
DAY
04

Where can we run Python code?

Online in browser
(without downloading any software)

IDE
(Integrated development Environment)

Python Interpreter
(By using some text editor)

Some Tools To Run PYTHON Code

```
>>>print('Welcome To Day 03")
Welcome To Day 03
```

Comments.....

this is not executed

Comments can be used to explain Python code.

Comments can be used to make the code more readable.

Comments can be used to prevent execution when testing code.

Creating a Comment

Comments starts with a #, and Python will ignore them:

Example

```
#This is a comment
print("Hello, World!")
```

OOPS !!

Python is an object oriented programming language

An object is simply a collection of data (variables) and methods (functions) that act on those data

Python Objects and Classes

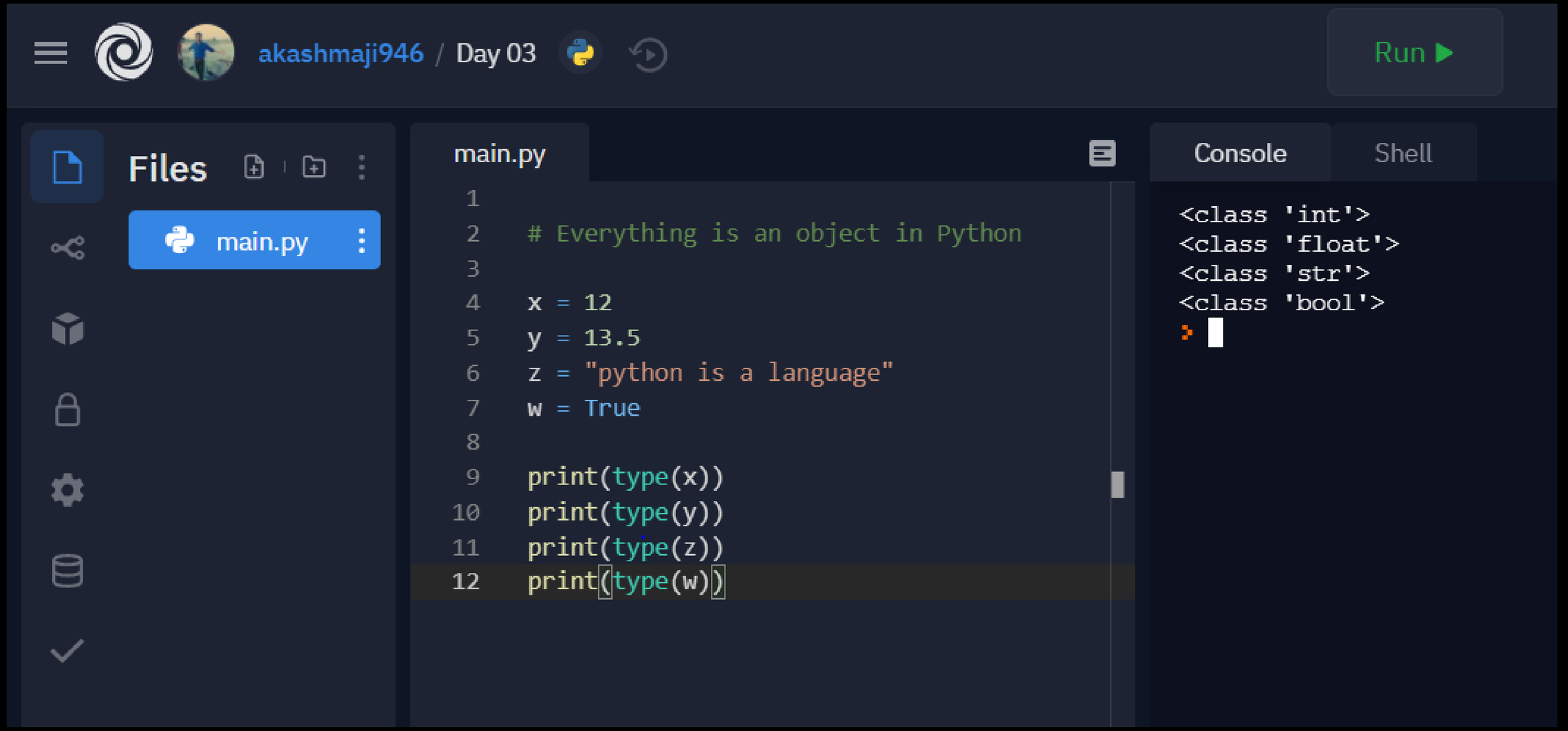
Everything is an object in Python

Python is an object oriented programming language.

Almost everything in Python is an object, with its properties and methods.

A Class is like an object constructor, or a "blueprint" for creating objects.

Classes are everywhere



WAIT ?

We are not able to understand

We will revisit these things later !

Use case

I want to print my name 10 times

How can I do this without using print() 10 times?

Use case

I want to print my name 10 times

How can I do this without using print() 10 times?

I want to check a number is even or odd

This seems tricky.

We need to discuss some fundamental concepts.

Computer Science Fundamentals

- **Operators**
- **Conditional Statements**
- **Iterative Statements**

We will learn them one by one

Operators & Expressions

Operators

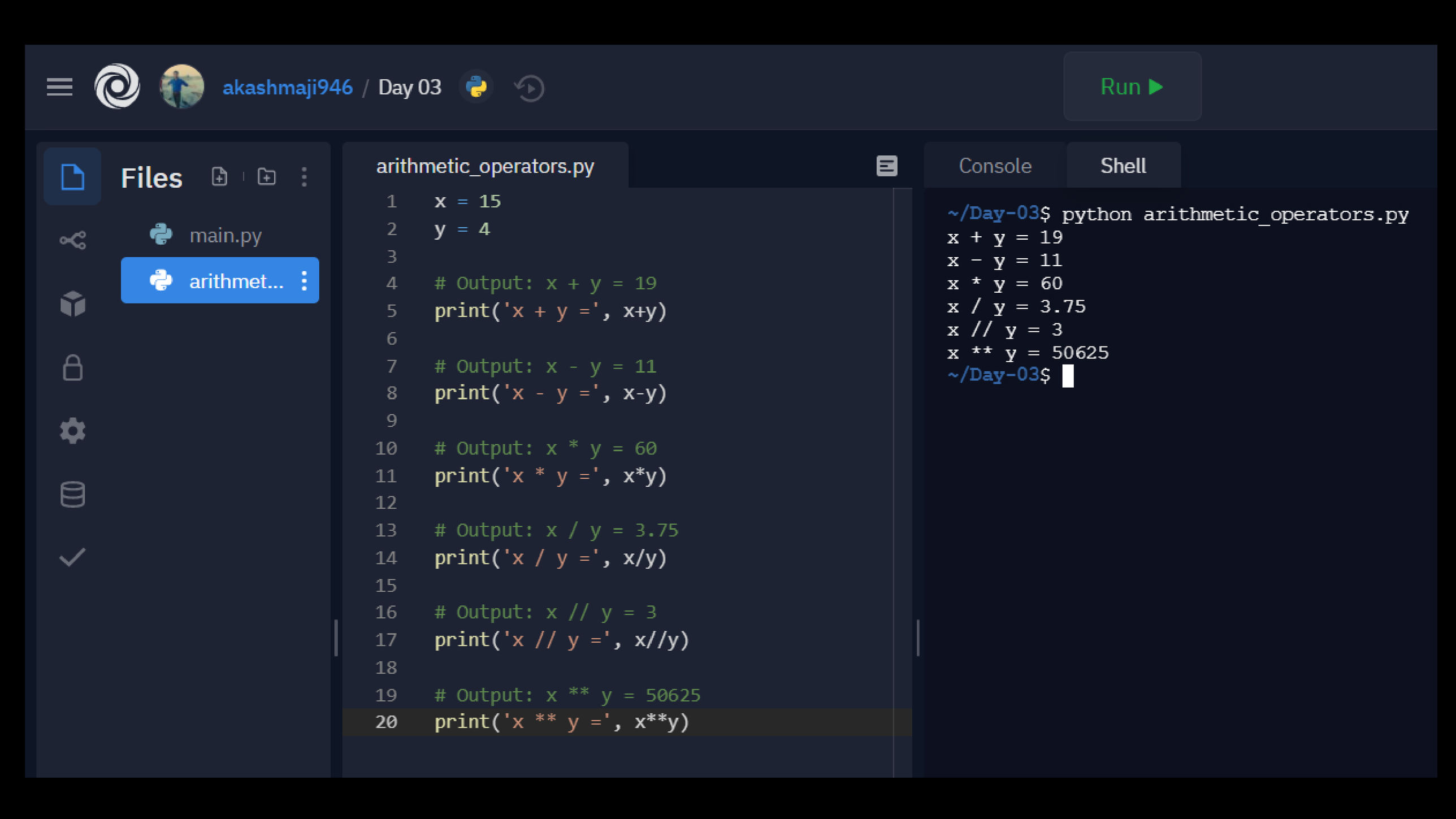
used for performing some operation
between **one or more** operands

ARITHMETIC OPERATORS

Arithmetic operators

Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication, etc.

Operator	Meaning	Example
+	Add two operands or unary plus	x + y+ 2
-	Subtract right operand from the left or unary minus	x - y- 2
*	Multiply two operands	x * y
/	Divide left operand by the right one (always results into float)	x / y
%	Modulus - remainder of the division of left operand by the right	x % y (remainder of x/y)
//	Floor division - division that results into whole number adjusted to the left in the number line	x // y
**	Exponent - left operand raised to the power of right	x**y (x to the power y)

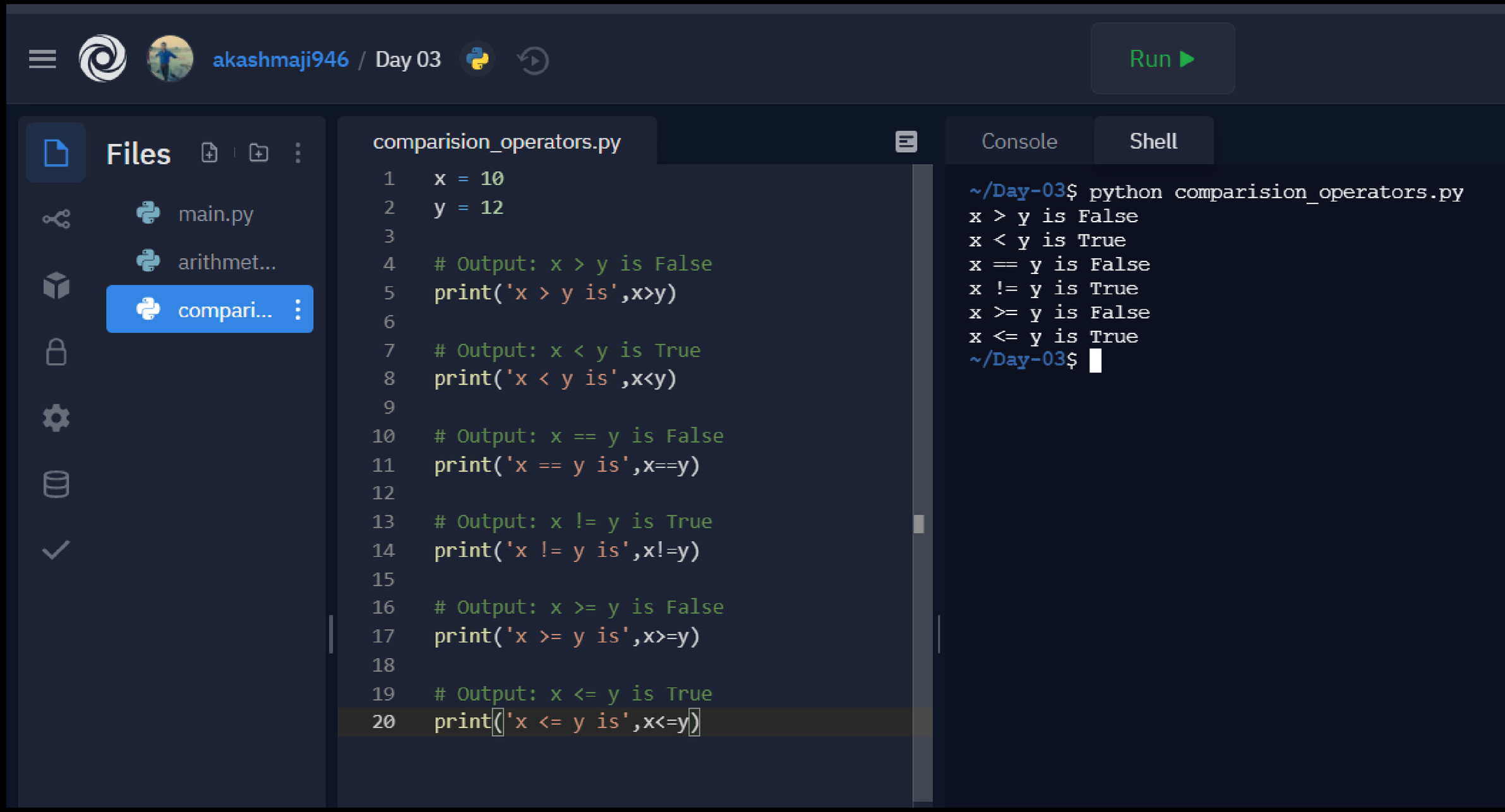

akashmaji946 / Day 03
Run
Files
main.py
arithmet...
arithmetic_operators.py
1 x = 15
2 y = 4
3
4 # Output: x + y = 19
5 print('x + y =', x+y)
6
7 # Output: x - y = 11
8 print('x - y =', x-y)
9
10 # Output: x * y = 60
11 print('x * y =', x*y)
12
13 # Output: x / y = 3.75
14 print('x / y =', x/y)
15
16 # Output: x // y = 3
17 print('x // y =', x//y)
18
19 # Output: x ** y = 50625
20 print('x ** y =', x**y)
Console
Shell
~/Day-03$ python arithmetic_operators.py
x + y = 19
x - y = 11
x * y = 60
x / y = 3.75
x // y = 3
x ** y = 50625
~/Day-03$

Comparison operators

Comparison operators

Comparison operators are used to compare values. It returns either True or False according to the condition.

Operator	Meaning	Example
>	Greater than - True if left operand is greater than the right	x > y
<	Less than - True if left operand is less than the right	x < y
==	Equal to - True if both operands are equal	x == y
!=	Not equal to - True if operands are not equal	x != y
>=	Greater than or equal to - True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to - True if left operand is less than or equal to the right	x <= y


akashmaji946 / Day 03
Run
Files
main.py
arithmet...
compari...
comparision_operators.py
1 x = 10
2 y = 12
3
4 # Output: x > y is False
5 print('x > y is',x>y)
6
7 # Output: x < y is True
8 print('x < y is',x<y)
9
10 # Output: x == y is False
11 print('x == y is',x==y)
12
13 # Output: x != y is True
14 print('x != y is',x!=y)
15
16 # Output: x >= y is False
17 print('x >= y is',x>=y)
18
19 # Output: x <= y is True
20 print('x <= y is',x<=y)
Console
Shell
~/Day-03$ python comparision_operators.py
x > y is False
x < y is True
x == y is False
x != y is True
x >= y is False
x <= y is True
~/Day-03$

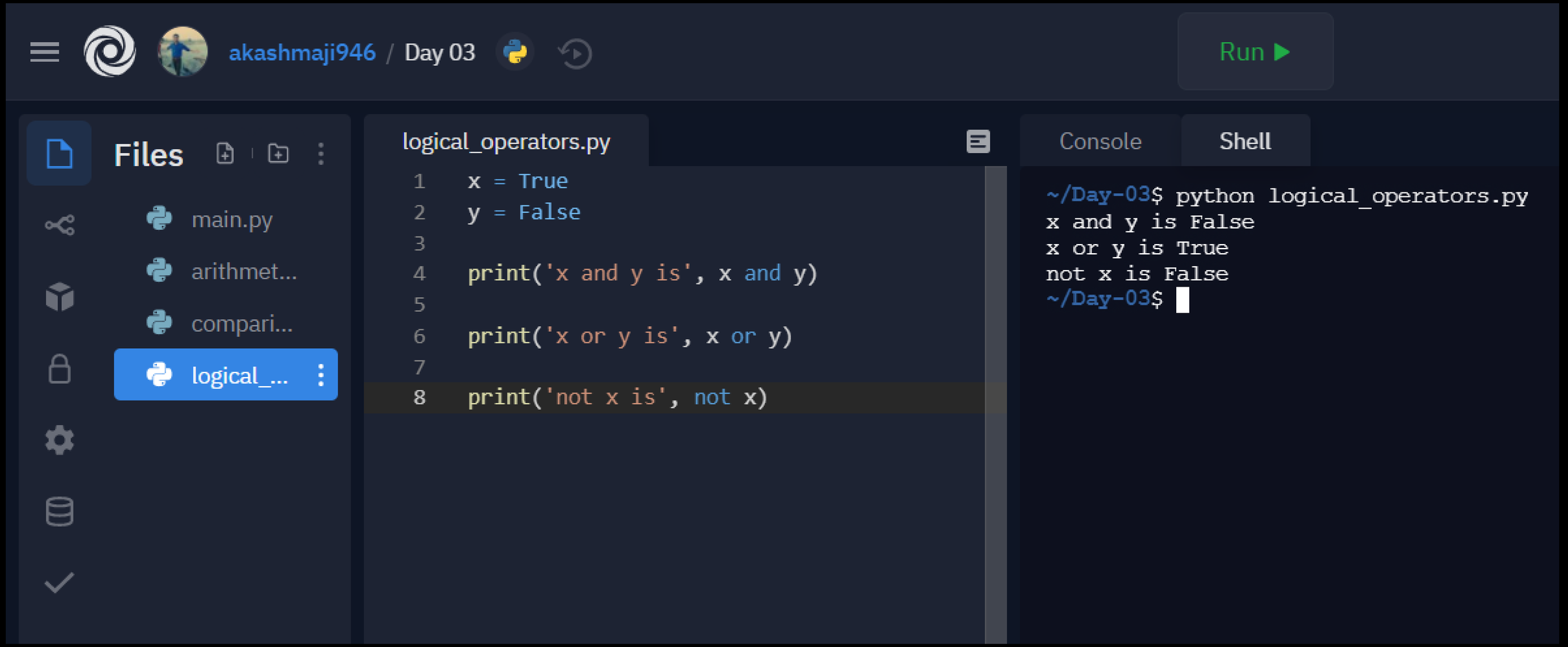
Logical Operators

Logical operators

Logical operators are the and, or, not operators.

They implement boolean logic

Operator	Meaning	Example
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	True if operand is false (complements the operand)	not x


akashmaji946 / Day 03
Run
Files
main.py
arithmet...
compari...
logical_...
logical_operators.py
1 x = True
2 y = False
3
4 print('x and y is', x and y)
5
6 print('x or y is', x or y)
7
8 print('not x is', not x)
Console
Shell
~/Day-03$ python logical_operators.py
x and y is False
x or y is True
not x is False
~/Day-03$

Bitwise operators

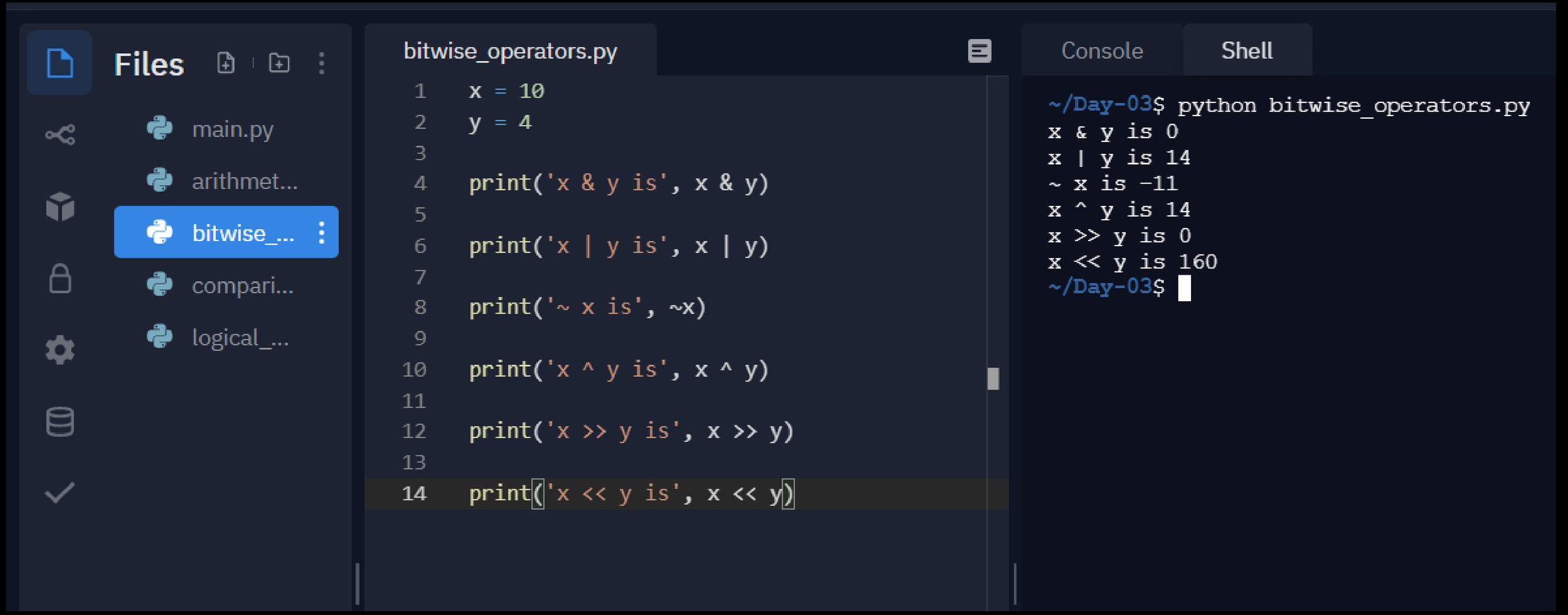
Bitwise operators

Bitwise operators act on operands as if they were strings of binary digits. They operate bit by bit, hence the name.

For example, 2 is 10 in binary and 7 is 111.

In the table below: Let `x` = 10 (`0000 1010` in binary) and `y` = 4 (`0000 0100` in binary)

Operator	Meaning	Example
&	Bitwise AND	x & y = 0 (`0000 0000`)
|	Bitwise OR	x | y = 14 (`0000 1110`)
~	Bitwise NOT	~x = -11 (`1111 0101`)
^	Bitwise XOR	x ^ y = 14 (`0000 1110`)
>>	Bitwise right shift	x >> 2 = 2 (`0000 0010`)
<<	Bitwise left shift	x << 2 = 40 (`0010 1000`)


Files
main.py
arithmet...
bitwise_...
compari...
logical_...
bitwise_operators.py
1 x = 10
2 y = 4
3
4 print('x & y is', x & y)
5
6 print('x | y is', x | y)
7
8 print('~ x is', ~x)
9
10 print('x ^ y is', x ^ y)
11
12 print('x >> y is', x >> y)
13
14 print('x << y is', x << y)
Console
Shell
~/Day-03$ python bitwise_operators.py
x & y is 0
x | y is 14
~ x is -11
x ^ y is 14
x >> y is 0
x << y is 160
~/Day-03$

Assignment operators

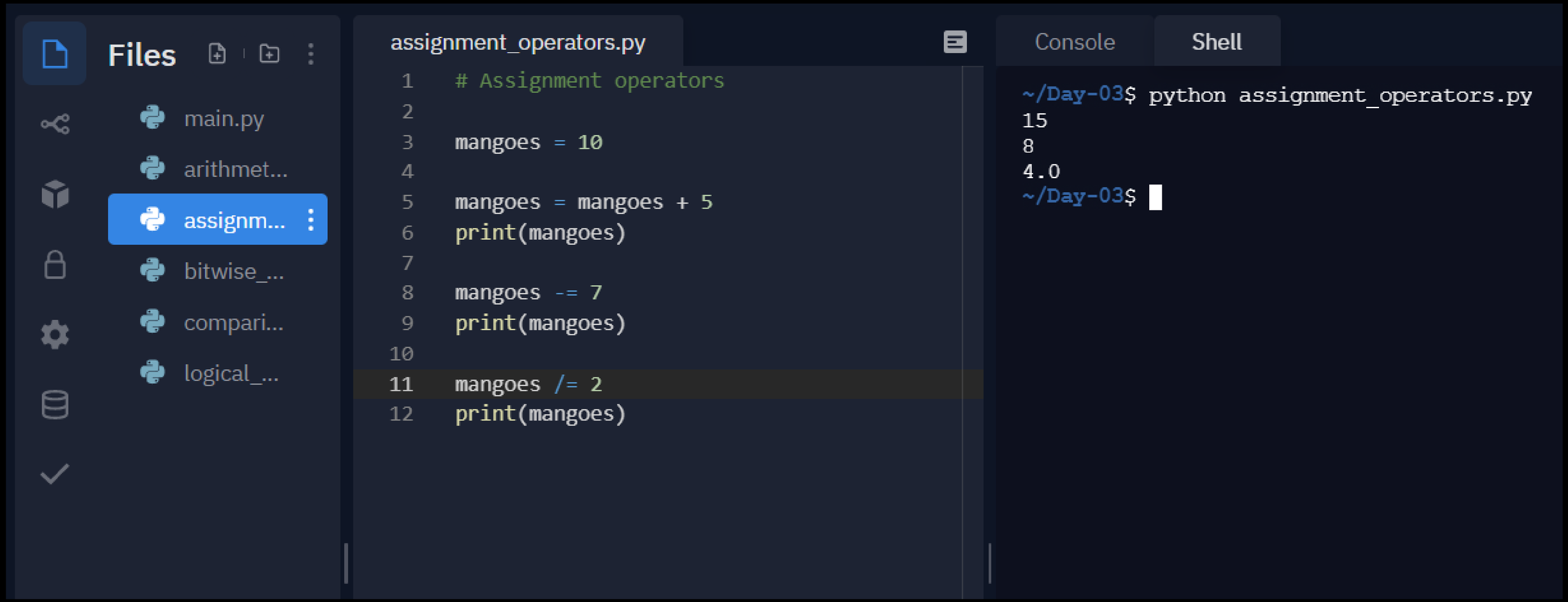
Assignment operators

Assignment operators are used in Python to assign values to variables.

`a = 5` is a simple assignment operator that assigns the value 5 on the right to the variable a on the left.

There are various compound operators in Python like `a += 5` that adds to the variable and later assigns the same. It is equivalent to `a = a + 5`.

Operator	Example	Equivalent to
=	x = 5	x = 5
+=	x += 5	x = x + 5
-=	x -= 5	x = x - 5
*=	x *= 5	x = x * 5
/=	x /= 5	x = x / 5
%=	x %= 5	x = x % 5
//=	x //= 5	x = x // 5
**=	x **= 5	x = x ** 5
&=	x &= 5	x = x & 5
|=	x |= 5	x = x | 5
^=	x ^= 5	x = x ^ 5


Files
main.py
arithmet...
assignm...
bitwise_...
compari...
logical_...
assignment_operators.py
1 # Assignment operators
2
3 mangoes = 10
4
5 mangoes = mangoes + 5
6 print(mangoes)
7
8 mangoes -= 7
9 print(mangoes)
10
11 mangoes /= 2
12 print(mangoes)
Console
Shell
~/Day-03$ python assignment_operators.py
15
8
4.0
~/Day-03$

Special operators

Python language offers some special types of operators like the identity operator or the membership operator.

Identity operators

is and is not are the identity operators in Python.

They are used to check if two values (or variables) are located on the same part of the memory.

Two variables that are equal does not imply that they are identical.

```
x1 = 5
y1 = 5
x2 = 'Hello'
y2 = 'Hello'
x3 = [1,2,3]
y3 = [1,2,3]

# Output: False
print(x1 is not y1)

# Output: True
print(x2 is y2)

# Output: False
print(x3 is y3)
```

Membership operators

Membership operators

- in and not in are the membership operators in Python.

- They are used to test whether a value or variable is found in a sequence (string, list, tuple, set and dictionary).

- In a dictionary we can only test for presence of key, not the value.

```
x = 'Hello world'
y = {1:'a',2:'b'}

# Output: True
print('H' in x)

# Output: True
print('hello' not in x)

# Output: True
print(1 in y)

# Output: False
print('a' in y)
```

THANK YOU