	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: M.I. Edgar Tista García

Asignatura: Estructura de Datos y Algoritmos I

Grupo: 1

No de Práctica(s): 12

Integrante(s): Díaz Hernández Marcos Bryan

No. de Equipo de cómputo empleado:

No. de Lista o Brigada: 9

Semestre: 2020-2

Fecha de entrega: 13 de mayo de 2020

Observaciones:

CALIFICACIÓN: _____

Objetivo de la práctica

Aplicar el concepto de recursividad para la solución de problemas.

Introducción

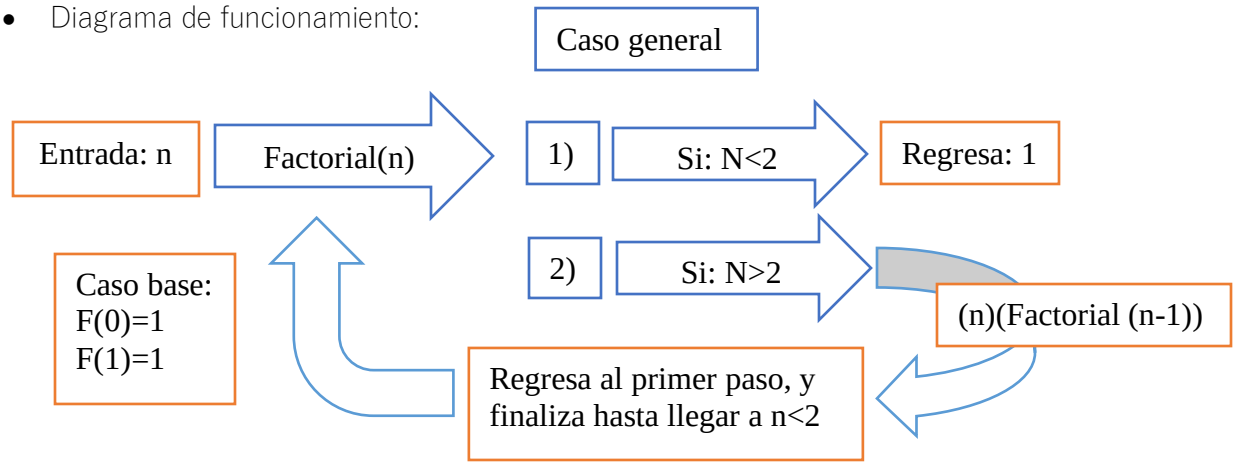
Esta guía está elaborada con la intención de que yo pueda comprender cada uno de los ejercicios, además de poder expresar lo que he aprendido en la elaboración del reporte.

Ejercicios de la guía de laboratorio

- Primer ejemplo: Código – Factorial

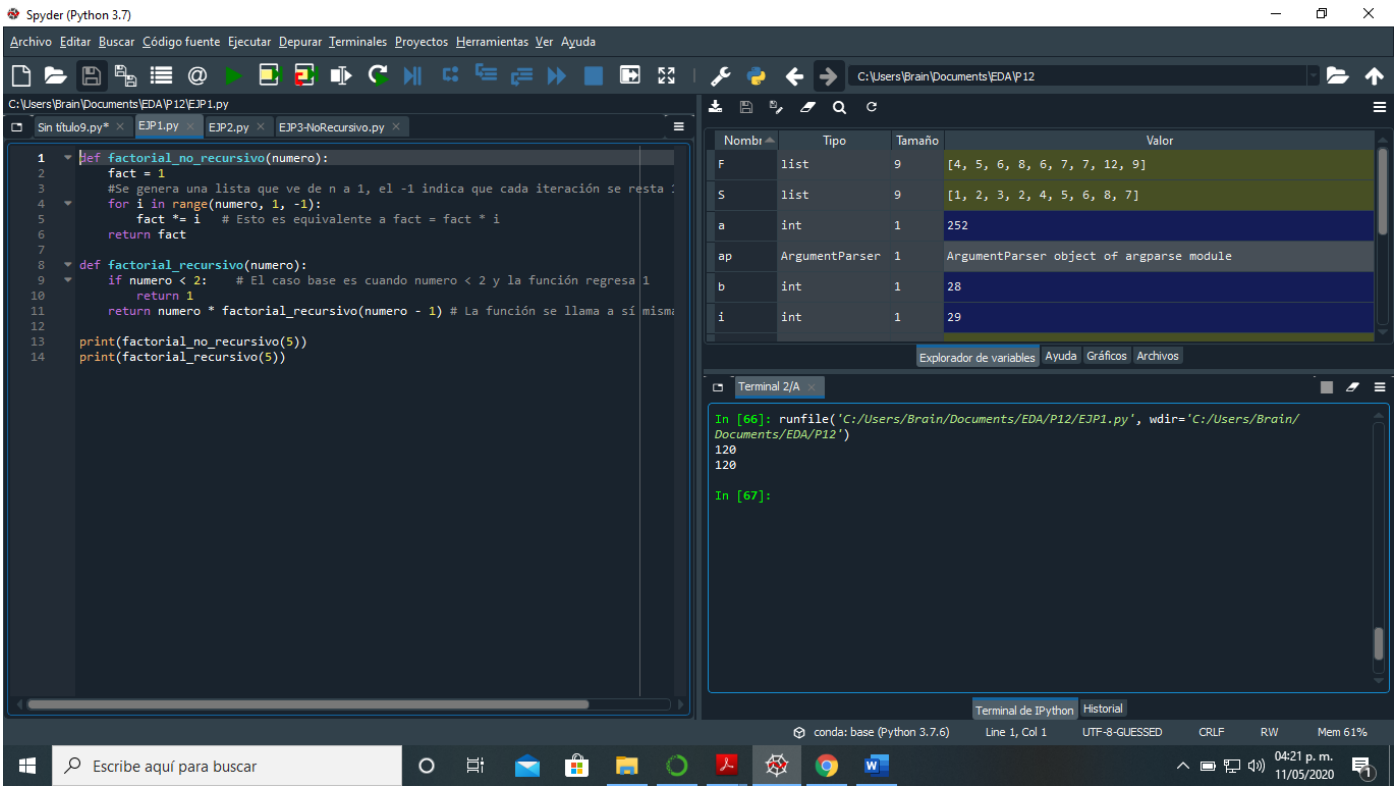
El primer ejercicio consiste poder calcular la factorial de un número, en este caso por medio de una función que se llama a sí misma, siendo un método de resolución, por el hecho de que es posible de resolver por medio de otro método no recursivo, lo principal es que la función regrese a los valores o casos base de la función para poder dar solución a cualquier caso n.

- Diagrama de funcionamiento:



El programa es reiterativo, me gusta verlo como una función cíclica que se repite hasta que llega al límite, en este caso al caso base de los valores, con los cuales puede resolver las partes pendientes que quedan como el:"(n)(Factorial(n-1))", porque son operaciones que esperan a obtener el valor que envuelve el volver a realizar las operaciones de la función.

- Evidencia de implementación.



- Segundo ejemplo: Código – Huellas de Tortuga

El segundo ejercicio consiste en la gráfica de las huellas de una tortuga, la condición principal es que se forme un círculo que cada vez se abre más y que la distancia que recorre conforme al tiempo sea cada vez mayor, y la distancia entre las huellas sea mayor después de cada huella.

Un problema con el ejercicio recursivo fue que no encontraba el parámetro de las huellas, por lo que el resultado era erróneo y no proyectaba ninguna animación, al intentar resolverlo o buscar el error, simplemente no lo encontré por lo que decidí no modificar nada (Imagen 1).

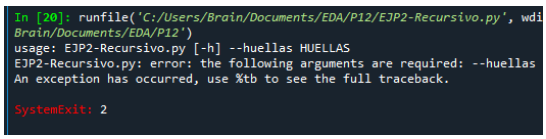
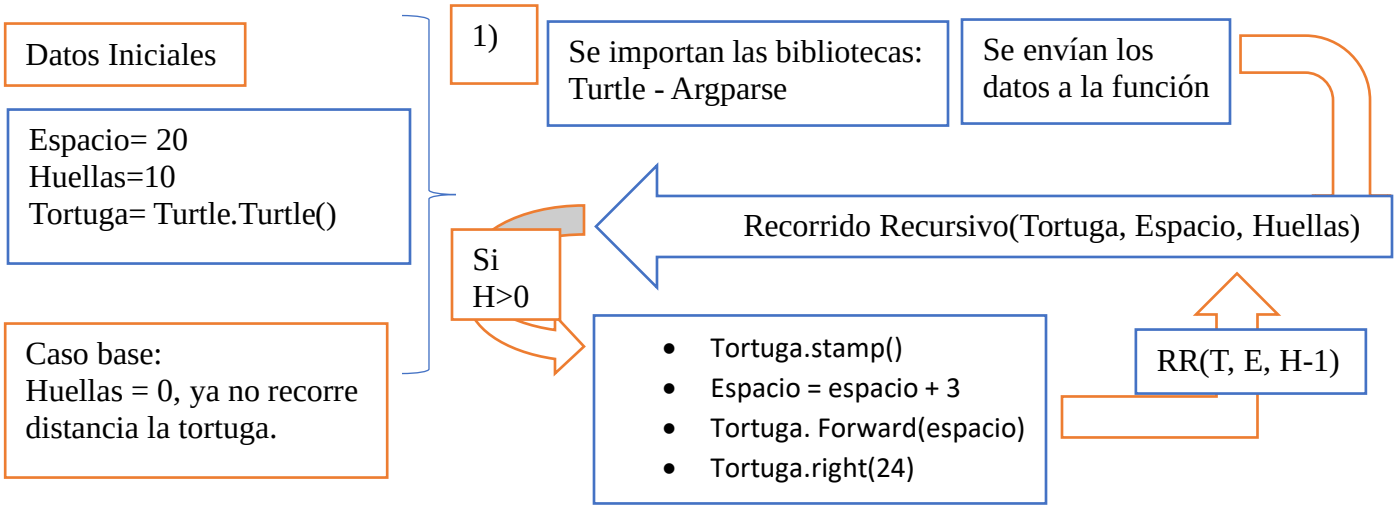


Imagen 1.

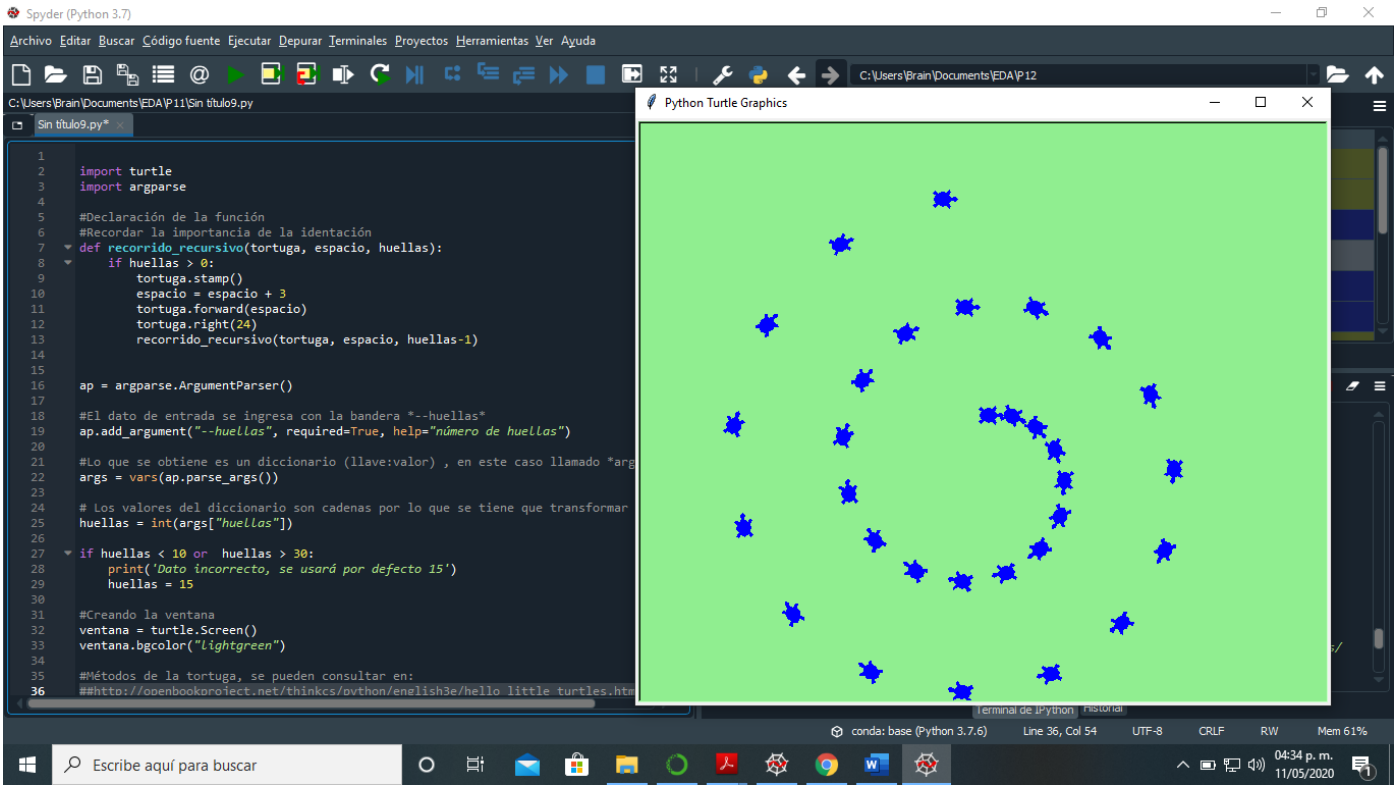
Para el caso del no recursivo, se utiliza un ciclo que repite las operaciones, una y otra vez, hasta terminar todas las huellas que se desean.

- Diagrama de funcionamiento:



De forma teórica el programa debería mostrar la imagen de una tortuga que se va moviendo y después de “n” huellas se detiene, por lo que el caso base se basa en las huellas que la tortuga puede dar, aparte de las demás instrucciones que son sobre la configuración de la ventana y de la imagen de la tortuga, la función recursiva en si consiste en reducir las huellas o los pasos, pertenece a la recursividad porque puede dar una solución al llegar a un caso base y en este caso es más progresivo ya que se va buscando este caso base sin dejar una operación esperar. En el caso del no recursivo, es la inversa de esta operación que hace un incremento con un for.

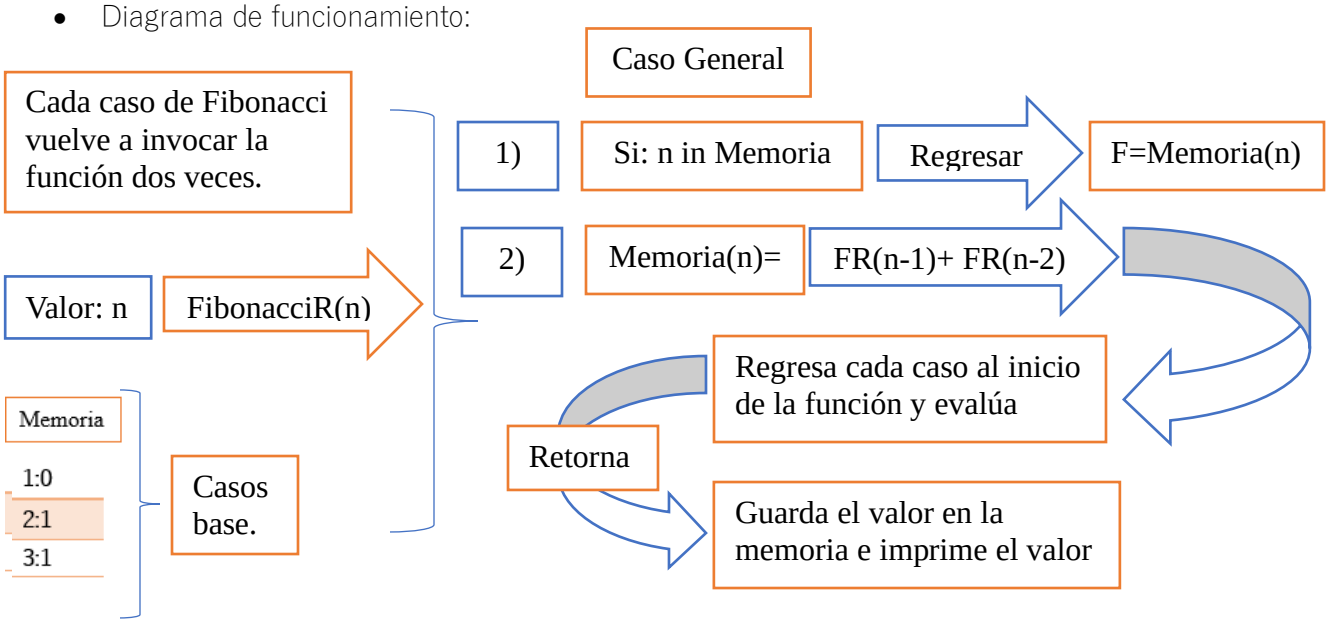
- Evidencia de implementación.



- Tercer ejemplo: Código – Fibonacci

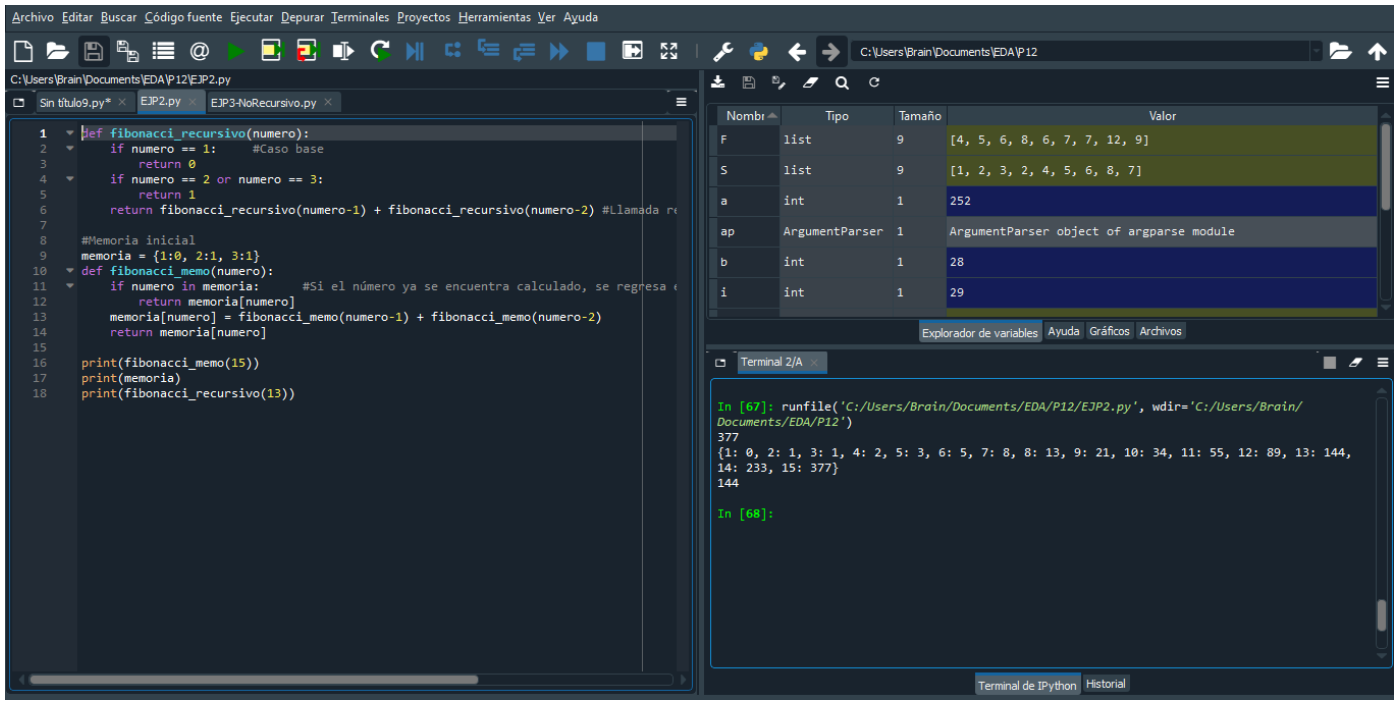
El tercer ejercicio consiste en encontrar el valor “n” de la sucesión de Fibonacci, como se había visto en otras practicas es posible resolver los ejercicios, mediante varios métodos, para este ejercicio se proponen dos formas recursivas, donde la ventaja de una sobre la otra es que se realizan menos operaciones y se guardan los valores, que anteriormente se llegaron a pedir.

- Diagrama de funcionamiento:



El algoritmo como los anteriores busca el caso base para poder realizar las operaciones que va dejando sin realizar, como es el caso del diagrama que, al llamarse de nuevo dos veces, no resuelve la suma hasta que haya que obtenga los valores de Fibonacci de los valores (n-1) y (n-2).

- Evidencia de implementación.

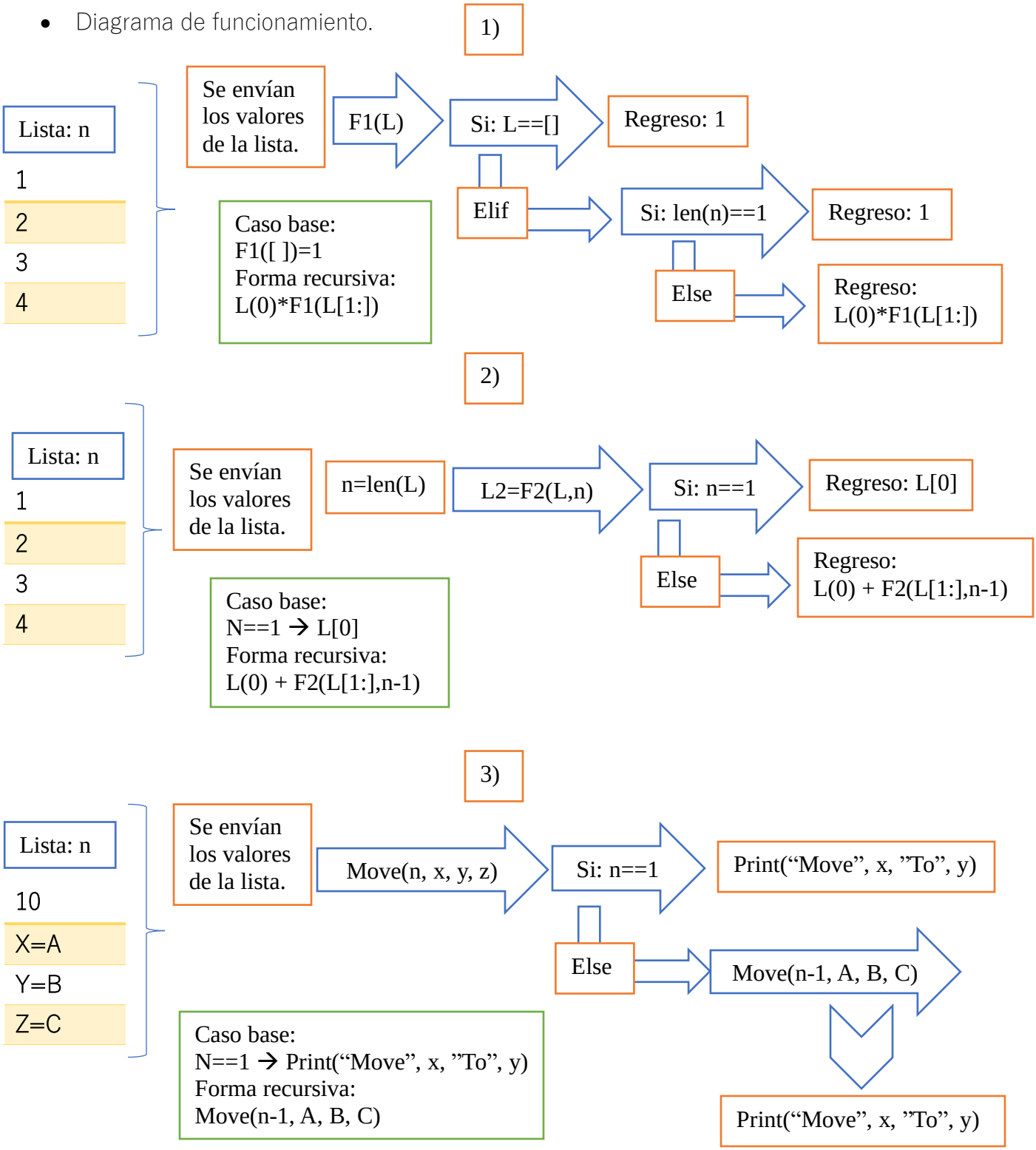


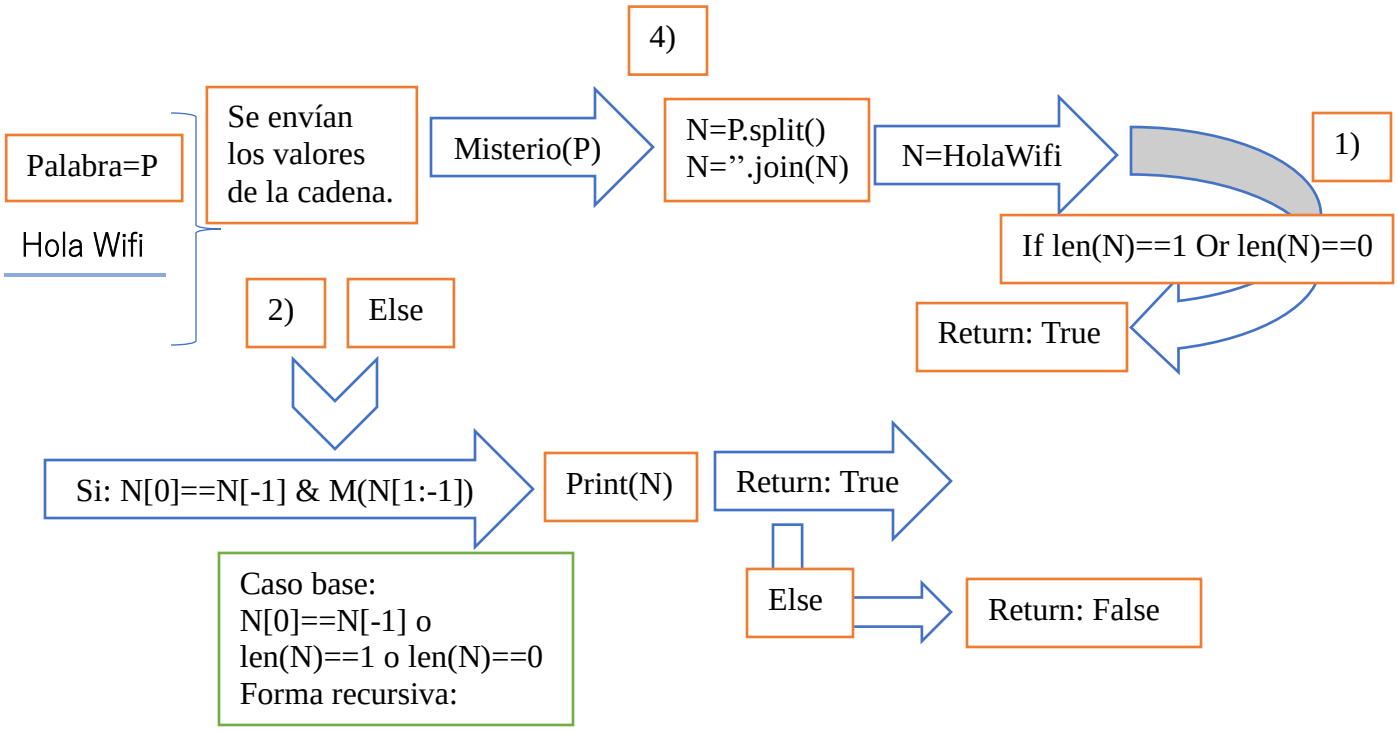
Ejercicios de la práctica

- Ejercicio 1.

Consistía en ejecutar los distintos códigos de funciones recursivas, de tal manera que se haga un análisis de su funcionamiento, y un análisis de las entradas y salidas, además de describir el caso base y la forma recursiva.

- Diagrama de funcionamiento.





- Relación con teoría.

1) Para el primer ejercicio se lleva a cabo el proceso de recursividad para poder obtener en valor de la multiplicación de los elementos de una lista, para ello se llama a sí misma la función al punto de que llega al caso base, cuando llega la caso base este tiene la solución más sencilla, y se puede obtener la solución total de las operaciones que se quedaron esperando el valor del caso base.
2) El segundo ejercicio consiste en la suma de los elementos de una lista por medio de recursividad, de ahí que las suma que se van acumulando tras cada llamada de la función, se tengan que detener y espera el resultado o el caso donde aparezca el caso base de la recursividad, y poder dar la solución al problema.
3) El tercero tiene un error en la tercera instrucción del else (Imagen 1), porque esta instrucción crea una doble llamada de recursividad, lo que incrementa exponencialmente el número de operaciones que se tienen que realizar debido a que cada vez que se resta a "(n-1)", se hace una llamada de nuevo por lo que ella llamada hace dos llamadas después, y así consecutivamente hasta el punto de realizar muchas operaciones.

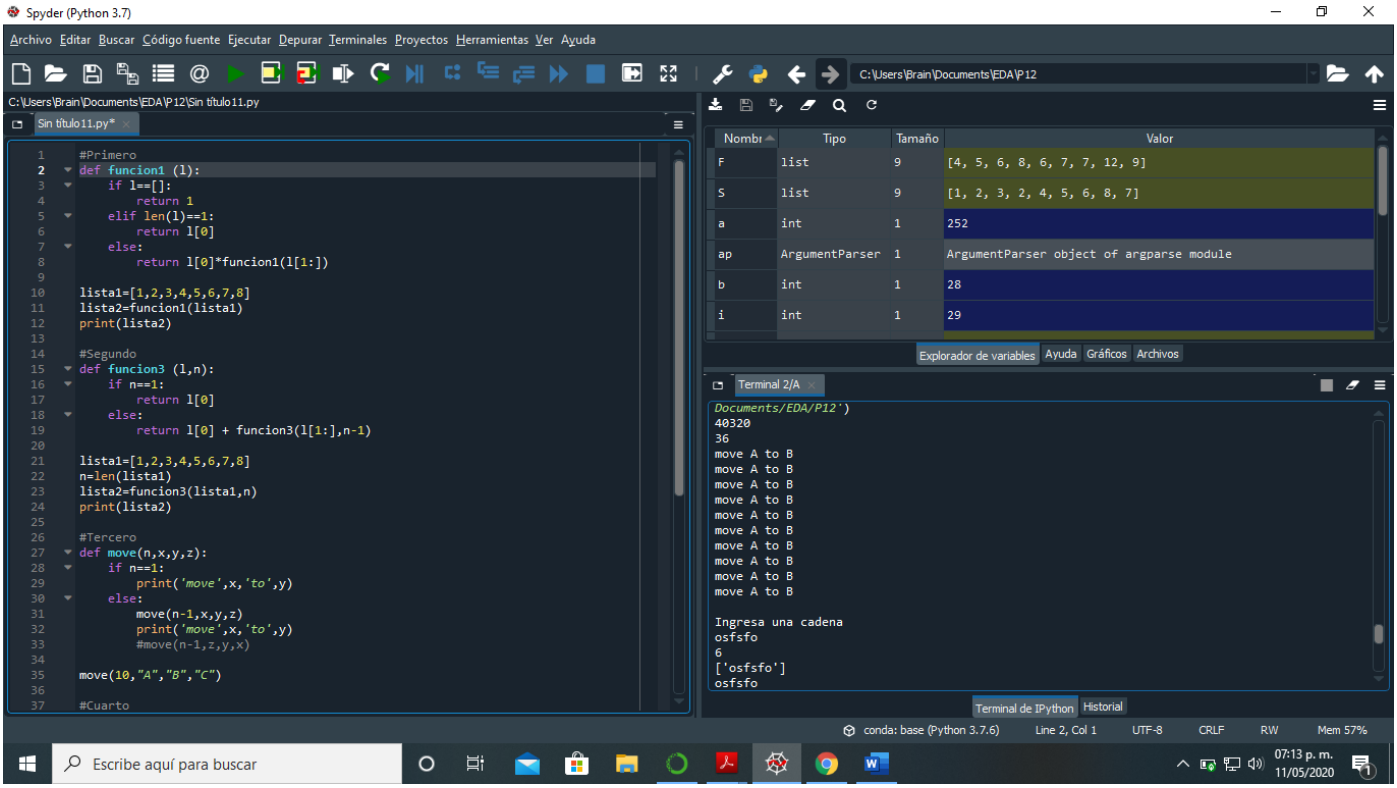
```
move(n-1,x,y,z)
print('move',x,'to',y)
#move(n-1,z,y,x)
```

Imagen 1.

De forma ideal debería dar las combinaciones entre las letras A,B,C y llegar al caso base.

4) El ultimo ejercicio consiste en encontrar palabras simétricas, es decir palabras que inicien con ciertas letras y terminen con esas mismas letras, debido a esto las palabras que cumplen con la simetría hacen el llamado de la función dentro de la función y a las cadenas se les elimina las partes externas o los limites de palabra, y esta nueva palabra se envía a la función para comprobar su longitud y pueda llegar al caso base donde existen dos: el primero donde la longitud de la palabra es de 0 o 1, y la segunda donde los elemento extremos sean iguales.

- Evidencia de implementación



Conclusiones.

En la elaboración de la practica aprendí a identificar las partes esenciales de una función recursiva, como lo son el caso base y la forma de recursividad, además de poder analizar algunos algoritmos y ver como se implementan en forma de código, e incluso refrescar algunos conceptos de la programación en Python, por último, los ejercicios fueron bastante ilustrativos de lo visto en clase virtual.