Loops y recursión

Rangos (range)

-Representan una secuencia de números.
-Se definen con un límite inferior y un límite superior.
-Son inclusivos.
-Se separan con dos puntos (..).
-Son equivalentes a una lista:
  1..10 -> [1,2,3,4,5,6,7,8,9,10]
-Es más eficiente que una lista de números secuenciales, puesto que solo se almacenan dos enteros, el del inicio y el del final.
-Son enumerables, cada número se genera conforme se itere sobre el rango.
-La función Enum puede usarse con rangos.

Ejemplo:

```
iex> 1..01
1..1
iex> 1..10
1..10
iex> li..ls = 1..10
1..10
iex> li
1
iex> ls
10
iex> li = 10
10
iex> ls = 20
20
iex> li..ls
10..20
iex> ls..li
20..10
```

Se puede generar una lista a partir de un rango:

```
iex> Enum.to_list(1..10)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
iex> Enum.to_list(10..1)
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

Contar cuantos elementos hay en el rango:

```
iex> rango = 10..25
10..25
iex> Enum.count(rango)
16
```

Determinar si un elemento x se encuentra dentro del rango:

```
iex> rango = 10..25
10..25
iex> Enum.member?(rango,9)
false
iex> Enum.member?(rango,20)
true
```

Otra forma de saberlo es con el operador *in*:

```
iex> 9 in rango
false
iex> 20 in rango
true
```

Funciones de *Enum*:

```
iex(1)> Enum.
EmptyError              OutOfBoundsError        all?/1
all?/2                  any?/1                  any?/2
at/2                    at/3                    chunk_by/2
chunk_every/2           chunk_every/3           chunk_every/4
chunk_while/4           concat/1                concat/2
count/1                 count/2                 dedup/1
dedup_by/2              drop/2                  drop_every/2
drop_while/2            each/2                  empty?/1
fetch!/2                fetch/2                 filter/2
find/2                  find/3                  find_index/2
find_value/2            find_value/3            flat_map/2
flat_map_reduce/3       frequencies/1           frequencies_by/2
group_by/2              group_by/3              intersperse/2
into/2                  into/3                  join/1
join/2                  map/2                   map_every/3
map_intersperse/3       map_join/2              map_join/3
map_reduce/3            max/1                   max/3
max_by/2                max_by/4                member?/2
min/1                   min/3                   min_by/2
min_by/4                min_max/1               min_max/2
min_max_by/2            min_max_by/3            random/1
reduce/2                reduce/3                reduce_while/3
reject/2                reverse/1               reverse/2
```

```
reverse_slice/3      scan/2               scan/3
shuffle/1            slice/2              slice/3
sort/1               sort/2               sort_by/2
sort_by/3            split/2              split_while/2
split_with/2         sum/1                take/2
take_every/2         take_random/2        take_while/2
to_list/1            uniq/1               uniq_by/2
unzip/1              with_index/1         with_index/2
zip/1                zip/2
```