I1-2015-1

April 27, 2015

1 Pregunta 1

Una posible solución al problema se presenta a continuación:

1.0.1 Solución

In [3]:
```
class Camara:
    '''
    Esta clase esta solo para poder interpretar el programa.
    No era necesario implementarla.
    '''

    def capturar(self):
        '''
        Creado solo para que el programa pueda ser interpretado.
        '''
        return "00101010"


class CamaraIP:

    @classmethod
    def from_data(cls, data):
        '''
        Creamos una instancia de la clase a partir de una data.
        No es necesario crear esta funcion, se puede asumir.
        '''
        camara = cls()
        camara.serie = data['serie']
        camara.modo = data['modo']
        camara.fps = data['fps']
        camara.mac = data['mac']
        camara.camara_normal = Camara()
        return camara


class Sistema:

    def __init__(self, data_camaras_ip, max_ips=100):
        '''
        Al momento de crear una instancia generamos las IPs y
        creamos las Camaras IP de la data recibida.
        La cantidad maxima de IPs queda a gusto del programador.
```

```
            '''
            self.ips_disponibles = [Sistema.ip_gen() for i in range(max_ips)]
            self.camaras_ip = [CamaraIP.from_data(d) for d in data_camaras_ip]

        def asignar_ip(self):
            '''
            Asignamos todas las IP que se puedan hasta que se acaben.
            '''
            for camara in self.camaras_ip:
                if len(self.ips_disponibles) > 0:
                    camara.ip = self.ips_disponibles.pop()
                else:
                    print("No quedan IPs disponibles")

            # Este retorno es solo por para mostrar que funciona.
            return {camara.mac: camara.ip for camara in self.camaras_ip}

        def capturar(self):
            return [c.camara_normal.capturar() for c in self.camaras_ip]

        @staticmethod
        def ip_gen():
            '''
            Generamos de manera bien Pythonistica una IP aleatoria.
            Para motivos de la prueba se puede asumir que las IPs existen y
            no es necesario esto.
            '''
            import random
            return '.'.join(str(random.randint(0, 255)) for i in range(4))


    if __name__ == '__main__':
        listado_camaras = [
            {'serie': 123, 'modo': '640x480', 'fps': 30, 'mac': 'AACC23'},
            {'serie': 456, 'modo': '1024x720', 'fps': 30, 'mac': 'AACC89'}
        ]

        sistema = Sistema(data_camaras_ip=listado_camaras)
        print("asignar_ip output: {}".format(sistema.asignar_ip()))
        print("capturar output: {}".format(sistema.capturar()))
```

```
asignar_ip output: {'AACC89': '214.10.203.198', 'AACC23': '81.230.183.233'}
capturar output: ['00101010', '00101010']
```

1.0.2 Distribución del puntaje

Puntaje:

Importante: El puntaje asignado aquí se multiplica por 15/6

- (1.0 ptos) Presencia del listado de IPs.
- (2.0 ptos) Generar listas de cámaras.
  - Solo se asignará (0.5 ptos) si guarda la lista tal y como llegó.
- (1.0 ptos) Asignar correctamente las IPs a las Cámaras IP
  - Solo se asignará (0.5 ptos) si no maneja la ausencia de IPs libres restantes.

- (2.0 ptos) Implementacion correcta de `capturar()`.

2 Pregunta 2

a) Solución propuesta por profesores:

```
In [3]: L = [ ("biblioteca","estante") , ("biblioteca","cubiculo") , ("estante","libro") , ("libro","ho
            ("cubiculo","mesa") , ("cubiculo","silla") , ("cubiculo","puerta"), ("libro","puerta")]

        def componentes(item, lista):
            # Usar sets es adecuado para que no se repitan entidades que ya están
            result = set(lista[a][1] for a in range(len(lista)) if lista[a][0] == item)
            if len(result):
                l_aux = []
                for i in result:
                    # El filter remueve los pares que tienen como segundo elemento el mismo componente
                    # pues ya sabemos que no nos servirán
                    l_aux.extend(componentes(i, list(filter((lambda x: x[1] != i), lista))))
                result = result.union({li for li in l_aux})
                return result
            return set()

        select = "biblioteca"
        print("Componentes({})-> {}".format(select, componentes(select, L)))
```

```
Componentes(biblioteca)-> {'estante', 'libro', 'silla', 'mesa', 'puerta', 'cubiculo', 'hojas'}
```

2.0.3 Distribución del puntaje

Puntaje:

- (5.0 ptos) Seleccionar/encontrar componentes contenidos en el `item` en cuestión, en cada iteración.
- (5.0 ptos) Extender adecuadamente la lista o set de resultados:
  - Por cada componente.
  - Por los componentes de cada componente.
- (5.0 ptos) Utilizar bien recursión.
  - Si se usa `for` (que resuelva bien) para reemplazar la recursión, se asigna la mitad del puntaje.

b) Dado el siguiente código en Python explique lo que hace cada método y genere la salida descrita en el `main`:

```
In [13]: class Nodo:

             def __init__(self, valor, izquierda=None, derecha=None):
                 self.valor = valor
                 self.izquierda = izquierda
                 self.derecha = derecha

             def m1(self, f):
                 r=''
                 if self.izquierda:
                     r+= f('pre',self.valor) + self.izquierda.m1(f)
```

```
            r+= f('in', self.valor)

            if self.derecha:
                r+= self.derecha.m1(f) + f('post', self.valor)
            return r

    class Arbol:

        def __init__(self, raiz):
            self.raiz = raiz

        def m2(self, f):
            if self.raiz: return self.raiz.m1(f)
            else: return None

        def __repr__(self):
            def m3(s, valor):
                if s == 'pre':
                    return '('
                elif s == 'in':
                    return str(valor)
                else:
                    return ')'
            return self.m2(m3)


    T = Arbol(Nodo('+', Nodo('-', Nodo(3), Nodo(4)), Nodo(2)))
    print(T)
```

```
((3-4)+2)
```

2.0.4 Respuesta

- `Arbol.__repr__`: En este caso, llama al método 'm2' de la instancia 'T'(del tipo 'Arbol'), pasándole como parámetro su función `m3`. En `m3`, según el parámetro `s` (del tipo `string`), se determina lo que se imprime en consola. Si `s` es igual a "pre", se imprime "("; si es "in", se imprime la representación del parámetro `valor` pasado a la función; si es cualquier otra cosa (como "post") se imprimirá ").

- `Arbol.m2`: Llama al método `m1` del campo `raiz` (del tipo `Nodo` en esta instancia), pasándole como parámetro la función `m3` de `T.__repr__` (donde `T` es una instancia de `Arbol`). Particularmente se pasa `m3` en este caso; podría ser cualquier función bajo otras circunstancias.

- `Arbol.__init__`: Inicializador de objetos del tipo `Arbol`. Se asigna un valor o referencia al campo `raiz` del objeto. En este caso, se está simulando una estructura de datos de tipo árbol, donde el campo `raiz` (del tipo `Nodo`) viene a ser el nodo raíz de esta estructura.

- `Nodo.__init__`: Inicializador de objetos del tipo `Nodo`. Le asigna obligatoriamente un valor o referencia al campo `valor` del objeto y opcionalmente le asigna valores o referencias a los campos `izquierda` y/o `derecha`. En este caso, se usó para guardar referencias de objetos tipo `Nodo` en `izquierda` y `derecha`.

- `Nodo.m1`: En este caso, llama a la función `__repr__.m3` del objeto `T`(del tipo `Arbol`) para que imprima recursivamente según los parámetros que le va pasando. Abstrayéndonos de la clase `Nodo`, toma el nodo raíz del árbol que simula la instancia `T` de la clase `Arbol` y viaja estrictamente hacia la izquierda del árbol. Por cada nodo que pasa, imprime un paréntesis izquierdo (sin contar el nodo raíz). Luego, imprime el campo `valor` de ese último nodo y si hay un nodo a la derecha, va a repetir el mismo procedimiento como si este último fuera el nodo raíz, hasta llegar a una hoja. Va imprimir el campo

`valor` de esta hoja. Luego, retorna por su camino. Por cada vez que se vuelva por el lado izquierdo de un nodo, se imprime el campo `valor`. Si se viene desde la derecha, se imprime un paréntesis derecho. Si llega a un nodo donde no ha tomado alguno de sus caminos, se desvía de su camino de regreso y toma el de ida por uno de esos caminos no recorridos, tomando el nodo en cuestión como nodo raíz. Se repite el procedimiento descrito previamente, hasta recorrer todos los caminos del árbol.

2.0.5 Distribución del puntaje

Puntaje:

- (2.0 ptos) Explicación `Arbol.__repr__`
- (1.0 ptos) Explicación `Arbol.m2`.
- (1.0 ptos) Explicación `Nodo.__init__` y `Arbol.__init__`
- (3.0 ptos) Explicación `Nodo.m1`
  - Se permiten otras explicaciones: esta explicación es didáctica, pero se aceptan explicaciones más teóricas (más orientadas
  - a cómo opera el método en lugar de abstrarse y recorrerlo como árbol).
- (3.0 ptos) Salida correcta.

3 Pregunta 4

a) Explique qué es un decorador y para qué puede servir

3.0.6 Respuesta

Un decorador es una función que recibe una función `f1`, y retorna una función `f2` distinta, que normalmente es una versión modificada de la función que se recibió como parámetro.

Los decoradores nos permiten agregar algún comportamiento o datos adicionales, incluso podemos decorar clases para modificarlas. Un beneficio es que nos evitan la necesidad de modificar el código de la función original.

3.0.7 Distribución del puntaje

El puntaje asignado es uno de los que aparece acá:

- (0.0 ptos) Respuesta incorrecta
- (2.5 ptos) Correcta definición o correcta utilidad, pero no ambas
- (5.0 ptos) Correcta definición y correcta utilidad

b) Implemente la función **factorial** usando **reduce**

```
In [7]: from functools import reduce

        def factorial(n):
            return reduce(lambda x,y: x*y, range(1, n + 1))

        # Ejemplos de que funciona
        print(factorial(3))
        print(factorial(4))
        print(factorial(5))
        print(factorial(9))
```

```
6
24
120
362880
```

3.0.8 Distribución del puntaje

El puntaje asignado es uno de los que aparece acá:

- (0.0 ptos) Incorrecto, o no uso de **reduce**
- (7.5 ptos) Errores leves o problemas de eficiencia
- (10.0 ptos) Correcto

c) Dadas dos listas de números `L1` y `L2`, genere un código que imprima el producto de todos los números contenidos en la lista `L3` cuyo elemento i-ésimo (`L3[i]`) es el que resulta de multiplicar `L1[i]*L2[i]`. Si `L1` y `L2` son de distinto tamaño, los elementos que sobran en la lista más larga deben no considerarse para la multiplicación. **No puede usar for ni if**

In [6]:
```
# Listas de ejemplo

L1 = [2,6,4,5]
L2 = [3,7,9,8,3500]

# Codigo
from functools import reduce

f = lambda x,y: x*y

L3 = list(map(f, L1, L2))
print(reduce(f, L3))
```

```
362880
```

3.0.9 Distribución del puntaje

- (5.0 ptos) Por computar correctamente L3 a partir de L1 y L2
- (5.0 ptos) Por computar el resultado final e imprimirlo
  - Se asignará solo 3.0 ptos si esta línea estaba correcta, pero en el ítem anterior se obtuvo 0.0 ptos.
- Si se usó **for**, **if** o **while** no lleva puntaje

d) Escriba la salida que el siguiente código genera con lo que se define en el main (justifique su respuesta)

In [9]:
```
def version(modo):
    def _version(cls):

        def _debug(self):
            attr = vars(self)
            print('MODO DEPURACION')
            for a, v in attr.items():
                print('{}: {} - {}'.format(a, v, type(v)))

        def _release(self):
            return None

        dict_modos = {'DEBUG': _debug, 'RELEASE': _release}
        setattr(cls, '__call__', dict_modos[modo])

        return cls
    return _version
```

```
        #@version('RELEASE')
        @version('DEBUG')
        class Imagen:
            def __init__(self, size):
                self.size = size
                self._data = None

            @property
            def data(self):
                return self._data

            @data.setter
            def data(self, data):
                self._data = data

        if __name__ == '__main__':
            img = Imagen((20,30))
            img.data = [255, 50, 100]
            img()
```

```
MODO DEPURACION
size: (20, 30) - <class 'tuple'>
_data: [255, 50, 100] - <class 'list'>
```

3.0.10 Distrubución de puntaje

- (7.0 ptos) Por todas las líneas correctas (se asignará puntaje proporcional)
  - Si las primeras líneas son correctas y existen líneas adicionales, habrá descuento de 2.3 puntos
- (3.0 ptos) Buena justificación que incluya lo que hace el decorador `@version`, y en qué momento se imprime lo que se imprime y por qué