# Práctico #2

## Test Automation With Python Noviembre 2016.

**Forma de entrega:** Via Email a jbc.develop@gmail.com

Se evalúa:

- Cantidad de características implementadas
- PEP 8 (con flake8 3.0)
- PEP 20: Zen de Python
- Uso de características de Python

**Entregar Jueves 8 de diciembre del 2016**

### Enunciado

- Implementar la clase Persona que recibe en el constructor nombre y edad y cuyo `repr(Persona(...))` es `<Persona 'nombre'>`

```
>>> from practico2 import Persona, Grupo
>>> p0 = Persona("juan", 18)
>>> p1 = Persona("tito", 34)
>>> p2 = Persona("carlos", 36)
```

- Dos clases personas se puede sumar y generan un grupo (se ejemplifica el `repr()` de las instancias de Grupo)

```
>>> g0 = p0 + p1
>>> g0
<Grupo '2 personas'>
```

- Los grupos y las personas son inmutables. Los grupos se crean con un iterable fijo de personas.
- Los grupos carecen de orden.
- Dos personas son iguales si tienen el mismo nombre y edad.

```
>>> bool(Persona("juan", 18) == Persona("juan", 18))
True
>>> bool(Persona("juan", 18) == Persona("juan", 19))
False
```

- Dos grupos son iguales si tienen las mismas personas.

```
>>> bool(Grupo({p0, p1}) != Grupo([p1, p0]))
False
>>> bool(Grupo((p0, p1)) != Grupo([p2, p0]))
True
```

- Un grupo puede sumar a otra persona y genera un nuevo grupo con todas las personas del grupo mas la nueva persona.

```
>>> g1 = g0 + p2
>>> g1
<Grupo '3 personas'>
```

- Se puede restar una persona de grupo.

```
>>> g2 = g1 - p2
>>> g2
<Grupo '2 personas'>
```

- Dos grupos se pueden sumar y restar. En el caso de suma la suma genera un grupo con las personas de los DOS grupos sin repetir; y en el caso de restar se crea un grupo donde solo están las personas del primer grupo donde que no están en el segundo.

```
>>> g0 - g2
<Grupo '2 personas'>
```

- Los grupos son iterables y iteran sobre las personas.

```
>>> for persona in g0:
...     print persona
<Persona 'juan'>
<Persona 'tito'>,
```

- Los grupos tienen longitud (cantidad de personas)

```
>>> len(g0)
2
>>> len(Grupo([]))
0
```

- Un grupo es verdadero solo si tiene alguna persona.

```
>>> if g0:
...     print "tiene"
tiene

>>> if Grupo([]):
...    print "tiene"
...
```

- Se puede saber si una persona esta en un grupo con el operador “in”

```
>>> p0 in g0
True
>>> p2 in g0
False
```

- El grupo tiene un método `edad_promedio` que calcula la edad promedio del grupo (float) o lanza un `EmptyGroupError` si el grupo está vacío (implementar la exception)

```
>>> g0.edad_promedio()
26.0
>>> Group([]).edad_promedio()
Traceback (most recent call last):
  ...
EmptyGroupErrorr: grupo vacio
```

- **Todos los ejemplos deberían andar con sus implementaciones.**
- Puede que alguna clase del módulo `collections` sea de ayuda.