Universidade Federal de Minas Gerais
Ciência da Computação

Linguagens de Programação - Haniel Moreira Barbosa

Lista de Exercícios 6

Data de Entrega: 14/10/2020
Elaborada por Cleiton Silva

Considere a definição de um List data type em Python para responder os exercícios desta lista.

```
class ConsCell:
  def __init__(self, h, t):
    """Creates a new cell with head == h, and tail == t."""
    self.head = h
    self.tail = t

class List:
  """Describes a simple list data type."""

  def __init__(self, n):
    """creates a new list with n as the first element."""
    self.start = n

  def cons(self, e):
    """Adds a new element into the list; hence, procuding a new list
       ."""
    return List(ConsCell(e, self.start))

  def __str__(self):
    """Returns a textual representation of this list."""
    ans = "["
    cell = self.start
    while cell != 0:
      ans = ans + repr(cell.head)
      cell = cell.tail
      if cell != 0:
        ans = ans + ", "
    ans = ans + "]"
    return ans

def test():
    a = List(0)
    b = a.cons(2)
    c = b.cons("Hi!")
```

```
34      d = b.cons(True)
35      e = d.cons(False)
36      print("List a = ", a.__str__())
37      print("List b = ", b.__str__())
38      print("List c = ", c.__str__())
39      print("List d = ", d.__str__())
40      print("List e = ", e.__str__())
41
42  test()
```

1. Adicione um método length à classe List, de modo tal que x.length() retorne a quantidade de elementos existentes na lista x.

   Code:
   a = List(0)
   b = a.cons(2)

   Input:
   print("Length(a) = ", a.length())
   print("Length(b) = ", b.length())

   Output:
   Length(a) = 0
   Length(b) = 1

2. Adicione um método contains à classe List, de modo tal que x.contains(n) retorne true se o valor inteiro n ocorrer na lista x, do tipo List. O método deverá retornar false caso contrário.

   Code:
   a = List(0)
   b = a.cons(2)
   c = b.cons("hi")

   Input:
   print(b.contains(1))
   print(b.contains(2))
   print(c.contains("hi"))

   Output:
   False
   True
   True

3. Adicione um método sum à classe List, de modo tal que x.sum() retorne a soma dos elementos inteiros que ocorrem na lista x.

   Code:
   a = List(0)
   b = a.cons(2)

c = b.cons("hi")
d = c.cons(1)

Input:
print(d.sum())

Output:
3

4. Adicione um método concat à classe List, de modo tal que x.concat(y) retorne uma instância de List que seja igual a lista x na ordem reversa seguida da lista y. Não podem haver efeitos colaterais nem em x, tampouco em y.

   Code:
   a = List(0)
   b = a.cons(2)
   c = b.cons("Hi")
   d = b.cons("hello")

   Input:
   print(c.concat(d))

   Output:
   $[2, 'Hi', 'hello', 2]$

5. Considere o programa abaixo e responda o que acontecerá em cada linha da função test com um comentário. As opções possíveis são:

   (i) *Algo será impresso.* Neste caso, escreva o que será impresso.

   (ii) *Um erro será produzido em tempo de execução.*

```
1  class Animal:
2      atrb_animal = 0
3      def __init__(self, nome):
4          self.nome = nome
5      def __str__(self):
6          return self.nome + " eh um animal"
7      def comer(self):
8          print(self.nome + ", que eh um animal, esta comendo.")
9
10 class Mamifero(Animal):
11     def __str__(self):
12         return(self.nome + " eh um mamifero")
13     def beber_leite(self):
14         print(self.nome + ", que eh um mamifero, esta bebendo
               leite.")
15
16 class Cao(Mamifero):
17     def __str__(self):
18         return self.nome + " eh um cachorro"
```

```
19      def latir(self):
20          print(self.nome + " esta latindo bem auto.")
21      def comer(self):
22          print(self.nome + " late quando come.")
23          self.latir()
24
25  def test():
26      a1 = Animal("Tigrinho")
27      a2 = Mamifero("Oncinha")
28      a3 = Cao("Mameluco")
29      print(a1)           # 1
30      print(a2)           # 2
31      print(a3)           # 3
32      a1.comer()          # 4
33      a2.beber_leite()    # 5
34      a2.comer()          # 6
35      a3.latir()          # 7
36      a3.beber_leite()    # 8
37      a3.comer()          # 9
38      a1.beber_leite()    # 10
39      a1 = a3
40      a1.latir()          # 11
```

6. Considerando o programa abaixo, realize sua extensão seguindo as instruções:

```
1   class Visitor:
2     "A parameterized list visitor."
3     def __init__(self, cb):
4       self.cb = cb
5     def __str__(self):
6       return "Visitor with callback: {0}".format(self.cb)
7     def visit(self, n):
8       for i in range(0, len(n)):
9         n[i] = self.cb.update(n[i])
10      return n
11
12  class CallbackBase:
13    "The basic callback"
14    def __init__(self):
15      self.f = lambda x: x+1
16    def __str__(self):
17      return "basic callback"
18    def shouldUpdate(self, i):
19      return True
20    def update(self, i):
21      return self.f(i) if self.shouldUpdate(i) else i
```

(a) Implemente a classe CallbackCube que estende CallbackBase tal que um objeto

do tipo Visitor, parametrizado com um objeto do tipo CallbackCube, ao visitar uma lista de inteiros atualiza cada um de seus elementos com seu cubo. A função de impressão de CallbackCube deve imprimir “cube callback”.

(b) Implemente a classe CallbackOdd que estende CallbackBase tal que um objeto do tipo Visitor, parametrizado com um objeto do tipo CallbackOdd, ao visitar uma lista de inteiros atualiza apenas seus elementos ímpares. A função de impressão de CallbackImp deve imprimir “odd callback”.

(c) Implemente a classe CallbackMult que estende CallbackBase e CallbackOdd e apenas sobrescreve o método de impressão, produzingo ”multiple callback”. Descreva qual será o comportamento de um objeto do tipo Visitor, parametrizado com um objeto do tipo CallbackMult, ao visitar uma lista de n elementos inteiros.