

Tema:	PROGRAMACIÓN CON PYTHON	Fecha:
Practica	3	31/10/22
Alumna	Nava Gálvez Laura Karen Santillán Muñoz Jacqueline Obregón Cerón Hebert	

INTRODUCCIÓN

Python es un lenguaje de programación ampliamente utilizado en las aplicaciones web, el desarrollo de software, la ciencia de datos y el machine learning (ML). Los desarrolladores utilizan Python porque es eficiente y fácil de aprender, además de que se puede ejecutar en muchas plataformas diferentes. El software Python se puede descargar gratis, se integra bien a todos los tipos de sistemas y aumenta la velocidad del desarrollo.

Python es un lenguaje de programación potente y fácil de aprender. Tiene estructuras de datos de alto nivel eficientes y un simple pero efectivo sistema de programación orientado a objetos. La elegante sintaxis de Python y su tipado dinámico, junto a su naturaleza interpretada lo convierten en un lenguaje ideal para scripting y desarrollo rápido de aplicaciones en muchas áreas, para la mayoría de plataformas.

Python es un lenguaje de programación de alto nivel que se utiliza para desarrollar aplicaciones de todo tipo. A diferencia de otros lenguajes como Java o .NET, se trata de un lenguaje interpretado, es decir, que no es necesario compilarlo para ejecutar las aplicaciones escritas en Python, sino que se ejecutan directamente por el ordenador utilizando un programa denominado interpretador, por lo que no es necesario “traducirlo” a lenguaje máquina.

Los beneficios de Python incluyen los siguientes:

- Los desarrolladores pueden leer y comprender fácilmente los programas de Python debido a su sintaxis básica similar a la del inglés.
- Python permite que los desarrolladores sean más productivos, ya que pueden escribir un programa de Python con menos líneas de código en comparación con muchos otros lenguajes.
- Python cuenta con una gran biblioteca estándar que contiene códigos reutilizables para casi cualquier tarea. De esta manera, los desarrolladores no tienen que escribir el código desde cero.
- Hay muchos recursos útiles disponibles en Internet si desea aprender Python. Por ejemplo, puede encontrar con facilidad videos, tutoriales, documentación y guías para desarrolladores.
- Python se puede trasladar a través de diferentes sistemas operativos de computadora, como Windows, macOS, Linux y Unix.

OBJETIVO

- Desarrollar una interfaz visual en Python que es permita hacer uso de diversos factores de este paradigma de programación.
- Programar un sistema similar a un punto de venta sin bases de datos
- Generar las interfaces necesarias que permitan la manipulación de la información de entrada, dentro del sistema y de salida del sistema
- Crear interfaces claras, intuitivas y atractivas para los usuarios.



DESARROLLO

```
tienda.py - examen.py - Visual Studio Code
EXPLORADOR
EXAMEN PYT
logo.ico
tienda.py
tienda.py
1 from tkinter import *
2 import os
3 import sys
4
5 raiz = Tk()
6
7 #-----X-----INICIO SESION-----
8
9 raiz.title("TIENDA")
10 raiz.resizable(0,0)
11 raiz.iconbitmap("logo.ico")
12 raiz.geometry("800x600")
13 raiz.config(bg="black")
14
15 miFrame=Frame()
16 miFrame.pack()
17 miFrame.config(bg="green")
18 miFrame.config(width="400", height="1")
19 miFrame.place(x="200", y="50", width="400", height="500")
20 miFrame.config(bd=7)
21 miFrame.config(relief="ridge")
22
23 bien=Label(miFrame, text="BIENVENIDO")
24 bien.pack()
25 bien.config(bg="green", fg="white", font=("georgia", 35))
26 bien.place(x="40", y="20")
27
28 usu=Label(miFrame, text="Username:")
29 usu.pack()
30 usu.config(bg="green", fg="white", font=("arial", 10))
31 usu.place(x="40", y="120")
32
33 con=Label(miFrame, text="Password:")
34 con.pack()
35 con.config(bg="green", fg="white", font=("arial", 10))
36 con.place(x="40", y="200")
37
```

```
tienda.py - examen.py - Visual Studio Code
EXPLORADOR
EXAMEN PYT
logo.ico
tienda.py
tienda.py
47 ccon.config(justify= CENTER , show= )
48 ccon.place(x="150", y="200")
49
50 #-----FUNCIONES-----
51
52 def inicio():
53     if user.get()=="Laura" and pas.get()=="NGLKCHSM0301":
54         abrirv()
55     else:
56         err=Label(miFrame, text="ERROR, intento de nuevo!")
57         err.pack()
58         err.config(bg="black", fg="white", font=("arial", 10))
59         err.place(x="120", y="450")
60
61 def abrirv():
62     raiz.withdraw()
63     eleccion=Ioplevel()
64     eleccion.geometry("800x600")
65     eleccion.title("TIENDA")
66     eleccion.resizable(0,0)
67     eleccion.iconbitmap("logo.ico")
68     eleccion.config(bg="gray")
69     def clientes():
70         cli=Ioplevel()
71         cli.geometry("800x600")
72         cli.title("INFORMACION CLIENTES")
73         cli.resizable(0,0)
74         cli.iconbitmap("logo.ico")
75         cli.config(bg="white")
76         inf=Label(cli, text="INFORMACION DE LOS CLIENTES")
77         inf.pack()
78         inf.config(bg="white", fg="blue", font=("georgia", 30))
79         inf.place(x="30", y="20")
80
81         tabla1=ttk.Treeview(cli, columns=("Nombre", "Apellido", "Apellido", "Edad"))
82         tabla1.insert("",END,text="JRI",values=("Juan", "Ramirez", "Perez", "40"))
83         tabla1.insert("",END,text="IL2",values=("Ignacio", "Lopez", "Rodriguez", "34"))
84         tabla1.insert("",END,text="FE3",values=("Eduardo", "Escamilla", "Leon", "28"))
```



```
tienda.py - examen.pyt - Visual Studio Code

EXPLORADOR
EXAMEN PYT
logo.ico
tienda.py 2

tienda.py > ...

157
158 #-----ID DEL PRODUCTO-----
159 cproid=Entry(vent, textvariable=idp)
160 cproid.grid(row=0)
161 cproid.pack()
162 cproid.config(justify="center")
163 cproid.place(x="40", y="130")
164
165 #-----NOMBRE PRODUCTO-----
166 cnomp=Entry(vent, textvariable=nomproducto)
167 cnomp.pack()
168 cnomp.config(justify="center")
169 cnomp.place(x="470", y="130")
170
171 #-----CANTIDAD VENDIA-----
172 ccamp=Entry(vent, textvariable=cantidad)
173 ccamp.pack()
174 ccamp.config(justify="center")
175 ccamp.place(x="40", y="210")
176
177 inf3= Label(vent, text="Productos Vendidos")
178 inf3.pack()
179 inf3.config(bg="white", fg="black", font=("georgia", 10))
180 inf3.place(x="330", y="230")
181
182 listbox=Listbox(vent, width="120", height="16")
183 listbox.place(x="40", y="260")
184
185 def ticket():
186     tick=TopLevel()
187     tick.title("TICKET")
188     tick.resizable(0,0)
189     tick.iconbitmap("logo.ico")
190     tick.geometry("400x500")
191     tick.config(bg="white")
192     valor_entrada=cnomp.get()
193     valor2_entrada=ccamp.get()
```

```
tienda.py - examen.pyt - Visual Studio Code

EXPLORADOR
EXAMEN PYT
logo.ico
tienda.py 2

tienda.py > ...

246 total.config(bg="white", fg="black", font=("arial", 10))
247 total.place(x="40", y="330")
248
249 gr=Label(tick, text="GRACIAS POR SU COMPRA")
250 gr.pack()
251 gr.config(bg="white", fg="black", font=("arial", 15))
252 gr.place(x="50", y="420")
253
254 pri=Label(tick, text=valor_entrada)
255 pri.pack()
256 pri.config(bg="white", fg="black", font=("arial", 10))
257 pri.place(x="200", y="150")
258
259 pri1=Label(tick, text=valor2_entrada)
260 pri1.pack()
261 pri1.config(bg="white", fg="black", font=("arial", 10))
262 pri1.place(x="200", y="180")
263
264 if valor3_entrada=="ANSK":
265     pri2=Label(tick, text="$1000")
266     pri2.pack()
267     pri2.config(bg="white", fg="black", font=("arial", 10))
268     pri2.place(x="200", y="210")
269
270 pri3=Label(tick, textvariable=variable1)
271 pri3.pack()
272 pri3.config(bg="white", fg="black", font=("arial", 10))
273 pri3.place(x="200", y="240")
274
275 pri4=Label(tick, textvariable=variable1)
276 pri4.pack()
277 pri4.config(bg="white", fg="black", font=("arial", 10))
278 pri4.place(x="200", y="270")
279
280 pri5=Label(tick, textvariable=variable2)
281 pri5.pack()
282 pri5.config(bg="white", fg="black", font=("arial", 10))
283 pri5.place(x="200", y="300")
```



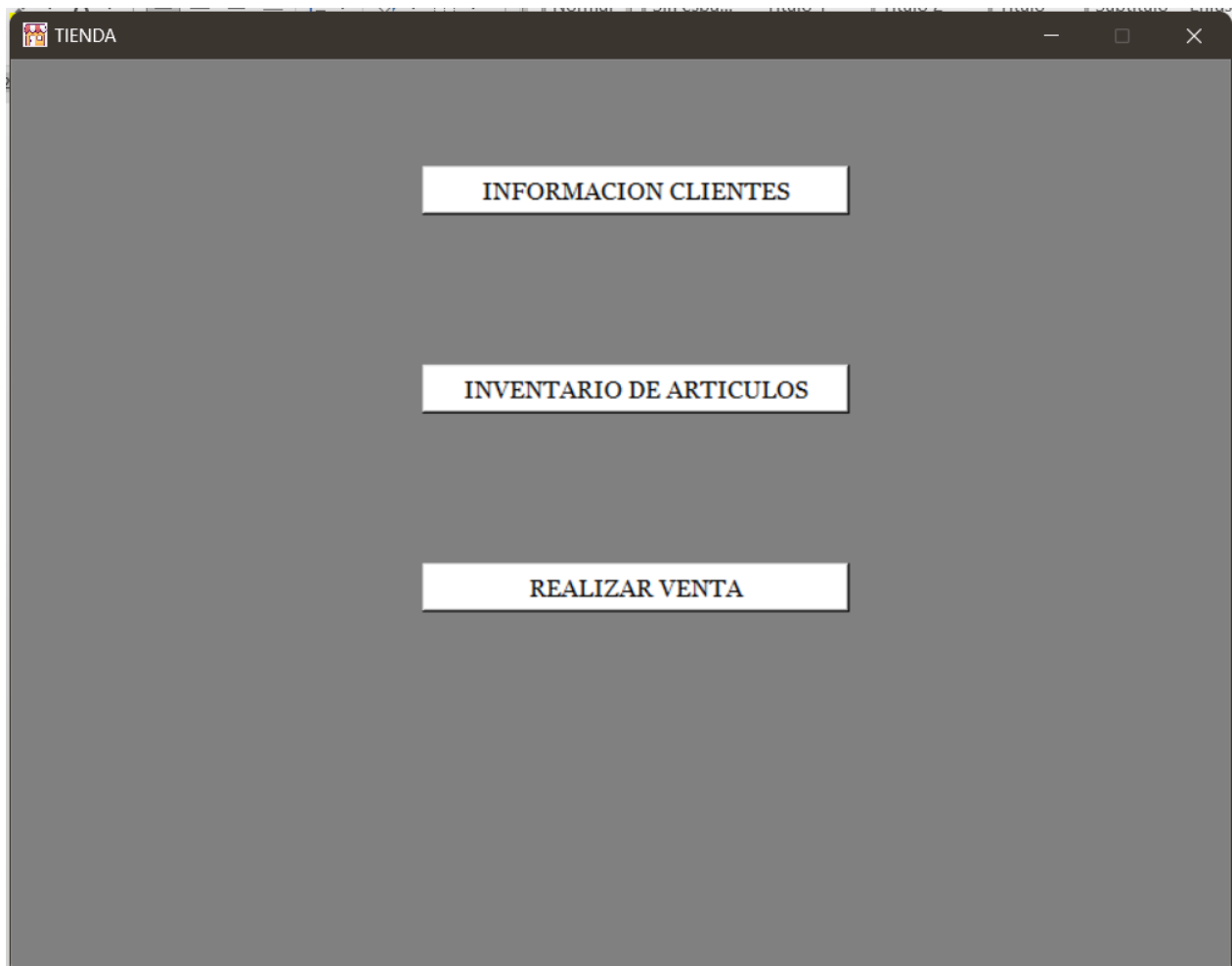
RESULTADO

TIENDA

BIENVENIDO

Username:

Password:





PRODUCTOS

ID	Nombre	Color	Cantidad
ABCD	Lapiz	Negro	20
DEFG	Libreta	Rojo	10
HIJK	Hojas	colores	7
LMNO	Goma	Blanca	8
PQRS	Tijeras	Colores	4
TUVW	Resistol	Verde	28
XYZA	Cartulina	Colores	16
BCDE	USB	Azul	30
FGHI	Micro SD	Negro	40
JKLM	Mapas	Colores	13

INFORMACION DE LOS CLIENTES

ID	Nombre	Apellido Paterno	Apellido Materno
MM1	Maria	Martínez	Perez
AL2	Alicia	Lopez	Rodriguez
JS3	Javier	Suarez	Leon
VO4	Victor	Olvera	Lopez
PG5	Patricia	Gonzales	Hernandez

VENTAS

VENTA

Cual es el ID del producto que deseas vender?

ABCD

Cual es el nombre del producto?

Lapiz

¿Que cantidad sera vendida?

2

Productos Vendidos

{ID del producto es: } ABCD {Nombre del producto es: } Lapiz {Cantidad del producto es: } 2

REALIZAR VENTA

GENERAR TICKET

VENTAS

VENTA

Cual es el ID del producto que deseas vender?

DEFG

Cual es el nombre del producto?

Libreta

¿Que cantidad sera vendida?

3

Productos Vendidos

{ID del producto es: } DEFG {Nombre del producto es: } Libreta {Cantidad del producto es: } 3
{ID del producto es: } ABCD {Nombre del producto es: } Lapiz {Cantidad del producto es: } 2

REALIZAR VENTA

GENERAR TICKET

TICKET

IVA: \$ 3.2

TICKET

Papeleria UAEH

Producto vendido: Lapiz

Cantidad vendida: 2

Precio Unitario: \$20

Importe: \$ 2.0

Subtotal: \$ 2.0

TOTAL: \$ 23.2

GRACIAS POR SU COMPRA

TICKET

IVA: \$

TICKET

Papeleria UAEH

Producto vendido: Libreta

Cantidad vendida: 3

Precio Unitario: \$30

Importe: \$ 12.0

Subtotal: \$ 12.0

14.4

TOTAL: \$ 104.4

GRACIAS POR SU COMPRA

DISCUSIÓN

Después de realizar la practica pudimos aprender más la programación de Python y las diferentes librerías que utilizan, además de conocer mejor su estructura y la diferente forma de programación a la de otros lenguajes, aprendimos más sobre sus medos y estructura y su manera de utilizarlas y aplicarlas en la práctica, se realizaron varias investigaciones para conocer de mejor manera el lenguaje y de esa forma poder realizarla, a que anteriormente se tenía un conocimiento previo pero no tan detallado como lo tenemos ahora después de realizar esta práctica y documentación. Nos agradó mucho realizarla y ahora ampliamos más nuestros conocimientos por lo que fue de gran utilidad para aprender más.

CUESTIONARIO.

1.- ¿Qué paradigmas de programación utiliza Python?

Python es un lenguaje multiparadigma, lo cual utiliza distintos tipos de paradigmas como lo es la programación orientada a objetos ya que todo en Python es un objeto, aunque también tiene programación imperativa, funcional, procedural y reflexiva

2.- ¿Cuál es la clasificación de los tipos de datos estándar en Python?

No solo en Python, en la mayoría de Lenguajes de Programación los datos se encuentran dentro de una variable, una vez que un dato esta dentro de una variable es posible usarla para un determinado objetivo dentro del proyecto.

Para describir los tipos de datos a continuación usaré variables.

String: Este tipo de datos son de tipo caracteres o textos, se pueden declarar dentro de una variable usando comillas simples o dobles.

Números: Los datos de tipo número son infaltables en cualquier Lenguaje de Programación, ya que nos permiten realizar cálculos importantes dentro de un proyecto, este tipo de datos se puede declarar dentro de una variable, no es necesario el uso de comillas.

Diccionarios: Este tipo de dato está compuesto por un par de elementos que son clave: valor. Se parece a un array asociativo en el cual cada elemento o clave guarda un valor en su interior.

Tuplas (Tuples): Este tipo de datos son como las listas, las cuales contienen una colección de elementos de diferentes tipos de datos, estos elementos se separan con una coma y se encierran dentro de paréntesis.

Listas: Este tipo de dato es similar a una matriz en el Lenguaje de Programación C, estas pueden contener distintos tipos de datos y se separan por comas

3.- ¿Cuáles son las palabras reservadas en Python, (escribe el listado y una breve descripción)?

- **False** – Valor booleano, resultado de operaciones de comparación u operaciones lógicas en Python
- **None** – Representa a un valor nulo
- **True** – Valor booleano, igual que false, resultado de operaciones de comparación u operaciones lógicas en Python
- **__peg_parser__** – Llamado huevo de pascua, relacionado con el lanzamiento del nuevo analizador PEG no está definido aún.
- **And** – Operador lógico
- **As** – Se utiliza para crear un alias al importar un módulo.
- **Assert** – Se utiliza con fines de depuración
- **Async** – Proporcionada por la biblioteca ‘asyncio’ en Python. Se utiliza para escribir código concurrente en Python
- **Await** – Proporcionada por la biblioteca ‘asyncio’ en Python. Se utiliza para escribir código concurrente en Python
- **Break** – Se utiliza en el interior de los bucles for y while para alterar su comportamiento normal
- **Class** – Se usa para definir una nueva clase definida por el usuario
- **Continue** – Se utiliza en el interior de los bucles for y while para alterar su comportamiento normal
- **Def** – se usa para definir una función definida por el usuario
- **Del** – Para eliminar un objeto
- **Elif** – Se usa en declaraciones condicionales, igual ‘else’ e ‘if’
- **Else** – Se usa en declaraciones condicionales, igual ‘elif’ e ‘if’
- **Except** – Se usa para crear excepciones, qué hacer cuando ocurre una excepción, igual que ‘raise’ y ‘try’
- **Finally** – Su uso garantiza que el bloque de código dentro de él se ejecute incluso si hay una excepción no controlada
- **For** – Utilizado para hacer bucles. Generalmente lo usamos cuando sabemos la cantidad de veces que queremos que se ejecute ese bucle
- **From** – Para importar partes específicas de un módulo

- **Global** – Para declarar una variable global.
- **If** – Se usa en declaraciones condicionales, igual ‘else’ y ‘elif’
- **Import** – Para importar un módulo
- **In** – Para comprobar si un valor está presente en una lista, tupla, etc. Devuelve True si el valor está presente, de lo contrario devuelve False
- **Is** – Se usa para probar si las dos variables se refieren al mismo objeto. Devuelve True si los objetos son idénticos y False si no
- **Lambda** – Para crear una función anónima
- **Nonlocal** – Para declarar una variable no local
- **Not** – Operador lógico
- **Or** – Operador lógico
- **Pass** – Es una declaración nula en Python. No pasa nada cuando se ejecuta. Se utiliza como marcador de posición.
- **Raise** – Se usa para crear excepciones, qué hacer cuando ocurre una excepción, igual que ‘except’ y ‘try’
- **Return** – Se usa dentro de una función para salir y devolver un valor.
- **Try** – Se usa para crear excepciones, qué hacer cuando ocurre una excepción, igual que ‘raise’ y ‘except’
- **While** – Se usa para realizar bucles.
- **With** – Se usa para simplificar el manejo de excepciones
- **Yield** – Se usa dentro de una función al igual que ‘return’, salvo que ‘yield’ devuelve un generador.

4.- ¿Qué operadores de asignación se pueden utilizar en Python?

- **Operador =**

Asigna a la variable de la izquierda el contenido que le ponemos a la derecha.

- **Operador +=**

El operador += en $x+=1$ es equivalente a $x=x+1$. Se puede jugar un poco con el operador += y aplicarlo sobre variables que no necesariamente son números.

- **Operador -=**

El operador -= es equivalente a restar y asignar el resultado a la variable inicial. Es decir, $x-=1$ es equivalente a $x=x-1$. El operador es muy usado para decrementar el valor de una variable.

- **Operador *=**

El operador *= equivale a multiplicar una variable por otra y almacenar el resultado en la primera, es decir $x*=2$ equivale a $x=x*2$.

- **Operador /=**

El operador /= equivale a dividir una variable por otra y almacenar el resultado en la primera, es decir, $x/=2$ equivale a $x=x/2$.

- **Operador %=**

El operador %= equivale a hacer el módulo de la división de dos variables y almacenar su resultado en la primera.

- **Operador //=**

El operador //= realiza la operación cociente entre dos variables y almacena el resultado en la primera. El equivalente de $x//=2$ sería $x=x/2$.

- **Operador **=**

El operador **= realiza la operación exponente del primer número elevado al segundo, y almacena el resultado en la primera variable. El equivalente de $x**=2$ sería $x=x**2$.

- **Operador &=**

El operador &= realiza la comparación & bit a bit entre dos variables y almacena su resultado en la primera. El equivalente de $x\&=1$ sería $x=x\&1$

- **Operador |=**

El operador |= realiza el operador | elemento a elemento entre dos variables y almacena su resultado en la primera. El equivalente de $x|=2$ sería $x=x|2$

- **Operador ^=**

El operador ^= realiza el operador ^ elemento a elemento entre dos variables y almacena su resultado en la primera. El equivalente de $x^=2$ sería $x=x^2$

- **Operador» =**

El operador `>>=` es similar al operador `>>` pero permite almacenar el resultado en la primera variable. Por lo tanto, `x>>=3` sería equivalente a `x=x>>3`. Es importante tener cuidado y saber el tipo de la variable `x` antes de aplicar este operador, ya que se podría dar el caso de que `x` fuera una variable tipo `float`. En ese caso, tendríamos un error porque el operador `>>` no está definido para `float`.

- **Operador «=**

Muy similar al anterior, `<<=` aplica el operador `<<` y almacena su contenido en la primera variable. El equivalente de `x<<=1` sería `x=x<<1`

5.- ¿Cómo se realizan los comentarios de línea y multilinea en Python?

- Escribiendo el símbolo de numeral (#) al comienzo de la línea de texto donde queremos nuestro comentario.
- Escribiendo triple comilla (') al principio y al final del comentario, en este caso los comentarios pueden ocupar más de una línea.

6.- ¿Qué es una lista en Python y describe los métodos que utiliza para la lista?

Las listas en Python son un tipo contenedor, compuesto, que se usan para almacenar conjuntos de elementos relacionados del mismo tipo o de tipos distintos.

Junto a las clases *tuple*, *range* y *str*, son uno de los tipos de secuencia en Python, con la particularidad de que son mutables. Esto último quiere decir que su contenido se puede modificar después de haber sido creada.

- `append()` : Añade un ítem al final de la lista
- `clear()` : Vacía todos los ítems de una lista
- `extend()` : Une una lista a otra
- `count()` : Cuenta el número de veces que aparece un ítem
- `index()` : Devuelve el índice en el que aparece un ítem (error si no aparece)
- `insert()` : Agrega un ítem a la lista en un índice específico
- `pop()` : Extrae un ítem de la lista y lo borra

- `remove()` : Borra el primer ítem de la lista cuyo valor concuerde con el que indicamos
- `reverse()` : Le da la vuelta a la lista actual
- `sort()` : Ordena automáticamente los ítems de una lista por su valor de menor a mayor:

Para crear una lista en Python, simplemente hay que encerrar una secuencia de elementos separados por comas entre paréntesis cuadrados [].

7.- ¿Qué es una tupla y cuáles son los métodos que se utilizan en Python para las tuplas?

La clase tuple en Python es un tipo contenedor, compuesto, que en un principio se pensó para almacenar grupos de elementos heterogéneos, aunque también puede contener elementos homogéneos.

Junto a las clases list y range, es uno de los tipos de secuencia en Python, con la particularidad de que son inmutables. Esto último quiere decir que su contenido NO se puede modificar después de haber sido creada.

En general, para crear una tupla en Python simplemente hay que definir una secuencia de elementos separados por comas

Método	Descripción
<code>index(elemento)</code>	Obtiene el índice de la primera ocurrencia del elemento en la tupla. Si el elemento no se encuentra, se lanza la excepción <code>ValueError</code> .
<code>count(elemento)</code>	Devuelve el número de ocurrencias del elemento en la tupla.



8.- ¿Qué son los conjuntos en Python y cuáles son los métodos que se utilizan para los conjuntos?

Representan un objeto capaz de almacenar elementos únicos, que además están indexados, por lo que son de gran utilidad para buscar elementos. En Python los conjuntos pueden almacenar valores numéricos o de cualquier tipo, siempre y cuando los valores sean hasheables.

9.- ¿Qué estructuras de control utilizaste en el desarrollo de la practica?


Se utilizaron IF-ELSE Anidados para el control de los productos registrados.

10.- Si utilizaste alguna de las funciones de Python, describe el procedimiento

No se utilizó ninguna de las funciones de Python realizado para utilizarla.



REFERENCIAS

 *Operadores de asignación*. (s. f.). El Libro De Python. Recuperado 1 de noviembre de 2022, de <https://ellibrodepython.com/operadores-asignacion>

El tutorial de Python — documentación de Python - 3.11.0. (s. f.). Recuperado 31 de octubre de 2022, de <https://docs.python.org/es/3/tutorial/>

Peña, M. J. (2022, 26 agosto). *Palabras reservadas en Python y su significado* / EIP. Másteres Online N° 1 Empleabilidad. <https://eiposgrados.com/blog-python/palabras-reservadas-python/>

¿Qué es Python? / Guía de Python para principiantes de la nube / AWS. (s. f.). Amazon Web Services, Inc. Recuperado 31 de octubre de 2022, de <https://aws.amazon.com/es/what-is/python/>

Santander Universidades. (2022, 16 septiembre). *¿Qué es Python? / Blog*. Becas Santander. <https://www.becas-santander.com/es/blog/python-que-es.html>

uniwebsidad. (s. f.). *7.1. Tuplas (Algoritmos de Programación con Python)*. <https://uniwebsidad.com/libros/algoritmos-python/capitulo-7/tuplas>

Conjuntos en Python – set. (2021, 14 febrero). El Pythonista. <https://elpythonista.com/conjuntos-en-python-set>

Colectiva, N. (2019, 11 noviembre). *Tipos de Datos Estándar en Python*. Blog Nube Colectiva. <https://blog.nubecolectiva.com/tipos-de-datos-estandar-en-python/>