

**UNIVERSIDAD NACIONAL DE TRUJILLO**

Facultad de Ingeniería

Programa de Ingeniería Mecatrónica

LABORATORIO N° 6

“ARCHIVOS - MÓDULOS - LIBRERÍAS DIVERSAS”

**DESARROLLO DE GUIA DE LABORATORIO**

PROGRAMACIÓN I

**ESTUDIANTE(S) :**

**SÁNCHEZ ROJAS JHONATAN ARTEMIO**

**VALDIVIEZO JIMÉNEZ VÍCTOR JAVIER**

**VIGO VILLAR CRISTHIAN AARON**

**DOCENTE :**

**ASTO RODRIGUEZ EMERSON MAXIMO**

**CICLO :**

**2022 I**

Trujillo, Perú

2022

**INDICE**

[RESUMEN 3](#_Toc108776522)

[DESARROLLO DEL LABORATORIO 4](#_Toc108776523)

[1.1. Desarrollo de la experiencia 4](#_Toc108776524)

[a) Ejercicio 1 4](#_Toc108776525)

[b) Ejercicio 2 7](#_Toc108776526)

[c) Ejercicio 3 8](#_Toc108776527)

[d) Ejercicio 4 10](#_Toc108776528)

[1.2. Resultados de la experiencia 13](#_Toc108776529)

[a) Ejercicio 1 13](#_Toc108776530)

[b) Ejercicio 2 14](#_Toc108776531)

[c) Ejercicio 3 15](#_Toc108776532)

[d) Ejercicio 4 16](#_Toc108776533)

[1.3. Desarrollo de test de comprobación 17](#_Toc108776534)

[1.4. Recomendaciones 20](#_Toc108776535)

[1.5. Conclusiones 20](#_Toc108776536)

[REFERENCIAS BIBLIOGRÁFICAS 21](#_Toc108776537)

[ANEXOS 22](#_Toc108776538)

[1.1 Código completo del ejercicio 1: 22](#_Toc108776539)

[1.2 Código completo del ejercicio 2: 23](#_Toc108776540)

[1.3 Código completo del ejercicio 3: 24](#_Toc108776541)

[1.4 Código completo del ejercicio 4: 25](#_Toc108776542)

# RESUMEN

El presente informe de laboratorio supone un análisis especifico del desarrollo de 4 actividades propuestas sobre programación. Las cuales piden la elaboración de códigos en Python para el desarrollo de cierto programa que se ejecute de la forma especificada en la actividad. La realización de estas actividades surge en torno a la práctica de un recién empezado curso de Python, por lo que, supone el uso de herramientas básicas como las listas, las funciones, los diccionarios entre otros. Y, de las ya conocidas formas condicionales como el “if” o las formas para la iteración como el “while” y el “for”.

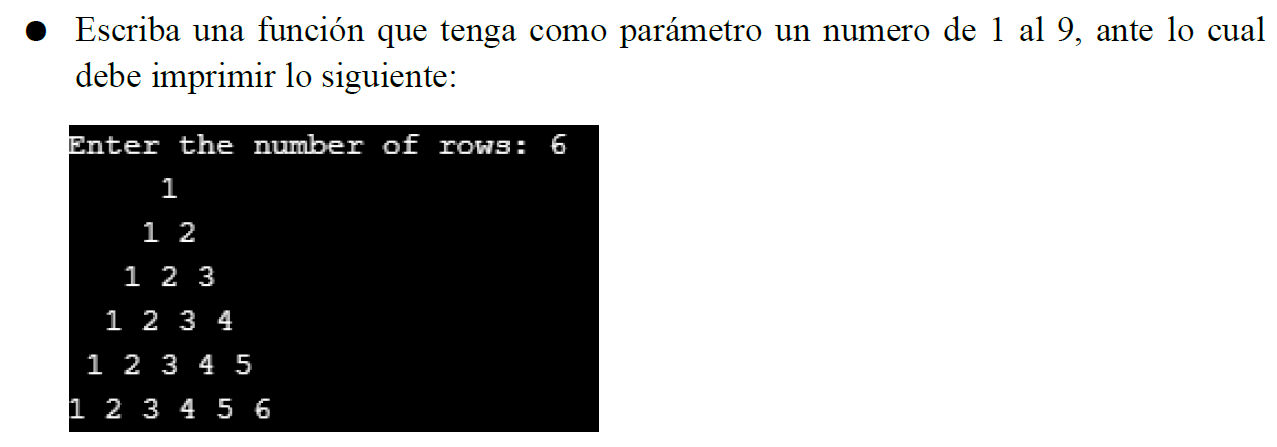
Finalmente, se pudo llegar a los correctos resultados pedidos en cada ejercicio, concluyendo en el fácil manejo de Python en comparación con otros lenguajes de programación usados anteriormente.

# DESARROLLO DEL LABORATORIO

## Desarrollo de la experiencia

### Ejercicio 1

Vigo Villar Cristhian Aaron



Para este ejercicio, utilizaremos funciones, bucles como while y for, además de condicionales if, elif y else.

* Simplemente como decoración le colocaremos el siguiente título y para que se vea un poco centrado estará seguido de una tabulación:

print("\t .:PIRAMIDE DE NÚMEROS:.")

* Ahora haremos el uso de una función, ya que lo que intento hacer es que cuando el usuario ingrese un número y sea negativo, decirle que ingrese uno positivo y se ejecute nuevamente la función generadora de pirámide de números; además de solicitar si desea seguir jugando o no, por lo que para eso considerare “1” para si y “2” para no, porque al pedir respuesta escrita y aún habiendo definido el input como str, me daba error, así que por eso lo deje en int.

def piramide (x):

* El desear que se ejecute correctamente la función para un número positivo me lleva a la siguiente condición:

    if x>0:

* Y para que se cumpla lo dicho en el segundo paso, colocamos un else y volvemos a generar la misma función para que no de error al colocar un número negativo o 0 y además mejorar la experiencia de la persona que probará el programa.

    else:

        x = int(input("Ingrese un número entero positivo: "))

        piramide (x)

* Ubicandonos en la identación correcta que es una tabulación después del if, aplicamos el bucle while y break ya que no depende de nosotros el terminar el bucle, sino del usuario.

        while True:

            break

* Ahora lo que queremos es escribir muchas veces números, por lo que utilizamos el bucle for y lo hacemos repetirse con el iterador “i” para un rango de 1 a x+1 ya que siempre termina en un número antes del final y x debido a que es la variable independiente de nuestra función **piramide(x)**, colocándolo entre while y break con su identación respectiva.

while True:

            for i in range(1,x+1):

            break

* Dentro de for colocaremos 2 for más, uno que inserte espacios que separen los números y den forma de pirámide. Por ejemplo: si son 3 números, debe tener 2 espacios arriba, 1 al medio y al final ninguno: el color negro representa los **espacios antes del número** y el rojo los **espacios entre cada número**. El otro for cumplirá la función de escribir los números, pero como se debe ejecutar para cada espaciado antes del número anterior, colocaremos estos 2 for en la misma identación. Aclaramos que estos espacios se deben escribir seguidos, por lo que en el primer for colocamos (“ “,end=””) ya que no necesitamos otro espacio y en el segundo (k+1, end=” “)ya que necesitamos separar los números.



            for i in range(1,x+1):

                for j in range(x-i):

                    print(" ", end = "")

                for k in range (i):

                    print (k+1, end = " ")

                print ("\n")

* Y ya tenemos practicamente listo nuestro programa, pero como me plantee al inicio que el usuario decida si continua o no, debemos hacer uso de los condicionales.

def piramide (x):

    if x>0:

        while True:

            for i in range(1,x+1):

                for j in range(x-i):

                    print(" ", end = "")

                for k in range (i):

                    print (k+1, end = " ")

                print ("\n")

            break

    else:

        x = int(input("Ingrese un número entero positivo: "))

        piramide (x)

* Para eso, dentro de while, utilizamos if y elif, dentro de while porque queremos que se repita otra vez la función y después de elif el break, ya que ahí queremos que acabe. Entonces pedimos otra variable que sea entero con input y según eso creamos los condicionales.

            a = int(input('¿Desea generar otra pirámide? ("1" para si y "2" para no): '))

            if a == 1:

                b=int(input("Ingrese otro número: "))

                piramide (b)

            elif a == 2:

                print("...JUEGO FINALIZADO...")

* Una vez terminada la función solo nos queda pedir al usuario que ingrese un número para generar su pirámide.

n=int(input("Ingrese su número: "))

piramide (n)

### Ejercicio 2

Valdiviezo Jimenez, Victor Javier

* Escribir un programa que consulte si uno quiere saber sobre algún tema en específico, y luego responda sobre lo consultado usando información de Wikipedia. El programa debe responder usando voz.
* Empezamos importando los módulos que se van a necesitar para la realización del programa, dándonos cuenta, en el ítem, que serán necesarios los módulos de síntesis de voz, reconocimiento de audio y Wikipedia. Además, los inicializamos.

import pyttsx3

import speech\_recognition

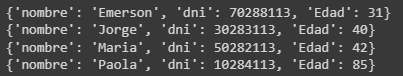
import wikipedia

engine = pyttsx3.init()

recognizer = speech\_recognition.Recognizer()

### Ejercicio 3

Valdiviezo Jimenez, Victor Javier

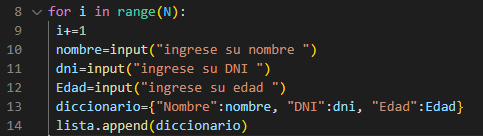
* Escriba un programa que solicite los datos de diversas personas (nombre, DNI y edad) y finalmente los imprima ordenados de menor a mayor. Utilice diccionarios en su solución. El resultado esperado se muestra a continuación: 
* Para este ejercicio, primero empezamos tomando en cuenta, que nos piden de “diversas personas”; por consiguiente, definimos un dato N entero, colocado manualmente, para precisar esta cantidad.



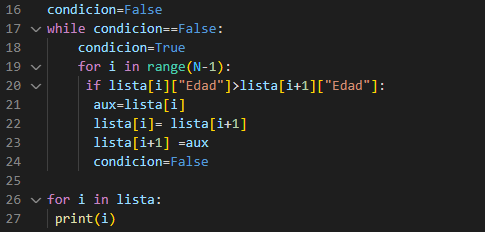
* Luego, pasamos a crear una lista vacía, a la cual le iremos agregando sus elementos más adelante. En este caso, sus elementos serían los diccionarios con los datos de cada persona.



* Para el proceso de la creación de los diccionarios, nos apoyamos del comando “for” iterando en un rango de i hasta N. En donde en cada iteración nos pedirá los datos (nombres, DNI y edad) de la persona, para luego pasar a colocarlos en un diccionario que se agregará a la lista vacía, gracias al comando “append”.

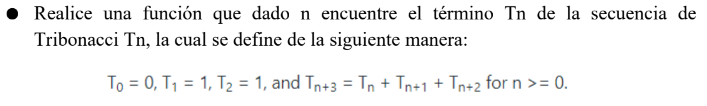


* Finalmente, ordenamos los diccionarios en la lista de menor a mayor (edad), por el método de la burbuja. Y hacemos “print” de la lista ordenada, pero con el comando “for” para que nos salga cada elemento (diccionario) en una sola línea.



### Ejercicio 4

Sánchez Rojas, Jhonatan Artemio



* Antes de comenzar con la explicación del código utilizado para ejecutar la secuencia de Tribonacci, colocaré una tabla que muestra los valores de cada término en la secuencia.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Secuencia de Tribonacci | | | | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ….. |
| 0 | 1 | 1 | 2 | 4 | 7 | 13 | 24 | 44 | 81 | 149 | 274 | 504 | 927 | 1705 | 3136 | ….. |

* Por temas de estética al momento de ejecutar el código, le coloqué un título:

print("Secuencia de Tribonacci")

* Definí la función y le coloqué un nombre que este relacionado a la secuencia Tribonacci:

def trib(n):

* Posteriormente, apliqué el condicional if para establecer los parámetros que debe cumplir al momento de ejecutar el código:

def trib(n):

    if (n<=0 and n>=0):

     return n

    if (1<=n and n<=2):

     return 1

    if (n>=3):

     return trib(n-1) + trib(n-2) + trib(n-3)

* El mismo ejercicio nos dice como está definida la secuencia, pero utilicé una definición mas sencilla a mi parecer, la cual es:
* Para este ejercicio se debe identificar al término N en la sucesión, el cual viene siendo el último término en la sucesión, por lo tanto, definimos a “x” como el término N en la sucesión:

x=int(input("Ingrese el término N: "))

* Como se habla de una sucesión, la función se aplicará varias veces dependiendo del término N que coloquemos, para lo cual se establecerá un rango que depende de x haciendo uso de for:

for x in range(x):

    print (trib(x+1))

* En el paso anterior se colocó trib(x+1) para que la secuencia empiece desde el término 1 y no desde el término 0.
* Por último, se coloca un texto que nos indica cual es el término N:

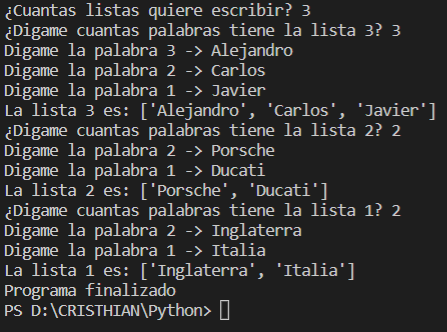
print("El término N es el último número en la secuencia")

## Resultados de la experiencia

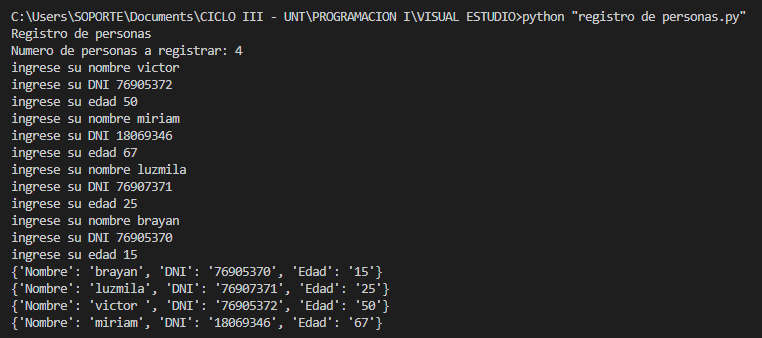
Link del repositorio: <https://github.com/jhonatan0926/Laboratorio-6-Grupo1-PI-UNT_2022>

### Ejercicio 1

### Ejercicio 2

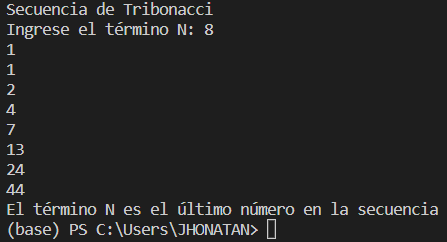


### Ejercicio 3

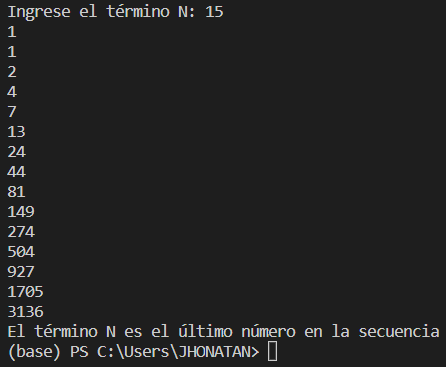


### Ejercicio 4

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Secuencia de Tribonacci | | | | | | | | | | | | | | | | |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  | ….. |
| 0 | 1 | 1 | 2 | 4 | 7 | 13 | 24 | 44 | 81 | 149 | 274 | 504 | 927 | 1705 | 3136 | ….. |



Término N



Término N

Comparando los resultados con la tabla, notamos que son los mismos, por lo tanto, el código esta ejecutado de manera correcta.

Cabe aclarar que cuando N es un número mas alto, el código tardará un poco más de tiempo en ejecutarse.

## Desarrollo de test de comprobación

**a) Explique las diferencias entre una lista de Python y un array de PSeint.**

**Lista de Python:**

Una lista de Python es una colección de elementos que pueden contener elementos de diferentes tipos de datos, como números, valores de caracteres lógicos, entre otros. Esta es una colección ordenada que admite la indexación negativa. Puede usar [] para crear una lista que contenga valores de datos.

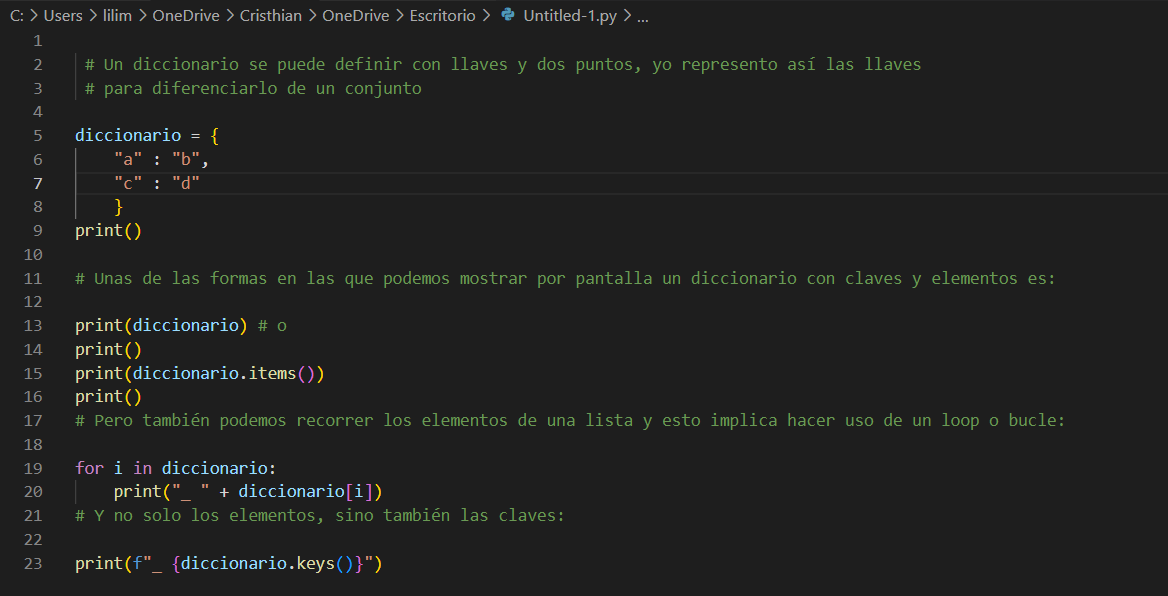
El contenido de la lista se puede fusionar y copiar fácilmente utilizando las funciones integradas de Python. (ACERVO LIMA, 2021)

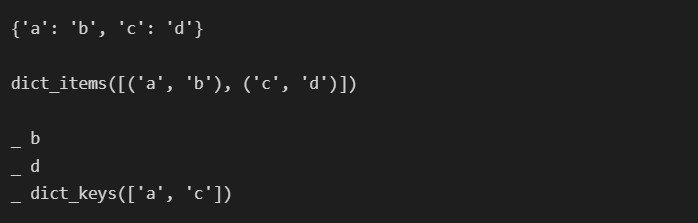
**Array de Pseint:**

Los arreglos en PSeint son estructuras que almacenan valores del mismo tipo, tales como números o cadenas. Tiene una cierta longitud (la cual se determina). Lo primero es declarar su tamaño (dimensión), dando el nombre del arreglo o matriz y su longitud dentro de []. (Disco Duro de Roer, 2013)

1. La lista de Python permite tener una colocar elementos que no necesariamente deben pertenecer al mismo tipo de datos, en cambio, en el array de Pseint los elementos deben ser del mismo tipo, de lo contrario habría un error al momento de ejecutar el código.
2. En la Lista de Python no hace falta determinar la dimensión y longitud, en cambio el array de Pseint si lo requiere para ejecutar el código.
3. Ejecutar una lista de Python es mas sencillo que ejecutar un array de Pseint, dado que, el lenguaje Python es más simplificado.

**b) Explique las diferentes formas de recorrer un diccionario.**

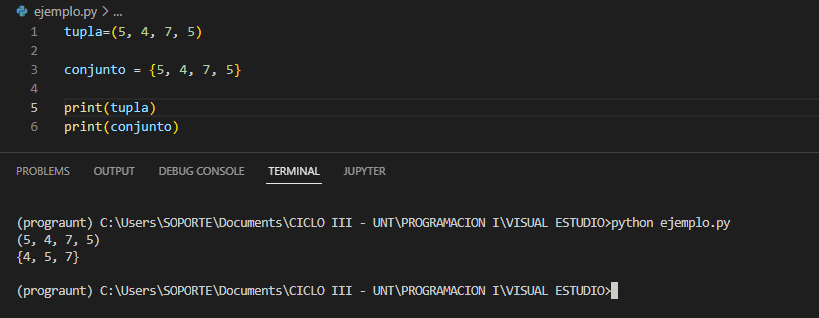




**c) Explique la diferencia entre las tuplas y los conjuntos. De un ejemplo de uno en cada caso.**

Las tuplas son colecciones de datos secuenciales (importa la secuencia de sus elementos) muy parecidas a las listas, pero que, a diferencia de estas, son inmodificables, lo que quiere decir que sus elementos se mantienen constantes y no pueden ser cambiados. Por otro lado, los conjuntos también son colecciones de datos pero que en su uso hacen alusión mayormente al significado matemático de la palabra, por lo que elementos repetidos los tomara en cuenta como uno solo y para datos numéricos, los ordenara de menor a mayor (sin importa la secuencia de los elementos). Además, entre los conjuntos nos permiten realizar algunas operaciones como la intersección, unión, diferencia y diferencia simétrica.

Ejemplos:



## Recomendaciones

* Al generar la pirámide, el poner espacios en vez de tabulaciones, es muy importante, ya que al inicio lo intente así y me salía triángulos rectángulos, cambiaba el código y solo se movía de izquierda a derecha, nunca partía del centro, por eso se hizo uso de espacios vacíos antes de cada número.
* Al momento del uso de método de la burbuja en el ordenamiento por edades del registro de las personas, es muy importante el tomar en cuenta las condiciones booleanas para el “while”, ya que esto permitirá hacer un recorrido múltiple de la lista, hasta que se haya ordenado completamente; caso contrario el “for”, se realizará una sola vez, lo que significaría un ordenamiento a medias, o por pares aledaños, de la lista; y no un ordenamiento total de los elementos.
* Al momento de hacer el código de la función de la secuencia de Tribonacci, no siempre uno debe seguir al pie de letra la definición que establece el ejercicio, uno puede darle la forma que mas le convenga, esto ya es criterio de cada persona.

## Conclusiones

* El intentar imaginar en la mente que es lo que se quiere realizar haciendo un bosquejo simple, nos servirá mucho para no enredarse al empezar a programar y culminar el código de forma rápida, sin llegar a estresarse porque no se ejecuta lo deseado.
* El lenguaje Python es muy simplificado, lo cual es de mucha ayuda al momento de generar un código. Además, la variedad de cosas que se pueden hacer con sus diferentes colecciones de datos como las listas, nos abre un sinfín de procedimientos nuevos que no se podían hacer en el PSeInt.
* El desarrollo de Python exclusivamente en inglés, es una de las pocas barreras que nos trae este lenguaje; sin embargo, es un problema menor en comparación a los múltiples beneficios que nos brinda.

# REFERENCIAS BIBLIOGRÁFICAS

ACERVO LIMA. (15 de Abril de 2021). Diferencia entre lista y matriz en Python. Obtenido de ACERVO LIMA: https://es.acervolima.com/diferencia-entre-lista-y-matriz-en-python/

Disco Duro de Roer. (1 de Diciembre de 2013). Arreglos O Arrays En Pseudocódigo. Obtenido de Disco Duro de Roer: https://www.discoduroderoer.es/arreglos-o-arrays-en-pseudocodigo/

Foundation, P. S. (12 de Julio de 2022). Documentación de python. Obtenido de https://docs.python.org/es/3.10/contents.html

Line!, C. T. (2020, septiembre 30). Listas, Tuples, Sets, Strings y Diccionarios en PYTHON. https://www.youtube.com/watch?v=CCUNuqqn7PQ&t=392s

Samaniego, T. (2017, marzo 12). Tuplas & conjuntos en python. Mi Diario Python; dcaraballo. https://pythondiario.com/2017/03/tuplas-conjuntos-en-python.html

# ANEXOS

## Código completo del ejercicio 1:

print("\t .:PIRAMIDE DE NÚMEROS:.")

def piramide (x):

    if x>0:

        while True:

            for i in range(1,x+1):

                for j in range(x-i):

                    print(" ", end = "")

                for k in range (i):

                    print (k+1, end = " ")

                print ("\n")

            a = int(input('¿Desea generar otra pirámide? ("1" para si y "2" para no): '))

            if a == 1:

                b=int(input("Ingrese otro número: "))

                piramide (b)

            elif a == 2:

                print("...JUEGO FINALIZADO...")

            break

    else:

        x = int(input("Ingrese un número entero positivo: "))

        piramide (x)

n=int(input("Ingrese su número: "))

piramide (n)

## Código completo del ejercicio 2:

num\_list = int(input("¿Cuantas listas quiere escribir? "))

list = []

for j in range(1,num\_list+1):

    z=num\_list-j+1

    pal = int(input(f"¿Digame cuantas palabras tiene la lista {z}? "))

    for k in range (1, pal+1):

        a = str(input(f"Digame la palabra {pal-k+1} -> "))

        list.append(a)

    print (f"La lista {z} es: {list}")

    list.clear()

print("Programa finalizado")

## Código completo del ejercicio 3:

1. print("Registro de personas")
2. N=int(input("Numero de personas a registrar: "))
3. global lista
4. lista=list()
5. for i in range(N):
6. i+=1
7. nombre=input("ingrese su nombre ")
8. dni=input("ingrese su DNI ")
9. Edad=input("ingrese su edad ")
10. diccionario={"Nombre":nombre, "DNI":dni, "Edad":Edad}
11. lista.append(diccionario)
12. condicion=False
13. while condicion==False:
14. condicion=True
15. for i in range(N-1):
16. if lista[i]["Edad"]>lista[i+1]["Edad"]:
17. aux=lista[i]
18. lista[i]= lista[i+1]
19. lista[i+1] =aux
20. condicion=False
22. for i in lista:
23. print(i)

## Código completo del ejercicio 4:

print("Secuencia de Tribonacci")

def trib(n):

    if (n<=0 and n>=0):

     return n

    if (1<=n and n<=2):

     return 1

    if (n>=3):

     return trib(n-1) + trib(n-2) + trib(n-3)

x=int(input("Ingrese el término N: "))

for x in range(x):

    print (trib(x+1))

print("El término N es el último número en la secuencia")