



## Caratula para entrega de Prácticas

Facultad de Ingeniería

Laboratorio de docencia

## Laboratorios de computación salas A y B

Profesor: \_\_\_\_\_ Ing. Marco Antonio Martinez Quintana \_\_\_\_\_

Asignatura: \_\_\_\_\_ Estructura de Datos y Algoritmos I (1227) \_\_\_\_\_

Grupo: \_\_\_\_\_ 17 \_\_\_\_\_

No. de Práctica(s): \_\_\_\_\_ 12° \_\_\_\_\_

Integrante(s): \_\_\_\_\_ Avila Laguna Ricardo \_\_\_\_\_

No. de Equipo de  
cómputo empleado: \_\_\_\_\_ 10 \_\_\_\_\_

No. Lista o Brigada: \_\_\_\_\_ 6 \_\_\_\_\_

Semestre: \_\_\_\_\_ 2° \_\_\_\_\_

Fecha de entrega: \_\_\_\_\_ Abril del 2020 \_\_\_\_\_

Observaciones: \_\_\_\_\_

CALIFICACIÓN: \_\_\_\_\_

## 1° Objetivos

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

## 2° Introducción

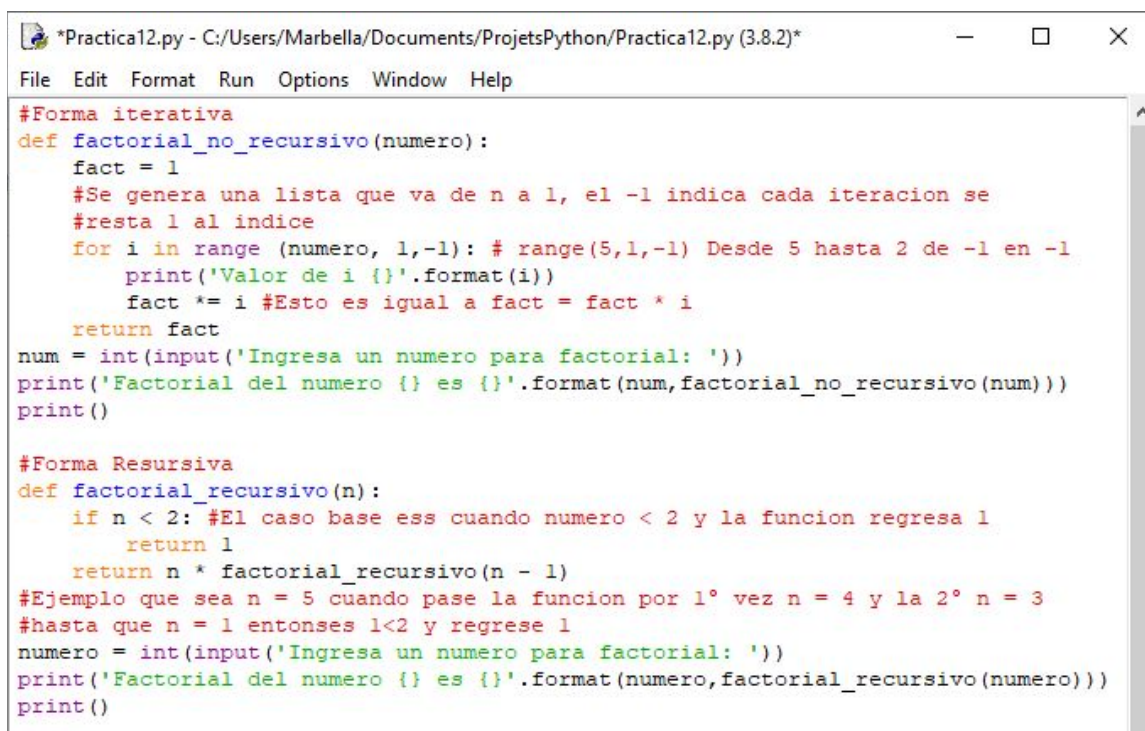
El propósito de la recursividad es dividir un problema en problemas más pequeños, de tal manera que la solución del problema se vuelva trivial. Básicamente, la recursión se puede explicar como una función que se llama así misma.

## 3° Desarrollo y Resultados

### Actividades:

- Revisar el concepto y las reglas de la recursividad y sus implicaciones.
- Ejecutar programas guardados en archivos desde la notebook.

### Código:

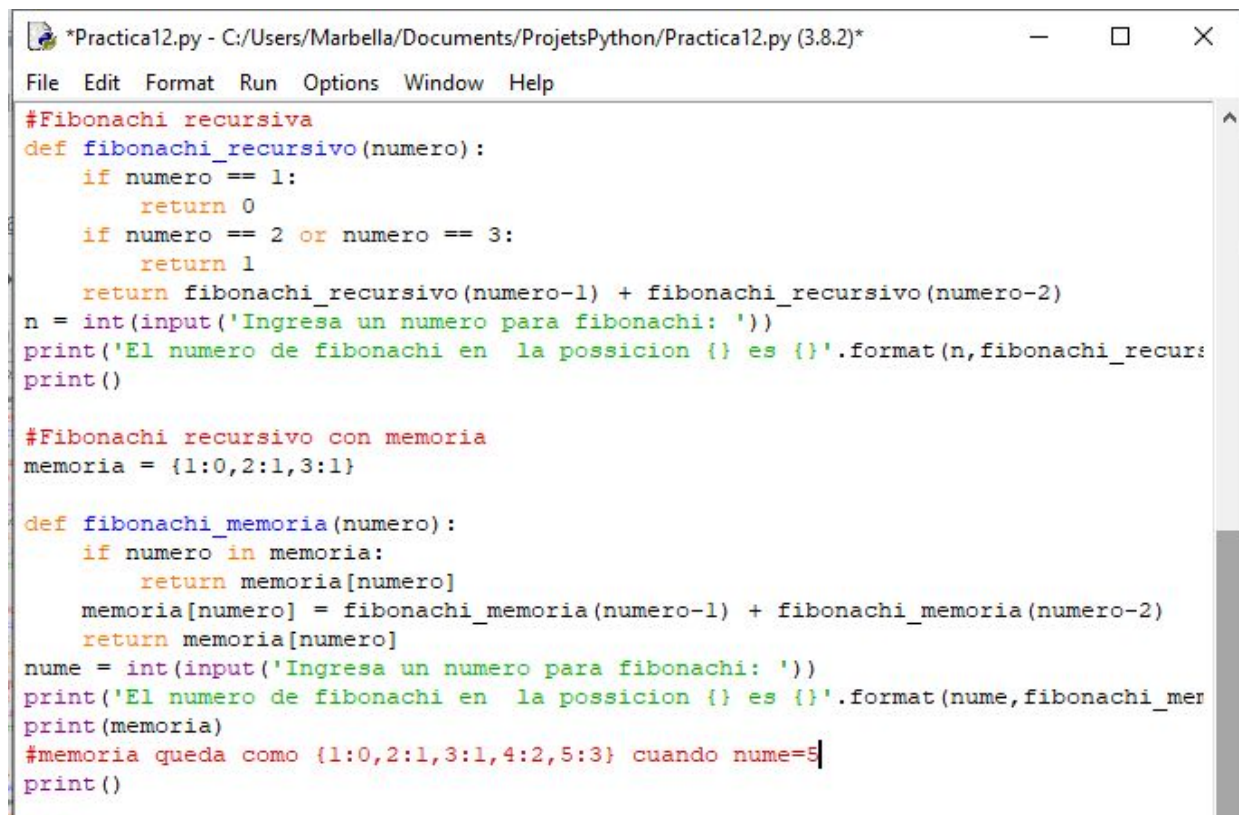


```
*Practica12.py - C:/Users/Marbella/Documents/ProjetsPython/Practica12.py (3.8.2)*
File Edit Format Run Options Window Help

#Forma iterativa
def factorial_no_recursivo(numero):
    fact = 1
    #Se genera una lista que va de n a 1, el -1 indica cada iteracion se
    #resta 1 al indice
    for i in range (numero, 1,-1): # range(5,1,-1) Desde 5 hasta 2 de -1 en -1
        print('Valor de i {}'.format(i))
        fact *= i #Esto es igual a fact = fact * i
    return fact
num = int(input('Ingresa un numero para factorial: '))
print('Factorial del numero {} es {}'.format(num,factorial_no_recursivo(num)))
print()

#Forma Resursiva
def factorial_recursivo(n):
    if n < 2: #El caso base ess cuando numero < 2 y la funcion regresa 1
        return 1
    return n * factorial_recursivo(n - 1)
#Ejemplo que sea n = 5 cuando pase la funcion por 1° vez n = 4 y la 2° n = 3
#hasta que n = 1 entonses 1<2 y regrese 1
numero = int(input('Ingresa un numero para factorial: '))
print('Factorial del numero {} es {}'.format(numero,factorial_recursivo(numero)))
print()
```

Para la práctica número 12 aprendimos la importancia de implementar la creatividad dentro de los programas en Python, entendimos las ventajas y desventajas que conlleva y lo aplicamos comparando varias funciones como la actual donde para obtener el factorial de un número utilizamos ciclos para lograrlo, pero con la recursividad solo será necesario plantear un caso base y volver a llamar a la misma función.



```
*Practica12.py - C:/Users/Marbella/Documents/ProjetsPython/Practica12.py (3.8.2)*
File Edit Format Run Options Window Help

#Fibonacci recursiva
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)
n = int(input('Ingresa un numero para fibonacci: '))
print('El numero de fibonacci en la position {} es {}'.format(n, fibonacci_recursivo(n)))
print()

#Fibonacci recursivo con memoria
memoria = {1:0, 2:1, 3:1}

def fibonacci_memoria(numero):
    if numero in memoria:
        return memoria[numero]
    memoria[numero] = fibonacci_memoria(numero-1) + fibonacci_memoria(numero-2)
    return memoria[numero]
nume = int(input('Ingresa un numero para fibonacci: '))
print('El numero de fibonacci en la position {} es {}'.format(nume, fibonacci_memoria(nume)))
print(memoria)
#memoria queda como {1:0, 2:1, 3:1, 4:2, 5:3} cuando nume=5
print()
```

Para el segundo ejemplo tenemos la función que dará el número de la posición n de la serie de fibonacci el cual está constituida por estructuras selectivas y con ayuda de otra función, a comparación de la segunda función la cual es recursiva sólo es necesario establecer un caso base para volver a llamar a la misma función 2 veces y sumarlás, como extra nos ayudamos con un diccionario para guardar los resultados..

```

from turtle import *
from argparse import *

for i in range(30):
    tess.stamp()
    size += 3
    tess.forward(size)
    tess.right(24)
def recorrido_recursivo(tortuga, espacio, huellas):
    if huella > 0:
        tortuga.stamp()
        espacio += 3
        tortuga.forward(espacio)
        tortuga.right(24)
        recorrido_recursivo(tortuga, espacio, huellas-1)

ArgumentParser.add_argument("--huellas", required=True, help="Numero de Huellas")
args = vars(ArgumentParser.parse_args())
huellas = int(args["huellas"])
print()

```

Ln: 47 Col: 55

Por último esta parte no fue debidamente puesta para programarse pero lo que se quería llegar es poder graficar una función la cual simulaba las huellas del pasar de una tortuga la cual se va moviendo en forma circular, donde el espacio entre las huellas se va haciendo cada vez menor.

## 4° Conclusiones

Avila Laguna Ricardo :

Los objetivos de esta practica se cumplieron por que aprendimoss el concepto y características de la recursividad junto con las ventajas que nos puede proporcionar, al igual aplicamos los conceptos para resolver problemas.

## Bibliografía

<http://lcp02.fi-b.unam.mx/>