NTTF®
Securing your Future with your own Hands

DATA STRUCTURES USING PYTHON

2.2 LIST COMPREHENSION

- List comprehension is the concise way of creating list from the once that already exist
- It provide a shorter syntax to create new list from existing list and their values
- List comprehension is generally more compact and faster than normal function and loop for creating list
- Every list comprehension can be rewritten in for loop.but every for loop can’t be rewritten in the form of list comprehension

List Comprehension

- List comprehension is a complete substitution of for loop,lambda function as well as the functions map(),filter(),reduce()
- The number you passed to the range() function is actually the number of integers that you want to generate , starting from zero
- This means that range(10) will return [0,1,2,3,4,5,6,7,8,9]

List comprehension

Syntax:

Variable=[Expression for item in iterable]

Expression-operation or value

Item-variable

Iterable-source of data(list,tuple,range)

List comprehension

Example-

```
l=[i for i in range(10)]
Print(l)
```

Output

[0,1,2,3,4,5,6,7,8,9]

Normal List

Examples-square of a number without list comprehension

```
square=[]
for i in range(6):
        square.append(i**2)
Print(square)
```

Output

[0,1,4,9,16,25]

Using List comprehension

Examples-square of a number using list comprehension

```
sqr=[i**2 for i in range(6)]
Print(sqr)
```

Output

[0,1,4,9,16,25]

List comprehension with condition

Syntax:

Variable=[Expression for item in iterable if condition]

Expression-operation or value

Item-variable

Iterable-source of data(list,tuple,range)

Normal List with condition

Examples-find even number without list comprehension

```
even=[]
for i in range(10):
        if(i%2==0):
                even.append(i)
print(even)
```

Output

[0,2,4,6,8]

Using List comprehension with condition

Examples-find even number using list comprehension

```
even=[i for i in range(10) if  i%2==0]
Print(even)
```

Output

[0,2,4,6,8]

List comprehension with multiple If condition

Syntax:

Variable=[Expression for item in iterable If condition If condition]

Expression-operation or value

Item-variable

Iterable-source of data(list,tuple,range)

Normal list with multiple If condition

Example : multiplication of 6

```
multi=[]
For i in range(60):
        if(i%2==0):
            if(i%6==0):
                multi.append(i)
Print(multi)
```

Output

[0,6,12,18,24,30,36,42,48,54]

List Comprehension with multiple If condition

Example multiplication of 6

```
multi=[i for i in range(60) if(i%2==0) if(i%6==0)]
Print(multi)
```

Output

[0,6,12,18,24,30,36,42,48,54]

List Comprehension with slicing

What is slicing

Slicing is the extraction of a part of string,list,tuple.

Syntax:

List[start:stop:step size]

List Comprehension with slicing

Example: list slicing

```
list=[1,2,3,4,5,6,7,8]                    #last index 8 excluding
Print(list[-6:-1:2])
```

Output

[3,5,7]

List Comprehension with slicing

Example: tuple slicing

```
tup=(1,2,3,4,5,6,7)
Print(tup[-4:-1:2])
```

output

[4,6]

List Comprehension with slicing

Example: string slicing

```
name="bangalore"                    #last index excluding
Print(name[1:7:2])
```

Output

[a,g,l]

List Comprehension with slicing

- Multiplication of 2 using list slicing

```
a=[1,2,3,4,5]
result=[i*2 for i in a[:3]]
Print(result)
```

output

```
[2,4,6]
```

List Comprehension with slicing

```
a=[1,2,3,4,5,6]
Result=[i for i in a[-3:] if i%2==0]
Print(result)
```

Output

```
[4,6]
```

Advantages

- More time efficient and space efficient than loop
- Require a fewer lines of code
- Transforms iterative statement into a formula