http://www.cs.cornell.edu/courses/cs1110/2022sp

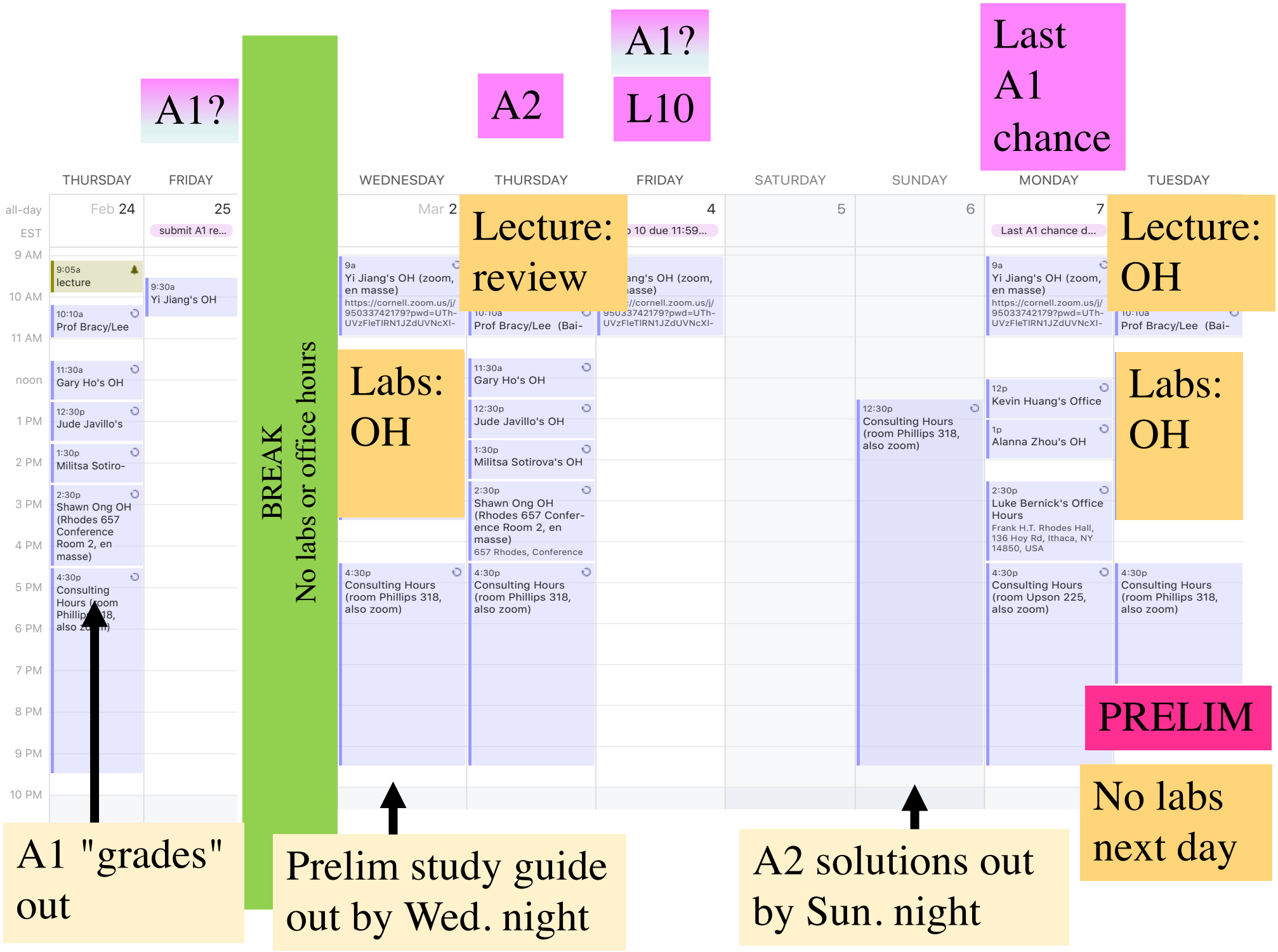
Lecture 10:
Lists and Sequences

(Sections 10.0-10.2, 10.4-10.6, 10.8-10.13, 12.1, 12.2)

CS 1110

Introduction to Computing Using Python

[E. Andersen, A. Bracy, D. Gries, L. Lee, S. Marschner, C. Van Loan, W. White]


A1?
A2
A1?
L10
Last A1 chance
BREAK
No labs or office hours
Lecture: review
Lecture: OH
Labs: OH
Labs: OH
PRELIM
THURSDAY
FRIDAY
WEDNESDAY
THURSDAY
FRIDAY
SATURDAY
SUNDAY
MONDAY
TUESDAY
all-day
EST
Feb 24
25
Mar 2
4
5
6
7
submit A1 re...
Last A1 chance d...
9 AM
10 AM
11 AM
noon
1 PM
2 PM
3 PM
4 PM
5 PM
6 PM
7 PM
8 PM
9 PM
10 PM
9:05a lecture
9:30a Yi Jiang's OH
10:10a Prof Bracy/Lee
11:30a Gary Ho's OH
12:30p Jude Javillo's
1:30p Militsa Sotiro-
2:30p Shawn Ong OH (Rhodes 657 Conference Room 2, en masse)
4:30p Consulting Hours (room Phillips 318, also zoom)
9a Yi Jiang's OH (zoom, en masse) https://cornell.zoom.us/j/95033742179?pwd=UTh-UVzFleTlRN1JZdUVNcXI-
10:10a Prof Bracy/Lee (Bai-
11:30a Gary Ho's OH
12:30p Jude Javillo's OH
1:30p Militsa Sotirova's OH
2:30p Shawn Ong OH (Rhodes 657 Confer-ence Room 2, en masse) 657 Rhodes, Conference
4:30p Consulting Hours (room Phillips 318, also zoom)
12:30p Consulting Hours (room Phillips 318, also zoom)
9a Yi Jiang's OH (zoom, en masse) https://cornell.zoom.us/j/95033742179?pwd=UTh-UVzFleTlRN1JZdUVNcXI-
12p Kevin Huang's Office
1p Alanna Zhou's OH
2:30p Luke Bernick's Office Hours Frank H.T. Rhodes Hall, 136 Hoy Rd, Ithaca, NY 14850, USA
4:30p Consulting Hours (room Upson 225, also zoom)
10:10a Prof Bracy/Lee (Bai-
4:30p Consulting Hours (room Phillips 318, also zoom)
A1 "grades" out
Prelim study guide out by Wed. night
A2 solutions out by Sun. night
No labs next day

Today: A1 feedback out, revisions enabled

- Set your CMS notifications to get email when "one of your grades is changed"

- Watch for instruction announcement, but:
  the *expected* first "grade" is **-99999**
  = "there's something we'd like you to fix"

- Revising will change -99999 to -9999 to … until 10/10!

Magical traditions: names have power

- **Function:** a genie in a bottle you can **call** on.
  - You put input into the bottle, the genie assigns them "private nicknames" (the **parameter** names)
  - Does "hidden magic"/"scratch work" inside its **call frame** -- the "bottle".
  - Can *delegate* by calling other genies.

- **Call stack:** list of *pending* delegated function calls (to-do list).

- **Object:** can be affected by a function that knows its "secret name", or **id**
  - Created by a special function call "let there be a new Point": **Point(…)**; returns the secret name of new object so you can access it --- *if* you store it somewhere safe (a variable).

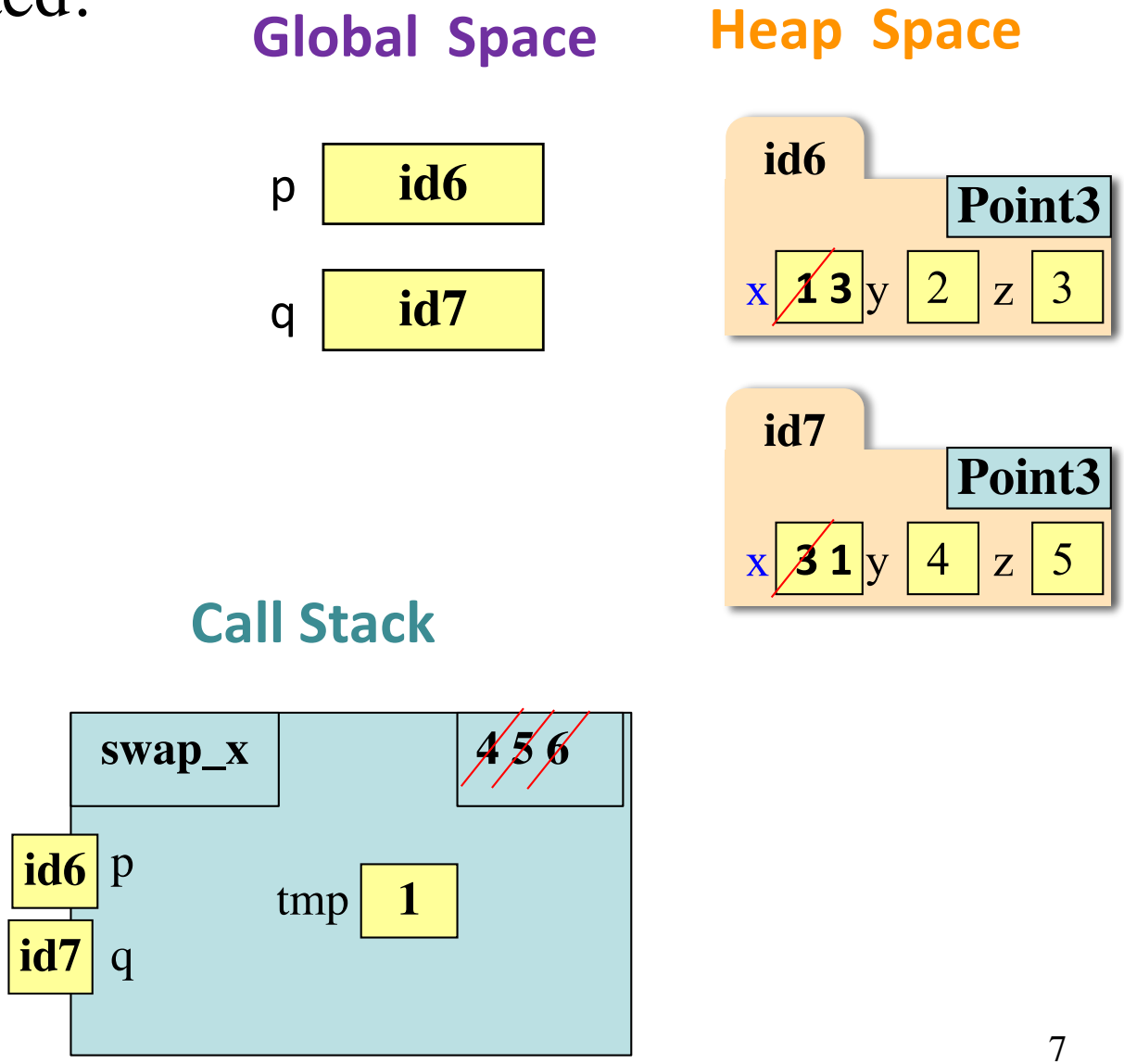
Analogies to ('foreign') languages

- The "dot" (.) is like an apostrophe: x.y is like "x's y", or the "y that belongs to x"

- **methods:** functions :: irregular verbs : verbs
  - Different calling syntax: some_string.

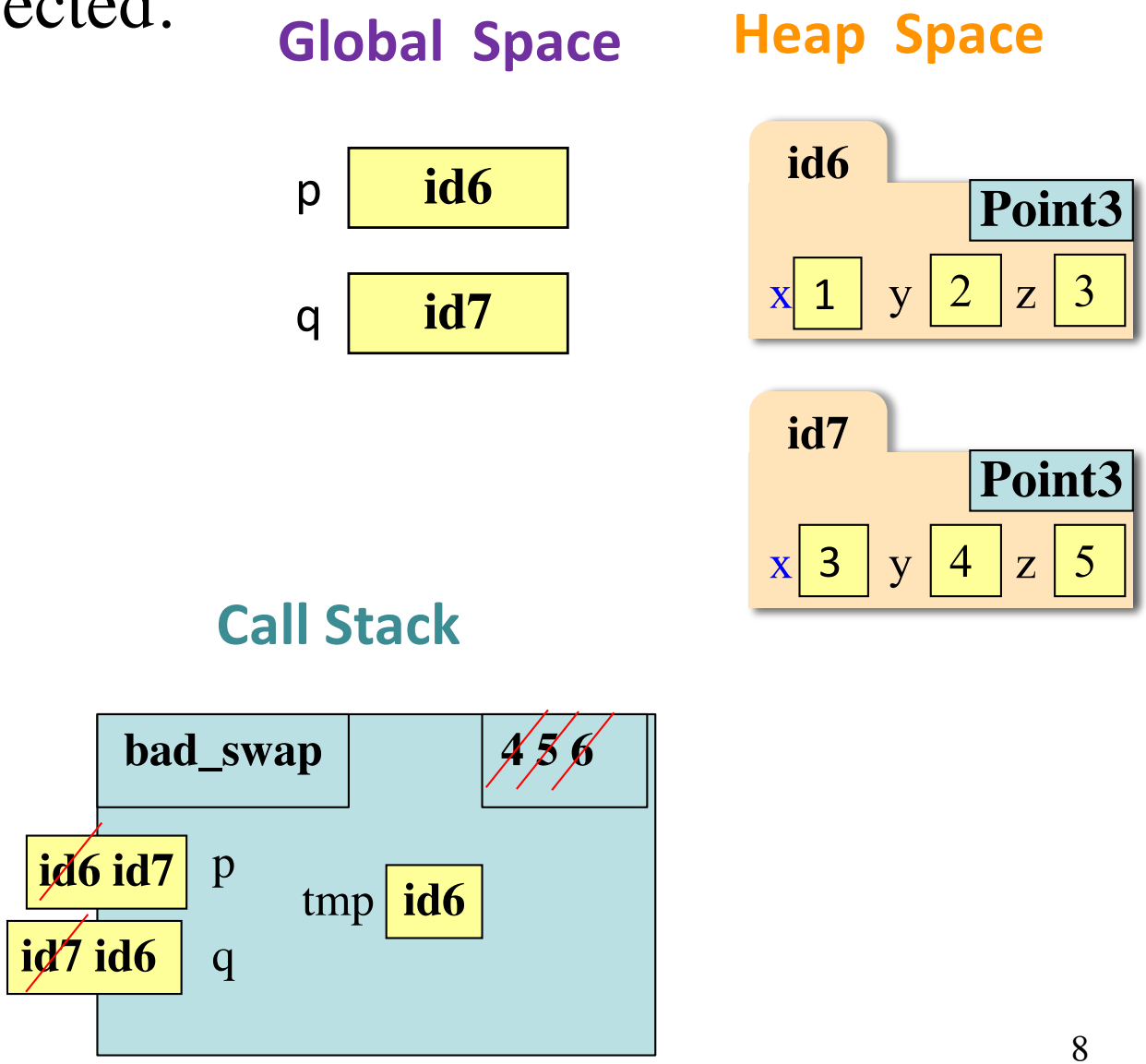
Depicted: when line 6 of swap_x() has been executed, but before function ends (returns None).
The objects *were* affected.

```
1   import shapes
2
3   def swap_x(p, q):
4       tmp = p.x
5       p.x = q.x
6       q.x = tmp
7
8   p = shapes.Point3(1,2,3)
9   q = shapes.Point3(3,4,5)
10
11  swap_x(p, q)
```



Depicted: when line 6 of bad_swap() has been executed, but before function ends (returns None).
The objects *weren't* affected.

```
1    import shapes
2
3    def bad_swap(p, q):
4        tmp = p
5        p = q
6        q = tmp
7
8    p = shapes.Point3(1,2,3)
9    q = shapes.Point3(3,4,5)
10
11   bad_swap(p, q)
```



Call stack example (Spring 2021 A2)

```
7     def replace_first(s, target, rep):
8…        """Returns: copy of s with the FIRST instance of target in s
      replaced by rep…"""
16        pos = s.index(target)
17        before = s[:pos]
18…       after = s[pos + len(target):]
20        return before + rep + after
21
22    def replace2(s, target, rep):
23…       """Replace first two occurrences of target in s…"""
27        before = s[:s.index(target)+len(target)]
28        after = s[s.index(target)+len(target):]
29        if s.count(target) > 1:
30            after = replace_first(after, target, rep)
31        else:
32…           WARNING = <does something>
34…       return replace_first(before,target,rep)+after

37    x = "Millissippi"
```

```
replace2(x, 'L', 's')
```

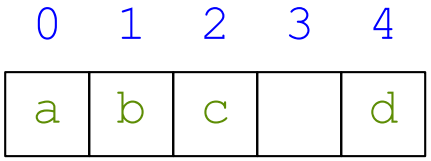
vs.

```
replace2(x, 'LL', 's')
```

Sequences: Lists of Values

String

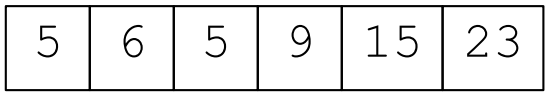
- s = 'abc d'



- Put characters in quotes
  - Use \' for quote character
- Access characters with []
  - s[0] is 'a'
  - s[5] causes an error
  - s[0:2] is 'ab' (excludes c)
  - s[2:] is 'c d'
- len(s) → 5, length of string

List

- x = [5, 6, 5, 9, 15, 23]



- Put values inside [ ]
  - Separate by commas
- Access **values** with []
  - x[0] is 5
  - x[6] causes an error
  - x[0:2] is [5, 6] (excludes 2nd 5)
  - x[3:] is [9, 15, 23]
- len(x) → 6, length of list

Sequence is a name we give to both

Lists Have Methods Similar to String

```
x = [5, 6, 5, 9, 15, 23]
```

- <list>.index(<value>)
  - Return position of the value
  - **ERROR** if value is not there
  - x.index(9) evaluates to 3
- <list>.count(<value>)
  - Returns number of times value appears in list
  - x.count(5) evaluates to 2

But to get the length of a list you use a function, not a class method:

len(x)

~~x.len()~~

Representing Lists

Wrong:

Global Space

x | 5, 6, 7, -2

Correct:

Global Space

x | id1

Heap Space

id1

list

0	5
1	7
2	4
3	-2

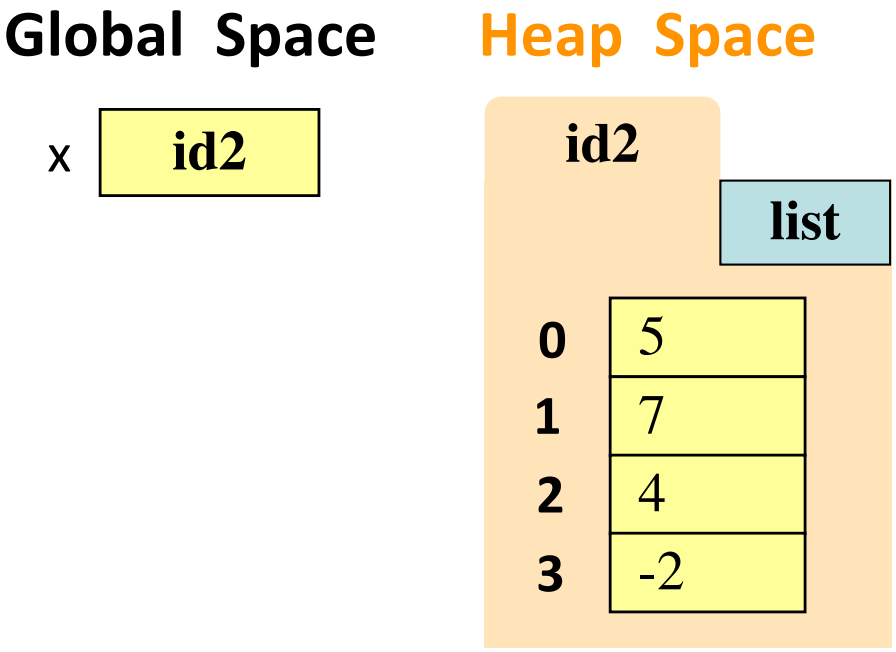
Indices

x = [5, 7, 4,-2]

Lists: objects with special syntax (like nouns with "weird" plurals)

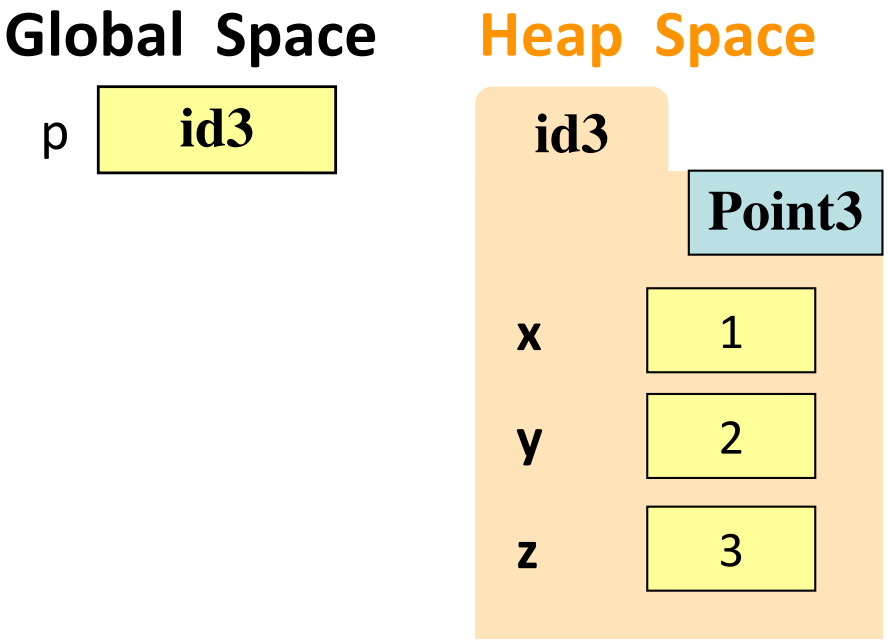
List

- Attributes are indexed
  - Example: x[2]



Objects

- Attributes are named
  - Example: p.x



Lists Can Hold Any Type

Expression evaluates to value; value goes in list

```
list_of_integers = [5, 7, 3+1, -2]
list_of_strings = ['h', 'i', '', 'there!']
```

Global Space

list_of_integers **id1**

list_of_strings **id2**

Heap Space

id1

list

0	5
1	7
2	4
3	-2

id2

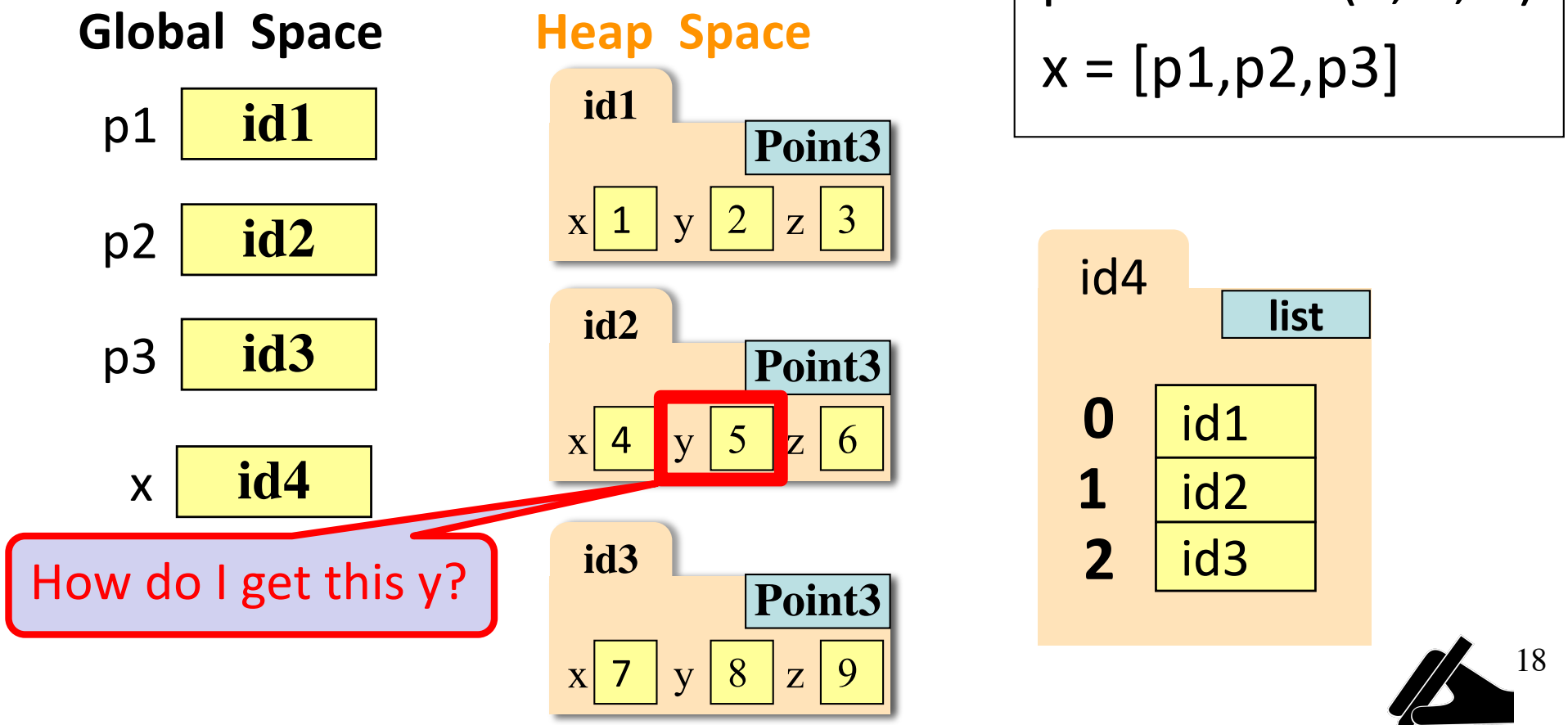
list

0	'h'
1	'i'
2	''
3	'there!'

Lists of Objects

- List elements are variables
  - Can store base types and ids
  - Cannot store folders

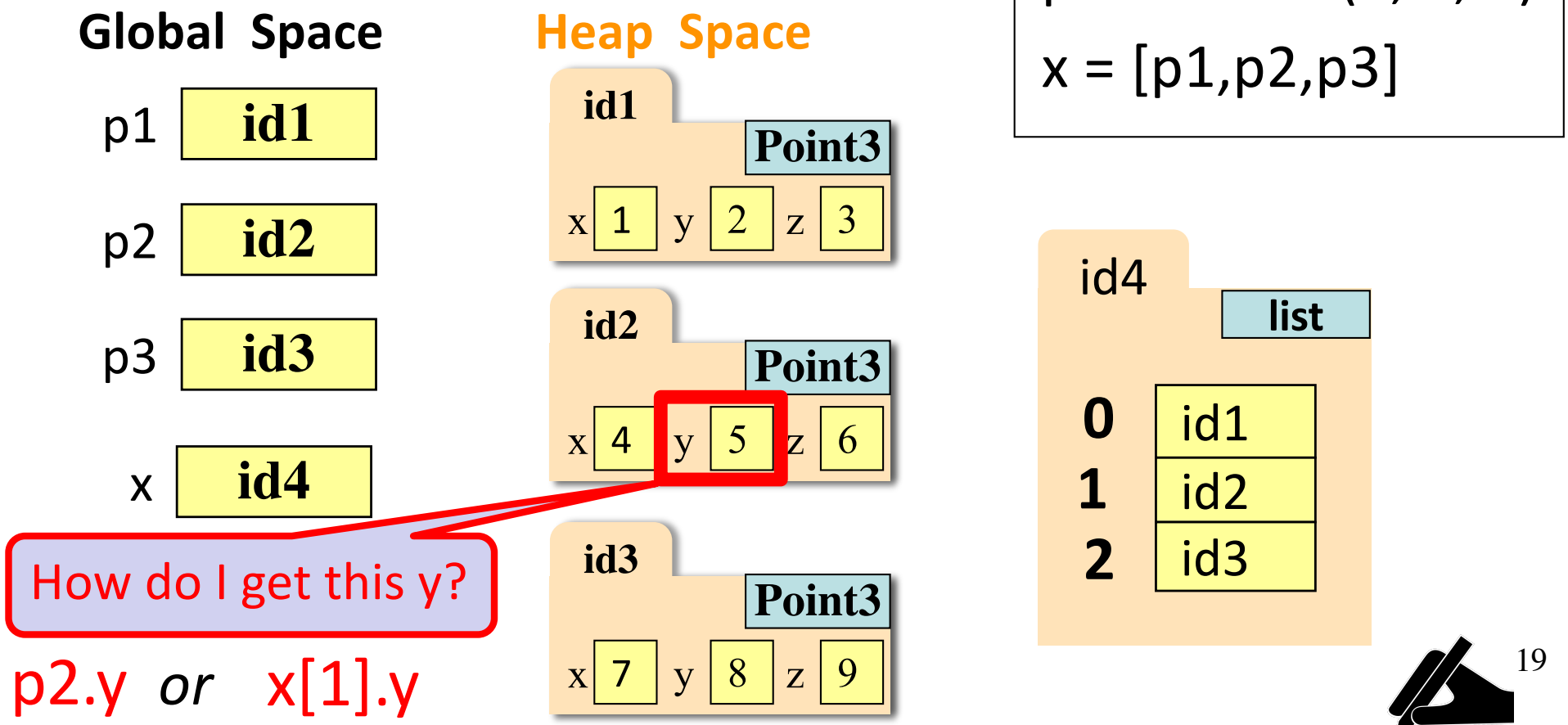
```
p1 = Point3(1, 2, 3)
p2 = Point3(4, 5, 6)
p3 = Point3(7, 8, 9)
x = [p1,p2,p3]
```



Lists of Objects

- List elements are variables
  - Can store base types and ids
  - Cannot store folders

```
p1 = Point3(1, 2, 3)
p2 = Point3(4, 5, 6)
p3 = Point3(7, 8, 9)
x = [p1,p2,p3]
```



List is *mutable*; strings are not

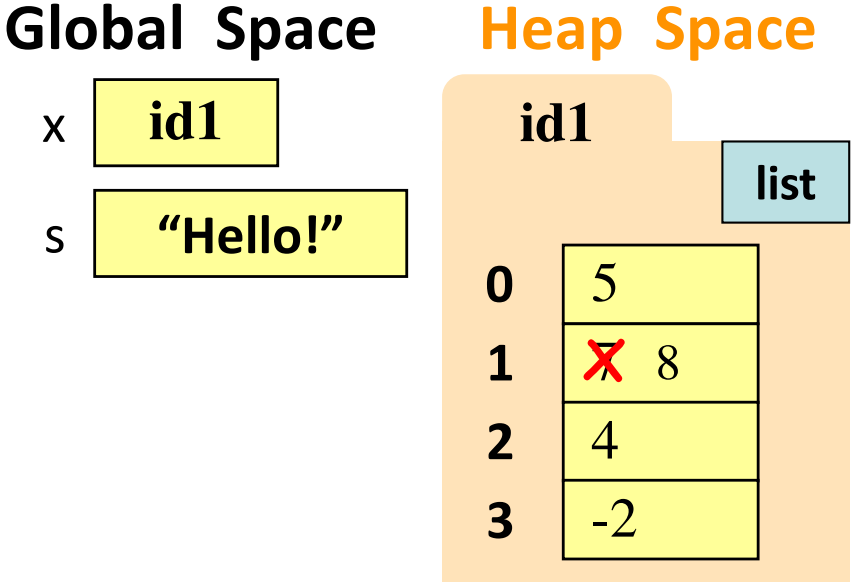
- **Format**:

  <var>[<index>] = <value>

  - Reassign at index
  - Affects folder contents
  - Variable is unchanged

- Strings cannot do this
  - Strings are **immutable**

```
x = [5, 7,4,-2]
x[1] = 8
s = “Hello!”
s[0] = ‘J’
TypeError: 'str' object does not
support item assignment
```



List Methods Can Alter the List

x = [5, 6, 5, 9]

y = [15, 16, 15, 19]

See Python API for more

- <list>.append(<value>)
  - Adds a new value to the end of list
  - x.append(-1) *changes* the list to [5, 6, 5, 9, -1]
- <list>.insert(<index>,<value>)
  - Puts value into list at index; shifts rest of list right
  - **y**.insert(2,-1) *changes* the list to [15, 16, -1, 15, 19]
- <list>.sort()

What do you think this does?

Q2: Swap List Values?

```
def swap(b, h, k):
    """Procedure swaps b[h] and b[k] in b
       Precondition: b is a mutable list, h
       and k are valid positions in the list"""”
1   temp= b[h]
2   b[h]= b[k]
3   b[k]= temp
```

Global Space

x | id4

Heap Space

id4

0	5
1	4
2	7
3	6
4	8

```
x = [5,4,7,6,8]
swap(x, 3, 4)
print(x[3])
```

What gets printed?

A: 8
B: 6
C: Something else
D: I don’t know

List Slices Make Copies:
a slice of a list is a new list

```
x = [5, 6, 5, 9]
y = x[1:3]
```

Global Space

x	id5
y	id6

Heap Space

id5

list

0	5
1	6
2	5
3	9

id6

list

0	6
1	5

copy means
new folder

Q3: List Slicing

- Execute the following:

```
>>> x = [5, 6, 5, 9, 10]
>>> y = x[1:]
>>> y[0] = 7
```

- What is x[1]?

A: 7
B: 5
C: 6
D: **ERROR**
E: I don't know

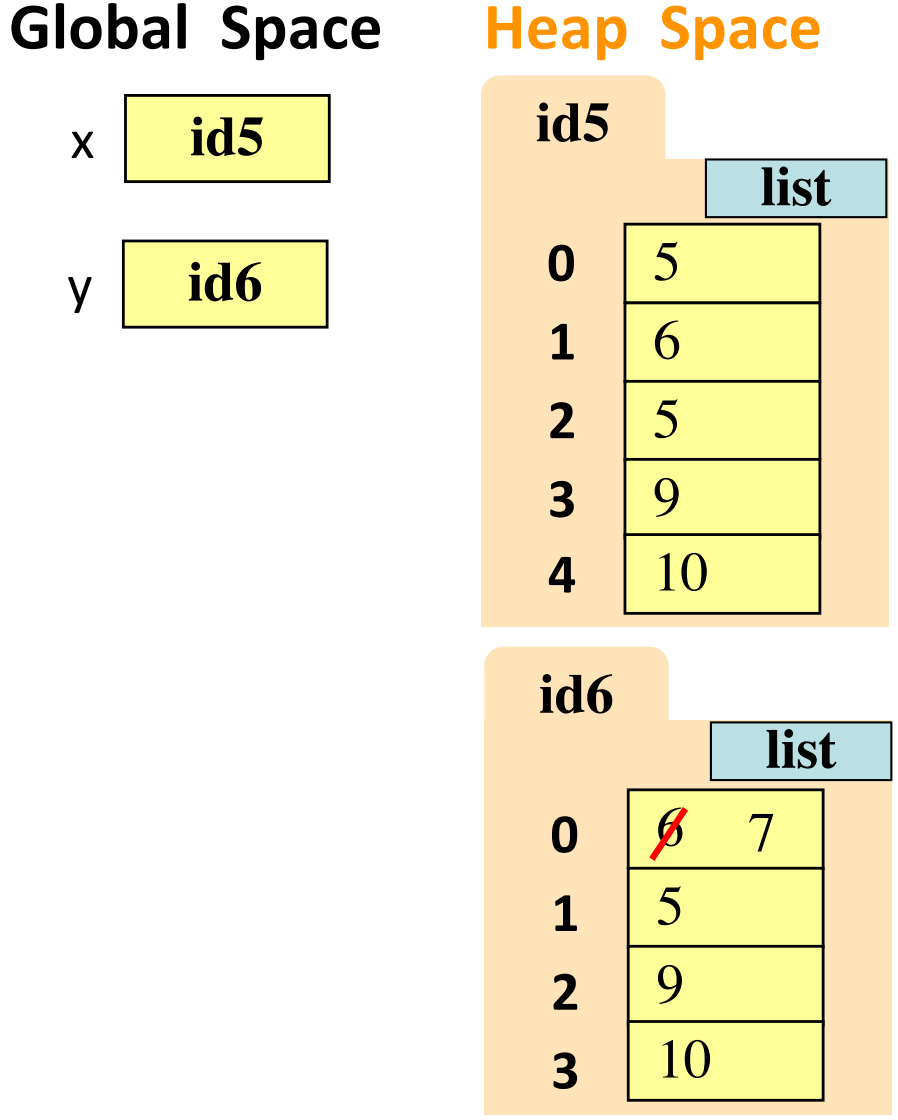
A3: List Slicing

- Execute the following:

```
>>> x = [5, 6, 5, 9, 10]
>>> y = x[1:]
>>> y[0] = 7
```

- What is x[1]?

A: 7
B: 5
C: 6 **CORRECT**
D: **ERROR**
E: I don't know



Q4

- Execute the following:

```
>>> x = [5, 6, 5, 9, 10]
>>> y = x
>>> y[1] = 7
```

- What is x[1]?

A: 7
B: 5
C: 6
D: **ERROR**
E: I don't know

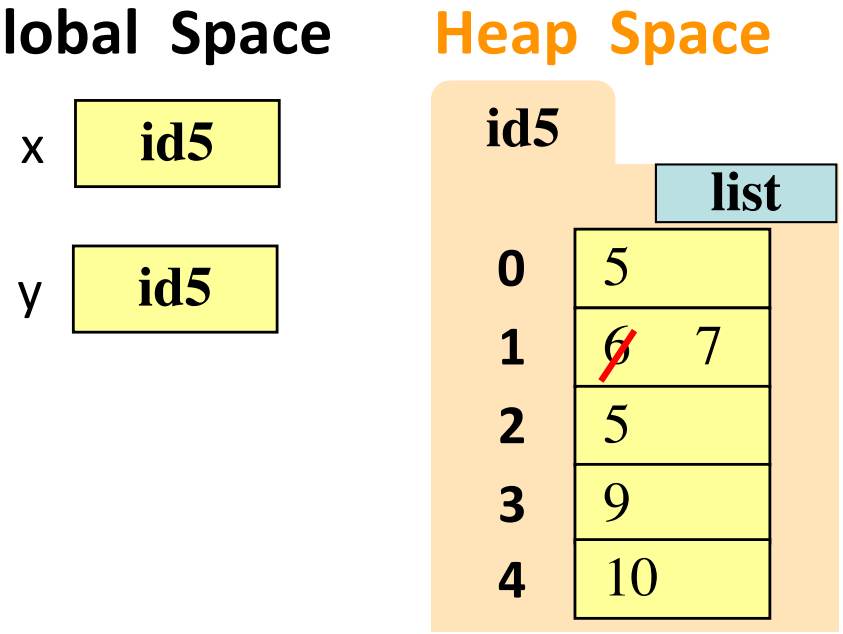
A4

- Execute the following:

```
>>> x = [5, 6, 5, 9, 10]
>>> y = x
>>> y[1] = 7
```

- What is x[1]?

A: 7 **CORRECT**
B: 5
C: 6
D: **ERROR**
E: I don't know



Things that Work for All Sequences

s = 'slithy'		x = [5, 6, 9, 6, 15, 5]
s.index('s') → 0	methods	x.index(5) → 0
s.count('t') → 1		x.count(6) → 2
len(s) → 6	built-in fns	len(x) → 6
s[4] → "h"	slicing	x[4] → 15
s[1:3] → "li"		x[1:3] → [6, 9]
s[3:] → "thy"		x[3:] → [6, 15, 5]
s[–2] → "h"		x[–2] → 15
s + ' toves' → "slithy toves"	operators	x + [1, 2] → [5, 6, 9, 6, 15, 5, 1, 2]
s * 2 → "slithyslithy"		x * 2 → [5, 6, 9, 6, 15, 5, 5, 6, 9, 6, 15, 5]
't' in s → True		15 in x → True

Lists and Strings Go Hand in Hand

text.split(<sep>): return a list of words in text (separated by <sep>, or whitespace by default)

<sep>.join(words): concatenate the items in the list of strings words, separated by <sep>.

```
>>> text = 'A sentence is just\n a list of words'
>>> words = text.split()
>>> words
['A', 'sentence', 'is', 'just', 'a', 'list', 'of', 'words']
>>> lines = text.split('\n')
>>> lines
['A sentence is just', ' a list of words']
>>> hyphenated = '-'.join(words)
>>> hyphenated
'A-sentence-is-just-a-list-of-words'
>>> hyphenated2 = '-'.join(lines[0].split()+lines[1].split())
>>> hyphenated2
'A-sentence-is-just-a-list-of-words'
```

Turns string into a list of words

Turns string into a list of lines

Combines elements with hyphens

Merges 2 lists, combines elements with hyphens

Returning multiple values

- Can use lists/tuples to **return** multiple values

```
INCHES_PER_FOOT = 12

def to_feet_and_inches(height_in_inches):
    feet = height_in_inches // INCHES_PER_FOOT
    inches = height_in_inches % INCHES_PER_FOOT
    return [feet, inches]

all_inches = 68  # Prof. Bracy wrote this
data= to_feet_and_inches(all_inches)
print(You are “+str(data[0])+” feet, “+str(data[1])+” inches.”)
```