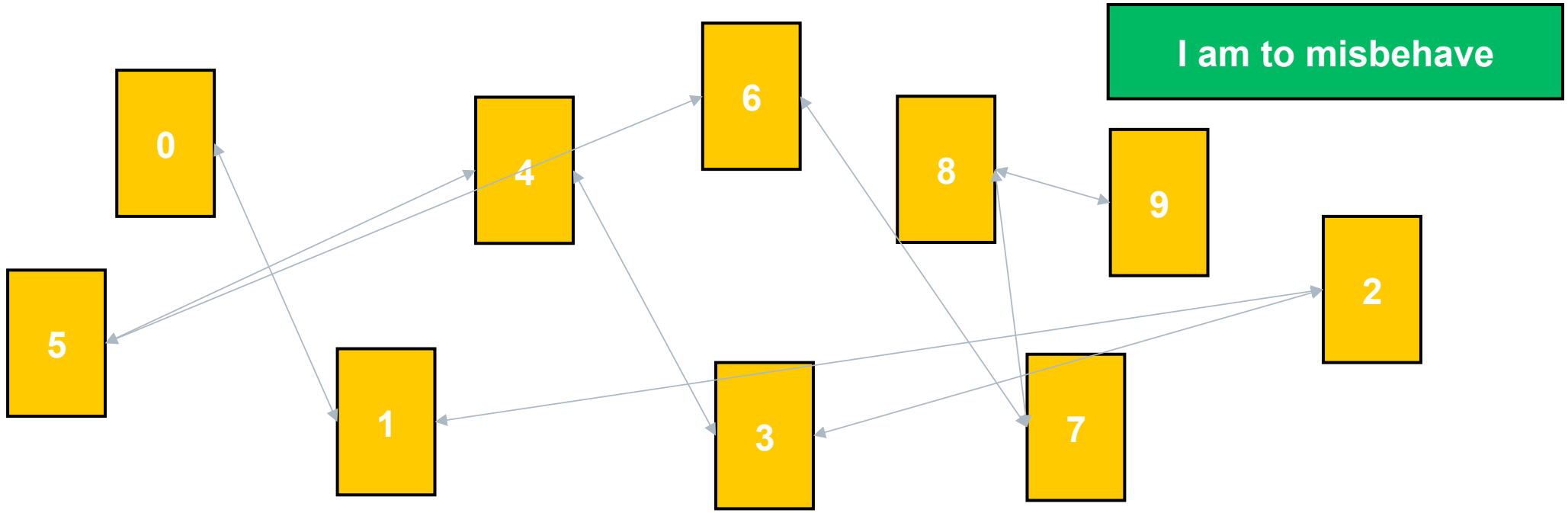
Replacing a number in a 10 integer list with a string



A python list uses pointers for flexibility

› This method of storing data increases flexibility

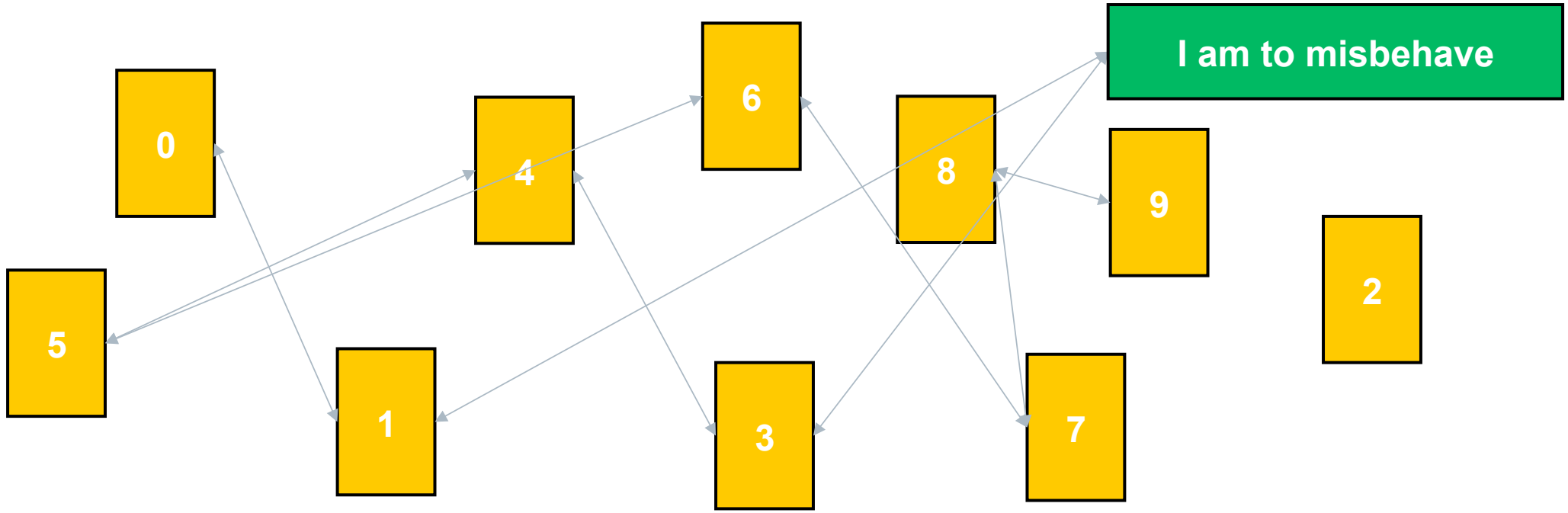
Replacing a number in a 10 integer list with a string



A python list uses pointers for flexibility

› This method of storing data increases flexibility

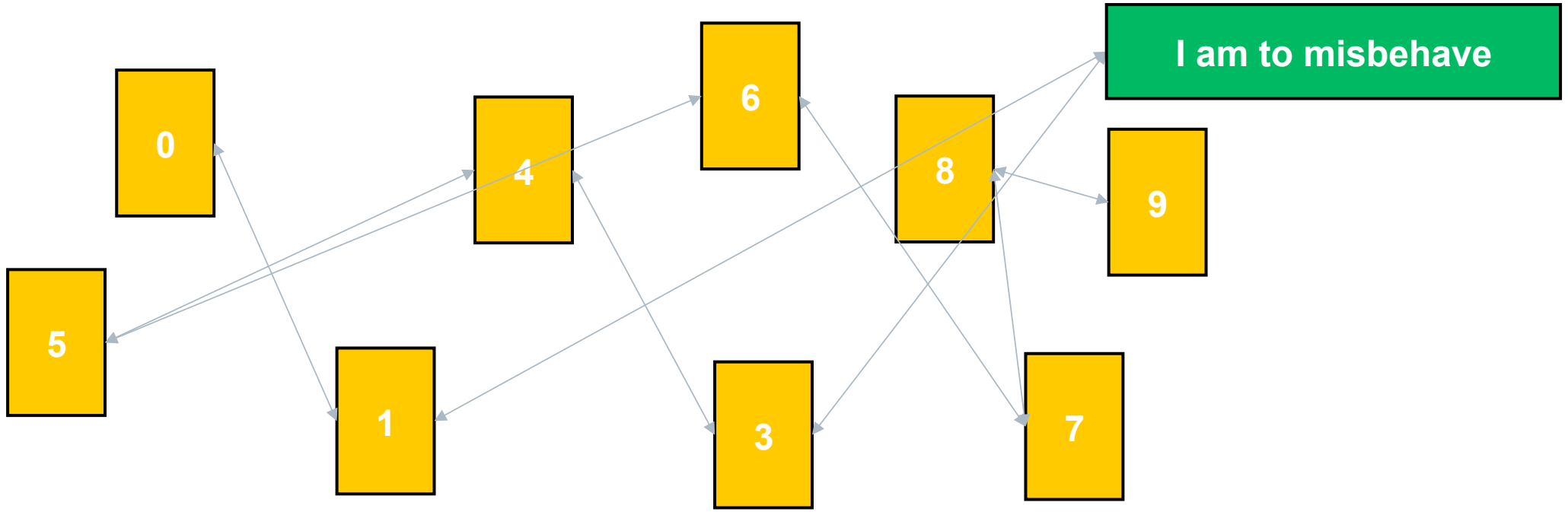
Replacing a number in a 10 integer list with a string



A python list uses pointers for flexibility

› This method of storing data increases flexibility

Replacing a number in a 10 integer list with a string



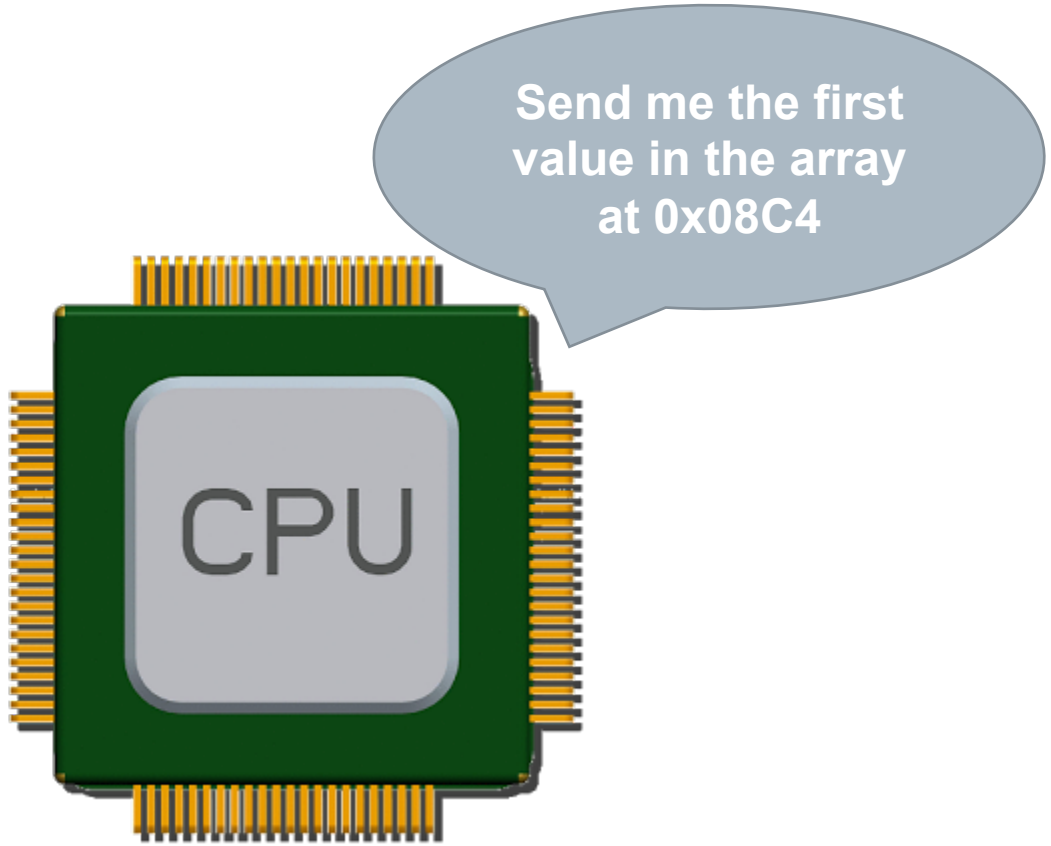
Numpy gives us access to true arrays

- Your numpy objects will have limited flexibility
- Data in your numpy objects will have to be homogeneous
- There will be some overhead in terms of computing time to turn your python object into a numpy object

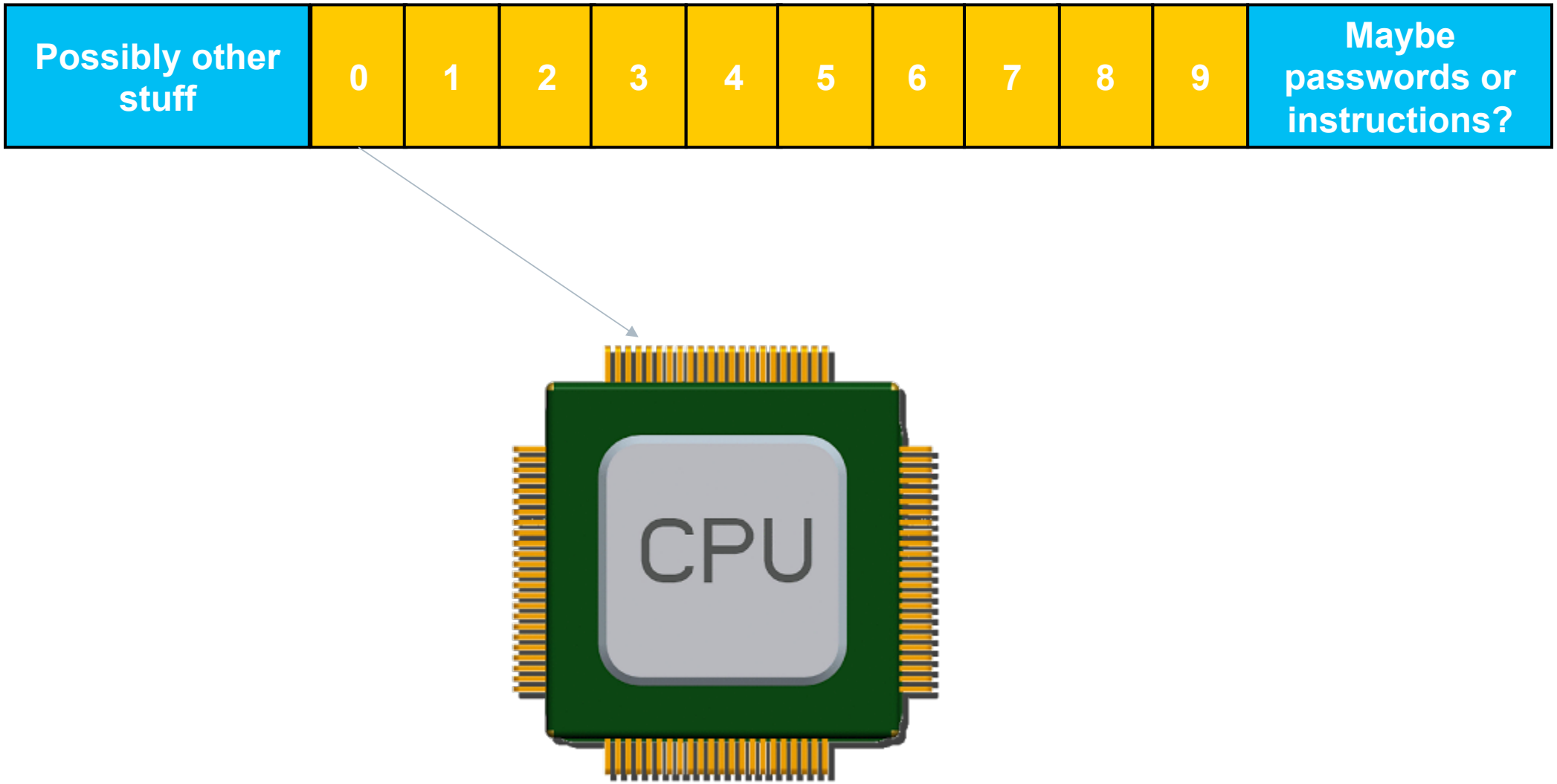
- Why on Earth would we want to do this?

A true array (from C++ or similar languages)

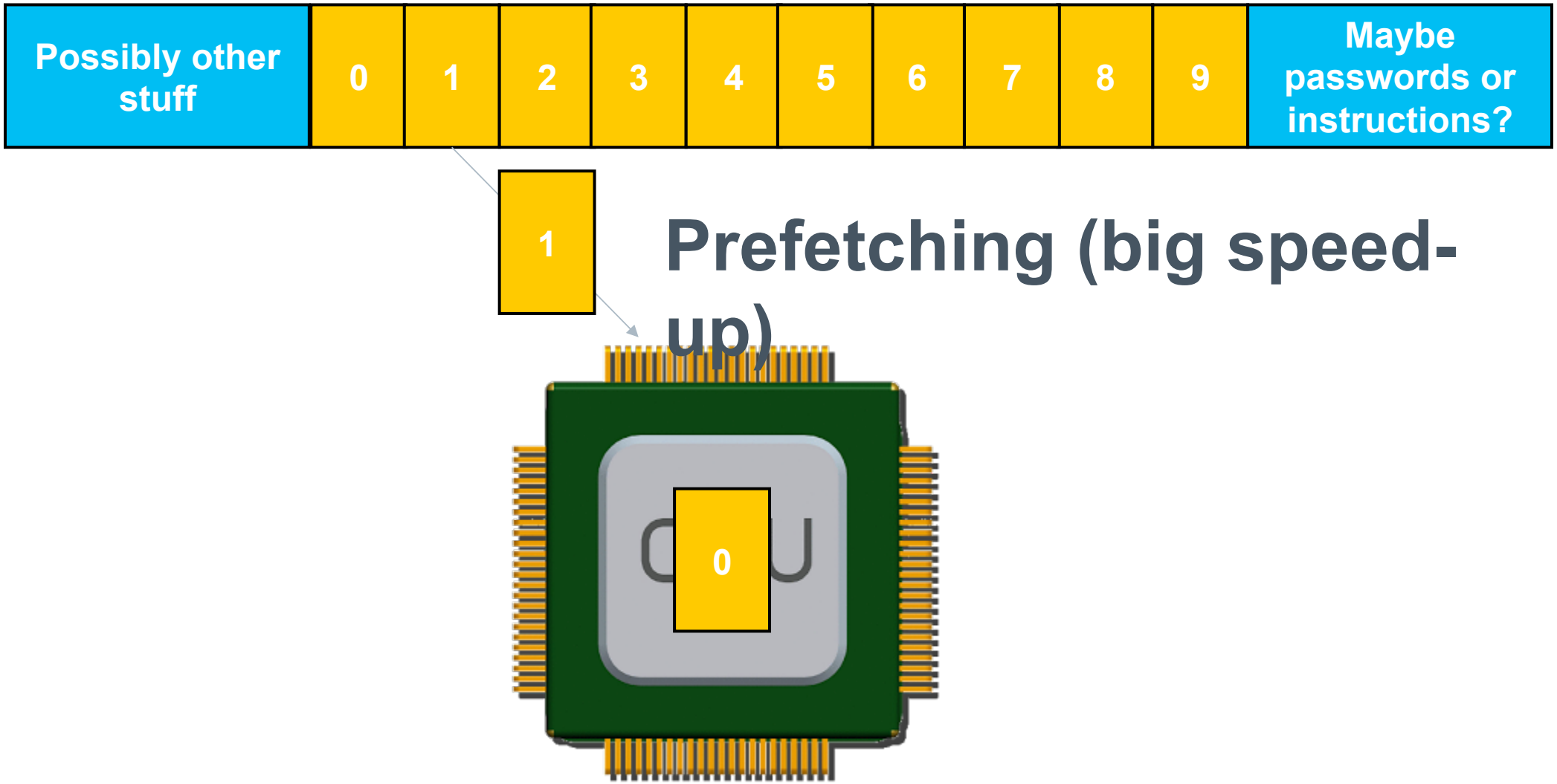
Possibly other stuff	0	1	2	3	4	5	6	7	8	9	Maybe passwords or instructions?



A true array (from C++ or similar languages)



A true array (from C++ or similar languages)



Let's prove this
(and learn some useful functions)

```
3  import numpy
4  import random
5  import datetime
6
7  length = 100
8  firstMatrixPython = [random.gauss(0, 1)] * length
9  secondMatrixPython = [random.gauss(0, 1)] * length
10 firstMatrixNumpy = numpy.random.randn(length)
11 secondMatrixNumpy = numpy.random.randn(length)
12 iterations = 100000
13
14 def manualDotProduct(first, second):
15     result = 0
16     for i in range(len(first)):
17         result += first[i] * second[i]
18     return result
19
20 def numpyDotProduct(first, second):
21     return first.dot(second)
22
23 def runManual(first, second, iterations):
24     for i in range(iterations):
25         dontCare = manualDotProduct(first, second)
26
27 def runNumpy(first, second, iterations):
28     for i in range(iterations):
29         dontCare = numpyDotProduct(first, second)
```

And the commands…

```
30
31  start = datetime.datetime.now()
32  runManual(firstMatrixPython, secondMatrixPython, iterations)
33  print("Python and manual run in %s" %(datetime.datetime.now() - start))
34
35  start = datetime.datetime.now()
36  runManual(firstMatrixNumpy, secondMatrixNumpy, iterations)
37  print("Numpy and manual run in %s" %(datetime.datetime.now() - start))
38
39  start = datetime.datetime.now()
40  runNumpy(firstMatrixNumpy, secondMatrixNumpy, iterations)
41  print("Numpy automatic run in %s" %(datetime.datetime.now() - start))
42
```

Results

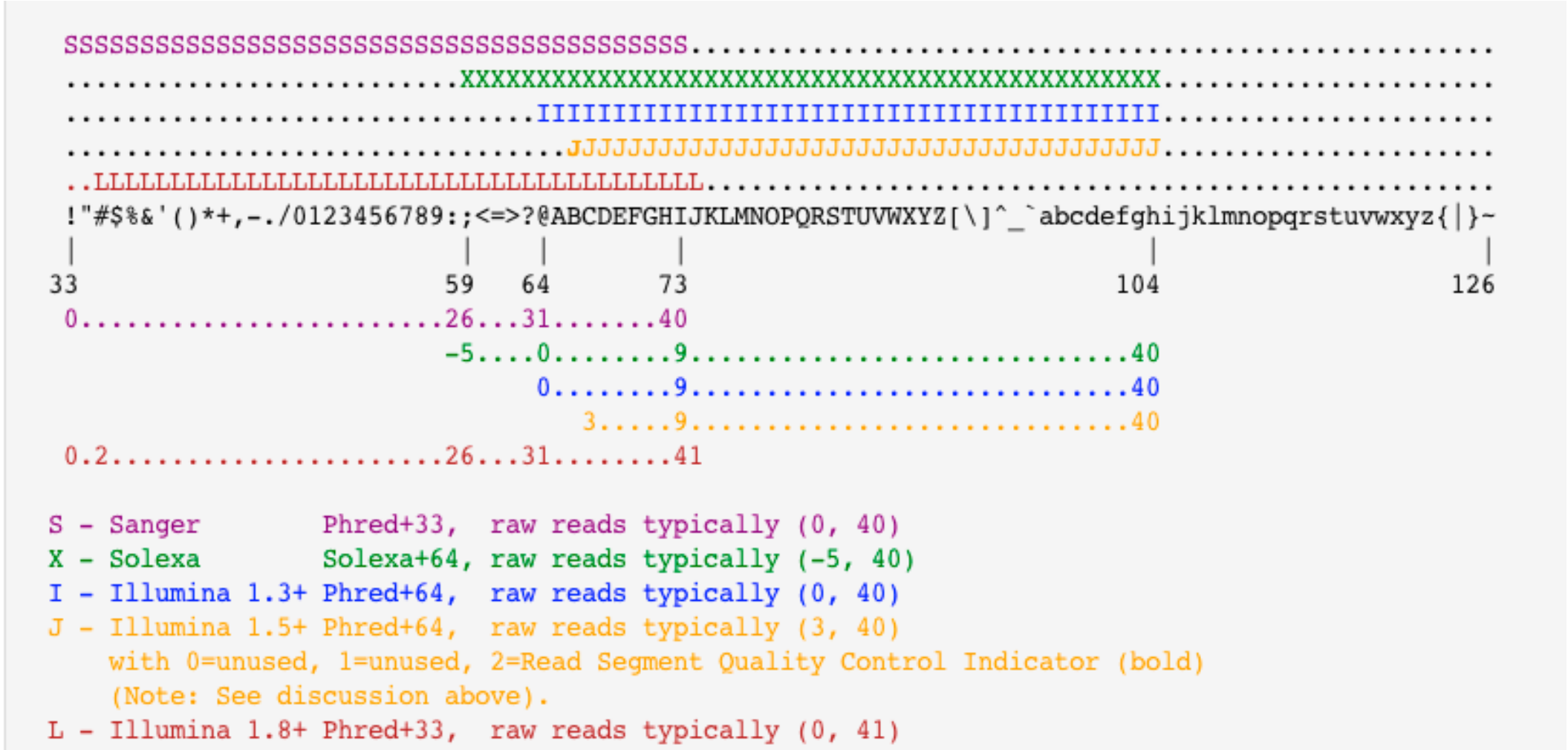
```
C:\Users\mweinstein\Documents\pythonClass2>python numpySpeedComparison.py
Python and manual run in 0:00:00.651783
Numpy and manual run in 0:00:03.438755
Numpy automatic run in 0:00:00.058057
```

- Python is already pretty fast for this operation
- Going back and forth between numpy and standard python makes it much less fast
- Running as much of this in numpy as possible gave about a 10x speed increase

Topics

- Review of basic data structures
- Accessing and working with objects in python
- Numpy
  - ~~How python actually stores data in memory~~
  - ~~Why numpy can help~~
  - ~~Dot product example~~
  - Extending our SAMLine object from yesterday
  - Making and analyzing our quality score matrix
- Pandas
- Matplotlib
  - Making a histogram
- Scipy

Quality Scores



Time to code a little more

- Create a new file in your working folder called qualityStringHandler.py
- Make a copy of your working samReader.py called samReader2.py

How to convert ASCII values

```
C:\Users\mweinstein\Documents\pythonClass2>python
Python 3.6.2 (v3.6.2:5fd33b5, Jul  8 2017, 04:57:36) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> ord("!")
33
>>> ord("@")
64
>>> ord("M")
77
>>> chr(33)
'!'
>>> chr(64)
'@'
>>> chr(77)
'M'
>>>
```

Our new class for quality strings

```
3  class QualityString(object):
4
5      def __init__(self, rawQualityString, base = 33):
6          assert type(rawQualityString) == str, "Quality string must be a string."
7          self.qualityString = rawQualityString.strip()
8          self.qualityArray = self.calculateQualityArray(base)
9
10     def calculateQualityArray(self, base = 33):
11         collection = []
12         for character in self.qualityString:
13             collection.append(ord(character) - base)
14         return collection
15
16     def __str__(self):
17         return qualityString
18
```

Modify SAMLine to handle the new data

```
21 class SAMLine(object):
22
23     def __init__(self, rawLine):
24         import dnaSequenceHandler
25         import qualityStringHandler
26         self.rawLine = rawLine.strip()
27         lineArray = self.rawLine.split("\t")
28         self.readID = lineArray[0]
29         self.flag = int(lineArray[1])
30         self.chromosome = lineArray[2]
31         self.readStartPosition = int(lineArray[3])
32         self.alignmentPhred = int(lineArray[4])
33         self.cigar = lineArray[5]
34         self.mateChromosome = lineArray[6]
35         if self.mateChromosome == "=":
36             self.mateChromosome =  self.chromosome
37         self.mateStartPosition = int(lineArray[7])
38         self.readLength = int(lineArray[8])
39         self.sequence = dnaSequenceHandler.DNASequence(lineArray[9])
40         self.quality = qualityStringHandler.QualityString(lineArray[10])
41         self.otherValues = lineArray[11:]
42
43     def __str__(self):
44         return self.rawLine
45
46 if __name__ == "__main__":
47     samLines = readSAMFileLines("sampleData.sam")
48     for line in samLines[:10]:
49         print(line)
50         print(line.quality.qualityArray)
```

Result

```
C:\Users\mweinstein\Documents\pythonClass2>python samReader2.py
Read 500172 lines
SRR067577.2766  99      chr14   73240003        60      101M    =       73240004        102     GCTAATTTTTTATATTTTTAGTAGAGACAGAGTTTCACTATATTGGTCAGGCTGGTCTCAAACTCCTGACCTCAGGCGATCC
ACCCGCCTCAGCCTCCCAA     IIIIIIIIIIIIFIIIIIFIHIIIIHIDIIIFIIIDHIIIIHHHHGHIHGIGHHEGGGDEGEEGGDGEGEDEDD@DD7DD@BB@>=B############   NM:i:0  MD:Z:101        AS:i:101        XS:i:63
[40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 37, 40, 40, 40, 40, 40, 37, 40, 39, 40, 40, 40, 40, 39, 40, 35, 40, 40, 40, 37, 40, 40, 40, 35, 39, 40, 40, 40, 40, 39, 39, 3
9, 39, 38, 39, 40, 39, 38, 40, 38, 39, 39, 36, 38, 38, 38, 35, 36, 38, 36, 36, 38, 38, 35, 38, 36, 38, 36, 35, 36, 35, 35, 31, 35, 35, 22, 35, 35, 31, 33, 33, 31, 29, 28, 33, 2,
2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2]
SRR067577.2766  147     chr14   73240004        60      101M    =       73240003        -102    CTTATTTTTTATATTTTTAGTAGAGACAGAGTTTCACTATATTGGTCAGGCTGGTCTCAAACTCCTGACCTCAGGCGANNCA
CCCGCCTCAGCCTCCCAAA     ##########################################################@@@@@636;6/,30004!!;'<@@@AAAAA<<AAAAAAAA   NM:i:3  MD:Z:2A75T0C21  AS:i:94 XS:i:40
[2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
2, 2, 31, 31, 31, 31, 31, 21, 18, 21, 26, 21, 14, 11, 18, 15, 15, 15, 19, 0, 0, 26, 6, 27, 31, 31, 31, 32, 32, 32, 32, 32, 27, 27, 32, 32, 32, 32, 32, 32, 32, 32]
SRR067577.3006  99      chr14   57829687        60      101M    =       57829756        170     CAATCTATTTAAAGTAATCCCTTGCACCTCTTATGCTCCCCCACAGCCCTTATAATATTTTAAGAGCATGTCTTTTTGTTTA
CATTTTTCCCATTAAATTG     IIIHIHIIIIIIIIIHIHIHIIHIIIHGGIIIIIGHHIIIIHIHHIFIBIEDHFFFGGGEHD0BB>E@BCEDEGDECACF9CCBBEB>B@>??<A;??;A##   NM:i:0  MD:Z:101        AS:i:101        XS:i:0
[40, 40, 40, 39, 40, 39, 40, 40, 40, 40, 40, 40, 40, 40, 39, 40, 39, 40, 39, 40, 40, 39, 40, 40, 40, 39, 38, 38, 40, 40, 40, 40, 40, 38, 39, 39, 40, 40, 40, 40, 39, 40, 39, 39, 4
0, 37, 40, 33, 40, 36, 35, 39, 37, 37, 37, 38, 38, 38, 36, 39, 35, 15, 33, 33, 29, 36, 31, 33, 34, 36, 35, 36, 38, 35, 36, 34, 32, 34, 37, 24, 34, 34, 33, 33, 36, 33, 29, 33, 31,
 29, 30, 30, 27, 32, 26, 30, 30, 26, 32, 2, 2]
SRR067577.3006  147     chr14   57829756        60      101M    =       57829687        -170    GTCTTTTTGTTTACATTTTTCCCATTAAATTGTGAGCTCTGGGATAGACCCTTTATCCAGTTAATCTTTTTGTCCCTCCAAA
GCCAGAATGTACAGCTCTG     ##########?4??/@5@7@7DBDC>>?::@@E@BBB,EBACEBDEABEEDGGEGHGDHHHFCFIHHGGHHFIHIFIIIIIHIFIIIIGIIIIIHIIIII   NM:i:0  MD:Z:101        AS:i:101        XS:i:0
[2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 30, 19, 30, 30, 14, 31, 20, 31, 22, 31, 22, 35, 33, 35, 34, 29, 29, 30, 25, 25, 31, 31, 36, 31, 33, 33, 33, 11, 36, 33, 32, 34, 36, 33, 35, 36,
32, 33, 36, 36, 35, 38, 38, 36, 38, 39, 38, 35, 39, 39, 39, 37, 34, 37, 40, 39, 39, 38, 38, 39, 39, 37, 40, 39, 40, 37, 40, 40, 40, 40, 40, 39, 40, 37, 40, 40, 40, 40, 38, 40, 40
, 40, 40, 40, 39, 40, 40, 40, 40, 40]
SRR067577.3000  99      chr14   41116341        60      101M    =       41116407        167     AAGAAATGAGAAGTAATCAGAAAACCACTTAAGGTGAATGGAAATAGGGACACAGCATACCAAAACTTAGGGATGCAGTAGA
AGCAGTGCTTAGCAGGAAT     IGIIIIIIIIIHHIIIIIHIIGIIIIIIIBGGGGBIHEIIHGGGHHGDEGEGHGGGEEGHBBBBBD@@CBA>B@E><3/<??;??;<6=>0<<;@######   NM:i:0  MD:Z:101        AS:i:101        XS:i:0
[40, 38, 40, 40, 40, 40, 40, 40, 40, 40, 40, 39, 39, 40, 40, 40, 40, 40, 39, 40, 40, 38, 40, 40, 40, 40, 40, 40, 40, 33, 38, 38, 38, 38, 33, 40, 39, 36, 40, 40, 39, 38, 38, 38, 3
9, 39, 38, 35, 36, 38, 36, 38, 39, 38, 38, 38, 36, 36, 38, 39, 33, 33, 33, 33, 33, 35, 31, 31, 34, 33, 32, 29, 33, 31, 36, 29, 27, 18, 14, 27, 30, 30, 26, 30, 30, 26, 27, 21, 28,
 29, 15, 27, 27, 26, 31, 2, 2, 2, 2, 2, 2]
SRR067577.3000  147     chr14   41116407        60      101M    =       41116341        -167    CTAGGGATGCAGTAGAAGCAGTGCTTAGCAGGAATTTTATAATTGTAAATGCCTGTACTAAGAAATTTAAAAACATTTNNAC
TCATTAACCCAGTTAAAAA     #@@@9@@@@@@@@@@@8@@@@A6A@A@@@@@@A<<AAAAAAAAAAAAAAAAAAAAAA<<AAAAAAA?<<A<9>7:8:!!9@@BB@AAAAAAAAAAAAAAA   NM:i:3  MD:Z:0T77C0A21  AS:i:96 XS:i:0
[2, 31, 31, 31, 24, 31, 31, 31, 31, 31, 31, 31, 31, 31, 31, 23, 31, 31, 31, 31, 32, 21, 32, 31, 32, 31, 31, 31, 31, 31, 31, 32, 27, 27, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32
```

Goal:

- Generate a histogram to look at the average quality score distribution in the first 50 bases of any read of length 100 or more from the sample data
- What we know
  - We can already break down this data very well
  - We will have to filter on read length (easy, since we already store it)
  - We will have to iterate over lines and build up a matrix
  - We will have to take an average across each row (read) in the matrix
  - We will have to generate a histogram of these values
- New file: histogramMaker.py

How to get the list of lists

```
>>> import samReader2
>>> data = samReader2.readSAMFileLines("sampleData.sam")
Read 500172 lines
>>> collection = []
>>> for line in data:
...  if line.readLength >= 100:
...   collection.append(line.quality.qualityArray[0:50])
...  if len(collection) >= 10:
...   break
...
>>> collection
[[40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 37, 40, 40, 40, 40, 40, 37, 40, 39, 40, 40, 40, 40, 39, 40, 35, 40, 40, 40, 37, 40, 40, 40, 35, 39, 40, 40, 40, 40, 39, 39,
39, 39, 38, 39, 40, 39], [40, 40, 40, 39, 40, 39, 40, 40, 40, 40, 40, 40, 40, 40, 39, 40, 39, 40, 39, 40, 40, 39, 40, 40, 40, 39, 38, 38, 40, 40, 40, 40, 40, 38, 39, 39, 40, 40,
40, 40, 39, 40, 39, 39, 40, 37, 40, 33, 40, 36], [40, 38, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 39, 39, 40, 40, 40, 40, 40, 39, 40, 40, 38, 40, 40, 40, 40, 40, 40, 40, 33, 38, 38,
38, 38, 33, 40, 39, 36, 40, 40, 39, 38, 38, 38, 39, 39, 38, 35, 36, 38], [40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 39, 40, 40, 40, 39, 40, 40, 38, 40, 40, 40, 40, 35,
40, 40, 40, 36, 39, 40, 40, 40, 40, 40, 37, 40, 39, 33, 40, 40, 38, 40, 40, 35, 40, 40, 36, 39], [38, 38, 38, 38, 34, 40, 40, 35, 40, 40, 40, 39, 40, 40, 36, 40, 40, 40, 38, 40,
38, 40, 39, 40, 40, 39, 40, 40, 39, 40, 38, 36, 39, 37, 38, 38, 38, 36, 36, 36, 38, 39, 35, 36, 29, 33, 33, 36, 31, 36], [40, 39, 40, 39, 40, 40, 40, 34, 40, 40, 40, 40, 40, 40,
36, 40, 40, 36, 40, 36, 40, 40, 36, 40, 36, 36, 40, 34, 36, 40, 39, 39, 39, 39, 39, 36, 39, 39, 38, 38, 39, 39, 38, 38, 39, 36, 39, 36, 36, 38], [40, 40, 40, 40, 40, 40, 40, 40,
39, 40, 40, 40, 40, 40, 40, 40, 40, 40, 39, 40, 40, 40, 36, 40, 40, 40, 38, 40, 40, 39, 40, 39, 39, 40, 40, 33, 40, 40, 40, 36, 40, 37, 35, 32, 35, 35, 37, 33, 37, 36], [40, 40,
39, 39, 40, 39, 40, 40, 37, 40, 39, 40, 38, 38, 40, 40, 40, 40, 40, 38, 39, 40, 40, 40, 40, 40, 39, 33, 40, 38, 40, 40, 39, 40, 38, 40, 33, 31, 37, 40, 33, 40, 37, 38, 40, 40
, 33, 39, 40], [39, 36, 40, 40, 40, 40, 40, 40, 39, 40, 40, 40, 40, 40, 40, 40, 40, 40, 37, 40, 39, 35, 40, 40, 40, 40, 40, 37, 40, 39, 35, 40, 40, 40, 39, 38, 40, 40, 39, 32, 38
, 36, 35, 27, 36, 37, 34, 39, 27, 37], [39, 39, 38, 35, 39, 39, 36, 39, 39, 39, 39, 36, 37, 39, 39, 39, 39, 39, 39, 39, 38, 39, 39, 35, 39, 37, 39, 38, 37, 35, 36, 35, 36, 36, 31
, 36, 36, 35, 35, 35, 36, 34, 33, 33, 34, 34, 33, 33, 32, 33]]
```

π

How to turn a list of lists into a numpy matrix

```
, 36, 35, 27, 36, 37, 34, 39, 27, 37], [39, 39, 38, 35, 39, 39, 36, 39, 39, 39, 39, 36,
, 36, 36, 35, 35, 35, 36, 34, 33, 33, 34, 34, 33, 33, 32, 33]]
>>> import numpy
>>> dataMatrix = numpy.matrix(collection)
>>> dataMatrix
matrix([[40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 37, 40, 40, 40,
         40, 40, 37, 40, 39, 40, 40, 40, 40, 39, 40, 35, 40, 40, 40, 37, 40,
         40, 40, 35, 39, 40, 40, 40, 40, 39, 39, 39, 39, 38, 39, 40, 39],
        [40, 40, 40, 39, 40, 39, 40, 40, 40, 40, 40, 40, 40, 40, 39, 40, 39,
         40, 39, 40, 40, 39, 40, 40, 40, 39, 38, 38, 40, 40, 40, 40, 40, 38,
         39, 39, 40, 40, 40, 40, 39, 40, 39, 39, 40, 37, 40, 33, 40, 36],
        [40, 38, 40, 40, 40, 40, 40, 40, 40, 40, 40, 39, 39, 40, 40, 40, 40,
         40, 39, 40, 40, 38, 40, 40, 40, 40, 40, 40, 40, 33, 38, 38, 38, 38,
         33, 40, 39, 36, 40, 40, 39, 38, 38, 38, 39, 39, 38, 35, 36, 38],
        [40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 40, 39, 40, 40, 40,
         39, 40, 40, 38, 40, 40, 40, 40, 35, 40, 40, 40, 36, 39, 40, 40, 40,
         40, 40, 37, 40, 39, 33, 40, 40, 38, 40, 40, 35, 40, 40, 36, 39],
        [38, 38, 38, 38, 34, 40, 40, 35, 40, 40, 40, 39, 40, 40, 36, 40, 40,
         40, 38, 40, 38, 40, 39, 40, 40, 39, 40, 40, 39, 40, 38, 36, 39, 37,
         38, 38, 38, 36, 36, 36, 38, 39, 35, 36, 29, 33, 33, 36, 31, 36],
        [40, 39, 40, 39, 40, 40, 40, 34, 40, 40, 40, 40, 40, 40, 36, 40, 40,
         36, 40, 36, 40, 40, 36, 40, 36, 36, 40, 34, 36, 40, 39, 39, 39, 39,
         39, 36, 39, 39, 38, 38, 39, 39, 38, 38, 39, 36, 39, 36, 36, 38],
        [40, 40, 40, 40, 40, 40, 40, 40, 39, 40, 40, 40, 40, 40, 40, 40, 40,
         40, 39, 40, 40, 40, 36, 40, 40, 40, 38, 40, 40, 39, 40, 39, 39, 40,
         40, 33, 40, 40, 40, 36, 40, 37, 35, 32, 35, 35, 37, 33, 37, 36],
        [40, 40, 39, 39, 40, 39, 40, 40, 37, 40, 39, 40, 38, 38, 40, 40, 40,
         40, 40, 40, 38, 39, 40, 40, 40, 40, 40, 39, 33, 40, 38, 40, 40, 39,
         40, 38, 40, 33, 31, 37, 40, 33, 40, 37, 38, 40, 40, 33, 39, 40],
        [39, 36, 40, 40, 40, 40, 40, 40, 39, 40, 40, 40, 40, 40, 40, 40, 40,
         40, 37, 40, 39, 35, 40, 40, 40, 40, 40, 37, 40, 39, 35, 40, 40, 40,
         39, 38, 40, 40, 39, 32, 38, 36, 35, 27, 36, 37, 34, 39, 27, 37],
        [39, 39, 38, 35, 39, 39, 36, 39, 39, 39, 39, 36, 37, 39, 39, 39, 39,
         39, 39, 39, 38, 39, 39, 35, 39, 37, 39, 38, 37, 35, 36, 35, 36, 36,
         31, 36, 36, 35, 35, 35, 36, 34, 33, 33, 34, 34, 33, 33, 32, 33]])
>>>
```

And how to generate a matrix of just means

```
        [39, 39, 38, 35, 39, 39, 36, 39, 39, 39, 39, 36, 37, 39, 39, 39, 39,
         39, 39, 39, 38, 39, 39, 35, 39, 37, 39, 38, 37, 35, 36, 35, 36, 36,
         31, 36, 36, 35, 35, 35, 36, 34, 33, 33, 34, 34, 33, 33, 32, 33]])
>>> dataMeans = dataMatrix.mean(axis = 1)
>>> dataMeans
matrix([[ 39.4 ],
        [ 39.36],
        [ 38.88],
        [ 39.26],
        [ 37.74],
        [ 38.42],
        [ 38.7 ],
        [ 38.68],
        [ 38.2 ],
        [ 36.58]])
>>> dataMeans = dataMeans.transpose()
>>> dataMeans
matrix([[ 39.4 ,  39.36,  38.88,  39.26,  37.74,  38.42,  38.7 ,  38.68,
          38.2 ,  36.58]])
>>>
```

A function to get a matrix of qualities

```
3  readLengthMinimum = 100
4  analysisLength = 50
5
6  def getQualityMeanMatrix(samFile, readLengthMinimum, analysisLength, analysisPositionStart = 0, dataLimit = False):
7      assert readLengthMinimum >= analysisLength + analysisPositionStart, "Analysis length must be shorter than or equa
8      import samReader2
9      import numpy
10     samLines = samReader2.readSAMFileLines(samFile)
11     qualityList = []
12     for line in samLines:
13         if len(line.quality.qualityArray) >= readLengthMinimum:
14             qualityList.append(line.quality.qualityArray[:analysisLength + analysisPositionStart])
15         if dataLimit:
16             if len(qualityList) >= dataLimit:
17                 break
18     qualityMatrix = numpy.matrix(qualityList)
19     meanMatrix = qualityMatrix.mean(axis = 1)
20     return meanMatrix#.transpose()
21
```

A function to make a histogram

```
22  def makeHistogram(meanMatrix):
23      import matplotlib.pyplot as plt
24      plt.hist(meanMatrix, bins = 41)
25      plt.title("Average quality score in the first 50 bases")
26      plt.xlabel("Phred score")
27      plt.ylabel("Frequency")
28      #plt.show()
29      plt.savefig("qualityByRead.png")
30
31  qualityMeanMatrix = getQualityMeanMatrix("sampleData.sam", 100, 50)
32  print("Mean matrix shape:")
33  print(qualityMeanMatrix.shape)
34  makeHistogram(qualityMeanMatrix)
```

Result?

```
C:\Users\mweinstein\Documents\pythonClass2>python histogramMaker.py
Read 500172 lines
Mean matrix shape:
(500000, 1)
```