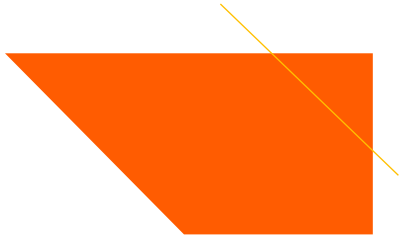
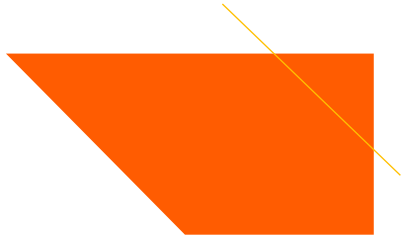
Introduction to Problem Solving in Python

COSI 10A



Class objectives

- Introduction to Lists (Section 7.1)



Random

Question 3

```
$ python3 dice_sum.py
Desired dice sum: 5
6 + 5 = 11
5 + 2 = 7
4 + 3 = 7
6 + 6 = 12
3 + 4 = 7
4 + 2 = 6
5 + 3 = 8
4 + 2 = 6
5 + 5 = 10
1 + 4 = 5
$
```

Question 3

```
def dice_sum():
    desired_sum = int(input("Desired dice sum: "))
    die1 = 0
    die2 = 0
    while die1 + die2 != desired_sum:
        die1 = random.randint(1, 6)
        die2 = random.randint(1, 6)
        print(die1, "+", die2, "=", (die1 + die2))
```

```
$ python3 dice_sum.py
Desired dice sum: 5
6 + 5 = 11
5 + 2 = 7
4 + 3 = 7
6 + 6 = 12
3 + 4 = 7
4 + 2 = 6
5 + 3 = 8
4 + 2 = 6
5 + 5 = 10
1 + 4 = 5
$
```

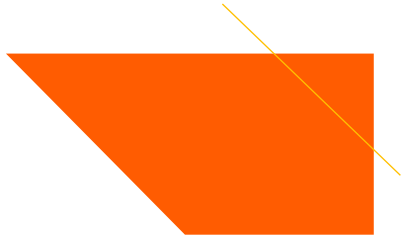
Question 4

- Write a function called `print_factor` that accepts an integer as its parameter and prints the factors of that number, separated by the word “and”

Question 4

- Write a function called `print_factor` that accepts an integer as its parameter and prints the factors of that number, separated by the word “and”

```
def print_factors(n):
    # print all factors other than n, if any, followed by " and "
    for i in range(1, n):
        if n % i == 0:
            print(i, "and ", end="")
    # print last factor, n itself (fencepost)
    print(n)
```



Lists

Can we solve this problem?

- Consider the following program (input underlined):

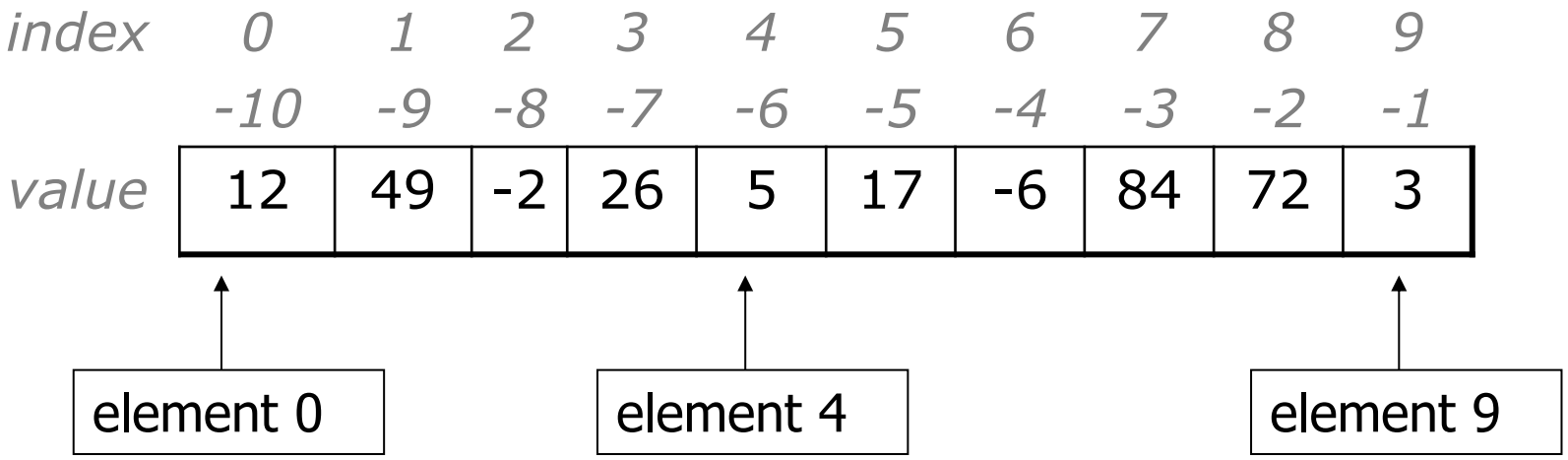
```
How many days' temperatures? 7
Day 1's high temp: 45
Day 2's high temp: 44
Day 3's high temp: 39
Day 4's high temp: 48
Day 5's high temp: 37
Day 6's high temp: 46
Day 7's high temp: 53
Average temp = 44.6
4 days were above average
```

Why the problem is hard?

- We need each input value twice:
  - To compute the average (a cumulative sum)
  - To count how many were above average
- We could read each value into a variable... but we:
  - Don't know how many days are needed until the program runs
  - Don't know how many variables to declare
- We need a way to declare many variables in one step

Lists

- **A list** is an object that stores many values
  - **Element**: One value in a list
  - **Index**: A 0-based integer to access an element from a list.



List initialization

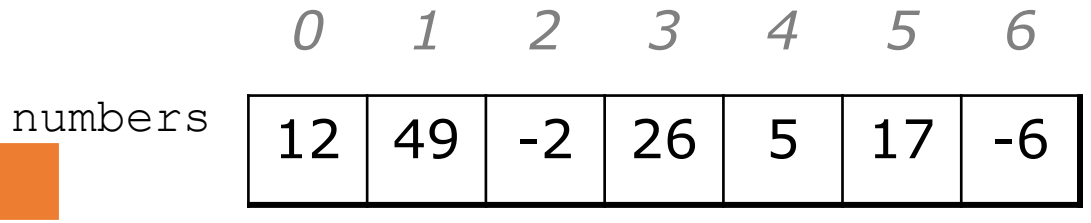
Syntax:

```
name = [value, value, … value]
```

Useful when you know what the list's elements will be

Example:

```
numbers = [12, 49, -2, 26, 5, 17, -6]
```

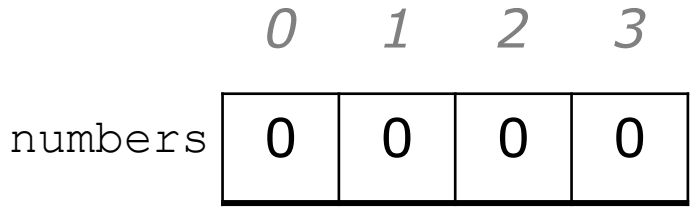


Syntax:

```
name = [value] * count
```

Example:

```
numbers = [0] * 4
```



Accessing elements

Syntax (Access)

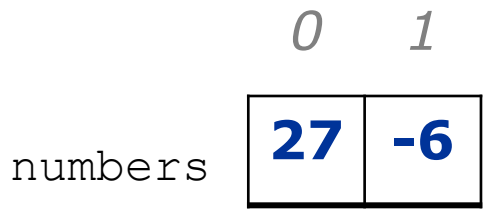
```
name[index]
```

Syntax (Modify)

```
name[index] = value
```

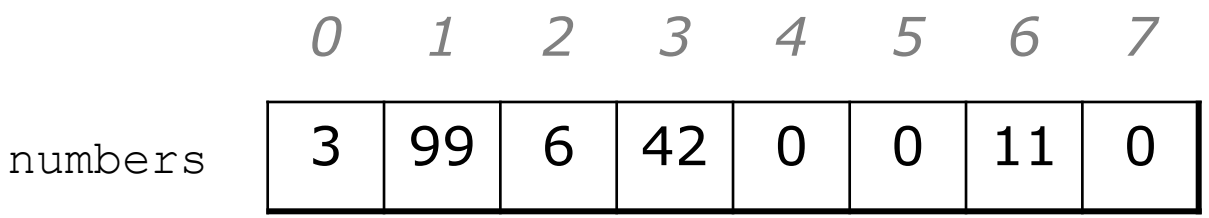
Example:

```
numbers = [0] * 2
numbers[0] = 27
numbers[1] = -6

print(numbers[0])

if (numbers[1] < 0):
   print("Element 1 is negative.")
```



Accessing elements

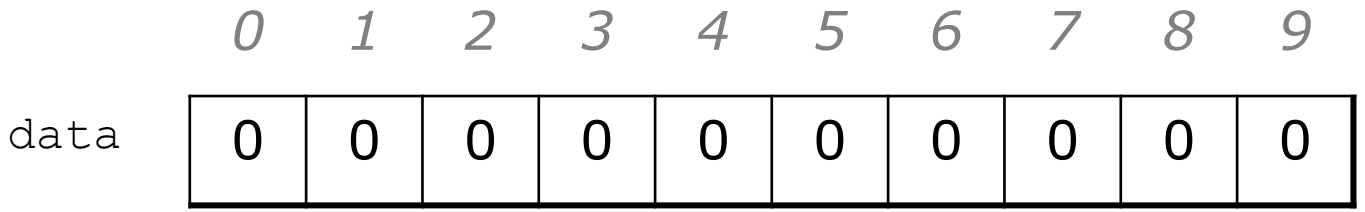
```
numbers = [0] * 8
numbers[0] = 3
numbers[1] = 99
numbers[2] = 6

x = numbers[0]
numbers[x] = 42
numbers[numbers[2]] = 11 # use numbers[2] as index
```



Out-of-bounds

- Legal indexes to use are between **– list's length** and the **list's length - 1**.
  - Reading or writing any index outside this range will cause an `IndexError: list assignment index out of range`
- Example:

```
data = [0] * 10
print(data[0])          # okay
print(data[9])          # okay
print(data[-20])        # error
print(data[10])         # error
```



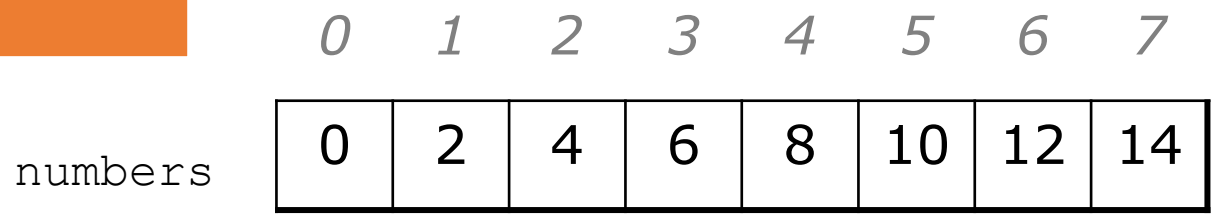
Lists and `for` loops

- It is common to use `for` loops to access list elements

```
for i in range(0, 8):
    print(str(numbers[i]) + " ", end='')
print()   # output: 0 4 11 0 44 0 0 2
```

- Sometimes we assign each element a value in a loop

```
for i in range(0, 8):
        numbers[i] = 2 * i
```



len()

- Use `len()` to find the number of elements in a list

```
for i in range(0, len(numbers)):
    print(numbers[i] + " ", end='') # output: 0 2 4 6 8 10 12 14
```

Lists and `for` loops

- You can also loop directly over lists, just as with strings

```
list = [1, 3, 6, 23, 43, 12]
for number in list:
          print(str(number + " ", end='')
print()  # output: 1 3 6 23 43 12
```

Weather question

- Consider the following program (input underlined):

```
How many days' temperatures? 7
Day 1's high temp: 45
Day 2's high temp: 44
Day 3's high temp: 39
Day 4's high temp: 48
Day 5's high temp: 37
Day 6's high temp: 46
Day 7's high temp: 53
Average temp = 44.6
4 days were above average
```

Weather answer

```
# Reads temperatures from the user, computes average and # days above average
def weather():
    days = int(input("How many days' temperatures? "))

    temps = [0] * days      # list to store days' temperatures
    sum = 0

    for i in range(0, days):      # read/store each day's temperature
        temps[i] = int(input(("Day " + str(i + 1) + "'s high temp: ")))
        sum += temps[i]
    average = sum / days

    count = 0         # see if each day is above average
    for i in range(0, days):
        if (temps[i] > average):
            count += 1
    # report results
    print("Average temp = " + str(average))
    print(str(count) + " days above average")
```

Weather question 2

- Modify the weather program to print the following output:

```
Type in a temperature or "done" to finish
Day 1's high temp: 45
Day 2's high temp: 44
Day 3's high temp: 39
Day 4's high temp: 48
Day 5's high temp: 37
Day 6's high temp: 46
Day 7's high temp: 53
Day 7's high temp: done
Average temp = 44.6
4 days were above average.
```

List declaration

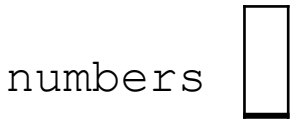
Syntax:

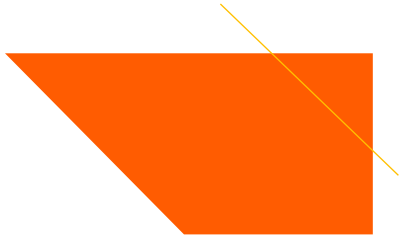
```
name = []
```

Creates an empty list

Example:

```
numbers = []
```





List functions

Function	Description
`append(x)`	Add an item to the end of the list. Equivalent to `a[len(a):] = [x]`.
`extend(L)`	Extend the list by appending all the items in the given list. Equivalent to `a[len(a):] = L`
`insert(i, x)`	Inserts an item at a given position. i is the index of the element before which to insert, so `a.insert(0, x)` inserts at the front of the list.
`remove(x)`	Removes the first item from the list whose value is *x*. Errs if there is no such item.
`pop(i)`	Removes the item at the given position in the list and returns it. `a.pop()` removes and returns the last item in the list.
`clear()`	Remove all items from the list.
`index(x)`	Returns the index in the list of the first item whose value is *x*. Errs if there is no such item.
`count(x)`	Returns the number of times *x* appears in the list.
`sort()`	Sort the items of the list
`reverse()`	Reverses the elements of the list
`copy()`	Return a copy of the list.

Weather question 2

- Modify the weather program to print the following output:

```
Type in a temperature or "done" to finish
Day 1's high temp: 45
Day 2's high temp: 44
Day 3's high temp: 39
Day 4's high temp: 48
Day 5's high temp: 37
Day 6's high temp: 46
Day 7's high temp: 53
Day 7's high temp: done
Average temp = 44.6
4 days were above average.
```