PYTHON COMPREHENSIONS - COMPLETE NOTES

Comprehensions are a short and powerful way to create:

- Lists
- Sets
- Dictionaries
- Generators

They make code:

1. Cleaner
2. Faster
3. More Pythonic

WHY COMPREHENSIONS?

Used for:

- Filtering items
- Transforming items
- Creating new collections
- Cleaning data (strip, lower, remove duplicates)
- Flattening nested structures

TYPES OF COMPREHENSIONS

1. List Comprehension
2. Set Comprehension
3. Dictionary Comprehension
4. Generator Comprehension

LIST COMPREHENSION

# Basic example

```
squares = [x * x for x in range(5)]
print(squares)   # [0,1,4,9,16]
```

# With condition

```
even_nums = [x for x in range(10) if x % 2 == 0]
print(even_nums)
```

```
# Convert names to uppercase

names = ["ram", "shyam", "mohan"]
upper_names = [n.upper() for n in names]
print(upper_names)
```

LIST COMPREHENSION - FILTERING EXPLAINED

Syntax:

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

```
menu = ["Iced Green Tea", "Black Tea", "Iced Lemon Tea"]
iced_teas = [my_tea for my_tea in menu if "Iced" in my_tea]
print(iced_teas)

# Filter by length

menu = ["Masala Chai", "Iced Lemon Tea", "Green Tea", "Iced
Peach Tea", "Ginger chai"]
long_names = [item for item in menu if len(item) > 10]
print(long_names)

iced_tea = [tea for tea in menu if "Iced" in tea]
print(iced_tea)
```

FLATTEN NESTED LIST

```
nested = [[1, 2], [3, 4], [5, 6]]
flat = [num for sub in nested for num in sub]
print(flat)
```

SET COMPREHENSION

```
unique_set = {x for x in [1, 2, 2, 3, 3, 4]}
print(unique_set)

squares_set = {x * x for x in range(5)}
print(squares_set)
```

DICTIONARY COMPREHENSION

```
square_map = {x: x * x for x in range(5)}
print(square_map)

even_square_map = {x: x * x for x in range(10) if x % 2 == 0}
print(even_square_map)
```

GENERATOR COMPREHENSION

```
gen = (x * x for x in range(5))
print(next(gen))
print(next(gen))
print(list(gen))
```

REAL-LIFE USE CASES

- Filtering (even numbers)
- Uppercase / lowercase conversion
- Extracting specific items
- Removing duplicates
- Creating new dicts
- Cleaning datasets
- Flattening nested lists

DIFFERENCE BETWEEN LIST, SET, DICT COMPREHENSION

```
[] → List
{} → Set (no duplicates)
{key:value} → Dictionary
```

SET COMPREHENSION EXAMPLES

```
favourite_chais = [
    "Masala Chai", "Green Tea", "Masala Chai",
    "Lemon Tea", "Green Tea", "Elaichi Chai"
]

unique_chai = {chai for chai in favourite_chais}
print(unique_chai)

unique_chai_len = {chai for chai in favourite_chais if
len(chai) > 8}
print(unique_chai_len)
```

```
recipes = {
    "Masala Chai": ["ginger", "cardamom", "clove"],
    "Elaichi Chai": ["cardamom", "milk"],
    "Spicy Chai" : ["ginger", "black pepper", "clove"]
}

unique_spices = {spice for ingredients in recipes.values() for
spice in ingredients}
print(unique_spices)
```

DICTIONARY COMPREHENSION NOTES

```
numbers = [1, 2, 3, 4, 5]
squared_dict = {num: num**2 for num in numbers}
print(squared_dict)

tea_prices_inr = {"Masala Chai": 40, "Green Tea": 50, "Lemon
Tea": 200}
tea_prices_usd = {tea: price/80 for tea, price in
tea_prices_inr.items()}
print(tea_prices_usd)

tea_prices_filtered = {tea: price_usd for tea, price_usd in
tea_prices_usd.items() if price_usd > 1}
print(tea_prices_filtered)
```

LIST COMPREHENSIONS vs GENERATOR EXPRESSIONS

```
nums = [i*i for i in range(10)]
print(nums)

squares_gen = (i*i for i in range(10))
print(next(squares_gen))
print(next(squares_gen))
print(list(squares_gen))

daily_sales = [5,10,12,7,3,8,9,15]
total_cups = sum(sale for sale in daily_sales if sale > 5)
print(total_cups)

total_cups_list = sum([sale for sale in daily_sales if sale > 5])
print(total_cups_list)
```

Memory difference

```
import sys
big_range = range(10_000_000)

lst = [x for x in big_range]
gen = (x for x in big_range)

print(sys.getsizeof(lst))
print(sys.getsizeof(gen))
```

EXTRA: Lazy nested generator

```
list_of_lists = ([i + j for j in range(3)] for i in range(5))
flat_gen = (val for sub in list_of_lists for val in sub)
print(list(flat_gen))
```

All Comprehension Examples

1. List Comprehension: Square Values

```
lst1 = [3, 6, 12, 7, 19, 25]

sq_list = [val**2 for val in lst1]

print("Given List:", lst1)
print("Square List:", sq_list)
```

2. List Comprehension: Positive & Negative

```
lst2 = [3, 6, -12, 7, -19, 25, -56, 23, -78, 89, 0, 12]

positive_list = [val for val in lst2 if val > 0]
negative_list = [val for val in lst2 if val < 0]

print("Given List:", lst2)
print("Positive Numbers:", positive_list)
print("Negative Numbers:", negative_list)
```

3. Without Comprehension (Normal Method)

```
lst3 = [3, 6, 12, 7, 19, 25]
sq_list2 = []

for val in lst3:
    sq_list2.append(val**2)

print("Given List:", lst3)
print("Square List:", sq_list2)
```

4. Dictionary Comprehension

```
lst4 = [3, 6, 12, 7, 19, 25]

sq_dict = {val: val**2 for val in lst4}

for n, s in sq_dict.items():
    print(f"{n} --> {s}")
```

5. Read Space-Separated Values

```
print("Enter values separated by space:")
values_space = [int(val) for val in input().split()]
print("Given List:", values_space)
```

6. Read Comma-Separated Values (string)

```
print("Enter values separated by comma:")
values_comma = [val for val in input().split(",")]
print("Given List:", values_comma)
```

7. Read Comma-Separated Values (force string)

```
print("Enter values separated by comma:")
values_comma2 = [str(val) for val in input().split(",")]
print("Given List:", values_comma2)
```

8. Set Comprehension

```
lst5 = [3, 6, 12, 7, 19, 25]

sq_set = {val**2 for val in lst5}

print("Given List:", lst5)
print("Square Set:", sq_set)
```

9. Tuple (Generator) Comprehension

```
lst6 = [3, 6, 12, 7, 19, 25]

gen_obj = (val**2 for val in lst6)
tpl = tuple(gen_obj)

print("Given List:", lst6)
print("Square Tuple:", tpl)
```

PYTHON COMPREHENSION PRACTICE QUESTIONS

PART 1
LIST COMPREHENSION (EASY)

1. Create a list of squares (1 to 10)

2. Generate a list of even numbers between 1 and 50

3. Convert list of names to uppercase
   `names = ["ram", "shyam", "radha", "mohan"]`

4. Extract only positive numbers
   `nums = [12, -4, 0, 9, -7, 5]`

5. Filter items with length > 4
   `fruits = ["apple", "banana", "kiwi", "fig", "melon"]`

PART 2
LIST COMPREHENSION WITH CONDITIONS (MEDIUM)

1. Extract words containing 'a'
   `items = ["tea", "coffee", "milk", "water", "juice"]`

2. Numbers divisible by 5 but NOT by 10
   `nums = [5, 10, 15, 20, 25, 30]`

3. Flatten nested list
   `nested = [[1, 2], [3, 4], [5, 6]]`

4. Convert Celsius to Fahrenheit
   `temps = [0, 10, 20, 30, 40]`

5. Extract vowels from string
   `word = "education"`

PART 3
SET COMPREHENSION (MEDIUM)

1. Create a set of squares 1-10

2. Unique values from list
```
numbers = [1, 2, 2, 3, 3, 3, 4, 5]
```

3. Unique words with length > 3
```
words = ["tea", "coffee", "chai", "tea", "coffee"]
```

4. Unique characters from string
```
text = "programming"
```

5. Unique ingredients from dictionary
```
recipes = {
    "chai": ["ginger", "cardamom"],
    "coffee": ["milk", "sugar"],
    "kadha": ["ginger", "clove"]
}
```

PART 4
DICTIONARY COMPREHENSION (MEDIUM-HARD)

1. Dictionary of number → square (1 to 10)

2. Convert km to meters (value * 1000)
```
dist = {"A": 2, "B": 5, "C": 10}
```

3. Filter items with value > 10
```
prices = {"tea": 10, "coffee": 40, "juice": 25, "water": 5}
```

4. Convert list to dict using enumerate
```
items = ["chai", "coffee", "milk"]
```

5. Swap key:value in dictionary
```
data = {"a": 1, "b": 2, "c": 3}
```

PART 5
GENERATOR COMPREHENSION (ADVANCED)

1. Generator for squares of first 10 numbers

2. Sum of odd numbers using generator
   ```
   nums = [1, 2, 3, 4, 5, 6, 7]
   ```

3. Generator that yields uppercase chars only
   ```
   text = "PyThOn ProGRamMinG"
   ```

4. Count numbers > 50 using generator
   ```
   values = [10, 60, 80, 45, 30, 99]
   ```

5. Generator to flatten nested list
   ```
   nested2 = [[10, 20], [30, 40], [50, 60]]
   ```

PYTHON COMPREHENSIONS - INTERVIEW QUESTIONS

BASIC

1. What are Python comprehensions? Why are they used?
2. What types of comprehensions exist in Python?
3. Difference between list comprehension and generator comprehension?
4. Write a simple list comprehension to generate squares of numbers 1 to 10
5. How do you apply a condition inside a comprehension? Give example
6. Explain nested list comprehension with an example

INTERMEDIATE

1. Flatten a nested list using list comprehension
2. Remove duplicates from a list using comprehension
3. Swap keys and values in a dictionary using dictionary comprehension
4. Convert all strings in a list to uppercase using list comprehension
5. Create a set comprehension to get unique vowels from a string
6. Use a generator comprehension to calculate sum of squares 1 to 100
7. Dictionary comprehension to filter items based on values (e.g., value > 10)

ADVANCED

1. Can comprehensions contain multiple for loops? Give example
2. Difference between {x for x in iterable} and [x for x in iterable]
3. Explain memory efficiency of generators vs lists
4. Use if-else condition inside a comprehension (show example)
5. Flatten a nested dictionary or list of dictionaries using comprehension
6. One-liner to transpose a matrix using list comprehension
7. Use comprehensions for data cleaning (strip, lower, remove duplicates)
8. Can you chain multiple comprehensions together? Show example

BEHAVIOR CHECK

1. What happens if you use next() on a generator multiple times?
2. Difference between [x*x for x in range(1000)] and (x*x for x in range(1000))
3. Can dictionary keys be generated dynamically in comprehension? Give example
4. How to use comprehensions with functions like sum(), any(), all(), max()?