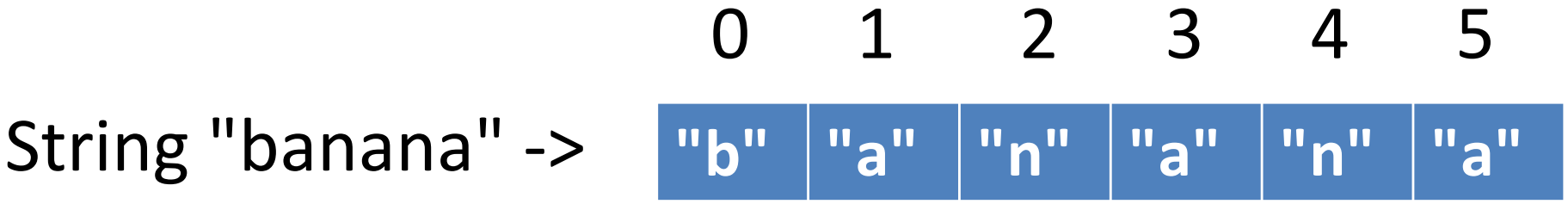
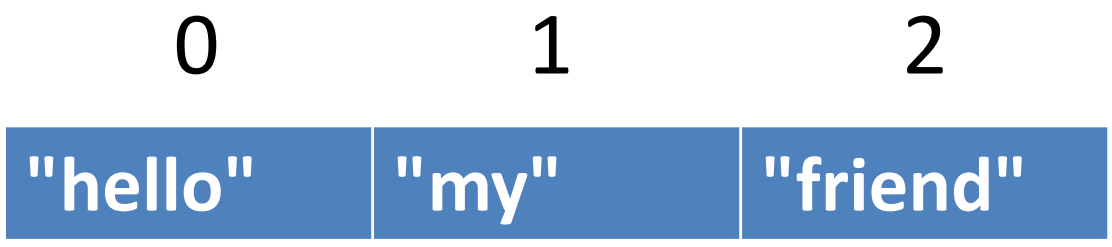
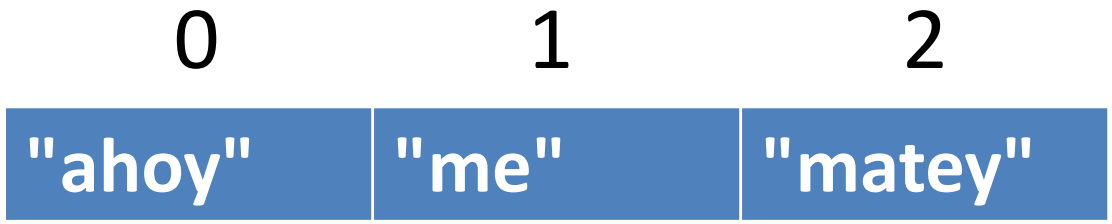
Lists I

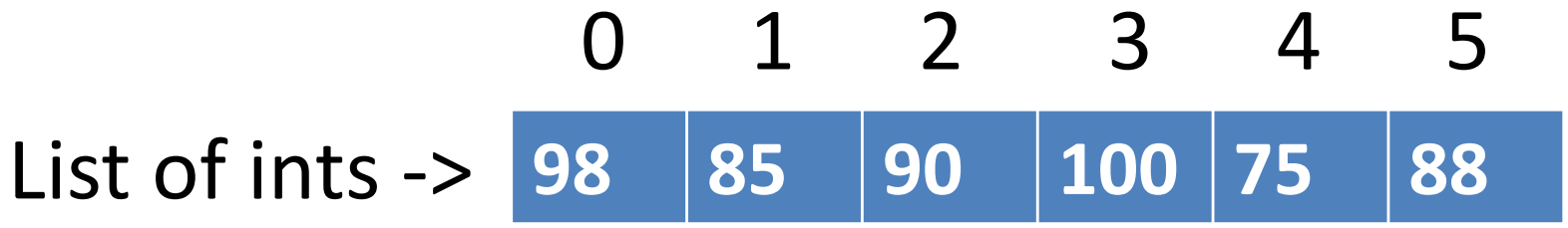
What is a list?

- Lists are strings on steroids.
- A string stores an ordered sequence of single characters.
- A list stores an ordered sequence of *any* data type.


0 1 2 3 4 5
String "banana" ->
"b" "a" "n" "a" "n" "a"


0 1 2
"hello" "my" "friend"
<- Lists of strings


0 1 2
"ahoy" "me" "matey"


0 1 2 3 4 5
List of ints ->
98 85 90 100 75 88

Why use lists?

- Lists exist so programmers can store multiple related variables together.
- Useful when we don't know ahead of time how many items we are going to store.
  - Lists solve this problem because a single list can hold from zero to practically any number of items in it.

Basic list operations

- Lists are **created** using square brackets around items separated by commas.

```
mylist = [1, 2, 3]
numbers = [-9.1, 4.77, 3.14]
fred = ["happy", "fun", "joy"]
```

- Lists are accessed using indices/positions just like strings.
- Most (but not all) string functions also exist for lists.

Strings	Lists
string_var = "abc123"	list_var = [item1, item2, ...]
string_var = ""	list_var = [ ]
len("abc123") len(string_var)	len([3, 5, 7, 9]) len(list_var)
string_var[p] string_var[p:q]	list_var[p] list_var[p:q]
str3 = str1 + str2 str3 = "abc" + "def"	list3 = list1 + list2 list3 = [1, 2, 3] + [4, 5, 6]
"i" in "team" -> False	7 in [2, 4, 6, 8] -> False

One important difference

- Strings are ***immutable***
  - You can't change a string without making a copy of it.

```
s = "abc"
s[0] = "A"          # illegal!
s = "A" + s[1:]     # legal
```

One important difference

- Lists are ***mutable***
  - Can be changed "in-place" (without explicit copying)

```
L = [2, 4, 6, 8, 10]
L[0] = 15              # legal
L.append(26)           # legal
```

Compare mutable and immutable

- How can we switch the first and last letter in a string?

- How can we switch the first and last items in a list?

Three common ways to make a list

- Make a list that already has stuff in it:

```
lst = [4, 7, 3, 8]
```

- Make a list of a certain length that has the same element in all positions:

```
lst = [0] * 4   #makes the list [0,0,0,0]
```

  Common when you need a list of a certain length ahead of time.

- Make an empty list:

```
lst = []
```

  Common when you're going to put things in the list coming from the user or a file.

Simple list problems

- How would we write a function to convert a number from 1-12 into the corresponding month of the year as a string?

```
def getmonth(month):
```

Final exam

- Currently scheduled for Sat, May 5, 8:30am
- Tentative (but pretty sure) I'll offer two alternatives:
  - Tue, May 1, 1pm (probably)
  - Wed, May 2, 5:30pm (almost certainly)

Simple list problems

What does this code do?

```
lst = [2] * 3
lst2 = [4] * 2
lst3 = lst + lst2
for x in range(0, len(lst3), 2):
    lst3[x] = -1
```

```
while True:
    num = int(input("Enter number: "))
    if num == -1:
        break
    print("Your number is", num)
```

```
lst = []
while True:
    num = int(input("Enter number: "))
    if num == -1:
        break
    lst.append(num)
```

- **After all the numbers are read in**:
  - **Easy:** write a loop to print out the sum of all the numbers in the list. (This loop should be separate from the file-reading loop.)
  - write a loop to find the largest and smallest numbers in the list.
  - **Harder:** write a loop to print out use a for loop to print out sums of adjacent pairs of numbers in the list (don't use sliding window; use indices)
    - Hint: You don’t need the sliding window technique; instead, use math with list indices.

- Make a text file with some integers in it, one per line.
- Write a program to read all the numbers and store them in a list.
- **After all the numbers are read in**:
  - **Easy:** write a loop to print out the sum of all the numbers in the list. (This loop should be separate from the file-reading loop.)
  - write a loop to find the largest and smallest numbers in the list.
  - **Harder:** write a loop to print out use a for loop to print out sums of adjacent pairs of numbers in the list (don't use sliding window; use indices)
    - Hint: You don’t need the sliding window technique; instead, use math with list indices.