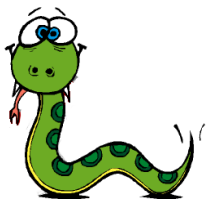
Instituto Tecnológico de Costa Rica
Área Académica de Ingeniería en Computadores
Introducción a la Programación / Taller de programación.
Profesor: Jeff Schmidt Peralta



EJERCICIOS INICIALES (c) SOBRE RECURSIVIDAD. II SEMESTRE 2019

1. La sumatoria de cocientes tiene la siguiente fórmula:

   $$\sum_{i=1}^{n} i/(i*(i+1))$$

   Escriba una función `suma_coc(n)` que reciba el límite superior de la sumatoria y calcule el resultado hasta ese número. Pruebe la función con diferentes valores de para `n` e indique cómo se comporta. Ejemplo con n = 50, 100, 200, 500, 1000

2. La siguiente sucesión numérica recibe únicamente valores de n enteros mayores o iguales a cero. Programe esta sucesión numérica.

   ```
   f(0) = 0
   f(1) = 1
   f(n) = 2 * f(n-2) + f(n-1)
   ```

   Los primeros cinco valores de esta sucesión son `0,1,1,3,5,11`.. Retorne el valor de la sucesión para el `n` dado, así como la cantidad de llamadas recursivas que se realizaron.

3. Hacer una función llamada `invertir(num)`, que reciba un número entero e invierta el orden de sus elementos (sin utilizar funciones de conversión de datos):

   ```
   >>> invertir(1234)
   4321
   ```

4. Escribir una función en recursividad de pila `cambia`, que reciba un número entero y cambie los dígitos que sean un divisor de 2, o sea, 2, 4, 6, 8 por un cero, retornando un número entero.

   ```
   >>> cambia(1488)        >>> cambia(73571)
   1000                    73571
   >>> cambia(4275)
   75
   ```

LISTAS: hacer las funciones recursivas sobre listas:

- Usando slicing
- Usando índices.

5. Escriba una función a llamar `iota(num)` que reciba un número natural y obtenga una lista en orden ascendente de números naturales menores al número dado.

```
>>> iota(5)
[0, 1, 2, 3, 4]
>>> iota(1)
[0]
```

6. Escriba una función `cuales` que reciba un número (que debe ser entero) y retorne una lista con los dígitos entre 0 y 4.

```
>>> cuales(482401)      >>> cuales(0)
([4,2,4,0,1])          ([0])
>>> cuales(97)
([])
```

7. Escriba una función `hay_pares` que reciba una lista y retorne un boolean indicando si al menos hay un número par.

```
>>> hay_par([4,8,2,4,0,1])
True
>>> hay_par([9, 7])
False
```

8. Escriba una función a llamar `cambiar(lista, ele)` que reciba una lista y un elemento y cambie por un cero la primera aparición de ese elemento en la lista.

```
>>> cambiar([0, 5, 1, 2, 5, 3, 4], 5)
[0, 0, 1, 2, 5, 3, 4]
>>> cambiar([0, 5, 1, 2, 5, 3, 4], 8)
[0, 5, 1, 2, 5, 3, 4]
```

9. Escriba una función a llamar `cambiar_todos(lista, ele)` que reciba una lista y un elemento y cambie por un cero todas las apariciones de ese elemento en la lista.

```
>>> cambiar_todos([0, 5, 1, 2, 5, 3, 4], 5)
[0, 0, 1, 2, 0, 3, 4]
>>> cambiar_todos ([0, 5, 1, 2, 5, 3, 4], 8)
[0, 5, 1, 2, 5, 3, 4]
```

10. Se desea recibir una lista no nula y devolver otra compuesta por dos sublistas, la primera tendrá a los elementos pares, mientras que la segunda contendrá a los impares.

```
>>> separa([12,14,16,20,30] )    >>> separa([12,1,20,3])
[[12,14,16,20,30], []]           [[12,20],[1,3]]
```