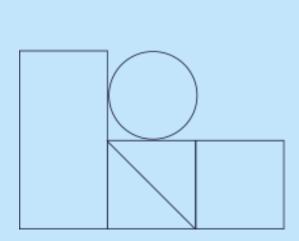
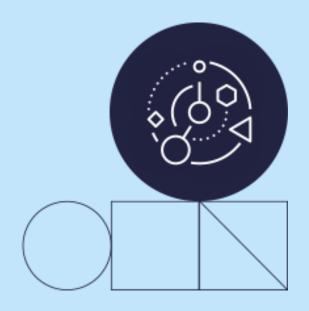
Programando en Python

Ejemplo de complejidad y Big-O





Índice

Genero una lista de 100 millones de números aleatorios	3
Compejidad Constante O(k) Constant Time	3
Complejidad O(n) Linear Time	3
Complejidad O(n²) Quadratic Time	4
Complejidad O(log(n)) Logarithmic Time	4
Complejidad O(log(n)) Logarithmic Time	5
Complejidad O(log(log(n))) Logarithmic from Logarithmic Time	5
Cálculo de la complejidad Algorítmica del Ejemplo	6

Genero una lista de 100 millones de números aleatorios

```
import numpy as np
x =
list(np.random.randint(low=1,high=500000,
size=99999999))
```

Complejidad Constante O(k) Constant Time

```
%%time
def constante(x:list) -> list:
    return x

constante(x)
```

```
Wall time: 0 ns
[470443,
 337976,
 383958,
 404514,
 383032,
 427315,
 473003,
 225481,
 169894,
 475938,
 15745,
 210740,
 124051,
 192964,
 347048...
```

Complejidad O(n) Linear Time

```
%%time
def iterador_normal(x:list) -> list:
    contador = len(x)
    while(contador > 0):
        contador -= 1
    return x

iterador_normal(x)
```

Wall time: 5.31 s

```
[470443,
337976,
383958,
404514,
383032,
427315,
473003,
225481,
169894,
475938,
15745,
210740,
124051,
192964,
347048...
```

Complejidad O(n²) Quadratic Time

```
%%time
def iterador_anidado(x:list) -> list:
    contador_externo = len(x)//1000
    contador_interno = len(x)//1000

    while(contador_externo > 0):
        contador_externo -= 1

        for i in range(contador_interno):
              i
        return x

iterador_anidado(x)
```

```
Wall time: 5min 29s
[470443,
 337976,
 383958,
 404514,
 383032,
 427315,
 473003,
 225481,
 169894,
 475938,
 15745,
 210740,
 124051,
 192964,
 347048,
 311380...
```

Complejidad O(log(n)) Logarithmic Time

```
%%time
def iterador_multiplicado(x:list) ->
list:

   iterador = len(x)
   incremento = 1
   while(iterador > 0):
        iterador -= incremento
        incremento *= (incremento + 1)

   return x

iterador_multiplicado(x)
```

```
Wall time: 0 ns
[470443,
 337976,
 383958,
 404514,
 383032,
 427315,
 473003,
 225481,
 169894,
 475938,
 15745,
 210740,
 124051,
 192964,
 347048...
```

Complejidad O(log(n)) Logarithmic Time

```
%%time
def iterador_dividido(x:list) ->
list:

   iterador = -len(x)
   incremento = 1
   while(iterador < 0):
       iterador /= incremento
       incremento += 1

   return x

iterador_dividido(x)</pre>
```

```
Wall time: 0 ns
[470443,
 337976,
 383958,
 404514,
 383032,
 427315,
 473003,
 225481,
 169894,
 475938,
 15745,
 210740,
 124051,
 192964,
 347048,
 311380...
```

Complejidad O(log(log(n))) Logarithmic from Logarithmic Time

```
%%time
import math

def
iterador_incremento_exponencial(x:list) -
> list:

   iterador = len(x)
   incremento = 1
   while(iterador > 0):
       iterador -= pow(incremento, 2)
       incremento += 1

   return x

iterador_incremento_exponencial(x)
```

```
Wall time: 0 ns
[470443,
 337976,
 383958,
 404514,
 383032,
 427315,
 473003,
 225481,
 169894,
 475938,
 15745,
 210740,
 124051,
 192964,
 347048...
```

Cálculo de la complejidad Algorítmica del Ejemplo

```
list(np.random.randint(low=1,high=500000,
size=999))
%%time
def calculo_bit_o_ejemplo(y:list) ->
str:
    iterador = -len(y) # k
    incremento = 1 # k
    q_list = y # k
   while(iterador < 0): # log(n)</pre>
        iterador /= incremento # k
        incremento += 1 # k
    for p in y: # n
        for q in y: # n -> n * n
            for r in y: # n -> n * n * n
    return "La Complejidad es n*n*n, n
cubo"
calculo_bit_o_ejemplo(y)
```

Wall time: 14.7 s

'La Complejidad es n*n*n, n cubo'