Operators

This lesson will focus on different types of operators in Python.

WE'LL COVER THE FOLLOWING

Operators

Operators are used to perform arithmetic and logical operations on data. They use the provided data and give results. Some operators follow the *in-fix* (a + b) notation while some follow the *pre-fix* (- a) notation. With **in-fix** notation, the operands are placed on the left and right side of the operator. With **pre-fix** notation, the operator is placed before the operand. We will look at the different kinds of operators in Python in this lesson.

Arithmetic operators

Arithmetic operators are used to perform arithmetic operations such as addition, multiplication, division, etc. They can be used in conjunction with each other, hence they have an order of *precedence* as well. Some of the most common operators are listed below in order of precedence.

- `()`: Parenthesis. Whatever is in the parenthesis will be evaluated first.
- `**`: Exponent (**In-fix**)
- `%`, `*`, `/`, `//`: Modulo, Multiplication, Division, Floored Division (**In-fix**)
- `+`, `-`: Addition, Subtraction (**In-fix**)

Let’s look at an example.

```
print(3+4)
print(3-4.25)

# Precedence
result = 3 + 5 // 2 * 5 / (9**2)
x = 3
print(result)
print(result+x)
```

In **line 1**, we see an example of addition. In **line 2**, the subtraction operator is being used. We can see the printed result of both these lines. Then there is an example of precedence in **line 5**. The expression in parenthesis is being evaluated first and then the rest of the expression is evaluated. The answer is stored in the variable `result`. Let’s focus on how this expression is evaluated

Expression to be evaluated

3 + 5 // 2 * 5 / (9**2)

1 of 6

9**2 = 81

3 + 5 // 2 * 5 / 81

5 // 2 = 2

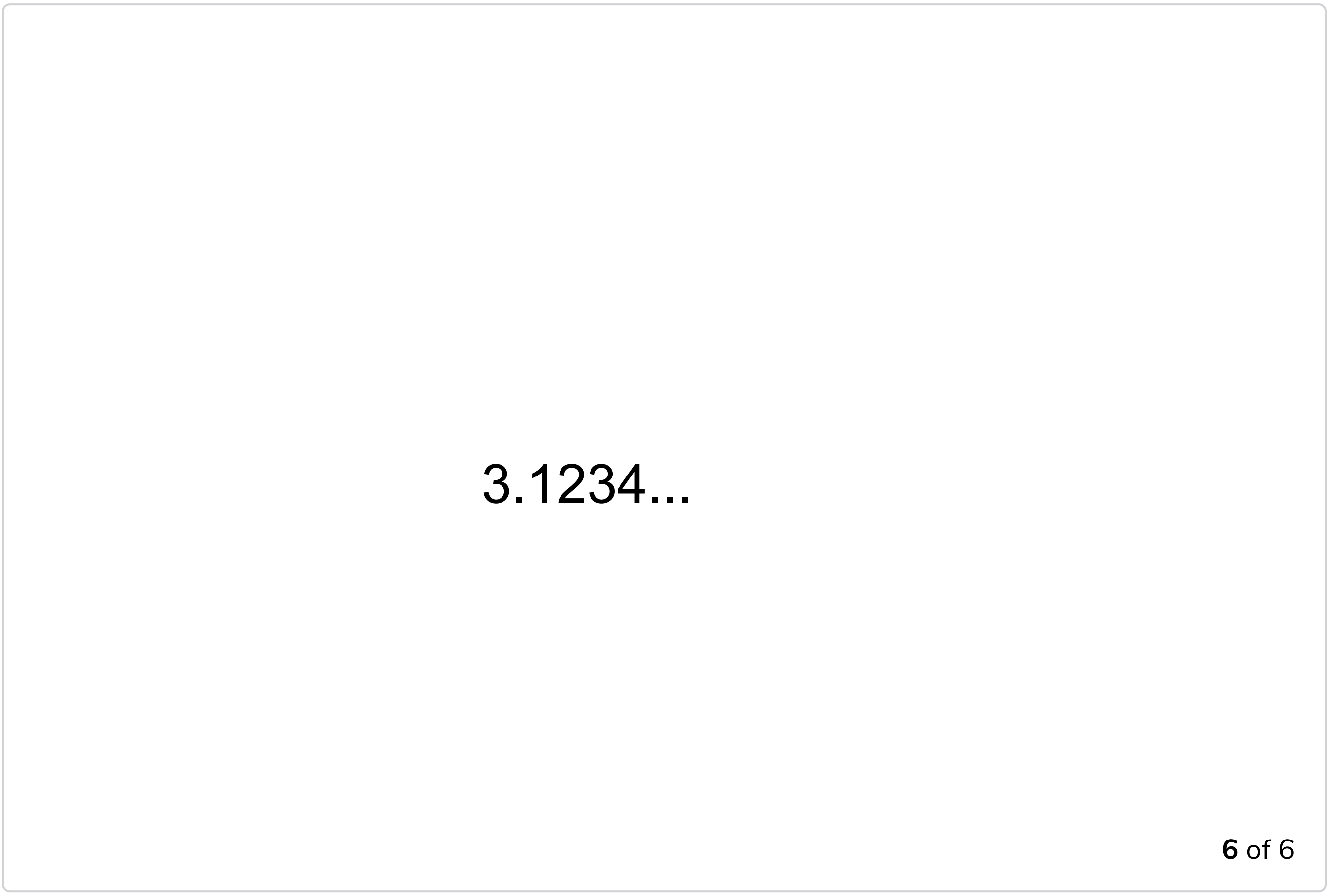
3 + 2 * 5 / 81

2 * 5 = 10

3 + 10 / 81

10 / 81 = 0.1234...

3 + 0.1234...



```
3.1234...
```

After evaluating the expression in the parenthesis, there are a bunch of operators that have the same precedence. So, Python decides to go from left to right. It first evaluates floored/integer division `//`, then multiplication `*`, followed by division `/`.

In the next line, we create another variable `x` and assign it the value `3`. In the end, we print `result` in **line** 7 and the result of the addition of `result` and `x` in **line 8**.

Comparison operators

Comparison operators are used for comparing values mathematically. They give answers in Booleans i.e. `True` and `False`. Some of the most common operators are listed below in order of precedence.

- `<`: Less than.
- `>`: Greater than
- `==`: Equal to
- `!=`: Not equal to

- `!=`: Not equal to
- `<=`: Less than or equal to
- `>=`: Greater than or equal to
- `is`: equal to
- `is not`: not equal to

Let’s look at an example.

```
num1 = 5
num2 = 10
num3 = 10

print("num2 > num1 : ", num2 > num1)
print("num1 > num2 : ", num1 > num2)

print("num2 is num3 : ",num2 is num3)
print("num3 is not num1 : ", num3 is not num1)

print(3 + 10 == 5 + 5)
print(3 <= 2)
```

In the first three lines, we create and assign values to variables `num1`, `num2`, and `num3`. In the rest of the code, we evaluate expressions that are either `True` or `False`. In **line 5**, we check whether `num2` is greater than `num1`. In **line 6**, we check whether `num1` is greater than `num2`. In **line 8**, we check whether `num2` is equal to `num3`. In **line 9**, we check whether `num3` is not equal to `num1`. In **line 11**, the expressions to the left and right of the `==` operator are evaluated first. Then equality is evaluated. `13==10` evaluates to `False`. Hence, `False` is printed. In the same way, the expression in **line 12** is evaluated to `False`.

Logical operators

Logical operators are used for evaluating Boolean logic expressions. Logical operators consist of:

- `and`: Evaluate the **AND** between two Booleans (**in-fix**).
- `or`: Evaluate the **OR** between two Booleans (**in-fix**).
- `not`: Evaluate the **Not** of a Boolean (**pre-fix**).

Let’s look at an example.

```
result = True or False
print(result)

result_2 = False or False
print(result_2)

result_3 = True and False
print(result_3)

result_4 = False
print(not result_4)
```

In **line 1**, we create a variable `result` which stores the result of the expression (`True` `OR` `False`). In the next line, we print the result. In the same way, `result_2`, `result_3` and `result_4` store the results of the Boolean expression which are printed in the following lines.

String operators

Some operators can be used with strings to perform different operations. Some of the most common operators are:

- `+`: Used to *concatenate* (join) two strings (**in-fix**).
- `*`: Used to multiply a string. Repeat the string. (**in-fix**).
- `in`: Used to search in a string. Returns `True`/`False` (**in-fix**).

Let's see an example.

```
message = 'Hello'
temp = 'World'
print(message + temp)

s = 'am'
print(s*3)

print(s in message)
```

In **lines 1-2** we create two string variables. We use the `+` operator between them and print the result. The strings joined to form a single string.

In **line 5**, we create a string. We repeat the string in pattern with the `*` operator and show the result in **line 6**. Note that the operator `*` does not modify the original string `s`, rather it creates a copy of it with the repeating pattern.

The operator `in` is used to search a string in the other string. In our case, we see that the string `s` is not in `message`, hence `False` is printed on the screen.

This brings our discussion on operators to an end. In the next lesson, we will look at conditionals.