IT Академія
від stfalcon.com

Python

Python

- Program contains modules
- Module contains operators
- Operator contains expressions
- Expressions create and process objects

Native types

Object Type	Example
Number	*1234, 3.1415, 3+4j, 0b111, Decimal()*
String	*'spam', "Bob's", b'a\x01c', u'sp\xc4m'*
List	*[1, [2, 'three'], 4.5], list(range(10))*
Dictionary	*{'food': 'spam', 'taste': 'yum'}, dict(hours=10)*
Tuple	(1, 'spam', 4, 'U'), tuple('spam'), namedtuple
File	open('eggs.txt'), open(r'C:\ham.bin', 'wb')
Other core types	Booleans, types, None
Program unit types	Functions, modules, classes

Operators

Operators	Description
lambda args: expression	Anonymous function generation
x if y else z	Ternary selection (x is evaluated only if y is true)
x or y	Logical OR (y is evaluated only if x is false)
x and y	Logical AND (y is evaluated only if x is true)
not x	Logical negation
x in y, x not in y	Membership (iterables, sets)
x is y, x is not y	Object identity tests
x < y, x <= y, x > y, x >= y	Comparsion
x == y, x != y	Value equality operators

Operators

Operators	Description
x | y	Bitwise OR, set union
x & y	Bitwise AND, set intersection
x << y, x >> y	Shift x left or right by y bits
x + y	Addition, concatenation;
x – y	Subtraction, set difference
x * y	Multiplication, repetition;
x % y	Remainder, format;
x / y, x // y	Division: true and floor
x ** y	Power (exponentiation)

Operators

Operators	Description
x[i]	Indexing (sequence, mapping, others)
x[i:j:k]	Slicing
x(...)	Call (function, method, class, other callable)
x.attr	Attribute reference
(...)	Tuple, expression, generator expression
[...]	List, list comprehension
{...}	Dictionary, set, set and dictionary comprehensions

Numbers Python < 2.7 and 3

```
1.3.1415 * 2
2.# 6.2830000000000004
3.print(3.1415 * 2)
4.# 6.283
```

Numbers

int	float	complex
10	0.0	3.14j
100	15.20	45.j
-786	-21.9	9.322e-36j
080	32.3+e18	.876j

Numbers Python > 2.7 and 3

```
1.3.1415 * 2
2.# 6.283
```

Math modules

```
1.import math
2.math.pi
3.#3.141592653589793
4.math.sqrt(85)
5.#9.219544457292887
```

```
1.import random
2.random.random()
3.0.7082048489415967
4.random.choice([1, 2, 3, 4])
5.# 1
```

Strings

```
1.str = 'Hello World'
2.print(len(str))
3.#11
4.print(str[0])
5.#H
6.print(str[0:4])
7.#Hell
8.print(str[-1])
9.#d
10.print(str[:-5])
11.#Hello
12.print(str + 'test')
13.#Hello Worldtest
14.print(str * 4)
15.#Hello WorldHello WorldHello WorldHello World
```

String Immutability

```
1.str = "abc"
2.str[0] = 'd'
3.#TypeError: 'str' object does not support
item assignment
4.str = 'd' + str[1:]
5.#dbc
```

String Immutability

```
1.str = "abc"
2.str[0] = 'd'
3.#TypeError: 'str' object does not support
item assignment
4.str = 'd' + str[1:]
5.#dbc
```

String Immutability

```
1.str = "hello"
2.l = list(str)
3.print(l)
4.#['h', 'e', 'l', 'l', 'o']
5.l[0] = 't'
6.print(l)
7.#['t', 'e', 'l', 'l', 'o']
8.str = ''.join(l)
9.print(str)
10.#tello
```

String Type-Specific Methods

```
1.str = 'test'
2.print(str.find('s'))
3.#2
4.print(str.replace('s','sch'))
5.#tescht
6.
7.line = 'abc,def,ghi'
8.print(line.split(','))
9.#['abc', 'def', 'ghi']
10.
11.print(line.upper())
12.#ABC,DEF,GHI
```

String Formatting

```
1.str1 = '%s, eggs, and %s' % ('spam', 'SPAM!') #
Formatting expression (all)
2.print(str1)
3.#'spam, eggs, and SPAM!'
4.str2 = '{0}, eggs, and {1}'.format('spam', 'SPAM!')
# Formatting method (2.6+, 3.0+)
5.print(str2)
6.#'spam, eggs, and SPAM!'
7.str3 = '{}, eggs, and {}'.format('spam', 'SPAM!') #
Numbers optional (2.7+, 3.1+)
8.print(str3)
9.#'spam, eggs, and SPAM!'
10.
```

List

```
1.list = [ 'abcd', 786 , 2.23, 'john', 70.2 ]
2.tinylist = [123, 'john']
3.
4.print (list) # Prints complete list
5.print (list[0]) # Prints first element of the list
6.print (list[1:3]) # Prints elements starting from 2nd till 3rd
7.print (list[2:]) # Prints elements starting from 3rd element
8.print (tinylist * 2) # Prints list two times
9.print (list + tinylist) # Prints concatenated lists
```

List operations

```
1.list = ['one', 'two', 'three', 'four']
2.print(list) #['one', 'two', 'three', 'four']
3.list.append('five')
4.print(list) #['one', 'two', 'three', 'four', 'five']
5.print(list.pop(3)) #four
6.print(list) #['one', 'two', 'three', 'five']
7.list.sort()
8.print(list) #['five', 'one', 'three', 'two']
9.list.reverse()
10.print(list) #['two', 'three', 'one', 'five']
```

Dictionary

```
1.dict = {}
2.dict['one'] = "This is one"
3.dict[2] = "This is two"
4.
5.tinydict = {'name': 'john','code':6734, 'dept': 'sales'}
6.
7.print(dict['one']) #"This is one"
8.print(dict[2]) # "This is two"
9.print(tinydict) # {'name': 'john', 'code': 6734, 'dept':'sales'}
10.print(tinydict.keys()) # dict_keys(['name', 'code', 'dept'])
11.print(tinydict.values()) # dict_values(['john', 6734, 'sales'])
```

Tuple

```
testTuple = ("apple", "banana", "cherry")
print(testTuple) #('apple', 'banana', 'cherry')
print(testTuple[1]) #banana

tmp = list(testTuple)
tmp[1] = "kiwi"
testTuple = tuple(tmp)
print(testTuple) #('apple', 'kiwi', 'cherry')

#thistuple[3] = "orange" # This will raise an error

thistuple = ("apple",)
print(type(thistuple)) #<class 'tuple'>

#NOT a tuple
thistuple = ("apple")
print(type(thistuple)) #<class 'str'>
```

Tuple

```
#testTuple = tuple("apple", "banana", "cherry")
#TypeError: tuple expected at most 1 argument, got 3

testTuple = tuple(("apple", "banana", "cherry"))
print(type(testTuple)) #<class 'tuple'>

tuple1 = (1, 2, 3)

tuple3 = tuple1 + testTuple
print(tuple3) #(1, 2, 3, 'apple', 'banana', 'cherry')

print(tuple3.count("apple")) #1
```

IT Академія

від stfalcon.com