Python Collections – From Basic to Advanced

# Introduction to Python Collections

Python has four built-in collection data types: List, Tuple, Set, and Dictionary.  
They differ by order, mutability, and whether they allow duplicates.

# Lists

Lists are ordered, mutable, and allow duplicates.  
Example:  
```python  
fruits = ["apple", "banana", "cherry"]  
fruits.append("mango")  
print(fruits)  
```

# Tuples

Tuples are ordered and immutable.  
Example:  
```python  
coordinates = (10, 20)  
x, y = coordinates  
print(x, y)  
```

# Sets

Sets are unordered, mutable, and do not allow duplicates.  
Example:  
```python  
A = {1, 2, 3}  
B = {3, 4, 5}  
print(A.union(B))  
```

# Dictionaries

Dictionaries store key-value pairs.  
Example:  
```python  
student = {"name": "Gopi", "age": 25}  
student["city"] = "Mumbai"  
print(student)  
```

# Advanced Collections (collections module)

Includes namedtuple, deque, Counter, defaultdict, OrderedDict, ChainMap.  
Example:  
```python  
from collections import Counter  
cnt = Counter(['apple', 'banana', 'apple'])  
print(cnt)  
```

Python Collections – Part 2 (Advanced Topics)

# Nested Collections and Comprehensions

Example 1: Nested List Comprehension  
```python  
matrix = [[j for j in range(3)] for i in range(3)]  
flat = [num for row in matrix for num in row]  
```  
  
Example 2: Dictionary Comprehension  
```python  
names = ['Ram', 'Sita', 'Laxman']  
ages = [25, 24, 23]  
info = {name: age for name, age in zip(names, ages)}  
```

# Sorting and Filtering Collections

Sort List of Dicts by Value  
```python  
students = [{'name':'Anil','score':85},{'name':'Bala','score':95}]  
sorted\_students = sorted(students, key=lambda x: x['score'], reverse=True)  
```  
  
Filter Example  
```python  
nums = [10,15,20,25]  
evens = list(filter(lambda x: x%2==0, nums))  
```  
  
Map and Reduce  
```python  
from functools import reduce  
nums = [1,2,3,4]  
squared = list(map(lambda x:x\*\*2, nums))  
sum\_total = reduce(lambda a,b:a+b, nums)  
```

# Conversion Between Collections

List → Set: set([1,2,2,3])  
Set → List: list({1,2,3})  
Tuple → List: list((1,2,3))  
List → Tuple: tuple([1,2,3])  
Dict Keys → List: list(student.keys())  
Dict Values → Set: set(student.values())

# Advanced Dictionary Operations

Merge Dictionaries  
```python  
a={'x':1,'y':2}; b={'y':3,'z':4}  
merged = {\*\*a, \*\*b}  
```  
  
Sort by Values  
```python  
marks={'A':90,'B':70,'C':80}  
sorted\_marks = dict(sorted(marks.items(), key=lambda item:item[1], reverse=True))  
```  
  
Nested Dictionary  
```python  
students={'Ram':{'Math':90}, 'Sita':{'Science':92}}  
print(students['Ram']['Math'])  
```

# Advanced Set Operations

Frozen Set  
```python  
fs = frozenset([1,2,3])  
print(fs)  
```  
  
Subset and Superset  
```python  
A={1,2}; B={1,2,3}  
print(A.issubset(B)); print(B.issuperset(A))  
```

# Real-World Mini Projects

Inventory System  
```python  
inventory = {"Fruits":[{"name":"Apple","qty":50},{"name":"Banana","qty":30}]}  
inventory["Fruits"].append({"name":"Mango","qty":15})  
for item in inventory["Fruits"]:  
 if item["name"]=="Banana":  
 item["qty"]+=10  
```  
  
Word Frequency Counter  
```python  
from collections import Counter  
text = "data data engineering with python python python"  
freq = Counter(text.split())  
print(freq.most\_common())  
```  
  
Student Marks Average  
```python  
students={"A":[85,90,78],"B":[70,88,82]}  
averages={s:sum(m)/len(m) for s,m in students.items()}  
print(averages)  
```

# Performance Tips & Best Practices

Fast lookups → set/dict (O(1) average)  
Ordered iteration → list/OrderedDict  
Immutable data → tuple/namedtuple  
Counting → Counter  
Queue/Stack → deque  
Numeric → array

# Summary of Advanced Functions

zip(), enumerate(), sorted(), reversed(), any(), all(), sum(), min(), max(), len(), type(), isinstance()