



ENGIN 604 INTRODUCCIÓN A PYTHON PARA LAS FINANZAS

ESTRUCTURA DE CONTROL ITERATIVO - PAUTA

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.

```
lista1 = ['Facebook', 'Apple', 'Amazon', 'Netflix']

for tech in lista1:
    print('Empresa: '+tech)
```

```
## Empresa:Facebook
## Empresa:Apple
## Empresa:Amazon
## Empresa:Netflix
```

- b. Muestre en la consola cada elemento en la lista con su respectivo índice.

```
for i, tech in enumerate(lista1):
    print('índice: '+str(i)+' '+ 'Empresa: '+tech)
```

```
## índice:0,Empresa:Facebook
## índice:1,Empresa:Apple
## índice:2,Empresa:Amazon
## índice:3,Empresa:Netflix
```

- c. Muestre en la consola cada letra que compone a cada elemento (empresa) en la lista.

```
for tech in lista1:
    for i, letra in enumerate(tech):
        print('n°: '+str(i)+' '+ 'Letra: '+letra+' '+ 'Empresa: '+tech)
```

```
## n°:0,Letra:F,Empresa:Facebook
## n°:1,Letra:a,Empresa:Facebook
## n°:2,Letra:c,Empresa:Facebook
## n°:3,Letra:e,Empresa:Facebook
## n°:4,Letra:b,Empresa:Facebook
## n°:5,Letra:o,Empresa:Facebook
## n°:6,Letra:o,Empresa:Facebook
## n°:7,Letra:k,Empresa:Facebook
## n°:0,Letra:A,Empresa:Apple
```

```
## n°:1,Letra:p,Empresa:Apple
## n°:2,Letra:p,Empresa:Apple
## n°:3,Letra:l,Empresa:Apple
## n°:4,Letra:e,Empresa:Apple
## n°:0,Letra:A,Empresa:Amazon
## n°:1,Letra:m,Empresa:Amazon
## n°:2,Letra:a,Empresa:Amazon
## n°:3,Letra:z,Empresa:Amazon
## n°:4,Letra:o,Empresa:Amazon
## n°:5,Letra:n,Empresa:Amazon
## n°:0,Letra:N,Empresa:Netflix
## n°:1,Letra:e,Empresa:Netflix
## n°:2,Letra:t,Empresa:Netflix
## n°:3,Letra:f,Empresa:Netflix
## n°:4,Letra:l,Empresa:Netflix
## n°:5,Letra:i,Empresa:Netflix
## n°:6,Letra:x,Empresa:Netflix
```

2. Genere una secuencia desde el 0 hasta el 10 y muestre cada valor en la consola.

```
for num in range(0,11):
    print(num)
```

```
## 0
## 1
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10
```

3. Genere una secuencia desde el 0 hasta el 10 y muestre cada valor en la consola de mayor a menor.

```
for num in reversed(range(0,11)):
    print(num)
```

```
## 10
## 9
## 8
## 7
## 6
## 5
## 4
## 3
## 2
## 1
## 0
```

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 numéricos (no considerar `None`)

```
seq = [1, 2, None, 4, 5, 6, None]

total = 0 # variable igual a cero

for j in seq:
    if j == None:
        continue
    else:
        total += j

print('La suma total es: '+str(total))
```

```
## La suma total es: 18
```

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 numéricos antes del primer `none`.

```
seq1 = [1, 2, None, 4, 5, 6, None]

# variable igual a cero
total1 = 0

for j in seq1:
    if j == None:
        break
    else:
        total1 += j

print('La suma total es: '+str(total1))
```

```
## La suma total es: 3
```

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.

```
seq2 = list(range(1,8))

# forma 1
total2 = 0

for j in seq2:
    if j < 5:
        total2 += j
    elif j == 5:
        print(str(j)+' es igual a 5')
    else:
        print(str(j)+' es mayor a 5')
```

```
## 5 es igual a 5
## 6 es mayor a 5
## 7 es mayor a 5
```

```
total2
```

```
## 10
```

4. Sume todo los números desde 0 hasta 15 que sean múltiplos de 3 o 5.

```
seq2 = range(16)

total3 = 0

for k in seq2:
    if (k%3 == 0) or (k%5 == 0):
        total3 += k
    else:
        print(str(k)+' no es múltiplo de 3 o 5')
```

```
## 1 no es múltiplo de 3 o 5
## 2 no es múltiplo de 3 o 5
## 4 no es múltiplo de 3 o 5
## 7 no es múltiplo de 3 o 5
## 8 no es múltiplo de 3 o 5
## 11 no es múltiplo de 3 o 5
## 13 no es múltiplo de 3 o 5
## 14 no es múltiplo de 3 o 5
```

```
total3
```

```
## 60
```

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.

```
fib = []

fib.append(0)
fib.append(1)

x = 1

while x <= 12:
    res = fib[x] + fib[x - 1]
    fib.append(res)
    x += 1

fib
```

```
## [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 equivalente 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 condicional `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.