Day 3 of Training at Ansh Info Tech

Topics Covered

Sets in Python

- Basic Set Operations
- Set Methods
- Immutability and Sets
- Frozen Sets
- Set Applications
- 25 Practice Questions on Sets

Dictionaries in Python

- Basic Dictionary Operations
- Dictionary Iteration
- Dictionary Functions
- Dictionary Methods
- 30 Practice Questions on Dictionaries

Summary

Sets in Python

Sets in Python are collections of unique elements. You can create sets using curly braces `{}` or the `set()` constructor. Key operations include:

- **Union**: `A.union(B)` or `A | B`
- **Intersection**: `A.intersection(B)` or `A & B`
- **Difference**: `A.difference(B)` or `A - B`
- **Symmetric Difference**: `A.symmetric_difference(B)` or `A ^ B`
- **Superset**: `A.issuperset(B)` or `A >= B`
- **Subset**: `A.issubset(B)` or `A <= B`
- **Disjoint**: `A.isdisjoint(B)`

Set methods include `.add()`, `.discard()`, `.remove()`, `.update()`, and more. Frozen sets are immutable versions of sets, created using `frozenset()`.

Dictionaries in Python

Dictionaries are collections of key-value pairs. Keys must be unique and hashable. Key operations include:

- **Accessing elements**: `dict1['key']`
- **Modifying elements**: `dict1['key'] = value`
- **Dictionary Methods**: `.get(key)`, `.clear()`, `.items()`, `.keys()`, `.values()`, `.update()`, `.pop(key)`, `.popitem()`

Dictionaries are highly useful for mapping and lookup operations.

Practice Questions on Sets (25)

1. You have two sets representing students enrolled in Math and Science Classes. Find the set of students who are enrolled in both classes.
2. Given a set of unique employee IDs, add a new employee ID to the set.
3. You are managing two different projects and have two sets representing team members for each project. Find the set of team members who are working on either project or both projects.
4. You have a set of customer IDs who have made purchases this month. Remove a customer ID from the set who has canceled their order.
5. **And many more...**

Practice Questions on Dictionaries (30)

1. Create a dictionary representing a phone book with names as keys and phone numbers as values. Add a new contact to the phone book.
2. Write a function that takes a dictionary and returns a new dictionary with keys and values swapped.
3. Given a dictionary of student names and their grades, find the student with the highest grade.
4. Update the phone book dictionary by changing the phone number of an existing contact.
5. Write a function that merges two dictionaries, giving precedence to the second dictionary in case of key conflicts.
6. **And many more...**

set -> consist of unique elements only like maths {} <- used for definition set() -> set constructor for set creation

frozenset-> immutable and new elements cannot be added after definition frozenset() -> constructor for frozen set creation

Set Operations :-

- A.union(B) or A|B
- A.intersection(B)
- A.difference(B)
- A.symmetric_difference(B) or A^B
- A.issuperset(B) or >=
- A.issubset(B) or <=
- for proper subset we use <
- A.isdisjoint(B)
- set() -> returns an empty set

With single elements:

- in #Membership Check
- .add(-) #Add an Element to Set
- .discard(-) #Remove Element if present
- .remove(-) #Remove Element if present otherwise error
- .update(s1) #Add elements of the argument set to the set

Other Operations:

- s|=t
- s&=t
- s-=t

l1 = list(set(l1)) # Remove Duplicates from the list

{{1,2}, {3,4}} #Nesting of Sets gives an error for nesting of sets we use frozenset() in the inner sets

where A and B are sets

Dictionary-> Use key Value pair for accss

Every key must be unique, hashable

access using key print(dict1['name'])

dict()-> empty dictionary constructor

d = dict(key = 'value') #Explicit specify

d = dict([('key', 'value')]) # passing in a list of key/value pairs

di = dict(Name = 'Diksha')

Modify a Dict -> d['House_No'] = 42

Dictionary Methods

- .get(key) -> get value for the argument key
- .clear() -> clear all elements of the dictionary

- .items() -> gives a list of tuples, where each tuple contains a key-value pair
- .keys() -> gives list of keys
- .values() -> gives list of values
- .update(d2) -> adds new key-values to dictionary or modifies values of existing keys
- .pop(key) -> Removes a key-value based on key
- .popitem(key) -> Removes random key-value pair from the dictionary

```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
print("Checking Union")
print(A.union(B))
print(A|B)
print("Checking Intersection")
print(A.intersection(B))
print(A&B)
print("Checking Difference")
print(A.difference(B))
print(A-B)
print("Checking Symmetric Difference")
print(A.symmetric_difference(B))
print(A^B)
print("Checking Superset")
print(A.issuperset(B))
print(A>=B)
print(A.issuperset({1,2,3}))
print(A>= {1,2,3})
print("Checking Subset")
print(A.issubset(B))
print(A<=B)
print(A.issubset({1,2,3,4,5,6,7}))
print(A<= {1,2,3,4,5,6,7})
print("Checking Disjoint")
print(A.isdisjoint(B))
print(A.isdisjoint({1,2,3,4,5,6}))
print(B.isdisjoint({1,2,3,4,5,6}))
print("Checking Empty Set")
C = frozenset({1,2,3,4,5})
print(C)
D = set()
print(D)
```

```
Checking Union
{1, 2, 3, 4, 5, 6, 7, 8}
{1, 2, 3, 4, 5, 6, 7, 8}
Checking Intersection
{4, 5}
{4, 5}
Checking Difference
{1, 2, 3}
{1, 2, 3}
```

```
Checking Symmetric Difference
{1, 2, 3, 6, 7, 8}
{1, 2, 3, 6, 7, 8}
Checking Superset
False
False
True
True
Checking Subset
False
False
True
True
Checking Disjoint
False
False
False
Checking Empty Set
frozenset({1, 2, 3, 4, 5})
set()
```

1. Create a list of integers from 1 to 5.
2. Append the number 6 to the list.
3. Extend the list with another list: [7, 8, 9].
4. Access and print the third element of the list.
5. Slice the list to get elements from index 2 to 4.
6. Reverse the list.
7. Remove the element 7 from the list.
8. Check if the element 5 is present in the list.
9. Find the index of the element 8 in the list.
10. Create a new list that contains only the even numbers from the original list.

```
li = [1,2,3,4,5]
li.append(6)
li.extend([7,8,9])
print(li[2])
print(li[2:5])
li.reverse()
print(li)
li.remove(7)
print(li)
print(5 in li)
print(li.index(8))
even_list = [num for num in li if num % 2 == 0]
print(even_list)
```

1. Create a set with the first five prime numbers.

2. Add the number 7 to the set.
3. Remove the number 3 from the set.
4. Check if the set is a subset of {2, 3, 5, 7, 11}.
5. Find the union of the set with {7, 11, 13}.
6. Create a frozen set from the original set.
7. Check if the set has any common elements with {1, 4, 9}.
8. Remove all elements from the set.
9. Create a set of your favorite fruits and find the intersection with { 'apple', 'banana', 'orange' }.
10. Use set comprehension to create a set of squares for numbers 1 to 10.

```
from re import S
s1 = {2,3,5,7,11}
s1.add(7)
s1.remove(3)
print(s1)
s1.issubset({2,3,5,7,11})
s1.union({7,11,13})
s2 = frozenset(s1)
print(s2)
if(s1.isdisjoint({1,4,9})):
    print("Set has no Common Elements with {1, 4, 9}")
else:
    print("Set has Common Elements with {1, 4, 9}")
    print(s1.intersection({1,4,9}))
s1.clear()
fruits = { 'apple', 'banana', 'mango', 'cherry' }
print(fruits.intersection({'apple', 'banana', 'orange'}))
squares = set()
for i in range(1,11):
    squares.add(i**2)
print(squares)
```

```
{2, 5, 7, 11}
frozenset({2, 11, 5, 7})
Set has no Common Elements with {1, 4, 9}
{'apple', 'banana'}
{64, 1, 4, 36, 100, 9, 16, 49, 81, 25}
```

Start coding or generate with AI.

Exercise 2: Tuples

1. Create a tuple with three different data types: int, float, and string.

2. Access and print the second element of the tuple.
3. Concatenate the tuple with another tuple containing three elements.
4. Unpack the tuple into three variables: a, b, and c.
5. Check if the string "apple" is present in the tuple.
6. Create a new tuple by repeating the original tuple three times.
7. Find the index of the float element in the tuple.
8. Convert the tuple to a list.
9. Create a tuple of your favorite colors and concatenate it with the original tuple.
10. Use a tuple to swap the values of two variables.

```
t1 = (5, 7.5, "apple")
print(t1[1])
t2 = (3, "Mango", 6.2)
t3 = t1 + t2
print(t3)
a,b,c = t1
print(a)
print(b)
print(c)
for  i in t1:
    if(i == "apple"):
        print("apple is present in the tuple")
t4 = t1*3
print(t4)
t1 = (5, 7.5, "apple")
for i in t1:
    if type(i) == 'float':
        print(t1.index(i))
l1 = tuple(t1)
t5 = ('blue', 'yellow', 'red')
t1 = t1 + t5
print(t1)
#Use of tuple to swap the value of 2 variables
a = 10
b = 20
print(a,b)
t1 = (a,b)
(b,a) = t1
print(a,b)
```

```
7.5
(5, 7.5, 'apple', 3, 'Mango', 6.2)
5
7.5
apple
apple is present in the tuple
(5, 7.5, 'apple', 5, 7.5, 'apple', 5, 7.5, 'apple')
(5, 7.5, 'apple', 'blue', 'yellow', 'red')
```

```
10 20
20 10
```

Other Questions:

1. Basic Set Operations: What is a set in Python? Explain the difference between a set and a list in Python. How do you create an empty set in Python? Describe the basic set operations: union, intersection, and difference.
2. Set Methods: Name three common methods available for sets in Python. How do you add an element to a set? Explain the purpose of the remove() and discard() methods in sets.
3. Set Operations using Operators: What operators are used for set union and intersection in Python? Write code to perform the union of two sets. How do you check if one set is a subset of another set?
4. Immutability and Sets: Can you have a set of sets in Python? Explain why sets are mutable but elements of sets must be immutable. Provide an example of an immutable data type that can be used as a set element.
5. Frozen Sets: ▯ TESTING THE LEARNING SO FAR: What is a frozen set in Python? How does a frozen set differ from a regular set? In what scenarios might you choose to use a frozen set?
6. Removing Duplicates using Sets: How can you use a set to remove duplicates from a list? Write a short code snippet to demonstrate removing duplicates from a list using a set.
7. Real-world Applications: Provide an example of a real-world scenario where using a set in Python would be beneficial. How could sets be used in data analysis or processing?

Real-world Applications: Provide an example of a real-world scenario where using a set in Python would be beneficial. How could sets be used in data analysis or processing?

```
# prompt: Real-world Applications:
# Provide an example of a real-world scenario where using a set in Python would be benefici
# How could sets be used in data analysis or processing?

# Real-world Applications:

# Example of using a set in Python:
# A company wants to track the unique customers who have visited their website.
# They can use a set to store the customer IDs of all the visitors.
# This allows them to easily identify the number of unique visitors and avoid counting the s

customers = set()

# Adding customer IDs to the set
customers.add("customer1")
customers.add("customer2")
customers.add("customer3")

# Checking if a customer is in the set
if "customer2" in customers:
    print("Customer 2 is in the set")

# Getting the number of unique customers
num_customers = len(customers)
print("Number of unique customers:", num_customers)

# Using sets in data analysis or processing:
# Sets can be used to perform various data analysis tasks, such as:

# Identifying unique elements in a dataset
# Removing duplicates from a dataset
# Finding the intersection or union of two datasets
# Grouping data based on common characteristics

# Example of using sets in data analysis:
# A data analyst wants to find the unique words in a large text file.
# They can use a set to store the unique words and then count the number of unique words in

with open("text_file.txt", "r") as f:
    words = set(f.read().split())

# Getting the number of unique words
num_words = len(words)
print("Number of unique words:", num_words)
```

Practice Questions on Sets

1. You have two sets representing students enrolled in Math and Science Classes. Find the set of Students who are enrolled in both classes.

```
s1 = {'a', 'b', 'c', 'd', 'g'}
s2 = {'c', 'd', 'e', 'f', 'g'}
s3 = s1.intersection(s2)
print(s3)
```

2. Given a set of unique employee IDs, add a new employee ID to the set.

```
setID = {'E1', 'E2', 'E3', 'E4', 'E5'}
setID.add('E6')
print(setID)
```

3. You are managing two different projects and have 2 sets representing team members for each project. Find the set of team members who are working on either project or both projects.

```
t1 = {'a', 'b', 'c', 'd'}
t2 = {'c', 'd', 'e', 'f'}
t3 = t1.union(t2)
print(t3)
```

4. You have a set of customer IDs who have made purchases this month. Remove a customer ID from the set who has canceled their order.

```
setID = {'C1', 'C2', 'C3', 'C4', 'C5'}
setID.remove('C3')
print(setID)
```

5. Given two sets of product IDs, one representing products in stock and the other representing products that have been sold, find the set of products that are still in stock.

```
stock = {'P1', 'P2', 'P3', 'P4', 'P5'}
sold = {'P1', 'P3', 'P5'}
instock = stock.difference(sold)
print(instock)
```

6. You have a set of registered conference attendees. Check if a particular person is registered.

```
setAttendees = {'P1', 'P2', 'P3', 'P4', 'P5'}
if 'P3' in setAttendees:
  print("P3 is a registered Attendee")
else:
  print("P3 is not a registered Attendee")
```

7. Given two sets of book titles, one representing books available in the library and the other representing books borrowed by students, find the set of books that are not currently available in the library.

```
setAvailable = {'B1', 'B2', 'B3', 'B4', 'B5'}
setBorrowed = {'B1', 'B3', 'B5'}
setNotAvailable = setBorrowed
print(setNotAvailable)
```

8. You have a set of unique tags assigned to a blog post. Add multiple new tags to the set at once.

```
setTags = {'T1', 'T2', 'T3', 'T4'}
setTags.update({'T5', 'T6'})
print(setTags)
```

9. Given a set of employee skills and another set of skills required for a new project, find the set of skills that need to be acquired or improved.

```
setPresent = {'S1', 'S2', 'S3', 'S4', 'S5'}
setRequired = {'S2', 'S4', 'S6'}
setMissing = setRequired.difference(setPresent)
print(setMissing)
```

10. You have two sets of favorite movies from two different friends. Find the set of movies that are favorite to at least one of them but not both.

```
setFav1 = {'M1', 'M2', 'M3', 'M4', 'M5'}
setFav2 = {'M1', 'M3', 'M5', 'M7', 'M9'}
setFav = setFav1.symmetric_difference(setFav2)
print(setFav)
```

11. Given a set of emails collected from a signup form, ensure that the set contains only unique emails by checking the length before and after adding more emails.

```
setEmails = {'E1', 'E2', 'E3', 'E4', 'E5'}
print(len(setEmails))
setEmails.add('E6')
setEmails.add('E4')
print(len(setEmails))
```

```
5
6
```

12. You have a set of courses you are interested in and another set of courses you have completed. Find the set of courses you are still interested in but have not yet completed.

```
setInterested = {'C1', 'C2', 'C3', 'C4', 'C5'}
setCompleted = {'C1', 'C3', 'C5'}
setInterested = setInterested.difference(setCompleted)
print(setInterested)
```

13. Given a set of cities visited last year and another set of cities visited this year, find the set of cities that were visited in either year but not both.

```
setVisitedLastYear = {'C1', 'C2', 'C3', 'C4'}
setVisitedThisYear = {'C2', 'C5', 'C3', 'C9'}
setVisited = setVisitedLastYear.symmetric_difference(setVisitedThisYear)
print(setVisited)
```

14. You have a set of parts required to build a machine. Remove a part from the set if it is faulty.

```
setParts = {'P1', 'P2', 'P3', 'P4', 'P5'}
setParts.remove('P3')
print(setParts)
```

15. Given two sets representing two different sports clubs' members, find the set of members who are exclusive to the first club.

```
setMembers1 = {'M1', 'M2', 'M3', 'M4', 'M5'}
setMembers2 = { 'M1', 'M3', 'M5', 'M7', 'M9'}
setExclusive = setMembers2.difference(setMembers1) #Members who are not in Club 1
print(setExclusive)
```

16. You have a set of ingredients required for a recipe. Check if a specific ingredient is missing.

```
setIngredients = {'I1', 'I2', 'I3', 'I4', 'I5'}
if 'I6' in setIngredients:
  print("I6 is a not missing ingredient")
else:
  print("I6 is a missing ingredient")
```

17. Given a set of friends on social media and another set of mutual friends with a colleague, find the set of mutual friends.

```
setSocialFriends = {'F1', 'F2', 'F3', 'F4', 'F5'}
setMutualFriendsColleague = {'F1', 'F3', 'F5'}
setMutualFriends = setSocialFriends.intersection(setMutualFriendsColleague)
print(setMutualFriends)
```

18. You have a set of items in your inventory and another set representing items that need restocking. Find the set of items that are currently in stock and need restocking.

```
setInventory = {'I1', 'I2', 'I3', 'I4', 'I5'}
setRestocking = {'I1', 'I3', 'I5'}
setInStock = setInventory.intersection(setRestocking)
print(setInStock)
```

19. Given a set of languages spoken by employees in a company and another set of languages required for a new project, find the set of languages that are both spoken by employees and required for the project.

```
setEmployees = {'L1', 'L2', 'L3', 'L4', 'L5'}
setRequired = {'L2', 'L4', 'L6'}
setCommon = setEmployees.intersection(setRequired)
print(setCommon)
```

20. You have a set of completed tasks and another set of tasks for a project. Find the set of tasks that are yet to be completed.

```
setCompleted = {'T1', 'T2', 'T3', 'T4', 'T5'}
setTasks = {'T1', 'T3', 'T5', 'T7', 'T9'}
setYetToComplete = setTasks.difference(setCompleted)
print(setYetToComplete)
```

21. Given two sets of preferred travel destinations from two friends, find the set of destinations that both friends prefer.

```
setPref1 = {'D1', 'D2', 'D3', 'D4', 'D5'}
setPref2 = {'D1', 'D3', 'D5', 'D7', 'D9'}
setPref = setPref1.intersection(setPref2)
print(setPref)
```

22. You have a set of students who have submitted an assignment and another set of students who have attended a class. Find the set of students who have either submitted the assignment or attended the class.

```
setSubmitted = {'S1', 'S2', 'S3', 'S4', 'S5'}
setAttended = {'S1', 'S3', 'S6'}
setStudents = setSubmitted.union(setAttended)
print(setStudents)
```

23. Given a set of current employees and another set of new hires, find the set of all employees after the new hires join.

```
setCurrent = {'E1', 'E2', 'E3', 'E4', 'E5'}
setNew = {'E1', 'E3', 'E6'}
setAfterJoin = setCurrent.union(setNew)
print(setAfterJoin)
```

24. You have a set of countries visited in your lifetime and another set of countries on your travel wish list. Find the set of countries that you have already visited and are also on your wish list.

```
setVisited = {'C1', 'C2', 'C3', 'C4', 'C5'}
setWishList = {'C1', 'C3', 'C6'}
setRequired = setVisited.intersection(setWishList)
print(setRequired)
```

25. Given two sets representing the collection of books in two different libraries, find the set of books that are available in both libraries.

```
setC1 = {'B1', 'B2', 'B3', 'B4'}
setC2 = {'B1', 'B3', 'B5'}
setC3 = setC1.intersection(setC2)
print(setC3)
```

Practice Questions on Dictionaries

1. You have a dictionary representing the inventory of a store with item names as keys and quantities as values. Add a new item to the inventory with a specific quantity.

```
d1 = {'item1': 5, 'item2': 10}
d1['item3'] = 15
print(d1)
```

2. Given a dictionary of student names and their scores, find the score of a particular student.

```
d1 = {'student1': 80, 'student2': 90, 'student3': 75}
print("Score of Student 2: ")
print(d1['student2'])
```

3. You have a dictionary of country names as keys and their capitals as values. Update the capital of a specific country.

```
d1 = {'India':'Mumbai', 'Japan':'Tokyo', 'China': 'Beijing'}
print(d1)
d1['India'] = 'New Delhi'
print(d1)
```

4. Given a dictionary of product prices with product names as keys, remove a product from the dictionary.

```
d1 = {'P1':100, 'P2': 200, 'P3':150}
print(d1)
d1.pop('P2')
print(d1)
```

5. You have two dictionaries representing two different departments with employee names as keys and their salaries as values. Merge these two dictionaries into one.

```
d1 = {'E1':10000, 'E2': 20000, 'E3':15000}
d2 = {'E4':10000, 'E5': 20000, 'E6':15000}
d1.update(d2)
print(d1)
```

6. Given a dictionary of city names and their populations, check if a particular city is in the dictionary.

```
d1 = {'C1':10000, 'C2': 20000, 'C3': 150000}
if 'C2' in d1:
  print("C2 is in the dictionary")
else:
  print("C2 is not in the dictionary")
```

```
C2 is in the dictionary
```

7. You have a dictionary of book titles as keys and their authors as values. Add multiple new books to the dictionary at once.

```
d1 = {'B1':'A1', 'B2':'A2', 'B3':'A3'}
d1.update({'B4':'A4', 'B5':'A5'})
print(d1)
```

8. Given a dictionary of students' names and their grades, find the student with the highest grade.

```
d1 = {'S1':10, 'S2': 20, 'S3':15}
print(max(d1, key=d1.get))
maxVal = max(d1.values())
for key, value in d1.items():
    if value == maxVal:
        print(key)
```

```
S2
S2
```

9. You have a dictionary representing a phone book with names as keys and phone numbers as values. Remove a contact from the phone book.

```
d1 = {'a':1234567890, 'b': 0987654321, 'c': 9876543210}
print(d1)
d1.pop('b')
print(d1)
```

10. Given a dictionary of countries and their populations, create a new dictionary with only those countries that have a population greater than a specified number.

```
d1 = {'India':10000000, 'Japan': 20000000, 'China': 30000000}
d2 = {}
for key, value in d1.items():
    if value > 20000000:
        d2[key] = value
print(d2)
```

11. You have a dictionary of employee IDs as keys and their names as values. Update the name of an employee given their ID.

```
d1 = {'E1':'A', 'E2':'B', 'E3':'C'}
d1['E2'] = 'D'
print(d1)
```

12. Given a dictionary of product names and their prices, calculate the total value of all the products in the dictionary.

```
d1 = {'P1':100, 'P2': 200, 'P3':150}
print(sum(d1.values()))
```

13. You have a dictionary of movie titles as keys and their release years as values. Find all movies released before a specific year.

```
d1 = {'M1':2000, 'M2': 2005, 'M3':2010}
for key, value in d1.items():
    if value < 2005:
        print(key)
```

14. Given a dictionary of customer IDs and their order totals, find the customer with the highest order total.

```
d1 = {'C1':100, 'C2': 200, 'C3':150}
#print(max(d1, key=d1.get))
maxVal = max(d1.values())
for key, value in d1.items():
    if value == maxVal:
      print(key)
```

15. You have a dictionary of words and their definitions. Add a new word with its definition to the dictionary.

```
d1 = {'word1':'definition1', 'word2':'definition2', 'word3':'definition3'}
d1['word4'] = 'definition4'
d1.update({'word5':'definition5'})
print(d1)
```

```
{'word1': 'definition1', 'word2': 'definition2', 'word3': 'definition3', 'word4': 'defir
```

16. Given a dictionary of students and their grades, create a new dictionary where the grades are keys and the values are lists of students who have that grade.

```
d1 = {'S1':10, 'S2': 20, 'S3':15}
d2 = dict()
for key, value in d1.items():
  d2[value] = key
print(d2)
```

```
{10: 'S1', 20: 'S2', 15: 'S3'}
```

17. You have a dictionary representing a menu with dish names as keys and prices as values. Remove a dish from the menu and print the updated menu.

```
d1 = {'D1':100, 'D2': 200, 'D3':150}
print(d1)
d1.pop('D2')
print(d1)
```

18. Given a dictionary of employee names and their department names, find all employees in a specific department.

```
d1 = {'E1':'D1', 'E2':'D2', 'E3':'D3'}
for key, value in d1.items():
    if value == 'D1':
        print(key)
```

19. You have a dictionary of course names as keys and the number of enrolled students as values. Increase the number of enrolled students for a specific course by a given number.

```
d1 = {'C1':10, 'C2': 20, 'C3':15}
d1['C2'] = d1['C2'] + 5
print(d1)
```

20. Given a dictionary of book titles and their prices, create a list of all book titles that cost less than a specified amount.

```
d1 = {'B1':100, 'B2': 200, 'B3':150}
for key, value in d1.items():
    if value < 200:
        print(key)
```

21. You have a dictionary of country names as keys and their population as values. Sort the dictionary by population in ascending order.

```
d1 = {'India':10000000, 'Japan': 20000000, 'China': 30000000}
d2 = dict(sorted(d1.items(), key=lambda item: item[1]))
print(d2)
l1 = d1.items()
l1.sort(key=lambda x: x[1])
print(l1)
```

```
dict_items([('India', 10000000), ('Japan', 20000000), ('China', 30000000)])
```

22. Given a dictionary of employee names and their salaries, create a new dictionary with only those employees who earn more than a specified amount.

```
d1 = {'E1':10000, 'E2': 20000, 'E3':15000}
d2 = {}
for key, value in d1.items():
    if value > 15000:
        d2[key] = value
print(d2)
```

23. You have a dictionary of product names as keys and their quantities as values. Check if the inventory is empty and print an appropriate message.

```
d1 = {'P1':10, 'P2': 20, 'P3':15}
if len(d1) == 0:
  print("Inventory is empty")
else:
  print("Inventory is not empty")
```

24. Given a dictionary of city names and their corresponding weather conditions, find the weather condition for a specific city.

```
d1 = { 'C1':'Sunny', 'C2':'Rainy', 'C3':'Cloudy'}
print(d1['C2'])
print(d1.get('C2'))
```

```
Rainy
Rainy
```

25. You have a dictionary of student names and their test scores. Create a new dictionary where the keys are the test scores and the values are lists of students who achieved those scores.

```
d1 = {'S1':10, 'S2': 20, 'S3':15}
d2 = dict()
for key, value in d1.items():
  d2[value] = key
print(d2)
```

26. Given a dictionary of items and their prices, calculate the average price of all items.

```
d1 = {'P1':100, 'P2': 200, 'P3':150}
print(sum(d1.values())/len(d1))
```

27. You have a dictionary of employee names and their respective years of experience. Find the employee with the least experience.

```
d1 = {'E1':10, 'E2': 20, 'E3':15}
print(min(d1, key=d1.get))
minVal = min(d1.values())
for key, value in d1.items():
    if value == minVal:
        print(key)
```

28. Given a dictionary of names and ages, create a new dictionary where the keys are age groups (e.g., 20-29, 30-39) and the values are lists of names in those age groups.

```
d1 = {'N1':20, 'N2': 30, 'N3':25, 'N4':98, 'N5':32, 'N6':35, 'N7':48, 'N8':18}
d2 = {}

for name, age in d1.items():
    age_group = f"{age//10*10}-{age//10*10+9}"
    if age_group not in d2:
        d2[age_group] = []
    d2[age_group].append(name)

print(d2)
```

```
{'20-29': ['N1', 'N3'], '30-39': ['N2', 'N5', 'N6'], '90-99': ['N4'], '40-49': ['N7'], '
```