



## Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación  
Salas A y B

*Profesor:*

Martínez Quintana Marco Antonio

*Asignatura:*

Estructura de Datos y Algoritmos 1.

*Grupo:*

17

*No de Práctica(s):*

12

*Integrante(s):*

Ruiz Godoy Franco

*No. de Equipo de  
cómputo empleado:*

7

*No. de Lista o Brigada:*

*Semestre:*

2020-2

*Fecha de entrega:*

02 de Mayo de 2020

*Observaciones:*

**CALIFICACIÓN:**

## Objetivo.

El objetivo de esta guía es aplicar el concepto de recursividad para la solución de problemas.

## Introducción.

### Recursividad

El propósito de la recursividad es dividir un problema en problemas más pequeños, de tal manera que la solución del problema se vuelva trivial.

Para aplicar recursión se deben de cumplir tres reglas:

- Debe haber uno o más casos base.
- La expansión debe terminar en un caso base.
- La función se debe llamar a sí misma.

Para resolver un problema por medio de recursividad hay que generar problemas más pequeños.

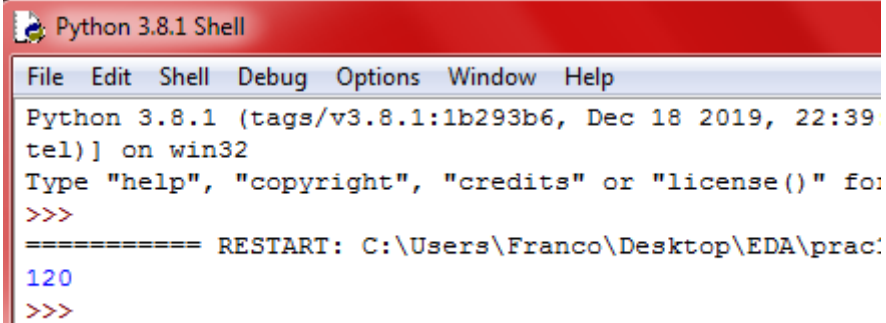
### Desventajas de la recursividad:

- A veces es complejo generar la lógica para aplicar recursión.
- Hay una limitación en el número de veces que una función puede ser llamada, tanto en memoria como en tiempo de ejecución.

## Desarrollo.

### Factorial iterativa.

```
def factorial_no_recursivo(numero):  
    fact = 1  
    for i in range(numero, 1, -1):  
        fact *= i  
    return fact  
  
print(factorial_no_recursivo(5))
```



Python 3.8.1 Shell

File Edit Shell Debug Options Window Help

Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:04) on win32  
Type "help", "copyright", "credits" or "license()" for more  
>>>  
===== RESTART: C:\Users\Franco\Desktop\EDA\prac:  
120  
>>>

## Factorial Recursivo.

```
File Edit Format Run Options Window Help
def factorial_recursivo(numero):
    if numero < 2:
        return 1
    return numero * factorial_recursivo(numero - 1)
print(factorial_recursivo(5))

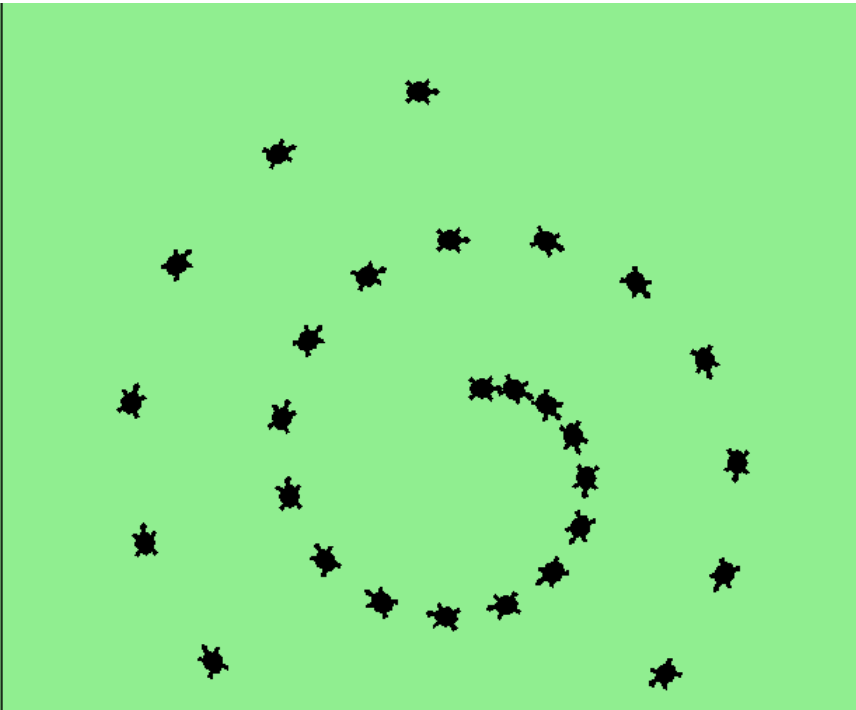
Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:
tel)] on win32
Type "help", "copyright", "credits" or "license()" fo
>>>
===== RESTART: C:/Users/Franco/Desktop/EDA/prac
120
>>> |
```

## Huellas de tortuga

```
import turtle
wn = turtle.Screen()
wn.bgcolor("lightgreen")
tess = turtle.Turtle()
tess.shape("turtle")
tess.color("black")

tess.penup()
size = 20
for i in range(30):
    tess.stamp()
    size = size + 3
    tess.forward(size)
    tess.right(24)

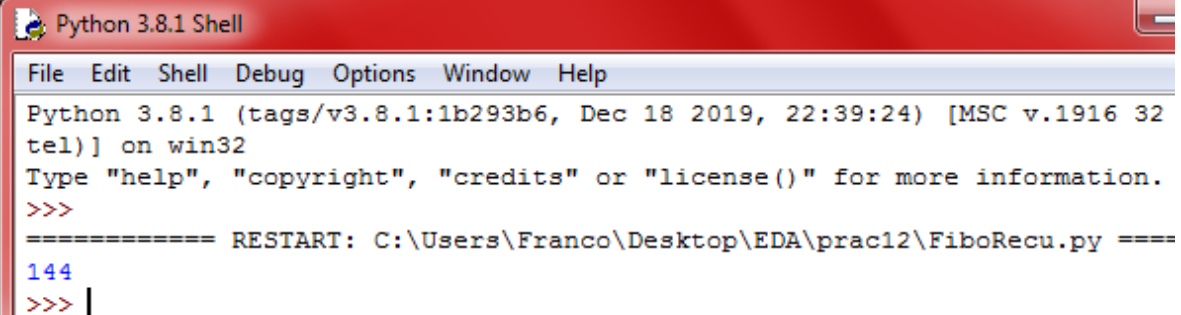
wn.mainloop()
```



# Fibonacci.

## Recursivo.

```
def fibonacci_recursivo(numero):  
    if numero == 1:  
        return 0  
    if numero == 2 or numero == 3:  
        return 1  
    return fibonacci_recursivo(numero-1) + fibonacci_recursivo(numero-2)  
print (fibonacci_recursivo(13))
```



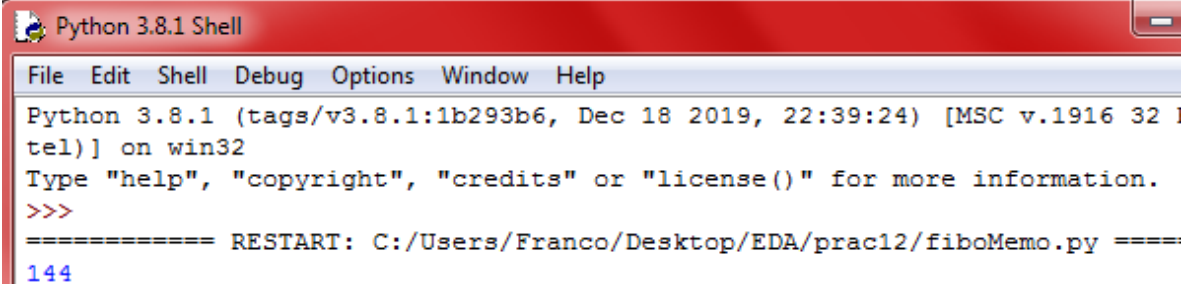
Python 3.8.1 Shell

File Edit Shell Debug Options Window Help

Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: C:\Users\Franco\Desktop\EDA\prac12\FiboRecu.py =====  
144  
>>> |

## Memorización.

```
memoria = {1:0, 2:1, 3:1}  
def fibonacci_memo(numero):  
    if numero in memoria:  
        return memoria[numero]  
    memoria[numero] = fibonacci_memo(numero-1) + fibonacci_memo(numero-2)  
    return memoria[numero]  
print(fibonacci_memo(13))
```



Python 3.8.1 Shell

File Edit Shell Debug Options Window Help

Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 22:39:24) [MSC v.1916 32 bit] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: C:/Users/Franco/Desktop/EDA/prac12/fiboMemo.py =====  
144

## Conclusión.

Como ya se sabe la recursividad es una función que consiste en llamarse a sí misma, además que la complejidad crece de forma exponencial en relación a la cantidad de llamados recursivos que hagamos. Trae ventajas la solución a problemas usuales y son de codificación corta pero a pesar de esto esta viene con soluciones complejas para generar la lógica para aplicar recursión y la limitación en el número de veces que una función puede ser llamada, tanto en memoria como en tiempo de ejecución.

## Referencias.

- Wentworth , P. W., Elkner, J. E., Downey , A. B. D., & Meyers, C. M. (s. f.). 3. Hello, little turtles! — How to Think Like a Computer Scientist: Learning with Python 3. Recuperado 2 de mayo de 2020, de [http://openbookproject.net/thinkcs/python/english3e/hello\\_little\\_turtles.html](http://openbookproject.net/thinkcs/python/english3e/hello_little_turtles.html)
- García Cano, E. E. G. C., & Solano Gálvez, J. A. S. G. (2017b, enero 20). Guía práctica de estudio 12: Recursividad. Recuperado 2 de mayo de 2020, de [http://lcp02.fi-b.unam.mx/static/docs/PRACTICAS\\_EDA1/eda1\\_p12.pdf](http://lcp02.fi-b.unam.mx/static/docs/PRACTICAS_EDA1/eda1_p12.pdf)
-