

Proyecto

# Introducción A La Programación I Cuatrimestre

Camila Argeñal Coronado

Eduardo miranda mora

Jordan Ivan Soto Morales

Samantha Perez Rojas

Universidad Fidelitas

Sede San Pedro

Prof. José Antonio Ortega Gonzalez

6 de Marzo del 2024

Codigo acceso a Git hub Repositorio.

<https://github.com/ivansotomoraes31/python-project-4.git>

*Avance I. 8 de marzo del 2024*

## Código

```
inventario_vehiculos = []
Registro_Cedulas = []

def agregar_vehiculo(marca, año, modelo, cilindraje, precio_alquiler, precio_vehiculo,
placa, cantidad):
    if placa not in inventario_vehiculos:
        inventario_vehiculos[placa] = {'marca': marca,
                                         'año': año,
                                         'modelo': modelo,
                                         'cilindraje': cilindraje,
                                         'precio_alquiler': precio_alquiler,
                                         'precio_vehiculo': precio_vehiculo,
                                         'cantidad_disponible': cantidad,
                                         'habilitado': True}
        print("Vehiculo agregado exitosamente")

def reservar_vehiculo(placa, cantidad):
    if placa in inventario_vehiculos:
        if inventario_vehiculos[placa]['habilitado']:
            if inventario_vehiculos[placa]['cantidad_disponible'] >= cantidad:
                inventario_vehiculos[placa]['cantidad_disponible'] -= cantidad
                print("Reserva realizada correctamente.")
            else:
                print("No hay suficientes vehiculos ")
        else:
            print("El vehiculo esta inhabilitado.")
    else:
        print("No existe ningun vehiculo con esa placa.")

def inhabilitar_vehiculo(placa):
    if placa in inventario_vehiculos:
        inventario_vehiculos[placa]['habilitado'] = False
        print("Vehiculo inhabilitado correctamente.")
    else:
        print("No se encontro ningun vehiculo con esa placa.")

def menu_administrador():
    opc = ""
    while opc != "3":
        print()
        print("-----Menu-----")
        print("[1] Gestion inventario Vehiculos")
        print("[2] Gestion de clientes")
        print("[3] Visualizar vehiculos ")
        opc = input("Seleccione una opcion: ")

    if opc == "1":
        print("----Gestion inventario vehiculos---")
        print("[1] Agregar vehiculos")
```

```

print("[2] Inhabilitar vehiculos")
opc_inventario = input("Seleccione una opcion: ")

if opc_inventario == "1":
    print()
    print("Agrega el vehiculo que deseas")
    Marca = input("Ingrese la marca del vehiculo: ")
    Año = input("Ingrese el año del vehiculo: ")
    Modelo = input("Ingrese el modelo del vehiculo: ")
    cilindraje = input("Ingrese el cilindraje del modelo: ")
    Precio_alquiler = float(input("Ingrese el precio de al alquiler: "))
    Precio_auto = float(input("Ingrese el precio del auto: "))
    placa = input("Ingrese el numero de placa")
    cantidad = float(input("Ingrese la cantidad de vehiculos disponibles"))
    agregar_vehiculo(Marca, Año, Modelo, cilindraje, Precio_alquiler, Precio_auto,
placa, cantidad)
    elif opc_inventario == "2":
        placa = input("Ingrese la placa del vehiculo a inhabilitar: ")
        inhabilitar_vehiculo(placa)

elif opc == "2":
    print("Gestion de clientes")
    print("----Menu-----")
    print("[1] Ingresar como invitado")
    print("[2] Ingresar como cliente registrado")
    cli_opc = input("Seleccione una opcion para la gestion de clientes: ")

    if cli_opc == "1":
        print("Cliente invitado")
        print("Seleccione una opcion")
        print("[1]Ver listado de vehículos")
        print("[2] ver reservas")
        reservas = input("Elige que deseas ver")

    elif cli_opc == "2":
        print("Iniciar sesion")
        Nombre = input("Ingrese su nombre:")
        NumCedula = input("Ingrese su numero de cedula:")
        print("Seleccione una opcion")
        print("[1]Ver listado de vehículos")
        print("[2] ver reservas")
        reservas = input("Elige que deseas ver")

    if NumCedula in Registro_Cedulas:
        print("Bienvenido de nuevo",Nombre)

    else:
        print("Cedula no encontrada, porfavor registre")
        Registro_Cedulas.append(NumCedula)
        Nombre = input("Ingrese su nombre:")
        Telefono = input("Ingrese su numero de telefono:")
        print("Se registro correctamente ")

elif opc == "3":

```

```
print("Visualizar vehículos")
for i in inventario_vehiculos:
    print(i)
print("Estos son los vehiculos")
```

```
menu_administrador()
```

## Requerimientos técnicos

### 1. Gestión inventario vehículos

#### 1.1 Diseño

```
inventario_vehiculos = []
Registro_Cedulas = []

def agregar_vehiculo(marca, año, modelo, cilindraje, precio_alquiler, precio_vehiculo,
placa, cantidad):
    if placa not in inventario_vehiculos:
        inventario_vehiculos[placa] = {'marca': marca,
                                        'año': año,
                                        'modelo': modelo,
                                        'cilindraje': cilindraje,
                                        'precio_alquiler': precio_alquiler,
                                        'precio_vehiculo': precio_vehiculo,
                                        'cantidad_disponible': cantidad,
                                        'habilitado': True}
        print("Vehiculo agregado exitosamente")

def reservar_vehiculo(placa, cantidad):
    if placa in inventario_vehiculos:
        if inventario_vehiculos[placa]['habilitado']:
            if inventario_vehiculos[placa]['cantidad_disponible'] >= cantidad:
                inventario_vehiculos[placa]['cantidad_disponible'] -= cantidad
                print("Reserva realizada correctamente.")
            else:
                print("No hay suficientes vehiculos ")
        else:
            print("El vehiculo esta inhabilitado.")
    else:
        print("No existe ningun vehiculo con esa placa.")

def inhabilitar_vehiculo(placa):
    if placa in inventario_vehiculos:
        inventario_vehiculos[placa]['habilitado'] = False
        print("Vehiculo inhabilitado correctamente.")
    else:
        print("No se encontro ningun vehiculo con esa placa.")

def menu_administrador():
    opc = ""
    while opc != "3":
```

```

print()
print("-----Menu-----")
print("[1] Gestion inventario Vehiculos")
print("[2] Gestion de clientes")
print("[3] Visualizar vehiculos ")
opc = input("Seleccione una opcion: ")

if opc == "1":
    print("----Gestion inventario vehiculos---")
    print("[1] Agregar vehiculos")
    print("[2] Inhabilitar vehiculos")
    opc_inventario = input("Seleccione una opcion: ")

    if opc_inventario == "1":
        print()
        print("Agrega el vehiculo que deseas")
        Marca = input("Ingrese la marca del vehiculo: ")
        Año = input("Ingrese el año del vehiculo: ")
        Modelo = input("Ingrese el modelo del vehiculo: ")
        cilindraje = input("Ingrese el cilindraje del modelo: ")
        Precio_alquiler = float(input("Ingrese el precio de al alquiler: "))
        Precio_auto = float(input("Ingrese el precio del auto: "))
        placa = input("Ingrese el numero de placa")
        cantidad = float(input("Ingrese la cantidad de vehiculos disponibles"))
        agregar_vehiculo(Marca, Año, Modelo, cilindraje, Precio_alquiler, Precio_auto,
placa, cantidad)
    elif opc_inventario == "2":
        placa = input("Ingrese la placa del vehiculo a inhabilitar: ")
        inhabilitar_vehiculo(placa)

```

### 1.1.1 Codigo usados

#### 1.1.1.1 Funcion Def

<pre>def agregar_vehiculo(ma rca, año, modelo, cilindraje, precio_alquiler, precio_vehiculo, placa, cantidad):</pre>	<pre>Def nombre_funcion(paramet ros):      contenido</pre>
--	--

#### 1.1.1.1.1 Forma de aplicación y base (se uso en momentos determinados del codigo)

Este en términos explicativos es un código cullo propósito es como su diminutivo lo indica crear una nueva función, donde dentro de esta se van a ejecutar una serie de instrucciones, esto se suele utilizar para darle mas automatización a los códigos, creando caminos o “funciones automáticas” para entregar un resultado proveniente de la información que se le brinde a esa función , se sabrá que se esta presenciando una función ya que aunque se les bauticé con un nombre para su identificación siempre se crean con la palabra reservada

**“def”**. en las condiciones para usarla están: brindarles parámetros que corresponderán a la información-variables que van a manipular dependiendo de la acción que se quiera realizar. Dentro de las variables podemos decir que se pueden usar condicionales, procedimientos, mostrar y agregar etc.

En este código se introdujo de forma externa del código antes del área de código correspondiente al menú principal, mas eso no signifique que se valla a mostrar.

El código, permanece inmóvil a menos que sea llamado por decisión del usuario, recibe los datos, en las veces que se uso para el propósito de “gestionar vehículos” se les brindo parámetros de condicionales donde se insertaría el elementos por el cual se quiere guiar, de ese punto definirá sus acciones de guardado de información en arreglos(del cual se hablara después) mostrar información y eliminar información, también a partir de que sus condicionales lo permitan.

#### 1.1.1.1.2 Propósito

Su propósito en este código fue variado, se uso la definición de funciones para guardar información de los vehículos los cuales luego serian ingresados a una lista, su propósito principal también fue darle al código mas automatización, legibilidad y practicidad.

Un ejemplo de su propósito fue el “agregar vehículos”: se llama a esta función desde el código principal, si la placa no se encontraba anteriormente registrada se pasa a una serie de agregados de variables que luego serán incorporados en una lista la cual almacena los conjuntos de informaciones también se guía por medio de la variable “placa” la cual es un gran determinante en muchas funciones y procedimientos dentro del código principal a lo largo de este.

En la función definida como “reserva de vehículos”: se procede a una serie de condiciones que le mostrara al usuario si su vehículo que quiere reservar estaba previamente disponible, en el caso de que no fuera asi se mostraría un mensaje diciéndole que su “vehículo no se encontraba disponible” , entre otras opciones dependiendo del estado del vehículo que fue buscado por los parámetros de “placa” y cantidad

Se utiliza en inhabilitar vehículos donde después de ser la función llamada procede a tomar una serie de condiciones que van a buscar su parámetro y definirán si se encuentra o no, definiendo de esta forma la “deshabilitarían” (se dará una explicación después)

En resumidas cuentas, el propósito de este código acerca de “definir funciones” se encargara de automatizar el código, brindando funciones que si son llamadas significan que deben cumplir con un camino de acciones y que cabe recalcar devolverán al usuario un resultado en este caso también que afectaran códigos como listas externas para cumplir con el propósito previamente definido (afectando las variables en el proceso manipulándolas, todo dependiendo de la acción a elegir) .

Su uso fue de agregar vehículo, reservar vehículos y inhabilitar vehículos dependiendo de los que el usuario pida, será llamada la función que empezara a ejecutarse para depues devolver un resultado.

### 1.1.1.2 condicional if

if placa not in inventario_vehiculos: if placa in inventario_vehiculos:	if elemento in conjunto_elementos:
--	---------------------------------------

#### 1.1.1.2.1 Forma de aplicación y base

La forma de un if se limita a darle una variable y una condición o serie de condiciones específicas que debe cumplir.

En este código se aplica de forma que de la revisión de una variable en un conjunto de datos en una lista, el código procede (dependiendo de la función que se llame y dependiendo de si la variable ya se encontraba) procede a lanzar datos respectivos a la variable que se dio en la condicional.

En este código podemos caracterizar que usa en una ocasión la diferencia de “if \_\_\_\_ not in \_\_\_\_:” lo cual se refiere a que si la variable no se encontraba en tal conjunto de datos procede a ejecutar acciones, al contrario del código “if \_\_\_\_ in \_\_\_\_:” que verifica que la variable ya sea existente para poder proseguir.

También se agrega el decir que esta codoco se utilizo aparte de las funciones, en un sub menú, mostrado después del menú principal, en opción de “gestión de inventario” seguidamente con los propósitos de que acciones ejecutar según los escrito por el usuario en los “inputs”.

#### 1.1.1.2.2 Propósito

Los propósitos que se le dieron en este código fueron de verificar la existencia de vehículos, en las funciones y después devolver los datos si las variables cumplían con la condición de existir o no dentro de un conjunto de una lista.

Ejemplo de lo dicho antes es su forma de ser aplicada en la función de agregar vehiculos, antes de que se agregue un vehículo, se pasa por la condicional “if \_\_\_\_ not in \_\_\_\_” esto definirá que la variable ya existía antes, para negar el agregado, todo determinado por una variable específica, que sería la variable principal, única y determinante “placa”.

Agregamos que esta variable fue la determinando también en ambas condicionales que se usaron en las funciones de reserva, agregado y inhabilitar.

en la función de inhabilitar se verifica que la variable existiera previamente en la lista respectiva donde se buscaría y luego procedería a eliminarla, en el caso que no, se mandaría un mensaje informando que no existía tal variable en primer lugar.

Un caso particular y específico fue en la función “reserva de vehículo” en la cual el elementos que se quería reservar debe cumplir con varias condiciones creando un subtipo de “if anidado”, dependiendo de la condición que cumpla que mandaría el mensaje de disposición.

Resumiendo su propósito en los menús es dar opciones múltiples y sub-menus para el usuario que el elegirá y que luego dependiendo de estas decisiones, como se dijo anteriormente se llamaran a las funciones.

### 1.1.1.3 False

Dato= false
-------------

#### 1.1.1.3.1 Forma de aplicación y base

El false es un tipo de dato, su función se limita a decir si un dato es como la palabra lo dice “verdadero o falso”

Se aplica en el código dentro de la variable “def inhabilitar\_vehiculo(placa):”, donde después de cumplir con condicionales específicas que prueben la existencia de la variable y ser buscada dentro de las listas, se procedería a convertir el dato a falso incluyendo toda la demás información en conjunto que lo acompañan.

#### 1.1.1.3.2 Propósito

Su propósito fue convertir los elementos “vehículos” a falsos, esto automáticamente deshabilitaría los vehículos para no ser opcionales en otras ejecuciones.

#### 1.1.1.4 Llamado de funciones (usado varias veces en el código)

agregar_vehiculo( Marca, Año, Modelo, cilindraje, Precio_alquiler, Precio_auto, placa, cantidad)	Función(parametro1,pa rametro2)
--	------------------------------------

#### 1.1.1.4.1 Forma de aplicación y base

El llamado de función se aplica colocando el nombre de la función previamente hecha, luego dándole los datos con los que se desea trabajar, luego los datos viajan afuera del código y regresan información ya sean mensajes o simplemente realizan cambios fuera de la vista general del código

#### 1.1.1.4.2 Propósito

Su propósito fue llamar a las funciones, se llamaría a una función dependiendo de lo que el usuario haya seleccionado en el sub menú destinado a la opción 1 en el menú principal (se hablara de este tema después) cada llamado de función tienen la misma finalidad; llamar a la función fuera del código principal, brindarle a la función los datos que necesita para trabajar y luego de que realizara su trabajo se devolviera al menú principal a devolver algún dato o mensaje y haber realizado cambios en los códigos como listas.

En resumidas cuentas su propósito central es llamar funciones para realizar un trabajo.

## 1.2 Planeamiento

### 1.2.1 identificación del problema

los problemas que se ven a lo largo del código son los siguientes

- Definir acciones dependiendo de la solicitud del usuario
- Condiciones que se evaluarán si se puede proseguir con lo que el usuario desea
- Existencia de los elementos que se desean buscar, inhabilitar o reservar

### 1.2.2 acciones dependiendo del problema

- Definir acciones dependiendo de la solicitud del usuario



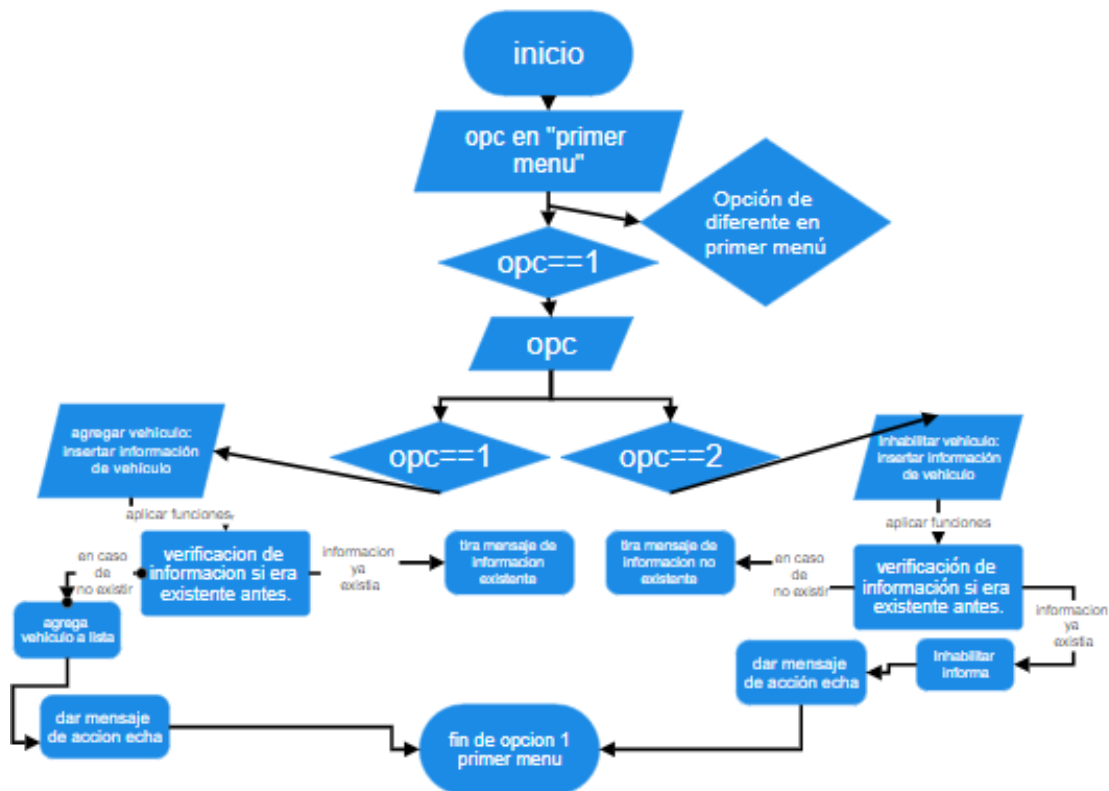
- Se dio solución a este problema con el uso de condicionales ifs los cuales evalúan la información otorgada en el input donde se pregunta al usuario la opción que desea y este escribirá una respuesta que coincida con las opciones programadas.
- Existencia de los elementos que se desean buscar, inhabilitar o reservar
  - Esto es referente a si los elementos y su conjunto respectivo que se necesitan están en las condiciones necesarias (existentes o no), se manejo este problema con (igualmente) el uso de “ifs” pero esta vez aplicándolas de forma que evalúen la existencia en una lista donde se almacenan todas las variables entre estas buscando la variable principal.

## 1.3 Algoritmo

- Pasos a seguir dependiendo de las condiciones y objetivo principal
  1. Pedir opción en menú principal
  2. comparo la opción de menú principal
  3. dar opciones de que tipo de gestión desea realizar
  4. comparo la opción de sub-menú
  5. (en caso de decisión de agregar vehiculo) pedir información del vehículo
  6. Insertar información completa del vehículo
  7. Llamado de función
  8. Revisión de información si es existente previamente o no
  9. Agregar vehículo
  10. Devolver información actual del vehículo
  11. (en caso de decisión de inhabilitar vehículo) pedir información primordial del vehículo
  12. Insertar información primordial del vehículo
  13. Llamado de función
  14. Revisión de información si es existente previamente o no
  15. Conversión de dato a falso
  16. Devolver información actual del vehículo

## 1.4 Diagrama

(Con propósito de mejor comprensión para el lector)



## 2. Ingreso como cliente registrado

### 2.1 Diseño

```

elif opc == "2":
    print("Inhabilitar vehiculos")
    print("----Menu-----")
    print("[1] Vehiculo dañado ")
    print("[2] Vehiculo reservado")

elif opc == "2":
    print("Gestion de clientes")
    print("----Menu-----")
    print("[1] Ingresar como invitado")
    print("[2] Ingresar como cliente registrado")
    cli_opc = input("Seleccione una opción para la gestión de clientes: ")

    if cli_opc == "1":
        print("Cliente invitado")

    elif cli_opc == "2":
        print("Iniciar sesion")
        Nombre = input("Ingrese su nombre:")
        NumCedula = input("Ingrese su numero de cedula:")

        if NumCedula in Registro_Cedulas:
            print("Bienvenido de nuevo",Nombre)

        else:
            print("Cedula no encontrada, porfavor registrese")
  
```

```
Registro_Cedulas.append(NumCedula)
Nombre = input("Ingrese su nombre:")
Telefono = input("Ingrese su numero de telefono:")
print("Se registro correctamente ")
```

## 2.1.1 Códigos usados

### 2.1.1.1 Condición de Elif (se repite a lo largo del código )

elif opc == "2":	If/elif (condicion):
------------------	----------------------

#### 2.1.1.1.1 Forma de aplicación y base

El propósito general del código “elif (condicional):” es dar una opción diferente de ejecución dependiendo de la condición que cumpla en el caso de que el “if (condicional):” no se haya cumplido este código como ya se sabe previamente.

En este código se utilizó para otorgar acciones a ejecutar, en este caso específico de que el usuario que este ejecutando el código haya declarado un “2” como variable “opc”.

se aplica a lo largo con diferentes variables y diferentes tipos de datos

Esta forma de código se repite a lo largo también realizando “if-anidados” que luego vamos a explicar.

#### 2.1.1.1.2 Propósito

El propósito de usar este código, es poder ejecutar una serie de pasos específicos para poder darle al usuario mas opciones elegir en base a la información que declare en la variable, un ejemplo del como se aplica este código es en la división de caminos en la sección declarada “gestión de clientes”(además de esta misma ser una de las opciones vigentes desde el primer menú dependiendo de la acción que el usuario quería ver o realizar) donde se proporcionan 2 opciones , lo cual dos alternativas que puede seguir el código, en una opción llamada “cliente invitado”, en la cual el usuario después de declararse como cliente (escribiendo un “1” en las opciones) no realizara acción alguna (hasta el momento en este primer avance).

Si el usuario no se declara como cliente en el menú de “Gestión de clientes” y escribe un “2” se estará declarando como cliente registrado donde después se le solicitara ingresar información de “iniciar sesión”.

Como se menciono antes se aplican varios “if” (if-anidados), esto se ve en la “confirmación de inicio sesión” , en la cual después de ingresar información se “inicia sesión”, se ira a realizar la condicional que confirma que la variable “cedula” se encuentra afirmativamente en la lista de variables que si están ingresadas(esta variable se encuentra entre los datos principales de cada usuario que ya estuvo registrado y es la que define si la información ya estaba agregada o se necesita realizar un registro). De aquí se parten 2 caminos (realizado por otro if).

En forma resumida referente al código “if”, si el usuario (la variable correspondiente a la cedula) si se encuentra entre las variables de la lista de ya ingresados, se da un mensaje que le dice su estado en el programa (ya estaba registrado).

En caso de que no sea así y no se encuentre la variable se aplica la condicional emergente (“else”), donde se piden datos que luego serán agregados a la lista de usuarios ingresados.

### 2.1.1.2 Lista (se repite en varias ocasiones del código )

Registro_Cedulas = [ ]	Variable =[ 0, 1, 2, 3]
------------------------	-------------------------

#### 2.1.1.2.1 Forma de aplicación y base

Este código se utiliza para agrupar un grupo de variables tanto de diferentes tipos como del mismo en una variable extra que guardara todos esos datos formando en el conjunto una "lista".

Es bien sabido que son las mas usadas y manipuladas al momento de programar, en ella se puede encontrar un dato especifico, realizar un recorrido de los datos, de paso de estos sumarlos, mostrarlos, editarlos al mismo tiempos etc.

Estas variables se caracterizan también porque sus elementos se etiquetan por su posición iniciando en cero (0), continuando con los números consecutivos de uno en uno, sin limite especifico.

En esta sección del código, su uso fue de almacenamiento de los datos que fueron pedidos en la cuarta condicional con respecto a la sección de "gestión cliente", luego dependiendo si la información se encontraba en la variable lista o no se elegiría una acción predeterminada(ya explicada previamente).

#### 2.1.1.2.2 Propósito

Su propósito en el código (esta sección) es tanto: Limitar las acciones, definir si se tienen que registrar personas, verificar información que dependiendo de esta se tomaran diferentes caminos, recorrer información para tanto buscar como mostrar información del cliente después de su registro o al momento de iniciar sesión y lo que se considera mas importante y que definirá los pasos esenciales es el insertar información en la lista ubicada en "registro cliente".

### 2.1.1.3 Ingresar datos en variables (usado en todo el código)

Nombre = input("Ingrese su nombre:")	Variable = input(Informacion)
---	----------------------------------

#### 2.1.1.3.1 Forma de aplicación y base

La descripción de este código es sencilla , se basa en un código cuyo único propósito es almacenar información, esta se define automáticamente como tipo "texto", si se desea cambiar a tipo numero se necesitara de del código que funciona como convertidor "variable=int(variable)", automáticamente transforma las variables en números.

La información se puede manipular, re asignar y estas variables de almacenamiento se puede asignar el mismo valor a otras variables, de la misma forma almacenamiento para compartir o trasladar momentáneamente como uno de sus usos.

#### 2.1.1.3.2 Propósito

El propósito de este código, se extiende en: almacenar la información de los clientes, ser un medio a la hora de definir las tomas de acciones determinados por las condiciones en las que también se usa la variable de almacenamiento, en esta área de código, ser un medio de hallazgo de información entre listas.

## 2.2 Planeamiento

### 2.2.1 identificación del problema

en los problemas que se ven a lo largo del código están.

- Definir acción dependiendo de la elección del usuario
- Condiciones a aplicar
- Enlistar información
- Verificar información dada

### 2.2.2 acciones dependiendo del problema

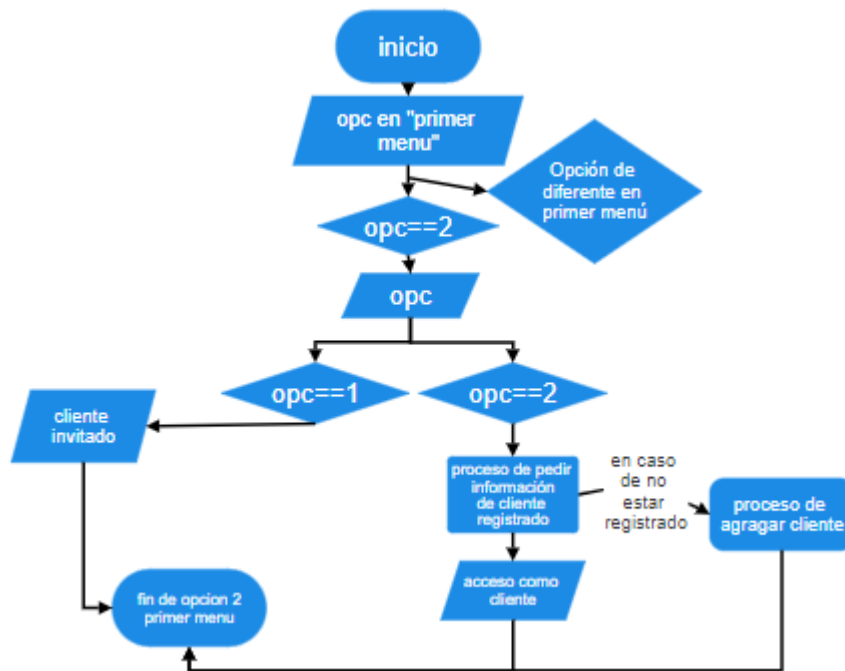
- Definir acción dependiendo de la elección del usuario
  - Aplicación de condicionales “if/if-elif/if-anidado”
  - También llamados como “menus”
- Condiciones a aplicar
  - Definido con “if” con información específica que evalúa la información de la variable
- Enlistar información
  - Aplicación de listas con conjunto de variables previamente ingresadas
- Verificar información dada
  - Arreglado con “if” con condiciones múltiples que evalúan la existencia de la información, en el caso de que ya existe previamente o no

## 2.3 Algoritmo

- Pasos a seguir dependiendo de las condiciones y objetivo principal
  1. Pedir opción en menú principal
  2. comparo la opción de menú principal
  3. pedir tipo de usuario
  4. analizar tipo de usuario
  5. definir acciones por tipo de usuario
  6. usuario es invitado
  7. usuario invitado dar mensaje
  8. usuario es registrado actual
  9. solicitar información del usuario registro
  10. buscar información del usuario en registros
  11. usuario es registrado anteriormente (se encontraba en registro)
  12. usuario se le da mensaje de bienvenida
  13. usuario no es registrado(no se encuentra en registro)
  14. solicitar información mas amplia
  15. agregar información al registro
  16. dar bienvenida al usuario nuevo

## 2.4 Diagrama

(Con propósito de mejor comprensión para el lector)



### 3 Visualizar vehículos

#### 3.1 Diseño

```

elif opc == "3":
    print("Visualizar vehículos")
    for i in inventario_vehiculos:
        print(i)
    print("Estos son los vehiculos")
  
```

##### 3.1.1 Código usados

###### 3.1.1.1 Recorido for

for i in inventario_vehiculo s: print(i)	for variableAlmacenamien to in rangoOconjunto de elementos accion
---	---

###### 3.1.1.1.1 Forma de aplicación y base

Este código se aplica con un ciclo for, el cual se le da la variable lista y una variable emergente que proporciona donde van a pasar las variables para luego ser impresas en pantalla.

Esta variable emergente mencionada cambiará su contenido consecutivamente, para cambiar su impresión, ocupando cada elemento de la lista hasta el final.

###### 3.1.1.1.2 Propósito

Su propósito es limitarse a mostrar el contenido de la lista correspondiente en orden y de forma continua hasta que se halla mostrado cada conjunto de información.

### 3.1.1.2 estructura if /if-elif/for

<code>print(i)</code>	<code>print(contenido)</code>
-----------------------	-------------------------------

#### 3.1.1.2.1 Forma de aplicación y base

Este código puede usarse de forma que pueda mostrar valores numéricos, datos, estados de datos, sus tipos y lo mas usado escribir mensajes en pantalla

En este código se aplica de forma que dentro del ciclo for el contenido a imprimir y que es cambiante se muestre al usuario hasta que termine el ciclo.

#### 3.1.1.2.2 Propósito

Su propósito fue de mostrar los elementos en la lista dentro del recorrido de datos hasta que ya no haya datos por mostrar dicho en sencillas palabras.

Nota: en el código se solicita imprimir la “variable emergente”

Ningún código realizado en este opción requiere de la autorización del usuario, se realizan automáticamente.

Y si no se encuentran datos dentro de la lista, el código simplemente no mostrara nada

## 3.2 Planeamiento

### 3.2.1 identificación del problema

- como recorrer los elementos
- como mostrar los elementos

### 3.2.2 acciones dependiendo del problema

- como recorrer los elementos
  - este problema se soluciono con un for combinado con la lista el cual se encargaría de pasar cada conjunto de elementos a la “variable emergente”

- como mostrar los elementos  
estos elementos serán mostrados dentro del bucle for proporcionados por la variable lista de “inventario\_vehiculos”.

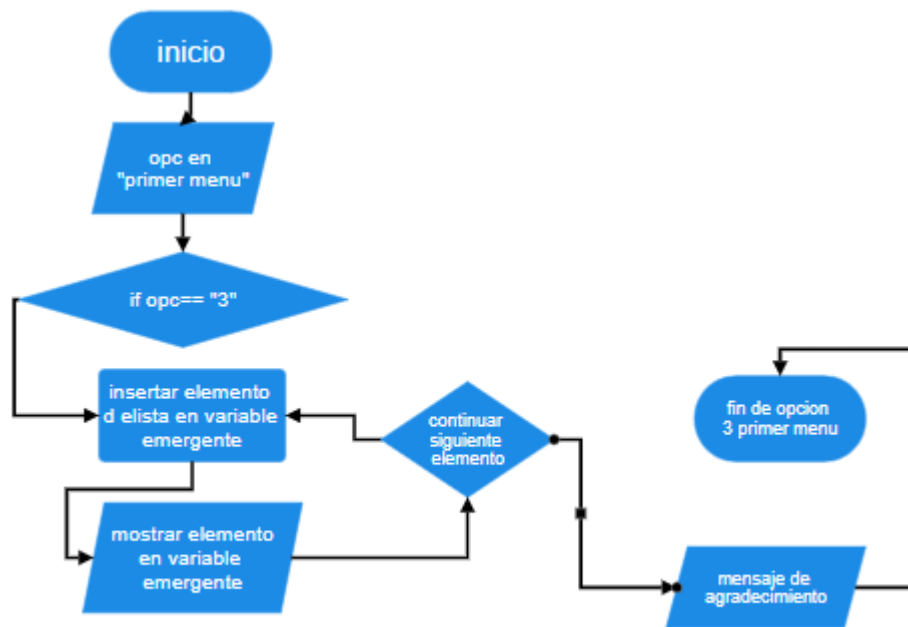
Explicando esto: una vez que el conjunto es pasado a la variable emergente, esta variable cambiaria cada vez que se recorra el ciclo hasta el final de la lista, por cada recorrido se usa un print que mostrara el contenido la variable emergente

## 3.3 Algoritmo

- Pasos a seguir dependiendo de las condiciones y objetivo principal
  1. Pedir opción en menú principal
  2. comparo la opción de menú principal
  3. entro a la indicada de opción 3
  4. inserto lista en ciclo con variable emergente
  5. insertar elemento en turno en la variable emergente
  6. muestro variable emergente
  7. repito acción 5 y 6 hasta ya no haver mas elementos
  8. mensaje y cierre del codigo

## 3.4 Diagrama

(Con propósito de mejor comprensión para el lector)



Avance 2. 22 de marzo del 2024

## Código

```

##avance 2 base
inventario_vehiculos = {}
Registro_Cedulas = []
reserva = []

sede_San_José=[["Registro de Tiempos"],["Registro De autos" ]
sede_Alajuela=[ ["Registro de Tiempos"],["Registro De autos"]]
sede_Guanacaste=[ ["Registro de Tiempos"],["Registro De autos"]]
sede_Limón=[["Registro de Tiempos"],["Registro De autos" ]
sede_Puntarenas=[ ["Registro de Tiempos"],["Registro De autos"]]
sede_Pérez_Zeledón=[["Registro de Tiempos"],["Registro De autos" ]
sedes_Lista=[sede_San_José,sede_Alajuela,sede_Guanacaste,sede_Limón,sede_Puntar
enas,sede_Pérez_Zeledón]

def Definir_Sede():
    import time
    print ("Horas Actuales \n",time.strftime("%H:%M:%S") )
  
```



```

print ("",time.strftime("%I:%M:%S") )
tiempo_actual=int(time.strftime("%H"))
print("-----")
print("---horario de sedes---")
print("1. San José: 24 horas, los 7 días de la semana.")
print("2. Alajuela: 24 horas, los 7 días de la semana.")
print("3. Guanacaste: Abren a las 4 am, cierran a las 11 pm.")
print("4. Limón: Abren a las 6 am, cierran a las 10 pm.")
print("5. Puntarenas: Abren a las 5 am, cierran a las 10 pm.")
print("6. Pérez Zeledón: Abren a las 7 am, cierran a las 10 pm.")
print("-----")
print("-----")

print("Porfavor seleccione la sede en la que se encuentra del 1 al 6: ")
print("1. San José")
print("2. Alajuela")
print("3. Guanacaste")
print("4. Limón")
print("5. Puntarenas")
print("6. Pérez Zeledón")
sede_destino=int(input())

if sede_destino==1:
    menu_administrador(sedes_Lista[0],tiempo_actual)

elif sede_destino==2 :
    menu_administrador(sedes_Lista[1],tiempo_actual)

elif sede_destino==3 :
    if tiempo_actual>3 and tiempo_actual<23:
        menu_administrador(sedes_Lista[2],tiempo_actual)
    else:
        print("sede de Guanacaste: Abren a las 4 am, cierran a las 11 pm.\n Ahora debe
estar cerrado, disculpas")

elif sede_destino==4 :
    if tiempo_actual>5 and tiempo_actual<22:
        menu_administrador(sedes_Lista[3],tiempo_actual)
    else:
        print("Sede de Limón: Abren a las 6 am, cierran a las 10 pm.\n Ahora debe estar
cerrado, disculpas")

elif sede_destino==5 :
    if tiempo_actual>4 and tiempo_actual<22:
        menu_administrador(sedes_Lista[4],tiempo_actual)
    else:
        print("Sede de Puntarenas: Abren a las 5 am, cierran a las 10 pm.\n Ahora debe
estar cerrado, disculpas")

elif sede_destino==6 :
    if tiempo_actual>6 and tiempo_actual<22:
        menu_administrador(sedes_Lista[5],tiempo_actual)
    else:
        print("Sede de Pérez Zeledón: Abren a las 7 am, cierran a las 10 pm.\n Ahora
debe estar cerrado, disculpas")

```

```

else:
    print("Sede inexistente,una disculpa")

def agregar_vehiculo(marca, año, modelo, cilindraje, precio_alquiler, precio_vehiculo,
placa, cantidad,Sede):
    if placa not in inventario_vehiculos:
        inventario_vehiculos[placa] = {'marca': marca,
                                        'año': año,
                                        'modelo': modelo,
                                        'cilindraje': cilindraje,
                                        'precio_alquiler': precio_alquiler,
                                        'precio_vehiculo': precio_vehiculo,
                                        'cantidad_disponible': cantidad,
                                        'habilitado': True}
        Sede[1].append(inventario_vehiculos[placa])

    print("Vehiculo agregado exitosamente")

def reservar_vehiculo(placa, cantidad):
    if placa in inventario_vehiculos:
        if inventario_vehiculos[placa]['habilitado']:
            if inventario_vehiculos[placa]['cantidad_disponible'] >= cantidad:
                inventario_vehiculos[placa]['cantidad_disponible'] -= cantidad
                print("Reserva realizada correctamente.")
            else:
                print("No hay suficientes vehiculos ")
        else:
            print("El vehiculo esta inhabilitado.")
    else:
        print("No existe ningun vehiculo con esa placa.")

def inhabilitar_vehiculo(placa):
    if placa in inventario_vehiculos:
        inventario_vehiculos[placa]['habilitado'] = False
        print("Vehiculo inhabilitado correctamente.")
    else:
        print("No se encontro ningun vehiculo con esa placa.")

def menu_administrador(Sede,tiempo_actual):
    import time
    Hora_entrada= time.strftime("%H:%M:%S")
    Sede[0].append(Hora_entrada)

    opc = ""
    while opc != "4":
        Hora_Salida_Calcu=int(time.strftime("%H"))
        print()
        print("Hora de entrada:",time.strftime("%I:%M:%S"))
        print("-----Menu-----")
        print("[1] Gestion inventario Vehiculos")
        print("[2] Gestion de clientes")
        print("[3] Visualizar vehiculos ")
        print("[4] Gestión de sedes")
        opc = input("Seleccione una opcion: ")

```

```

if opc == "1":
    print("----Gestion inventario vehiculos---")
    print("[1] Agregar vehiculos")
    print("[2] Inhabilitar vehiculos")
    opc_inventario = input("Seleccione una opcion: ")

    if opc_inventario == "1":
        print()
        print("Agrega el vehiculo que deseas")
        Marca = input("Ingrese la marca del vehiculo: ")
        Año = input("Ingrese el año del vehiculo: ")
        Modelo = input("Ingrese el modelo del vehiculo: ")
        cilindraje = input("Ingrese el cilindraje del modelo: ")
        Precio_alquiler = float(input("Ingrese el precio de al alquiler: "))
        Precio_auto = float(input("Ingrese el precio del auto: "))
        placa = input("Ingrese el numero de placa")
        cantidad = float(input("Ingrese la cantidad de vehiculos disponibles"))
        agregar_vehiculo(Marca, Año, Modelo, cilindraje, Precio_alquiler, Precio_auto,
placa, cantidad,Sede)
    elif opc_inventario == "2":
        placa = input("Ingrese la placa del vehiculo a inhabilitar: ")
        inhabilitar_vehiculo(placa)

elif opc == "2":
    print("Gestion de clientes")
    print("----Menu-----")
    print("[1] Ingresar como invitado")
    print("[2] Ingresar como cliente registrado")
    cli_opc = input("Seleccione una opcion para la gestion de clientes: ")

    if cli_opc == "1":
        print("Cliente invitado")
        print("Seleccione una opcion")
        print("[1] Ver un listado de marcas de vehiculo")
        print("[2] Ver un listado de reservas")
        reserva = input("Seleccione que desear ver")

    elif cli_opc == "2":
        print("Iniciar sesion")
        Nombre = input("Ingrese su nombre:")
        NumCedula = input("Ingrese su numero de cedula:")

        if NumCedula in Registro_Cedulas:
            print("Bienvenido de nuevo",Nombre)
            print("Seleccione una opcion")
            print("[1] Ver un listado de marcas de vehiculo")
            print("[2] Ver un listado de reservas")
            reserva = input("Seleccione la opcion que deseas ver")
            print("")
        else:
            print("Cedula no encontrada, porfavor registrese")
            Registro_Cedulas.append(NumCedula)
            Nombre = input("Ingrese su nombre:")

```

```

        Telefono = input("Ingrese su numero de telefono:")
        print("Se registro correctamente ")

    elif opc == "3":
        print("Visualizar vehículos")
        i= Sede[1]
        for e in i:
            print(e)
        print("Estos son los vehiculos")

    elif opc == "4":
        eleccion_Sede=int(input("Elija una opcion: \n [1]Cambiar de sede \n [2] Ver
resgistro de entrada y salida"))
        if eleccion_Sede==1:
            print("Cambiar de sede")
            print("Hora de salida",time.strftime("%I:%M:%S") )
            Hora_Salida=time.strftime("%H:%M:%S")
            Sede[0].append(Hora_Salida)
            break
        if eleccion_Sede==2:
            print(Sede[0])

    else:
        print("Opción inválida")

    if Sede==sedes_Lista[2] and (tiempo_actual+Hora_Salida_Calcu>23 or
tiempo_actual+Hora_Salida_Calcu==23):
        Hora_Salida=time.strftime("%H:%M:%S")
        Sede[0].append(Hora_Salida)
        break

    elif Sede==sedes_Lista[3] and (tiempo_actual+Hora_Salida_Calcu>22 or
tiempo_actual+Hora_Salida_Calcu==22):
        Hora_Salida=time.strftime("%H:%M:%S")
        Sede[0].append(Hora_Salida)
        break

    elif Sede==sedes_Lista[4] and (tiempo_actual+Hora_Salida_Calcu>22 or
tiempo_actual+Hora_Salida_Calcu==22):
        Hora_Salida=time.strftime("%H:%M:%S")
        Sede[0].append(Hora_Salida)
        break

    elif Sede==sedes_Lista[5] and (tiempo_actual+Hora_Salida_Calcu>22 or
tiempo_actual+Hora_Salida_Calcu==22):
        Hora_Salida=time.strftime("%H:%M:%S")
        Sede[0].append(Hora_Salida)
        break

while True:
    Definir_Sede()

```

## 4 Cambiar sede

### 4.1 Diseño

```
##avance 2 base
inventario_vehiculos = {}
Registro_Cedulas = []

sede_San_José=[["Registro de Tiempos"],["Registro De autos" ]
sede_Alajuela=[ ["Registro de Tiempos"],["Registro De autos"]]
sede_Guanacaste=[ ["Registro de Tiempos"],["Registro De autos"]]
sede_Limón=[["Registro de Tiempos"],["Registro De autos" ]
sede_Puntarenas=[ ["Registro de Tiempos"],["Registro De autos"]]
sede_Pérez_Zeledón=[["Registro de Tiempos"],["Registro De autos" ]
sedes_Lista=[sede_San_José,sede_Alajuela,sede_Guanacaste,sede_Limón,sede_Puntar
enas,sede_Pérez_Zeledón]

def Definir_Sede():
    import time
    print ("Horas Actuales \n",time.strftime("%H:%M:%S") )
    print ("",time.strftime("%I:%M:%S") )
    tiempo_actual=int(time.strftime("%H"))
    print("-----")
    print("---horario de sedes---")
    print("1. San José: 24 horas, los 7 días de la semana.")
    print("2. Alajuela: 24 horas, los 7 días de la semana.")
    print("3. Guanacaste: Abren a las 4 am, cierran a las 11 pm.")
    print("4. Limón: Abren a las 6 am, cierran a las 10 pm.")
    print("5. Puntarenas: Abren a las 5 am, cierran a las 10 pm.")
    print("6. Pérez Zeledón: Abren a las 7 am, cierran a las 10 pm.")
    print("-----")
    print("-----")

    print("Porfavor seleccione la sede en la que se encuentra del 1 al 6: ")
    print("1. San José")
    print("2. Alajuela")
    print("3. Guanacaste")
    print("4. Limón")
    print("5. Puntarenas")
    print("6. Pérez Zeledón")
    sede_destino=int(input())

    if sede_destino==1:
        menu_administrador(sedes_Lista[0],tiempo_actual)

    elif sede_destino==2 :
        menu_administrador(sedes_Lista[1],tiempo_actual)

    elif sede_destino==3 :
        if tiempo_actual>3 and tiempo_actual<23:
            menu_administrador(sedes_Lista[2],tiempo_actual)
        else:
```

```
print("sede de Guanacaste: Abren a las 4 am, cierran a las 11 pm.\n Ahora debe estar cerrado, disculpas")
```

```
elif sede_destino==4 :  
    if tiempo_actual>5 and tiempo_actual<22:  
        menu_administrador(sedes_Lista[3],tiempo_actual)  
    else:  
        print("Sede de Limón: Abren a las 6 am, cierran a las 10 pm.\n Ahora debe estar cerrado, disculpas")
```

```
elif sede_destino==5 :  
    if tiempo_actual>4 and tiempo_actual<22:  
        menu_administrador(sedes_Lista[4],tiempo_actual)  
    else:  
        print("Sede de Puntarenas: Abren a las 5 am, cierran a las 10 pm.\n Ahora debe estar cerrado, disculpas")
```

```
elif sede_destino==6 :  
    if tiempo_actual>6 and tiempo_actual<22:  
        menu_administrador(sedes_Lista[5],tiempo_actual)  
    else:  
        print("Sede de Pérez Zeledón: Abren a las 7 am, cierran a las 10 pm.\n Ahora debe estar cerrado, disculpas")  
else:  
    print("Sede inexistente,una disculpa")
```

```
def agregar_vehiculo(marca, año, modelo, cilindraje, precio_alquiler, precio_vehiculo, placa, cantidad,Sede):
```

```
    if placa not in inventario_vehiculos:  
        inventario_vehiculos[placa] = {'marca': marca,  
                                         'año': año,  
                                         'modelo': modelo,  
                                         'cilindraje': cilindraje,  
                                         'precio_alquiler': precio_alquiler,  
                                         'precio_vehiculo': precio_vehiculo,  
                                         'cantidad_disponible': cantidad,  
                                         'habilitado': True}
```

```
    Sede[1].append(inventario_vehiculos[placa])
```

```
    print("Vehiculo agregado exitosamente")
```

```
.....  
//código
```

```
def menu_administrador(Sede,tiempo_actual):  
    import time  
    Hora_entrada= time.strftime("%H:%M:%S")  
    Sede[0].append(Hora_entrada)
```

```
opc = ""  
while opc != "4":  
    Hora_Salida_Calcu=int(time.strftime("%H"))  
    print()
```

```

print("Hora de entrada:",time.strftime("%I:%M:%S"))
print("-----Menu-----")
print("[1] Gestion inventario Vehiculos")
print("[2] Gestion de clientes")
print("[3] Visualizar vehiculos ")
print("[4] Gestión de sedes")
opc = input("Seleccione una opcion: ")

```

```

if opc == "1":
    print("----Gestion inventario vehiculos---")
    print("[1] Agregar vehiculos")
    print("[2] Inhabilitar vehiculos")
    opc_inventario = input("Seleccione una opcion: ")

```

**EDITADO[**

```

    if opc_inventario == "1":
        print()
        print("Agrega el vehiculo que deseas")
        Marca = input("Ingrese la marca del vehiculo: ")
        Año = input("Ingrese el año del vehiculo: ")
        Modelo = input("Ingrese el modelo del vehiculo: ")
        cilindraje = input("Ingrese el cilindraje del modelo: ")
        Precio_alquiler = float(input("Ingrese el precio de al alquiler: "))
        Precio_auto = float(input("Ingrese el precio del auto: "))
        placa = input("Ingrese el numero de placa")
        cantidad = float(input("Ingrese la cantidad de vehiculos disponibles"))
        agregar_vehiculo(Marca, Año, Modelo, cilindraje, Precio_alquiler, Precio_auto,
placa, cantidad,Sede)
]

```

.....  
//código  
.....

```

elif opc == "2":
    print("Gestion de clientes")
    print("-----Menu-----")
    print("[1] Ingresar como invitado")
    print("[2] Ingresar como cliente registrado")
    cli_opc = input("Seleccione una opcion para la gestion de clientes: ")

```

.....  
//código  
.....

EDITADO[

```

    elif opc == "3":
        print("Visualizar vehículos")
        for i in Sede:
            print(i)
        print("Estos son los vehiculos")
]

```

```

elif opc == "4":

```

```

    eleccion_Sede=int(input("Elija una opcion: \n [1]Cambiar de sede \n [2] Ver
resgistro de entrada y salida"))
    if eleccion_Sede==1:
        print("Cambiar de sede")
        print("Hora de salida",time.strftime("%I:%M:%S") )
        Hora_Salida=time.strftime("%H:%M:%S")
        Sede[0].append(Hora_Salida)
        break
    if eleccion_Sede==2:
        print(Sede[0])

    else:
        print("Opción inválida")

    if Sede==sedes_Lista[2] and (tiempo_actual+Hora_Salida_Calcu>23 or
tiempo_actual+Hora_Salida_Calcu==23):
        Hora_Salida=time.strftime("%H:%M:%S")
        Sede[0].append(Hora_Salida)
        break

    elif Sede==sedes_Lista[3] and (tiempo_actual+Hora_Salida_Calcu>22 or
tiempo_actual+Hora_Salida_Calcu==22):
        Hora_Salida=time.strftime("%H:%M:%S")
        Sede[0].append(Hora_Salida)
        break

    elif Sede==sedes_Lista[4] and (tiempo_actual+Hora_Salida_Calcu>22 or
tiempo_actual+Hora_Salida_Calcu==22):
        Hora_Salida=time.strftime("%H:%M:%S")
        Sede[0].append(Hora_Salida)
        break

    elif Sede==sedes_Lista[5] and (tiempo_actual+Hora_Salida_Calcu>22 or
tiempo_actual+Hora_Salida_Calcu==22):
        Hora_Salida=time.strftime("%H:%M:%S")
        Sede[0].append(Hora_Salida)
        break

while True:
    Definir_Sede()

```



#### 4.1.1 Código usados

##### 4.1.1.1 Llamado de función combinado conciclo

<code>while True:</code> <code>    Definir_Sede()</code>	<code>While condición:</code>  <code>    acción a</code> <code>    ejecutar</code>
---	---

##### 4.1.1.1.1 Forma de aplicación y base

Esta código funciona en conjunto con una función ya definida que se estará llamando dentro de un ciclo el cual mientras se mantenga funcionando el ciclo no dejara de ejecutar su acción que en este caso es llamar la función que se explicara mas adelante.

##### 4.1.1.1.2 Propósito

El propósito de este código es llamar al nuevo primer paso que seria la definición de la sede, la acción se seguirá ejecutando al tener el nuevo ciclo principal corriendo, en este caso el “while” que se estará ejecutando mientras su condición la cual es como la palabra lo dice “True” , Verdad o verdadera.

Este código dará el primer paso a lo que va a definir las sedes y se ejecutara permanentemente hasta “romper el código”.

##### 4.1.1.2 Declaración de arreglos multidimensionales a usar

<code>sede_San_José=[["Registro de Tiempos"],["Registro De autos" ]</code> <code>sede_Alajuela=[ ["Registro de Tiempos"],["Registro De autos"]]</code> <code>sede_Guanacaste=[ ["Registro de Tiempos"],["Registro De autos"]]</code> <code>sede_Limón=[["Registro de Tiempos"],["Registro De autos" ]</code> <code>sede_Puntarenas=[ ["Registro de Tiempos"],["Registro De autos"]]</code> <code>sede_Pérez_Zeledón=[["Registro de Tiempos"],["Registro De autos" ] ]</code> <code>sedes_Lista=[sede_San_José,sede_Alajuela,sede_Guanacaste,sede</code> <code>_Limón, sede_Puntarenas,sede_Pérez_Zeledón]</code>	<code>//Arreglo multidimensional</code>  <code>Nombre_arreglo_multidimensional =</code> <code>[[Arreglo Interno A ],[ Arreglo Interno</code> <code>B]]</code>  <code>//Lista de arreglos almacenados en</code> <code>un solo arreglo</code>  <code>Lista_arreglos=[arregloA, ArregloB,</code> <code>ArregloC,ArregloD]</code>
---	---

##### 4.1.1.2.1 Forma de aplicación y base

###### 1- Arreglos multiDimencionales

Estos arreglos se encargan de por decirlo en palabras comunes, dos listas, dentro del arreglo en corchetes cuadrados se coloca el nombre de la variable o de una lista ya previamente declarada que será usada para otro almacenamiento interno.

###### 2- lista de arreglos

Esta lista es otro tipo de Arreglo , en este se declara directamente la información de forma textual, mas no como listas, se declaran como si se estuviera dando información común en textos. Se empieza declarando el nombre del arreglo-lista, acto seguido se van insertando el

nombre de las listas, separados por comas SIN paréntesis cuadrados (en cierta forma los corchetes cuadrados harán la diferencia entre un array y un elemento de una lista o elemento de variable)

#### 4.1.1.2.2 Propósito

Este código es de suma importancia en toda la extensión de esta nueva versión del código , ya que estos arreglos con nombre de sedes únicos, mantienen su propia lista de vehículos y lista de tiempos ambas listas por separados, razón de esto es que se pueda llevar una lista de registro monitoreado de ambos elementos para cada sede diferente. Este código esta siendo usado dentro de la función “Definir función”, siendo parte del primer paso que va a ser guardar las informaciones y datos de los vehículos almacenados en otra función de la cual se hablara después.

Resumiendo, esta área del código permitirá guardar la información en listas diferentes las cuales tendrán sus subsecciones propias.

#### 4.1.1.3 Definir función de sedes

def Definir_Sede()	def Nombre_Funcion(Elementos a utilizar)
--------------------	--

##### 4.1.1.3.1 Forma de aplicación y base

Esta se aplica usando el código “Def” seguidamente del nombre con el que se va a a querer bautizar la función, luego dos paréntesis que van a contraer los elementos con los que se trabajara (cabe recalcar esto es opcional),aunque una función no necesita necesariamente de elementos para actuar ya que puede hacer cualquier acción que un código normal.

##### 4.1.1.3.2 Propósito

El propósito de esta función es que el usuario pueda elegir una sede dentro de sus contenido ala ves incluye el primer paso para el monitoreo de tiempo (se hablara mas profundo de esto mas adelante) en su propósito esta crear una parte del programa que funciones correctamente en una subsección la cual se va a seguir ejecutando ya que como se dijo antes; esta función esta siendo ejecutada por un ciclo que mientras sea verdadero se seguirá ejecutando , en efecto este ciclo no se va a detener por ningún motivo hasta que el codigp se rompa lo que significa , llamara a la función de forma permanente.

#### 4.1.1.4 Menu de sedes de condicionales anidadas junto con llamado de funciones

```

    print("Porfavor seleccione la sede en la que se encuentra
del 1 al 6: ")

.....

//lista de opciones

sede_destino=int(input())

if sede_destino==1:

    menu_administrador(sedes_Lista[0],tiempo_actual)

elif sede_destino==2 :

    menu_administrador(sedes_Lista[1],tiempo_actual)

elif sede_destino==3 :

    if tiempo_actual>3 and tiempo_actual<23:

        menu_administrador(sedes_Lista[2],tiempo_actual)

    else:

        print("sede de Guanacaste: Abren a las 4 am, cierran a
las 11 pm.\n Ahora debe estar serrado, disculpas")

elif sede_destino==4 :

    if tiempo_actual>5 and tiempo_actual<22:

        menu_administrador(sedes_Lista[3],tiempo_actual)

    else:

        print("Sede de Limón: Abren a las 6 am, cierran a las
10 pm.\n Ahora debe estar serrado, disculpas")

elif sede_destino==5 :

    if tiempo_actual>4 and tiempo_actual<22:

        menu_administrador(sedes_Lista[4],tiempo_actual)

    else:

        print("Sede de Puntarenas: Abren a las 5 am, cierran a
las 10 pm.\n Ahora debe estar serrado, disculpas")

```

### **Respuesta por el usuario**

if acción a seguir según la respuesta:

Acción a ejecutar  
Llamado de función con asignación de elementos

elif acción a seguir según la respuesta:

If condición para poder proseguir:

Acción a ejecutar  
Llamado de función con asignación de elementos

Else

acción en caso de no cumplir con la condición

elif condición Principal:

If condición para poder proseguir:

Acción a ejecutar  
Llamado de función con asignación de elementos

Else:

acción en caso de no cumplir con la condición

Else:

Acción en caso de no coincidir con opciones

<pre> elif sede_destino==6 :      if tiempo_actual&gt;6 and tiempo_actual&lt;22:          menu_administrador(sedes_Lista[5],tiempo_actual)      else:          print("Sede de Pérez Zeledón: Abren a las 7 am, cierran a las 10 pm.\n Ahora debe estar serrado, disculpas")      else:          print("Sede inexistente,una disculpa") </pre>	
---	--

#### 4.1.1.4.1 Forma de aplicación y base

En este código vemos combinación de condicionales anidadas junto con funciones

##### 1- condicionales anidadas

En esta área usamos el uso del código if, elif y else para hacer una serie de condiciones a ejecutar en caso dependiendo de la “respuesta” por el usuario (lo que haya asignado en la variable),

Seguido de la condición inicial se continuar con otra condición dada por if-else. Esta segunda condición aplicada en varias opciones mas no en todas, pone el condición la hora local la cual si no coincide con la condición mandara el mensaje emergente y volverá a dar las “opciones” al usuario, esta variable para realizar la ejecución asignada al tiempo será explicada mas adelante.

##### 2- Llamado de funciones.

En esta parte del código se le llama a la función principal con la diferencia de cada usa de que se da uso a la “lista de sedes” para asignar una sede multidimensional y esta será diferente dependiendo de la condición en la que haya entrado, incluyendo que se le da la variable tiempo, influyente también en las condiciones de algunas opciones.

#### 4.1.1.4.2 Propósito

Explicar su propósito es relativamente sencillo, al tener la necesidad de crear un menú de sedes, se vio mas dinámico y sencillo el signar una sección de la lista multi dimensional de listas juntas, antes de colocar el nombre diferente de cada lista de sede, incluyendo que se da la lógica de la asignación de listas dependiendo de cada condición se da también la asignación de espacios dentro de estas listas para almacenamiento único de vehículos y horarios, de paso se da la variable de tiempo con el propósito de llevar el control de entrada y salida de cada sede diferente.

Las segundas condiciones (condición anidada) giran en torno a la hora, si el tiempo coincide con las horas permitidas, se dará acceso a la función y su asignación de otro modo se necesita entrar a otra sede para que el usuario empiece a realizar sus acciones deseadas sin restricción.(se hablara del código asignado a las horas mas adelante)

#### 4.1.1.5 Efecto universal de codigos en menú

**principal** (efecto del código tiempo en menú administrador se hablara después)

<p>1- asignación de lista correspondiente de sede como elemento a usar en función de “Menu administrador”</p> <pre>def menu_administrador(Sede,tiempo_actual):</pre> <p>2-efecto de sede en inventario-agregar a sede única</p> <pre>def agregar_vehiculo(marca, año, modelo, cilindraje, precio_alquiler, precio_vehiculo, placa, cantidad,Sede):     //código     inventario_vehiculos[placa] = { //código }</pre> <p>Sede[1].append(inventario_vehiculos[placa])</p> <p>//llamado de la funcion</p> <pre>agregar_vehiculo(//elementos// ,Sede)</pre> <p>3-aplicación de sedes en mostrar veiculos</p> <pre>elif opc == "3":     print("Visualizar vehículos")     for i in Sede:         print(i)     print("Estos son los vehiculos")</pre>	<p>1- asignación de lista correspondiente de sede como elemento a usar en función de “Menu administrador”</p> <p>Def nombre de funcion (elementos dados(<i>incluyendo elemento de lista</i>)):</p> <p>2-efecto de sede en inventario-agregar a sede única</p> <p>Def nombre de funcion (elementos dados(<i>incluyendo elemento de lista</i>)):</p> <p>//código</p> <p>Diccionario con identificador-agregar = {</p> <p>//código</p> <p>}</p> <p>Nombre_de_elemento_tipo_lista[lugar asignado “fila” en la lista donde se agregara ].</p> <p>append(elemento a agregar)</p> <p>//llamado de la funcion</p> <p>Llamado_de_la_funcion(//elementos// ,<i>elemento tipo lista</i>)</p> <p>3-aplicación de sedes en mostrar veiculos</p> <p>elif condición con elemento a evaluar :</p> <p>for variable_emergente in</p> <p>variable_tipo_lista:</p> <p>accion a ejecutar(recorrido de imprecision de elementos almacenados uno por uno en la variable emergente)</p>
---	---

##### 4.1.1.5.1 Forma de aplicación y base

El código sede Se recuerda es originalmente una variable tipo lista multidimensional de 2 filas, la variable anteriormente dada como un elemento de la “lista\_sedes” se bautiza nuevamente como variable Sede(aunque sigue siendo una variable Sede multidimensional), la variable a lo largo se implementa de forma diferentes tales como:

- 1- asignar sede al menú principal para su uso como elemento dado.
- 2- la variable sede se da a otras funciones para que pueda dar un efecto en ellas libremente desde el llamado de la función hasta su uso en la función externa.
- 3-se tiene que especificar que se pide visualizar la nueva sede especifica, aplicando el recorrido de for comúnmente , despues aplicando la variable de sede respectiva.

##### 4.1.1.5.2 Propósito

Esto se hace con el propósito antes mencionado de que cada sede tenga su propio inventario, al darse como elemento en la función principal se puede sar uso a lo largo del código, se da en la función de inventario para que en una función previamente definida para crear el nuevo elemento del inventario se guarde depues en la sede dada y escogida en un

principio, cabe destacar que al ser una lista multidimensional (siempre recordandop este detalle) se especifica en cual “fila” será guardado el inventario, esto se ve con el apartado de numero en la función entre corchetes cuadrados.

Otro uso se dio en el apartado de visualización de elementos los cuales, antes mostaraban el inventario en su, en esta nueva versión tienen que mostrar el contenido de el inventario en el avance respectivo.

4.1.1.6 códigos de tiempo y sus efectos a lo largo del código.



<p>1- En definición de sedes</p> <pre>def Definir_Sede():      import time      print ("Horas Actuales \n",time.strftime("%H:%M:%S") )      print ("",time.strftime("%I:%M:%S")) tiempo_actual=int(time.strftime("%H"))  2-condicion a evaluar en condicionales en definición de sedes      if sede_destino==1:  menu_administrador(sedes_Lista[0],tiempo_actual)      elif sede_destino==2 :  menu_administrador(sedes_Lista[1],tiempo_actual)      elif sede_destino==3 :         if tiempo_actual&gt;3 and tiempo_actual&lt;23:  menu_administrador(sedes_Lista[2],tiempo_actual)         else:             print("sede de Guanacaste: Abren a las 4 am, cierran a las 11 pm.\n Ahora debe estar serrado, disculpas")      elif sede_destino==4 :         if tiempo_actual&gt;5 and tiempo_actual&lt;22:  menu_administrador(sedes_Lista[3],tiempo_actual)         else:             print("Sede de Limón: Abren a las 6 am, cierran a las 10 pm.\n Ahora debe estar serrado, disculpas")      elif sede_destino==5 :         if tiempo_actual&gt;4 and tiempo_actual&lt;22:  menu_administrador(sedes_Lista[4],tiempo_actual)         else:             print("Sede de Puntarenas: Abren a las 5 am, cierran a las 10 pm.\n Ahora debe estar serrado, disculpas")      elif sede_destino==6 :</pre>	<p>1-En definición de sedes</p> <pre>nombre_de_la_funcion():      Importacion_de_Librería Expecifica  tiempo_actual=int(time.strftime("%H")) variable= extencion de librería  2-condicion a evaluar en condicionales en definición de sedes      if condición a evaluar:         llamado de función mas agregado de elementos (sede_especifica, variable de almacenamiento de extencion libreria)      elif condición a evaluar:         if condicion anidada con comparación de tiempo:             llamado de función mas agregado de elementos (sede_especifica, variable de almacenamiento de extencion libreria)         else:             acción emergente en caso de no cumplir con la condicion.         else:             acción emergente en caso de no cumplir primera condicion  3-iniciacion de tiempo y su monitoreo dentro del menú administrador def nombre_De_funcion(//elemento//,variable previa de tiempo):     importortacion de la librería     variable de monitoreoA=extencion de la libreria     nombre_genérico_para_la_sede_seleccionada [numero de fila].append(variable de monitoreoA)  4- guardar hora de salida de la sede  importortacion de la librería variable de monitoreob=extencion de la libreria          nombre_genérico_para_la_sede_seleccionada [numero de fila].append(variable de monitoreoB)  5-limitacion de tiempo fuera del ojo del usuario if condicion-comparacion de sede and comparación de hora en tiempo real:     Variable_de_almacenamiento de salida=extencion de librería especifica</pre>
---	--

<pre> if tiempo_actual&gt;6 and tiempo_actual&lt;22: menu_administrador(sedes_Lista[5],tiempo_actual) else:     print("Sede de Pérez Zeledón: Abren a las 7 am, cierran a las 10 pm.\n Ahora debe estar serrado, disculpas") else:     print("Sede inexistente,una disculpa")  3-iniciacion de tiempo y su monitoreo dentro del menú administrador def menu_administrador(Sede,tiempo_actual):     import time     Hora_entrada= time.strftime("%H:%M:%S")     Sede[0].append(Hora_entrada)  4- guardar hora de salida de la sede     print("Cambiar de sede")     print("Hora de salida",time.strftime("%I:%M:%S") )     Hora_Salida=time.strftime("%H:%M:%S")     Sede[0].append(Hora_Salida)  5-limitacion de tiempo fuera del ojo del usuario     if Sede==sedes_Lista[2] and (tiempo_actual+Hora_Salida_Calcu&gt;23 or tiempo_actual+Hora_Salida_Calcu==23):         Hora_Salida=time.strftime("%H:%M:%S")         Sede[0].append(Hora_Salida)         break      elif Sede==sedes_Lista[3] and (tiempo_actual+Hora_Salida_Calcu&gt;22 or tiempo_actual+Hora_Salida_Calcu==22):         Hora_Salida=time.strftime("%H:%M:%S")         Sede[0].append(Hora_Salida)         break      elif Sede==sedes_Lista[4] and (tiempo_actual+Hora_Salida_Calcu&gt;22 or tiempo_actual+Hora_Salida_Calcu==22):         Hora_Salida=time.strftime("%H:%M:%S")         Sede[0].append(Hora_Salida)         break      elif Sede==sedes_Lista[5] and (tiempo_actual+Hora_Salida_Calcu&gt;22 or tiempo_actual+Hora_Salida_Calcu==22):         Hora_Salida=time.strftime("%H:%M:%S")         Sede[0].append(Hora_Salida)         Break </pre>	<pre> nombre_genérico_para_la_sede_seleccionada [numero de fila].append(variable de monitoreoB) break=salir del ciclo y/o función actual </pre>
---	---

Se empieza llamando a la librería **“Time”**, esta librería posee una gran cantidad de extensiones referentes al tiempo, desde años hasta milisegundos, estas extensiones o su información a la hora de ejecutarlas puede llegar a almacenarse en variables o ser impresas en pantalla, aclarando que solo llega a funcionar con el uso del código **“import time”**, para dar uso de una extensión se llevaría a cabo el llamado de la función seguido del nombre de la extensión y sabiendo si se quiere almacenar como texto o almacenarlos dentro de arreglos.

#### 4.1.1.6.2 Propósito

Los propósitos de estos códigos son los siguientes:

1-Guardar entrada y salida Monitoreo desde definición de sede.

Una vez que la librería de tiempo es importada se crea una variable que monitorea el tiempo actual de forma automática, esta variable lectora del tiempo, será aplicado en las condicionales que darían paso al llamado de la función y su asignación de elementos. Las condiciones usan una comparación de horas exactas junto con mensajes emergentes indicando las horas permitidas para usar.

2-condicion de tiempo para entrada a una sede y para salir de la sede, variables a guardar y ejecutar en menú principal.

Usando las variables guardadas, estas variables serán usadas en las condicionales, además de almacenadas en la lista de sedes espacio multidimensional específico usado por llevar el monitoreo.

Se utilizan para llevar a cabo la acción de “salida automática dependiendo de la sede”, esto es porque si se excede el tiempo límite definido se saldrá del código de forma automática dependiendo de las condiciones.

1- En definición de sedes

Su propósito es obtener la hora local exacta la cual después se le brindará al usuario con el propósito de que sepa las sedes disponibles según las horas del tiempo actual nótese que se configura la impresión en horas de 24 horas y de 12 horas para más información.

Además la hora actual 24 Horas se guarda en tiempo real en una variable que después será utilizada en varias ocasiones.

2-condicion a evaluar en condicionales en definición de sedes

Como ya se mencionó antes en varias condiciones se utiliza la variable de hora actual previamente guardada para muchas algunas condiciones anidadas, esto con el propósito de crear las condiciones de horas disponibles, limitando el acceso a muchas sedes, dependiendo de la hora y su coincidencia de 24 horas.

3-iniciación de tiempo y su monitoreo dentro del menú administrador

def menu\_administrador(Sede,tiempo\_actual):

import time

Hora\_entrada= time.strftime("%H:%M:%S")

Sede[0].append(Hora\_entrada)

En el momento de entrar al menú administrador, dentro de los elementos dados, se da la sede elegida y disponible, junto con la hora actual en la que se encontraba al momento de ingresar, seguido se vuelve a importar la librería **Time**, para de esta forma poder guardar la variable de tiempo actual u “hora de entrada” dentro de una variable que después será

agregada a una de las dos listas dimensionales de la sede correspondiente e iniciar de esta forma su historial de entrada y salida único de una sede.

4- guardar hora de salida de la sede (de forma voluntaria)

```
print("Cambiar de sede")
```

```
print("Hora de salida",time.strftime("%I:%M:%S") )
```

```
Hora_Salida=time.strftime("%H:%M:%S")
```

```
Sede[0].append(Hora_Salida)
```

Este cambio o salida de una sede es voluntario del usuario(se explicara de esto mas adelante), esta realizado como una opción que puede realizar el usuario, en caso de ser elegida, ocurre que el programa antes de salir de la función , guarda ese lapso de tiempo actual en una variable que marcara la salida exacta, esta variable será almacenada en una de las listas dimensionales de la sede, marcando a si la hora en la que se dio el abandono, en la siguiente línea de código se dará el código predefinido de “romper” lo cual lo devolverá al código de definición de sede, finalizando en ese lapso de tiempo el poder guardar la hora en la que se entro en la función correspondiente del menú principal y hora de salida por elección de cambio de sede.

5-limitacion de tiempo fuera del ojo del usuario y guardado de hora de salida

Este Código esta echo con el propósito de ser un tipo de “limitante de tiempo” dependiendo de la sede, en caso de que alguna sede cumpla con la condicional a la ves que el horario se salga de los parámetros acordados, el código sacara al usuario en contra de su voluntad por respeto al código pasado que limita la hora, si no sale la hora de los para metros el Código podrá seguirse ejecutando libremente. Este código depende de la sede elegida ya que hay algunas sedes que no se les asigna tiempo limite.

Como extra del Código, se guarda la hora de salida en una variable que después será agregada a la lista dimensional de la sede respectiva antes de que el Código “rompa” la ejecución.

*En resumen de propósito de aplicación:*

*Se utiliza para monitoreo de entrada y salida.*

*Se utiliza para aplicar en las condiciones que sacar al usuario de la sede respectiva y en el principio de la primera función a llamar:*

*Siempre se va a guardar tanto el valor de la hora de “entrada” junto con el de “salida”*

*Indicar al usuario su hora actual= ayuda en que el usuario este al tanto del ambiente en el código.*

## 4.2 Planeamiento

### 4.2.1 identificación del problema

- como limitar acceso de sedes por su tiempo
- Como almacenar dos tipos diferentes de datos separados en cada sede además que cada sede tenga datos únicos.

- almacenamiento único de vehículos
- creación de registro de entrada y salida por hora

#### 4.2.2 acciones dependiendo del problema

- como limitar acceso de sedes por su tiempo
  - se le dio solución a este problema con la importación de la librería “Time” y usando el comando específico de “time.strftime(“%H”)” este Código predefinido con solo el elemento “H” en sus paréntesis, se devuelve una hora de 24 horas exacta a la actual (no dará los minutos o segundos a menos que se le indique).  
Acto seguido usando la variable donde se guardo, es variable se aplicara en las condicionales de entre los tiempos permitidos para el acceso a una sede.
- Como almacenar dos tipos diferentes de datos separados en cada sede además que cada sede tenga datos únicos.
  - Para esta problemática se decidió por dar uso a las variables dimensionales a la vez almacenarlos en una serie de lista que recopile esas listas dimensionales separadas, con esto se refiere a que; se empieza creando arreglos separados uno del otro, dichos arreglos serian las sedes, dentro de estos arreglos cada uno ya se crea inicializado con dos listas internas que separan los dos tipos de datos (tiempo y lista de vehículo), después se crea una “lista principal” la cual en esta se almacenan todas los arreglos(las sedes), el objetivo de esto es que a la hora de llamar a la función del menú principal y su asignación de elementos se pueda dinamizar asignando solo su campo en la “lista principal” asignado a la sede correspondiente elegida, la cual si recordamos aunque sea un elementos de una lista sigue siendo un arreglo dimensional que se puede ejecutar como uno sin problemas.
- almacenamiento único de vehículos
  - recordemos que al momento de llamado del menú principal dependiendo de la sede escogida se asigno una sede independiente, pero se le rebautizo con un nombre estándar (cambiando solo su nombre de forma externa pero aun conservando sus propiedades antes definidas) esto ayudo en que se pudiera usar el elemento en la función de agregado de vehículos, lo cual ya es una función aparte.  
Se le da el elemento de sede a la función, llevándola al Código externo, donde deja que se cree el vehículo, después todos esos elementos al vehículo correspondiente se guardan en la fila específica para vehículos en la sede la cual siempre se mantiene como una variable de almacenamiento bidimensional usando el Código  
“Sede[1].append(inventario\_vehiculos[placa])”
- creación de registro de entrada y salida por hora.
  - Para la solución de este problema se dio uso nuevamente de la librería “time” esto con el propósito de poder al principio del código guardar las horas de entrada , también se aplicaron las condicionales aparte en las cuales se encargaban de limitar el tiempo dependiendo de las horas y de las sedes.  
Al momento de querer guardar esos tiempo en las sedes como registros exactos se tomo un proceso similar a los vehículos, el cual fue tomar la

variable donde se guardo el tiempo exacto de entrada o de salida, después guardarla al momento de entrar al código en la fila correspondiente a tiempos y registros de entrada y salida, entrada como "Sede[0].append(Hora\_entrada)" y en cualquier acción que involucre la salida de la sede como "Sede[0].append(Hora\_Salida)".

### 4.3 Algoritmo

- Pasos a seguir dependiendo de las condiciones y objetivo principal
  1. escoger sede
    1. mostrar hora de 24 horas y hora de 12 horas como información para el usuario.
    2. Mostrar las sedes disponibles y sus horarios para comprensión del usuario.
    3. el usuario da su selección de sede
    4. caso de opción no disponible por horario, mandar mensaje de horario correspondiente a la sede como información para el usuario
    5. caso de que opción no este disponible por no estar definida antes se manda mensaje de disculpa y aclaración del problema de una sede inexistente.
    6. caso de que sede este disponible y existente dar acceso al menú principal y dándole datos para extender el funcionamiento de las sedes y sus hora
  2. empieza a medir tiempo como hora entrada a la sede
  3. se guarda entrada a de sede en su campo especifico asignado
  4. se mide el tiempo de la sede.
    1. en caso de que el tiempo no haya superado los limites continuar haciendo acciones libremente
    2. se supera el tiempo limite caso
      1. se guarda tiempo en hora de salida
      2. se guarda hora de salida en lista espacio especifico.
      3. Programa se cierra y se regresa al menú principal
    3. se escoge sede diferente.
    4. se guarda tiempo de salida
    5. tiempo de salida se guarda en espacio definido para tiempo en sede correspondiente,
  5. guardar elementos en sede escogida (esto apartado del control del usuario)
    1. usuario crea vehículo por elección.
    2. usuario llena los datos pedidos
    3. de forma fuera del control del usuario, este elemento vehiculo se almacena dentro la sede en una sección especifica para eso.

### 4.4 Diagrama

Estos códigos son externos a las ejecuciones que puede hacer el usuario, por ende no se presenta diagrama, los códigos trabajan de forma externa como almacenamientos y no impiden ninguna acción.

Por ende no cuentan con diagrama.

## 5 Ingreso como invitado y creación de gestión reservas

### 5.1 Diseño

```
if cli_opc == "1":
    print("Cliente invitado")
    print("Seleccione una opcion")
    print("[1] Ver un listado de marcas de vehiculo")
    print("[2] Ver un listado de reservas")
    reserva = input("Seleccione que desear ver")

elif cli_opc == "2":
    print("Iniciar sesion")
    Nombre = input("Ingrese su nombre:")
    NumCedula = input("Ingrese su numero de cedula:")

    if NumCedula in Registro_Cedulas:
        print("Bienvenido de nuevo",Nombre)
        print("Seleccione una opcion")
        print("[1] Ver un listado de marcas de vehiculo")
        print("[2] Ver un listado de reservas")
        reserva = input("Seleccione la opcion que desear ver")
        print("")
```

#### **Gestion de reservas aplicaciones como opciones extra(creación activa en futuro avance III)**

```
print("Cliente invitado")
print("Seleccione una opcion")
print("[1] Ver un listado de marcas de vehiculo")
print("[2] Ver un listado de reservas")
reserva = input("Seleccione que desear ver")

elif cli_opc == "2":
    print("Iniciar sesion")
    Nombre = input("Ingrese su nombre:")
    NumCedula = input("Ingrese su numero de cedula:")

    if NumCedula in Registro_Cedulas:
        print("Bienvenido de nuevo",Nombre)
        print("Seleccione una opcion")
        print("[1] Ver un listado de marcas de vehiculo")
        print("[2] Ver un listado de reservas")
        reserva = input("Seleccione la opcion que desear ver")
        print("")
```

#### 5.1.1 Código usados

##### 5.1.1.1

<pre>elif opc == "2":     print("Gestion de clientes")     print("----Menu----")     print("[1] Ingresar como invitado")     print("[2] Ingresar como cliente registrado")     cli_opc = input("Seleccione una opcion para la gestion de clientes: ")</pre>	<p>Elif condicion:</p> <p>Acción a ejecutar</p> <p>If condicional:</p> <p>Acción a ejecutar+ llamado de variable de almacenamiento</p>
---	--

<pre> if cli_opc == "1":     print("Cliente invitado")     print("Inventario de vehiculos:", inventario_vehiculos) </pre>	
---	--

#### 5.1.1.1.1 Forma de aplicación y base

Se realiza una serie de condicionales proporcionados por los codigos if-elif (if principal solo visible en la primera condición a evaluar, esta condición vista en el Código se puede identificar como la segunda), después de afirmar el cumplimiento de la primera seccion de condicionales se pasa a una segunda lista de condicionales o “ifs anidados” en las cuales se presenta el mismo ciclo de realizar la confirmación de condicional y luego ejecutar una acción. La acción que analizaremos es la acción destinada a la opción de “invitados”, al realizar la condicional directamente se muestra una impresión, seguido de la impresión de una variable y su contenido.

#### 5.1.1.1.2 Propósito

Se le quiere dar al usuario una serie de opciones en el menú principal para ejecutar como las antes mencionadas agregar vehículos, cambio de sedes y otros en este caso analizamos la segunda condicional o “segunda opción” referente a los clientes, en caso de que se aya seleccionado a opción 2 , el cliente tiene que definirse como invitado o registrado, en este caso veremos como invitado únicamente, su único propósito o opción dada es el visualizar los elementos en la lista del inventario de vehículos. Esas acciones es lo único permitido para un invitado y es lo que se realiza automáticamente.

### 5.2 Planeamiento

#### 5.2.1 identificación del problema

- diferenciar invitado del cliente registrado
- mostrar solo lo disponible a los invitados

#### 5.2.2 acciones dependiendo del problema

- diferenciar invitado del cliente registrado
  - la diferencia se da con una serie de “ifs” los cuales depues de el usuario dar su “respuesta de selección”, la respuesta se va a los condicionales para luego crear la acción correspondiente.
- mostrar solo lo disponible a los invitados
  - esto se realiza con la implementación por “,” seguido de la variable para poder así mostrar el contenido de la variable, esta acción de observar es lo único que se le permite a un invitado y se realiza automáticamente.

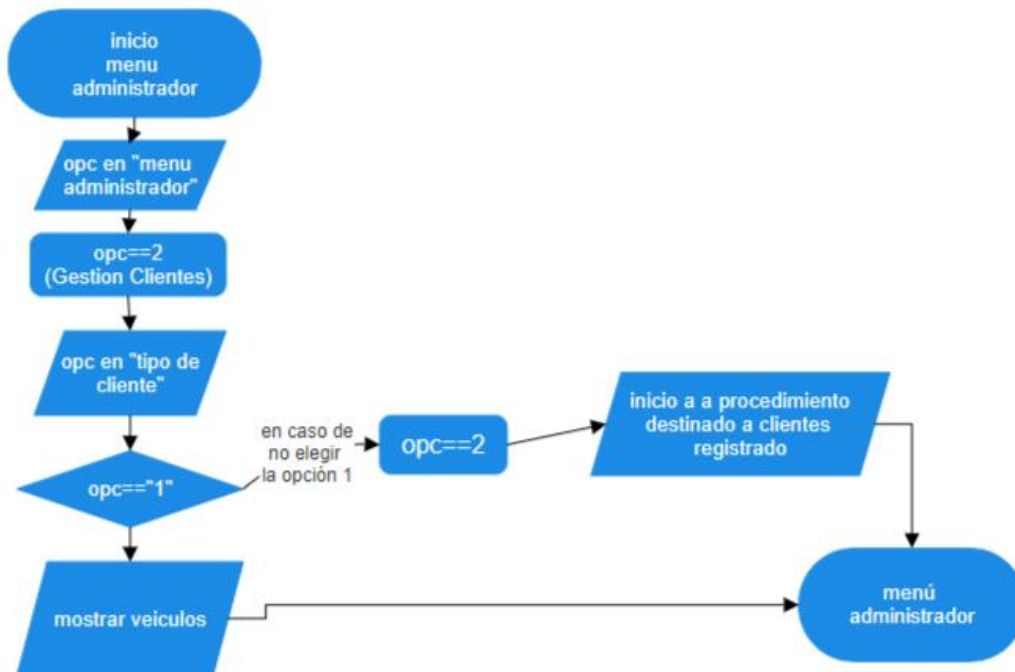
### 5.3 Algoritmo

- Pasos a seguir dependiendo de las condiciones y objetivo principal
- DESPUES DE DIJITAR LA OPCION 2 DEL MENU “DESTINADA A LA GESTION DE CLIENTES”
  1. Dar opciones de tipo de clientes
  2. pedir al usuario digitar el numero a tipo de cliente que es.
  3. en caso de cliente invitado, Mostrar la lista de inventario de vehículos.

### 5.4 Diagrama

(Con propósito de mejor comprensión para el lector)





## 6 Visualizar vehículos

### 6.1 Diseño

```

print("Visualizar vehículos")
i= Sede[1]
for e in i:
    print(e)
print("Estos son los vehiculos")

```

#### 6.1.1 Código usados

##### 6.1.1.1 Horadación de elementos de lista dimensional a variable

i= Sede[1]	Variable extra= campo específico en lista dimecional
------------	--

##### 6.1.1.1.1 Forma de aplicación y base

Esta variable se aplica de una forma sencilla, se crea una variable la cual se le va a heredar el campo específico del array bidimensional, heredando de esta forma los elementos de los campo específicos.

##### 6.1.1.1.2 Propósito

Esto se dio con el propósito de solo mostrar los elementos derivados a vehículos en esa sede específica, recordando que cada sede tiene vehículos únicos.

##### 6.1.1.2 Impresión de vehículos

for e in i: print(e)	For variable emergente in array
-------------------------	---------------------------------

	Accion- imprecision de elementos heredados-pasan por variable emergente
--	---

#### 6.1.1.2.1 Forma de aplicación y base

Esto se da aplicando el ciclo for donde se le aplica una variable emergente por donde van a pasar los elementos del array , esto se repetirá hasta que ya no existan elementos que pasen por la variable emergente

#### 6.1.1.2.2 Propósito

El propósito de este código fue para crear el recorrido de la lista con el contenido únicamente de los vehículos de aquella sede específica, mostrando los vehículos y sus detalles

### 6.2 Planeamiento

#### 6.2.1 identificación del problema

- como recorrer los elementos únicamente de vehículos
- como mostrar los elementos únicamente de vehículos

#### 6.2.2 acciones dependiendo del problema

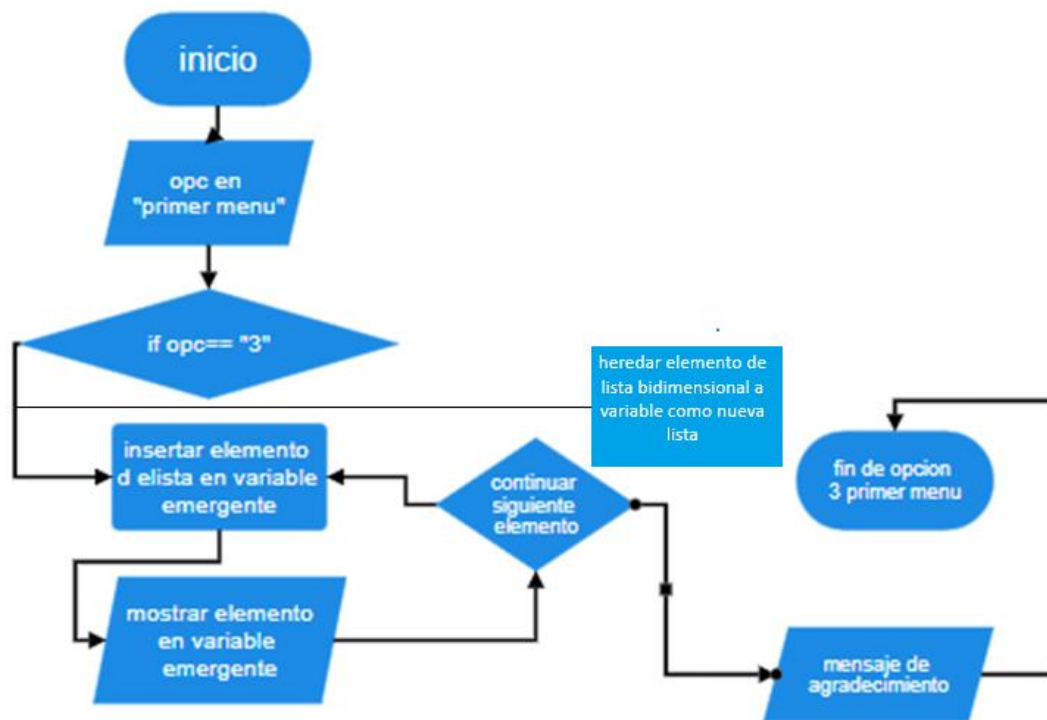
- como recorrer los elementos únicamente de vehículos
  - se dio solución a ese problema como paso inicial heredando la lista de la sede que contiene únicamente vehículos especificando su campo, esa variable nueva con la lista dimensional del array bidimensional se ubico en un bucle for que recorrería cada elemento de la nueva variable heredada con lista.
- como mostrar los elementos únicamente de vehículos
  - esto se le dio solución con un print que mostraría únicamente los elementos que pasen por la variable emergente del for, lo cuales recordemos son todos los elementos de la lista de vehículos unca de la sede correspondiente.

### 6.3 Algoritmo

1. Pedir opción en menú principal
2. comparo la opción de menú principal
3. entro a la indicada de opción 3
4. declarar conjunto de lista bidimensional en nueva variable, dándola como nueva lista
5. inserto nueva lista en ciclo con variable emergente
6. insertar elemento en turno en la variable emergente
7. muestro variable emergente
8. repito acción 6 y 7 hasta ya no haber mas elementos
9. mensaje y cierre del Código

### 6.4 Diagrama

(Con propósito de mejor comprensión para el lector)



## Cronograma

Avance	Requerimiento	Entrega
Avance I	Gestión inventario vehículos -Jordan Ivan Soto Morales	8 de marzo del 2024
	Ingreso como cliente registrado  - Camila Argeñal Coronado	

	<p>Mostrar vehículo</p> <p>-Eduardo miranda mora -Josias Aguilar Arroyo -Samantha Perez Rojas</p>	
	<p>Creación de repositorio ubicado en git hub</p> <p>-Jordan Ivan Soto Morales -Samantha Pérez rojas</p>	
Avance II	<p>Cambiar sede</p> <p>-Samantha Pérez rojas</p>	3 de abril del 2024
	<p>Ingreso como invitado</p> <p>- Camila Argeñal Coronado</p>	
	<p>Visualizar vehículos</p> <p>-Samantha perez rojas -Eduardo miranda mora</p>	
	<p>Creación de reserva</p> <p>-Josias Aguilar Arroyo</p>	
Avance III		17 de abril del 2024