

# **Recursividad**

## Computer Science

**CS1100 - Introducción a Ciencia de la Computación**

# Logro de la Sesión

Al finalizar esta sesión, estarás en la capacidad de:

- Desarrollar programas en Python, utilizando funciones y recursividad.

## Diseñando funciones:

Con los tipos de funciones que hemos estudiado, podemos resolver problemas como:

- Realizar cálculos sucesivos. Ejemplo: Calcular la suma de los cuadrados de 1 hasta  $N$  (ver ejemplo 01)
- Validar o verificar. Ejemplo: Encontrar el máximo en una lista (ver ejemplo 02)

# Función que realiza cálculos (ejemplo 01)

Crear una función para encontrar la suma de los cuadrados de 1 hasta N.

```
1  """Codigo que realiza la suma de cuadrados de 1 a N """
2  def suma_cuadrados(N):
3      suma = 0
4      for i in range(1, N + 1):
5          suma += i * i
6      return suma
7
8  num = int(input("Ingrese un numero: "))
9  print(suma_cuadrados(num))
```

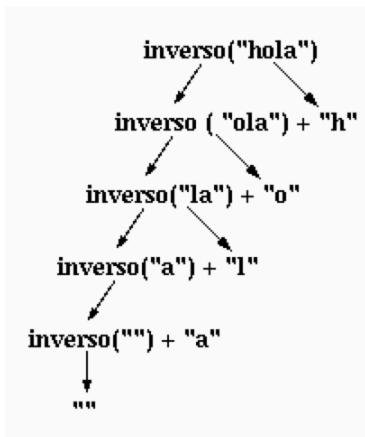
# Función que realiza validaciones (ejemplo 02)

## Encontrar el máximo en una lista

```
1 def Max(lista):
2     maximo = lista[0]
3     for num in lista:
4         if num > maximo:
5             maximo = num
6     return maximo
7
8 lista = list()
9 while True:
10     num = int(input("Ingrese otro numero (termine con 0): "))
11     if num != 0:
12         break
13     lista.append(num)
14
15 print("El numero maximo es: ", Max(lista))
```

# Algoritmo recursivo

Es un algoritmo que expresa la solución de un problema en términos de una llamada a si mismo (llamada recursiva o recurrente)



# Autoreferencia

La llamada de la función a si misma hace que se imprima el mensaje un número indeterminado de veces

```
1 def foo(s):  
2     print(s)  
3     foo(s)  
4  
5 foo('hola mundo')
```

El límite de recursividad previene una saturación de la memoria. Se obtiene el error:

RecursionError: maximum recursion depth exceeded while calling a Python object

## Condición de salida:

La llamada de la función a si misma hace que se imprima el mensaje cada vez con un caracter menos

```
1 def foo(s):  
2     if len(s) == 1:  
3         print(s)  
4     else:  
5         print(s)  
6         s = s[1:]  
7         foo(s)  
8  
9 foo('hola mundo')
```



# Recursividad en algoritmos: Suma de números

Forma recursiva de calcular la suma de dos números naturales  $a$  y  $b$ .

$$\text{suma}(a, b) = \begin{cases} \text{Si } b = 0, \text{ retornar } a \\ \text{Retornar } 1 + \text{suma}(a, b - 1) \end{cases} \quad (1)$$

```
1  """ Suma recursiva"""
2  def suma(a, b):
3      if b == 0:
4          return a
5      else:
6          return 1 + suma(a, b-1)
7
8  print(suma(21, 3))
```



# Recursividad en algoritmos: Suma de cuadrados

Forma recursiva de calcular la suma de cuadrados de 1 a  $n$  (ver ejemplo 01)

```
1 def suma_cuadrados(n):  
2     if n == 1:  
3         return 1  
4     else:  
5         return n*n + suma_cuadrados(n-1)  
6  
7 num = int(input("Ingrese un numero: "))  
8 print(suma_cuadrados(num))
```

## Recursividad en algoritmos: Factorial de un número

Forma recursiva de calcular el factorial de un número  $n$ .

$$N! = N \times (N - 1) \times (N - 2) \times \dots \times 2 \times 1$$

$$\text{factorial}(n) = \begin{cases} \text{si } n = 1, \text{ retornar } 1 \\ \text{Retornar } n \times \text{factorial}(n - 1) \end{cases} \quad (2)$$

# Ejercicio 1

## Enunciado

Usando la definición anterior implementar la función factorial.

## Ejercicio 2

### Enunciado

Crear una función recursiva que devuelva un numero a elevado a la potencia b.

## Ejercicio 3

### Enunciado

Escribir una función recursiva que encuentre el mayor elemento de una lista

# Evaluación

## Individual Work

- [www.hackerrank.com/cs1100-lab-01](http://www.hackerrank.com/cs1100-lab-01)

# Cierre

En esta sesión aprendiste:

- Desarrollar programas en Python, utilizando funciones y recursividad.