Week 9

- List – remove()
  - Will remove the first matching element.

```
q = ['a', 'b', 'c', 'd']
q.remove('b')
print(q)          # ['a', 'c', 'd']
```

1. Lecture Q&A – how to remove more than first matching element:

2. List Comprehensions

List – remove()

- Lecture Q&A: remove() - will remove the first matching element.
- How to remove more than first matching element. In a for loop:

```
q = ['a', 'b', 'c', 'a', 'b', 'c']
for item in q:
    if item == 'b’:
        q.remove(item)
```

- Alternatively, use list comprehension to create a new list:

```
q = ['a', 'b', 'c', 'a', 'b', 'c']
q = [item for item in q if item != 'b']
```

List Comprehensions

- List comprehensions provide a concise way to create lists. Syntax:

```
[expression for variable_name in iterable]
```

- Each element of iterable (e.g., list, tuple, range etc) is taken out as variable_name and evaluated by expression to create a new list.

```
squares = [x**2 for x in range(10)]
# [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

- The list comprehension consists of:
  - *brackets* `[]`
  - *expression* - `x**2`
  - followed by a *for* clause - `for x in range(10)`

List Comprehension vs for loop

- E.g., to make a new list where each element is the result of some operation:

```
squares = []
for x in range(10):
    squares.append(x**2)
print(squares)  # [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

- More concise list comprehension version:

```
squares = [x**2 for x in range(10)]
print(squares)    # [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

List Comprehensions with if

- We can add a **conditional statement** on the **iterable**.
- E.g., create a sequence of elements that satisfy a certain condition. Syntax:

```
[expression for variable_name in iterable if condition]
```

- Example:

```
numbers = [1, 2, 3, 4, 5]
squares = [number**2 for number in numbers if number > 2]
```

- Here the list comprehension consists of:
  - *brackets* `[]`
  - *expression* - `number**2`
  - *for* clause - `for number in numbers`
  - *conditional statement* - `if number > 2`

List Comprehensions with if

- Another list comprehension using an if clause:

```
even_squares = [x**2 for x in range(10) if x % 2 == 0]
print(even_squares)          #[0, 4, 16, 36, 64]
```

- The above code written without the use of a list comprehension

```
even_squares = []
for x in range(10):
    if x % 2 == 0:
      even_squares.append(x**2)
print(even_squares)          # [0, 4, 16, 36, 64]
```

Summary

- List comprehensions provide a concise way to create lists. Syntax:

```
[expression for variable_name in iterable if condition]
```

- The condition is optional.
- Result - a new list resulting from evaluating the *expression* in the context of the *for* and any *if* clauses.

```
a =[x*5 for x in range(5)]  #[0, 5, 10, 15, 20]

b = [x for x in range(5)if x%2 == 0]  #[0, 2, 4]
```