

UNIVERSIDAD NACIONAL DE TRUJILLO

FACULTAD DE INGENIERIA

ESCUELA DE INGENIERÍA MECATRÓNICA



LABORATORIO N°6 – GRUPOS IMPARES

CURSO: PROGRAMACIÓN I

INTEGRANTES:

VERA NORIEGA, BRAYAN STALIN

VÁSQUEZ SILVA, AUGUSTO MARTIN

VENTURA SERRANO, SEBASTIAN

DOCENTES:

ASTO RODRIGUEZ EMERSON MAXIMO

CICLO: III CICLO

TRUJILLO – PERÚ

Julio del 2022

Índice

RESUMEN	3
DESARROLLO DEL LABORATORIO	4
1.1 Resultados de la experiencia	4
Resultado 1:	4
Resultado 2:	5
Resultado 3:	6
Resultado 4:	7
TEST DE COMPROBACIÓN	8
BIBLIOGRAFÍA	10
ANEXOS	10

RESUMEN

Para el presente laboratorio se hará uso de los conocimientos adquiridos en las clases teóricas de la semana 7 y 8 de los temas “Introducción a Python – control de flujo” y “Estructuras de control” haciendo uso anaconda y Python guiándonos así de las grabaciones y explicaciones de internet para poder desarrollar el contenido de cada una de las cuestiones planteadas y adquirir mayores conocimientos en el uso del nuevo programa utilizado, después de analizar las distintas cuestiones hemos resuelto cada una de estas mismas con cada una de las peticiones que se nos menciona; Además logramos desarrollar con éxito el test de comprobación explicando así la diferente entre una lista de Python y un array de Pseint, explicar las diferentes formas de recorrer un diccionario y explicar las diferencias entre tuplas y conjuntos logrando así absorber cada uno de estos conocimientos para el progreso en el curso.

DESARROLLO DEL LABORATORIO

1.1 Resultados de la experiencia

Resultado 1:

Escriba una función que tenga como parámetro un numero de 1 al 9, ante lo cual debe imprimir lo siguiente:

```
Enter the number of rows: 6
  1
 1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
```

```
n=int(input("Ingrese un número de filas del 1 al 9:"))
while n<1 or n>9: #Esto hará que te pida el número infinitas veces hasta
que esté dentro del intervalo
    print("Error")
    n=int(input("Ingrese un número de filas nuevamente:"))

def Piramide (n): #Definimos la función Piramide

    k=n-1

    for i in range (0, n):

        for j in range (0, k):
            print(end=" ")#Agregará un espacio a la izquierda de cada
fila

        k=k-1

        for j in range (0, i+1):

            print (j+1, end=" ")# El end=" " Agregará un espacio entre
los números de las filas
            print ("\r") #Desplazará las filas de números a la izquierda
Piramide(n)
```

```
Ingrese un número de filas del 1 al 9:6
  1
 1 2
 1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
```

Resultado 2:

Escriba un programa que pida como parámetros cuántas listas se quieren crear y las solicite a continuación:

```

Generador de listas
¿Cuántas listas quiere escribir? 2
Dígame cuántas palabras tiene la lista 1: 3
Dígame la palabra 1: Ana
Dígame la palabra 2: Bárbara
Dígame la palabra 3: Carmen
Dígame cuántas palabras tiene la lista 2: 2
Dígame la palabra 1: Alberto
Dígame la palabra 2: Bernardo

La lista 1 es: ['Ana', 'Bárbara', 'Carmen']
La lista 2 es: ['Alberto', 'Bernardo']

```

```

from numpy import insert

m=1

print('Generador de listas')
n =int(input('digame cuantas listas quiere escribir'))
x=1
z=0
listas=list()
while z<n:
    z+=1
    while x < n+1:
        print('Digame cuantas palabras tiene la lista',x)
        a = int(input())
        b=list()
        y=0

        while y<a:
            y+=1
            print('digame la palabra',y)
            c=input()
            b.insert(y,c)

            if y==a-1:
                break

        x+=1
        listas.append(b)
    else:

        for b in listas:
            print('la lista',m, b)

```

```
m+=1
```

```
Generador de listas
digame cuantas listas quiere escribir 3
Digame cuantas palabras tiene la lista 1
4
digame la palabra 1
Doctora
digame la palabra 2
Ingeniero
digame la palabra 3
Arquitecto
digame la palabra 4
Mecanica
Digame cuantas palabras tiene la lista 2
4
digame la palabra 1
ingles
digame la palabra 2
español
digame la palabra 3
matematicas
digame la palabra 4
geografia
Digame cuantas palabras tiene la lista 3
4
digame la palabra 1
celular
audifonos
digame la palabra 3
mouse
digame la palabra 4
microfono
la lista 1 ['Doctora', 'Ingeniero', 'Arquitecto', 'Mecanica']
la lista 2 ['ingles', 'español', 'matematicas', 'geografia']
la lista 3 ['celular', 'audifonos', 'mouse', 'microfono']
```

Resultado 3:

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:

```
{'nombre': 'Emerson', 'dni': 70288113, 'Edad': 31}
{'nombre': 'Jorge', 'dni': 30283113, 'Edad': 40}
{'nombre': 'Maria', 'dni': 50282113, 'Edad': 42}
{'nombre': 'Paola', 'dni': 10284113, 'Edad': 85}
```

```
Datos={}
print ("Ingrese el número de personas: ")
n=int(input())
i=0
while i<n:
    print("Inserte los datos de la persona", i+1)
    nombres=input("Nombre: ")
    dni=(int(input("DNI: ")))
    edades=(int(input("Edad: ")))
```

```

i=i+1
Datos_Registrados={"Nombre": nombres, "DNI": dni,"Edad": edades}
Edad_orden=edades
Datos[Edad_orden]=Datos_Registrados

print("Los datos registrados son:")
for Registro in sorted(Datos.values(), key=lambda x: x["Edad"]):
    print(Registro)

```

```

Ingrese el número de personas:
4
Inserte los datos de la persona 1
Nombre: Lucia
DNI: 95465454
Edad: 21
Inserte los datos de la persona 2
Nombre: Sebastian
DNI: 76511132
Edad: 20
Inserte los datos de la persona 3
Nombre: Paco
DNI: 78454545
Edad: 16
Inserte los datos de la persona 4
Nombre: Tania
DNI: 56987854
Edad: 12
Los datos registrados son:
{'Nombre': 'Tania', 'DNI': 56987854, 'Edad': 12}
{'Nombre': 'Paco', 'DNI': 78454545, 'Edad': 16}
{'Nombre': 'Sebastian', 'DNI': 76511132, 'Edad': 20}
{'Nombre': 'Lucia', 'DNI': 95465454, 'Edad': 21}

```

Resultado 4:

Realice una función que dado n encuentre el término T_n de la secuencia de Tribonacci T_n , la cual se define de la siguiente manera:

$$T_0 = 0, T_1 = 1, T_2 = 1, \text{ and } T_{n+3} = T_n + T_{n+1} + T_{n+2} \text{ for } n \geq 0.$$

```

def tribonacci(n):

    if n==1:
        return 0
    if n==2 or n==3:
        return 1
    else:
        return tribonacci(n-1)+tribonacci(n-2)+tribonacci(n-3)

```

```
print(tribonacci(int(input('indique el tn a encontrar: '))))
# print('indique el Tn a encontrar')
# x=input(int())
# print(tribonacci(x))
```

```
indique el tn a encontrar: 8
24
```

TEST DE COMPROBACIÓN

i) Explique las diferencias entre una lista de Python y un array de PSeint:

Una de las diferencias que denota bastante entre una lista de Python y un array de PSeint es que en PSeint trabaja con la palabra clave Dimensión seguido del arreglo y su tamaño que va en corchetes, esto a su vez hace que el array sea de un tamaño fijo, por lo que en cambio la lista de Python es modificable y esto es gracias a que a diferencia de PSeint que es más básico, en Python existe una gran variedad de métodos que amplifica su complejidad, cómo ejemplo de esto se realizó un código con el comando append para mostrar su capacidad de modificar los datos.

```
lista_ejemplo.py > ...
1 lista=[]
2 elementos=["platano","manzana","sandia","melón"]
3 elementos.append("naranja")
4 print(elementos)
```

PROBLEMS OUTPUT **TERMINAL** JUPYTER DEBUG CONSOLE

D:\CICLO III MECATRÓNICA\PROGRAMACIÓN\LABORATORIO 06>python lista_ejemplo.py
['platano', 'manzana', 'sandia', 'melón', 'naranja']

ii) Explique las diferentes formas de recorrer un diccionario.

Keys:

Este método recorre el diccionario viendo cada clave a la vez.

```
keys.py > [?] Hechos_históricos
1 Hechos_históricos={"Descubrimiento de América":"1942", "Alunizaje Apolo 11":"1969"}
2 Acontecimiento=Hechos_históricos.keys()
3 print(Acontecimiento)
```

PROBLEMS OUTPUT **TERMINAL** JUPYTER DEBUG CONSOLE

D:\CICLO III MECATRÓNICA\PROGRAMACIÓN\LABORATORIO 06>python keys.py
dict_keys(['Descubrimiento de América', 'Alunizaje Apolo 11'])

Values:

Este método recorre el diccionario viendo cada valor uno a uno.

```
values.py > ...
1 Hechos_históricos={"Descubrimiento de América":"1492", "Alunizaje Apolo 11":"1969"}
2 Fecha=Hechos_históricos.values()
3 print(Fecha)

PROBLEMS OUTPUT TERMINAL JUPYTER DEBUG CONSOLE

D:\CICLO III MECATRÓNICA\PROGRAMACIÓN\LABORATORIO 06>python values.py
dict_values(['1492', '1969'])
```

Ítems:

Este método recorre el diccionario viendo tanto la clave como el valor.

```
items.py > ...
1 Hechos_históricos={"Descubrimiento de América":"1492", "Alunizaje Apolo 11":"1969"}
2 Datos=Hechos_históricos.items()
3 print(Datos)

PROBLEMS OUTPUT TERMINAL JUPYTER: VARIABLES DEBUG CONSOLE

D:\CICLO III MECATRÓNICA\PROGRAMACIÓN\LABORATORIO 06>python items.py
dict_items([('Descubrimiento de América', '1492'), ('Alunizaje Apolo 11', '1969')])
```

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

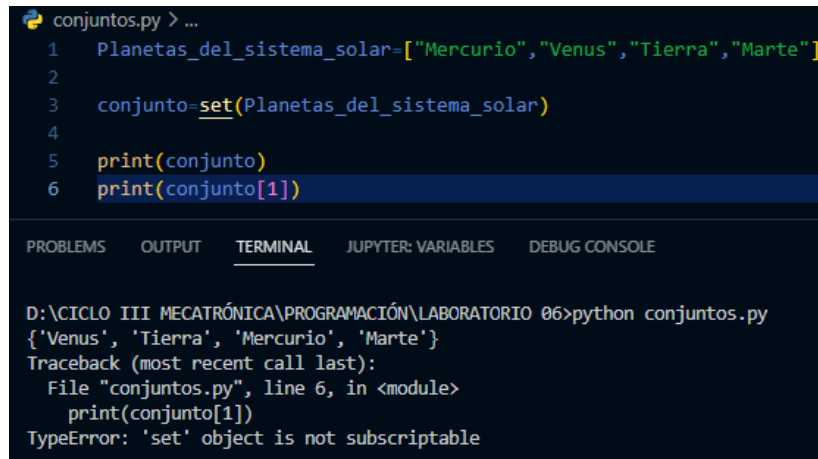
Las tuplas es una estructura de datos ordenada pero no modificable, como, por ejemplo:

```
tuplas.py > ...
1 coordenadas=(125, 247, 43)
2
3 x, y, z = coordenadas
4
5 print(x)
6 print(y)
7 print(z)

PROBLEMS OUTPUT TERMINAL JUPYTER: VARIABLES DEBUG CONSOLE

D:\CICLO III MECATRÓNICA\PROGRAMACIÓN\LABORATORIO 06>python tuplas.py
125
247
43
```

Pero con respecto a los conjuntos, esta no viene a ser ordenada cómo las tuplas, estos no tienen una posición como se puede ver:



```

conjuntos.py > ...
1  Planetas_del_sistema_solar=["Mercurio","Venus","Tierra","Marte"]
2
3  conjunto=set(Planetas_del_sistema_solar)
4
5  print(conjunto)
6  print(conjunto[1])

```

PROBLEMS OUTPUT **TERMINAL** JUPYTER: VARIABLES DEBUG CONSOLE

```

D:\CICLO III MECATRÓNICA\PROGRAMACIÓN\LABORATORIO 06>python conjuntos.py
{'Venus', 'Tierra', 'Mercurio', 'Marte'}
Traceback (most recent call last):
  File "conjuntos.py", line 6, in <module>
    print(conjunto[1])
TypeError: 'set' object is not subscriptable

```

Como se puede observar se presenta un error para el segundo print porque en la estructura de conjuntos no es ordenada siendo la diferencia entre las tuplas.

BIBLIOGRAFÍA

- Listas en Python. (2017, 19 septiembre). DevCode Tutoriales.
<https://devcode.la/tutoriales/listas-python/>
- G. (s. f.). ARREGLOS EN PSeInt 1 Arreglos en PSeInt Los arreglos son . . . - DOCUMENTOP.COM. Documentop.Com.
<https://documentop.com/arreglos-en-pseint-1-arreglos-en-pseint-los-arreglos-son-598c49441723dd5d69644de1.html#:~:text=Para%20crear%20un%20arreglo%20en,En%20Java%20desde%200.>
- Recorrer diccionarios en Python. (s. f.). Nextjournal.
<https://nextjournal.com/jgomo3/recorrer-diccionarios-en-python>
- Montero, J. (2012, 16 agosto). Python: Recorriendo un diccionario de principio a fin | El Club del Autodidacta. El Club del Autodidacta | Polifacetismo y creatividad. <http://elclubdelautodidacta.es/wp/2012/08/python-recorriendo-un-diccionario-de-principio-a-fin/>

ANEXOS

<https://github.com/BRAYAN-VERA/-laboratorio-6-GRUPO03-PI-UNT-2022->.