**📌 Basic Dictionary Questions**

**1️. What is a dictionary in Python?**

A dictionary is a **collection of key-value pairs**. It is **unordered**, **mutable** (can be changed), and does not allow duplicate keys.

**Example:**

student = {"name": "Alice", "age": 25, "grade": "A"}

print(student)

# Output: {'name': 'Alice', 'age': 25, 'grade': 'A'}

**2️. How do you create an empty dictionary?**

An empty dictionary can be created in two ways:

empty\_dict = {} # Using curly braces

empty\_dict2 = dict() # Using dict() function

print(empty\_dict) # Output: {}

print(empty\_dict2) # Output: {}

**3️. How do you create a dictionary with some initial values?**

You can create a dictionary by providing key-value pairs inside {}.

person = {"name": "John", "age": 30, "city": "New York"}

print(person)

# Output: {'name': 'John', 'age': 30, 'city': 'New York'}

**4️. Can dictionary keys be of different data types?**

Yes, dictionary keys can be of different immutable data types like **strings, numbers, or tuples**.

my\_dict = {1: "one", "two": 2, (3, 4): "tuple key"}

print(my\_dict)

✅ **Note:** **Lists and other dictionaries cannot be keys** because they are mutable.

**5️. What happens if you use a duplicate key in a dictionary?**

If a dictionary has duplicate keys, the **last assigned value** will overwrite the previous one.

my\_dict = {"a": 10, "b": 20, "a": 30}

print(my\_dict) # Output: {'a': 30, 'b': 20}

✅ The key "a" appears twice, so Python keeps **only the last assigned value (30).**

**6️. How do you access a value in a dictionary using a key?**

You can access a dictionary value using **dict[key]**.

person = {"name": "Alice", "age": 25}

print(person["name"]) # Output: Alice

❌ **If the key does not exist, an error will occur.**

**7️. What happens if you try to access a key that does not exist?**

Python raises a **KeyError** if the key does not exist.

person = {"name": "Alice"}

print(person["age"]) # ❌ KeyError: 'age'

✅ Use .get() to avoid errors:

print(person.get("age")) # Output: None

**8️. How can you check if a key exists in a dictionary?**

Use the in keyword.

person = {"name": "Alice", "age": 25}

print("name" in person) # Output: True

print("grade" in person) # Output: False

**9️. How do you add a new key-value pair to a dictionary?**

Assign a new key-value pair using dict[key] = value.

person = {"name": "Alice"}

person["age"] = 25

print(person)

# Output: {'name': 'Alice', 'age': 25}

**10. How do you update an existing value in a dictionary?**

Simply reassign the value using the key.

person = {"name": "Alice", "age": 25}

person["age"] = 26 # Update value

print(person)

# Output: {'name': 'Alice', 'age': 26}

**11. How do you remove a key-value pair from a dictionary?**

Use del or pop().

person = {"name": "Alice", "age": 25}

del person["age"]

print(person) # Output: {'name': 'Alice'}

✅ pop() removes the key **and returns its value**:

age = person.pop("age", "Key not found")

print(age) # Output: Key not found

**12. What is the difference between pop() and del in a dictionary?**

| **Method** | **Removes Key?** | **Returns Value?** |
| --- | --- | --- |
| del dict[key] | ✅ Yes | ❌ No |
| dict.pop(key) | ✅ Yes | ✅ Yes |

**13. How do you get all the keys from a dictionary?**

Use .keys().

person = {"name": "Alice", "age": 25}

print(person.keys())

# Output: dict\_keys(['name', 'age'])

**14. How do you get all the values from a dictionary?**

Use .values().

print(person.values())

# Output: dict\_values(['Alice', 25])

**15. How do you get all the key-value pairs from a dictionary?**

Use .items().

print(person.items())

# Output: dict\_items([('name', 'Alice'), ('age', 25)])

**16. What does the get() method do in a dictionary?**

The get() method returns a value **without error if the key is missing**.

print(person.get("name")) # Output: Alice

print(person.get("grade", "Not Found")) # Output: Not Found

**17. What is the difference between get() and direct key access?**

| **Method** | **Returns Value?** | **Raises Error If Key Missing?** |
| --- | --- | --- |
| dict[key] | ✅ Yes | ❌ Yes (KeyError) |
| dict.get(key) | ✅ Yes | ✅ No (returns None or default) |

**18. How can you merge two dictionaries?**

Use .update() or {\*\*dict1, \*\*dict2}.

dict1 = {"a": 1, "b": 2}

dict2 = {"b": 3, "c": 4}

dict1.update(dict2)

print(dict1)

Output: {'a': 1, 'b': 3, 'c': 4}

**19. What is the use of the update() method in dictionaries?**

The update() method merges two dictionaries.  
✅ If keys exist, their values are updated.  
✅ If keys do not exist, they are added.

person = {"name": "Alice"}

person.update({"age": 25, "city": "NY"})

print(person)

Output: {'name': 'Alice', 'age': 25, 'city': 'NY'}

**20. Can dictionary values be of different data types?**

Yes! A dictionary can store **strings, integers, lists, other dictionaries, etc.**

student = {"name": "Alice", "age": 25, "grades": [90, 85, 88]}

print(student)

✅ Output: {'name': 'Alice', 'age': 25, 'grades': [90, 85, 88]}

Great! Here are **detailed explanations** for the next **20 dictionary questions (21-40)** in Python. 😊

**Advanced Beginner Dictionary Questions**

**21. Can a dictionary key be a list? Why or why not?**

❌ **No**, because lists are **mutable** (they can change).

my\_dict = {[1, 2, 3]: "list"}

❌ TypeError: unhashable type: 'list'

✅ Only **immutable** types (e.g., strings, numbers, and tuples) can be dictionary keys.

**22. Can a dictionary key be a tuple?**

✅ **Yes, if the tuple is immutable** (contains only immutable elements).

my\_dict = {(1, 2): "tuple key"}

print(my\_dict[(1, 2)]) # Output: tuple key

**23. How do you iterate over a dictionary using a loop?**

Use a for loop to iterate over **keys**:

person = {"name": "Alice", "age": 25}

for key in person:

print(key, ":", person[key])

Output:

name : Alice

age : 25

✅ To iterate over **key-value pairs**, use .items().

**24. What is the purpose of the items() method in a dictionary?**

It returns **key-value pairs** as tuples.

person = {"name": "Alice", "age": 25}

print(list(person.items()))

Output: [('name', 'Alice'), ('age', 25)]

**25. What does the keys() method do in a dictionary?**

It returns **all dictionary keys**.

print(person.keys()) # Output: dict\_keys(['name', 'age'])

**26. What does the values() method do in a dictionary?**

It returns **all dictionary values**.

print(person.values()) # Output: dict\_values(['Alice', 25])

**27. How do you find the length of a dictionary?**

Use len().

print(len(person)) # Output: 2

**28. How do you remove all items from a dictionary?**

Use .clear().

person.clear()

print(person) # Output: {}

**29. What is the difference between clear() and del dict\_name?**

| **Method** | **Effect** | **Dictionary Exists?** |
| --- | --- | --- |
| clear() | Empties dictionary | ✅ Yes |
| del dict\_name | Deletes dictionary | ❌ No |

person = {"name": "Alice"}

del person

print(person) # ❌ NameError: name 'person' is not defined

**30. How do you copy a dictionary? What is the difference between copy() and direct assignment?**

| **Method** | **Creates New Dictionary?** | **Affects Original?** |
| --- | --- | --- |
| copy() | ✅ Yes | ❌ No |
| = | ❌ No (reference only) | ✅ Yes |

original = {"a": 1, "b": 2}

copy\_dict = original.copy()

original["a"] = 100

print(original) # Output: {'a': 100, 'b': 2}

print(copy\_dict) # Output: {'a': 1, 'b': 2}

**31. What happens if you use a mutable key (like a list) in a dictionary?**

Python **throws an error** because mutable types cannot be keys.

my\_dict = {[1, 2]: "list key"}

❌ TypeError: unhashable type: 'list'

**32. How can you set a default value for a key that does not exist?**

Use .get() or setdefault().

person = {"name": "Alice"}

print(person.get("age", 20)) # Output: 20

**33. What does the setdefault() method do?**

It returns the value if the key exists; otherwise, it sets a **default value**.

person = {"name": "Alice"}

print(person.setdefault("age", 25)) # Output: 25

print(person) # Output: {'name': 'Alice', 'age': 25}

**34. How do you create a dictionary using dict() constructor?**

You can use dict() instead of {}.

my\_dict = dict(name="Alice", age=25)

print(my\_dict) # Output: {'name': 'Alice', 'age': 25}

**35. How do you create a dictionary using dictionary comprehension?**

squares = {x: x\*\*2 for x in range(5)}

print(squares)

# Output: {0: 0, 1: 1, 2: 4, 3: 9, 4: 16}

**36. How do you sort a dictionary by keys?**

Use sorted().

my\_dict = {"b": 2, "a": 1, "c": 3}

sorted\_dict = dict(sorted(my\_dict.items()))

print(sorted\_dict) # Output: {'a': 1, 'b': 2, 'c': 3}

**37. How do you sort a dictionary by values?**

Sort using a **lambda function**.

sorted\_dict = dict(sorted(my\_dict.items(), key=lambda item: item[1]))

print(sorted\_dict) # Output: {'a': 1, 'b': 2, 'c': 3}

**38. What is the difference between defaultdict and a normal dictionary?**

A **defaultdict** automatically provides a default value for missing keys.

from collections import defaultdict

dd = defaultdict(int) # Default value is 0

print(dd["missing"]) # Output: 0

39. **What is the difference between OrderedDict and a normal dictionary?**

Before Python 3.7, dictionaries **did not maintain order**. OrderedDict was used for this.

from collections import OrderedDict

ordered = OrderedDict([("a", 1), ("b", 2)])

print(ordered) # Output: OrderedDict([('a', 1), ('b', 2)])

**40. How do you count the frequency of elements in a list using a dictionary?**

Use a dictionary to store counts.

numbers = [1, 2, 2, 3, 3, 3]

freq = {}

for num in numbers:

freq[num] = freq.get(num, 0) + 1

print(freq) # Output: {1: 1, 2: 2, 3: 3}

✅ **Alternate Method:** Using collections.Counter():

from collections import Counter

print(Counter(numbers)) # Output: Counter({3: 3, 2: 2, 1: 1})