# 13/032023

lunes, 13 de marzo de 2023 15:10

Razonar lo que se aprende es una habilidad esencial que puede ayudarte a comprender y retener mejor la información que estás aprendiendo. Aquí hay algunos consejos para ayudarte a razonar sobre lo que estás aprendiendo:

**Haz preguntas**: Haz preguntas sobre lo que estás aprendiendo. Cuestiona el por qué y el cómo de las cosas. Intenta entender los conceptos detrás de los hechos y las cifras.

**Conecta con lo que ya sabes**: Conecta lo que estás aprendiendo con tus conocimientos previos. Intenta encontrar similitudes y diferencias entre los nuevos conceptos y lo que ya sabes.

**Analiza la información**: Analiza la información que estás aprendiendo. Desglosa los conceptos complejos en piezas más pequeñas y trata de entender cómo se relacionan entre sí.

Relaciona la información: Relaciona la información que estás aprendiendo con el mundo que te rodea. Piensa en cómo los conceptos que estás aprendiendo se aplican en la vida real.

Prueba tu comprensión: Prueba tu comprensión haciendo preguntas y resolviendo problemas. Trata de explicar los conceptos a alguien más.

Practica y aplica: Practica lo que estás aprendiendo y aplica los conceptos a situaciones reales. La práctica te ayudará a retener la información y a entender cómo aplicarla en diferentes situaciones.

Reflexiona: Reflexiona sobre lo que has aprendido. Piensa en cómo podrías haber abordado el tema de manera diferente y qué has aprendido de ello.

Recuerda que razonar sobre lo que se aprende es una habilidad que se desarrolla con la práctica. Cuanto más practiques, más fácil será para ti comprender y retener la información que estás aprendiendo.

lunes, 13 de marzo de 2023 15:21

**Algoritmo: Pasos**

Los pasos algorítmicos para implementar la recursividad en una función son los siguientes:

Paso 1: defina un caso base: **identifique el caso más simple para el cual la solución es conocida o trivial.** Esta es la condición de parada para la recursividad, ya que evita que la función se llame a sí misma infinitamente.

Paso 2: defina un caso recursivo: d**efina el problema en términos de subproblemas más pequeños.** Divida el problema en versiones más pequeñas de sí mismo y llame a la función recursivamente para resolver cada subproblema.

Paso 3: **asegúrese de que finalice la recursividad**: asegúrese de que la función recursiva finalmente alcance el caso base y no ingrese a un ciclo infinito.

paso4 - Combina las soluciones: Combina las soluciones de los subproblemas para resolver el problema original.

**¿Cómo se resuelve un problema particular usando la recursividad?**

La idea es representar un problema en términos de uno o más problemas más pequeños y agregar una o más condiciones base que detengan la recursividad. Por ejemplo, calculamos el factorial n si conocemos el factorial de (n-1). El caso base para el factorial sería n = 0. Devolvemos 1 cuando n = 0.

**¿Cuál es la diferencia entre recursividad directa e indirecta?**

```
// Un ejemplo de recursividad directa
void directRecFun()
{
   // Algo de código....
   directRecFun();
   // Algo de código...
}
// Un ejemplo de recursividad indirecta
vacío indirectoRecFun1()
{
   // Algo de código...
   indirectaRecFun2();
   // Algo de código...
}
vacío indirectoRecFun2(){
   // Algo de código...
   indirectaRecFun1();
   // Algo de código...
}
```

Recursividad

El proceso en el que una función se llama a sí misma directa o indirectamente se llama recursividad y la función correspondiente se llama función recursiva

***El backtracking** se puede definir com o una técnica algorítmica general que considera buscar todas las combinaciones posibles para resolver un problema computacional.*

Desde <https://www.geeksforgeeks.org/backtracking-algorithms/>

¿Qué es el algoritmo de retroceso?
El backtracking es una técnica algorítmica para resolver problemas recursivamente al tratar de construir una solución incrementalmente, una pieza a la vez, eliminando aquellas soluciones que fallan en satisfacer las restricciones del problema en cualquier punto del tiempo (por tiempo, aquí, se refiere a la tiempo transcurrido hasta alcanzar cualquier nivel del árbol de búsqueda).

Propiedades recursivas
Realiza las mismas operaciones viarias con diferentes entradas.

Cada paso intentamos entradas más pequeñas para hacer que el problema sea más pequeño
Se necesita un condición base para detener la recursividad

# Ejemplo

lunes, 13 de marzo de 2023 17:43

```
# A Python 3 program to
# demonstrate working of
# recursion


def printFun(test):

    if (test < 1):
        return
    else:

        print(test, end=" ")
        printFun(test-1)  # statement 2
        print(test, end=" ")
        return

# Driver Code
test = 3
printFun(test)

# This code is contributed by
# Smitha Dinesh Semwal
```

Desde <https://www.ge

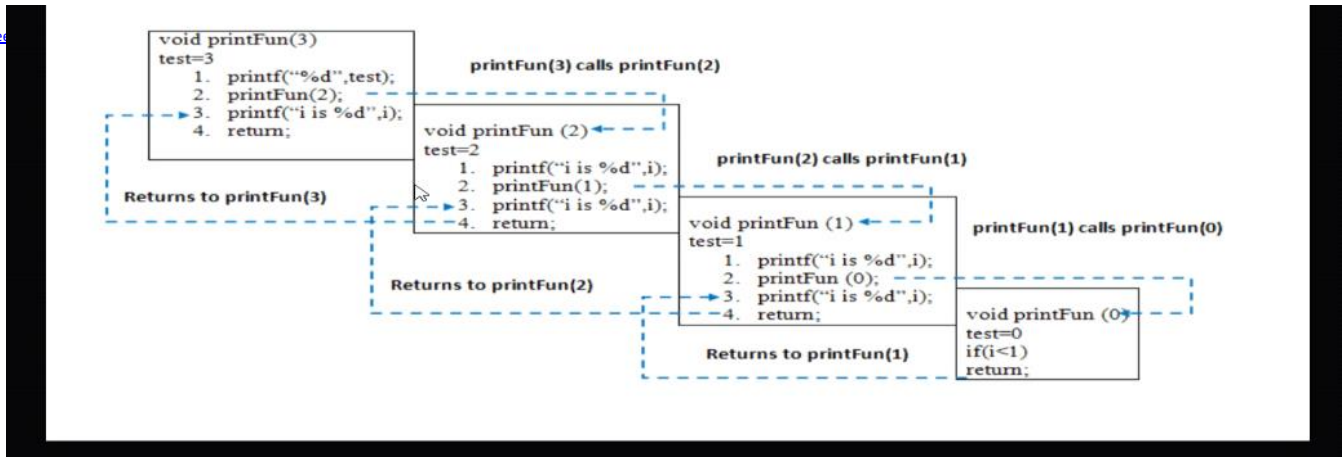
**Producción :**

```
3 2 1 1 2 3
```

**Tiempo Complejidad:** O(1)

**Espacio Auxiliar:** O(1)

Cuando se llama a **printFun(3)** desde main(), la memoria se asigna a **printFun(3)** y una prueba de variable local se inicializa a 3 y las declaraciones 1 a 4 se colocan en la pila como se muestra en el diagrama a continuación. Primero imprime '3'. En la declaración 2, se llama a **printFun(2)** y se asigna memoria a **printFun(2)** y una prueba de variable local se inicializa a 2 y las declaraciones 1 a 4 se colocan en la pila. De manera similar, **printFun(2)** llama **a printFun(1)** y **printFun(1)** llama **a printFun(0)** . **printFun(0)** va a la sentencia if y vuelve a **printFun(1)** . Las sentencias restantes de **printFun(1)** se ejecutan y vuelve a **printFun(2)** y así sucesivamente. En la salida, se imprimen los valores de 3 a 1 y luego se imprimen de 1 a 3. La pila de memoria se muestra en el siguiente diagrama.



# Escriba un programa y una relación de recurrencia para encontrar el factorial de n donde n>2.

lunes, 13 de marzo de 2023 18:05

A n se le guarda un espacio en la pila de memoria
La f(n-1) va ir decrementando de eso se trata la recurrencia de datos

**Ecuación matemática:**

```
1 si n == 0 o n == 1;
f(n) = n*f(n-1) si n > 1;
```

**Relación de recurrencia:**

```
T(n) = 1 para n = 0
T(n) = 1 + T(n-1) para n > 0
```

**Programa Recursivo:**

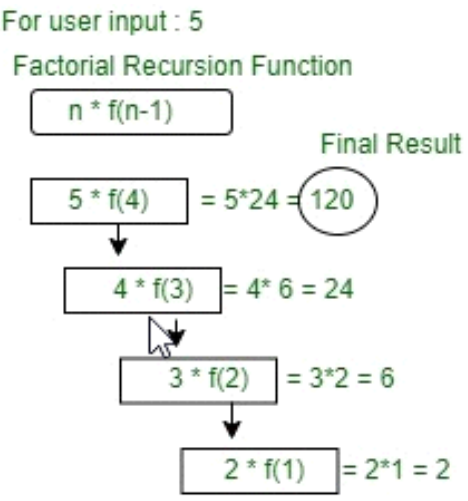
**Entrada:** n = 5

**Salida:**

factorial de 5 es: 120

**Implementación:**

```python
# Python3 code to implement factorial

# Factorial function
def f(n):

    # Stop condition
    if (n == 0 or n == 1):
        return 1;

    # Recursive condition
    else:
        return n * f(n - 1);


# Driver code
if __name__=='__main__':

    n = 5;
    print("factorial of",n,"is:",f(n))

    # This code is contributed by pratham76.
```



lunes, 13 de marzo de 2023 18:18

```python
# escriba un programa donde el factuarail de n sea >2
#5! = 5 x 4 x 3 x 2 x 1 = 120
#la idea es que se vaya decrementando hasta cuando llegue a 0 or 0

def factorial(n):
 # n se guarda en la pila y se va ir decrementando
    if (n == 0 or n == 1):#cuando => n=0 or n=1    :la funcion se detiene
        return 1
    else:
        return  n * factorial(n - 1)#esta funcion va ir decrementando  5,4 3,2,1 hasta que llegue a 1
                                    #f(n -1)SU VALOR INICIAL = 4
                                    #n = 5

if __name__=='__main__':
    n=5
    print("el facotrial de ", n , " is" , factorial(n))
```