Introduction to List Comprehension

- At the end of this class, students will be able to-
  - Learn that List comprehensions provide a concise way to create lists.
  - How to rewrite loops functional programming with the help of list comprehension.

list comprehension

- list comprehension provides an alternate mechanism to functional programming constructs map and filter.
- Applies the expression to each element in the list
- You can have 0 or more for or if statements
- If the expression evaluates to a tuple it must be in parenthesis

The general form of list comprehension is

- [<expr> for <variable> in <iterable>]

```
things = [2, 5, 9]

yourlist = [value * 2 for value in things]

print(yourlist)
```

Output:
[4, 10, 18]

Using if with List Comprehension

```
number_list = [ x for x in range(20) if x % 2 == 0]
print(number_list)
```

Output:
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]

Nested IF with List Comprehension

```
num_list = [y for y in range(100) if y % 2 == 0 if y % 5 == 0]
print(num_list)
```

Output:
[0, 10, 20, 30, 40, 50, 60, 70, 80, 90]

if...else With List Comprehension

```
obj = ["Even" if i%2==0 else "Odd" for i in range(10)]
print(obj)
```

Output:

['Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd', 'Even', 'Odd']

Creating new list with other list as basis

```
primes = [2, 3, 5, 7]
doubleprimes = [2*x for x in primes]
```

The same as

```
doubleprimes = list()
for x in primes:
    doubleprimes.append(2*x)
```

Any expression (the part before the for loop) can be used.

Example: A tab separated line of numbers are read from a file, convert the numbers from strings to floats.

```
for line in datafile:
    numbers = [float(no) for no in line.split()]
    # Do something with the list of numbers
```

Filtering with comprehension – using if

```
odd = [no for no in range(20) if no % 2 == 1]
numbers = [1, 3, -5, 7, -9, 2, -3, -1]
positives = [no for no in numbers if no > 0]
```

Nested for loops in comprehension

Example: Creating all combinations in tuples of two numbers, where no number is repeated in a combination.

```
combi = [(x, y) for x in range(10) for y in range(10) if x != y]
```

Flatten a list of lists (matrix) into a simple list

```
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
flatList = [no for row in matrix for no in row]
```

A list does not have to form the basis of the comprehension – any iterable will do, like sets or dicts.

Summary

- List comprehensions provide you a way of writing for loops more concisely.
- They can be useful when you want to create new lists from existing lists or iterables.