Python Study Notes: Comprehensions, Lambda, Map, Filter & Reduce

Introduction to Comprehensions

Definition:
Comprehension Python ka ek concise aur readable tareeqa hai collections (list, dictionary, set) create karne ka. Ye traditional loops ka short form hai aur efficient aur clean code provide karta hai.

Syntax (General):

- List: `[expression for item in iterable if condition]`
- Dictionary: `{key: value for item in iterable if condition}`
- Set: `{expression for item in iterable if condition}`

—

1 List Comprehension

Definition: List comprehension ek short aur readable tareeqa hai list create karne ka using loops aur conditions.

Roman Urdu Explanation:
List comprehension me hum direct list generate karte hain, jaise traditional loop aur append karna, magar one line me. Agar condition lagani ho to `if` use karte hain.

Syntax:

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

Examples:

```
# 1. Numbers 0-9
nums = [x for x in range(10)]

# 2. Even numbers 0-9
evens = [x for x in range(10) if x % 2 == 0]

# 3. Squares of numbers 1-5
squares = [x**2 for x in range(1,6)]

# 4. Cube of even numbers
cubes = [x**3 for x in range(1,6) if x % 2 == 0]
```

```
12
13 # 5. Convert strings to uppercase
14 words = ["hello","world"]
15 upper_words = [w.upper() for w in words]
16
17 # 6. Length of each word
18 lengths = [len(w) for w in words]
19
20 # 7. Filter words with length > 4
21 long_words = [w for w in words if len(w) > 4]
22
23 # 8. Flatten nested list
24 nested = [[1,2],[3,4]]
25 flat = [num for sub in nested for num in sub]
26
27 # 9. Add 5 to each number
28 added = [x+5 for x in range(5)]
29
30 # 10. Conditional labeling even/odd
31 labels = ["Even" if x % 2 == 0 else "Odd" for x in range(5)]
```

—

2 Lambda Functions

Definition: Lambda function ek anonymous function hai jo ek hi line me define hoti hai aur small operations ke liye use hoti hai.

Roman Urdu Explanation:
Lambda function ka naam nahi hota. Ye sirf ek expression execute karta hai aur return karta hai.

Syntax:

```
1 lambda arguments: expression
```

Examples:

```
1 # 1. Add 10
2 add_10 = lambda x: x + 10
3
4 # 2. Multiply two numbers
5 mul = lambda x,y: x*y
6
7 # 3. Square
8 square = lambda x: x**2
9
10 # 4. Cube
11 cube = lambda x: x**3
12
13 # 5. Even check
14 is_even = lambda x: x % 2 == 0
15
16 # 6. Max of two
```

```python
max_num = lambda x,y: x if x>y else y

# 7. Absolute value
abs_val = lambda x: x if x>=0 else -x

# 8. Length of string
str_len = lambda s: len(s)

# 9. First character
first_char = lambda s: s[0]

# 10. Sum of list
sum_list = lambda lst: sum(lst)
```

—

3 Map Function

Definition: Map function ek function ko iterable ke har element pe apply karta hai aur ek map object return karta hai.

Roman Urdu Explanation:
Map ka kaam hai "apply a function to each element" ek list ya iterable ke andar. Lambda ke saath commonly use hota hai.

Syntax:

```python
map(function, iterable)
```

Examples:

```python
# 1. Square numbers
nums = [1,2,3,4]
squares = list(map(lambda x: x**2, nums))

# 2. Cube numbers
cubes = list(map(lambda x: x**3, nums))

# 3. Add 5 to each number
added = list(map(lambda x: x+5, nums))

# 4. Convert to string
str_nums = list(map(str, nums))

# 5. Uppercase words
words = ["hello","world"]
upper_words = list(map(lambda x: x.upper(), words))

# 6. Length of words
lengths = list(map(lambda x: len(x), words))

# 7. Double each number
doubles = list(map(lambda x: x*2, nums))

```

```python
# 8. First letter of each word
first_letters = list(map(lambda x: x[0], words))

# 9. Boolean even check
evens = list(map(lambda x: x%2==0, nums))

# 10. Multiply two lists element-wise
a = [1,2,3]; b=[4,5,6]
prod = list(map(lambda x,y: x*y, a,b))
```

—

4 Filter Function

Definition: Filter function iterable ke elements ko condition ke basis pe filter karta hai aur ek filter object return karta hai.

Roman Urdu Explanation:
Filter function iterable me se sirf wo elements choose karta hai jo condition me true hon. Lambda ke saath bohot use hota hai.

Syntax:

```python
filter(function, iterable)
```

Examples:

```python
# 1. Even numbers
nums = [1,2,3,4,5]
evens = list(filter(lambda x: x%2==0, nums))

# 2. Odd numbers
odds = list(filter(lambda x: x%2!=0, nums))

# 3. Numbers > 3
gt3 = list(filter(lambda x: x>3, nums))

# 4. Words starting with 'h'
words = ["hello","world","hi"]
h_words = list(filter(lambda x: x.startswith('h'), words))

# 5. Words with length>3
long_words = list(filter(lambda x: len(x)>3, words))

# 6. Negative numbers
nums2 = [-1,0,1,-5,5]
neg = list(filter(lambda x: x<0, nums2))

# 7. Non-empty strings
strs = ["","hello"," "]
non_empty = list(filter(lambda x: x.strip()!="", strs))

# 8. Numbers divisible by 3
div3 = list(filter(lambda x: x%3==0, range(10)))
```

```
28
29 # 9. Uppercase words only
30 words2 = ["Hello","WORLD","python"]
31 upper = list(filter(lambda x: x.isupper(), words2))
32
33 # 10. Palindromes
34 words3 = ["madam","hello","level"]
35 palindromes = list(filter(lambda x: x==x[::-1], words3))
```

—

5 Reduce Function

Definition: Reduce function iterable ke elements ko ek single value me accumulate karta hai using a binary function. Ye Python me `functools` module me hoti hai.

Roman Urdu Explanation:
Reduce ka kaam hai elements ko step by step combine karna aur final ek value return karna, jaise sum ya product.

Syntax:

```
1 from functools import reduce
2 reduce(function, iterable, initializer)
```

Examples:

```
1 from functools import reduce
2
3 # 1. Sum of list
4 nums = [1,2,3,4]
5 sum_all = reduce(lambda x,y: x+y, nums)
6
7 # 2. Product of list
8 product = reduce(lambda x,y: x*y, nums)
9
10 # 3. Maximum element
11 max_num = reduce(lambda x,y: x if x>y else y, nums)
12
13 # 4. Minimum element
14 min_num = reduce(lambda x,y: x if x<y else y, nums)
15
16 # 5. Concatenate strings
17 words = ["Hello","World"]
18 sentence = reduce(lambda x,y: x+" "+y, words)
19
20 # 6. Factorial
21 fact = reduce(lambda x,y: x*y, range(1,6))  # 5! = 120
22
23 # 7. Sum of squares
24 sum_squares = reduce(lambda x,y: x+y, [x**2 for x in nums])
25
26 # 8. Largest even number
```

```
largest_even = reduce(lambda x,y: x if x>y else y, filter(lambda
    x:x%2==0, nums))

# 9. Multiply numbers + add 1
nums2 = [1,2,3]
custom = reduce(lambda x,y: x*y+1, nums2)

# 10. Flatten nested list
nested = [[1,2],[3,4],[5]]
flat = reduce(lambda x,y: x+y, nested)
```

—

Summary Table

Concept	Purpose	Syntax
List Comprehension	Create list concisely	[expr for item in iterable if
Lambda	Anonymous function	lambda args: expr
Map	Apply function to each element	map(func, iterable)
Filter	Filter elements based on condition	filter(func, iterable)
Combine elements into single value	reduce(func, iterable)	reduce(lambda x,y:x+y, nu