Descripción solución

```
#IMPORTS
import datetime
import re #sino no podre usar expresiones regulares que harán falta en un
futuro


#VARIABLES
cliente = []
lista_Clientes = []
diccionario_Clientes = {}
#cliente["nombre","apellidos","direccion","NIF","telefono","correo","habi
tual","fecha"]
numCliente = 0

#FUNCIONES

def introducirNombre():

    nombre = (input('Nombre del cliente:').strip()) #strip para eliminar
algun espacio que se haya podido poner por error
    nombre = nombre[0].upper() + nombre[1:] # la primera letra de un
nombre siempre va en mayúsculas
    print()#linia en blanco
    return nombre

def introducirApellidos():

    apellidos = (input('Apellidos del cliente:').strip())
    apellidos = apellidos[0].upper() + apellidos[1:] # idem que el nombre
    print()#linia en blanco
    return apellidos

def introduccirDirecion():

    direccion = (input('Dirección del cliente:'))
    print()#linia en blanco
    return direccion

def validarNIF():

    correcto = False
    while not (correcto):
        nif = (input('NIF del cliente:'))

        if nif[:8].isdigit() and len(nif) == 9 and nif[-1].isalpha():
#compruebo que sean digitos o letras
            print("NIF correcto.")
```

```
            return nif.upper() #la letra del DNI siempre va en mayuscula
        else:
            print("NIF incorrecto, debe volver a intentarlo.")
    print()#linia en blanco


def validarNumTelf():

    correcto = False
    while not (correcto):
        telefono = (input('Teléfono del cliente: ').strip())

        if telefono.isdigit() and len(telefono) == 9:
            print("Número de teléfono correcto.")
            telefono = int(telefono)
            return telefono
        else:
            print("Número de teléfono incorrecto.")
    print()#linia en blanco

    return telefono

def validarCorreo():

    correcto = False
    while not (correcto):
        correo = (input('Correo del cliente:'))

        patron = r'^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$' #r =
raw string(cadena sin escape, no es necesario \\), ^ = inicio de texto,
lugo todo lo que puede entrar antes del @ y luego lo que entra despues, .
separa dominio de la extension y el final es que minimo 2 letras $ = fin
del texto

        if re.fullmatch(patron, correo):
            print("Correo con el formato correcto.")
            return correo
        else:
            print("Formato del correo incorrecto, intentalo de nuevo.")
            print("Un ejemplo como debería ser: usuario@dominio.ext")
    print()#linia en blanco

    return correo

def clienteHabitual():

    correcto = False
    while not (correcto):
```

```
        respuesta = input(('¿Es un cliente habitual?(si/no):
')).strip().lower() #strip para eliminar algun espacio que se haya podido
poner por error, y slow para que sea más fácil la comprobación
        if respuesta == 'si':
            return  True

        elif respuesta == 'no':
            return  False
        else:
            print("Responde con 'si' o 'no' porfavor.")
    print()#linia en blanco

def newCliente():

    print('Desde opción 1')
    nombre = introducirNombre()
    apellidos = introducirApellidos()
    direccion = introduccirDirecion()
    nif = validarNIF()
    if nif in diccionario_Clientes:
        print("Cliente ya registrado anteriormente.")
        return  # salgo de la función

    telefono = validarNumTelf()
    correo = validarCorreo()
    habitual = clienteHabitual()
    fecha = datetime.date.today().strftime('%d/%m/%Y') #si hacemos la
fecha automaticamente no hace falta comprobar nada porque además añadimos
el formato que queremos


    #creamos el diccionario del cliente ya con todos los datos obtenidos
    cliente_info = {
        "nombre": nombre,
        "apellidos": apellidos,
        "direccion": direccion,
        "NIF": nif,
        "telefono": telefono,
        "correo": correo,
        "habitual": habitual,
        "fecha": fecha
    }


    #lo guardamos en la lista y en el diccionario

    lista_Clientes.append(cliente_info) #lista clientes
    diccionario_Clientes[nif] = cliente_info #el nif es la clave del
diccionario
```

```
    global numCliente
    numCliente +=1 #sumamos en el contador un cliente

    print(f"\nCliente {nombre} {apellidos} añadido correctamente")
    print("\nNúmero de clientes: ", numCliente)
    print()#linia en blanco



def deleteCliente():
    print('Desde opción 2')

    nifEliminar = validarNIF()


    if nifEliminar in diccionario_Clientes:
        print(f"Cliente con NIF {nifEliminar} eliminado
correctamente.\n")
        #elimino del diccionario
        diccionario_Clientes.pop(nifEliminar)
        #elimino de la lista
        #otra opcion pero no tan correcta porque lista_Clientes lo
seguiremos usando:
        #for i, c in enumerate(lista_Clientes):
        #if c["NIF"] == nifEliminar:
        #del lista_Clientes[i]
        lista_Clientes[:] = [c for c in lista_Clientes if c["NIF"] !=
nifEliminar] #: es toda la lista, y luego recorro para crear una nueva
lista sin añadir el que queremos eliminar
        global numCliente
        numCliente -=1 # eliminamos del contador un cliente
        print("\nNúmero de clientes: ", numCliente)


    else:
        print("El NIF no coincide con ningún cliente")
    print()#linia en blanco


def printCliente():
    print('Desde opción 3')

    nifMostrar = validarNIF()  # pedimos NIF

    if nifMostrar in diccionario_Clientes:
        print("\nNúmero de clientes: ", numCliente)
        cliente = diccionario_Clientes[nifMostrar]
        for k,v in cliente.items():
            print(f"{k}: {v}")
```

```
    else:
        print("No existe ningún cliente con ese NIF.")
    print()#linia en blanco


def printAllClientes():
    print('Desde opción 4')

    #puede que hayamos borrado todos los clientes o que no hayamos creado
ninguno
    if not diccionario_Clientes:
        print("No hay clientes para mostrar.\n")
        return
    #imprimiendo la lista solo el nombre y el nif de los clientes
    print("\nNúmero de clientes: ", numCliente)
    print("\nListado de todos los clientes:\n")
    for nif, cliente in diccionario_Clientes.items():
        print(f"NIF: {nif} → {cliente['nombre']} {cliente['apellidos']}")
    print()  # línea en blanco




def printGoodClientes():
    print('Desde opción 5')

    print("Número de clientes: ", numCliente)
    #imprimiendo la lista solo el nombre y si es habitual
    for cliente in lista_Clientes:
        if cliente["habitual"]:
            print(cliente["nombre"], cliente["apellidos"], "con NIF: ",
cliente["NIF"], "es cliente habitual")
    if not lista_Clientes:
        print("No hay clientes para mostrar.\n")
    print()#linia en blanco

def salir_Menu():
    print('Saliendo del menú')
    return True

def menu():
    salir = False
    while not (salir):
        print('______ MENU ______')
        print()
        print("1. Añadir Cliente \n"
              "2. Eliminar Cliente \n"
```

```
                "3. Mostrar Cliente \n"
                "4. Imprimir todos los clientes \n"
                "5. Imprimir los clientes más habituales \n"
                "6. Salir \n     ")

        op = (input('Seleccione una opción: '))

        if op == "1":
            newCliente()
        elif op == "2":
            deleteCliente()
        elif op == "3":
            printCliente()
        elif op == "4":
            printAllClientes()
        elif op == "5":
            printGoodClientes()
        elif op == "6":
            salir = salir_Menu()
        else:
            print()
            print('Debe seleccionar una opción de 1 a 6')
            print()

#PROGRAMA

menu()
```

El código se implementó de esta manera: primero ponemos los import, segundo cree las variables, tercero las funciones y por último el programa.

Hay algunas variables que se declaran “global” para que puedan usarse en todo el programa y no solo en el código.

Lo hice de esta manera porque creo que es la que mejor estructurada queda y más entendible es.

Cree una función por cada variable que pedía, de esta manera si había un error de código sabia donde era y algunas funciones como validarNIF() las podría aprovechar más adelante, aunque hay otras funciones como pedir el nombre o apellido eran muy fáciles, igualmente al poner los return tiene sentido esta manera de hacerlo.

Vosotros me disteis el while con el switch aunque este último tenia trampa porque estaba mal, no puedes llamarle exit() porque es una palabra reservada del lenguaje Python, por esa misma razón la reemplace por otra función que sea salir.

El default ya me lo proporcionasteis vosotros.