Python programming for beginners

Stefan Zhauryd
Instructor

S.Zhauryd All rights reserved © Confidential

Exponentiation (power)

**

example.py

```
1   print(2 ** 3)
2   print(2 ** 3.)
3   print(2. ** 3)
4   print(2. ** 3.)
```

ПРОБЛЕМЫ ТЕРМИНАЛ ···

```
8
8.0
8.0
8.0
```

when both ** arguments are integers, the result is an integer, too;

when at least one ** argument is a float, the result is a float, too.

All rights reserved © Confidential

Multiplication

*

```
print(2 * 3)
print(2 * 3.)
print(2. * 3)
print(2. * 3.)
```

```
6
6.0
6.0
6.0
```

still work

All rights reserved © Confidential

Division /

```
print(6 / 3)
print(6 / 3.)
print(6. / 3)
print(6. / 3.)
```

```
2.0
2.0
2.0
2.0
```

The result produced by the division operator **is always a float**, regardless of whether or not the result seems to be a float at first glance: 1 / 2, or if it looks like a pure integer: 2/1

All rights reserved © Confidential

Integer Division //

```
print(6 // 3)
print(6 // 3.)
print(6. // 3)
print(6. // 3.)

print(6 // 4)
print(6. // 4)
```

```
2
2.0
2.0
2.0
1
1.0
```

its result lacks the fractional part - it's absent (for integers), or is always equal to zero (for floats); this means that the results are always rounded;

it conforms to the integer vs. float rule.

All rights reserved © Confidential

Remainder (modulo) %

```
print(14 % 4)
print(14 % 4.)
print(14. % 4)
print(14. % 4.)
```

```
2
2.0
2.0
2.0
```

14 // 4 gives 3 → this is the integer quotient;

3 * 4 gives 12 → as a result of quotient and divisor multiplication;

14 - 12 gives 2 → this is the remainder.

All rights reserved © Confidential

NOTE!

Do not try to:

- **perform a division by zero;**
- **perform an integer division by zero;**
- **find a remainder of a division by zero.**

All rights reserved © Confidential

```
print((2 ** 4), (2 * 4.), (2 * 4))
```

```
16 8.0 8
```

```
print((-2 / 4), (2 / 4), (2 // 4), (-2 // 4))
```

```
-0.5 0.5 0 -1
```

Examples

```
print((2 % -4), (2 % 4), (2 ** 3 ** 2))
```

```
-2 2 512
```

All rights reserved © Confidential

Strings

```
>>> "abc" / 9
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    "abc" / 9
TypeError: unsupported operand type(s) for /: 'str' and 'int'
>>> "abc" ** 9
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    "abc" ** 9
TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'int'
>>> "abc" // 9
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    "abc" // 9
TypeError: unsupported operand type(s) for //: 'str' and 'int'
```

Only duplication and concatenation

```
>>> "abc" + "gca"
'abcgca'
>>> "abc" * 4
'abcabcabcabc'
```

All rights reserved © Confidential

Key takeaways

1. An expression is a combination of values (or variables, operators, calls to functions - you will learn about them soon) which evaluates to a value, e.g., 1 + 2.

2. Operators are special symbols or keywords which are able to operate on the values and perform (mathematical) operations, e.g., the * operator multiplies two values: x * y.

All rights reserved © Confidential

Key takeaways

3. Arithmetic operators in Python:

+ (addition),

- (subtraction),

* (multiplication),

/ (classic division - **always returns a float**),

% (modulus - divides left operand by right operand and returns the remainder of the operation, e.g., 5 % 2 = 1),

** (exponentiation - left operand raised to the power of right operand, e.g., 2 ** 3 = 2 * 2 * 2 = 8),

// (floor/integer division - returns a number resulting from division, but rounded down to the nearest whole number, e.g., 3 // 2.0 = 1.0)

All rights reserved © Confidential

Key takeaways

4. A unary operator is an operator with only one operand, e.g., -1, or +3.

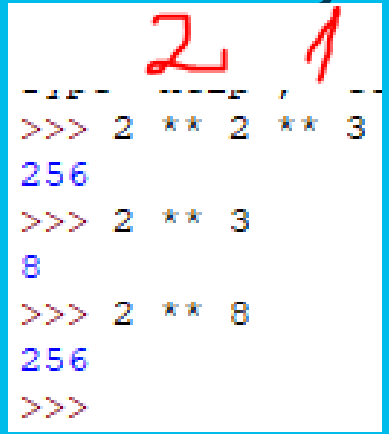
5. A binary operator is an operator with two operands, e.g., 4 + 5, or 12 % 5.

6. Some operators act before others - the hierarchy of priorities:

- **unary** + and - have the highest priority
- then: **, then: *, /, and %, and then the lowest priority: binary + and -.

All rights reserved © Confidential

Key takeaways



7. Subexpressions in parentheses are always calculated first, e.g.,

15 - 1 * (5 * (1 + 2)) = 0

8. The **exponentiation** operator uses **right-sided** binding, e.g., 2 ** 2 ** 3 = 256.

All rights reserved © Confidential

```
>>> import keyword
>>> print('Ключевые слова языка Python: {} {}'.format('\n', keyword.kwlist))
Ключевые слова языка Python:
 ['False', 'None', 'True', '__peg_parser__', 'and', 'as', 'assert', 'async',
'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except
', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda',
'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with',
'yield']
>>> |
```

They are called keywords or (more precisely) **reserved keywords**

['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']

All rights reserved © Confidential

Examples

```
anything = input("Enter a number: ")
something = anything ** 2.0
print(anything, "to the power of 2 is", something)
```

All rights reserved © Confidential

ЗАДАНИЯ

Название файлов, которые вы отправляете мне в telegram:

Vasia_Pupkin_class_work_L2_2.py

Формат сообщения которое вы присылаете мне

(после полного выполнения домашнего задания, только один раз) в Telegram:

Добрый день/вечер.

Я Вася Пупкин, и это мои домашние задания к лекции 2 часть 2.

И отправляете файл

Крайний срок сдачи 28/09 в 21:00 (можно раньше, но не позже)

All rights reserved © Confidential

a) input() is?
b) available operation with string?
c) ** is the left or right-side operator?
d) 1. Int, 2. myVariable, 3. 23true, 4. love_python
e) extension of python files/scripts?
f) 2 == (2 ** 2 - 2)?
g) 0 == True?

All rights reserved © Confidential

Create your possibilities. Bye bye.

All rights reserved © Confidential