

**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”.

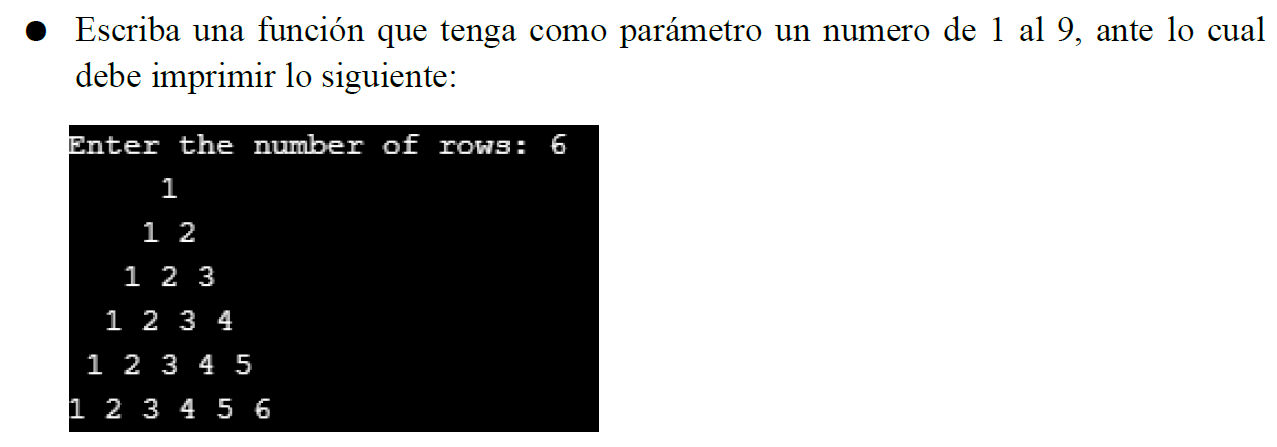
Finalmente, se pudo llegar a los correctos resultados pedidos en cada ejercicio, concluyendo en el fácil manejo de Python en comparación con otros lenguajes de programación usados anteriormente.

# 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...")

### 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()

* Definimos la función “sintesis(text)” que hará decir al programa el valor de “text” que se evalué en la función.

def sintesis(text):

 engine.say(text)

 engine.runAndWait()

* Definimos la función “reconocimiento ( )” que guardara la búsqueda que el usuario hará por voz en el valor “mensaje“ y finalmente retornara este mensaje.

def reconocimiento():

    try:

        with speech\_recognition.Microphone() as source:

            print("Escuchando... ")

            audio = recognizer.listen(source)

            mensaje = recognizer.recognize\_google(audio,language="es")

            print(mensaje)

    except:

       pass

    return mensaje

* Finalmente, definimos la función “buscar\_Wikipedia ( )” que pasara el valor de “mensaje” hacia el buscador de Wikipedia y arrojara el resultado mediante voz, usando la función “sintesis ( )”. Adicionalmente, agregamos un try except dentro de la función, para evitar la paralización del programa en caso no se reconozca la búsqueda.

def buscar\_wikipedia():

    try:

        sintesis("Hola, soy tu asistente de wikipedia, di lo que estas buscando")

        mensaje = reconocimiento()

        busqueda = mensaje

        wikipedia.set\_lang("es")

        respuesta = wikipedia.summary(busqueda, sentences = 1)

        sintesis(respuesta)

    except:

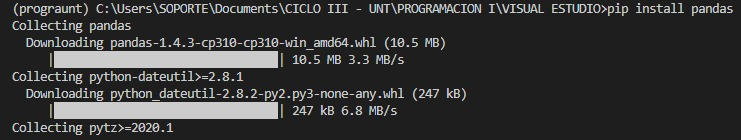
        sintesis("¡No se encontraron resultados para tu busqueda!")

buscar\_wikipedia()

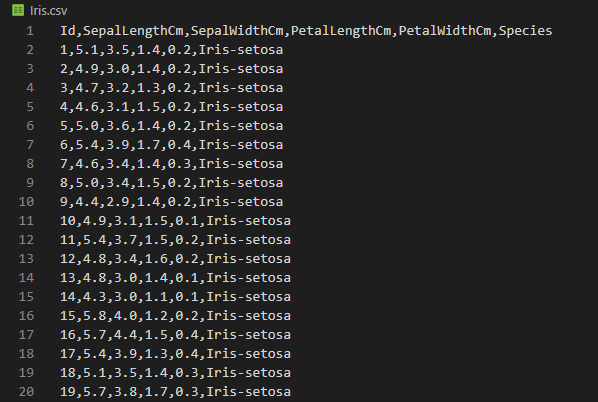
### Ejercicio 3

Valdiviezo Jimenez, Victor Javier

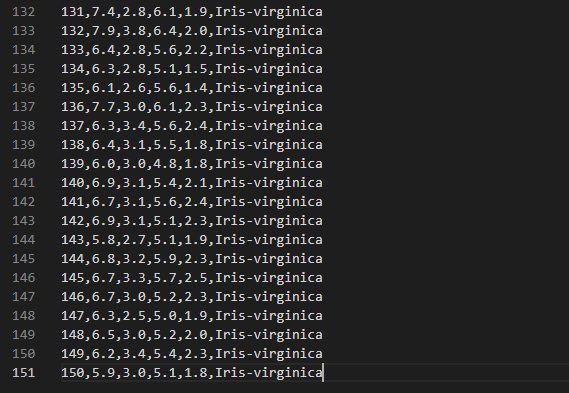
* **Escribir un programa que lea un archivo csv e imprima la información de forma agradable en la consola.**
* Para esto descargamos el módulo de pandas, que nos permite leer el archivo en forma de una tabla con encabezados.



* Seleccionamos el archivo csv con el que se va a trabajar. En este caso, el archivo elegido es “Iris.csv”, compuesto por 151 filas y 6 columnas. Significando la primera fila los encabezados de las columnas.



**…**



* Empezamos con la escritura del codigo, importando el archivo y el modulo pandas.

import csv

import pandas as pd

* Inicializamos el modulo panda con la funcion “read\_csv” y agregamos el comando “index\_col” para cambiar el idenficador propio de la dataframe a la columna “Id”.

df = pd.read\_csv("Iris.csv", index\_col = "Id")

* Finalmente imprimimos la funcion.

print(df)

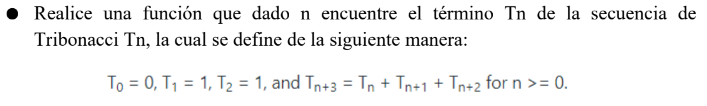
* Cabe resaltar que en casos donde las filas son bastantes, la dataframe se imprimira con las 5 primeras filas y las 5 ultimas. Esto es suficiente para el objetivo del item, que nos pide que la lectura sea agradable. Sin embargo, existen funciones para poder ver mas filas como es el “head(n)” o el “tail(n)”, la primera hace un llamado a las n primeras filas, mientras que la segunda a las n ultimas filas.

print(df.head(10))

print(df.tail(10))

### 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")

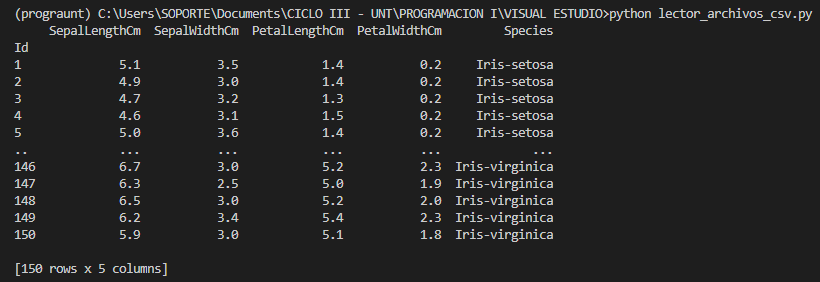
## Resultados de la experiencia

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

### Ejercicio 1

### Ejercicio 2

### Ejercicio 3



### Ejercicio 4

## Desarrollo de test de comprobación

* 1. **Explique que es el archivo \_\_init\_\_.py**
  2. **Explique qué significa la w, r, a, t, b, en el manejo de archivos.**

w: Abre el archivo para escribir nuevo contenido en reemplazo al anterior.

r: Abre el archivo como modo de lectura.

a: Abre el archivo para escribir nuevo contenido, pero a diferencia del “w”, este contenido es adicional al contenido anterior.

t:

b:

* 1. **Investigue y explique que son los decoradores en python.**

## 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:

## Código completo del ejercicio 2:

import pyttsx3

import speech\_recognition

import wikipedia

engine = pyttsx3.init()

recognizer = speech\_recognition.Recognizer()

def sintesis(text):

 engine.say(text)

 engine.runAndWait()

def reconocimiento():

    try:

        with speech\_recognition.Microphone() as source:

            print("Escuchando... ")

            audio = recognizer.listen(source)

            mensaje = recognizer.recognize\_google(audio,language="es")

            print(mensaje)

    except:

       pass

    return mensaje

def buscar\_wikipedia():

    try:

        sintesis("Hola, soy tu asistente de wikipedia, di lo que estas buscando")

        mensaje = reconocimiento()

        busqueda = mensaje

        wikipedia.set\_lang("es")

        respuesta = wikipedia.summary(busqueda, sentences = 1)

        sintesis(respuesta)

    except:

        sintesis("¡No se encontraron resultados para tu busqueda!")

buscar\_wikipedia()

## Código completo del ejercicio 3:

import csv

import pandas as pd

df = pd.read\_csv("Iris.csv", index\_col = "Id")

print(df)

## Código completo del ejercicio 4: