ENGIN 604 Introducción a Python para las Finanzas
Estructura de Control Iterativo

Profesor: *Gabriel E. Cabrera*
Ayudante: *Alex Den Braber*

1. *Loop* (ciclo) con & sin `range()`

1. Construya una lista que contenga el nombre de las siguientes empresas tecnológicas: 'Facebook', 'Apple', 'Amazon' y 'Netflix'.
   a. Muestre en la consola cada elemento en la lista.
   b. Muestre en la consola cada elemento en la lista con su respectivo índice.
   c. Muestre en la consola cada letra que compone a cada elemento (empresa) en la lista.
2. Genere una secuencia desde el 0 hasta el 10 y muestre cada valor en la consola.
3. Genere una secuencia desde el 0 hasta el 10 y muestre cada valor en la consola de mayor a menor.

2. Condicionales

1. Construya la siguiente secuencia de números incluyendo los `None`: 1, 2, `None`, 4, 5, 6, `None`. Luego utilizando un *loop* (ciclo) sume los elementos que son númericos (no considerar `None`)
2. Construya la siguiente secuencia de números incluyendo los `None`: 1, 2, `None`, 4, 5, 6, `None`. Luego utilizando un *loop* (ciclo) sume todos los elementos númericos antes del primer `none`.
3. Construya la siguiente secuencia de números: 1, 2, 3, 4, 5, 6, 7. Luego utilizando un *loop* (ciclo) sume los elementos menores al número 5.
4. Sume todo los números desde 0 hasta 15 que sean múltiplos de 3 o 5.
5. Construya un *loop* (ciclo) que genere la secuencia de Fibonacci la que se caracteriza porque cada número es la suma de los dos números previos. Por ejemplo los 14 primeros número de la secuencia son: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233.

3. *Comprehension*

Una "*list comprehension*" permite formar nuevas listas (`list`) filtrando los elementos de una colección (secuencia) a partir de una sola expresión. Su forma tradicional es la siguiente:

```
[expr for val in collection if condition]
```

lo anterior es quivalente con:

```
res = []
for val in collection:
```

```
  if condition:
    result.append(expr)
```

Por ejemplo:

```
list2 = ['Facebook', 'Apple', 'Amazon', 'Netflix']

list2_title = [elem.upper() for elem in list2]
list2_title
```

```
## ['FACEBOOK', 'APPLE', 'AMAZON', 'NETFLIX']
```

Siendo equivalente en su forma estandar con:

```
list2_title2 = []

for elem in list2:
  list2_title2.append(elem.upper())

list2_title2
```

```
## ['FACEBOOK', 'APPLE', 'AMAZON', 'NETFLIX']
```

Incluso puedo agregar el condicionale `else` lo que cambia un poco la estructura de la *list comprehension* pero el objetivo es el mismo:

```
[num if (num%3 == 0) or (num%5 == 0) else None for num in range(16)]
```

```
## [0, None, None, 3, None, 5, 6, None, None, 9, 10, None, 12, None, None, 15]
```

Siendo equivalente en su forma estandar con:

```
res2 = []

for num in range(16):
  if (num%3 == 0) or (num%5 == 0):
    res2.append(num)
  else:
    res2.append(None)

res2
```

```
## [0, None, None, 3, None, 5, 6, None, None, 9, 10, None, 12, None, None, 15]
```

Tambien se puede utilizar para "aplanar" (*flatten*) listas anidadas:

```
nested_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

[sub_elem for elem in nested_list for sub_elem in elem]
```

```
## [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Siendo equivalente en su forma estandar con:

```
flattened_list = []

for elem in nested_list:
  for sub_elem in elem:
    flattened_list.append(sub_elem)

flattened_list
```

```
## [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Para ver *comprehension* de diccionarios (`dict`) y tuplas (`tuple`) ver la página 67 y 68 del libro *Python for Data Analysis: Data Wrangling with Pandas, Numpy, and Ipython* de Wes McKinney.