

Introduction to Python

Operations and Variables

Topics

- 1) Arithmetic Operations
- 2) Floor Division vs True Division
- 3) Modulo Operator
- 4) Operator Precedence
- 5) String Concatenation
- 6) Augmented Assignment

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.

```
x = 17.0 - 10.0
```

```
print(x)          # 7.0
```

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

```
x = 17.0 - 10
```

```
print(x)          # 7.0
```

```
y = 17 - 10.0
```

```
print(y)          # 7.0
```

True Division vs Floor Division

The operator `/` is true division and the operator `//` returns floor division(round down after true divide). True divide `/` always gives the answer as a float.

```
print(23 // 7)      # 3
print(3 // 9)       # 0
print(-4 // 3)      # -2
print(6 / 5)        # 1.2
print(6 / 3)        # 2.0 NOT 2!
```

Remainder with %

The % operator computes the remainder after floor division.

- $14 \% 4$ is 2
- $218 \% 5$ is 3

$$\begin{array}{r} 3 \\ 4 \overline{) 14} \\ \underline{12} \\ 2 \end{array}$$

$$\begin{array}{r} 43 \\ 5 \overline{) 218} \\ \underline{20} \\ 18 \\ \underline{15} \\ 3 \end{array}$$

Applications of % operator:

- Obtain last digit of a number: $230857 \% 10$ is 7
- Obtain last 4 digits: $658236489 \% 10000$ is 6489
- See whether a number is odd: $7 \% 2$ is 1, $42 \% 2$ is 0

Modulo Operator

The operator % returns the modulus which is the remainder after floor division.

```
print(18 % 5)          # 3
print(2 % 9)           # 2, if first number is smaller, it's the answer
print(125 % 10)        # 5
print(0 % 10)          # 0
print(10 % 0)          # ZeroDivisionError
```

Why floor/modulo division is useful

Floor division allows us to extract the integer part of the division while the modulo operator extracts the remainder part of the division. Consider the question:

How many weeks and days are there in 25 days?

Answer: 3 weeks plus 4 days.

```
num_weeks = 25 // 7 # extracting number of weeks
print(num_weeks)    # 3
```

```
num_days = 25 % 7   # extracting number of days
print(num_days)     # 4
```


Why the modulo operator is useful

If today is a Tuesday, which day is 43 days from today?

Answer: 43 divided by 7 is 6 with a remainder of 1. Thus, it will be Wednesday.

```
print(43 % 7)    # 1
```

Even/odd: A number x is even if $x \% 2$ is 0 and odd if $x \% 2$ is 1.

```
num = int(input('Please enter an integer value: '))  
print(num, 'is even:', num % 2 == 0)
```

Output 1:

Please enter an integer value: 4
4 is even: True

Output 2:

Please enter an integer value: 13
13 is even: False

Expressions

Find the exact change for 137 cents using quarters, dimes, nickels and cents. Use the least number of coins.

How many quarters? $137 // 25 = 5$ quarters (Floor Division!)

What's leftover? $137 \% 25 = 12$ cents

How many dimes? $12 // 10 = 1$ dime

What's leftover? $12 \% 10 = 2$ cents

How many nickels? $2 // 5 = 0$ nickels.

What's leftover? $2 \% 5 = 2$ cents.

How many pennies? $2 // 1 = 2$ pennies

What's leftover? $2 \% 1 = 0$ cents. Done!

Extracting Digits

Given a three-digit integer. Extract its the ones, tens and hundreds digits.

For example, if the integer is 352. Its ones digit is the 2, its tens digit is the 5 and its hundreds digit is the 3.

```
number = 352
print("ones:", number % 10)      # ones: 2
number = number // 10            # number = 35 (discards last digit)
print(number)                    # 35
print("tens:", number % 10)      # tens: 5
number = number // 10            # number = 3 (discards last digit)
print(number)                    # 3
```

Note: Modulo 10 (% 10) extracts the last digit. Floor division (// 10) discards the last digit. Later in another lecture, we will see how to generalize this to any number of digits.

Extracting Digits

Alternatively:

```
number = 352
ones = number % 10
tens = (number // 10) % 10
hundreds = number // 100
print("ones:", ones, "tens", tens, "hundreds:", hundreds)
```

Output:

ones: 2 tens: 5 hundreds: 3

Exponentiation and Negation

```
x = 2 ** 3
```

```
print(x)          # 8
```

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

```
x = -5
```

```
y = --5
```

```
print(x)          # -5
```

```
print(y)          # 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(PEMDAS).

Operator Precedence

```
x = -2 ** 4
```

```
print(x)    # -16
```

```
y = 7 - 4 * 5 % (1 + 2)
```

```
print(y)    # 5
```

7 - 4 * 5 % (1 + 2)

7 - 4 * 5 % 3

7 - 20 % 3

7 - 2

5

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**;

```
x = 4
```

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

```
print(x)    # 5
```


Augmented Assignment

```
x = 3
```

```
x *= 2 + 5
```

```
print(x)                # 21
```

```
number = 5
```

```
number *= number
```

```
print(number)           # 25
```

String Concatenation

Two strings can be combined, or **concatenated**, using the + operator:

```
string1 = "abra"  
string2 = "cadabra"  
magic_string = string1 + string2
```

```
first = "Michael"  
last = "Smith"  
full_name = first + " " + last
```

Concatenating a string and a number raises a `TypeError`. Must first cast the number into a string using `str()`.

```
apples = "I have " + 3 + "apples"    # error!  
apples = "I have " + str(3) + "apples" # correct!
```

Lab 1: Modulo Operator

Create a new repl on repl.it. Write code to match the following console output. Underline numbers are user inputs; you must use the `input()` function.

Enter a three-digit number: 245

The sum of the digits of 245 is 11.

Enter amount in cents: 137

Number of quarters: 5

Number of dimes: 1

Number of nickels: 0

Number of pennies: 2

References

1) 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.