Python Operators

Review arithmetic, comparison, assignment, and other Python operators with this short e-book!

Contents

- 1: Arithmetic operators
- 4: Comparison operators
- 5: Assignment operators
- 6: Logic operators
- 7: Identity operators
- 8: Membership operators

Addition

Can be used to add numbers (integers or floats) or join (concatenate) strings.

```
44 + 19.5 # 63.5
```

```
"My name is " + "James" # My name is James
```

You cannot add strings and numbers together!

```
"The correct number is" + 5
# TypeError: can only concatenate str (not "int") to str
```

Subtraction

Can be used mathematically, between integers and floats:

```
93.1 - 50 # 43.1
```

You can also subtract sets to return elements of set_1 that aren't in set_2:

```
set_1 = {3, 6, 9, 7, 5}
set_2 = {3, 6, 9, 95}
set_difference = set_1 - set_2 # {5, 7}
```

Multiplication

Has the typical arithmetic use.

Multiplying most data types usually duplicates their contents.

```
8 * 5 # 40
```

```
"Hello " * 2 # Hello Hello
[100] * 2 # [100, 100]
(100, 1) * 2 # (100, 1, 100, 1)
```

Division

Can be true division, which always results in a float.

```
30 / 2 # 15.0
30.5 / 2 # 15.5
```

Or floor division, which always rounds down to the nearest whole number.

```
30.5 // 2 # 15.0
```

Modulo

- Based on Euclidean division (see ref #2 below).
- Start with a dividend and a divisor.
- Used to get the remainder of the division between both operands.

```
10 % 3 # 1
9 % 3 # 0
8 % 3 # 2
```

Exponentiation

Raises the first operand to the power of the second operand.

```
2 ** 3 # 2 * 2 * 2 == 8
```

Extra Resources

- 1. How do mathematics work in Python?
- 2. Python's modulo operator and floor division
- 3. Python's divmod function

Comparison operators

- Equal

Used to compare two values. They can be numbers, strings, or anything else.

```
4 == 4  # True
4 == 5  # False
"Hello" == "Hello"  # True
"Hello" == "HELLO"  # False
(100, 50) == 50  # False
(100, 50) == (500, 219)  # False
```

— Not equal

The opposite of ==.

```
4 != 4 # False
4 != 5 # True
```

— Greater than Less than

```
10 > 10  # False
11 > 10  # True
10 < 10  # False
9 < 10  # True
```

- Greater than or equal to Less than or equal to

```
10 >= 10 # True
10 <= 10 # True
```

Assignment operators

Operator	Example	Explanation
=	var = 2	Makes var equal to the value 2.
+=	var += 2	Adds 2 to var.
-=	var -= 2	Subtracts 2 from var.
*=	var *= 2	Multiplies var by 2.
**=	var **= 2	Elevates var to the power of 2.
/=	var /= 2	Divides var by 2.
//=	var //= 2	Divides var by 2 using floor division.
% =	var % = 2	Calculates the remainder of var / 2.

Extra Resources

Logic Operators

- And

Returns the first value if it evaluates to False, otherwise it returns the second value.

Statement	Result
True and True	True
True and False	False
False and False	False
False and True	False

-Or

Returns the first value if it evaluates to **True**, otherwise it returns the second value.

Statement	Result
True or True	True
True or False	True
False or False	False
False or True	True

- Not

Returns True if the operand evaluates to False, or False if it evaluates to True.

Statement	Result
not True	False
not False	True

Identity operators

– is

Returns True if the operands have the same identity (i.e. are exactly the same value).

```
users = ['James', 'Charlie', 'Ana']
print(id(users)) # 2425142160952

users_copy = users
print(id(users_copy)) # 2425142160952

print(users_copy is users) # True

users_copy = ['James', 'Charlie', 'Ana']
print(id(users_copy)) # 2425142558551

print(users_copy is users) # False
```

Above, users_copy is users returns False because although the two lists have the same values inside them, they are not the same list.

- is not

Returns True if the operands don't have the same identity.

```
users = ['James', 'Charlie', 'Ana']
print(id(users)) # 2425142160952

users_copy = users
print(id(users_copy)) # 2425142160952

print(users_copy is users) # False

users_copy = ['James', 'Charlie', 'Ana']
print(id(users_copy)) # 2425142558551

print(users_copy is users) # True
```

Extra Resources

1. Logical comparisons in Python: and & or

Membership operators

- in

Returns True if an element exists inside an object.

```
users = ['James', 'Charlie', 'Ana']
print("Johnny" in users) # False
```

not in

Returns True if an element does not exist inside an object.

```
users = ['James', 'Charlie', 'Ana']
print("Johnny" not in users) # True
```