Smith College
Computer Science

Lecture Notes CSC111

Week 4

Dominique Thiébaut
dthiebaut@smith.edu

End Chapter 3

Skip Chapters 4

Start Chapter 5

Arithmetic in Binary with Logic…

```
      1234
+     3189
----------
=     4423
```

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
…

Arithmetic in Binary with Logic…

```
      1234
+     3189
----------
=     4423
```

```
     1 1 1 1
       1011
+      1101
-----------
=     11000
```

```
  0
  1
 10
 11
100
```

Arithmetic in Binary with Logic…

```
                         1 1 1 1
     1234                 1011
+    3189            +    1101
---------            ---------
=    4423            =   11000
```

```
  0
  1
 10
 11
100
```

Arithmetic in Binary with Logic…

```
         1234
+        3189
-------------
=        4423
```

```
     1 1 1 1
        1011
+       1101
------------
=      11000
```

```
  0
  1
 10
 11
100
```

```
    0
+   0
-----
=   0
```

```
    0
+   1
-----
=   1
```

```
    1
+   0
-----
=   1
```

```
   1
    1
+   1
-----
=  10
```

Arithmetic in Binary with Logic…

```
           1234
      +    3189
      ---------
      =    4423
```

```
        1 1 1 1
           1011
      +    1101
      ---------
      =   11000
```

```
  0
  1
 10
 11
100
```

```
     0          0          1          1
+    0     +    1     +    0     +    1
------     ------     ------     ------
=   00     =   01     =   01     =   10
```

Arithmetic in Binary with Logic…

```
     1234
+    3189
---------
=    4423
```

```
  1 1 1 1
     1011
+    1101
---------
=   11000
```

```
  0
  1
 10
 11
100
```

```
    0          0          1          1
+   0      +   1      +   0      +   1
-----      -----      -----      -----
=  00      =  01      =  01      =  10
   cs         cs         cs         cs
```

carry → c, sum → s

Arithmetic in Binary with Logic…

```
      0           0           1           1
  +   0       +   1       +   0       +   1
  -----       -----       -----       -----
  =  00       =  01       =  01       =  10
     cs          cs          cs          cs
```

Arithmetic in Binary with Logic…

```
    0          0          1          1
+   0      +   1      +   0      +   1
-----      -----      -----      -----
=  00      =  01      =  01      =  10
   cs         cs         cs         cs
```

d1	d2	c s
0	0	0 0
0	1	0 1
1	0	0 1
1	1	1 0

Arithmetic in Binary with Logic…

```
     0          0          1          1
+    0     +    1     +    0     +    1
------     ------     ------     ------
=   00     =   01     =   01     =   10
    cs         cs         cs         cs
```

d1	d2	c s
0	0	0 0
0	1	0 1
1	0	0 1
1	1	1 0

d1	d2	c s
F	F	F F
F	T	F T
T	F	F T
T	T	T F

Arithmetic in Binary with Logic…

```
      0          0          1          1
+     0     +    1     +    0     +    1
-------     -------    -------    -------
=    00     =   01     =   01     =   10
     cs         cs         cs         cs
```

d1	d2	c s
0	0	0 0
0	1	0 1
1	0	0 1
1	1	1 0

d1	d2	c s
F	F	F F
F	T	F T
T	F	F T
T	T	T F

```
c = d1   ?   d2
```

Arithmetic in Binary with Logic…

```
     0          0          1          1
+    0     +    1     +    0     +    1
------     ------     ------     ------
=   00     =   01     =   01     =   10
    cs         cs         cs         cs
```

d1	d2	c s
0	0	0 0
0	1	0 1
1	0	0 1
1	1	1 0

d1	d2	c s
F	F	F F
F	T	F T
T	F	F T
T	T	T F

```
c = d1 and d2
```

Arithmetic in Binary with Logic…

```
    0          0          1          1
+   0      +   1      +   0      +   1
-----      -----      -----      -----
=  00      =  01      =  01      =  10
   cs         cs         cs         cs
```

d1	d2	c s
0	0	0 0
0	1	0 1
1	0	0 1
1	1	1 0

d1	d2	c s
F	F	F F
F	T	F T
T	F	F T
T	T	T F

```
c = d1 and d2
s = d1 and not d2
    or
    not d1 and d2
  =
```

Arithmetic in Binary with Logic…

```
      0          0          1          1
  +   0      +   1      +   0      +   1
  ------     ------     ------     ------
  =  00      =  01      =  01      =  10
     cs         cs         cs         cs
```

d1	d2	c s
0	0	0 0
0	1	0 1
1	0	0 1
1	1	1 0

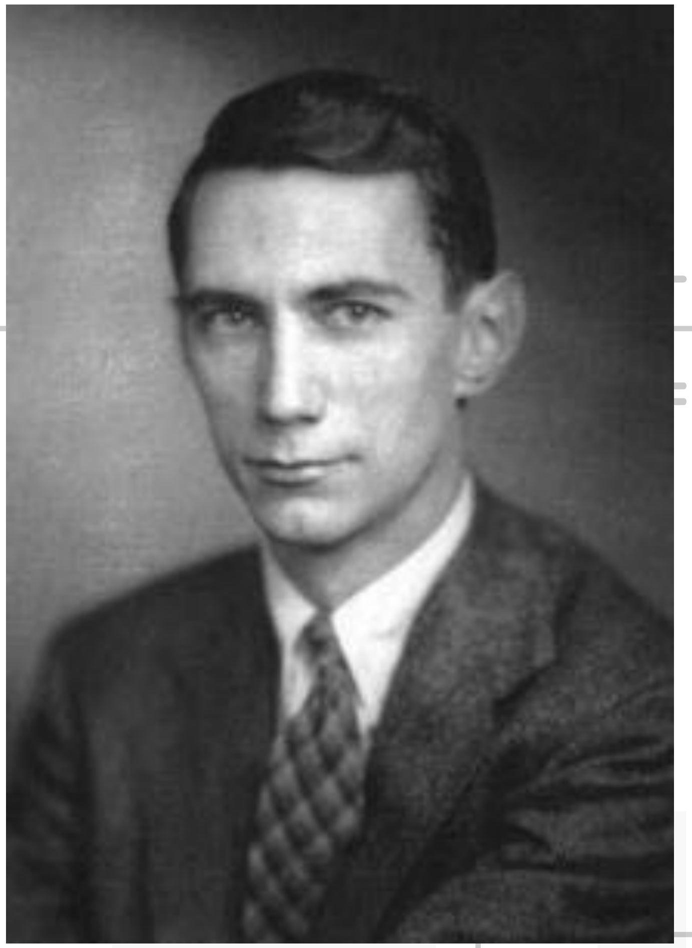
d1	d2	c s
F	F	F F
F	T	F T
T	F	F T
T	T	T F

```
c = d1 and d2
s = d1 and not d2
    or
    not d1 and d2
  = d1 xor d2
```

Arithmetic in Binary with Logic…

```
    0        1        1
+   0        0    +   1
-----    -----    -----
=  00       01    =  10
   cs       cs       cs
```

Claude Shannon's Master's Thesis



d1	d2	c s
0	0	0 0
0	1	0 1
1	0	0 1
1	1	1 0

F	F	F F
F	T	F T
T	F	F T
T	T	T F

```
c = d1 and d2
s = d1 and not d2
    or
    not d1 and d2
  = d1 xor d2
```

The Lesson

- Additions in binary can be done with logic operators
- Subtraction, multiplication, division can be done with logic operators
- Logic operators can be easily implemented with transistors
- transistors can be miniaturized
- transistors work at the speed of electricity (2/3 speed of light)
- Billions of transistors can be manufactured in a square inch
- Computers are deterministic machines that can be made small and extremely fast

What you should remember…

- A **bit** is a device that stores either 1 or 0

- A bit is a device that stores either 1 or 0
- By extension, a bit is either 1 or 0

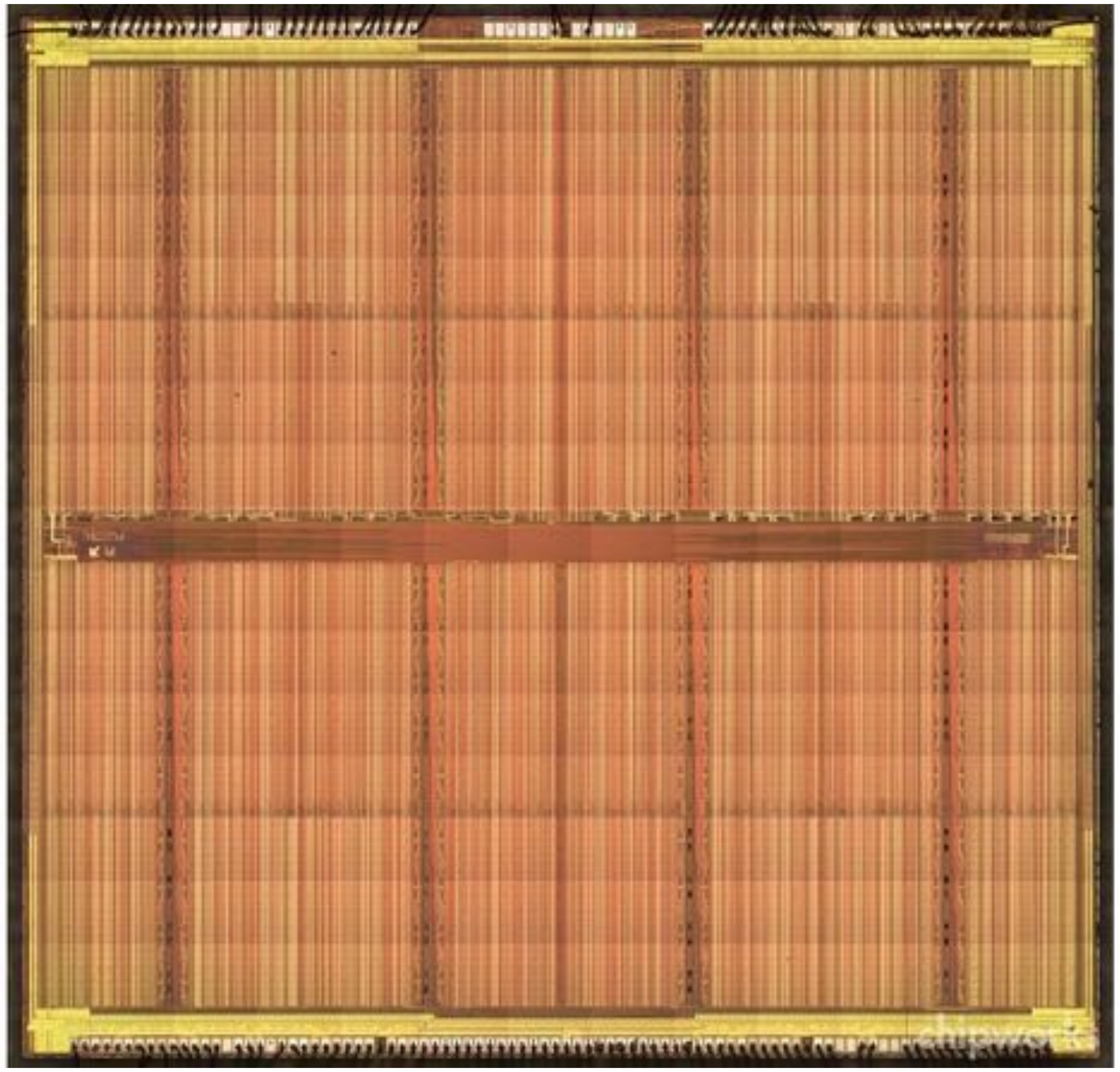
- A bit is a device that stores either 1 or 0
- By extension, a bit is either 1 or 0
- A bit is a unit of information

- A bit is a device that stores either 1 or 0
- By extension, a bit is either 1 or 0
- A bit is a unit of information
- 2 bits take on 1 of 4 states: 00, 01, 10, 11

- A bit is a device that stores either 1 or 0
- By extension, a bit is either 1 or 0
- A bit is a unit of information
- 2 bits take on 1 of 4 states: 00, 01, 10, 11
- 3 bits: 000, 001, 010, 011, 100, 101, 110, 111

- A bit is a device that stores either 1 or 0
- By extension, a bit is either 1 or 0
- A bit is a unit of information
- 2 bits take on 1 of 4 states: 00, 01, 10, 11
- 3 bits: 000, 001, 010, 011, 100, 101, 110, 111
- 8 bits = 1 byte
  00000000, 00000001, … to 11111111
  256 possible combinations of 0s and 1s

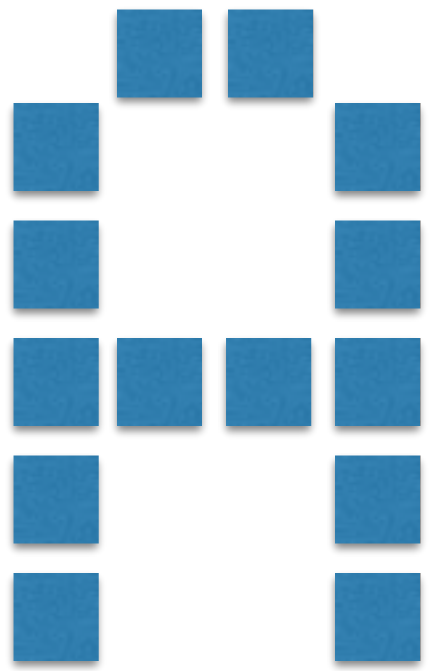
GoldStar
GM71C4400AJ70
9326 KOREA
GM71C1000J70
9324 KOREA
1
30
http://www.DeviceLog.com


chipworks

A

Character

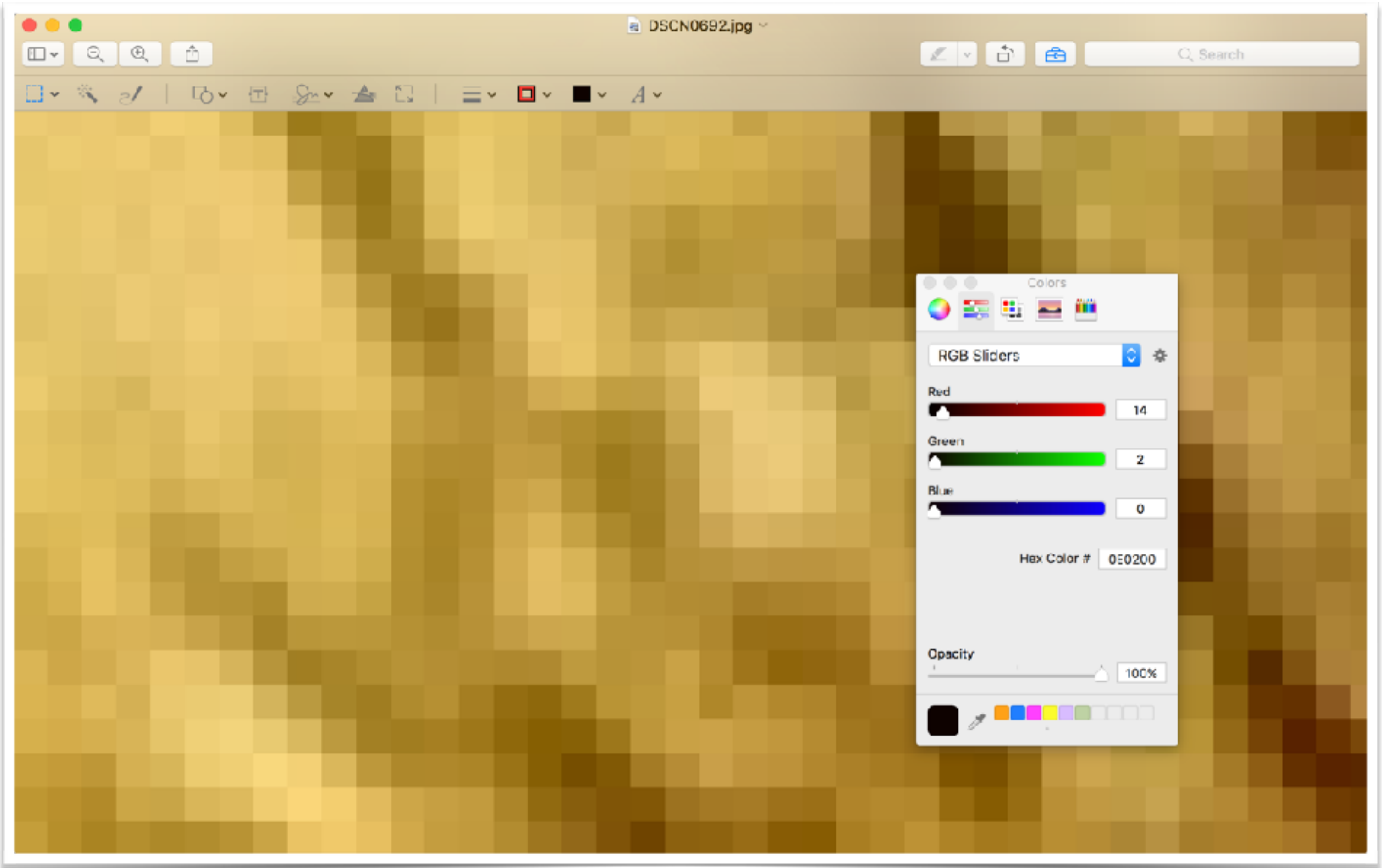
01000001



0110
1001
1001
1111
1001
1001

Pixel

RED	GREEN	BLUE
10001000	**01101010**	**00001000**



RGB Sliders

Red 14

Green 2

Blue 0

Hex Color # 0E0200

Opacity 100%

End Chapter 3

Skip Chapters 4

Start Chapter 5

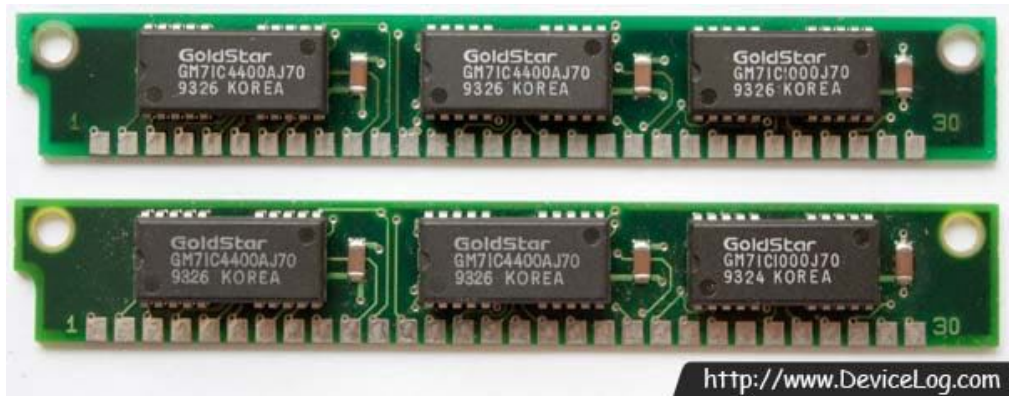
Chapter 5 in Zelle

Indexing in Strings

Indexing in Lists

Splitting Strings into Lists

String Methods


http://www.DeviceLog.com

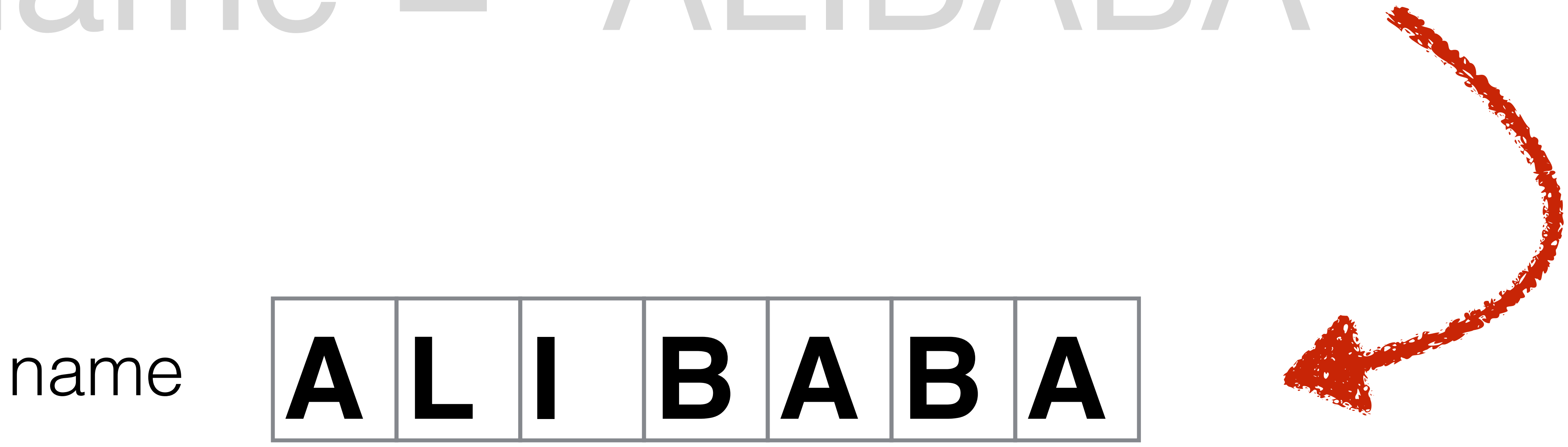
We like to keep information in numbered boxes in memory

Data is kept in collections of "things"

For example: strings
Strings are collections of characters

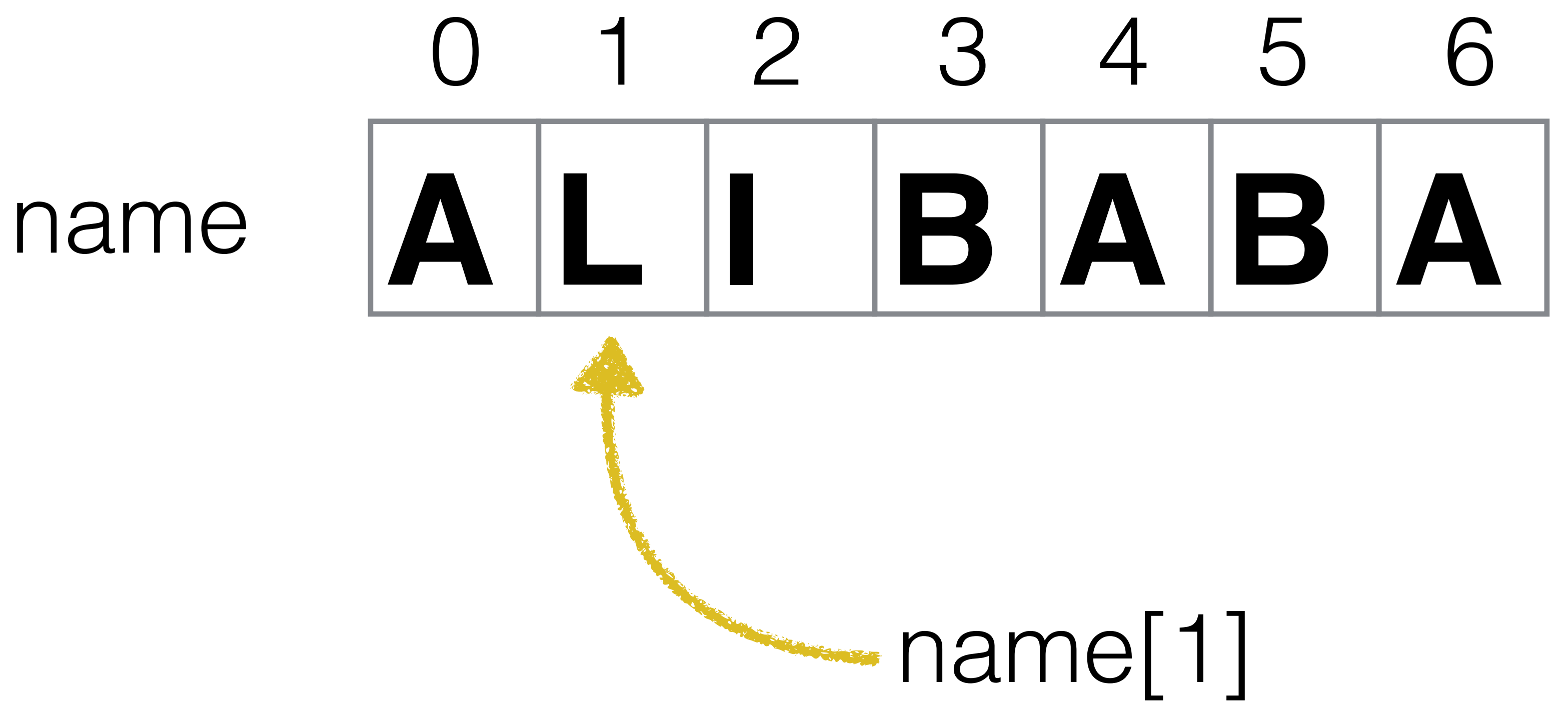
```
name = "ALIBABA"
```

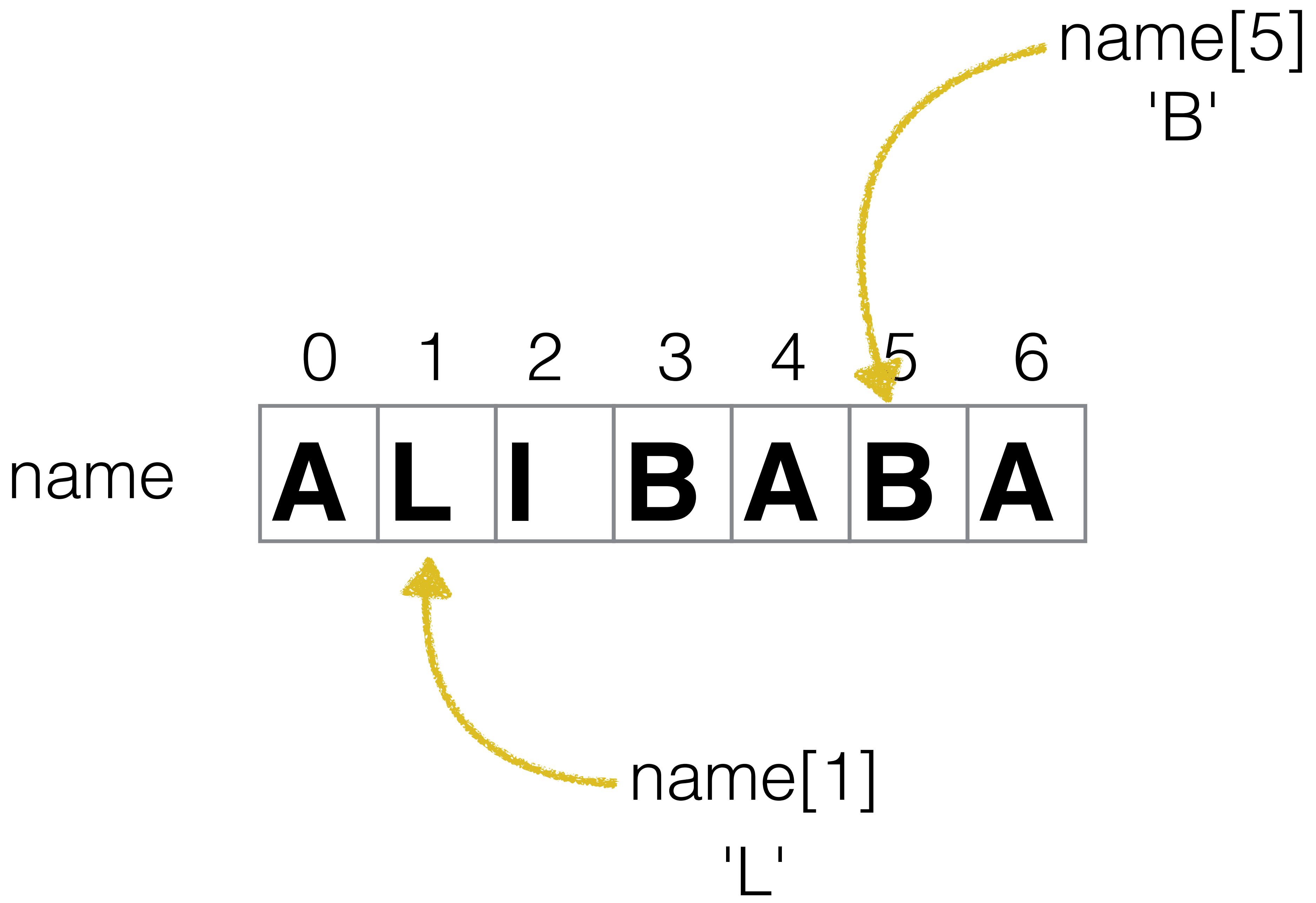
name = "ALIBABA"

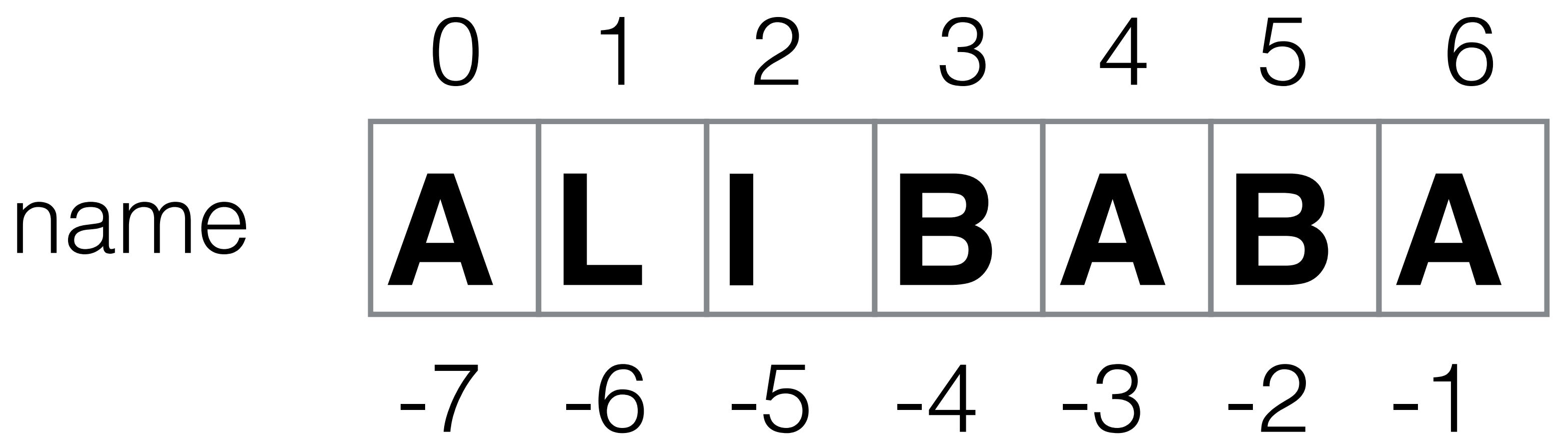


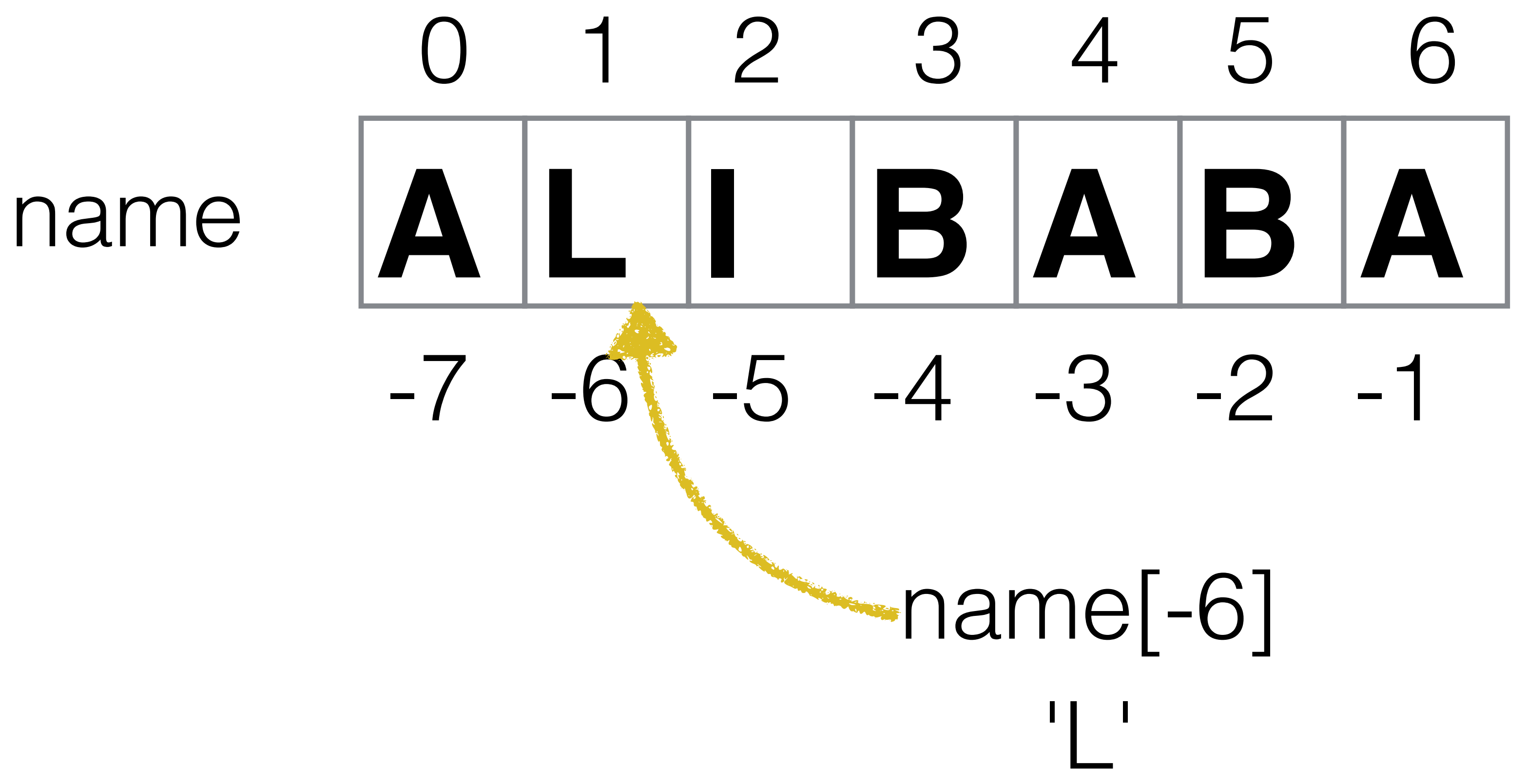
Important conceptual change in the way we look at string

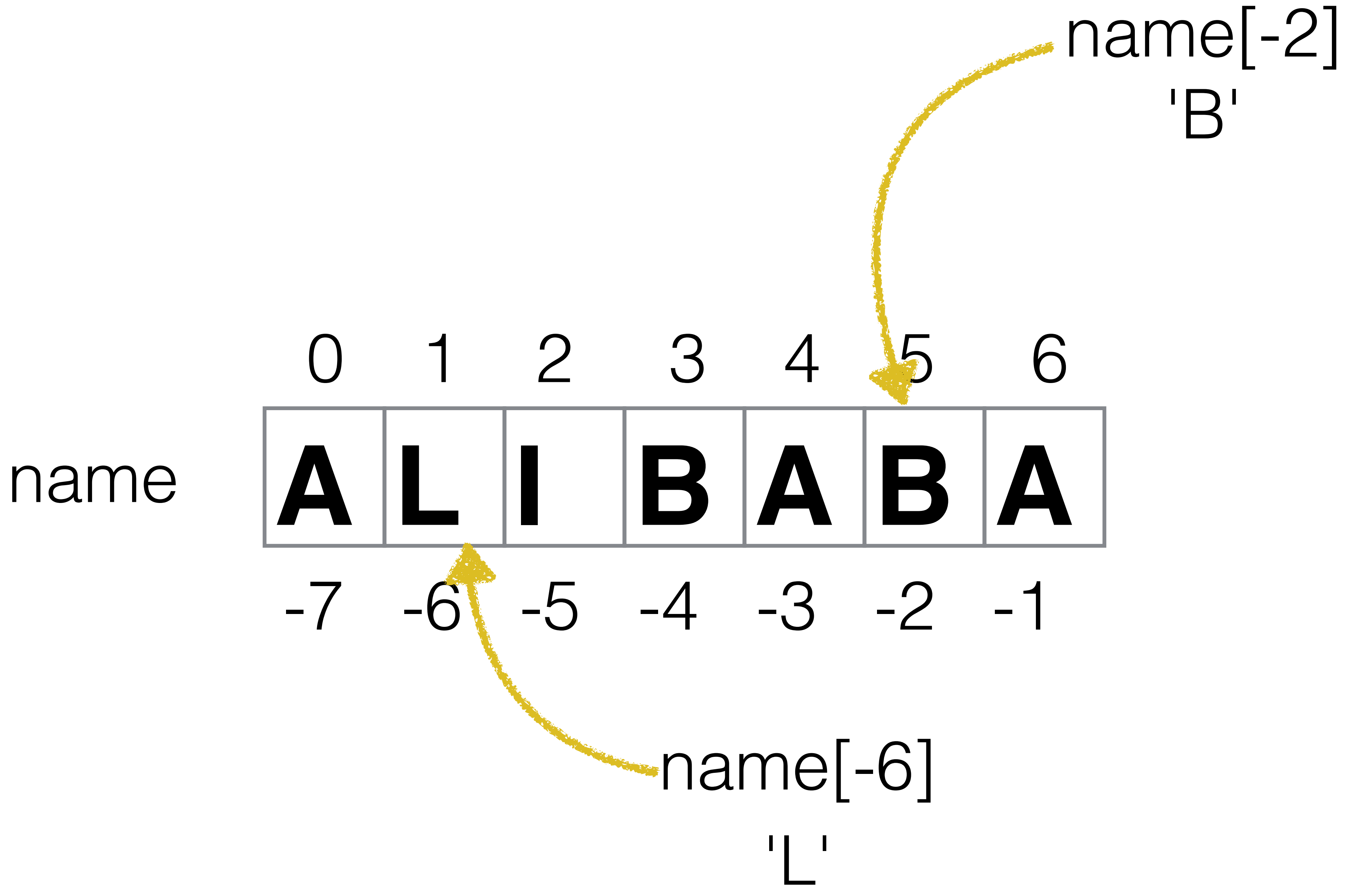
	0	1	2	3	4	5	6
name	**A**	**L**	**I**	**B**	**A**	**B**	**A**


0
1
2
3
4
5
6
name
A
L
I
B
A
B
A
name[1]


name[5]
'B'
0
1
2
3
4
5
6
name
A
L
I
B
A
B
A
name[1]
'L'

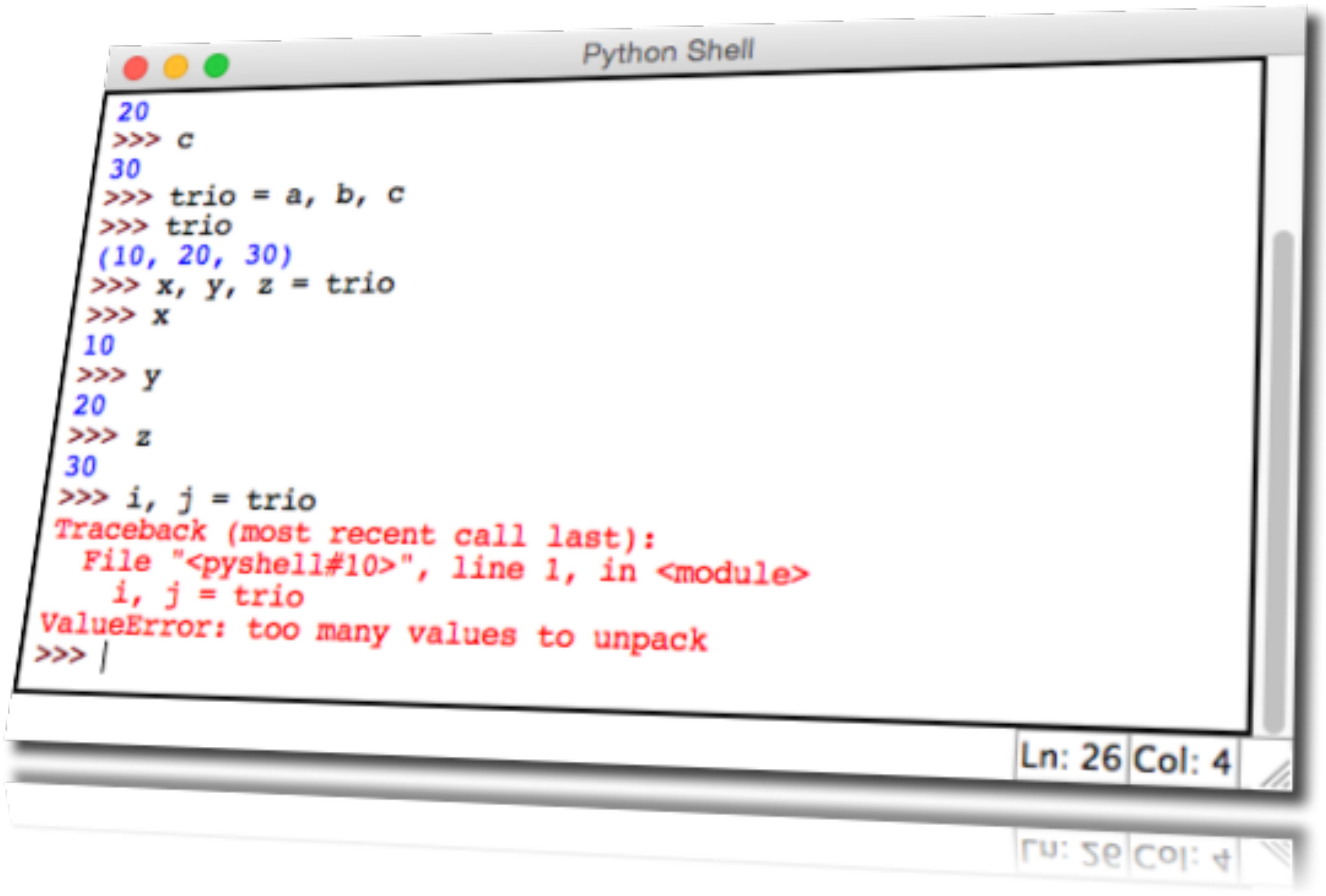

0
1
2
3
4
5
6
name
A
L
I
B
A
B
A
-7
-6
-5
-4
-3
-2
-1


0 1 2 3 4 5 6
name
A L I B A B A
-7 -6 -5 -4 -3 -2 -1
name[-6]
'L'


name[-2]
'B'
0 1 2 3 4 5 6
name
A L I B A B A
-7 -6 -5 -4 -3 -2 -1
name[-6]
'L'

There are two different ways to access the last character of a string. Which are they?

Demo Time!



Python Shell

```
20
>>> c
30
>>> trio = a, b, c
>>> trio
(10, 20, 30)
>>> x, y, z = trio
>>> x
10
>>> y
20
>>> z
30
>>> i, j = trio
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    i, j = trio
ValueError: too many values to unpack
>>> 
```

Ln: 26 Col: 4

Logistic (lab cancelled)

Indexing in Strings

Indexing in Lists

Splitting Strings into Lists

String Methods

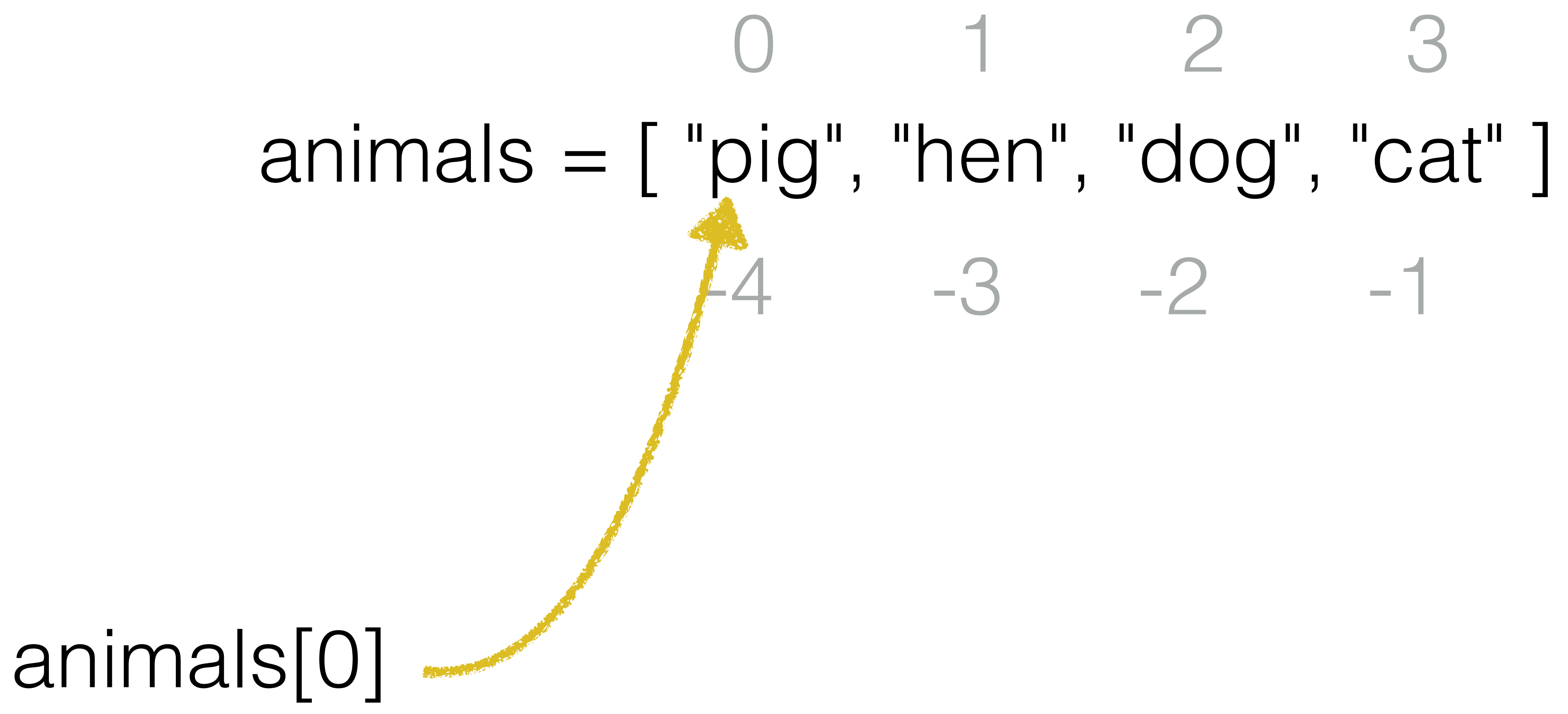
Strings are collections of characters
Lists are collections of various data types

```
animals = [ "pig", "hen", "dog", "cat" ]
```

```
             0        1        2        3
animals = [ "pig", "hen", "dog", "cat" ]
            -4       -3       -2       -1
```

```
          0      1      2      3
animals = [ "pig", "hen", "dog", "cat" ]
         -4     -3     -2     -1
```

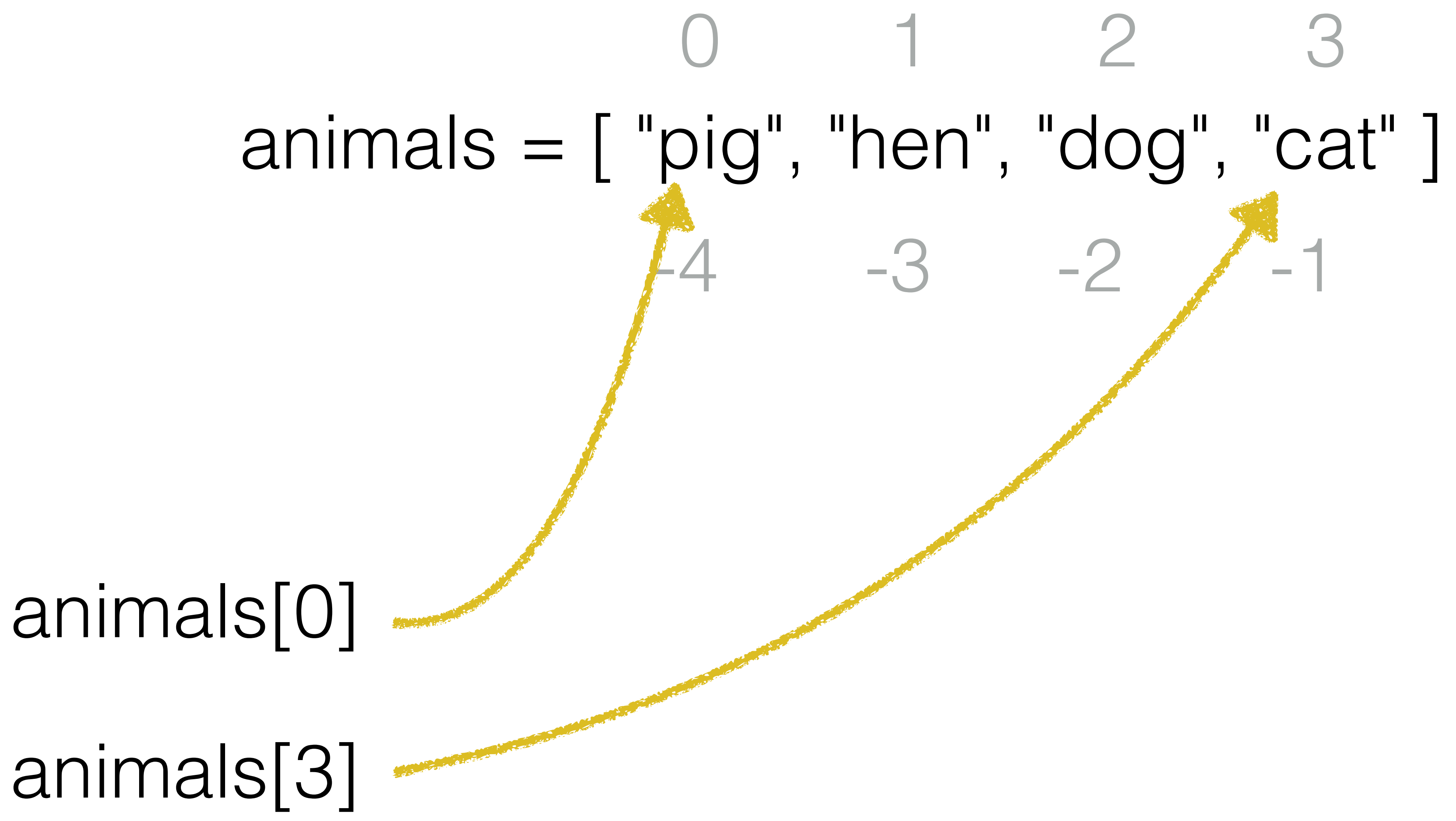
```
animals[0]
```



```
              0       1        2        3
animals = [ "pig", "hen", "dog", "cat" ]
             -4      -3       -2       -1

animals[0]
```

```
             0       1       2       3
animals = [ "pig", "hen", "dog", "cat" ]
            -4      -3      -2      -1
```

```
animals[0]

animals[3]
```



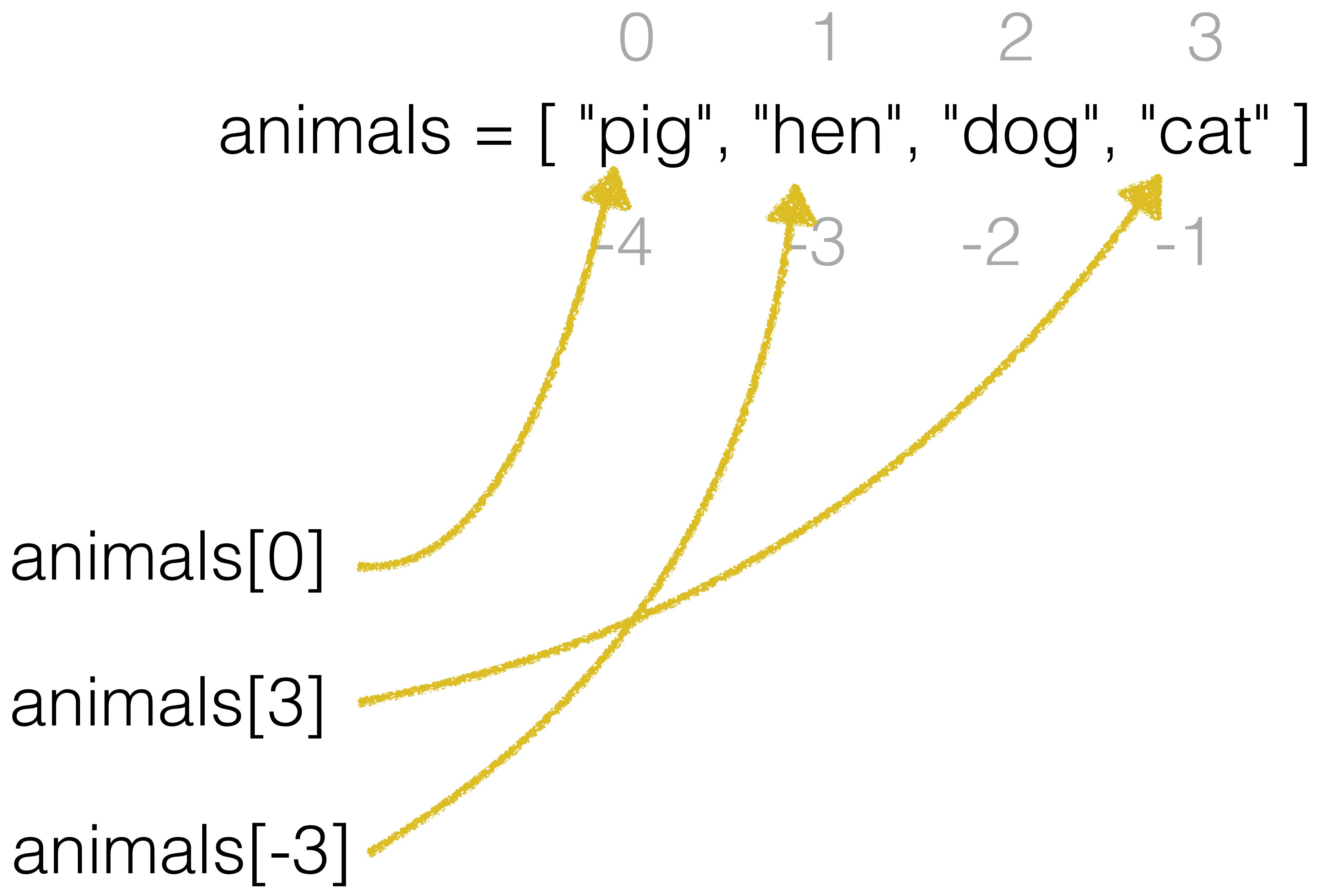
```
              0      1      2      3
animals = [ "pig", "hen", "dog", "cat" ]
             -4     -3     -2     -1

animals[0]

animals[3]
```

```
               0       1       2       3
animals = [ "pig", "hen", "dog", "cat" ]
              -4      -3      -2      -1
```

animals[0]

animals[3]

animals[-3]


0 1 2 3
animals = ["pig", "hen", "dog", "cat"]
-4 -3 -2 -1
animals[0]
animals[3]
animals[-3]

Playing with Python Semantic

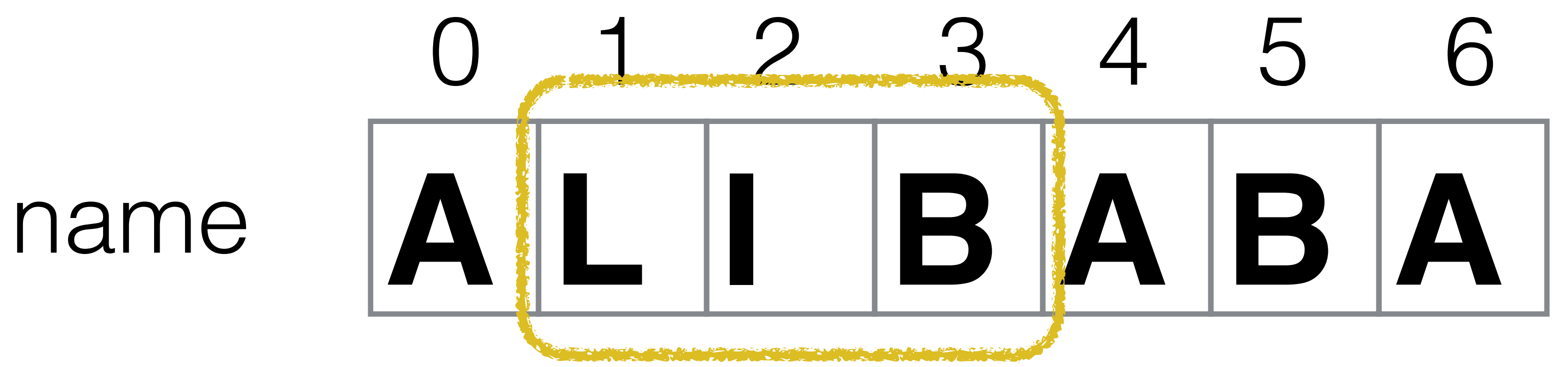
```
farm = [“pig”, “dog”, “horse”, “hen” ]
```

Find as many different ways of printing all the animals in the farm as you can…

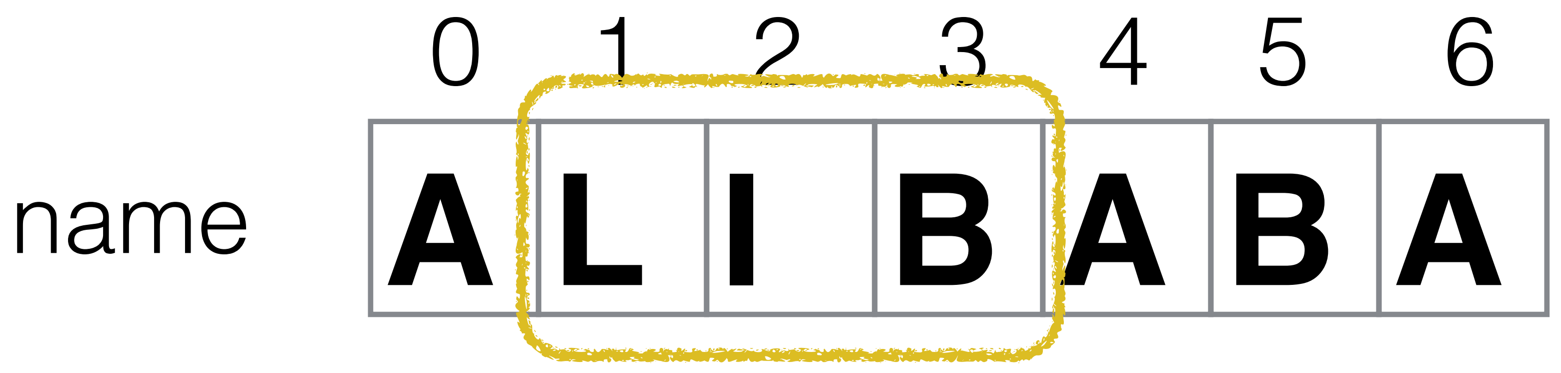
See some solutions here:

http://www.science.smith.edu/dftwiki/index.php/CSC111_Programs_Created_in_Class_2018#2.2F19.2F18

Slicing a String

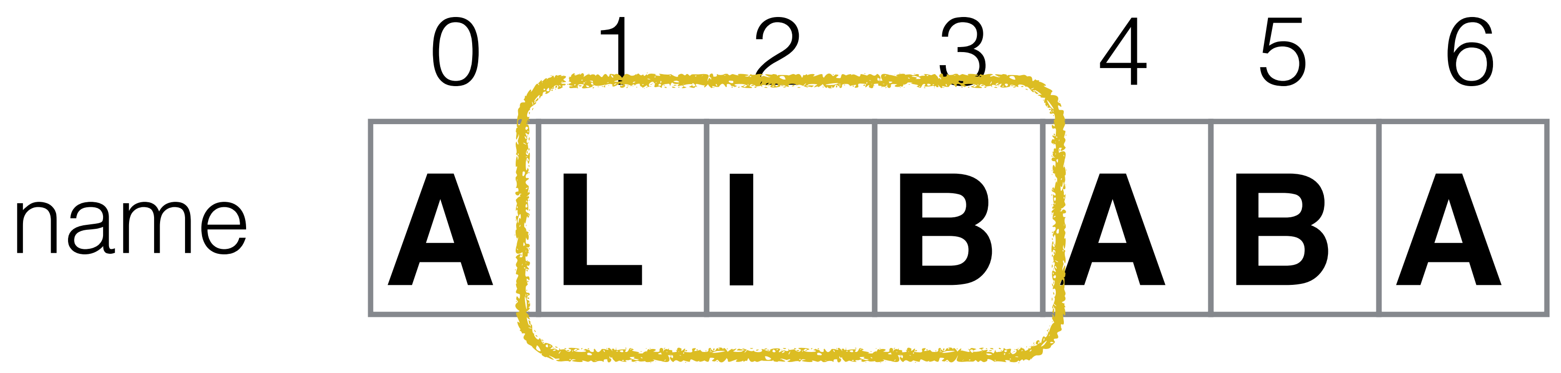


Slicing a String

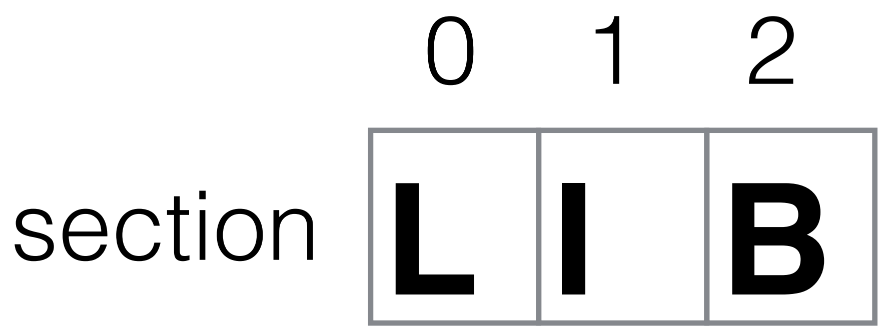


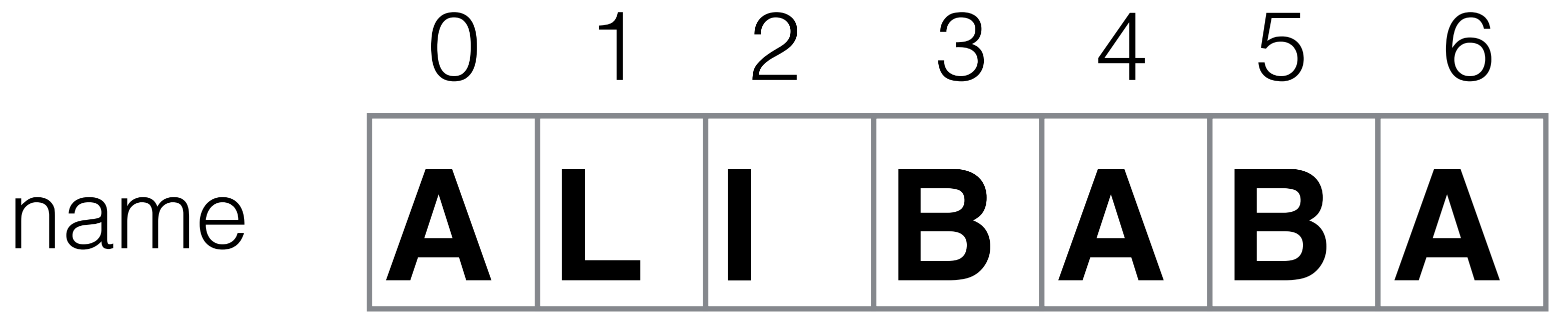
section = name[1 : 4]

Slicing a String

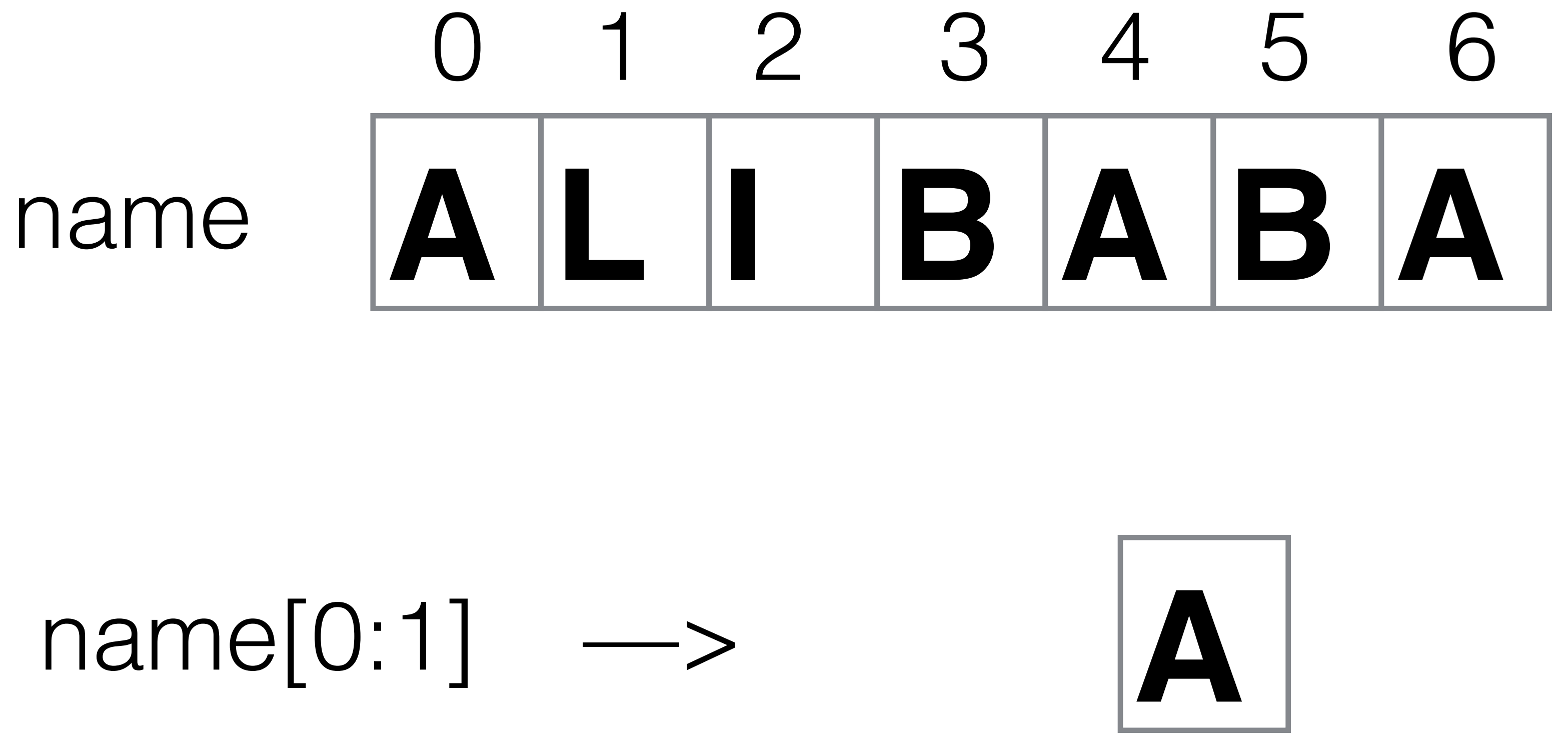


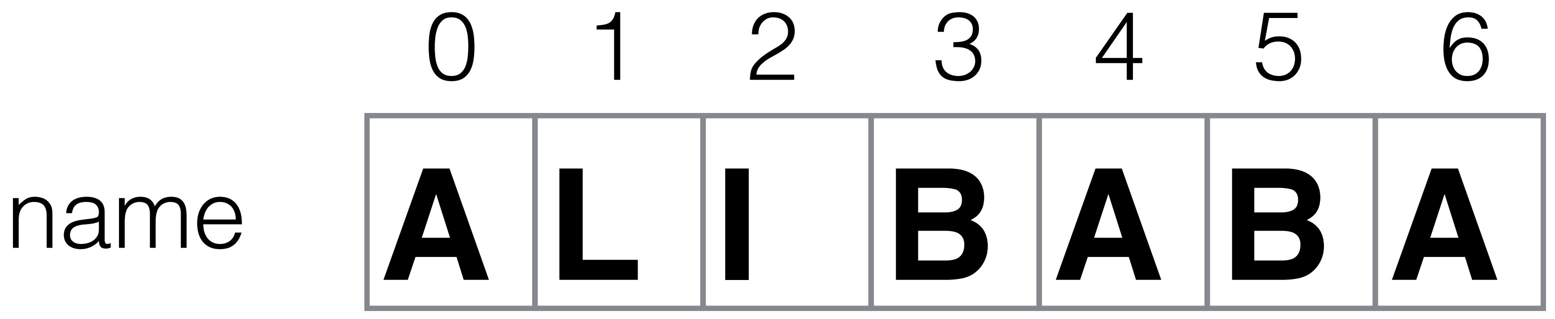
section = name[1 : 4]

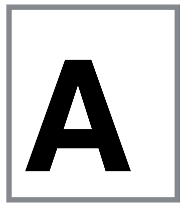




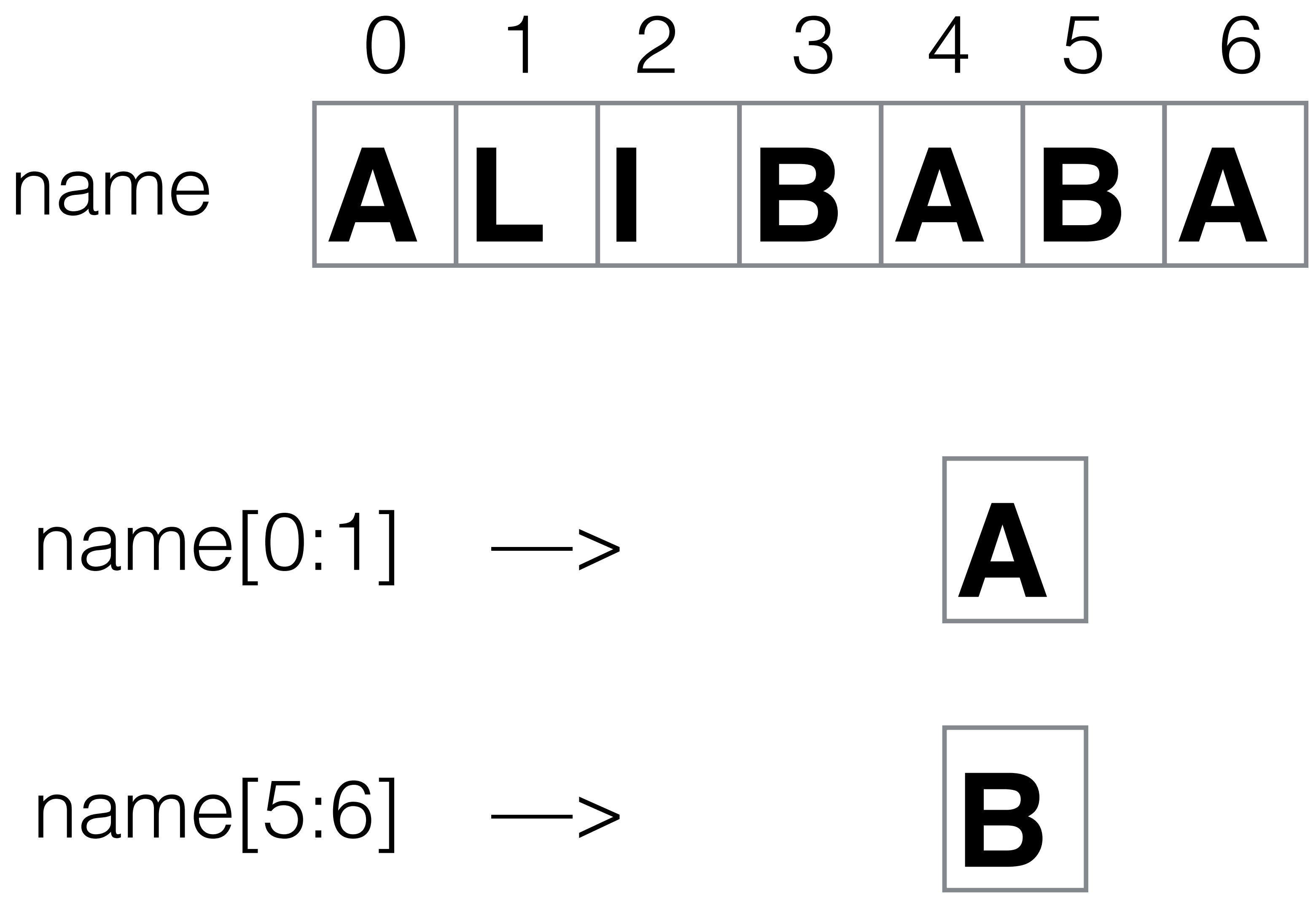
name[0:1]   —>


0 1 2 3 4 5 6
name
A L I B A B A
name[0:1] —>
A



name[0:1] —> 

name[5:6] —>



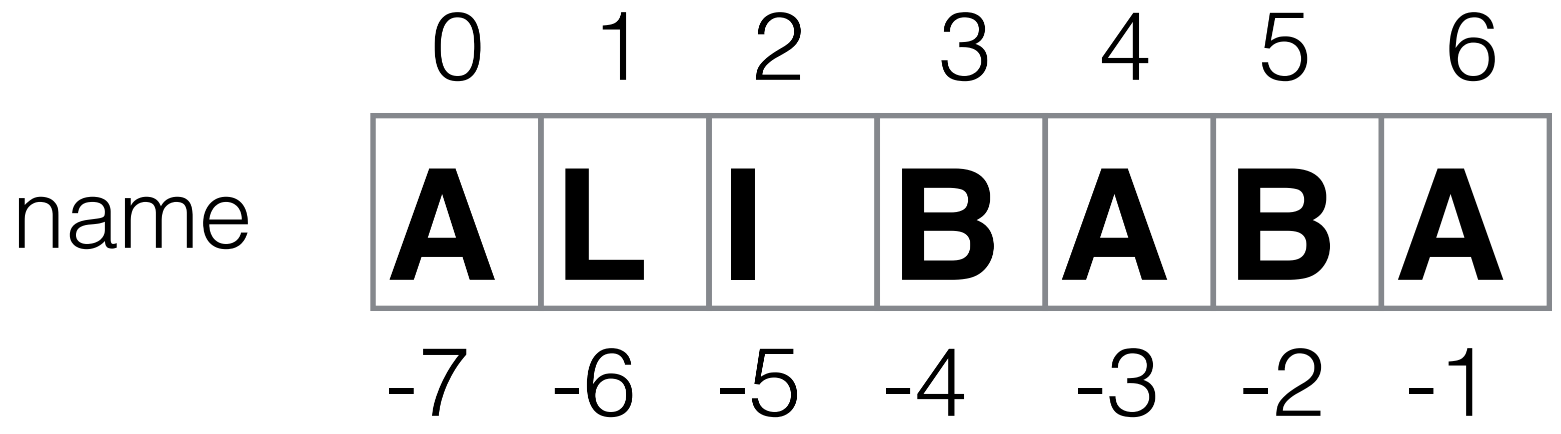
0 1 2 3 4 5 6
name
A L I B A B A
name[0:1] —>
A
name[5:6] —>
B

	0	1	2	3	4	5	6
name	**A**	**L**	**I**	**B**	**A**	**B**	**A**
	-7	-6	-5	-4	-3	-2	-1

name[0:1] —> $\boxed{\textbf{A}}$

name[5:6] —> $\boxed{\textbf{B}}$

name[-2:-1] —>

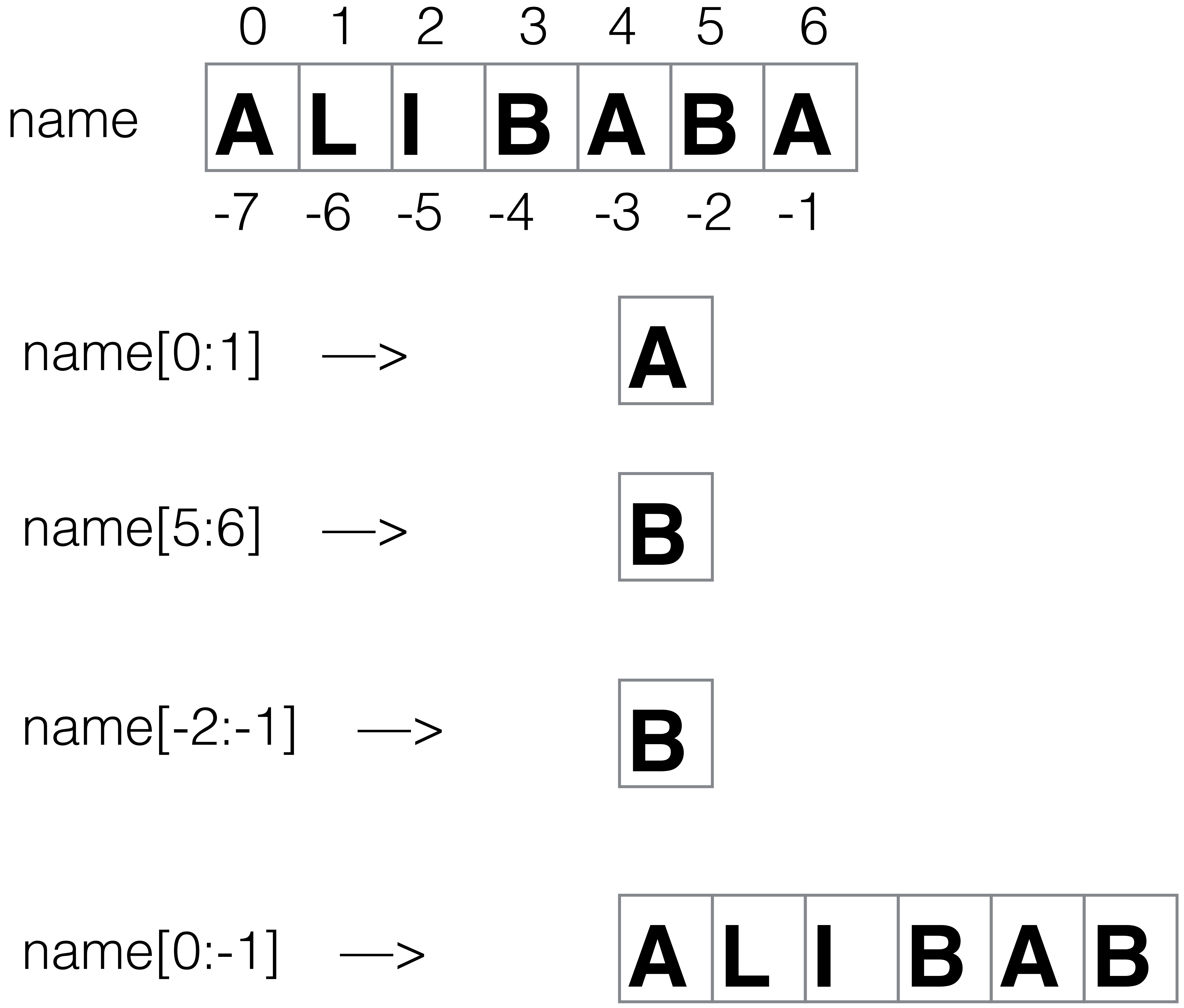


name[0:1] —> $\boxed{\textbf{A}}$

name[5:6] —> $\boxed{\textbf{B}}$

name[-2:-1] —> $\boxed{\textbf{B}}$

name[0:-1] —>


0 1 2 3 4 5 6
name
A L I B A B A
-7 -6 -5 -4 -3 -2 -1
name[0:1] —> A
name[5:6] —> B
name[-2:-1] —> B
name[0:-1] —> A L I B A B

Two Special Slices

```
part = name[ : 5 ]
```

```
part = name[ 3 :  ]
```

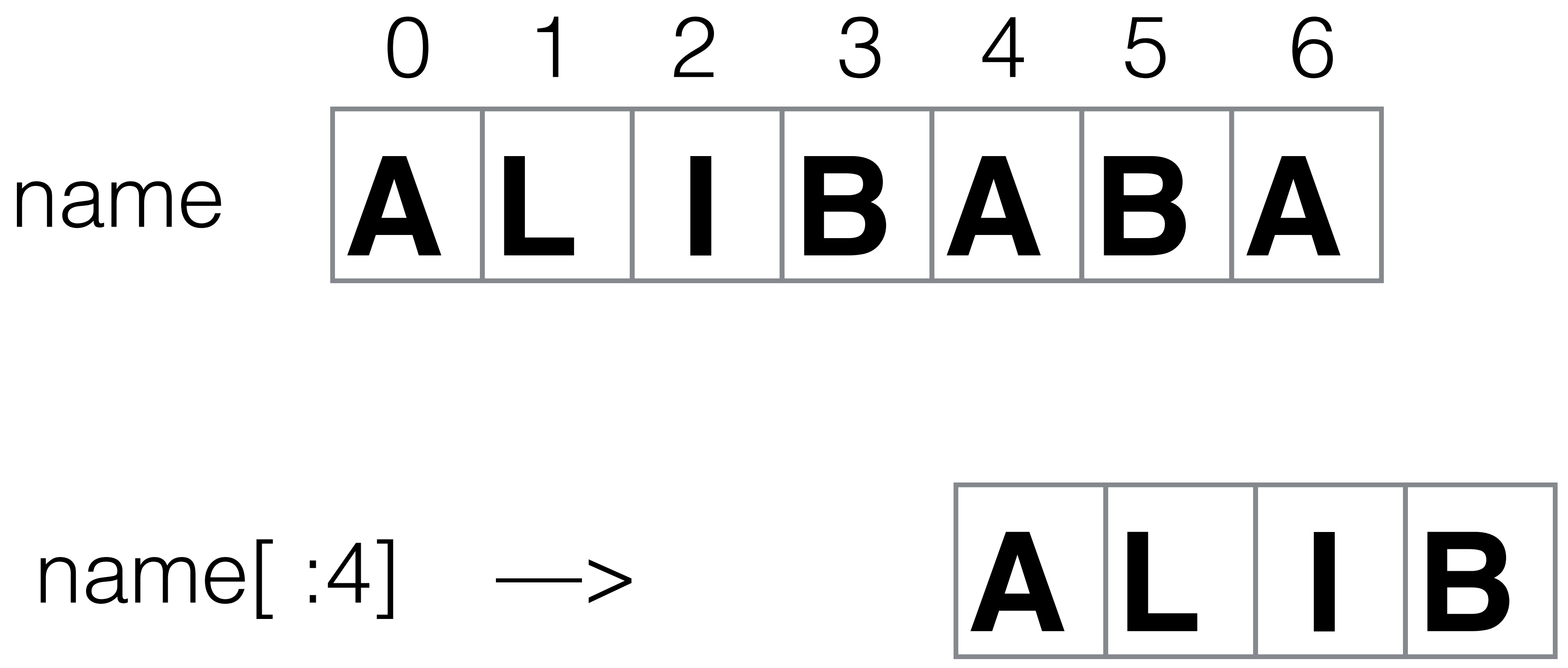
Two Special Slices

part = name[: 5]   <— **from beginning to 5**

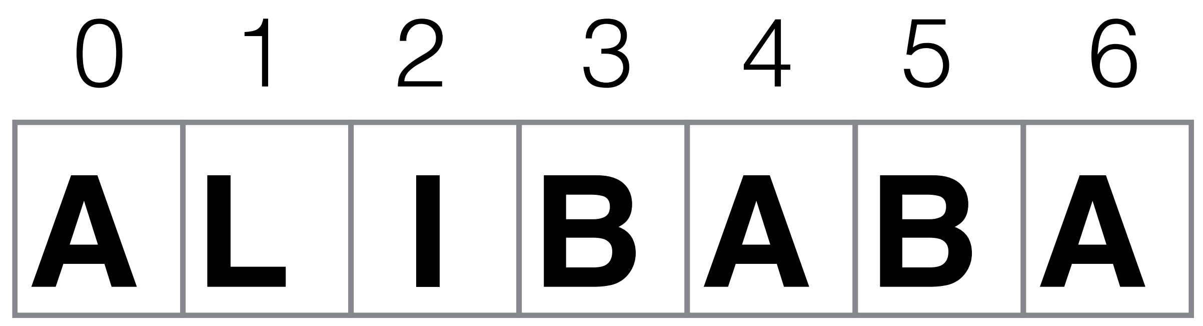
part = name[3 :]   <— **from 3 to end, including last**

	0	1	2	3	4	5	6
name	**A**	**L**	**I**	**B**	**A**	**B**	**A**

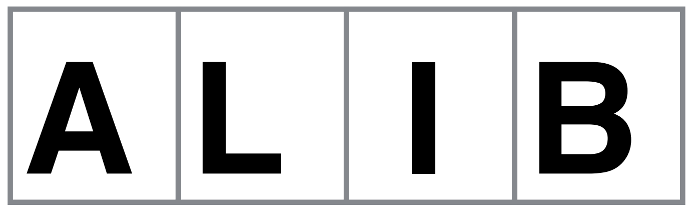
name[:4]   —>


0 1 2 3 4 5 6
name
A L I B A B A
name[:4] —>
A L I B

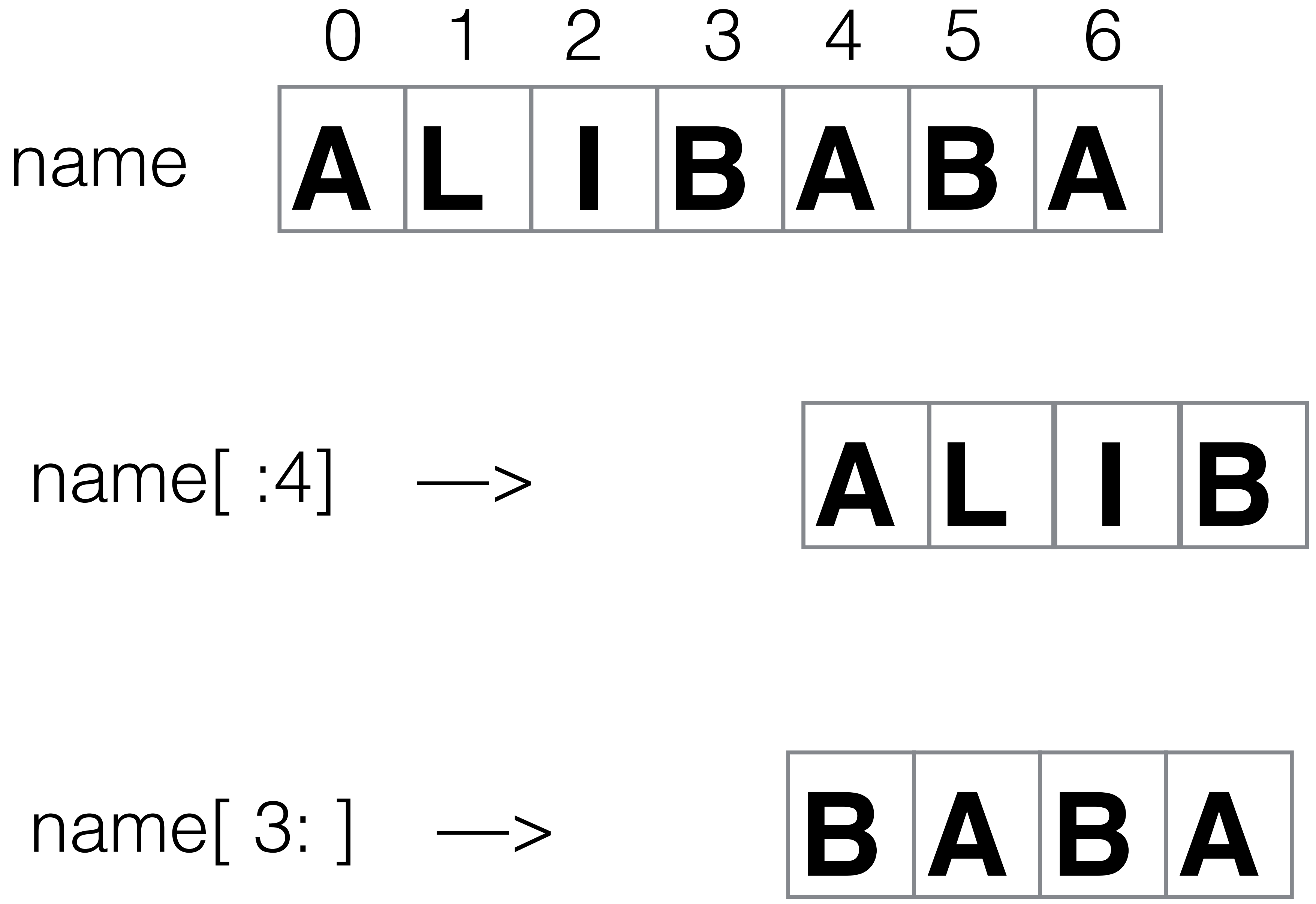
name



name[:4] —>



name[3:] —>


0 1 2 3 4 5 6
name
A L I B A B A
name[:4] —>
A L I B
name[3:] —>
B A B A

https://www.thespruce.com/classic-french-rum-baba-recipe-1375132

We stopped here last time...

Outline

- **No lab today or tomorrow, but…**
- Homework 4 prep page available!
- Review
- Continue with indexing and slicing…
- ***Correction***: We're covering **Chapter 5**, not 6

- **No lab today or to**
- Homework 4 prep
- Review
- Continue with inde

- ***Correction***: We're covering **Chapter 5**, not 6

Image credit: http://www.power-animals.com/2014/02/19/why-are-guilty-dogs-so-funny/

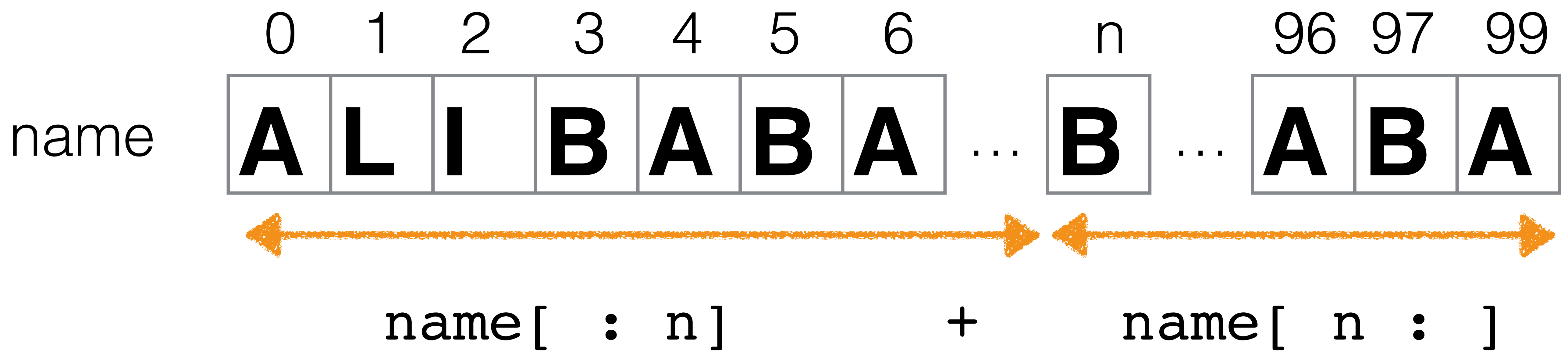
Interesting Property

```
name = "Some string of characters"

name2 = name[ : 7] + name[ 7 : ]
```

Interesting Property

```
name = "Some string of characters"

name2 = name[ : n] + name[ n : ]
```

name2 contains the same string as name


0 1 2 3 4 5 6 n 96 97 99
name
A L I B A B A … B … A B A
name[: n] + name[n :]

Exer

Extract the **drive** and **extension** information from a file name:

```
name = "H:/Documents/solutionsHw4.doc"
```

Replace the “doc” extension by “txt” in the file name:

```
name = "H:/Documents/solutionsHw4.doc"
```

Get the first and last name of a person and create
a computer account with the first letter of the first name, and the last name, concatenated.

Solu

```
>>> name = "H:/Documents/solutionsHw4.doc"
>>> name
'H:/Documents/solutionsHw4.doc'
>>> drive = name[0]
>>> drive
'H'
>>> drive = name[0:2]
>>> drive
'H:'
>>> extension = name[-3: ]
>>> extension
'doc'
>>>
```

```
>>> name = name[0:-3] +
"txt"
>>> name
'H:/Documents/
solutionsHw4.txt'
```

```
>>> fname = input( "First name
First name? Rui
>>> lname = input( "Last name?
Last name?  Hwang
>>> account = fname[0] + lname
>>> account
'RHwang'
```

Exercis

```
Transforming dates:
Transform a string, such as "02162018"
into 16 Feb 2018.
```

```
Get the first and last name from
a person,  and display a
“triangle” made of her
full name.  For example,
fname = "Maria"
lname = “LUCE”
Output of program:
M
Ma
Mar
Mari
Maria
MariaL
MariaLU
MariaLUC
MariaLUCE
```

Solutions

```
months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun",
                  "Jul", "Aug", "Sep", "Oct", "Nov", "Dec" ]

date = "02162018"
m = int( date[0:2] )
d = date[2:4]
y = date[4: ]
print( d, months[m-1], y )
```

```
fname = "Maria"
lname = "LUCE"
name = fname+lname
numChars = len( name )
for i in range( 1, numChars+1 ):
    print( name[0:i] )
```

Review

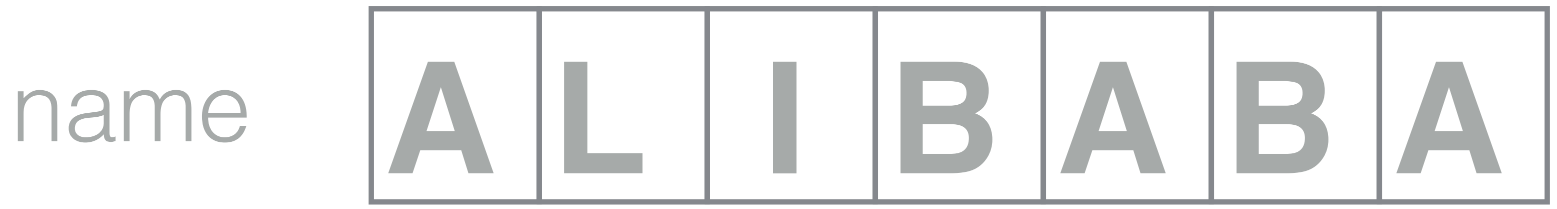
- Strings are **lists** of characters

- Strings are **lists** of characters

name

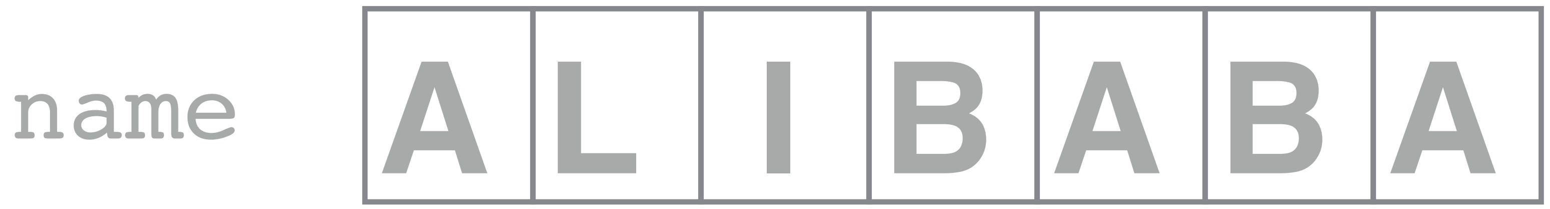
A	L	I	B	A	B	A

- Strings are **lists** of characters



- Lists are *lists* of items, too!

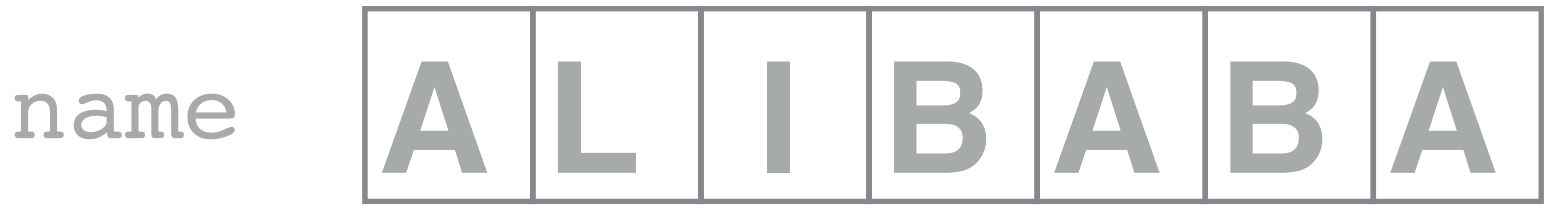
- Strings are **lists** of characters



- Lists are *lists* of items, too!

```
farm = [ "dog", "cat", "pig" ]
```

- Strings are **lists** of characters



- Lists are *lists* of items, too!

```
farm = [ "dog", "cat", "pig" ]
```

- They can be indexed, and sliced

- Strings are **lists** of characters

`name`	A	L	I	B	A	B	A

- Lists are *lists* of items, too!

```
farm = [ "dog", "cat", "pig" ]
```

- They can be indexed, and sliced

```
name[-1]     name[0:2]

farm[-1]     farm[0:2]
```

- Strings are **lists** of characters

`name`	A	L	I	B	A	B	A

- Lists are *lists* of items, too!

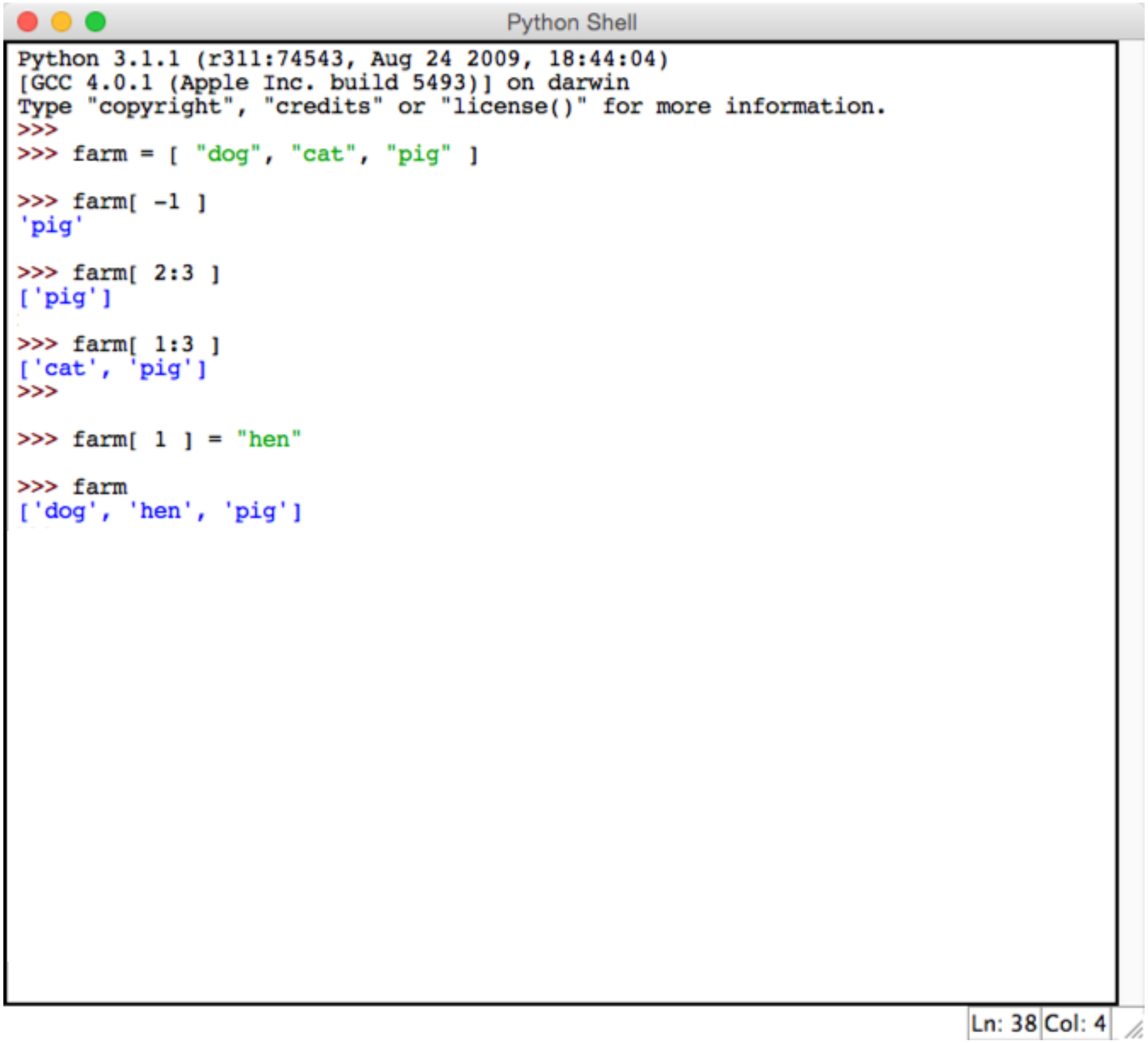
```
farm = [ "dog", "cat", "pig" ]
```

- They can be indexed, and sliced

A	`name[-1]`	`name[0:2]`	A L
`"pig"`	`farm[-1]`	`farm[0:2]`	`["dog","cat"]`

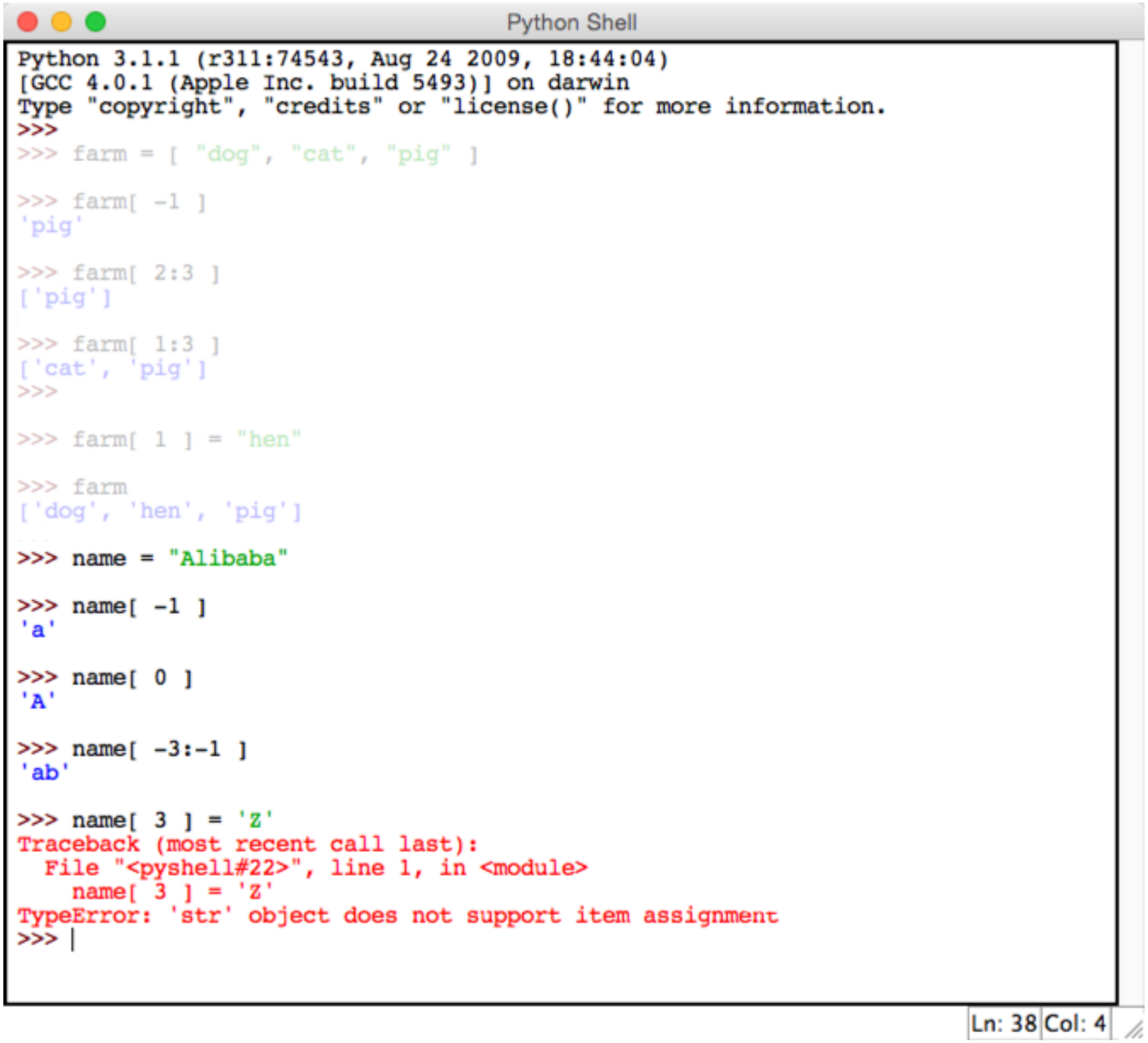
Lists and Strings behave similarly

but are different in an important way


Python Shell

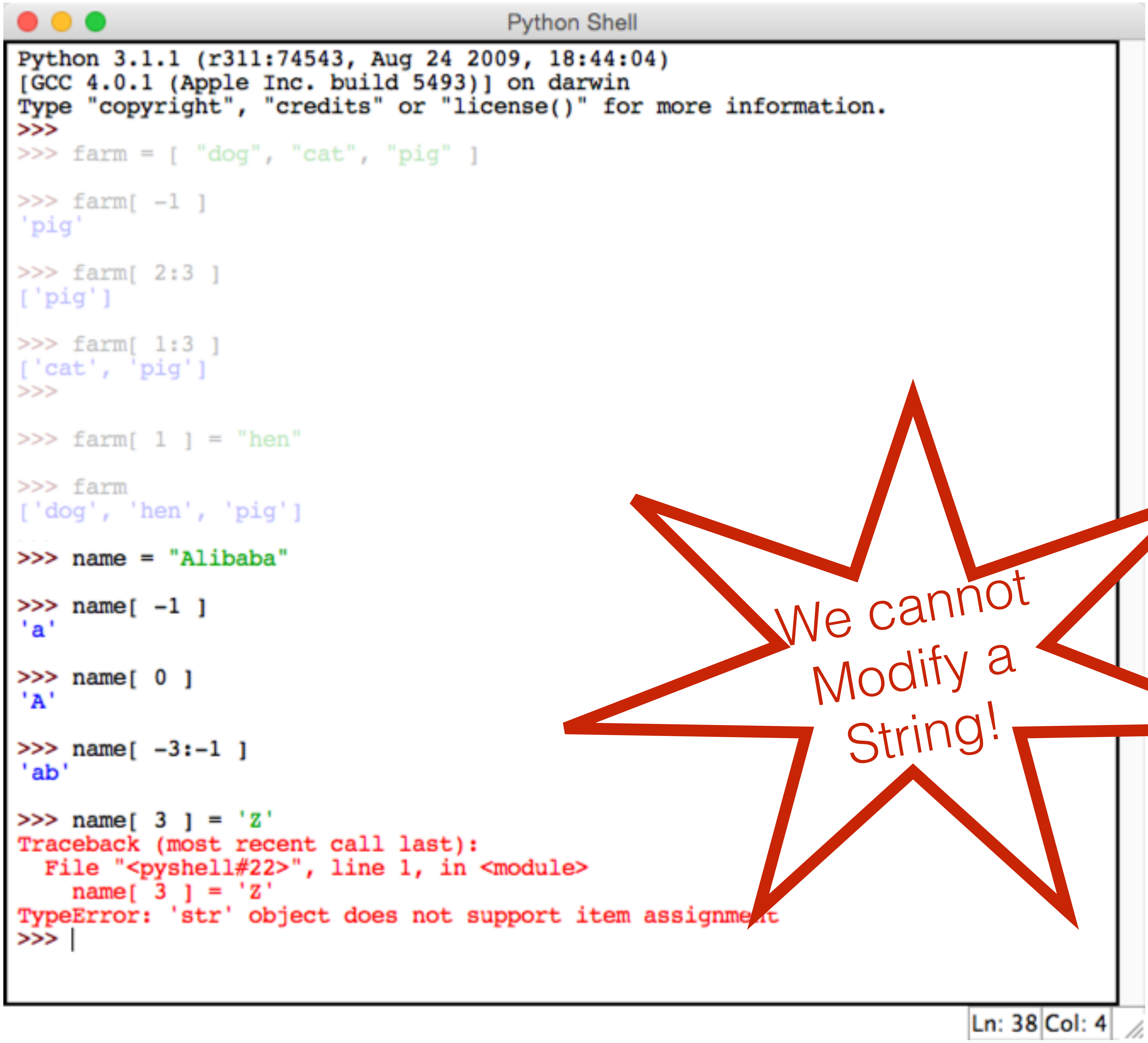
```
Python 3.1.1 (r311:74543, Aug 24 2009, 18:44:04)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
>>> farm = [ "dog", "cat", "pig" ]

>>> farm[ -1 ]
'pig'

>>> farm[ 2:3 ]
['pig']

>>> farm[ 1:3 ]
['cat', 'pig']
>>>

>>> farm[ 1 ] = "hen"

>>> farm
['dog', 'hen', 'pig']
```

Ln: 38 Col: 4


Python Shell

```
Python 3.1.1 (r311:74543, Aug 24 2009, 18:44:04)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
>>> farm = [ "dog", "cat", "pig" ]

>>> farm[ -1 ]
'pig'

>>> farm[ 2:3 ]
['pig']

>>> farm[ 1:3 ]
['cat', 'pig']
>>>

>>> farm[ 1 ] = "hen"

>>> farm
['dog', 'hen', 'pig']

>>> name = "Alibaba"

>>> name[ -1 ]
'a'

>>> name[ 0 ]
'A'

>>> name[ -3:-1 ]
'ab'

>>> name[ 3 ] = 'Z'
Traceback (most recent call last):
  File "<pyshell#22>", line 1, in <module>
    name[ 3 ] = 'Z'
TypeError: 'str' object does not support item assignment
>>> |
```

Ln: 38 Col: 4



```
Python 3.1.1 (r311:74543, Aug 24 2009, 18:44:04)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
>>> farm = [ "dog", "cat", "pig" ]

>>> farm[ -1 ]
'pig'

>>> farm[ 2:3 ]
['pig']

>>> farm[ 1:3 ]
['cat', 'pig']
>>>

>>> farm[ 1 ] = "hen"

>>> farm
['dog', 'hen', 'pig']

>>> name = "Alibaba"

>>> name[ -1 ]
'a'

>>> name[ 0 ]
'A'

>>> name[ -3:-1 ]
'ab'

>>> name[ 3 ] = 'Z'
Traceback (most recent call last):
  File "<pyshell#22>", line 1, in <module>
    name[ 3 ] = 'Z'
TypeError: 'str' object does not support item assignment
>>> |
```

We cannot Modify a String!

Strings are Immutable

Lists (with [...]) are mutable

Lists (with (…)) are immutable

OOP

Logistic (lab cancelled)

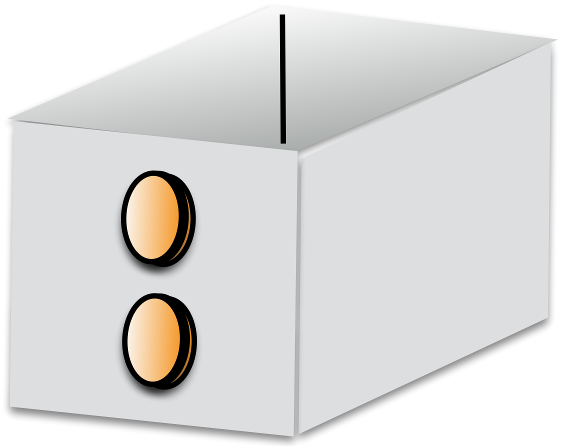
Indexing in Strings

Indexing in Lists

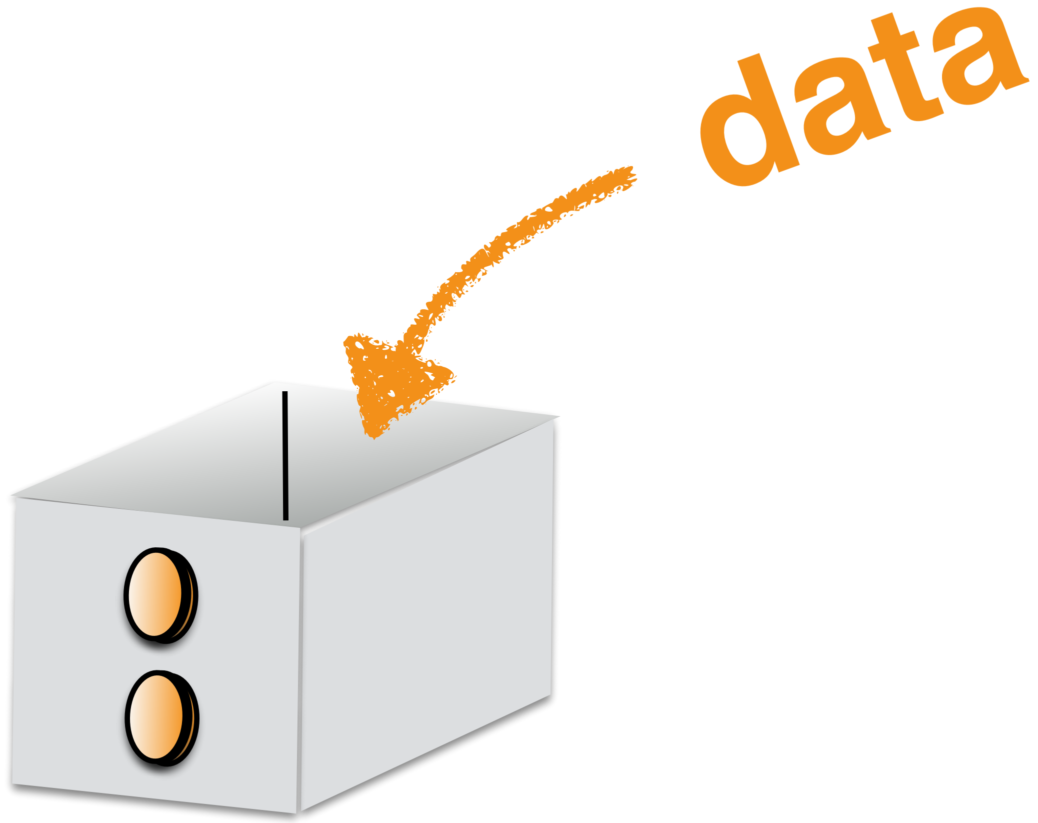
String Objects and Methods

Splitting Strings into Lists

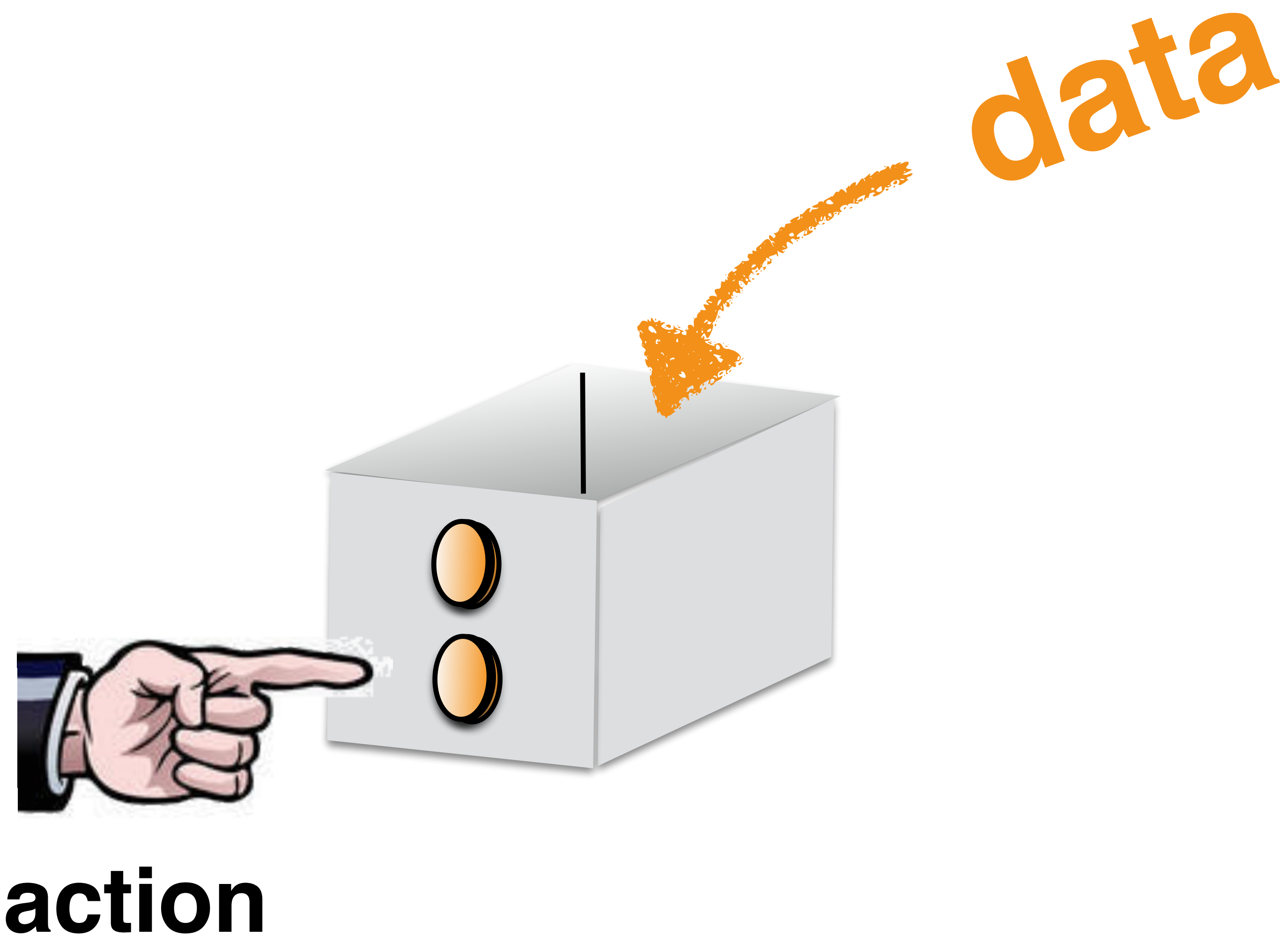
Objects



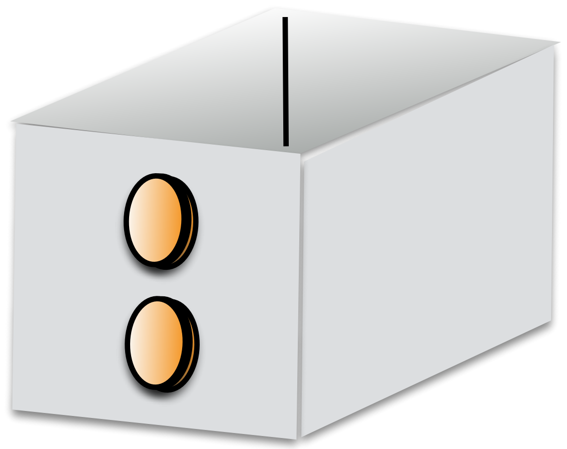
Objects



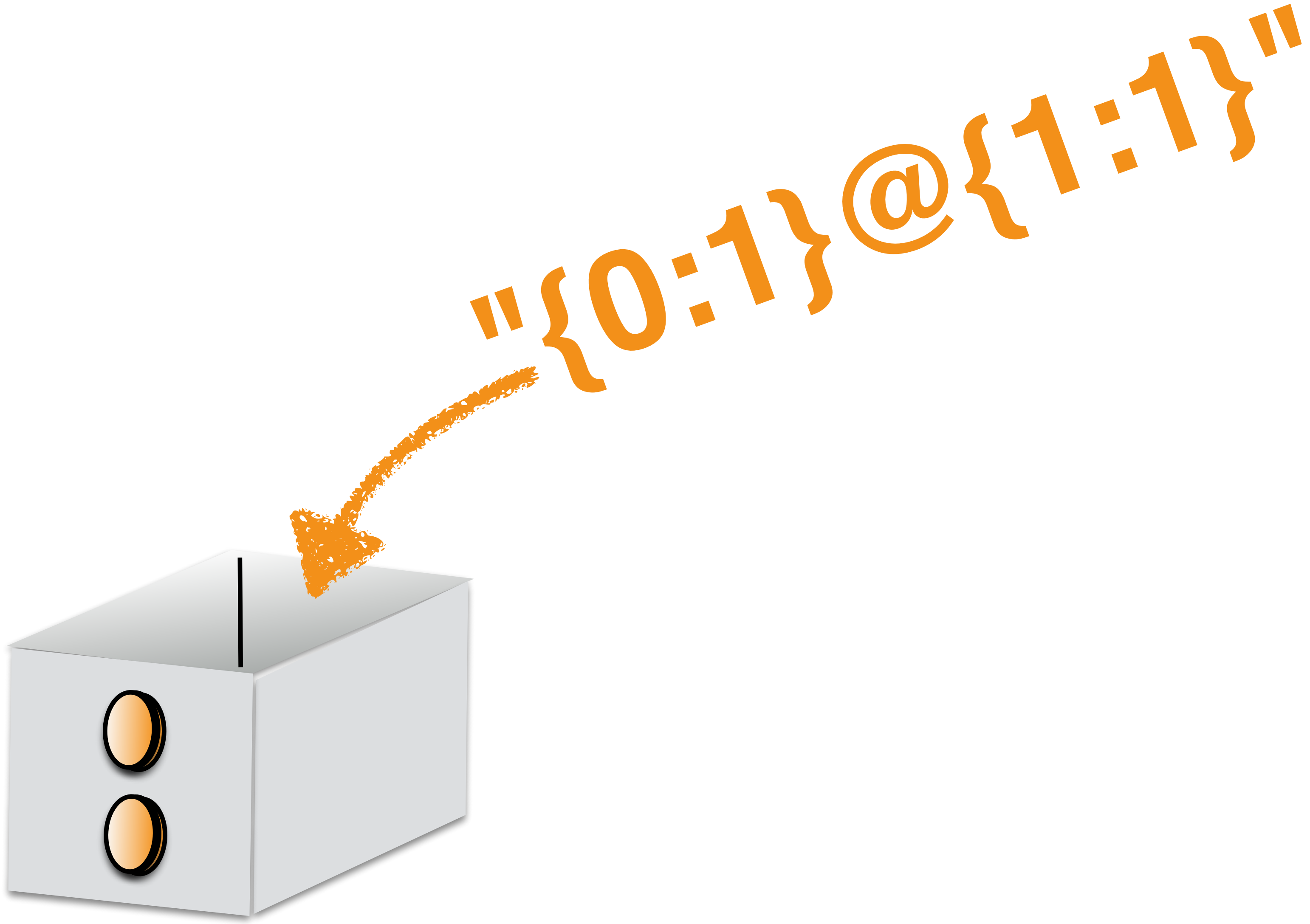
Objects



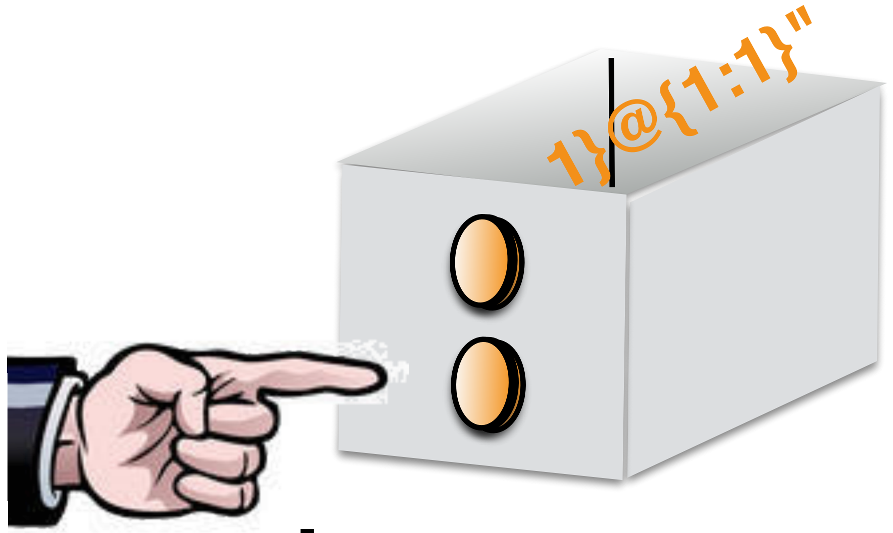
Examples



Examples

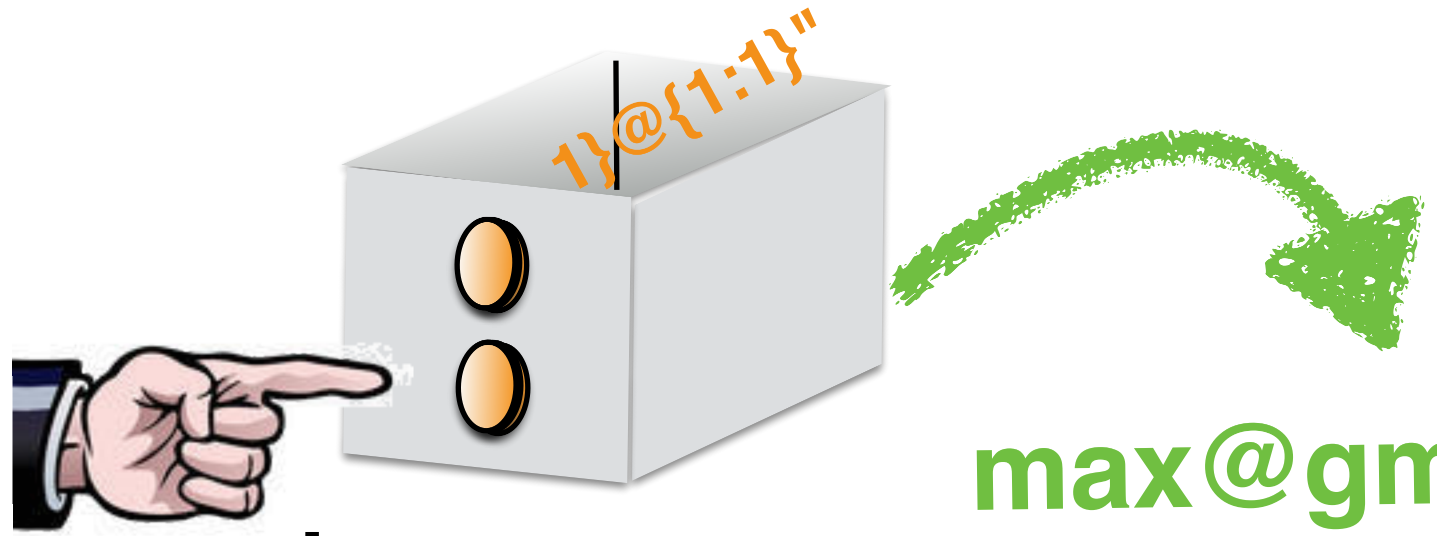


Examples



format action
("max", "gmail.com")

Examples

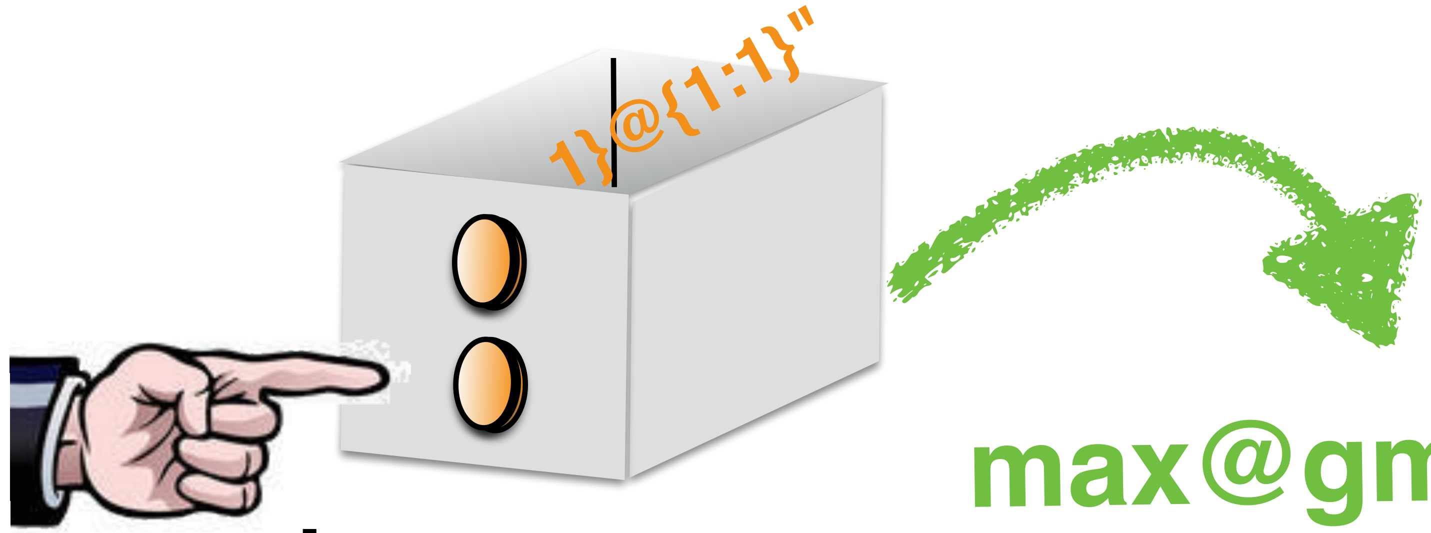


format action ("max", "gmail.com")

max@gmail.com

Examples

```
"{0:1}@{1:1}".format( "max", "gmail.com" )
```



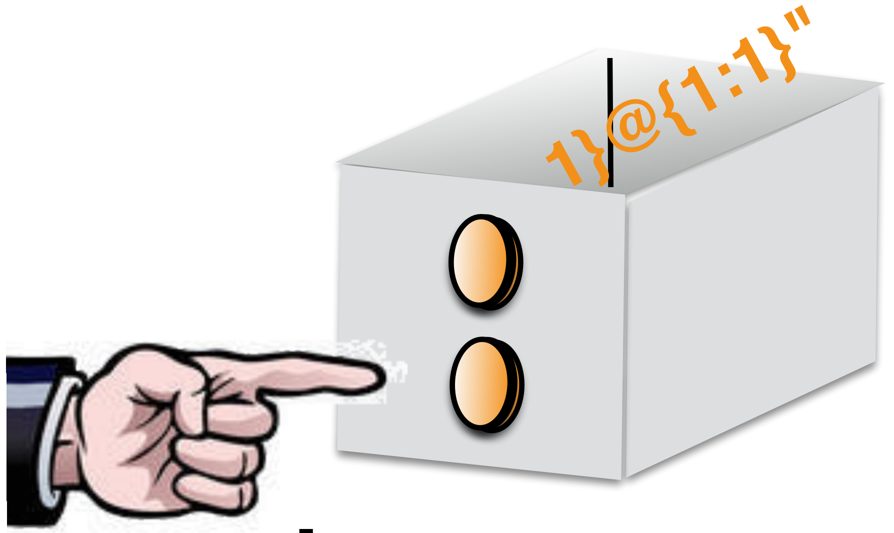
format action
("max", "gmail.com")

max@gmail.com

Examples

method

```
"{0:1}@{1:1}".format( "max", "gmail.com" )
```



format action
("max", "gmail.com")

max@gmail.com

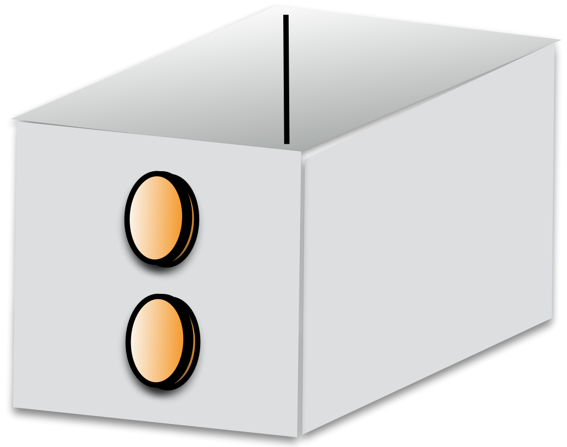
Logistic (lab cancelled)

Indexing in Strings

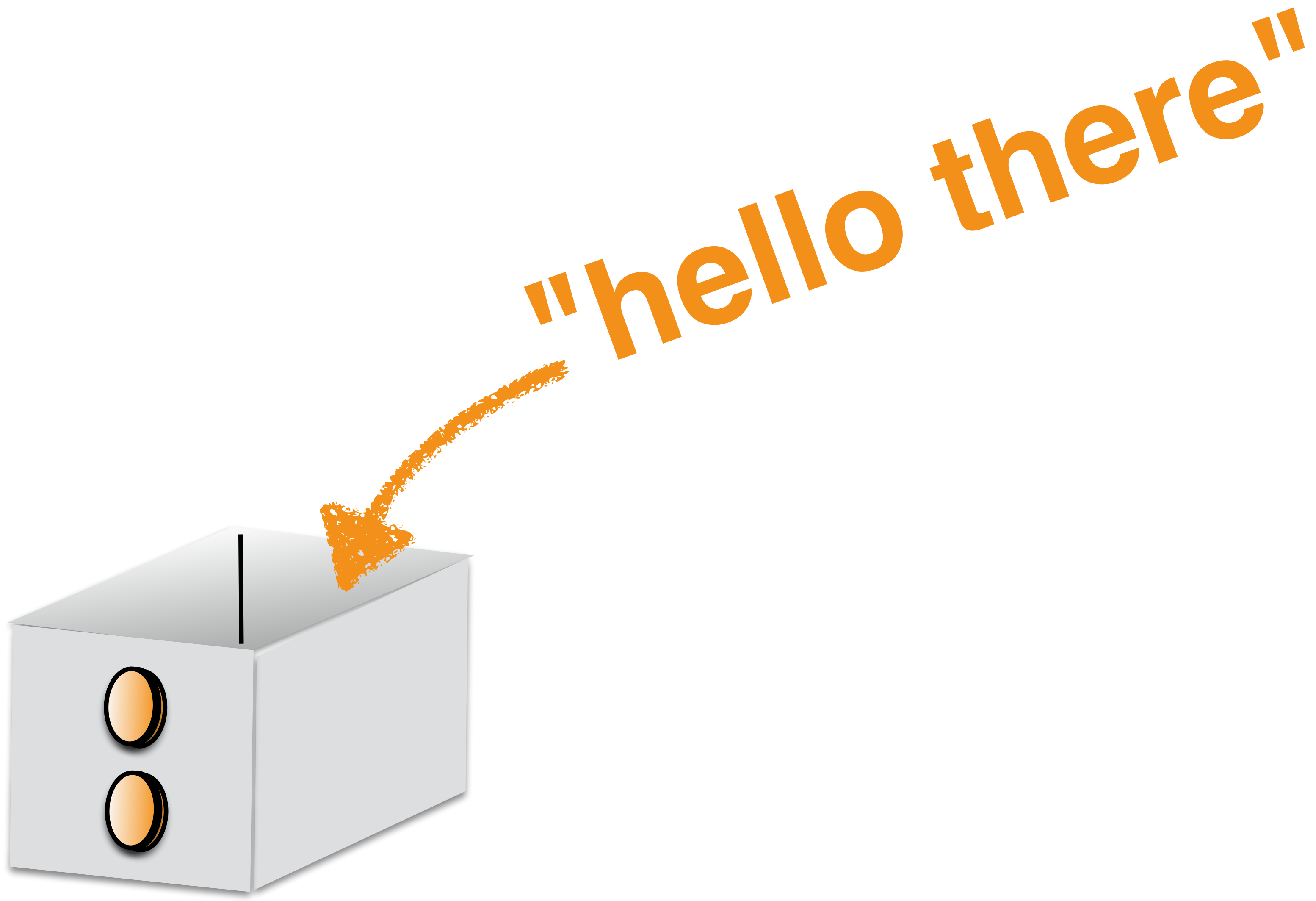
Indexing in Lists

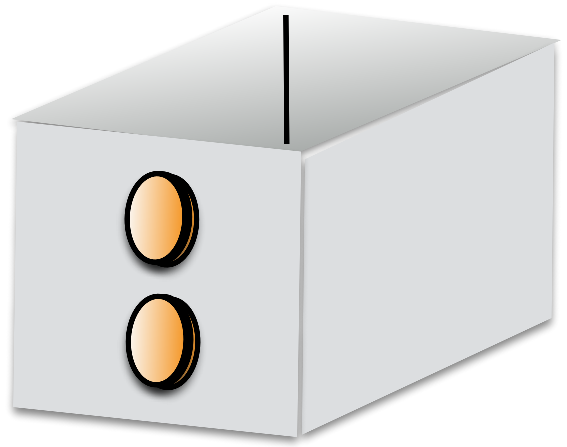
String Objects and Methods

Splitting Strings into Lists

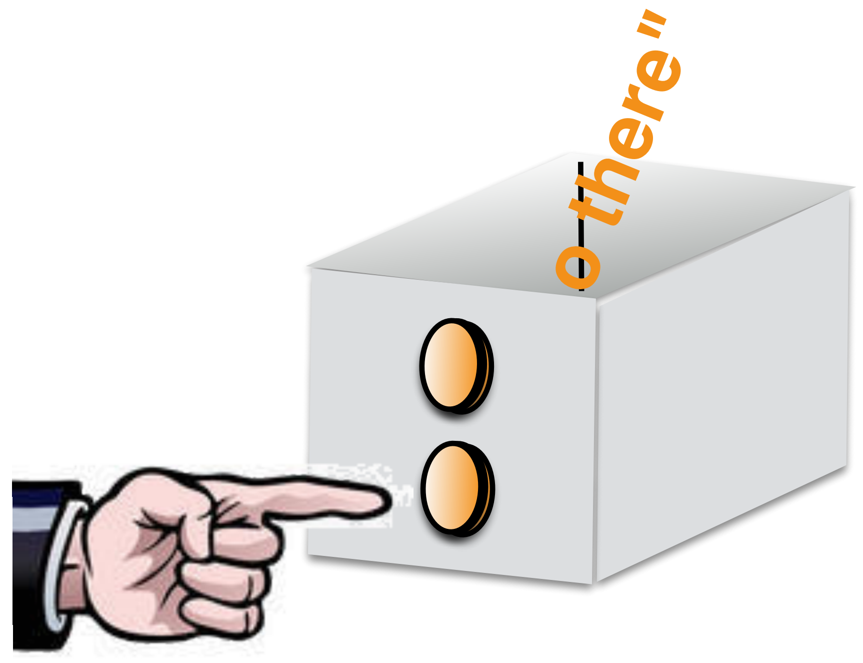


- `upper()`

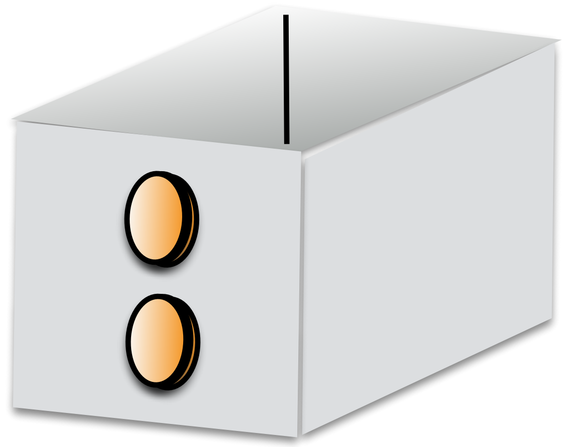




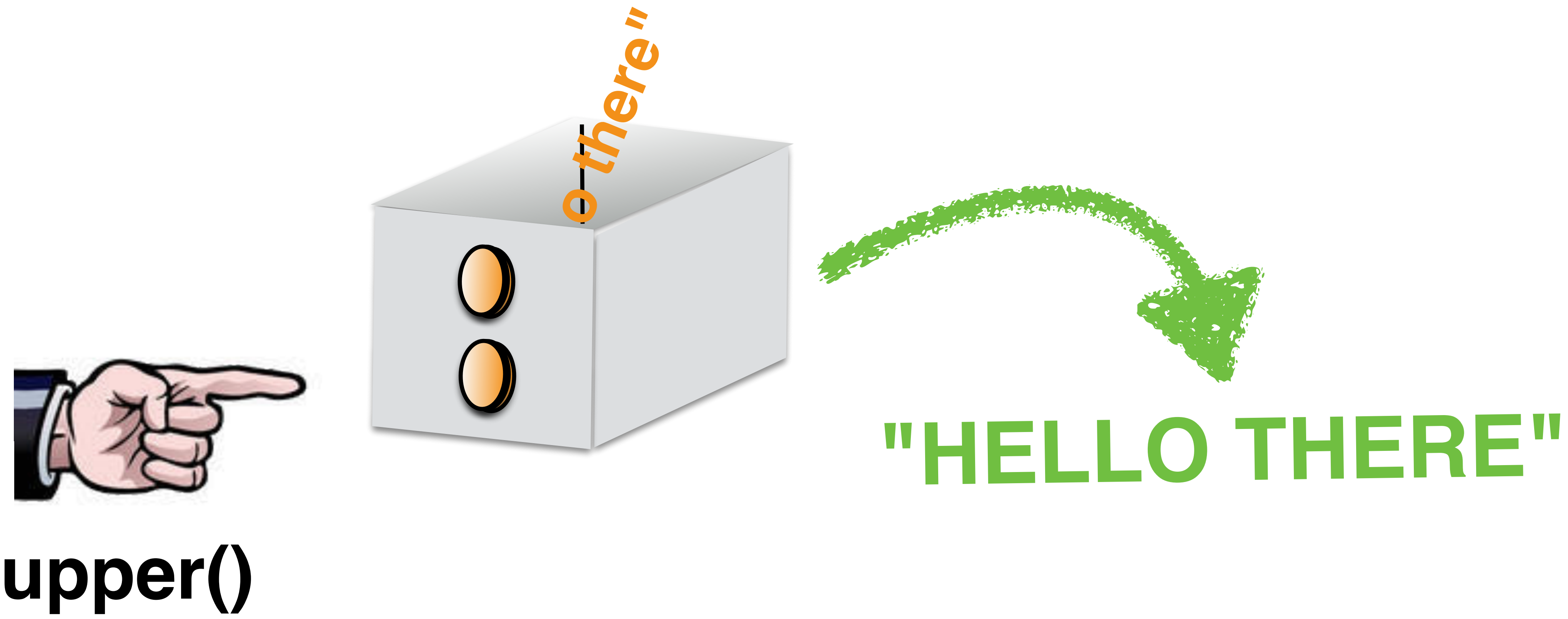
- `upper()`

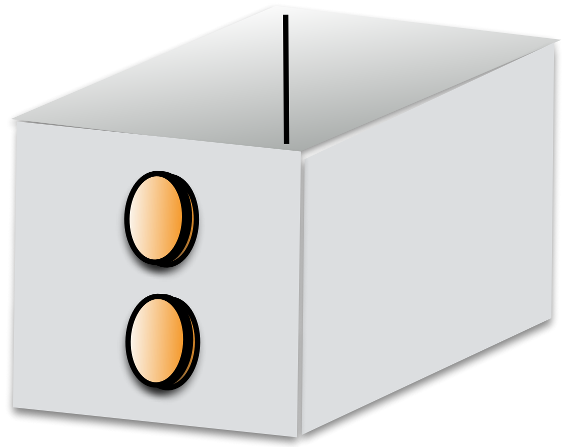


upper()



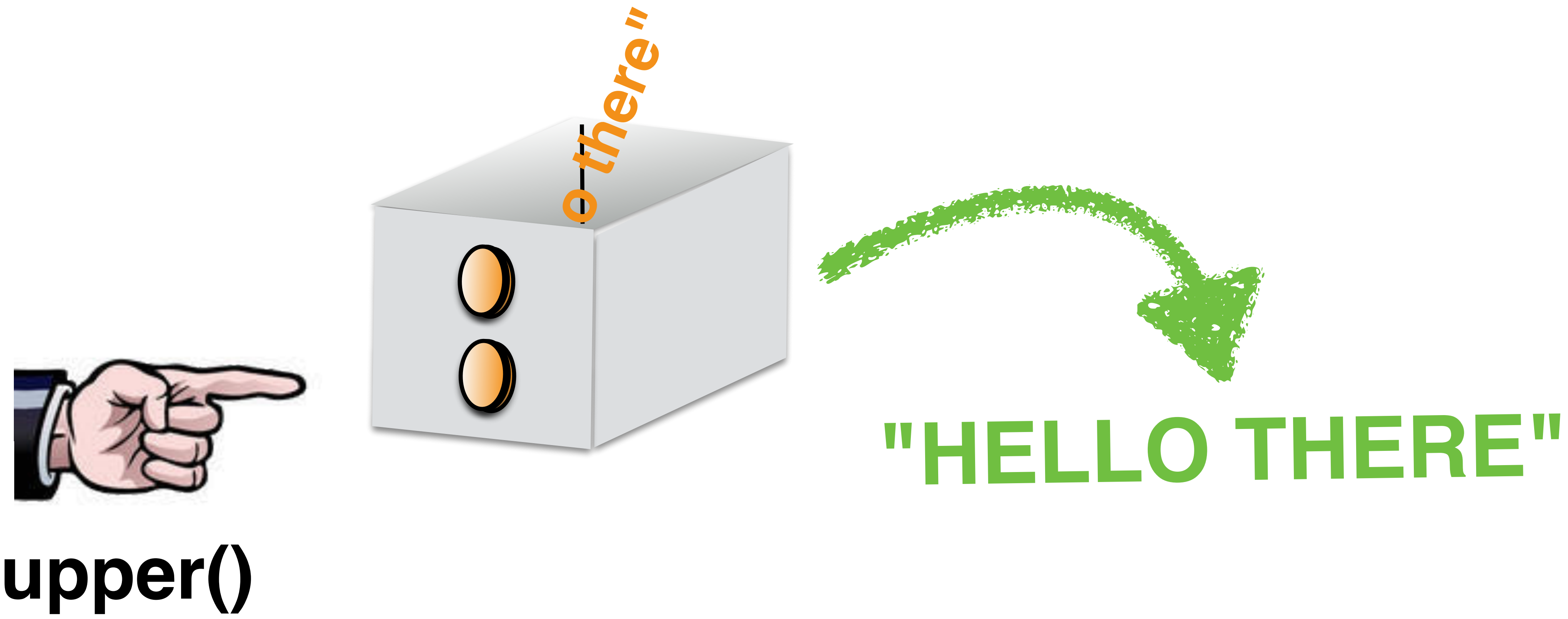
- `upper()`

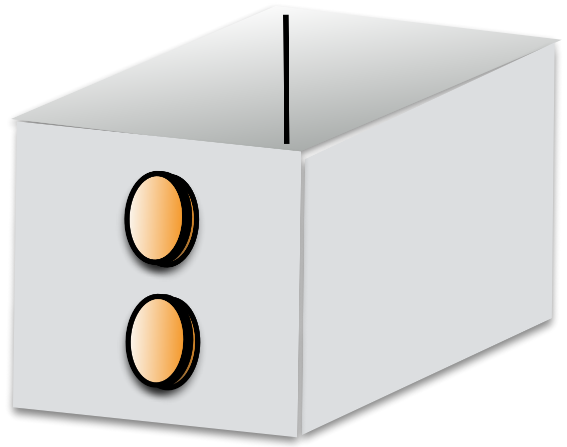




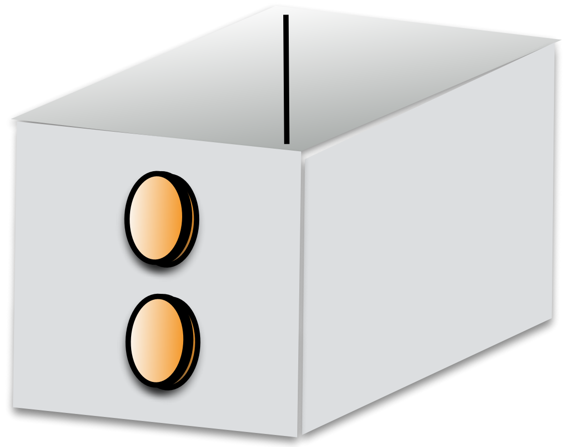
- upper()

```
"hello there".upper()
```



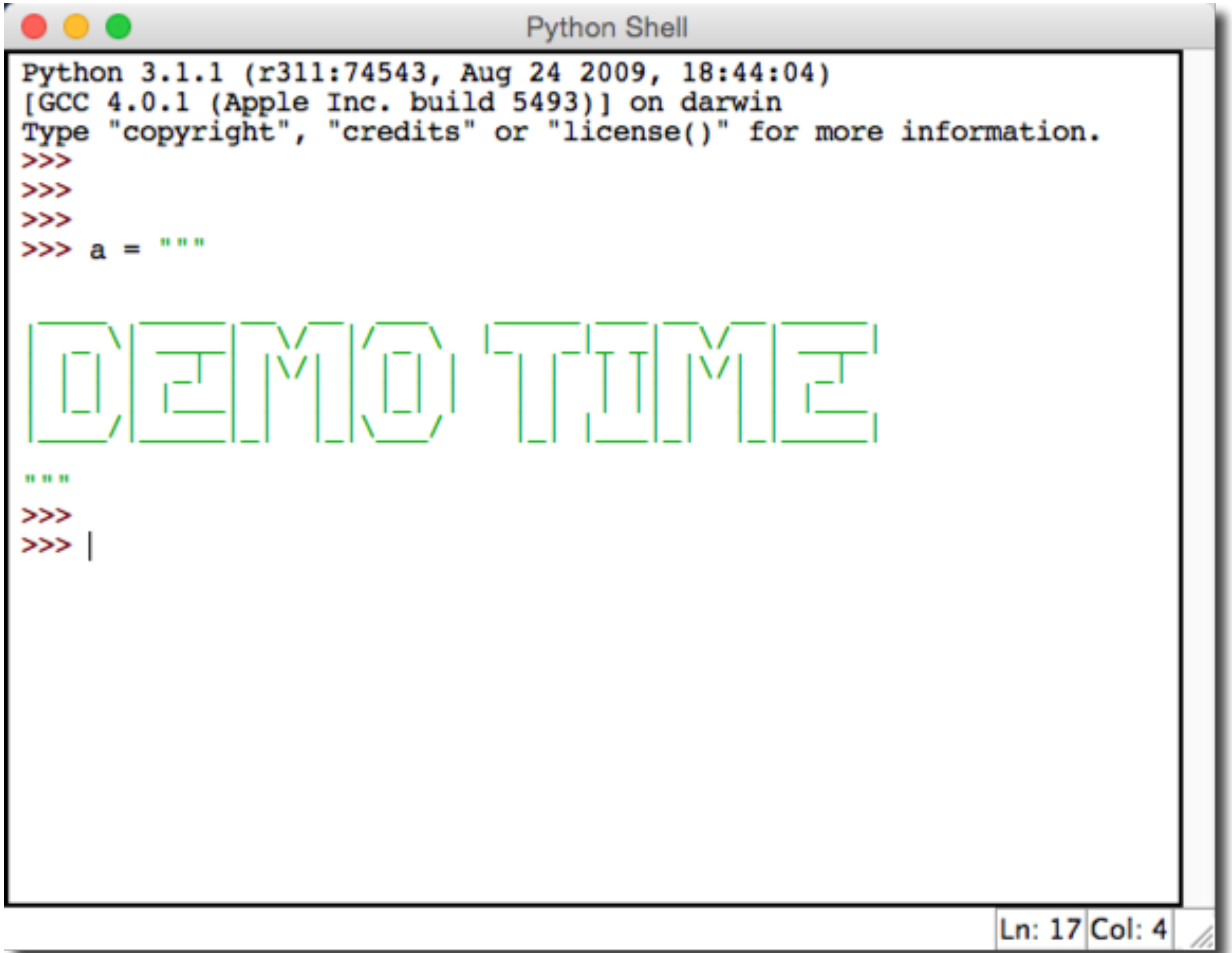


- `upper()`
- `lower()`
- `center( n )`
- `capitalize()`
- `title()`



- upper()
- lower()
- center(n)
- capitalize()
- title()

- format(…, …)
- find(…)
- replace(…, …)



```
Python 3.1.1 (r311:74543, Aug 24 2009, 18:44:04)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
>>>
>>>
>>> a = """



"""
>>>
>>> |
```

We stopped here last time…

Programming
Hacking

Programming Hacking

- Be organized
- Start Early
- Test the System (submit early)
- Be resourceful
- Copy/Paste instead of upload
- Keep working copies of code on your computer, not on Moodle

https://docs.python.org/3/library/stdtypes.html?highlight=upper#string-methods

python.org

https://python.org

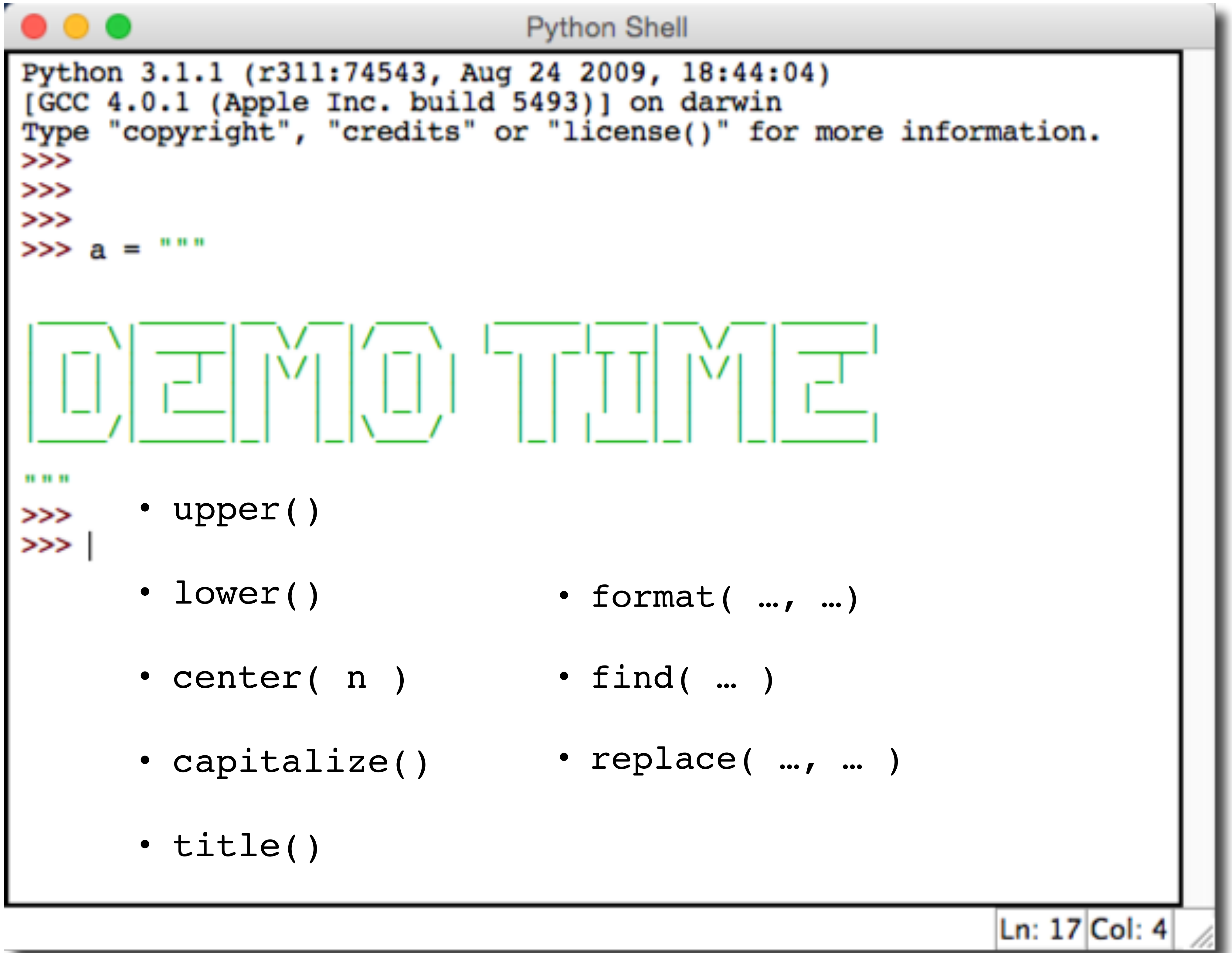
Documentation

Docs

Python 3.x

Python Standard Library

Search for …..



```
Python 3.1.1 (r311:74543, Aug 24 2009, 18:44:04)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
>>>
>>>
>>> a = """
```

```
"""
>>>
>>> |
```

- upper()
- lower()
- center(n)
- capitalize()
- title()
- format(…, …)
- find(…)
- replace(…, …)

Exercise

In a list of Smith student records, one student in each line, we want to block the Smith Id, e.g.990123456, with 990XXXXXX.

Put together an algorithm for doing just that.

Exercise

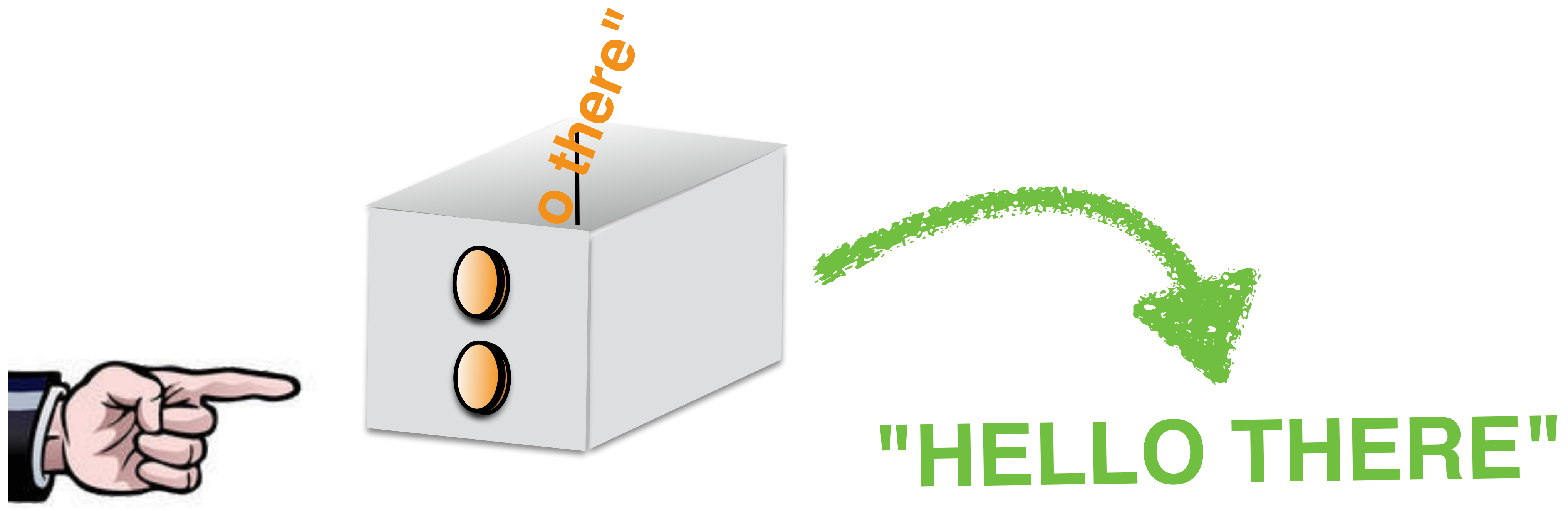
In a list of Smith student records, one student in each line, we want to block the Smith Id, e.g.990123456, with 990XXXXXX.

Put together an algorithm for doing just that.

```
# blockOut990XXXXXX.py
# D. Thiebaut
# blocks Id number in a list of
# student records.

def main():
    records = [ "Alex Monday, 990123456, Tyler House",
                "Lujun Xie, 990999999, Ducket House",
                "Maria Helena Morena, 990666777, King House"  ]

    for line in records:
        print( line )

main()
```

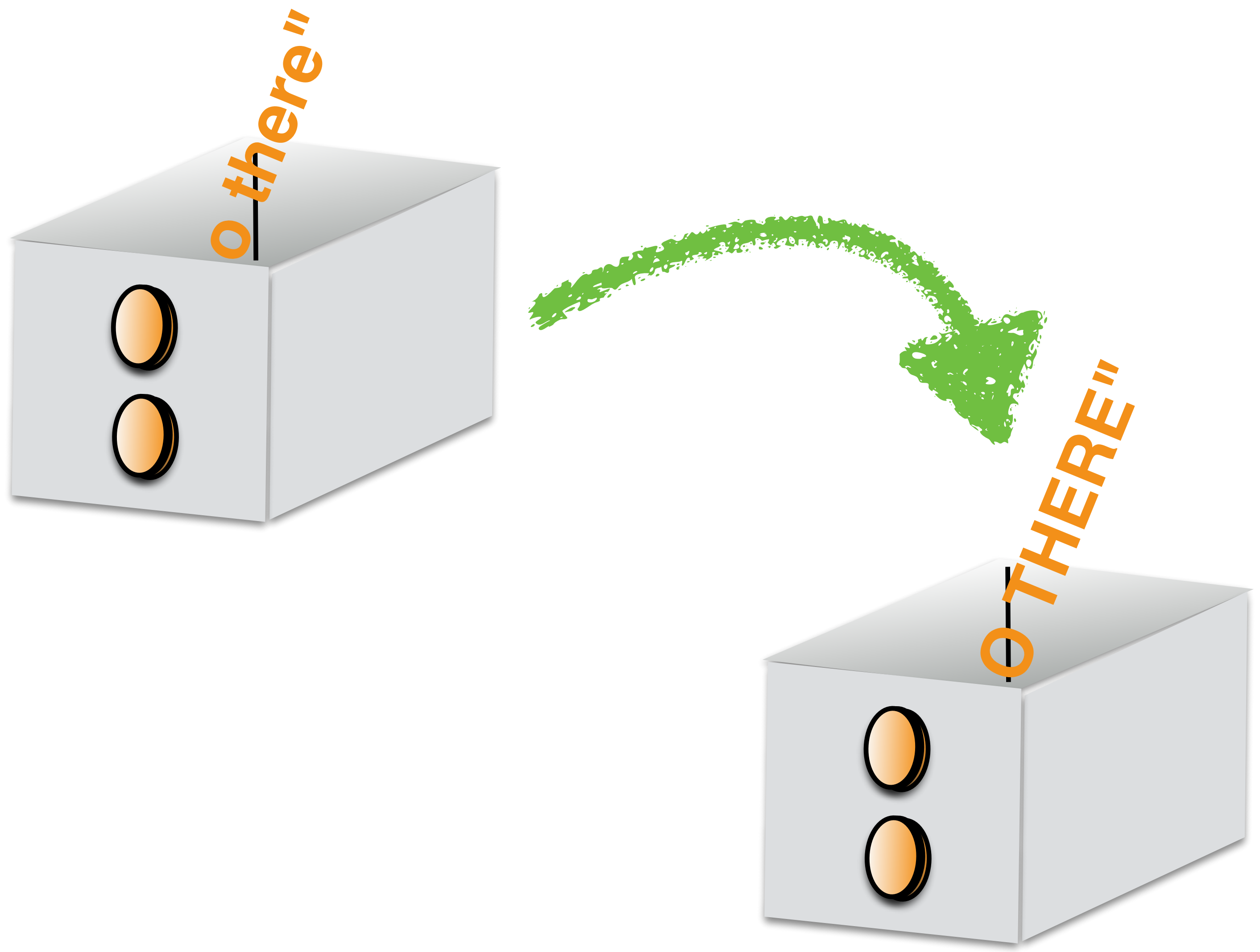
Multiple Transformations

```
"hello there".upper()
```

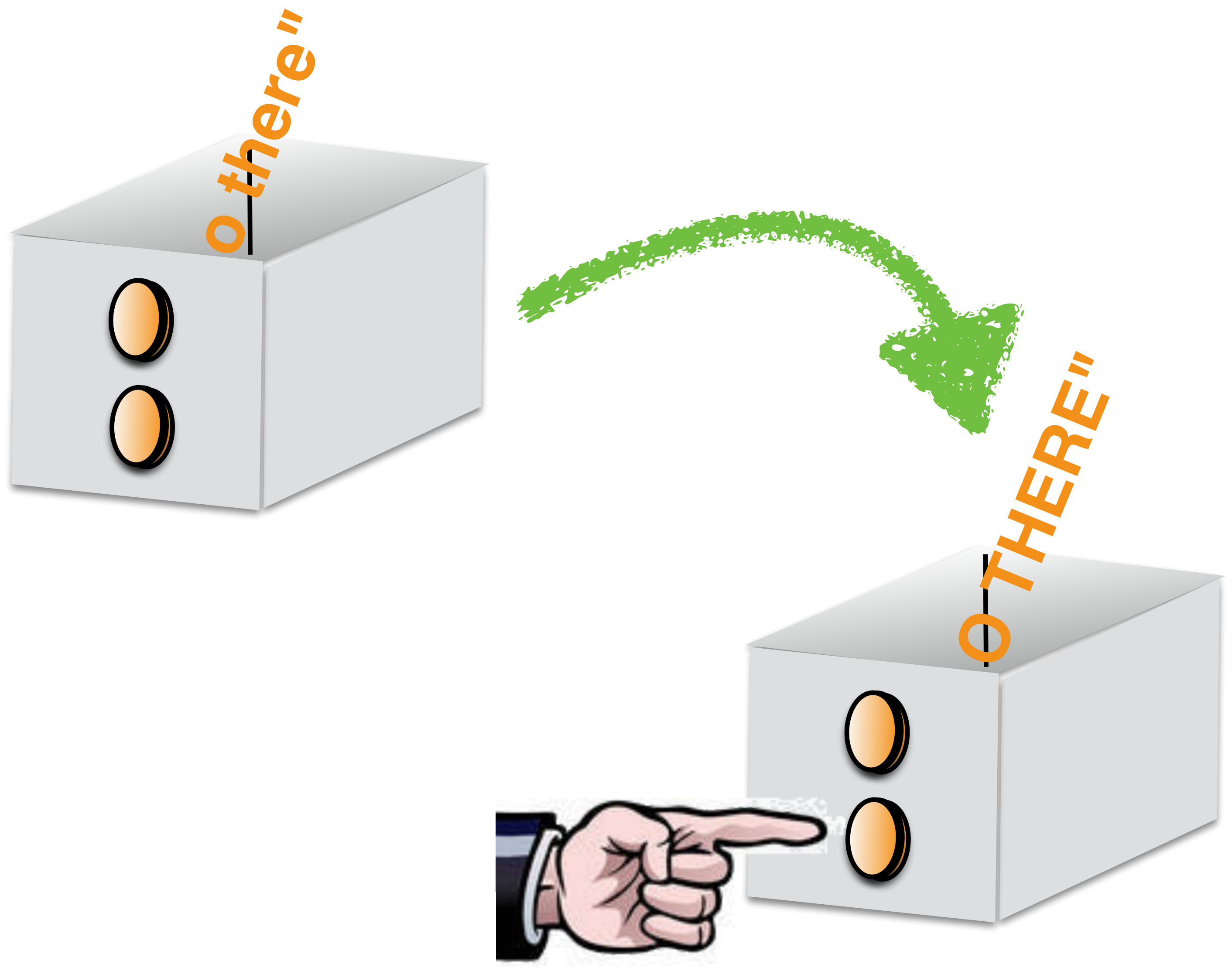


upper()

```
"hello there".upper()
```

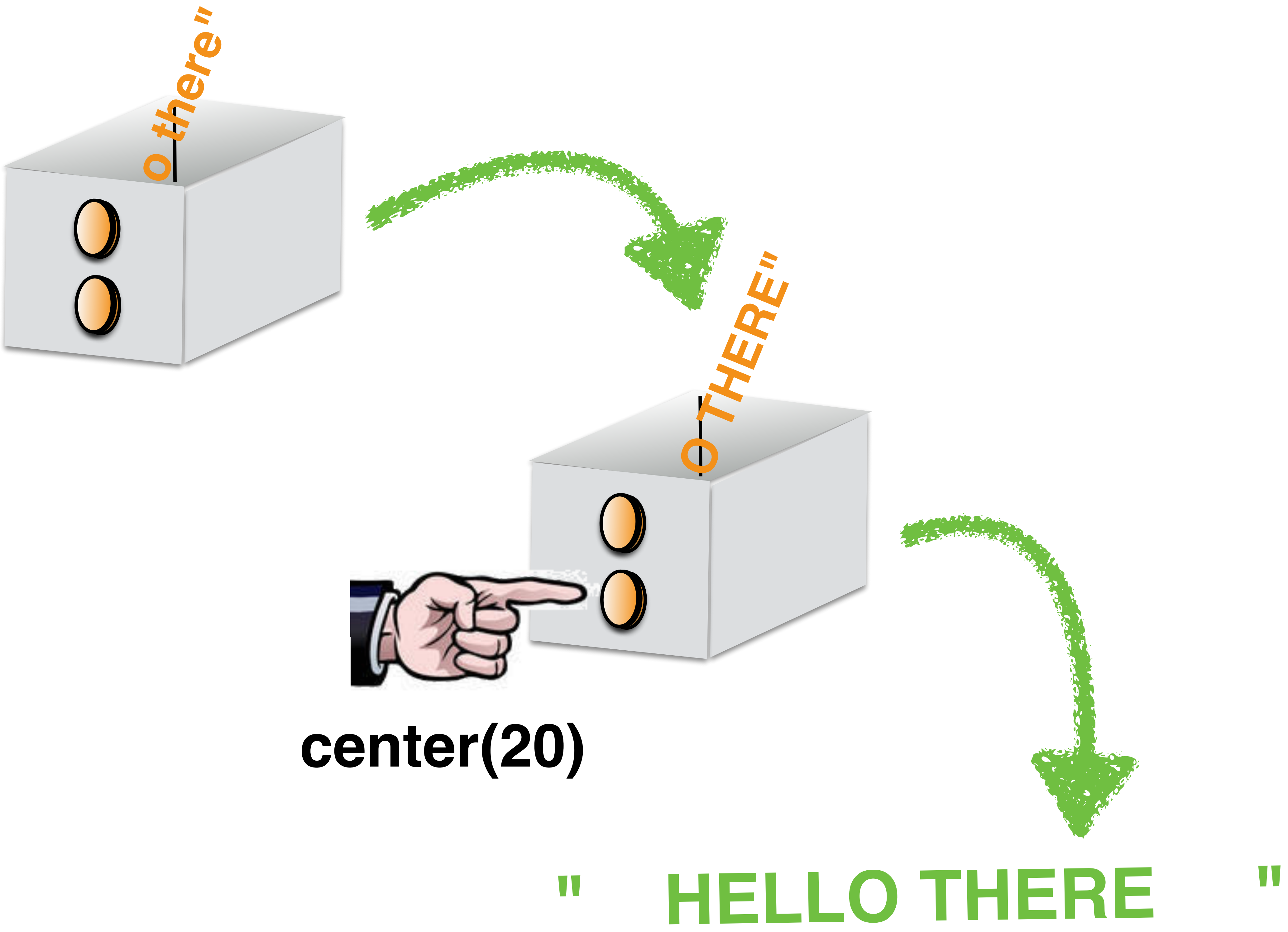


```
"hello there".upper()
```



center(20)

```
"hello there".upper().center( 20 )
```



Exercises (Part 1)

Write a program that **prompts** the user for her first and last name, and **prints** both, in the **proper case**, **centered** in 60 spaces

Write a program that takes a string, where a **phone number** is located. The phone number always **start at Index 7**, and contains 10 numbers (no spaces). **Print** the **phone number only**, in the form:

```
 (xxx) xxx-xxxx
```

Exercises (Part 2)

```
book = ["Ulysses",
"James Joyce",
"Stately, plump Buck Mulligan came
from the stairhead," ,
"bearing a bowl of lather on which
a mirror and a razor lay crossed."
]
```

Project Gutenberg
http://www.gutenberg.org/cache/epub/4300/pg4300.txt

Write a program that takes the list ***book*** (above), and prints it, the **title** centered, all caps, in a line of 60 chars, and the **author**'s name, capitalized, and centered in 60 chars, followed by a **blank line**, followed by the **first sentence** (whichever way it comes out).

Split(), the workhorse of string methods

Functions

Function Parameters

Functions Returning Values

split()

```
line = "The quick red fox jumped over the dog"
line = "The quick red fox jumped over the dog"

line.split( ' ' )        # that's a space between the quotes
```

```
line = "The□quick□red□fox□jumped□over□the□dog"


line.split( '□' )
```

```
line = "The␣quick␣red␣fox␣jumped␣over␣the␣dog"

line.split( '␣' )

[ "The", "quick",  "red",
  "fox", "jumped", "over",
  "the", "dog" ]
```

```
line = "The quick red fox jumped over the dog"


line.split( 'o' )

          [ "The quick red f",
           "x jumped",
           "ver the d",
           "g" ]
```

```
line = "hello Ruth"

line.split( 'h' )

        ["", "ello Rut", ""]
```

```
line = """The quick red fox
jumped over
the lazy brown sleeping dog"""

line.split( ??? )

        [ "The quick red fox",
        "jumped over",
        "the lazy brown sleeping dog" ]
```

```
line = """The quick red fox
jumped over
the lazy brown sleeping dog"""

line.split( "\n" )
```

```
[ "The quick red fox",
"jumped over",
"the lazy brown sleeping dog" ]
```

```
line = """The quick red fox
jumped over
the lazy brown sleeping dog"""
```

Equivalent

```
line = "The quick red fox\njumped over\nthe lazy…dog"
```

(no room on slide!)

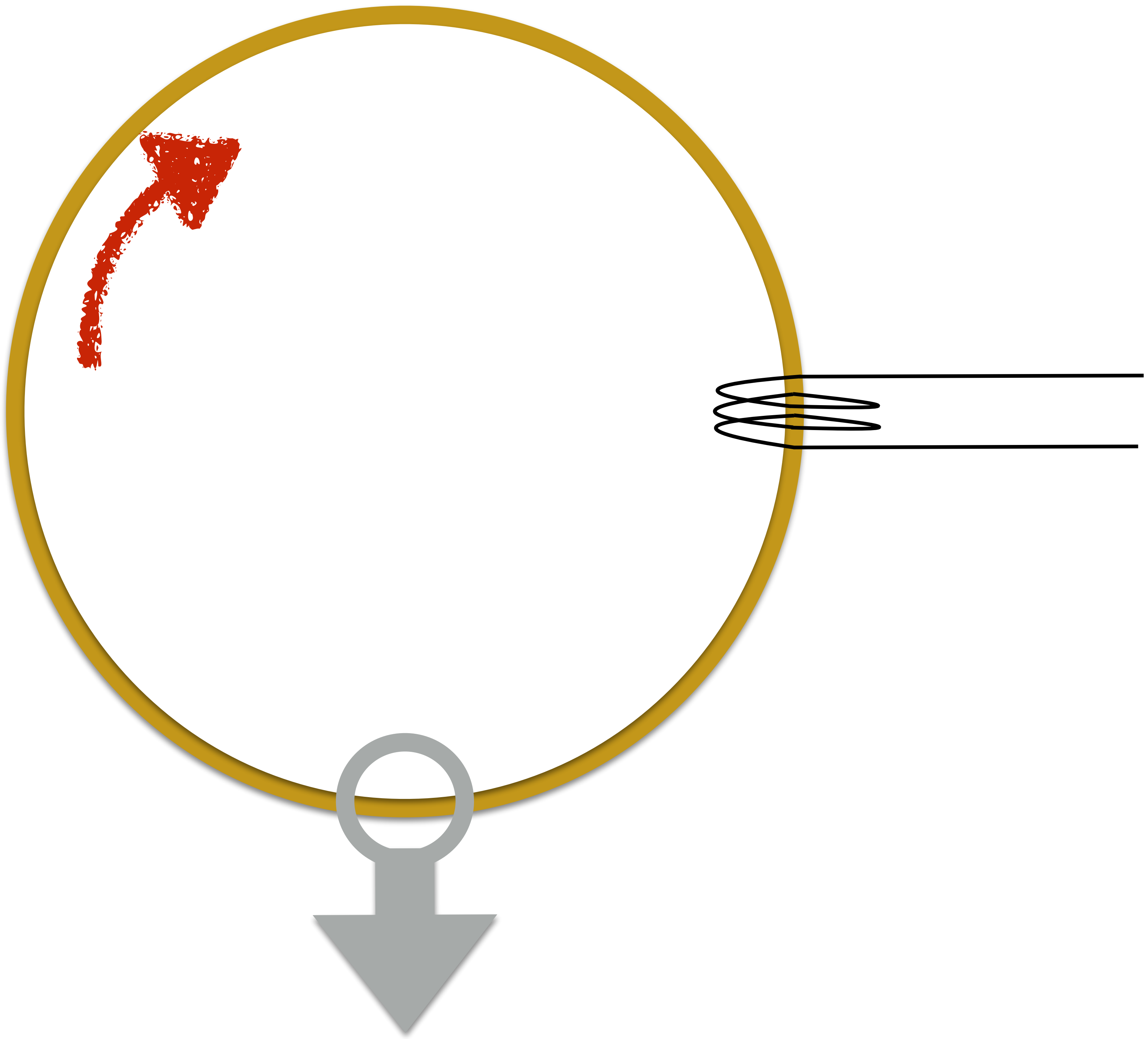
Exercise

```
poem = """Chocolate
Chocolate is the first luxury.
It has so many things wrapped up in it:
Deliciousness in the moment,
childhood memories,
and that grin-inducing
feeling of getting a reward for being good.
--Mariska Hargitay"""

# display each line centered in 60 spaces.
# first line all uppercase.
# last line right justified in 60 spaces.
```

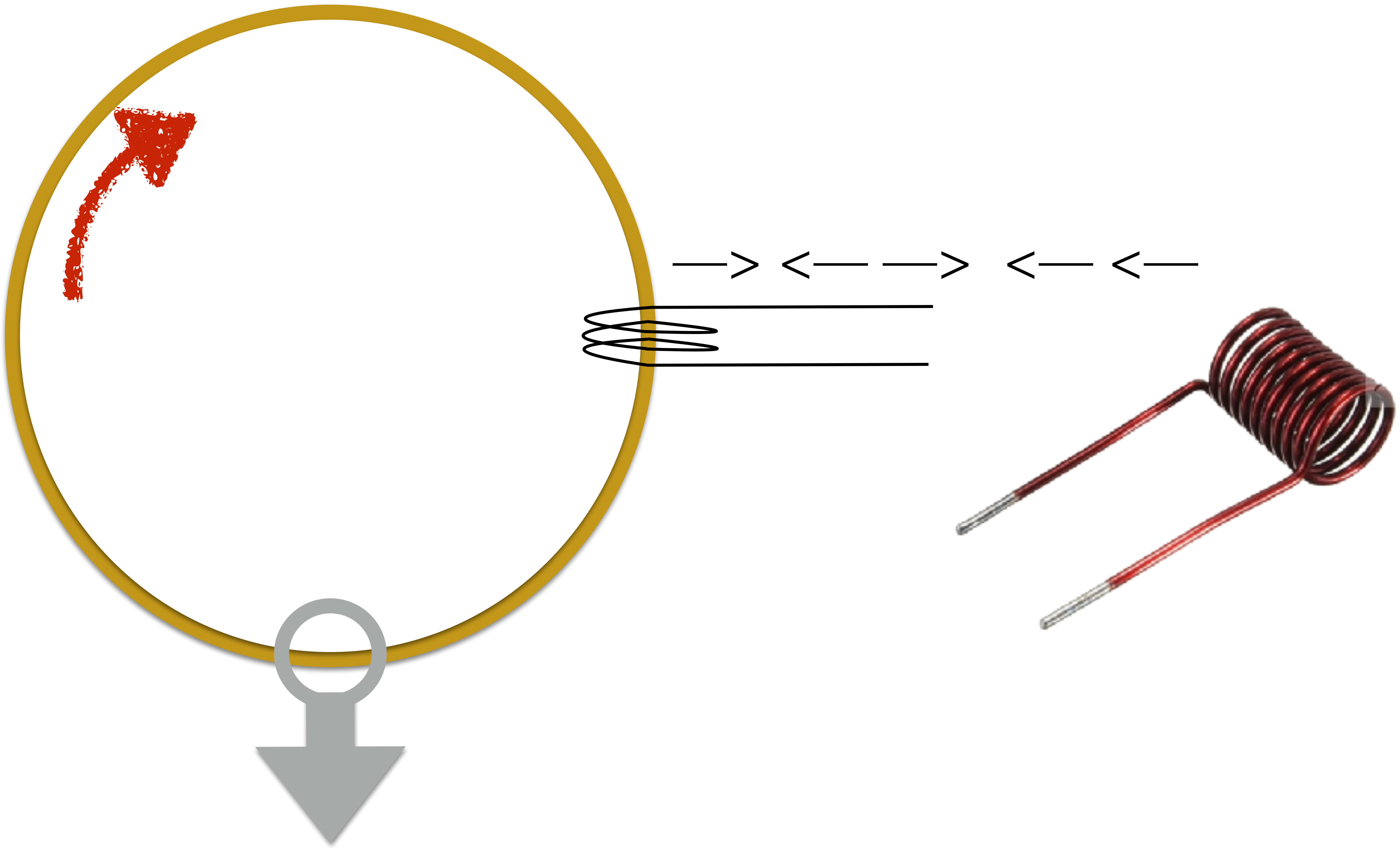
File Processing

How a Hard Disk Works

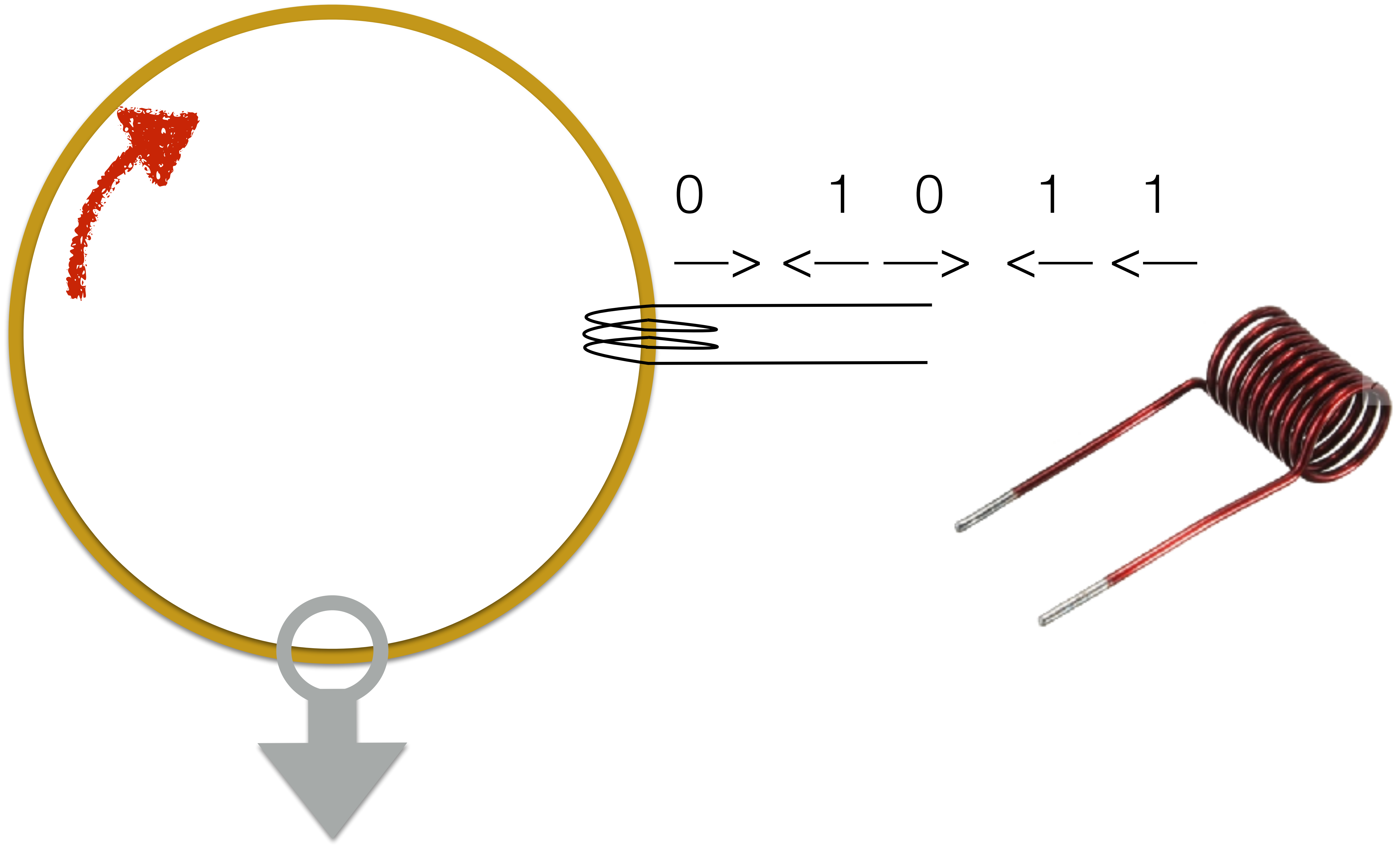
1



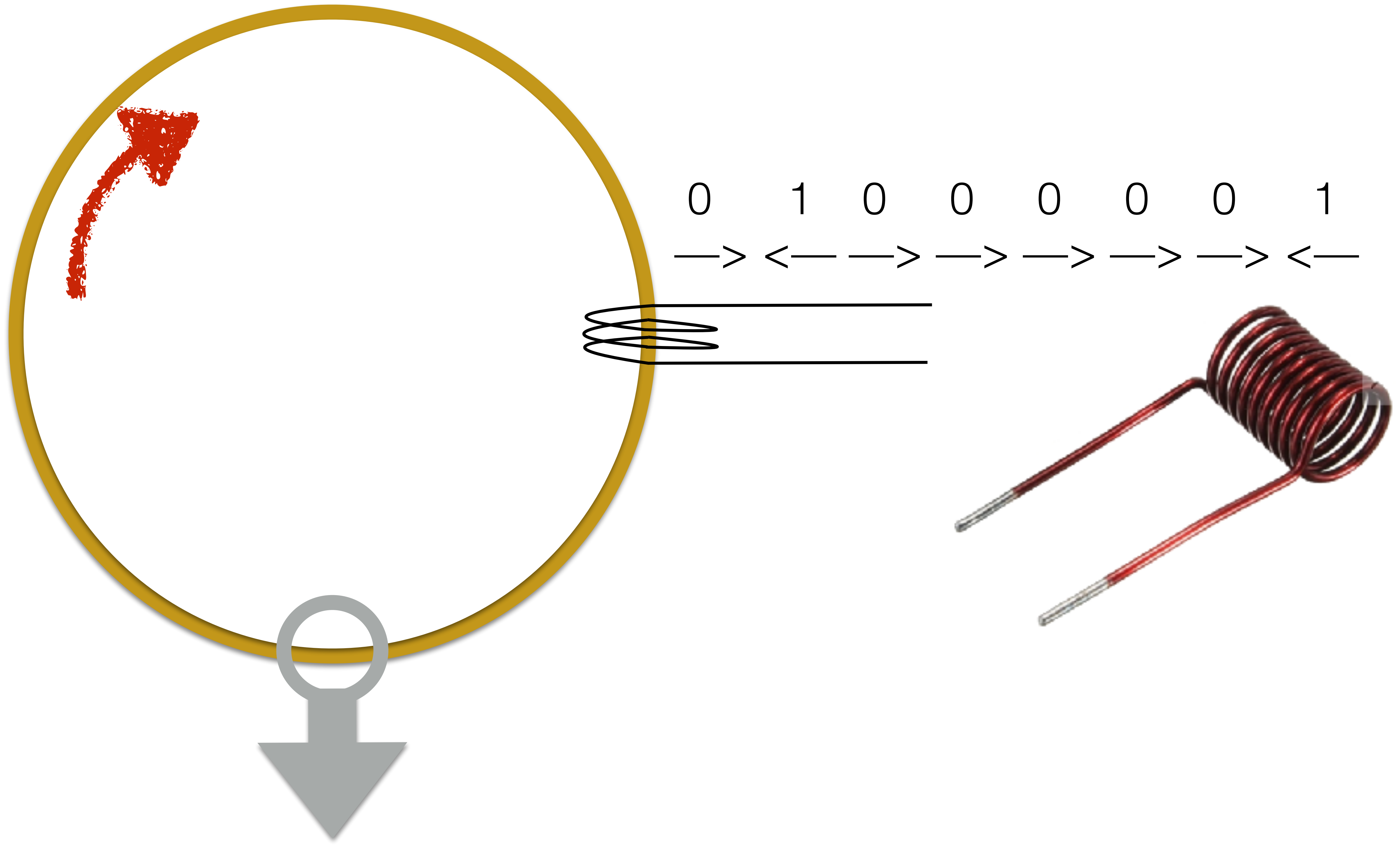
N	S	N	N	N	S	N	S	N	S	N	N	N	N	S	N
S	N	S	S	S	N	S	N	S	N	S	S	S	S	N	S



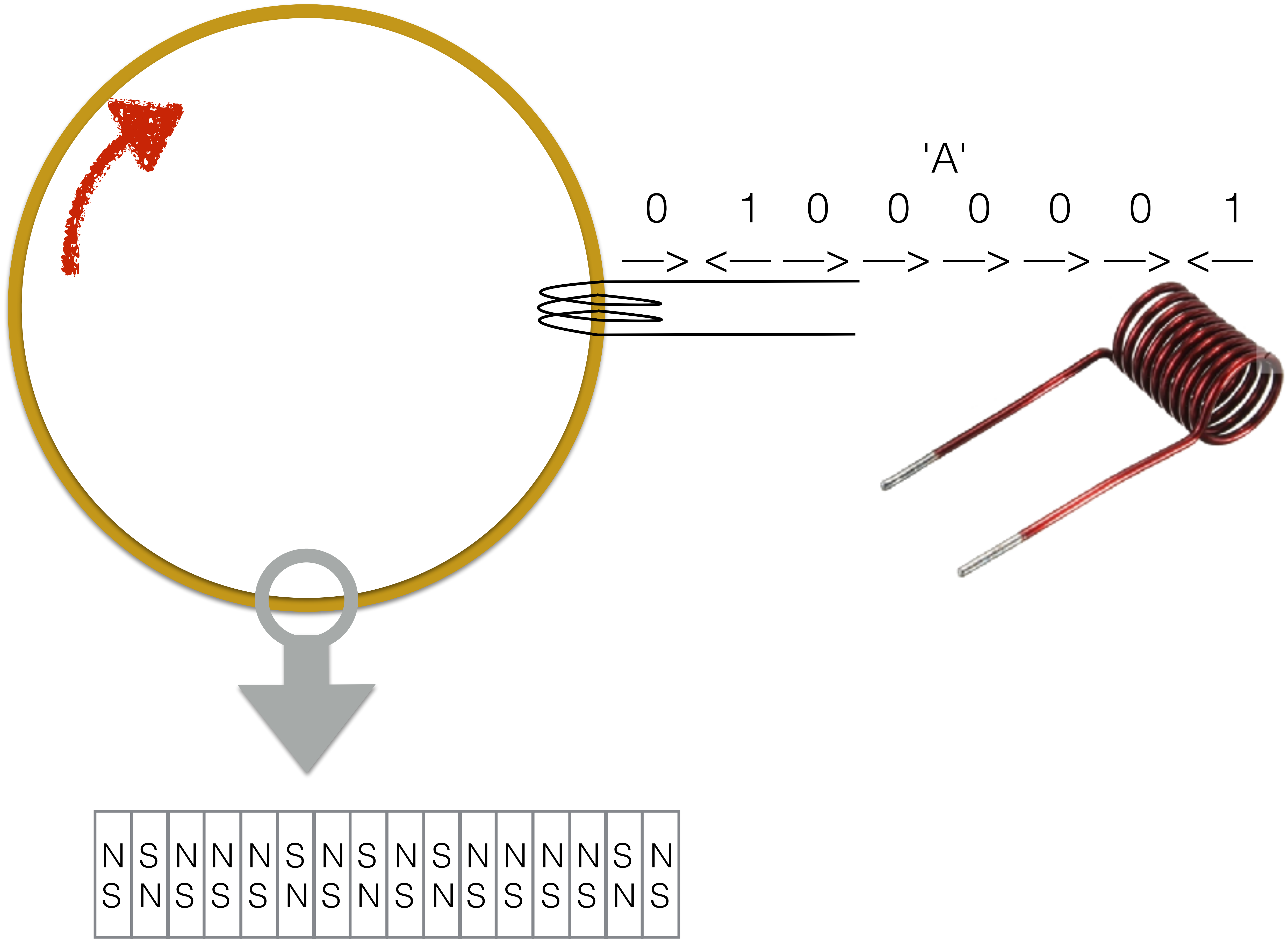
N	S	N	N	N	S	N	S	N	S	N	N	N	N	S	N
S	N	S	S	S	N	S	N	S	N	S	S	S	S	N	S



N	S	N	N	N	S	N	S	N	S	N	N	N	N	S	N
S	N	S	S	S	N	S	N	S	N	S	S	S	S	N	S



N	S	N	N	N	S	N	S	N	S	N	N	N	N	S	N
S	N	S	S	S	N	S	N	S	N	S	S	S	S	N	S


'A'
0 1 0 0 0 0 0 1
—> <— —> —> —> —> —> <—
N S N N N S N S N S N N N N S N
S N S S S N S N S N S S S S N S

RA60P

What are Files?

- Containers of bits, organized in bytes
- May contain text, images, music, movies, programs, applications, list of files (folders)…
- In Python, we will first play with **text files**

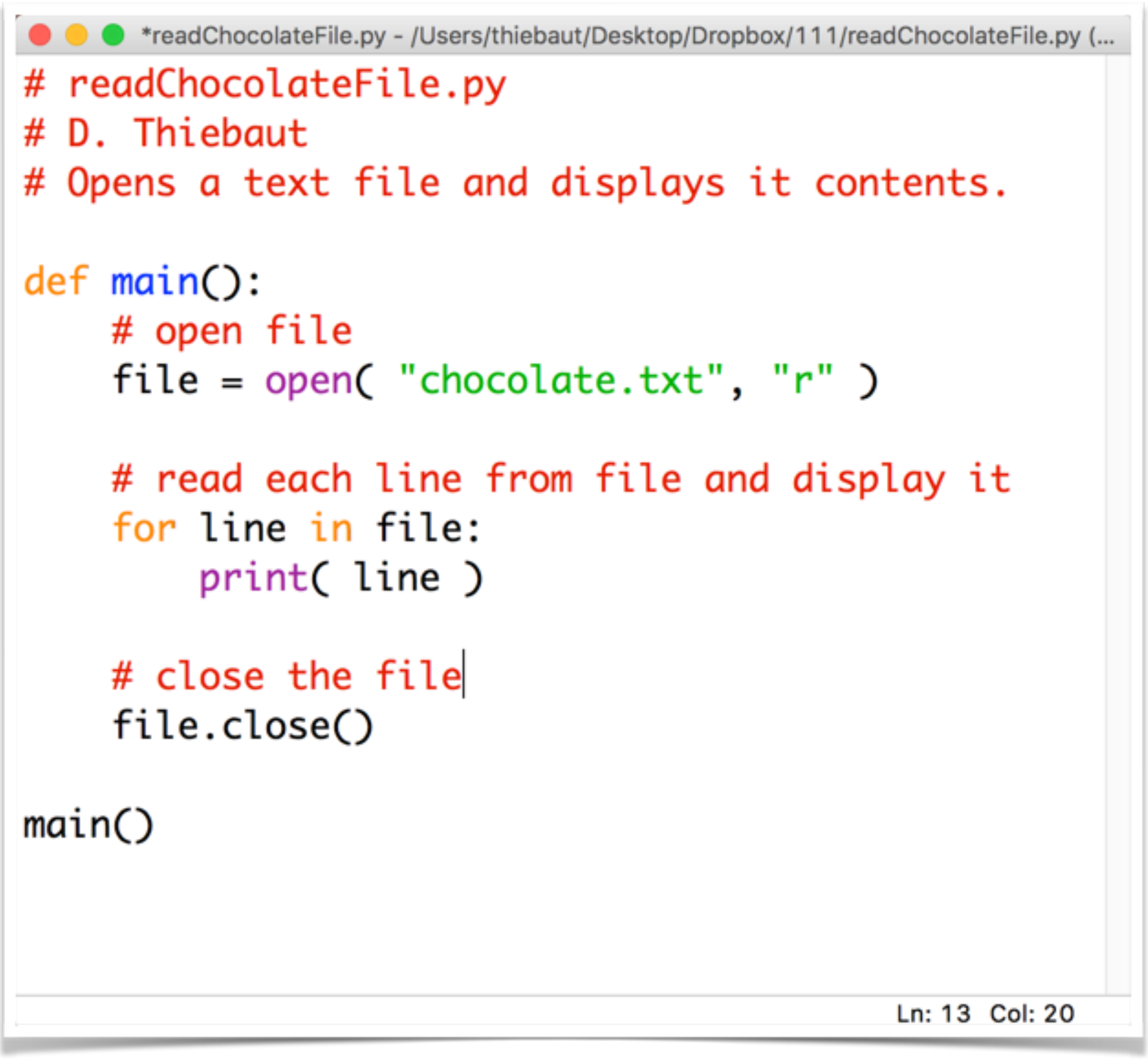
Mini Lab

- Create a file containing following text (use Notepad or TextEdit):

  Strength is the capacity to break
  a Hershey bar into four pieces
  with your bare hands - and then
  eat just one of the pieces.
  —Judith Viorst

- Save it under the name ***chocolate.txt*** in the same directory where you store your python programs

- Write the following Python program:


*readChocolateFile.py - /Users/thiebaut/Desktop/Dropbox/111/readChocolateFile.py (...

```
# readChocolateFile.py
# D. Thiebaut
# Opens a text file and displays it contents.

def main():
    # open file
    file = open( "chocolate.txt", "r" )

    # read each line from file and display it
    for line in file:
        print( line )

    # close the file
    file.close()

main()
```

Ln: 13 Col: 20