Week 2 Day 1: Introduction to Operators

Operators are special symbols or keywords that tell Python to perform specific operations on one or more values (called operands).

Python has several types of operators:

1. Arithmetic Operators
2. Assignment Operators
3. Comparison Operators
4. Logical Operators
5. Identity Operators
6. Membership Operators
7. Bitwise Operators (Advanced)
8. Operators Used on Sequence Data Types

1. Arithmetic Operators

Used to perform mathematical operations.

Operator	Description	Example	Output
+	Addition	5 + 2	7
-	Subtraction	5 - 2	3
*	Multiplication	5 * 2	10
/	Division (float)	5 / 2	2.5
//	Floor Division	5 // 2	2
%	Modulus (remainder)	5 % 2	1
**	Exponentiation (power)	2 ** 3	8

2. Assignment Operators

Used to assign or update the value of a variable.

Operator	Description	Example	Equivalent To
=	Assign value	x = 5	—

+=	Add and assign	x += 2	x = x + 2
-=	Subtract and assign	x -= 3	x = x - 3
*=	Multiply and assign	x *= 2	x = x * 2
/=	Divide and assign	x /= 2	x = x / 2
//=	Floor divide and assign	x //= 2	x = x // 2
%=	Modulus and assign	x %= 2	x = x % 2
**=	Exponent and assign	x **= 3	x = x ** 3

3. Comparison (Relational) Operators

Used to compare two values and return True or False based on the comparison.

Operator	Description	Example	Output
==	Equal to	5 == 5	True
!=	Not equal to	5 != 3	True
>	Greater than	5 > 3	True
<	Less than	5 < 3	False
>=	Greater than or equal to	5 >= 5	True
<=	Less than or equal to	3 <= 5	True

4. Logical Operators

Used to combine multiple conditions.

Operator	Description	Example	Output
and	True if both are true	5 > 3 and 3 > 1	True
or	True if at least one is true	5 > 3 or 3 < 1	True
not	Reverses the result	not(5 > 3)	False

5. Identity Operators

Used to check if two variables refer to the same object (same memory location).

Operator	Description	Example	Output
is	True if same object	x is y	True/False
is not	True if not same object	x is not y	True/False

6. Membership Operators

Used to test if a value is in or not in a sequence (like list, string, or tuple).

Operator	Description	Example	Output
in	True if value exists	'a' in 'cat'	True
not in	True if value does not exist	3 not in [1, 2, 4]	True

7. Bitwise Operators (Advanced)

Used to perform operations on the binary representation of numbers.

Operator	Description	What It Does	Example	Output
&	Bitwise AND	Returns 1 if both bits are 1	5 & 3 → 101 & 011	1
|	Bitwise OR	Returns 1 if either bit is 1	5 | 3	7
^	Bitwise XOR	Returns 1 if bits are different	5 ^ 3	6
~	Bitwise NOT	Flips all the bits	~5	-6
<<	Left Shift	Shifts bits left (multiplies by 2^n)	5 << 1	10
>>	Right Shift	Shifts bits right (divides by 2^n)	5 >> 1	2

8. Operators Used on Sequence Data Types

Used to perform operations on sequence data types like strings, lists, and tuples.

Operator	Description	Example	Output
+	Concatenation	'Hello' + 'World'	'HelloWorld'
*	Repetition	'Hi' * 3	'HiHiHi'
in	Membership test	'a' in 'apple'	True
not in	Non-membership test	'x' not in 'apple'	True
len()	Length of sequence	len([1,2,3])	3
min()	Smallest item in sequence	min([2,4,1,5])	1
max()	Largest item in sequence	max([2,4,1,5])	5
index()	Finds index of value	[10,20,30].index(20)	1
count()	Counts occurrences of value	[1,2,2,3].count(2)	2

Additional Operators in Python

This document covers additional operators and operator-like expressions in Python that were not explicitly covered in the main lesson on Python operators. These include unary operators, conditional (ternary) operators, slicing, type conversion, and functions from the `operator` module.

1. Unary Operators

Unary operators operate on a single operand to produce a new value.

Operator	Description	Example	Output
+	Unary plus (indicates positive value)	+5	5
-	Unary minus (negates the number)	-5	-5
~	Bitwise NOT (flips all bits)	~5	-6

2. Conditional (Ternary) Operator

The conditional operator allows for a compact form of an if-else statement. It is used to choose one of two values based on a condition.

Operator Form	Description	Example	Output
x if condition else y	Returns x if the condition is True, otherwise y	5 if 3 > 2 else 10	5

3. Slicing Operator

The slicing operator is used to extract a portion (subsequence) from a sequence data type like a string, list, or tuple.

Operator	Description	Example	Output
[:]	Returns a full copy of the sequence	nums[:]	[1, 2, 3, 4, 5]
[start:stop]	Returns elements from index start to stop-1	nums[1:4]	[2, 3, 4]
[start:stop:step]	Returns elements from start to stop-1, skipping by step	nums[::2]	[1, 3, 5]

4. Type Conversion Functions (Type Operators)

These are built-in functions that act as operators to convert between data types.

Function	Description	Example	Output
int()	Converts a value to an integer	int(5.8)	5
float()	Converts a value to a float	float(5)	5.0
str()	Converts a value to a string	str(123)	'123'
list()	Converts to a list	list((1,2,3))	[1, 2, 3]
tuple()	Converts to a tuple	tuple([1,2,3])	(1, 2, 3)
set()	Converts to a set	set([1,2,2,3])	{1, 2, 3}
dict()	Creates a dictionary	dict(a=1, b=2)	{'a': 1, 'b': 2}

bool()	Converts a value to True or False	bool(0)	False
complex()	Creates a complex number	complex(2,3)	(2+3j)

5. Chained Comparison Operators

Python allows chaining of comparison operators to test multiple conditions in one expression.

Form	Description	Example	Output
x < y < z	Checks if x is less than y and y is less than z	5 < 10 < 20	True
a == b != c	Checks if a equals b and b is not equal to c	5 == 5 != 3	True

6. Operator Module Functions

Python provides the `operator` module, which contains function equivalents of many built-in operators. These are useful for functional programming and dynamic expressions.

Function	Description	Example	Output
operator.add(a, b)	Performs addition	operator.add(2, 3)	5
operator.sub(a, b)	Performs subtraction	operator.sub(5, 2)	3
operator.mul(a, b)	Performs multiplication	operator.mul(3, 4)	12
operator.truediv(a, b)	Performs true division	operator.truediv(5, 2)	2.5
operator.floordiv(a, b)	Performs floor division	operator.floordiv(5, 2)	2
operator.mod(a, b)	Returns modulus	operator.mod(5, 2)	1
operator.pow(a, b)	Performs exponentiation	operator.pow(2, 3)	8
operator.eq(a, b)	Checks equality	operator.eq(3, 3)	True
operator.ne(a, b)	Checks inequality	operator.ne(3, 2)	True
operator.contains(seq, item)	Checks membership	operator.contains([1,2,3], 2)	True

Additional Operators in Python

This document covers additional operators and operator-like expressions in Python that were not explicitly covered in the main lesson on Python operators. These include unary operators, conditional (ternary) operators, slicing, type conversion, and functions from the `operator` module.

1. Unary Operators

Unary operators operate on a single operand to produce a new value.

Operator	Description	Example	Output
+	Unary plus (indicates positive value)	+5	5
-	Unary minus (negates the number)	-5	-5
~	Bitwise NOT (flips all bits)	~5	-6

2. Conditional (Ternary) Operator

The conditional operator allows for a compact form of an if-else statement. It is used to choose one of two values based on a condition.

Operator Form	Description	Example	Output
x if condition else y	Returns x if the condition is True, otherwise y	5 if 3 > 2 else 10	5

3. Slicing Operator

The slicing operator is used to extract a portion (subsequence) from a sequence data type like a string, list, or tuple.

Operator	Description	Example	Output
[:]	Returns a full copy of the sequence	nums[:]	[1, 2, 3, 4, 5]
[start:stop]	Returns elements from index start to stop-1	nums[1:4]	[2, 3, 4]

[start:stop:step]	Returns elements from start to stop-1, skipping by step	nums[::2]	[1, 3, 5]

4. Type Conversion Functions (Type Operators)

These are built-in functions that act as operators to convert between data types.

Function	Description	Example	Output
int()	Converts a value to an integer	int(5.8)	5
float()	Converts a value to a float	float(5)	5.0
str()	Converts a value to a string	str(123)	'123'
list()	Converts to a list	list((1,2,3))	[1, 2, 3]
tuple()	Converts to a tuple	tuple([1,2,3])	(1, 2, 3)
set()	Converts to a set	set([1,2,2,3])	{1, 2, 3}
dict()	Creates a dictionary	dict(a=1, b=2)	{'a': 1, 'b': 2}
bool()	Converts a value to True or False	bool(0)	False
complex()	Creates a complex number	complex(2,3)	(2+3j)

5. Chained Comparison Operators

Python allows chaining of comparison operators to test multiple conditions in one expression.

Form	Description	Example	Output
x < y < z	Checks if x is less than y and y is less than z	5 < 10 < 20	True
a == b != c	Checks if a equals b and b is not equal to c	5 == 5 != 3	True

6. Operator Module Functions

Python provides the `operator` module, which contains function equivalents of many built-in operators. These are useful for functional programming and dynamic expressions.

Function	Description	Example	Output
operator.add(a, b)	Performs addition	operator.add(2, 3)	5
operator.sub(a, b)	Performs subtraction	operator.sub(5, 2)	3
operator.mul(a, b)	Performs multiplication	operator.mul(3, 4)	12
operator.truediv(a, b)	Performs true division	operator.truediv(5, 2)	2.5
operator.floordiv(a, b)	Performs floor division	operator.floordiv(5, 2)	2
operator.mod(a, b)	Returns modulus	operator.mod(5, 2)	1
operator.pow(a, b)	Performs exponentiation	operator.pow(2, 3)	8
operator.eq(a, b)	Checks equality	operator.eq(3, 3)	True
operator.ne(a, b)	Checks inequality	operator.ne(3, 2)	True
operator.contains(seq, item)	Checks membership	operator.contains([1,2,3], 2)	True