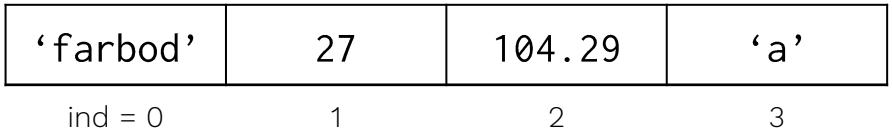
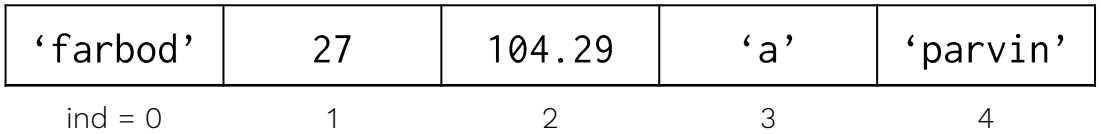
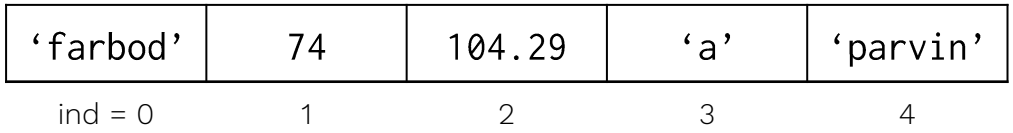
# Lists

- Defining: `my_list = ['farbod', 27, 100.29, 'a']`
- Length: `len(my_list)` ☞ returns `4`
- Indexing and slicing: similar to strings
  - 

- Adding items: `my_list.append('parvin')`:
  - 

- Modifying: `my_list[1]=74` ☞ 

# Lists

- Concatenating: `[1, 2, 78, 10] + [41, 35]` 👉 `[1, 2, 78, 10, 41, 35]`
- Iterating through loop:
  - ```
    for item in my_list:
        print(item)
    print('this line is out of the loop')
    ```
- Deleting by position: `del my_list[-1]`
- Deleting by value: `my_list.remove['farbod']`
- Sorting: `my_list.sort()` ***OR*** `sorted(my_list)`
- Reverse sorting: `users.sort(reverse=True)` OR `sorted(users, reverse=True)`

*Sort the original list* *returns a copy*
    ```

# Lists

- Checking if a value is in the list: `my_list = [1, 1, 2, 3]`
  - `2 in my_list` ☞ returns `True`
- Counting occurrences: `my_list.count(1)` ☞ returns `2`
- 2 major uses of `range()`:
  - "for" loops:
```
for number in range(1001):
    print(number)
# printing the number 0 to 1000
```
  - A list of numbers: `numbers = list(range(1, 1001))`
- Finding the minimum, maximum and sum of all values:

| `lowest=min(numbers)` | `highest= max(numbers)` | `total=sum(numbers)` |
|---|---|---|
| `0` | `1000` | `0+1+2+…+1000` |

# Tuples

- Defining: `dimensions = (1920, 1080)` OR `dimensions = 1920, 1080`
- accessing: `dimensions[0]` 👉 returns `1920`
- You can't change the values in a tuple once it's defined (immutable) ✋
- Looping through a tuple:
  - ```
    for dimension in dimensions:
        print(dimension)
    ```
- overwriting a tuple: `dimensions = (800, 600)`
- Counting occurrences: `dimensions.count(800)` 👉 returns `1`
- List of tuples: `[(0, ‘a’), (1, ‘b’), (2, ‘c’), (3, ‘d’)]`
    ```

# .join() and .split() methods

- In Python, a string can be split on a delimiter. `str.split()` method returns a list of strings
  - `a = "this is a string"`
  - `a.split(" ")` (OR `a.split()`) 👉 returns `['this', 'is', 'a', 'string']`
  - `a.split('s')` 👉 returns `['thi', ' i', ' a ', 'tring']`

- Joining a list of strings into a single string:
  - `a = ['this', 'is', 'a', 'string']`
  - `' '.join(a)` 👉 returns `"this is a string"`
  - `'-'.join(a)` 👉 returns `"this-is-a-string"`

# enumerate

- A lot of times when dealing with lists, we get a need to keep a count of iterations. *enumerate() method adds a counter to a list* and returns it in a form of enumerate object. This enumerate object can then be used directly in *for loops* or be converted into *a list of tuples* using list().
- Syntax: `enumerate(iterable, start=0)`

- `lst_1 = [‘a’, ‘b’, ‘c’]`
  - `list(enumerate(lst_1))` ☞ returns `[(0, ‘a’), (1, ‘b’), (2, ‘c’)]`
  - ```
    for count, elem in enumerate(lst_1):
        print(count, elem)
    ```
    ```
    0 ‘a’
    1 ‘b’
    2 ‘c’
    ```
    ```

# List copying

## Reference copying

- `a = [1, 2, 3]`
- `b = a`
- `b.append(4)`
- `print(a)` ☞ `[1, 2, 3, 4]`

## Value copying

- `a = [1, 2, 3]`
- `b = a[:]`
- `b.append(4)`
- `print(a)` ☞ `[1, 2, 3]`