📁

python tutorial

Operation Type	Description	Operators Used
Mathematical Calculations	Used for arithmetic operations	`+`, `-`, `*`, `/`, `//`, `%`, `**`
Comparison	Used to compare values	`==`, `!=`, `>`, `<`, `>=`, `<=`
Logical Operations	Used to combine conditional statements	`and`, `or`, `not`
Assignment	Used to assign values to variables	`=`, `+=`, `-=`, `*=`, `/=`, `//=`, `%=`, `**=`, `&=`, `
Bitwise Operations	Used for manipulating binary representations of integers	`&`, `
Identity Checks	Used to compare the memory locations of objects	`is`, `is not`
Membership Tests	Used to check if a value is present in a sequence	`in`, `not in`
Memory Management	Used for memory allocation and reference counting (implicit in Python)	Not directly accessible, handled by Python interpreter
Decision Making	Used in conditional statements (often with comparison and logical operators)	Typically uses comparison and logical operators

Python Operators: Types and Examples

1. Arithmetic Operators

Used for mathematical operations

Addition (+)

```
x = 5 + 3
print(f"Addition: 5 + 3 = {x}")  # Output: 8
```

Subtraction (-)

```
y = 10 - 4
print(f"Subtraction: 10 - 4 = {y}")  # Output: 6
```

Multiplication (*)

```
z = 3 ∗ 7
print(f"Multiplication: 3 ∗ 7 = {z}")  # Output: 21
```

Division (/)

```
a = 20 / 4
print(f"Division: 20 / 4 = {a}")  # Output: 5.0
```

Floor Division (//)

```
b = 17 // 5
print(f"Floor Division: 17 // 5 = {b}")  # Output: 3
```

Modulus (%)

```
c = 17 % 5
print(f"Modulus: 17 % 5 = {c}")  # Output: 2
```

Exponentiation (**)

```
d = 2 ** 3
print(f"Exponentiation: 2 ** 3 = {d}")  # Output: 8
```

2. Comparison Operators

Used to compare values

Equal to (==)

```
print(f"Equal to: 5 == 5 is {5 == 5}")  # Output: True
```

Not equal to (!=)

```
print(f"Not equal to: 5 != 6 is {5 != 6}")  # Output: True
```

Greater than (>)

```
print(f"Greater than: 7 > 3 is {7 > 3}")  # Output: True
```

Less than (<)

```
print(f"Less than: 2 < 8 is {2 < 8}")  # Output: True
```

Greater than or equal to (>=)

```
print(f"Greater than or equal to: 5 >= 5 is {5 >= 5}")  # Output: True
```

Less than or equal to (<=)

```
print(f"Less than or equal to: 4 <= 3 is {4 <= 3}")  # Output: False
```

3. Logical Operators

Used to combine conditional statements

and

```
x = 5
print(f"x > 3 and x < 10 is {x > 3 and x < 10}")  # Output: True
```

or

```
y = 12
print(f"y < 5 or y > 10 is {y < 5 or y > 10}")  # Output: True
```

not

```
z = False
print(f"not z is {not z}")  # Output: True
```

4. Assignment Operators

Used to assign values to variables

Simple assignment (=)

```
a = 5
print(f"a = 5: {a}")  # Output: 5
```

Add and assign (+=)

```
b = 10
b += 3  # Equivalent to b = b + 3
```

```
print(f"b += 3: {b}")  # Output: 13
```

Subtract and assign (-=)

```
c = 8
c -= 2  # Equivalent to c = c - 2
print(f"c -= 2: {c}")  # Output: 6
```

Multiply and assign (*=)

```
d = 4
d *= 3  # Equivalent to d = d * 3
print(f"d *= 3: {d}")  # Output: 12
```

Divide and assign (/=)

```
e = 15
e /= 3  # Equivalent to e = e / 3
print(f"e /= 3: {e}")  # Output: 5.0
```

Floor divide and assign (//=)

```
f = 17
f //= 5  # Equivalent to f = f // 5
print(f"f //= 5: {f}")  # Output: 3
```

Modulus and assign (%=)

```
g = 18
g %= 5  # Equivalent to g = g % 5
print(f"g %= 5: {g}")  # Output: 3
```

Exponent and assign (**=)

```
h = 2
h **= 3  # Equivalent to h = h ** 3
print(f"h **= 3: {h}")  # Output: 8
```

5. Bitwise Operators

Used to perform bitwise calculations on integers

Bitwise AND (&)

```
x = 5  # 101 in binary
y = 3  # 011 in binary
print(f"Bitwise AND: 5 & 3 = {x & y}")  # Output: 1 (001 in binary)
```

Bitwise OR (|)

```
print(f"Bitwise OR: 5 | 3 = {x | y}")  # Output: 7 (111 in binary)
```

Bitwise XOR (^)

```
print(f"Bitwise XOR: 5 ^ 3 = {x ^ y}")  # Output: 6 (110 in binary)
```

Bitwise NOT (~)

```
print(f"Bitwise NOT: ~5 = {~x}")  # Output: -6
```

Left shift (<<)

```
print(f"Left shift: 5 << 1 = {x << 1}")  # Output: 10 (1010 in binary)
```

Right shift (>>)

```
print(f"Right shift: 5 >> 1 = {x >> 1}")  # Output: 2 (10 in binary)
```

6. Identity Operators

Used to compare the memory locations of two objects

is

```
a = [1, 2, 3]
b = [1, 2, 3]
c = a
print(f"a is b: {a is b}")  # Output: False
print(f"a is c: {a is c}")  # Output: True
```

is not

```
print(f"a is not b: {a is not b}")  # Output: True
print(f"a is not c: {a is not c}")  # Output: False
```

7. Membership Operators

Used to test if a sequence is present in an object

in

```
fruits = ["apple", "banana", "cherry"]
print(f"'banana' in fruits: {'banana' in fruits}")  # Output: True
print(f"'grape' in fruits: {'grape' in fruits}")  # Output: False
```

not in

```
print(f"'orange' not in fruits: {'orange' not in fruits}")  # Output: True
print(f"'apple' not in fruits: {'apple' not in fruits}")  # Output: False
```

Condtional statements :

Loops :

String :

string interview questions

list:

Loops in python :

misllaneous topic:

Tuples:

Sets :

Dictionary :