



<http://www.cs.cornell.edu/courses/cs1110/2021sp>

## CS 1110 Prelim 1 Practice/Review Session

## Announcements

- A3 due Sun Mar 28
- Prelim 1 Tues Mar 30 at 6:30pm in-person (university-scheduled)
  - Check CMS for your exam info if you requested alternate time/format
  - In-person: Bring pens/pencils/erasers (bring several). Bring a watch or even an actual clock if you have one. No smart watches/phones! You may not be able to see the wall clock in Barton from your seat. Bring Cornell ID.
  - Online: Your proctor will contact you about a mock exam. **You *must* do the mock exam** to be allowed to write the actual exam.
- Read Prelim 1 Study Guide. *Note spring different from fall.*
- Tues Mar 30 lecture and lab time → office hours
- Wedn Mar 31 no labs (so no new lab exercises next week)

## Exam Topics

- String slicing functions
- Call frames and the call stack
- Functions on mutable objects
- Testing and debugging
- Conditionals
- Lists and simple iteration

Dictionary *not* on Prelim 1

Prelim 1 Review

Today:

- *Start with [lists and iteration](#)—not in posted old review slides*
- *[Testing and debugging](#)*
- *Other topics if time allows*

3

## Lists, Iteration, Strings

def count\_non\_space\_chars(myList):

"""Returns: number of non-space characters in the strings in myList.

Example: count\_non\_space\_chars(['U', 'r', ', ', 'gr8']) returns 5

Precondition: myList is a list of strings. Each string in myList can contain only spaces, letters, digits."""

You *should know* the methods that we actually have used in assignments and labs. We will give you the less-frequently used methods on the exam.

4

def count\_non\_space\_chars(myList):

"""Returns: number of non-space characters in the strings in myList.

Example: count\_non\_space\_chars(['U', 'r', ', ', 'gr8']) returns 5

Precondition: myList is a list of strings. Each string in myList can contain only spaces, letters, digits."""

count = 0

for s in myList:

numSp = s.count(' ')

numNonSp = len(s) - numSp

count = count + numNonSp

return count

### **Remember:**

- Be goal oriented—start with return statement (if it is needed). Work your way back up. Be flexible.
- Name a variable for any value you need but don't know yet
- For-loop
- Accumulation pattern
- How to call string methods

7

## Lists, Iteration, Types

def inflate(myList, p\_percent):

"""Inflate each number in myList by p\_percent while maintaining the type (int or float). For any int in myList, round down the inflation.

Precondition: myList is a list of positive numbers (int and/or float).

Precondition: p\_percent is a positive number (int or float)."""

An example:

```
>>> aList = [100, 100.0, 1, 1.0]
>>> p = 1.6
>>> inflate(aList, p)
>>> aList
[101, 101.6, 1, 1.016]
```

Prelim 1 Review

8

```
def inflate(myList, p_percent):
    """Inflate each number in myList by p_percent while maintaining the
    type (int or float). For any int in myList, round down the inflation.
    Precondition: myList is a list of positive numbers (int and/or float).
    Precondition: p_percent is a positive number (int or float)."""
```

9

```
def inflate(myList, p_percent):
    """Inflate each number in myList by p_percent while maintaining the
    type (int or float). For any int in myList, round down the inflation.
    Precondition: myList is a list of positive numbers (int and/or float).
    Precondition: p_percent is a positive number (int or float)."""
    p_frac= p_percent/100
    for k in range(len(myList)):
        delta= myList[k]*p_frac
        if type(myList[k])==int:
            delta= int(delta)
        myList[k] += delta
```

**Remember:**

- Give yourself an example if question doesn't provide one
- Using for-loop on list: do you need to modify list? If so you need the indices—use `range`
- List syntax
- How to work with types (ops, checking, casting)
- In general, read specs again after finishing code. Did you really solve problem asked?

10

## Constructing test cases

```
def before_space(s):
    """Returns: the substring before the first space character in string s.
    Precondition: string s contains at least one space."""
```

Come up with at least three *distinct* test cases. Write the test input, expected output, and rationale.

11

## Constructing test cases

```
def before_space(s):
    """Returns: the substring before the first space character in string s.
    Precondition: string s contains at least one space."""
```

- First think about the pre-condition to see what we know about the string `s`
  - It has at least one space char → it can have more than one → adjacent? non-adjacent?
- No precondition on where the space char appears in `s` → can be anywhere → start? middle? end?
- With these ideas, can construct distinct test cases with rationales for each one

12

## Constructing test cases

```
def before_space(s):
    """Returns: the substring before the first space character in string s.
    Precondition: string s contains at least one space."""
```

Examples:

- " abc" → "" – single space char at the start
- "abc " → "abc" – single space char at the end
- "a bc" → "a" – single space char in the “middle” (not start or end)
- " abc" → "" – many space chars at the start
- "abc " → "abc" – many space chars at the end
- "ab c" → "ab" – many space chars in the middle
- "a b c" → "a" – many non-adjacent space chars

13

## What should I be testing?

**Common Cases:** typical usage

**Edge Cases:** live at the boundaries

- **Target location in list:** first, middle, last elements
- **Input size:** 0,1,2, many (length of lists, strings, etc.)
- **Input Orders:** e.g., max(big, small), max(small, big)...
- **Element values:** negative/positive, zero, odd/even
- **Element types:** int, float, str, etc.
- **Expected results:** negative, 0, 1, 2, many

*Not all categories/cases apply to all functions.  
Use your judgement!*

14

## Functions on Objects

- Class: Rect

- Constructor function: Rect(x,y,width,height)
- Remember constructor is just a function that gives us an object of that type and returns its identifier

Attribute	Description
x	float, x coord of lower left corner
y	float, y coord of lower left corner
width	float, > 0, width of rectangle
height	float, > 0, height of rectangle

Prelim 1 Review

15

def move(r, xc, yc):

"""Set the attributes of Rect `r` such that its center lies on the x- and y-coordinates `xc` and `yc`, respectively.

Precondition: r is a Rect object.

Precondition: xc, yc are each a float."""

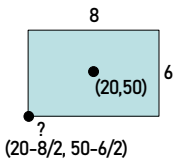
16

def move(r, xc, yc):

"""Set the attributes of Rect `r` such that its center lies on the x- and y-coordinates `xc` and `yc`, respectively.

Precondition: r is a Rect object.

Precondition: xc, yc are each a float."""



17

def move(r, xc, yc):

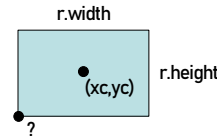
"""Set the attributes of Rect `r` such that its center lies on the x- and y-coordinates `xc` and `yc`, respectively.

Precondition: r is a Rect object.

Precondition: xc, yc are each a float.

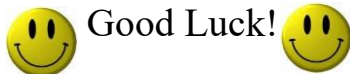
$$r.x = xc - r.width/2$$

$$r.y = yc - r.height/2$$



**Remember:**

- Draw a diagram to help yourself think
- Label the diagram with example/known values. Then generalize labels using parameter and attribute names.
- Important problem solving step! First use example values to understand the problem and figure out relationships among knowns and unknowns.
- Dot-notation for accessing attributes of an object



Prelim 1 Review

21