PyCon China 2020

PyConChina 2020
PYTHON 中国开发者大会 2020

Python For Good

那些用起来很爽
但用不好可能会被人打的骚操作

Loco

NightTeam

快速将两个分别存放有key和value的列表合并成一个字典


```
>>> a = ["key1", "key2", "key3"]
>>> b = ["value1", "value2", "value3"]
>>> dict(zip(a, b))
{'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}
```

快速将两个分别存放有key和value的列表合并成一个字典


```
>>> result = ["key1", "value1", "key2", "value2", "key3", "value3"]
>>> result[0::2]
['key1', 'key2', 'key3']
>>> result[1::2]
['value1', 'value2', 'value3']
>>> dict(zip(result[0::2], result[1::2]))
{'key1': 'value1', 'key2': 'value2', 'key3': 'value3'}
```

快速按头、尾、中间部分切割元素，并将它们分别赋值给三个变量


```
>>> a = "123456789"  # 也可以是list之类的
>>> a1, *a2, a3 = a
>>> a1
'1'
>>> a2
['2', '3', '4', '5', '6', '7', '8']
>>> a3
'9'
```

快速解压内含嵌套列表的列表，并同时将嵌套列表内的值也赋值给不同的变量


```
>>> result = [1, 2, [3, 4], 5]
>>> [a, b, [c, d], e] = result
>>> a
1
>>> b
2
>>> c
3
>>> d
4
>>> e
5
```

快速解压内含嵌套列表的列表，并同时将嵌套列表内的值也赋值给不同的变量


```
>>> result = [1, 2, [3, 4, 5]]
>>> [a, b, [c, *_]] = result
>>> a
1
>>> b
2
>>> c
3
```

遍历嵌套且长短不一的列表时，按头、尾切割，并将它们分别赋值给两个变量


```
>>> result = [["items", "item1", "item2", "item3"], ["status", 1]]
>>> for key, *values in result:
...     print(key)
...     print(values)
...
>>> items
['item1', 'item2', 'item3']
>>> status
[1]
```

遍历嵌套且长短不一的列表时，按头、尾切割，并将它们分别赋值给两个变量


```
>>> {key:values for key, *values in result}
{'items': ['item1', 'item2', 'item3'], 'status': [1]}
```

快速解压一个字典，并将它里面的key和value们分别赋值给不同的变量


```
>>> a = {"key1": "value1", "key2": "value2", "key3": "value3"}
>>> (key1, value1), (key2, value2), (key3, value3) = a.items()
>>> key1
'key1'
>>> value1
'value1'
>>> key2
'key2'
>>> value2
'value2'
```

快速解压一个字典，并将它里面的key和value们分别赋值给不同的变量


```
>>> result = {"code": 200, "data": {"balabala": 111}, "msg": None}
>>> (_, code), (_, data), (_, msg)  = result.items()
>>> code
200
>>> data
{'balabala': 111}
>>> msg
>>>
```

operator

动态构建出Python中的各种运算符

运算	语法	函数
加法	`a + b`	`add(a, b)`
字符串拼接	`seq1 + seq2`	`concat(seq1, seq2)`
包含测试	`obj in seq`	`contains(seq, obj)`
除法	`a / b`	`truediv(a, b)`
除法	`a // b`	`floordiv(a, b)`
按位与	`a & b`	`and_(a, b)`
按位异或	`a ^ b`	`xor(a, b)`
按位取反	`~ a`	`invert(a)`
按位或	`a | b`	`or_(a, b)`
取幂	`a ** b`	`pow(a, b)`
标识	`a is b`	`is_(a, b)`
标识	`a is not b`	`is_not(a, b)`
索引赋值	`obj[k] = v`	`setitem(obj, k, v)`
索引删除	`del obj[k]`	`delitem(obj, k)`
索引取值	`obj[k]`	`getitem(obj, k)`
左移	`a << b`	`lshift(a, b)`
取模	`a % b`	`mod(a, b)`
乘法	`a * b`	`mul(a, b)`
矩阵乘法	`a @ b`	`matmul(a, b)`
取反（算术）	`- a`	`neg(a)`
取反（逻辑）	`not a`	`not_(a)`
正数	`+ a`	`pos(a)`
右移	`a >> b`	`rshift(a, b)`
切片赋值	`seq[i:j] = values`	`setitem(seq, slice(i, j), values)`
切片删除	`del seq[i:j]`	`delitem(seq, slice(i, j))`
切片取值	`seq[i:j]`	`getitem(seq, slice(i, j))`
字符串格式化	`s % obj`	`mod(s, obj)`
减法	`a - b`	`sub(a, b)`
真值测试	`obj`	`truth(obj)`
比较	`a < b`	`lt(a, b)`
比较	`a <= b`	`le(a, b)`
相等	`a == b`	`eq(a, b)`
不等	`a != b`	`ne(a, b)`
比较	`a >= b`	`ge(a, b)`
比较	`a > b`	`gt(a, b)`

动态构建出Python中的各种运算符


```
>>> import operator
>>> a = "666111666"
>>> operator.getitem(a, slice(3, 6))
'111'
```

动态创建函数


```
>>> from types import FunctionType
>>>
>>> func = FunctionType(compile(
...     ("def func():\n"
...      "    print(1)\n"
...      "    return 2"),
...     "<string>",
...     "exec"
... ).co_consts[0], globals())
>>> print(func())
1
2
```

动态导入


```
>>> import importlib
>>>
>>> module = importlib.import_module("operator")
```

动态导入


```
>>> import importlib
>>>
>>> module = importlib.import_module("operator")
>>> module.add(1, 1)  # 加法运算符
2
```

动态调用


```
>>> import importlib
>>>
>>> module = importlib.import_module("operator")
>>> func = getattr(module, "add")
>>> func(1, 1)
2
```

PyCon China 2020

Python For Good

PyConChina 2020 | PYTHON 中国开发者大会 2020

THANK YOU

知乎：loco

微信公众号：NightTeam

个人微信号：locoz666