operators dictionary list tuple set if else explain all concept one by one with all interview questions

I'll explain all Python concepts one by one with comprehensive explanations and interview questions.

## 1. Operators

## Explanation

Operators are special symbols that perform operations on variables and values.

**Types of Operators:**

- Arithmetic: `+`, `-`, `*`, `/`, `//`, `%`, `**`
- Comparison: `==`, `!=`, `>`, `<`, `>=`, `<=`
- Logical: `and`, `or`, `not`
- Assignment: `=`, `+=`, `-=`, `*=`, `/=`
- Identity: `is`, `is not`
- Membership: `in`, `not in`
- Bitwise: `&`, `|`, `^`, `~`, `<<`, `>>`

## Interview Questions

### Q1: What's the difference between `==` and `is`?

python

```python
a = [1, 2, 3]
b = [1, 2, 3]
c = a

print(a == b)  # True - same values
print(a is b)  # False - different objects
print(a is c)  # True - same object
```

### Q2: Explain floor division vs regular division

python

```python
print(7 / 2)   # 3.5 - regular division
print(7 // 2)  # 3 - floor division
print(-7 // 2) # -4 - rounds towards -∞
```

### Q3: Operator precedence example

python

```python
result = 2 + 3 * 4 ** 2  # 2 + (3 * (4 ** 2)) = 50
```

## 2. Dictionary

### Explanation

Dictionary is an unordered collection of key-value pairs.

**Characteristics:**

- Mutable
- Keys must be hashable (immutable)
- No duplicate keys
- O(1) average time complexity for operations

### Interview Questions

#### Q1: Different ways to create a dictionary

python

```python
# Method 1: Literal
dict1 = {'a': 1, 'b': 2}

# Method 2: dict() constructor
dict2 = dict(a=1, b=2)

# Method 3: From list of tuples
dict3 = dict([('a', 1), ('b', 2)])
```

```python
# Method 4: Dictionary comprehension
dict4 = {x: x**2 for x in range(5)}
```

### Q2: Common dictionary methods

python

```python
my_dict = {'a': 1, 'b': 2, 'c': 3}

# get() - returns value or default
print(my_dict.get('d', 0))  # 0

# setdefault() - sets default if key doesn't exist
my_dict.setdefault('d', 4)

# update() - merges dictionaries
my_dict.update({'e': 5})

# pop() - removes and returns value
value = my_dict.pop('a')
```

### Q3: Dictionary comprehension with condition

python

```python
# Create dict with even squares
squares = {x: x**2 for x in range(10) if x % 2 == 0}
print(squares)  # {0: 0, 2: 4, 4: 16, 6: 36, 8: 64}
```

## 3. List

## Explanation

List is an ordered, mutable collection of items.

### Characteristics:

- Ordered
- Mutable
- Allows duplicates
- Can contain mixed data types

## Interview Questions

### Q1: List methods and operations

```python
my_list = [1, 2, 3]

# Adding elements
my_list.append(4)        # [1, 2, 3, 4]
my_list.extend([5, 6])   # [1, 2, 3, 4, 5, 6]
my_list.insert(1, 1.5)   # [1, 1.5, 2, 3, 4, 5, 6]

# Removing elements
my_list.remove(1.5)      # Remove first occurrence
popped = my_list.pop()   # Remove and return last element
```

### Q2: List slicing

```python
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

print(numbers[2:7])      # [2, 3, 4, 5, 6]
print(numbers[::2])      # [0, 2, 4, 6, 8] - step 2
print(numbers[::-1])     # Reverse list
print(numbers[-3:])      # Last 3 elements
```

### Q3: List comprehension vs for loop

```python
# Traditional way
squares = []
for i in range(5):
    squares.append(i**2)

# List comprehension (more Pythonic)
squares = [i**2 for i in range(5)]

# With condition
even_squares = [i**2 for i in range(10) if i % 2 == 0]
```

## 4. Tuple

## Explanation

Tuple is an ordered, immutable collection of items.

### Characteristics:

- Immutable
- Ordered
- Allows duplicates
- Faster than lists
- Can be used as dictionary keys

## Interview Questions

### Q1: When to use tuples vs lists?

```python
# Use tuples for heterogeneous data (like structs)
person = ('John', 25, 'Engineer')

# Use lists for homogeneous data
numbers = [1, 2, 3, 4, 5]

# Tuples as dictionary keys
locations = {
    (35.6895, 139.6917): "Tokyo",
    (40.7128, -74.0060): "New York"
}
```

### Q2: Tuple packing and unpacking

```python
# Packing
coordinates = 4, 5, 6

# Unpacking
x, y, z = coordinates

# Extended unpacking
first, *middle, last = (1, 2, 3, 4, 5)
print(middle)  # [2, 3, 4]
```

### Q3: Single element tuple

```python
# Wrong way - this is an integer
not_tuple = (5)

# Correct way - comma makes it a tuple
is_tuple = (5,)
print(type(is_tuple))  # <class 'tuple'>
```

## 5. Set

## Explanation

Set is an unordered collection of unique, hashable items.

### Characteristics:

- Unordered
- No duplicates
- Mutable
- Mathematical set operations

## Interview Questions

### Q1: Set operations

```python
A = {1, 2, 3, 4}
B = {3, 4, 5, 6}

print(A | B)  # Union: {1, 2, 3, 4, 5, 6}
print(A & B)  # Intersection: {3, 4}
print(A - B)  # Difference: {1, 2}
print(A ^ B)  # Symmetric difference: {1, 2, 5, 6}
```

### Q2: Remove duplicates from list using set

python

```
numbers = [1, 2, 2, 3, 4, 4, 5]
unique_numbers = list(set(numbers))
print(unique_numbers)  # [1, 2, 3, 4, 5]
```

### Q3: Set comprehension

```python
# Set of squares
squares = {x**2 for x in range(10)}

# Set from string (unique characters)
unique_chars = {char for char in "hello world"}
print(unique_chars)  # {'h', 'e', 'l', 'o', ' ', 'w', 'r', 'd'}
```

## 6. if-else Statements

### Explanation

Conditional statements for decision making in code.

#### Types:

- `if`
- `if-else`
- `if-elif-else`
- Nested if
- Ternary operator

### Interview Questions

#### Q1: Different if-else patterns

```python
# Basic if-else
age = 18
if age >= 18:
    print("Adult")
else:
    print("Minor")
```

```python
# if-elif-else chain
score = 85
if score >= 90:
    grade = 'A'
elif score >= 80:
    grade = 'B'
elif score >= 70:
    grade = 'C'
else:
    grade = 'F'

# Ternary operator
status = "Adult" if age >= 18 else "Minor"
```

### Q2: Chained comparisons

python

```python
x = 5
# Instead of: if x > 0 and x < 10:
if 0 < x < 10:
    print("x is between 0 and 10")
```

### Q3: Using if with data structures

python

```python
# Check if list is not empty
my_list = [1, 2, 3]
if my_list:  # Pythonic way
    print("List has items")

# Check if key exists in dict
my_dict = {'a': 1}
if 'a' in my_dict:
    print("Key exists")

# Multiple conditions
if my_list and len(my_list) > 2 and my_list[0] == 1:
    print("Complex condition met")
```

## Advanced Interview Questions

### Q1: Memory efficiency - tuples vs lists

```python
import sys

list_obj = [1, 2, 3, 4, 5]
tuple_obj = (1, 2, 3, 4, 5)

print(sys.getsizeof(list_obj))   # Larger
print(sys.getsizeof(tuple_obj))  # Smaller
```

### Q2: Shallow vs deep copy

```python
import copy

# Shallow copy
original_list = [[1, 2], [3, 4]]
shallow_copy = original_list.copy()
shallow_copy[0][0] = 99
print(original_list)  # [[99, 2], [3, 4]] - affected!

# Deep copy
original_list = [[1, 2], [3, 4]]
deep_copy = copy.deepcopy(original_list)
deep_copy[0][0] = 99
print(original_list)  # [[1, 2], [3, 4]] - not affected
```

### Q3: Dictionary merging (Python 3.9+)

```python
dict1 = {'a': 1, 'b': 2}
dict2 = {'b': 3, 'c': 4}

# Python 3.9+ - union operator
merged = dict1 | dict2
print(merged)  # {'a': 1, 'b': 3, 'c': 4}

# Traditional way
merged = {**dict1, **dict2}
```

### Q4: Most common element in list

python

```python
from collections import Counter

numbers = [1, 2, 2, 3, 3, 3, 4, 4, 4, 4]
counter = Counter(numbers)
most_common = counter.most_common(1)
print(most_common)  # [(4, 4)]
```

### Q5: List vs Tuple performance

python

```python
import timeit

# Creation time
list_time = timeit.timeit('x = [1, 2, 3, 4, 5]', number=1000000)
tuple_time = timeit.timeit('x = (1, 2, 3, 4, 5)', number=1000000)

print(f"List creation: {list_time}")
print(f"Tuple creation: {tuple_time}")
```

These concepts form the foundation of Python programming. Understanding them thoroughly will help you in both interviews and real-world programming scenarios. Practice implementing these concepts with different examples to strengthen your