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.
The objects weren’t affected.

**Sequences: Lists of Values**

- **String**
  - s = 'abc def'
  - 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 'def'
  - len(s) \rightarrow 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[4] causes an error
    - x[0:2] is [5, 6] (excludes 2 but includes 5)
    - x[3:] is [9, 15, 23]
  - len(x) \rightarrow 6, length of list

**Representing Lists**

- **Wrong:**
  - Global Space
    - x = [5, 7, 4, -2]

- **Correct:**
  - Global Space
    - x = [5, 7, 4, -2]
Lists Can Hold Any Type

- Lists can hold any type (numbers, strings, objects)
- Expression evaluates to value; value goes in list

Lists of Objects

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

List is mutable; strings are not

- Format:
  - `<var>[<index>]` = `<value>`
    - Reassign at index
    - Affects folder contents
    - Variable is unchanged

Q2: Swap List Values?

```python
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""
    temp = b[h]
    b[h] = b[k]
    b[k] = temp
```

x = [5, 6, 5, 9]
s = "Hello!"
x[1] = 8
s[0] = 'J'

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

$x = [5, 6, 5, 9]$

$y = x[1:3]$  

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
- $s$.count('y') → 1
- len($s$) → 6
- $s[4]$ → "h"
- $s[1:3]$ → "ly"
- $s[3]$ → "thy"
- $s[-2]$ → "h"
- $s + 'toves'"$ → "slithy toves"
- $s * 2$ → "slithy"  

$x + [1, 2] → [5, 6, 9, 6, 15, 5, 1, 2]$

$'t' in s$ → True

15 in $x$ → True
Lists and Strings Go Hand in Hand

- `text.split(sep)` returns a list of words in text (separated by `sep`, or whitespace by default)
- `sep.join(words)` concatenates the items in the list of strings `words`, separated by `sep`.

```
>>> text = 'A sentence is just
     a list of words'
>>> words = text.split()
>>> words
['A', 'sentence', 'is', 'just', 'a', 'list', 'of', 'words']
>>> lines = text.split('
')
>>> 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'
```

Returning multiple values

- Can use lists/tuples to return multiple values

```python
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 {} feet, {} inches.'.format(str(data[0]), str(data[1])))
```

39 40