

Introduction to Python

Basic Syntax

Variables and Operations

Topics

- 1) Operations
 - a) Arithmetic
 - b) Comparison
 - c) Boolean
- 2) Variables
- 3) Assignments and Augmented Assignments
- 4) User input

Arithmetic Operations

Operator	Name	Description
$a + b$	Addition	Sum of a and b
$a - b$	Subtraction	Difference of a and b
$a * b$	Multiplication	Product of a and b
a / b	True division	Quotient of a and b
$a // b$	Floor division	Quotient of a and b , removing fractional parts
$a \% b$	Modulus	Remainder after division of a by b
$a ** b$	Exponentiation	a raised to the power of b
$-a$	Negation	The negative of a
$+a$	Unary plus	a unchanged (rarely used)

Mixing Types

Any expression that two floats produce a float.

```
In[1]: 17.0 - 10.0
```

```
Out [1]: 7.0
```

When an expression's operands are an int and a float, Python automatically converts the int to a float.

```
In[2]: 17.0 - 10
```

```
Out [1]: 7.0
```

```
In[3]: 17 - 10.0
```

```
Out [1]: 7.0
```

Why floor(integer) division is useful

Sometimes we only want the integer part of division. Consider the question:

How many weeks are there in 25 days?

Answer: 3 weeks plus 4 days.

```
In[1]: 25 // 7
```

```
Out [1]: 3
```

Note that we take the floor of the division which is rounding **down** to the nearest integer. Be careful when the operands are negative:

```
In[2]: -25 // 7
```

```
Out [1]: -4
```

Why the modulus operator is useful

Sometimes we only want the remainder part of the division. Consider the question:

If today is a Tuesday, which day is 59 days from today?

Answer: 59 divided by 7 is 8 with a remainder of 3. Thus it will be Friday.

```
In[1]: 59 % 7
```

```
Out [1]: 3
```

Exponentiation and Negation

```
In[1]: 2 ** 3
```

```
Out[3]: 8
```

Negation is a **unary operator**. It applies to only one operand. Other operations such as $+$, $-$, $*$, $/$, $//$, $\%$ are **binary operators**, they apply to two operands.

```
In[2]: -5
```

```
Out [3]: -5
```

```
In[3]: --5
```

```
Out[3]: 5
```

Operator Precedence

Precedence	Operator	Operation
highest	**	exponentiation
	-	negation
	*, /, //, %	multiplication, division, floor division, modulus
lowest	+, -	adding, subtraction

Operators on the same row are applied left to right. Exponentiation, however, is applied right to left. Expressions in parenthesis are evaluated first.

Operator Precedence

In[1]: -2 ** 4

Out[3]: -16

In[2]: 7 - 4 * 5 % (1 + 2)

Out [3]: 5

7 - 4 * 5 % (1 + 2)

7 - 4 * 5 % 3

7 - 20 % 3

7 - 2

5

Comparison Operators

Operation	Description
a == b	a equal to b
a != b	a not equal to b
a < b	a less than b
a > b	a greater than b
a <= b	a less than or equal to b
a >= b	a greater than or equal to b

Note that = is for assignment and == is for comparison.

These operators return either True or False.

Comparison Operators

```
In[1]: 10 == 5
```

```
Out[1]: False
```

```
In[2]: 3 <= 7
```

```
Out[2]: True
```

```
In[3]: 3 != 7
```

```
Out[3]: True
```

Boolean Operations

Python provides operators to combine the values using the standard concepts of “and”, “or”, and “not”.

These operators are expressed using the words and, or, and not:

X	Y	X or Y
True	True	True
True	False	True
False	True	True
False	False	False

X	Y	X and Y
True	True	True
True	False	False
False	True	False
False	False	False

X	not X
True	False
False	True

Boolean Operations

```
In[1]: x = 4
```

```
(x < 6) and (x > 2)
```

```
Out[1]: True
```

```
In[2]: (x > 10) or (x % 2 == 0)
```

```
Out[2]: True
```

```
In[3]: not (x < 6)
```

```
Out[3]: False
```

Variables

We can use variables to refer to values that can be used later.

You can create a new variable by given it a value.

```
In[1]: x = 4  
      x
```

```
Out[1]: 4
```

Variable names can use letters, digits, and the underscore symbol (but they can't start with a digit).

= is not equality

Unlike in math, = is not equality in Python. It is an assignment: assign the expression on the right side of = to the variable on the left.

```
In[1]: x = 4
        x = x + 1    # in math, this has no solutions!
        x            #
```

```
Out[1]: 5
```

Assignment is not symmetric.

```
In[1]: x = 4    # correct!
        10 = y   # error!
```

Augmented Assignment

An **augmented assignment** combines an assignment statement with an operator to make the statement more concise.

Shorthand

variable += **value**;

variable -= **value**;

variable *= **value**;

variable /= **value**;

variable %= **value**;

Equivalent version

variable = **variable** + **value**;

variable = **variable** - **value**;

variable = **variable** * **value**;

variable = **variable** / **value**;

variable = **variable** % **value**;

```
In[1]: x = 4
```

```
    x += 1    # equivalent to x = x + 1
```

```
    x
```

```
Out[1]: 5
```


Augmented Assignment

```
In[1]: x = 3  
       x *= 2 + 5  
       x
```

```
Out[1]: 21
```

```
In[1]: number = 5  
       number *= number  
       number
```

```
Out[1]: 25
```

Input

Programs may use the input function to obtain information from the user.

```
In [1]: print('Please enter some text:')  
        x = input()  
        print('Text entered:', x)  
        print('Type:', type(x))
```

Please enter some text:

hello



Text entered: hello

Type: <class 'str'>

The second line shown in the output is entered by the user, and the program prints the first, third, and fourth lines.

Input

Since user input almost always requires a message to the user about the expected input, the input function optionally accepts a string that it prints just before the program stops to wait for the user to respond.

```
In [1]: x = input('Please enter an integer value: ')
        y = input('Please enter another integer value: ')
        num1 = int(x)
        num2 = int(y)
        print(num1, '+', num2, '=', num1 + num2)
```

Please enter an integer value: 4

Please enter another integer value: 5

4 + 5 = 9

Input

Or even more succinctly.

```
In [1]: num1 = int(input('Please enter an integer value: '))  
        num2 = int(input('Please enter another integer value: '))  
        print(num1, '+', num2, '=', num1 + num2)
```

Please enter an integer value: 4

Please enter another integer value: 5

4 + 5 = 9

References

I) Vanderplas, Jake, A Whirlwind Tour of Python, O'reilly Media.

This book is completely free and can be downloaded online at O'reilly's site.