# 2.3. Operators and Operands

You can build complex expressions out of simpler ones using **operators**. Operators are special tokens that represent computations like addition, multiplication and division. The values the operator works on are called **operands**.

The following are all legal Python expressions whose meaning is more or less clear:

```
20 + 32
5 ** 2
(5 + 9) * (15 - 7)
```

The tokens `+`, `-`, and `*`, and the use of parentheses for grouping, mean in Python what they mean in mathematics. The asterisk (`*`) is the token for multiplication, and `**` is the token for exponentiation. Addition, subtraction, multiplication, and exponentiation all do what you expect.

Remember that if we want to see the results of the computation, the program needs to specify that with the word `print`. The first three computations occur, but their results are not printed out.

Save & Run    4/17/2021, 9:44:55 AM - 2 of 2    Show in CodeLens

```
20 + 32
5 ** 2
(5 + 9) * (15 - 7)
print(7 + 5)

```

```
12
```

Activity: 1 -- ActiveCode (ac2_3_1)

In Python 3, which we will be using, the division operator `/` produces a floating point result (even if the result is an integer; `4/2` is `2.0`). If you want truncated division, which ignores the remainder, you can use the `//` operator (for example, `5//2` is `2`).

Save & Run    4/17/2021, 9:45:14 AM - 2 of 2    Show in CodeLens

```
print(9 / 5)
print(5 / 9)
print(9 // 5)

```

```
1.8
0.5555555555556
1
```

Activity: 2 -- ActiveCode (ac2_3_2)

Pay particular attention to the examples above. Note that `9//5` truncates rather than rounding, so it produces the value 1 rather 2.

The truncated division operator, `//`, also works on floating point numbers. It truncates to the nearest integer, but still produces a floating point result. Thus `7.0 // 3.0` is `2.0`.

Save & Run    Original - 1 of 1    Show in CodeLens

```
print(7.0 / 3.0)
print(7.0 // 3.0)

```

```
2.33333333333
2.0
```

Activity: 3 -- ActiveCode (ac2_3_3)

The **modulus operator**, sometimes also called the **remainder operator** or **integer remainder operator** works on integers (and integer expressions) and yields the remainder when the first operand is divided by the second. In Python, the modulus operator is a percent sign (`%`). The syntax is the same as for other operators.

Save & Run 4/17/2021, 9:47:02 AM - 2 of 2 Show in CodeLens

```
print(7 // 3)    # This is the integer division operator
print(7 % 3)     # This is the remainder or modulus operator

```

```
2
1
```

Activity: 4 -- ActiveCode (ac2_3_4)

In the above example, 7 divided by 3 is 2 when we use integer division and there is a remainder of 1.

The modulus operator turns out to be surprisingly useful. For example, you can check whether one number is divisible by another—if `x % y` is zero, then `x` is divisible by `y`. Also, you can extract the right-most digit or digits from a number. For example, `x % 10` yields the right-most digit of `x` (in base 10). Similarly `x % 100` yields the last two digits.

**Check your understanding**

data-3-1: What value is printed when the following statement executes?

```
print(18 / 4)
```

[✓ A. 4.5]
B. 5
C. 4
D. 4.0
E. 2

Check me Compare me

✓ Because the result is not an integer, a floating point answer is produced.

Activity: 5 -- Multiple Choice (question2_3_1)

data-3-2: What value is printed when the following statement executes?

```
print(18.0 // 4)
```

A. 4.5
B. 5
C. 4
[✓ D. 4.0]
E. 2

Check me Compare me

✓ - Yes, even though it truncates, it produces a floating point result because 18.0 is a float

Activity: 6 -- Multiple Choice (question2_3_2)

data-3-3: What value is printed when the following statement executes?

```
print(18 % 4)
```

A. 4.25

B. 5

C. 4

[✓ D. 2]

Check me Compare me

✓ The % operator returns the remainder after division.

Activity: 7 -- Multiple Choice (question2_3_3)

You have attempted 8 of 7 activities on this page

✓ Completed. Well Done!

2.2. Values and Data Types">

ues and Data Types">

2.4. Function Calls">

| Back to top