ME1 Computing

Provide feedback (anonymously) at:

www.menti.com

with code **44 88 7**

List of variables

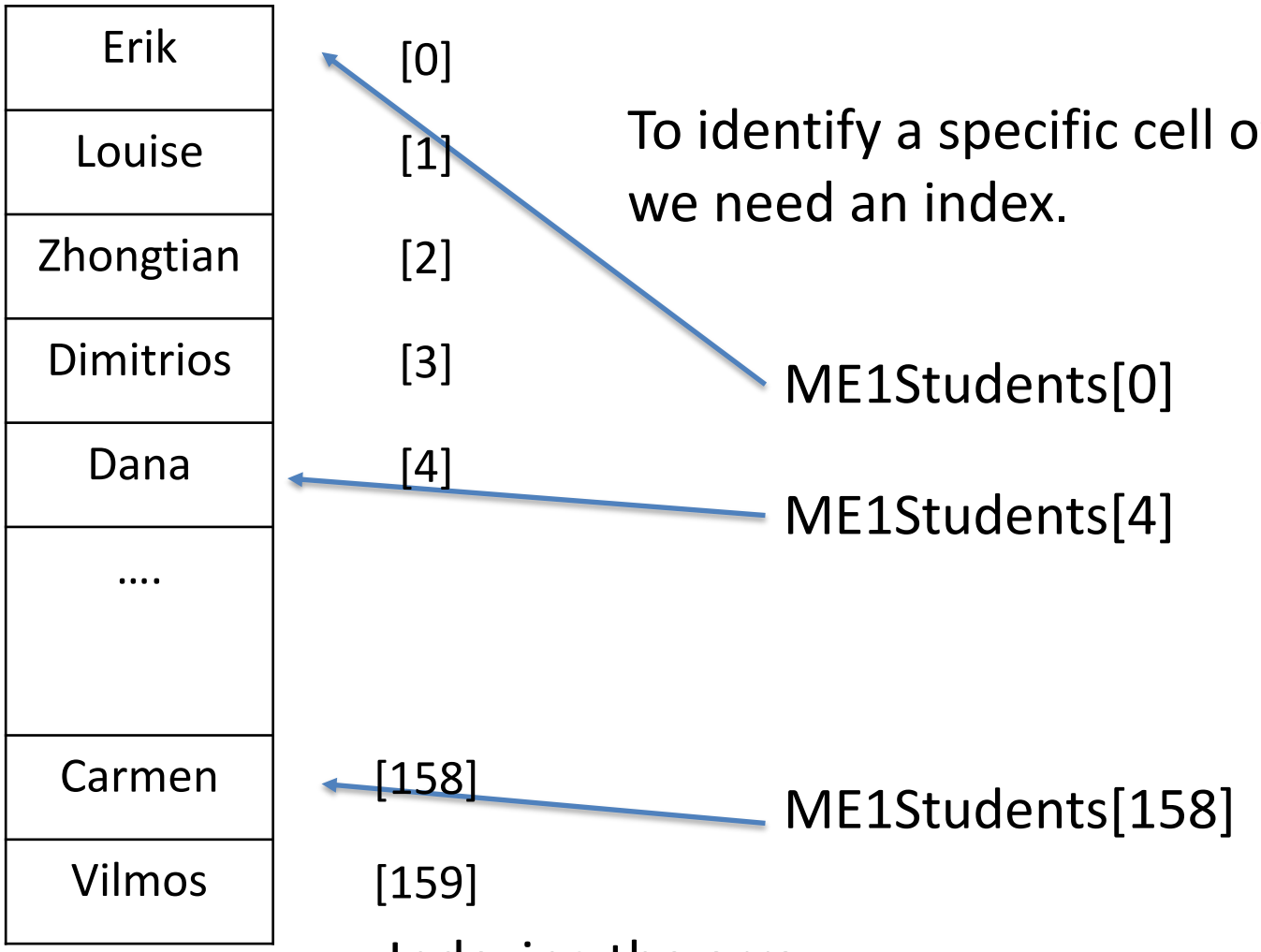
List of students in this room:

Erik
Louise
Zhongtian
Dimitrios
Dana
….
Carmen
Vilmos

Student1 = ‘Erik’
Student2 = ‘Louise’
Student3 = ‘Zhongtian’
Student4 = ‘Dimitrios’
Student5 = ‘Dana’
and so on….
Student159 = ‘Carmen’
Student160 = ‘Vilmos’

List indexing

ME1Students



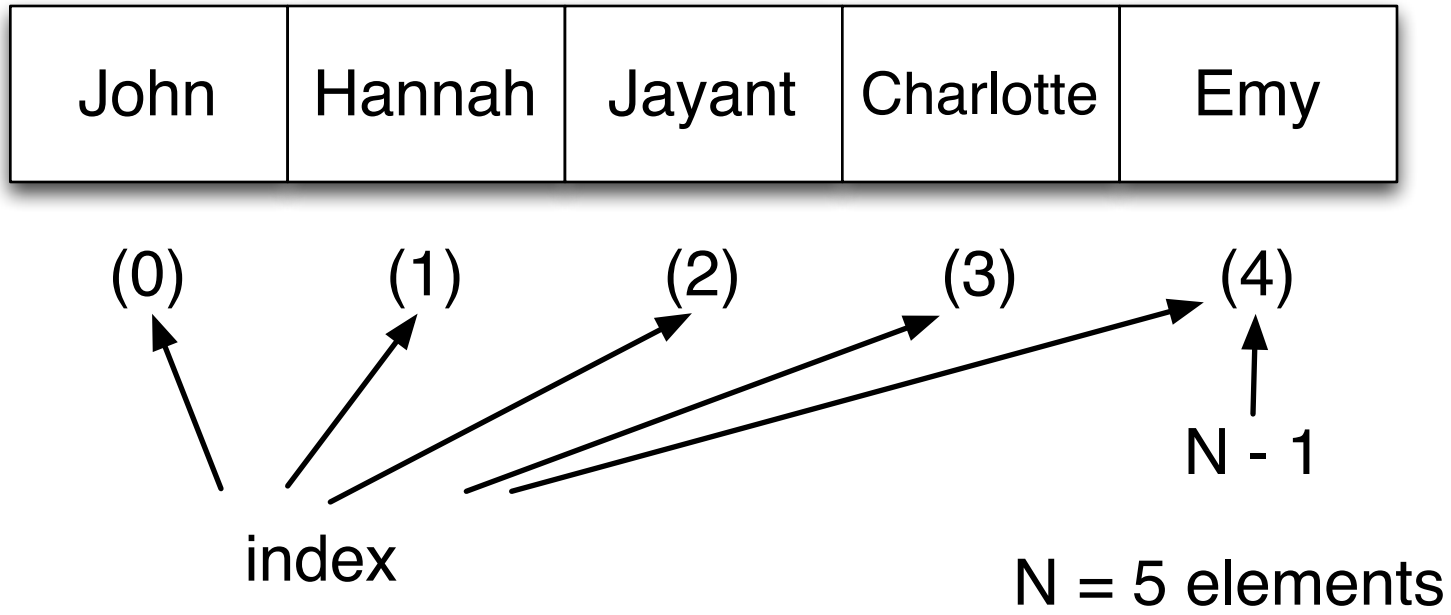
An array is a variable made of many cells.

To identify a specific cell of the array, we need an index.

Indexing the array

List indexing

BrightStudents




```
N = len(BrightStudents)
```

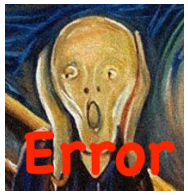
N elements → Index range from 0 to N-1

List indexing

ME1Students

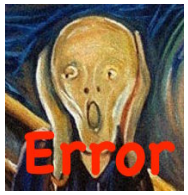
Erik	[0]
Louise	[1]
Zhongtian	[2]
Dimitrios	[3]
Dana	[4]
….	
Carmen	[158]
Vilmos	[159]

```
FirstPrize = ME1Students[160]
```



An array has a length

```
FirstPrize = ME1Students[2.5]
```

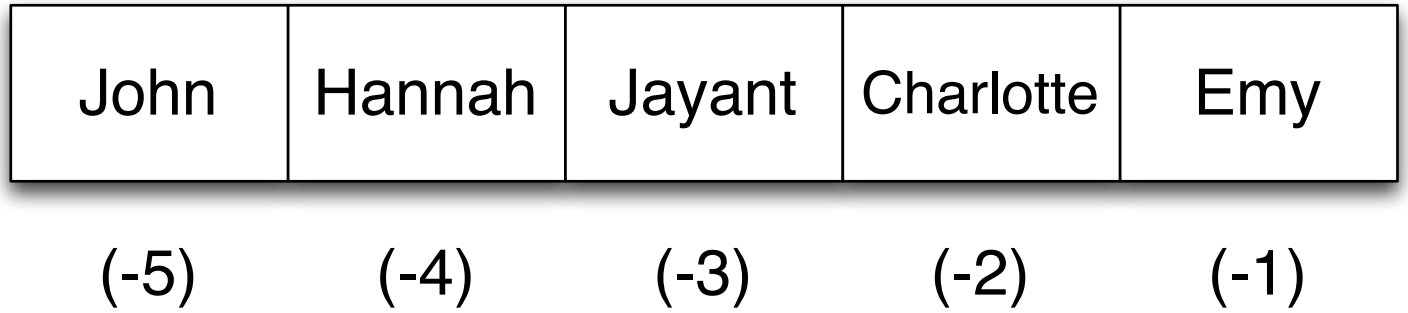


Indexing an array must be with an integer number

```
index = 2
FirstPrize = ME1Students[index]
index = index + 2
FirstPrize = ME1Students[index]
```

List indexing

BrightStudents



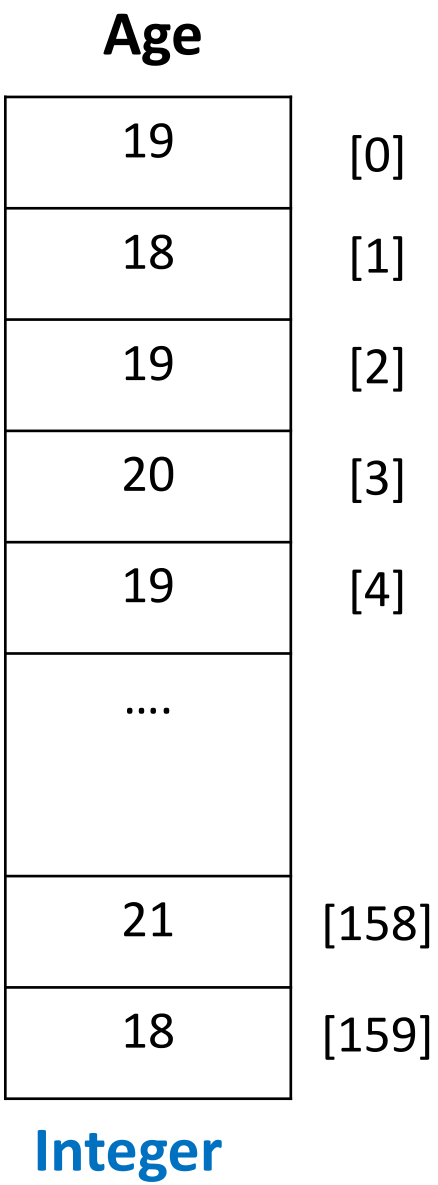
Indices can be negative too: counting backward from the end

Index range from -1 to -N

Arrays can be of any type

	Name	Age	Height
[0]	Erik	19	1.58
[1]	Louise	18	1.89
[2]	Zhongtian	19	1.70
[3]	Dimitrios	20	1.65
[4]	Dana	19	1.75
	....	....	....
[158]	Carmen	21	1.71
[159]	Vilmos	18	1.64
	String	**Integer**	**Real**

Creating lists



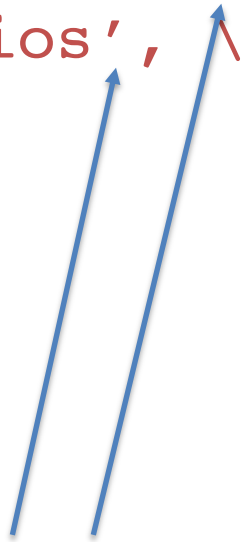

```
Age = [19,18,19,20,19,     ,21,18]
```

Creating lists

Name

Erik	[0]
Louise	[1]
Zhongtian	[2]
Dimitrios	[3]
Dana	[4]
….	
Carmen	[158]
Vilmos	[159]

```
Name=['Eric','Louise', \
 'Zhongtian', 'Dimitrios', \
 'Carmen', 'Vilmos']
```



Note how to break a command into multiple lines

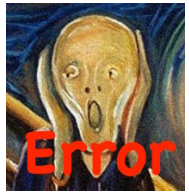
Lists operations: alter values in elements

BrightStudents

John	Hannah	Jayant	Charlotte	Emy
(0)	(1)	(2)	(3)	(4)

```
BrightStudents[2] = 'Pauline'
```

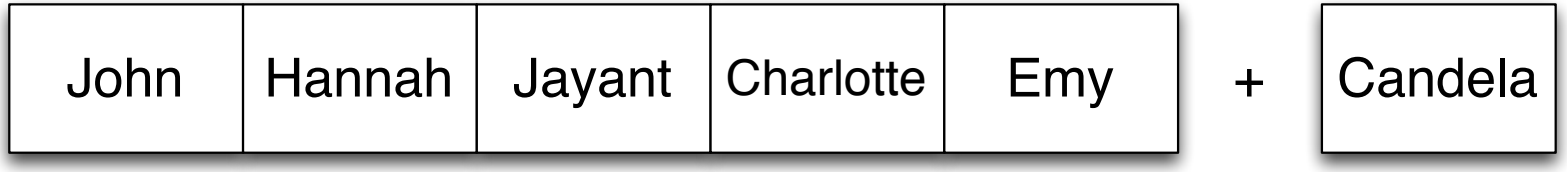
```
BrightStudents[5] = 'Candela'
```



We cannot add / append new elements this way

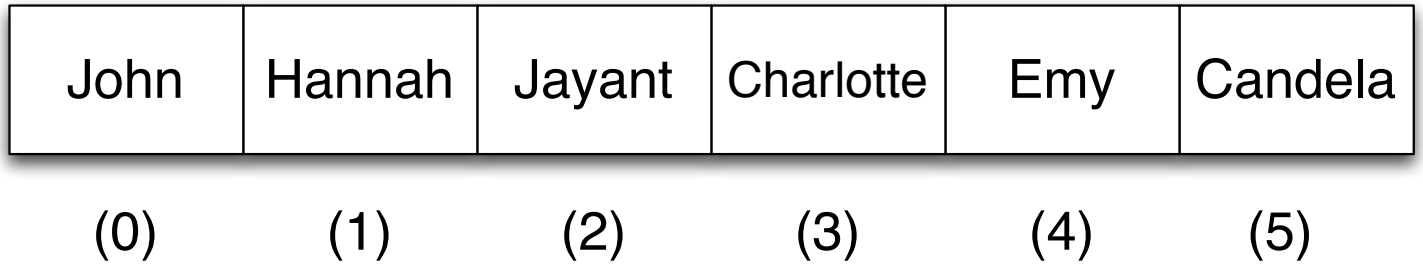
Lists concatenation

BrightStudents

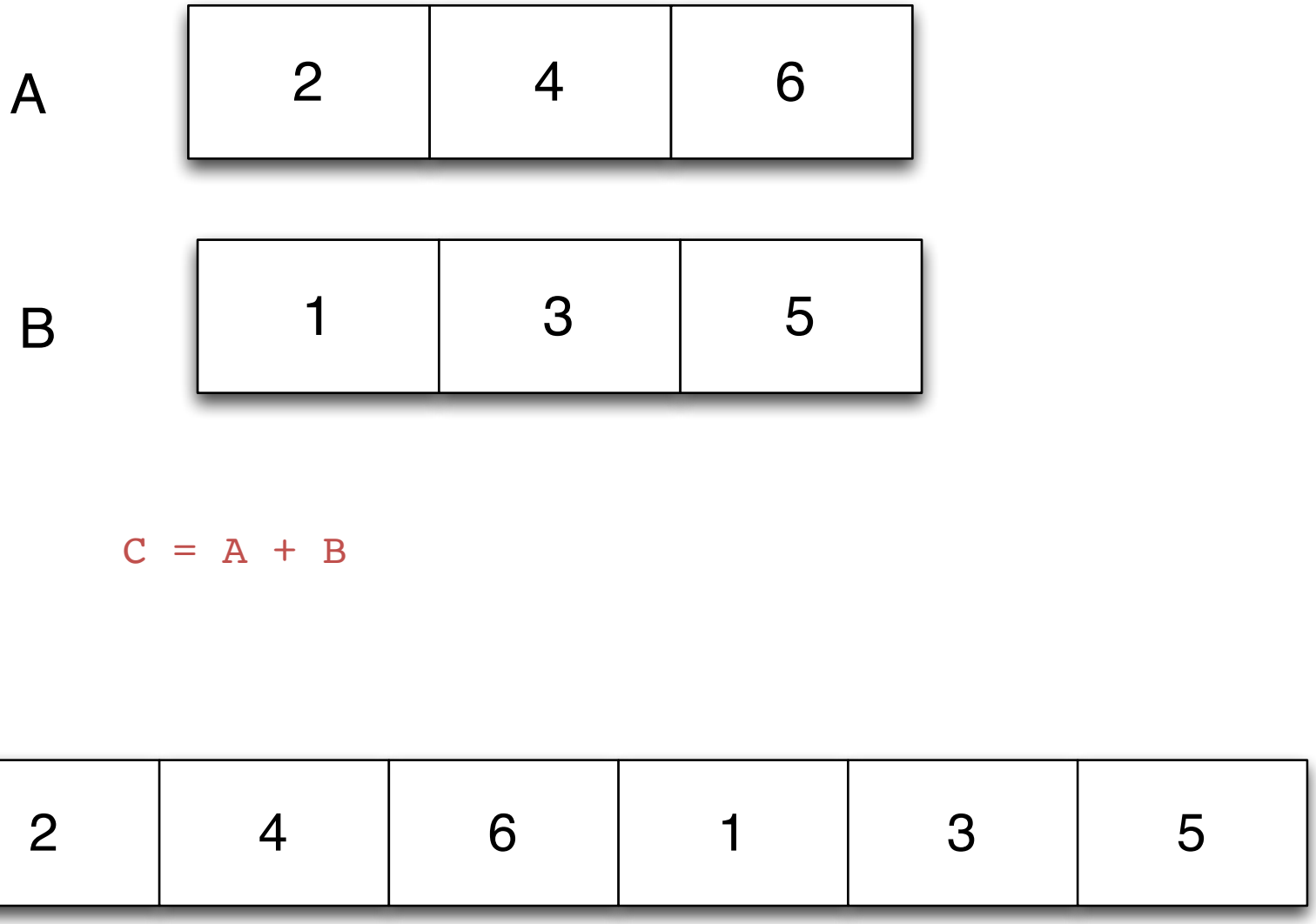



```
BrightStudents = BrightStudents + ['Candela']
```

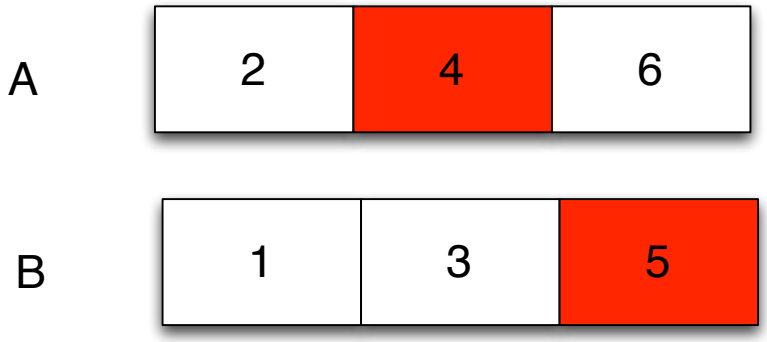
BrightStudents



Lists concatenation



Lists operations



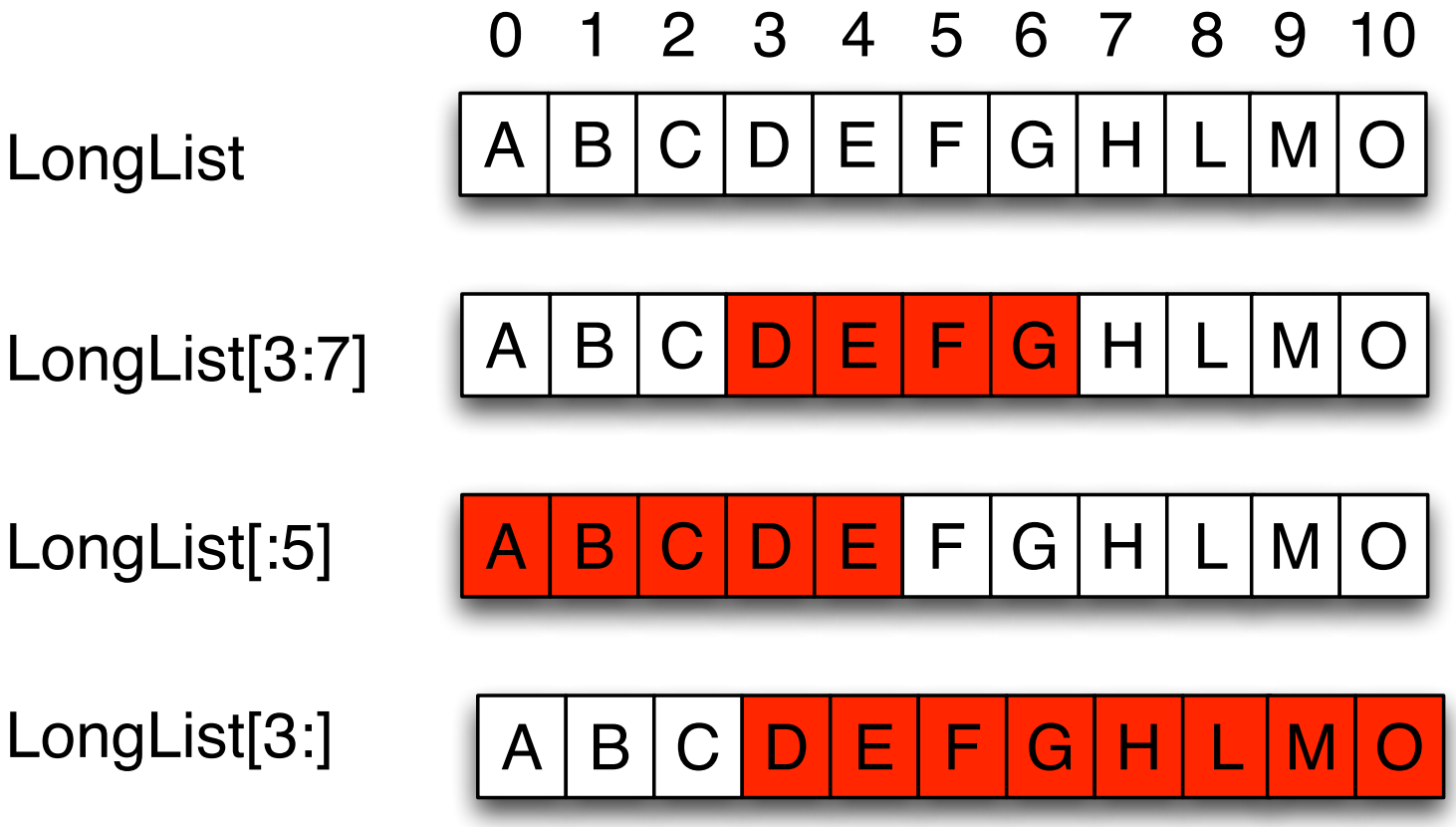

```
C = A[1] + B[2]
```

C | 9

```
C = A[1] - B[2]
```

C | -1

Lists slicing



Strings and Lists

```
Sentiment = 'ILOVEYOU'
```

Sentiment


```
Sentiment[1]
```

L

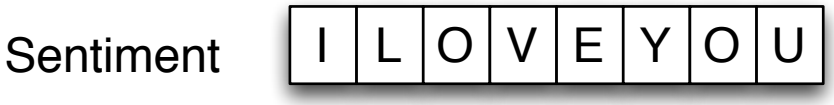
```
Sentiment[4]
```

E

Mutable - Immutable

```
Sentiment = 'ILOVEYOU'
```

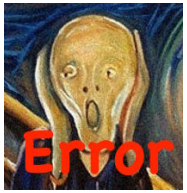
```
Sentiment = ['I','L','O','V','E','Y','O','U']
```




```
Sentiment[1]
```

String

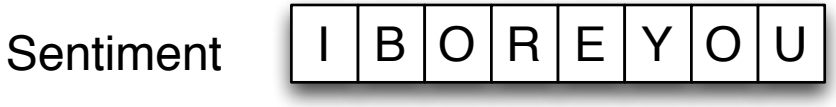
```
Sentiment[1] = 'B'
Sentiment[3] = 'R'
```



Immutable

List

```
Sentiment[1] = 'B'
Sentiment[3] = 'R'
```

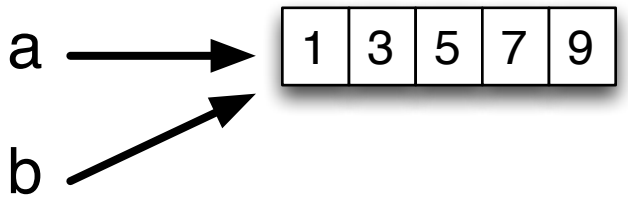


Mutable

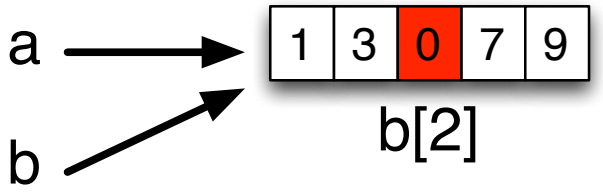
Aliasing

```
a = [1,3,5,7,9]
```

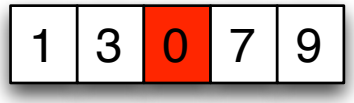
```
b = a
```




```
b[2] = 0
```




```
print(a)
```



Be very careful with aliasing

Traversing a list

```
Royals = ['Elisabeth','Charles','Camilla','William', \
'Kate','Henry','Meghan','George','Louis','Charlotte']
```

```
print(Royals[0])
print(Royals[1])
print(Royals[2])
print(Royals[3])
print(Royals[4])
print(Royals[5])
print(Royals[6])
print(Royals[7])
print(Royals[8])
print(Royals[9])
```

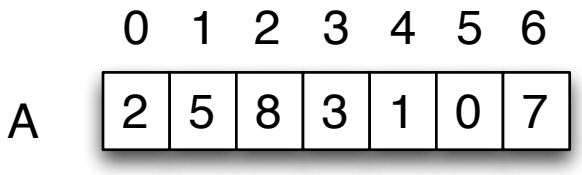
```
for members in Royals:
    print(members)
```

Indent four spaces

members

Traversing a list

```
A = [2, 5, 8, 3, 1, 0, 7]
```

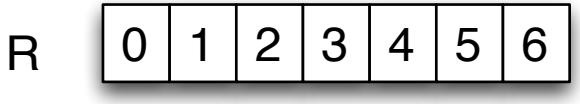



```
A[0] = A[0] * 2
A[1] = A[1] * 2
A[2] = A[2] * 2
A[3] = A[3] * 2
A[4] = A[4] * 2
A[5] = A[5] * 2
A[6] = A[6] * 2
```

```
for values in A:
    ??? =  values * 2
```

N = 7

```
R = range(0,N)
```



range(0,N) generated integers from 0 to N-1

Traversing a list


```
for index in R:
    A[index] =  A[index] * 2
```

Generating a list

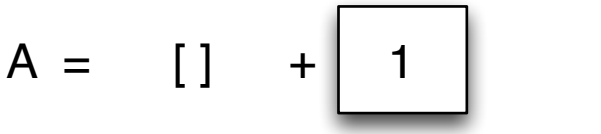
concatenation

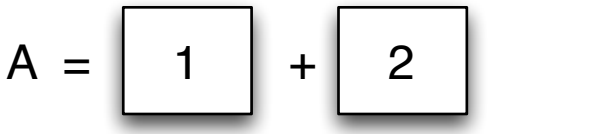
Generate a list of integer numbers from 1 to 100

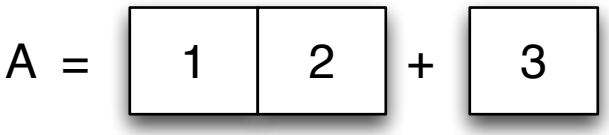
```
R = range(1,101)

A = []
```

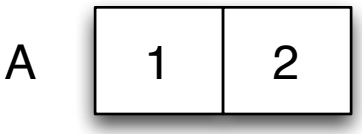
```
for i in R:
    A = A + [i]
```



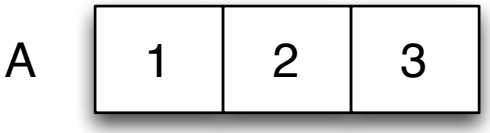




i = 1



i = 2



i = 3

Key points to remember

- Lists are sequences of values
- Values in a list are accessible by addressing their position within the list
- Lists are mutable
- Strings can be seen as lists of characters, but they are immutable
- New elements are appended to an existing list