# Ayudantía 5 - Recursión

November 10, 2022

<

**Fundamentos de Programación - Sección E-3**
**Profesor:** Cristian Sepúlveda - cristian.sepulvedas@usach.cl
**Ayudante:** John Serrano - john.serrano@usach.cl

## 1 ¿Que es recursión?

Si quieres saber lo que es recursión... puedes debes entender recursión

Recursión es practicamente una nueva forma de hacer ciclos utilizando funciones. Consiste en que practicamente, una función se llama a si misma, con algún cambio en sus parametros. Esto se realiza para ir achicando el problema hasta un problema mas conocido y usualmente se utiliza en problemas donde se debe repetir un mismo cálculo varias veces. Tenemos dos casos en una función recursiva: * Caso Base: Consiste en un problema muy pequeño, donde la solución es conocida o "trivial". Puede haber mas de un caso base. * Caso Recursivo: Es un problema que depende de la solución de otros problemas mas pequeños. Usualmente hay múltiples casos recursivos.

## 2 Ejemplo 1: La Sumatoria

Considerando la sumatoria de los primeros 100 terminos, se busca sumar los 100 primeros terminos. Se debe resolver una pequeña parte del ejercicio y dejar el resto en las mismas palabras del ejercicio original. Sabemos que:

$$\sum_{i=1}^{100} i = 100 + \sum_{i=1}^{99} i$$

Pero sabemos que la sumatoria de los primeros 99 terminos es:

$$\sum_{i=1}^{99} i = 99 + \sum_{i=1}^{98} i$$

Notemos que los casos anteriores corresponden a **casos recursivos**, ya que la solución va dependiendo de la solución de problemas chicos (la sumatoria de 100 depende de la sumatoria de 99. La sumatoria de 99 depende de la sumatoria de 98, etc).

Entonces, ¿cuales seria el **caso base**? Pues como corresponden a problemas cuya solución es conocida, siguiendo con el ejemplo de las sumatorias, sabemos que la sumatoria de 1 es simplemente 1.

$$\sum_{i=1}^{1} i = 1$$

Por lo tanto, todo lo anterior se traduce a lo siguiente:

- Si i > 1, entonces la sumatoria de i terminos es es: i + la sumatoria de (i-1)
- Si i == 1, entonces la sumatoria de i es 1.

Lo anterior corresponde a la **ecuación de recurrencia**. Llevando esto a un código de Python:

[1]:
```
# BLOQUE DE DEFINICIONES
# DEFINICION DE FUNCIONES

'''
Entrada: Un numero entero (Integer) mayor o igual a 1
Salida: La sumatoria de los i terminos
Descripcion: Funcion recursiva que calcula la sumatoria de i terminos, cuando
el indice inicial es igual a 1.
'''
def sumatoria(i):
    if i == 1:
        return 1 # CASO BASE
    elif i == 0:
        return 0
    else:
        return i + sumatoria(i-1)  # CASO RECURSIVO


# BLOQUE PRINCIPAL
# ENTRADA
numero = int(input("Ingrese un numero entero mayor o igual a 1: "))

# PROCESAMIENTO
resultado_sumatoria = sumatoria(numero)

# SALIDA
print("El resultado de la sumatoria de", numero, "terminos es: ",␣
 ↪resultado_sumatoria)
```

```
Ingrese un numero entero mayor o igual a 1: 100
El resultado de la sumatoria de 100 terminos es:  5050
```

Hay ejercicios que son más faciles resolviendolos con recursión. Otros, son mas faciles con iteraciones (Ciclo While). Todo va a depender del contexto del problema y de lo que nos soliciten. **Para la PEP, pueden solicitar utilizar recursión, o bien resolver tanto con iteraciones y con recursión.**

### 2.1 Precaución: Error de Stack / RecursionError

Si bien la recursión es una buena herramienta, debemos tener cuidado con el uso de esta, ya que si no sabemos utilizarla bien, nuestra función no está muy bien definida o introducimos las entradas incorrectamente, podemos obtener como resultado un error de stack. Esto sucede cuando se realizan muchas llamadas recursivas y la memoria (Stack) se llena, a tal punto que es imposible continuar con la ejecución del código.

```
    if 1 == 1:
RecursionError: maximum recursion depth exceeded in comparison
```

## 3 Ejemplo 2: Factorial

El factorial se puede definir de la siguiente forma:

$$n! = \begin{cases} 1 & si \quad n = 1 \\ n \cdot (n-1)! & si \quad n \neq 1 \end{cases}$$

En Python, lo anterior es:

[2]:
```python
# BLOQUE DE DEFINICIONES
# DEFINICION DE FUNCIONES

'''
Entrada: Un numero entero (Integer) mayor o igual a 1
Salida: El factorial de n
Descripcion: Funcion recursiva que calcula el factorial de n
'''
def factorial(n):
    if n == 1:
        return 1 # CASO BASE
    else:
        return n * factorial(n-1)  # CASO RECURSIVO


# BLOQUE PRINCIPAL
# ENTRADA
numero = int(input("Ingrese un numero entero mayor o igual a 1: "))

# PROCESAMIENTO
resultado_factorial = factorial(numero)

# SALIDA
print("El factorial de", numero, "es:", resultado_factorial)
```

```
Ingrese un numero entero mayor o igual a 1: 5
El factorial de 5 es: 120
```

El factorial de 5 es:

$$5! = 5 \cdot 4!$$

$$4! = 4 \cdot 3!$$

$$3! = 3 \cdot 2!$$

$$2! = 2 \cdot 1!$$

$$1! = 1$$

Como ya llegamos a un caso base, ahora debemos resolver aquellos problemas que quedaron pendientes

$$2! = 2 \cdot 1! = 2 \cdot 1 = 2$$

$$3! = 3 \cdot 2! = 3 \cdot 2 = 6$$

$$4! = 4 \cdot 3! = 4 \cdot 6 = 24$$

$$5! = 5 \cdot 4! = 5 \cdot 24 = 120$$

**Por lo tanto, la recursión consiste en basicamente, ir achicando un problema hasta llegar a problemas conocidos o faciles de resolver y luego devolverse y resolver aquellos problemas grandes que quedaron pendientes y que dependen de la solución de los problemas pequeños.**

# 4 EJERCICIOS

**1.** Escriba en Python un programa que ordene de menor a mayor una lista de enteros ingresada por teclado utilizando la siguiente idea: ubique el menor numero de la lista en la primera posicion, y luego ordene el resto de la lista con una llamada recursiva.

**2.** Escriba un programa que calcule el maximo comun divisor de dos numeros enteros positivos ingresados por el usuario. Utilice para resolver el problema una funcion recursiva.

```
[ ]:
```