List Comprehension

List comprehension is an elegant and compact way to make new lists from existing iterables

```
new_list = [expression for member in iterable]
```

The expression is executed for each member in the iterable, and the result will be an item in the new list.

expression: a method, a built-in function, custom function, or any other valid expression that returns a value. That value is an item of the new list

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

fruits_upp = [x.upper() for x in fruits]
print(fruits_upp)
['APPLE', 'BANANA', 'CHERRY', 'KIWI', 'MANGO']


squares = [i*i for i in range(1,11)]
print(squares)
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

List Comprehension


```
new_list = [expression for member in iterable if condition]
```

```
fruits = ["apple", "banana", "cherry", "kiwi", "mango"]

newlist = [x for x in fruits if "a" in x]
print(newlist)
['apple', 'banana', 'mango']
```

```
Dfruits ={"apple":9,"banana":10,"cherry":3,"kiwi":4,"mango":5}

L=[k for k,v in Dfruits.items() if v > 5]
print(L)
['apple', 'banana']
```

Dictionary Comprehension

Like list comprehensions, Python allows dictionary comprehensions.
We can create dictionaries using simple expressions.
A dictionary comprehension takes the form

```
output_dict = {key:value for item in iterable}
```

```
words = ['data', 'science', 'machine', 'learning']
D={i:len(i) for i in words}
print(D)
{'data': 4, 'science': 7, 'machine': 7, 'learning': 8}
```

```
output_dict = {key:value for item in iterable if condition}
```

```
words = ['data', 'science', 'machine', 'learning']
D={i:len(i) for i in words if len(i) > 5}
print(D)
{'science': 7, 'machine': 7, 'learning': 8}
```

Using zip in dictionary comprehension

Use zip to make a dictionary out of multiple sequences

```
words = ['data', 'science', 'machine', 'learning']
values = [5, 3, 1, 8]

dict_a = {i:j for i, j in zip(words, values)}
print(dict_a)
```

same as

```
dict_a=dict(zip(words, values))
```

```
words = ['data', 'science', 'machine', 'learning']
values = [5, 3, 1, 8]
dict_b = {i:j for i, j in zip(words, values) if j > 4}
print(dict_b)
```

dictionary comprehension

Use items() to make a dictionary out of an exiting dictionary

```
D={'DATA': 25, 'SCIENCE': 9, 'MACHINE': 1, 'LEARNING': 64}

D1 = {i.lower():j*2 for i, j in D.items()}
print(D1)
```

Create lists using Loops

You can use a for loop to create a list of elements in three steps:

1. Initialize an empty list.
2. Loop over an iterable or range of elements.
3. Append (or extend) each element to the end of the list.

If you want to create a list containing the first ten perfect squares

```
squares = []

for i in range(1,11):
      squares.append(i*i)

print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Using list comprehension

```
squares = [i*i for i in range(1,11)]
print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Create lists using Loops

You can use a while loop to create a list of elements in three steps:

1. Initialize an empty list.
2. make a while loop
3. within the while loop append (or extend) each element to the end of the list.

If you want to create a list of 4 unique integer random numbers in range 1-10

```
import random
L=[]

while len(L)!=4:
      num=random.randint(1,10)
      if num not in L:
         L.append(num)
print(L)
```

Create dictionaries using for Loops

You can use a for loop to create a dictionary in three steps:

1. Initialize an empty dictionary.
2. Loop over an iterable or range of elements.
3. Add key-value pair with D[key]=value or D.update({key:value})

```
veggie=['spinach', 'broccoli', 'edamame', 'bell pepper', 'kale',
'cabbage', 'celery', 'asparagus', 'lettuce']

num=[30,15,25,11,3,4,1,5,6]

D={}
for k,v in zip(veggie,num):
    if 'a' in k and v < 10:
        D[k]=v # or D.update({k:v})

print(D)
{'kale': 3, 'cabbage': 4, 'asparagus': 5}
```

Use dictionary comprehensions

```
D={k:v for k,v in zip(veggie,num) if 'a' in k and v < 10}
print(D)
{'kale': 3, 'cabbage': 4, 'asparagus': 5}
```

Create dictionary using Loops

You can use a while loop to create a list of elements in three steps:

1. Initialize an empty dictionary.
2. make a while loop
3. within the while loop add key-value pair with D[key]=value or D.update({key:value})

If you want to create a dictionary where the keys are 4 integer random numbers in range 1-10, and the values are the corresponding squares

```
import random
D={}

while len(D)!=4:
      num=random.randint(1,10)
      D[num]=num**2

print(D)
```

Create strings using for Loops

You can use a for loop to create a string in three steps:

1. Initialize an empty string.
2. Loop over an iterable or range of elements.
3. Use concatenation to build up the string

```
num=[30,15,25,11,3,4,1,5,6]
```

Make a string whose elements are the numbers in the list

```
s1=''
for i in num:
    s1=s1+str(i) #concatenation
print(s1)
```

```
3015251134156
```

This is a summing loop

```
s1=0
for i in num:
    s1=s1+i  #addition
print(s1)
```

```
100
```

Example:

```
L=[ float(input("Enter a number: "))  for i in range(4) ]
print(L)
```

```
L=[]
for i in range(4):
    num=float(input("Enter a number: "))
    L.append(num)
```

Loops vs comprehensions - optional

- Comprehension is usually faster

```
import time
iterations = 100000000

start = time.time()

mylist = []
for i in range(iterations):
    mylist.append(i+1)

end = time.time()
print(end - start)

start = time.time()
mylist = [i+1 for i in range(iterations)]
end = time.time()
print(end - start)
```

- But pay attention to the size of the list because a list comprehension in Python works by loading the entire output list into memory! You can explore the module **tracemalloc** to trace memory allocations.