Módulo 01 - Iteradores

Tópicos em Linguagens

`https://github.com/fsantanna-uerj/LPX/`

Francisco Sant'Anna

`francisco@ime.uerj.br`

1. Iteradores

Tópicos em Linguagens

`https://github.com/fsantanna-uerj/LPX/`

Francisco Sant'Anna

`francisco@ime.uerj.br`

Iteradores

Iterador

21 línguas

Artigo Discussão Ler Editar Ver histórico

Origem: Wikipédia, a enciclopédia livre.

Em programação de computadores, um iterador se refere tanto ao objeto que permite ao programador percorrer um container, (uma coleção de elementos) particularmente listas,[1][2][3] quanto ao padrão de projetos *Iterator*, no qual um iterador é usado para percorrer um container e acessar seus elementos. O padrão Iterator desacopla os algoritmos dos recipientes, porém em alguns casos, os algoritmos são necessariamente específicos dos containers e, portanto, não podem ser desacoplados.

Vários tipos de iteradores são frequentemente fornecidos via uma interface de container. Embora a interface e a semântica de um determinado iterador são fixas, os iteradores são muitas vezes implementados em termos das estruturas subjacentes a uma implementação de container e são frequentemente acoplados ao container para permitir a semântica operacional do iterador. Note que um iterador segue uma rota e também dá acesso a elementos de dados do container, mas não realiza iteração (ou seja, não tem total liberdade, como sugere sua terminologia). Um iterador é comportamentalmente semelhante ao cursor de um banco de dados. Iterators datam da linguagem de programação CLU de 1974.

Iteradores

- Um objeto (com algum estado interno), que representa uma coleção.
- Exporta um método `next` que retorna o próximo elemento da coleção, ou sinaliza o seu término.

Exercício

1. Explique o que é um *iterador* com as suas próprias palavras. Dê exemplos.
2. Dê exemplos de programas **seus** que usem iteradores e explique como eles são usados.

2. Iteradores em Python

Tópicos em Linguagens

`https://github.com/fsantanna-uerj/LPX/`

Francisco Sant'Anna

`francisco@ime.uerj.br`

Python

```
for v in [1,2,3]:
    print(v)
```

```
for v in iter([1,2,3]):
    print(v)
```

```
it = iter([1,2,3])
print(it)
print(next(it))
print(next(it))
print(next(it))
print(next(it))
```

`01-iter/01-python.py`

- Iteração / Iteration
  - `for`, `next->next->next`
- Iterável / Iterable
  - `[1,2,3]`, coleção
- Iterador / Iterator
  - `it`, objeto com next
- Função `iter` transforma um *iterável* em um *iterador*

Python - Iterator

- Um *Iterador* é um objeto que contém um número contável de valores.
- Um *Iterador* é um objeto que pode ser iterado, i.e., você pode percorrer todos os seus valores.
- Um *Iterador* é um objeto que implementa a interface com os médotos `__iter__` e `__next__`, e que pode gerar a exceção `StopIteration`.
- https://www.w3schools.com/python/python_iterators.asp

Python - Iterator

```
class MyIter:
    def __iter__ (self):
        <...> # inicializa
    def __next__ (self):
        <...> # retorna valor
        raise StopIteration # (ou)

ob = MyIter()
it = iter(ob)
next(it)
next(it)
<...>
```

```
class Sequencia:
    def __init__ (self, v):
        self.max = v
    def __iter__ (self):
        self.cur = 1
        return self
    def __next__ (self):
        if self.cur > self.max:
            raise StopIteration
        v = self.cur
        self.cur += 1
        return v

for v in Sequencia(10):
    print(v)

ob = Sequencia(5)
it = iter(ob)
print(next(it))
```

01-iter/02-iter.py

Exercício 1.2.1

- Crie um iterador em Python que receba uma string e retorne todos os caracteres da string, um por um.
  - Baseie-se na declaração a seguir...
  - Use o teste a seguir...

```
class Caracteres:
    def __init__ (self, s
        <...>
    def __iter__ (self):
        <...>
    def __next__ (self):
        <...>
```

```
for c in iter(Caracteres("ola mundo")):
    print(c)

# saida
o
l
...
```

```
01-iter/ex-1.2.1.lua
01-iter/ex-1.2.1.ceu
```

Exercício 1.2.2

- Considere um personagem que se move em um mapa nas 4 direções: “dir”, “esq”, “cima”, “baixo”.
- Crie um iterador em Python em que o personagem faça um movimento contínuo “em quadrado”, com 10 passos seguidos em cada direção.

```
class Quadrado:
    def __init__ (self):
        <...>
    def __iter__ (self):
        <...>
    def __next__ (self):
        <...>
```

```
for dir in Quadrado():
    print(dir)

# saida
cima
cima
...
```

```
01-iter/ex-1.2.2.lua
01-iter/ex-1.2.2.ceu
```

3. Iteradores em Lua

Tópicos em Linguagens
`https://github.com/fsantanna-uerj/LPX/`

Francisco Sant'Anna
`francisco@ime.uerj.br`

Lua

```
for i,v in ipairs({10,20,30}) do
    print(v)
end
```

```
f,s,i0 = ipairs({1,2,3})
i1,v1 = f(s,i0)
print(i1,v1)
i2,v2 = f(s,i1)
print(i2,v2)
i3,v3 = f(s,i2)
print(i3,v3)
i4,v4 = f(s,i3)
print(i4,v4)
```

`01-iter/03-lua.lua`

- Iteração / Iteration
  - `for`, `f(s,i)`
- Iterável / Iterable
  - `{1,2,3}`
- Iterador / Iterator
  - `f`, `s`, `i`
- Função `ipairs` transforma um *vetor* em um *iterador*

Lua - Iterator

- O comando **`for`** genérico funciona usando funções, chamadas de *iteradores*.
- A cada iteração, a função iteradora é chamada para produzir um novo valor, parando quando esse novo valor é **`nil`**.
- O início do loop produz três valores: uma função iteradora, um estado, e um valor inicial para a variável de controle.
- A cada iteração, a função iteradora recebe dois valores: o estado e a variável de controle.
- A cada iteração, a função retorna, pelo menos, o próximo valor para a variável de controle.
- http://www.lua.org/manual/5.2/pt/manual.html#3.3.5

Python vs Lua

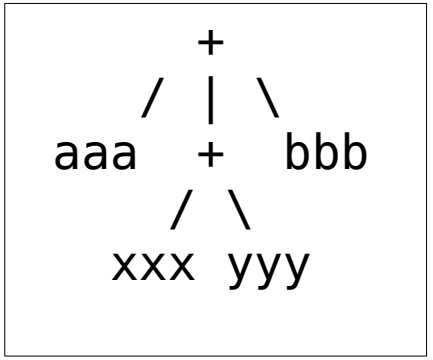
- Iteradores em Lua são *stateless*, pois não mantém nenhum estado interno para controle da iteração.
- Iteradores em Python são *stateful*, pois dependem de um objeto para manter o estado interno da iteração.
- Iteradores em Lua podem ser *stateful*, por exemplo usando *closures*.

`01-iter/ex-1.2.1-alt.lua`

Exercício 1.3.1

- Crie um iterador em Lua que receba uma arvore e retorne todas as folhas, da esquerda para a direita.
  - **Variante 1**: transforme a árvore em um vetor

```
a = {
    'aaa',
    {
        'xxx',
        'yyy',
    },
    'bbb'
}
```




```
for f in Arvore(a) do
    print(f)
end

-- saida
aaa
xxx
yyy
bbb
```

01-iter/ex-1.3.1.py

Exercício 1.3.2

- Crie um iterador em Python que receba uma arvore e retorne todas as folhas, da esquerda para a direita.
  - **Variante 2**: use um pilha auxiliar como estado

```
a = [
    “aaa”,
    [
        “xxx”,
        “yyy”
    ],
    “bbb”
]
```

```
for f in Arvore(a):
    print(f)

# saida
aaa
xxx
yyy
bbb
```

4. Iteradores

Tópicos em Linguagens

`https://github.com/fsantanna-uerj/LPX/`

Francisco Sant'Anna

`francisco@ime.uerj.br`

Iteradores

- Um objeto (com algum estado interno), que representa uma coleção.
- Exporta um método `next` que retorna o próximo elemento da coleção, ou sinaliza o seu término.

- Uma closure (com algum estado interno), que representa uma coleção.
- A função retorna o próximo elemento da coleção, ou sinaliza o seu término.

- [Lua] Uma construção de loop+função (sem estado interno), que itera sobre uma coleção representada por um estado+variável de controle.

Dificuldades

- Manter o estado interno do iterador (consultar e atualizar).
- Principalmente para coleções não lineares (e.g., árvores).

Exercício 1.4.1

- Crie um iterador em Lua que receba o nome de um arquivo e retorne todas as suas palavras.
  - Assuma que o arquivo só contém letras, espaços, ou linhas.

```
# x.txt

abc def
ghi jkl
mno pqr
stu vxz
```

```
for w in Arquivo("x.txt") do
    print(w)
end

-- saida
abc
def
ghi
jkl
...
```

Exercício 1.4.2

- Crie um iterador em Lua que retorne todos os números primos.

```
for n in Primos() do
    print(n)
end

-- saida
1
2
3
5
...
```

Módulo 01 - Iteradores

Tópicos em Linguagens

`https://github.com/fsantanna-uerj/LPX/`

Francisco Sant'Anna

`francisco@ime.uerj.br`