Iterátory, generátory

Petr Pošík

Katedra kybernetiky, FEL ČVUT v Praze

OI, B4B33RPH - Řešení problémů a hry, 2016

Iterable

Všechny kolekce v Pythonu jsou **iterable**:

- Můžeme projít jejich prvky ve `for` cyklu.
- Můžeme otestovat přítomnost prvku pomocí `in`.

Objekt, který je iterable, musí implementovat metodu `__iter__(self)`, která musí vrátit iterátor.

In [1]:

```
s = ['a', 2, 3]
it = iter(s)  # Calls s.__iter__() internally
it
```

Out[1]:

```
<list_iterator at 0x5756310>
```

Iterator protokol

Iterátor je každý objekt, který podporuje následující dvě metody:

- `iterator.__iter__(self)`, která vrací iterátor samotný, a
- `iterator.__next__(self)`, která vrací další prvek kolekce. Nejsou-li žádné další prvky k dispozici, musí vyhodit výjimku `StopIteration`.

In [2]:

```
iter(it) is it  # __iter__ returns the iterator itself
```

Out[2]:

```
True
```

In [3]:

```
next(it)
```

Out[3]:

```
'a'
```

In [4]:

```
next(it)
```

Out[4]:

```
2
```

In [5]:

```
next(it)
```

Out[5]:

```
3
```

In [6]:

```
next(it)
```

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
<ipython-input-6-2cdb14c0d4d6> in <module>()
----> 1 next(it)

StopIteration:
```

Generátory

Generátory představují snadný způsob, jak vytvořit iterátor.

- Funkce/metody, v jejichž těle je uveden příkaz `yield`.
- Na rozdíl od kolekcí se obvykle nedají použít opakovaně (musí se znovu vytvořit).

In [7]:

```
def gen():
    print('Part A')
    yield 1
    print('Part B')
    yield 2
    print('Part C')
```

In [8]:

```
type(gen)
```

Out[8]:

```
function
```

In [9]:

```
v = gen()
```

In [10]:

```
v
```

Out[10]:

```
<generator object gen at 0x0566D660>
```

In [11]:

```
next(v)
```

```
Part A
```

Out[11]:

```
1
```

In [12]:

```
next(v)
```

```
Part B
```

Out[12]:

```
2
```

In [14]:

```
next(v)
```

```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
<ipython-input-14-10c82e4dde46> in <module>()
----> 1 next(v)

StopIteration:
```

In [15]:

```
for num in gen():
    print(num)
```

```
Part A
1
Part B
2
Part C
```

In [16]:

```
print(list(gen()))
```

```
Part A
Part B
Part C
[1, 2]
```

Fibonacci jako generátor

In [17]:

```
def fibgen(max_num):
    """Generate Fibonacci numbers lower than max_num"""
    a, b = 0, 1
    while a < max_num:
        yield a
        a, b = b, a+b
```

In [18]:

```
for num in fibgen(50):
    print(num)
```

```
0
1
1
2
3
5
8
13
21
34
```

Generátorové výrazy

Už jste viděli tzv. list comprehensions:

In [19]:

```
s = [i*2 for i in range(10)]
```

In [20]:

```
s
```

Out[20]:

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
```

Podobně je možné vytvářet tzv. generator expressions:

In [21]:

```
exp = (i*2 for i in range(10))
type(exp)
```

Out[21]:

```
generator
```

In [22]:

```
exp
```

Out[22]:

```
<generator object <genexpr> at 0x059FB420>
```

In [23]:

```
for n in exp:
    print(n, end=', ')
```

```
0, 2, 4, 6, 8, 10, 12, 14, 16, 18,
```

Hodnoty v generátorech a generátorových výrazech se generují postupně, po jednom. Šetří tak paměť, není třeba vytvořit všechny prvky najednou.

Hádanka

Čtyři muži chtějí v noci přejít chatrný most, který unese maximálně 2 z nich. Protože je most samá díra, potřebují k přechodu lampu. Jdou-li 2 muži spolu, cesta jim trvá jsko pomalejšímu z nich. Přejít most jim postupně trvá 1, 2, 5 a 10 minut. Jak mají most přejít, aby jim to trvalo co nejkratší dobu?

Řešení hrubou silou: Nejspíše budou muset jít vždy 2 zleva doprava a jeden z těch, kteří jsou vpravo, se s lampu vrátí doleva. To budou opakovat, dokud nebudou všichni 4 na druhé straně.

Zkusme vygenerovat všechny možné plány a ohodnotit jejich délku.

In [24]:

```
from itertools import combinations
from copy import copy

def move2(left, right, moves):
    """Choose 2 men from those on left and move them to right."""
    for pair in combinations(left, 2):
        # Copy state
        lft, rgt, mvs = copy(left), copy(right), copy(moves)
        # Move the pair from left to right
        rgt.append(pair[0])
        rgt.append(pair[1])
        lft.remove(pair[0])
        lft.remove(pair[1])
        # Store the move
        mvs.append(pair)
        # If all on the other side
        if len(rgt) == 4:
            yield mvs
        else:
            yield from move1(lft, rgt, mvs)
```

In [25]:

```
def move1(left, right, moves):
    """Choose 1 man from those on right and move him to left."""
    for man in right:
        # Copy state
        lft, rgt, mvs = copy(left), copy(right), copy(moves)
        # Move one from right to left
        lft.append(man)
        rgt.remove(man)
        # Store the move
        mvs.append((man,))
        yield from move2(lft, rgt, mvs)
```

In [26]:

```
def plans():
    """Generate all plans"""
    left = [1,2,5,10]
    right = []
    yield from move2(left, right, [])
```

In [27]:

```
all_plans = list(plans())
time_plans = [(sum(max(group) for group in plan), plan) for plan in all_plans]
for i, plan in enumerate(time_plans):
    print(i, plan)
best = min(time_plans)
print('\nThe best plan is', best)
```

```
0 (17, [(1, 2), (1,), (5, 10), (2,), (1, 2)])
1 (23, [(1, 2), (1,), (5, 10), (5,), (1, 5)])
2 (33, [(1, 2), (1,), (5, 10), (10,), (1, 10)])
3 (20, [(1, 2), (1,), (5, 1), (2,), (10, 2)])
4 (23, [(1, 2), (1,), (5, 1), (5,), (10, 5)])
5 (19, [(1, 2), (1,), (5, 1), (1,), (10, 1)])
6 (20, [(1, 2), (1,), (10, 1), (2,), (5, 2)])
7 (33, [(1, 2), (1,), (10, 1), (10,), (5, 10)])
8 (19, [(1, 2), (1,), (10, 1), (1,), (5, 1)])
9 (17, [(1, 2), (2,), (5, 10), (1,), (2, 1)])
10 (24, [(1, 2), (2,), (5, 10), (5,), (2, 5)])
11 (34, [(1, 2), (2,), (5, 10), (10,), (2, 10)])
12 (20, [(1, 2), (2,), (5, 2), (1,), (10, 1)])
13 (24, [(1, 2), (2,), (5, 2), (5,), (10, 5)])
14 (21, [(1, 2), (2,), (5, 2), (2,), (10, 2)])
15 (20, [(1, 2), (2,), (10, 2), (1,), (5, 1)])
16 (34, [(1, 2), (2,), (10, 2), (10,), (5, 10)])
17 (21, [(1, 2), (2,), (10, 2), (2,), (5, 2)])
18 (26, [(1, 5), (1,), (2, 10), (5,), (1, 5)])
```

Nastavení notebooku

Ignorujte jej.