

CS 114 Tutorial 4

Oct. 3 2025

Goals for this Week:

- ❑ Look at assignment 02 (due next Friday)
- ❑ Be familiar with lists and strings
- ❑ Understand indexing and len()
- ❑ Look out for off by 1 errors
- ❑ Be comfortable with the concept of references
- ❑ Know the terms slicing, mutation, and sequences

A Quick Overview

Strings:

- A sequence of characters
- Represented using “”
- `my_string: str`
- Stores as a value

Lists:

- A sequence of anything
- Represented using `[]`
- `my_list: list[typing.Any]`
- Stores as a reference

A Quick Overview

Indexing

- Represents the position in a sequence, use `[]`
- Indices start at zero, this is different from `len()` which starts at 1

Operations:

- Add to a list with `.append()` or `.insert(index, value)`. Remove using `.pop()`
- You can also mutate a list with indices

Slicing:

- Creates a copy of the list
- `[from: to: step]` part of a list, `[:]` whole list, `[: :-1]` reverses the list

Create a function `switch_ends(lst)` that takes a list and switches the first and last values. Use at least one of the following: `pop`, `append`, slicing, `insert`. Your function should mutate `lst` and return `None`.

[1, 4, 6, 7] becomes [7, 4, 6, 1]

On paper, use indices to print the requested value in the list.

```
lst = [[['sea', 'why', 'are', 'bee'], ['what', 'where', 'when']],  
       ['hello', 'night', 'welcome'], ['good', 'bad', ['no']]]
```

How would you access the `n` in `'no'` and the `e` in `'are'`?

Go to vevox.com

Sign in using the session ID: 142-382-134



Annotations

```
x = [[0.3, 0.3, 0.7], [-1.8, -2.0, -0.3]]  
  
def lists(x):  
    return(x[0])
```

How would you annotate lists?

- A. $(x: [\text{float}]) \rightarrow \text{float}$
- B. $(x: \text{list}[\text{list}[\text{float}]]) \rightarrow \text{list}[\text{float}]$
- C. $(x: \text{list}[\text{list}]) \rightarrow [\text{float}]$
- D. $(x: \text{list}[\text{list}[\text{float}]]) \rightarrow \text{float}$
- E. $(x: \text{list}[\text{float}]) \rightarrow \text{float}$



Lists

```
x = ["a", "b", "c", "d", "e", "f"]  
print(len(x))
```

What length will be printed?

A. 6

B. 5

C. 4

D. 7



Lists

```
x = ["a", "b", "c", "d", "e", "f"]  
print(x[3])
```

What letter will be printed?

- A. "e" B. "b" C. "c" D. "d"



Lists

```
x = ["a", "b", "c", "d", "e", "f"]  
print(x[-3])
```

What letter will be printed?

- A. "e" B. "b" C. "c" D. "d"



Mutation vs Non-Mutation

```
def function(lst: list[str]) -> list[str]:  
    new_lst = []  
    for word in lst:  
        if len(word) <= 3:  
            new_lst.append(word)  
    lst.pop()  
    return(new_lst)
```

Did this function mutate lst?

A. yes

B. no

C. i don't know

Write a function `filter_by_range(lst, limits)` that takes a list of ints and a list containing two integers that describe the inclusive bounds of a range. The function will return a new list with only values from `lst` within the range of `limits`.

`assert filter_by_range([1,2,3,5], [2,4]) == [2,3], "between 2 and 4"`

`assert filter_by_range([-2,0,8,2,5], [-1, 6]) == [0, 2, 5], "between -1 and 6"`