UANL
UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACPYA
FACULTAD DE CONTADURÍA PÚBLICA Y ADMINISTRACIÓN

PRODUCTO INTEGRADOR DE APRENDIZAJE

Nombre de la Unidad de Aprendizaje	**Programación Avanzada**
Nombre del Proyecto:	**Libreta de Contactos en Consola**
Programa Educativo:	**Licenciado en Tecnologías de la Información**
Semestre:	**2**
Grupo:	**23**
Nombre del Maestro:	**Dr. Felipe Ramírez**

MIEMBROS DEL EQUIPO

MATRÍCULA	NOMBRE COMPLETO	Grupo
1928058	COLUNGA DÍAZ ALFREDO	23
1909640	MARIN MARTINEZ ALAN ORLANDO	23
1726329	ALVARADO RAMOS JESUS EDUARDO	23
1993455	CRUZ RAMIREZ ANTONIO	23

Contenido mínimo a evaluar	Cumplimiento
Índice	
Introducción .- incluye valores UANL aplicados	
Análisis y emisión de juicio	
Conclusiones individuales	
Conclusión del equipo	
Actividad en inglés	
Identificación de sub resultados de aprendizaje ANECA.	
Calificación PIA:	
Firma del maestro	

San Nicolás de los Garza, ciudad universitaria a (Mayo, 2020)

ÍNDICE

1) Introducción (ANECA 6.2)

El presente Producto Integrador de Aprendizaje trata una nueva perspectiva práctica para nosotros a partir de una manera particular de programar, estableciendo clases con la estructuración de código en unidades, y así crear objetos para lograr fines específicos dentro de nuestro programa previamente establecido.

Este trabajo nos resultó bastante útil dentro de lo que son los aprendizajes adquiridos, puesto que pudimos instruirnos a deducir resoluciones de una manera distinta a la que ya se tenía presente y acrecentamos nuestras capacidades para el aclaramiento de incógnitas racionales.

2) Valores UANL (ANECA 4.1)

Actitud de Servicio: Se lleva a cabo la resolución de una tesitura frecuente dentro de nuestra área profesional como lo es el poner en funcionamiento sistemas estructurados de compuestos de información.

Ética: Se trabaja desde la aplicación de decisiones conscientes y potestativas, procurando, no solo los beneficios individuales del equipo, sino que también el bien común.

Honestidad: Se busca ofrecer determinaciones firmes, apropiadas y congruentes del caso planteado empleando un correcto uso de la información dedicada.

Integridad: Se toman enserio los métodos y propósitos del Producto Integrador de Aprendizaje y se presenta un trabajo inédito.

Justicia: Se reconocen y respetan las idoneidades de cada uno de los integrantes del equipo y se comprende de manera objetiva la situación con la finalidad de trabajar de un modo justo.

Responsabilidad: Se reconocen y se valoran las consecuencias de los incidentes contemporáneos y se desarrolla un sistema de datos que sirve como pauta para ser reproducido y utilizado de manera eficiente para brindar apoyo a las actuales coyunturas.

Respeto: Se transigieren las limitaciones de cada integrante del equipo y se trabaja a partir de la consideración de las virtudes de estos.

Solidaridad: Se busca proveer cimientos a nuevos sistemas de datos para alcanzar mejores propuestas de solución a las diversas problemáticas inaplazables que demandan ser resueltas con presteza.

Trabajo en Equipo: Se hace uso de programas, aplicaciones y plataformas con estrategias asistidas por el docente, de la misma manera se exterioriza la resolución de problemas y aprendizajes previos en conjunto con los compañeros de equipo.

Verdad: Se aspira a la implementación de sistemas de datos que logran cumplir correctamente con su cometido respetando la información y recursos con los que se desempeñen.

3) Actividad en inglés

Activity in English. Produce a video and upload it on YouTube.

The video must explain some of the following topics:

1) Use of regular expressions
2) How to store objects in a list.
3) How to read a list, secuentially.
4) The data type.
5) How to get correct data.
6) Casting in Python.
7) Things like that!

The video must have a minimum duration of 2 minutes, and a maximum of 5 minutes.

https://youtu.be/X8X9rCAY9Qo

Don't mention all the team… just the presenter….

"Hi, there… Paco Valdés here!

In this video I will explain the important reasons to use regular expressions to validate data in Python, and…. Bla bla bla"

4) Caso práctico (Análisis)

Describir aquí el caso práctico, como tú lo entendiste o analizaste. Es el tema de una libreta de contactos.

Se tendrá un menú, que permitirá agregar, buscar, y borrar contactos. Además, podrá mostrar un listado de contactos, y permitirá guardar la información en un archivo CSV.

Para la creación de la libreta de contactos que se está solicitando necesitamos 2 programas principales en el primero clasepia.py importamos datetime para convertir cadenas de texto a fechas y así optimizar la manipulación de los datos, esta clase cuenta con 6 atributos que se utilizaran en el programa principalpia.py.

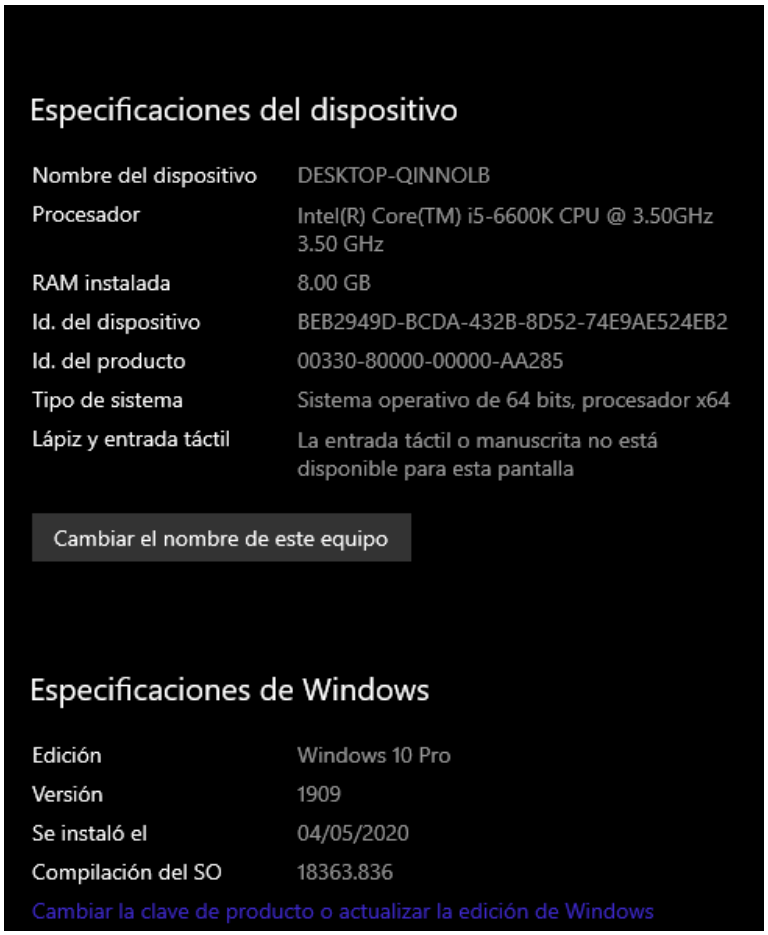
En principalpia.py es donde creamos el menú con todas sus funciones. Es esencial importar los módulos y librerías que se utilizaran y la clase Contactos del primer programa, para después definir la lista global donde se guardara los contactos, primero se cargamos el archivo y carga a la lista los contactos que tenga almacenados, en caso de no existir un archivo el programa se encarga de crearlo, llama a la función menú principal donde muestra en un ciclo las opciones que puede elegir el usuario.

- Opción 1 agrega un usuario a la lista, pero primero valida que los datos que pide estén en formato valido con la función pregunta.
- Opción 2 se crea una función que pregunta por un nickname, valida que este se ingresó de forma valida, busca el contacto y muestra el contenido que coincide, en el caso de no encontrar coincidencias está nos notifica que el contacto no existe, en esta opción principalmente se hace uso de la función buscarContacto que contiene el método de búsqueda que implementa un for.
- Opción 3 busca un teléfono por nickname y si encuentra coincidencias procede a eliminar.
- Opción 4 muestra secuencialmente todos los contactos, pero primero los ordena por nickname, en esta función utilizamos principalmente la función mostrarContacto que nos permite generar un bloque de código con referencia que se encarga de leer secuencialmente el archivo usando el flujo de datos.
- Opción 0, ordena la lista, remplaza los archivos y escribe cada contacto en cada línea, se verifica que exista el archivo de trabajo y si es así se verifica que exista un archivo de respaldo para proceder a borrarlo y se finaliza el ciclo, en caso de ingresar una opción no valida se iniciara un ciclo que nos indicara que la opción no es válida y volverá a mostrar el menú principal hasta que la opción sea válida.

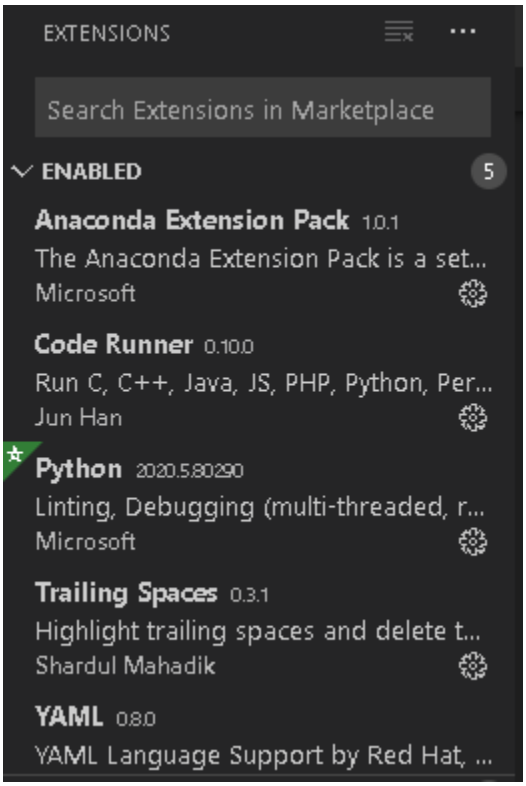
5) Ambientación (ANECA 1.2)

Especificaciones técnicas del equipo principal en el que se realizó la compilación y testeo de estos programas

Principalmente se trabajó en un pc de escritorio con Windows 10 Pro de 64 bits, versión 1909 en español latino, un procesador Intel Core I5 6600k @ 3.50 GHz, una memoria RAM de 8 GB a 3000 MHz en single channel.



El software utilizado fue Visual Studio Code en su versión 1.45.1 para Windows de 64 bits con las siguientes extensiones:



6) Codificación (ANECA 2.2)

a) Especificación de datos

Se desea crear una aplicación móvil que registre una libreta de contactos, que incluye los siguientes datos:

1. **`nickname`**: Es el nombre a través del cual el contacto desea ser referido. Se trata de caracteres alfanuméricos (sólo letras y números), debe una longitud entre 5 y 15 caracteres, debe empezar necesariamente con una letra.
2. **`nombre`**: Es el nombre completo de la persona. Se trata de caracteres alfanuméricos (sólo letras mayúsculas y espacios, sin acentos), debe una longitud entre 10 y 35 caracteres, debe empezar necesariamente con una letra.

3. **`correo`**: Es el correo electrónico de la persona. Debe estar en minúsculas, y ser un correo de estructura básica.
4. **`teléfono`**: Es el teléfono de la persona en formato «`99 9999-9999`».
5. **`fechanacimiento`**: Fecha de nacimiento de la persona. Debe ser una fecha en formato «`aaaa-mm-dd`», y debe ser una fecha válida.
6. **`gastos`**: Monto de dinero que usualmente gasta en un mes para sobrevivir y cubrir sus gastos. Debe ser una cantidad entera con dos decimales, mayor a cero.

La información de la libreta de contactos se almacenará en un archivo CSV llamado contactos_mobil.csv, delimitado por PIPES (|). Su encabezado es:

NICKNAME|NOMBRE|CORREO|TELEFONO|FECHANACIMIENTO|GASTOS

b) Definición de clase autónoma

`clasepia.py`, sólo contendrá la estructura de la clase llamada **Contacto**.

La clase debe tener un método constructor (__init__). Debe invocar los módulos que se ocupen, dependiendo de los tipos de datos que intervienen en la clase.

```
# Estructura de la clase <<Contacto>>
# Se importa módulo: datetime
import datetime

# Se crea la clase <<Contacto>>
class Contacto:
    def __init__(self, nickname, nombre, correo, telefono, fechaNac, gasto):
        # NICKNAME: Nombre a traves del cual el contacto desea ser referido
        # NOMBRE: NOmbre completo del contacto
        # CORREO: Correo electrónico del contacto
        # TELEFONO: Número telefónico del contacto
        # FECHANACIMIENTO: Fecha de nacimiento del contacto
        # GASTO: Gasto mensual del contacto
        self.nickname = nickname
        self.nombre = nombre
        self.correo = correo
        self.telefono = telefono
        self.fechaNac = fechaNac
        self.gasto = gasto
```

c) Especificaciones iniciales

`principalpia.py`, debe consumir la clase definida en **`clasepia.py`**.

Ver `entradaContacto.py` líneas 5-6.

Colocar aquí la parte del programa que se encarga de llamar los módulos (librerías) que serán consumidas en el programa: al menos, se requiere **re**, **datetime**, **csv** y **os**


```
# Se importa modulo para trabajar con el sistema operativo
import os
# Se importa módulo para trabajar con archivos csv
import csv
# Se importa módulo para trabajar con expresiones regulares
import re
# Se importa módulo para trabajar con datos tipo datetime
import datetime

captura = ""

# Función lambda para limpiar pantalla
LimpiarPantalla = lambda: os.system('cls')

# Variables globales
global nickname
global nombre
global correo
global telefono
global fechaNac
global gasto

# Acceso de datos a <<clasepia>>
from clasepia import Contacto
# Acceso módulo <<operator>>
from operator import attrgetter
```

n

d) Cargar información de CSV

Al arrancar el programa, lo que hará será leer el archivo **contactos_mobil.csv**, y cargará todos los contactos en una lista llamada **Contactos**, que contiene objetos basados en la clase **Contacto**.


```
# with genera bloque de código con referencia
# Cuando se escriba archivo_csv se trabajara con el contenido del archivo: contactos_mobil.csv
with open('contactos_mobil.csv') as archivo_csv:
    lector_csv = csv.reader(archivo_csv, delimiter='|')
    contador_lineas = 0
    # Creando una lista vacía.
    Contactos = []
    # Lectura secuencial.
    for linea_datos in lector_csv:
        if contador_lineas == 0:
            # Si es la primer línea, muestra los nombres de campo y no guarda nada en la lista.
            print(f'Los nombres de columna son {", ".join(linea_datos)}')
        else:
            # Si son datos (línea uno y posteriores)...
            # Genera  instancia de la clase <<Incidente>> y le proporciono al constructor los valores leidos del archivo.
            objeto_temporal = Contacto(linea_datos[0],linea_datos[1],linea_datos[2],linea_datos[3],linea_datos[4],linea_datos[5])
            Contactos.append(objeto_temporal)

        # Se incrementa el número de líneas, pase lo que pase.
        contador_lineas += 1
```

e) Menú principal

principalpia.py, debe ejecutar un procedimiento llamado **principal**, que ejecutará un menú permanente, que muestre lo siguiente:

```
[1] Agregar un contacto.
[2] Buscar un contacto.
[3] Eliminar un contacto.
[4] Mostrar contactos.
[0] Salir.
¿Qué quieres hacer? __
```


El menú general estará repitiéndose hasta que se proporcione «0». Cada elección, mandará llamar un procedimiento específico, que realice la acción solicitada. Se debe validar la opción introducida, usando expresiones regulares. Cada que aparezca el menú, deberá limpiar la pantalla antes.

```
# Función para validador de opciones
def RegEx(_txt,_regex):
    coincidencia = re.match(_regex, _txt)
    return bool(coincidencia)
```

```
# Funcion que ejecuta el menú, se repite hasta que el usuario ingrese "0"
def principal():
    while (True):
        LimpiarPantalla()
        print("LISTA DE OPCIONES")
        print(" ")
        print("[1] Agregar un contacto.")
        print("[2] Buscar un contacto.")
        print("[3] Eliminar un contacto.")
        print("[4] Mostrar contactos.")
        print("[0] Salir.")
        opcion_elegida = input("¿Qué deseas hacer?  > ")
        if RegEx(opcion_elegida,"^[123450]{1}$"):
            if opcion_elegida == "0":
                print("GRACIAS POR UTILIZAR EL PROGRAMA")
```

```
            finalizar()
            break
        if opcion_elegida == "1":
            print("Llamar procedimiento para la acción")
            ingresoDatos()
        if opcion_elegida == "2":
            print("Llamar procedimiento para la acción")
            VariableTelefono = input("inserte el nickname del contacto en mayusculas: ")
            indice_obtenido = buscarContacto(VariableTelefono)
            if indice_obtenido==-1:
                print("No se encontró el objeto")
            else:
                print(Contactos[indice_obtenido].TELEFONO)
                print(Contactos[indice_obtenido].NOMBRE)
                print(Contactos[indice_obtenido].CORREO)
        if opcion_elegida == "3":
            print("Llamar procedimiento para la acción")
            VariableTelefono = input("inserte el nickname del contacto a borrar en mayusculas: ")
            indice_obtenido = buscarContacto(VariableTelefono)
            if indice_obtenido==-1:
                print("No se encontró el objeto")
            else:
                print("el contacto borrado es: ")
                print(Contactos[indice_obtenido].TELEFONO)
                print(Contactos[indice_obtenido].NOMBRE)
                print(Contactos[indice_obtenido].CORREO)
                del Contactos[indice_obtenido]
        if opcion_elegida == "4":
            print("Llamar procedimiento para la acción")
            mostrarContactos()
        input("Pulse enter para continuar...")
    else:
        print("Respuesta no válida.")
        input("Pulse enter para contunuar...")

principal()
```

f) Barrido de lista de objetos (Mostrar contactos)

Implementar la opción **[4] Mostrar contactos**.

Elabora una función / procedimiento, que muestre la información de los contactos, de manera secuencial. Debe limpiar la pantalla al inicio, y al final, debe esperar "Presione enter para continuar". Al dar enter, vuelve al menú principal. La lista debe estar ordenada, por nickname.

Debe mostrarse algo así:

```
Nick name:                  nick
Nombre:                     nombre completo
Teléfono:                   99 9999-9999
Fecha nacimiento:           aaaa-mm-dd
Gastos:                     $99,999.99
---------------------- ---
Nick name:                  nick
Nombre:                     nombre completo
Teléfono:                   99 9999-9999
Fecha nacimiento:           aaaa-mm-dd
Gastos:                     $99,999.99
```

Para limpiar la pantalla al inicio de cada proceso utilizamos la siguiente función.

```
# Función lambda para limpiar pantalla
LimpiarPantalla = lambda: os.system('cls')
```

Creación de la opción 4 que permite llamar a la función mostrarContactos, en esta se genera un proceso por el cual se muestran los contactos previamente registrados.

```
        if opcion_elegida == "4":
            print("Llamar procedimiento para la acción")
            mostrarContactos()
        input("Pulse enter para continuar...")
```

En la función mostrarContactos indicamos los parámetros que utilizara nuestro programa para almacenar los datos ingresados.

```
# Función para imprimir contactos registrados
def mostrarContactos():
    # with genera bloque de código con referencia
    # Cuando se escriba archivo_csv se trabajara con el contenido del archivo: contactos_mobil.csv
    with open('contactos_mobil.csv') as archivo_csv:
        # Se habilita objeto que permitira leer de manera secuencial el contenido del archivo csv
        # archivo_csv es el puente entre el programa y el archivo
        # lector_csv es el flujo de datos entre el programa y el archivo
        lector_csv = csv.reader(archivo_csv, delimiter='|')

        # Contador de lineas
        contador_lineas = 0

        # Lee secuencialmente el archivo usando el flujo de datos (lector_csv)
        # La linea se coloca en el elemento linea
        # Al trabajar con <<linea_datos>> se trabaja con la linea del archivo
        # for dejará de leer cuando los datos del archivo se hayan terminado
        for linea_datos in lector_csv:
            # Si el contador es mayor a cero: muestra, dato por dato, los datos obtenidos en linea_datos
            if contador_lineas > 0:
                print(f'\tNICKNAME: {linea_datos[0]}.')
                print(f'\tNOMBRE: {linea_datos[1]}.')
                print(f'\tCORREP: {linea_datos[2]}.')
                print(f'\tTELEFONO: {linea_datos[3]}.')
                print(f'\tFECHA DE NACIMIENTO: {linea_datos[4]}.')
                print(f'\tGASTO: {linea_datos[5]}.')
                print(f'\t----------------------------------------')
            # Se incrementa el número de lineas sin importar la condición
            contador_lineas += 1
            # Se cierra el archivo csv
```

Para ordenar los datos almacenados utilizamos la función finalizar que se encarga de determinar si existe un archivo de trabajo y de no ser así esta misma genera uno.

```
# Función para el ordenado de datos
def finalizar():
    Contactos.sort(key=attrgetter("NICKNAME"),reverse=False)
    # Se guarda en la variable la ruta absoluta, del directorio actual de trabajo (cwd)
    ruta = os.path.abspath(os.getcwd())
```

```
  archivo_trabajo=ruta+"\\contactos_mobil.csv"
  archivo_respaldo=ruta+"\\contactos_mobil.bak"

  # Se determina si el archivo de trabajo ya existe
  if os.path.exists(archivo_trabajo):
     # Si el archivo existe, entonces verifica si hay respaldo y lo borra
     if os.path.exists(archivo_respaldo):
        os.remove(archivo_respaldo)

     # Establece el achivo de datos, como respaldo
     os.rename(archivo_trabajo,archivo_respaldo)

  # Genera el archivo CSV
  f = open(archivo_trabajo,"w+")
  # Se escriben los encabezados del archivo csv
  f.write("NICKNAME|NOMBRE|CORREO|NUMERO|FECHANACIMIENYO|GASTOS\n")
  # Se escribe en el csv, a partir de la lista de objetos
  for elemento in Contactos:
     f.write(f'{elemento.NICKNAME}|{elemento.NOMBRE}|{elemento.CORREO}|{elemento.TELEFONO}|{elemento.FECH
ANACIMIENTO}|{elemento.GASTO}\n')

  # Se cierra el archivo csv
  f.close()
  print("Elementos ordenados por NICKNAME")
```


g) Agregar datos validados (Agregar contactos)

Implementar la opción **[1] Agregar un contacto**.

Elabora una función / procedimiento, pregunte todos los datos de un contacto, de forma indefinida, hasta que se capturen correctamente. Para validar, deben usarse

expresiones regulares. Al final, una vez que se tienen los datos correctos, se cargan en una nueva instancia de Contacto, y se carga a la lista general que tiene todos los datos. Obviamente, sólo se puede agregar un contacto cuyo nickname no esté registrado. Si se intenta agregar uno que ya existe, debe decir que no se puede. Si se captura enter en nickname estando vacío, vuelve a menú.

Creamos una función llamada pregunta que se encarga de validar que los datos ingresados tienen el formato que necesitamos.


```
# Función para validar los datos
def pregunta(_formato,_pregunta="Datos: "):
    # Se especifica que "_captura" es global
    global _captura
    while True:
      _dato = input(_pregunta)
      valido = re.search(_formato,_dato)
      if (valido):
        _captura = _dato
        break
      else:
        print("Dato no válido. Intente de nuevo")
```

```
# Función para captura de datos
def ingresoDatos():
    global nickname
    global nombre
    global correo
    global telefono
    global fechaNac
    global gasto
    # Se pide el nickname
    pregunta("^[A-Z ]{1,15}$", "Ingrese nickname en mayusculas: ")
    nickname = _captura
    # Se pide el nombre completo
    pregunta("^[A-Z ]{1,35}$", "Ingrese nombre completo en mayusculas: ")
    nombre = _captura
    # Se pide el correo electrónico
    pregunta("[\w\.-]+@[\w\.-]+(\.[\w]+)+", "Ingrese correo electronico: ")
    correo = _captura
    # Se pide el número telefónico
    pregunta("^(\d{2} \d{4}-\d{4})", "Ingrese numero telefonico (99 9999-9999): ")
    telefono = _captura
```

```
    # Se pide la fecha de nacimiento
    pregunta("^[0-9]{4}-[0-9]{2}-[0-9]{2}", "Ingrese fecha de nacimiento (YYYY-MM-DD): ")
    fechaNac = _captura
    # Se pide el gasto mensual
    #pregunta("^([0-9]){1,7}$", "Ingrese gasto mensual: ")
    pregunta("^[0-9]+(\.[0-9]{1,2})?$", "Ingrese gasto mensual: ")
    gasto = _captura

    Contactos.append(Contacto(nickname,nombre,correo,telefono,fechaNac,gasto))
        if opcion_elegida == "1":
            print("Llamar procedimiento para la acción")
            ingresoDatos()
```



```
# Función para validar la existencia del número telefónico buscado
def buscarContacto(NicknameBuscar):
    contador =- 1
    indice_retorno =- 1
    for Contacto in Contactos:
        contador += 1
        if (Contacto.NICKNAME == NicknameBuscar):
            indice_retorno = contador
            break
    return indice_retorno
```

h) Buscar elementos en una lista (Buscar contacto)

Implementar la opción **[2] Buscar contacto**.

Elabora una función / procedimiento, pregunte por un nickname, de forma validada. Si hay un objeto en la lista cuyo Nick name que coincida con lo capturado, debe mostrarse el contenido del objeto coincidente. Si no existe, reporta que no existe. Si se captura enter en nickname estando vacío, vuelve al menú.

```
# Función para validar los datos
def pregunta(_formato,_pregunta="Datos: "):
```

```
  # Se especifica que "_captura" es global
  global _captura
  while True:
    _dato = input(_pregunta)
   valido = re.search(_formato,_dato)
   if (valido):
     _captura = _dato
     break
   else:
     print("Dato no válido. Intente de nuevo")
```



```
# Función para validar la existencia del número telefónico buscado
def buscarContacto(NicknameBuscar):
   contador =- 1
   indice_retorno =- 1
   for Contacto in Contactos:
      contador += 1
      if (Contacto.NICKNAME == NicknameBuscar):
         indice_retorno = contador
         break
   return indice_retorno
```

```
         if opcion_elegida == "2":
            print("Llamar procedimiento para la acción")
            VariableTelefono = input("inserte el nickname del contacto en mayusculas: ")
            indice_obtenido = buscarContacto(VariableTelefono)
            if indice_obtenido==-1:
               print("No se encontró el objeto")
            else:
               print(Contactos[indice_obtenido].TELEFONO)
               print(Contactos[indice_obtenido].NOMBRE)
               print(Contactos[indice_obtenido].CORREO)
```

i) Eliminar elementos en una lista (Buscar contacto)

Implementar la opción **[3] Eliminar un contacto**.

Elabora una función / procedimiento, pregunte por un nickname, de forma validada. Si hay un objeto en la lista cuyo Nick name que coincida con lo capturado, debe eliminarlo. Si se da enter en nickname sin tener valor, vuelve a menú.


```
# Función para validar la existencia del número telefónico buscado
def buscarContacto(NicknameBuscar):
    contador =- 1
    indice_retorno =- 1
    for Contacto in Contactos:
        contador += 1
        if (Contacto.NICKNAME == NicknameBuscar):
            indice_retorno = contador
            break
    return indice_retorno
```

```
        if opcion_elegida == "3":
            print("Llamar procedimiento para la acción")
            VariableTelefono = input("inserte el nickname del contacto a borrar en mayusculas: ")
            indice_obtenido = buscarContacto(VariableTelefono)
            if indice_obtenido==-1:
                print("No se encontró el objeto")
            else:
                print("el contacto borrado es: ")
                print(Contactos[indice_obtenido].TELEFONO)
                print(Contactos[indice_obtenido].NOMBRE)
                print(Contactos[indice_obtenido].CORREO)
                del Contactos[indice_obtenido]
```

j) Ordenar lista, guardar en CSV nuevo (Salir)

Implementar la opción **[0] Salir**.

Esta es una opción compleja, que realiza varias tareas. La primera, ordenar los elementos de la lista Contactos, por nickname. La segunda, pasar el archivo **`contactos_mobil.csv`** a **`contactos_mobil.bak`**. Borrar **`contactos_mobil.csv`**, y crear uno nuevo. La tercera parte, es escribir los elementos de la lista, en el CSV creado.

```
# Función para el ordenado de datos
```

```
def finalizar():
    Contactos.sort(key=attrgetter("NICKNAME"),reverse=False)
    # Se guarda en la variable la ruta absoluta, del directorio actual de trabajo (cwd)
    ruta = os.path.abspath(os.getcwd())
    archivo_trabajo=ruta+"\\contactos_mobil.csv"
    archivo_respaldo=ruta+"\\contactos_mobil.bak"

    # Se determina si el archivo de trabajo ya existe
    if os.path.exists(archivo_trabajo):
        # Si el archivo existe, entonces verifica si hay respaldo y lo borra
        if os.path.exists(archivo_respaldo):
            os.remove(archivo_respaldo)

        # Establece el achivo de datos, como respaldo
        os.rename(archivo_trabajo,archivo_respaldo)

    # Genera el archivo CSV
    f = open(archivo_trabajo,"w+")
    # Se escriben los encabezados del archivo csv
    f.write("NICKNAME|NOMBRE|CORREO|NUMERO|FECHANACIMIENYO|GASTOS\n")
    # Se escribe en el csv, a partir de la lista de objetos
    for elemento in Contactos:
        f.write(f'{elemento.NICKNAME}|{elemento.NOMBRE}|{elemento.CORREO}|{elemento.TELEFONO}|{elemento.FECH
ANACIMIENTO}|{elemento.GASTO}\n')

    # Se cierra el archivo csv
    f.close()
    print("Elementos ordenados por NICKNAME")
```


7) Conclusiones individuales

Alfredo Colunga Díaz: Al contribuir en la codificación del caso práctico y en la redacción del Producto Integrador de Aprendizaje me percaté de que una de las dificultades que tuvo mayor presencia durante la realización del proyecto fue la estructuración de las funciones en el código del programa y la lectura del archivo csv, pues a pesar de que se tenía un concepto claro de las necesidades planteadas, estas no se lograban cumplir con exactitud y su resolución fue moderadamente complicada de realizar.

Con el manejo del modelo de aprendizaje electivo y el empleo del lenguaje de programación Python pude prepararme de manera provechosa entorno a la programación orientada a objetos, acrecentando mis conocimientos acerca de los métodos usados en este modelo de programación, de igual manera aprendí a trabajar con un plan de aprendizaje organizado a partir de las cognotécnicas relacionadas con mi estilo de aprendizaje.

Alan Orlando Marín Martínez: La realización de la codificación y estructuración del programa resulto ser bastante más complicada de lo que se llegó a pensar en un principio, ya que, a mi pensar el modelo de aprendizaje electivo no resulto muy eficaz en mi persona, siendo este el reto más grande hacia mí.

Como aprendizaje nuevo logre, más que aprender, mejorar mis métodos de programación a la hora de generar funciones y utilizar la programación orientada a objetos.

Alvarado Ramos Jesus Eduardo: En la realización de este Producto Integrador Final pudimos poner a prueba nuestros conocimientos en mi caso poder practicar mediante la ejecución de este PIA con esto me fue más fácil realizar otras actividades como las evidencias vistas en clases, también aprender un poco sobre Python, a pesar de las circunstancias el equipo logro terminar este PIA y de manera correcta, así mismo coincidimos con lo expuesto en clases del maestro, como consecuencia de lo expuesto en el trabajo nos enfocamos en diferentes tópicos relacionados a Python.

Es muy sencillo realizar un trabajo en equipo siempre que se cuente con las herramientas adecuadas para hacerlo y la participación de todos. La ayuda del profesor que nos prestó toda su colaboración logro que de esta forma lográramos hacer un buen proyecto.

Antonio Cruz Ramírez: Este PIA a mi parecer fue el más complejo y también fue el que solicito más tiempo y dedicación, fue necesaria la comunicación clara al estar trabajando cuatro personas con los mismos dos programas, pero todo esto se logro gracias a las herramientas de comunicación instantánea con las que contamos en la actualidad.

Gracias a este PIA pude reforzar conocimientos que había acumulado hasta el momento, pero también aprendí y comprendí algunos conceptos que aun no me quedaban claros, por otra parte, se puede decir que simulamos la experiencia de realizar un proyecto para una empresa (aunque es claro que en escalas no se compra) para lo cual se necesita de mucha comunicación y trabajo en equipo.

8) Conclusiones de equipo

Para llevar a cabo la realización de este producto integrador fue de vital importancia la participación de cada uno de los integrantes, es necesario aclarar que gracias a esto se solucionaron de forma mas efectiva diversos inconvenientes que surgieron durante el proceso de codificación, también se puede decir que se redujo considerablemente la carga de trabajo lo cual ayudo en gran parte a que este PIA quedase igual o por lo menos se acercara a lo que solicito el maestro.

Pero aun con todo esto fue necesario hacer consultas en webs de programación para tener otros puntos de vista, también por parte de nuestro equipo agradecemos los

materiales otorgados por el Profesor ya que estos nos sirvieron como guía. Nuestro equipo coincide en que la comunicación, responsabilidad y la dedicación fueron elementos fundamentales para lograr terminar este proyecto.

9) Referencias

Colocar aquí las referencias y fuentes que utilicen para el desarrollo de este trabajo. En formato APA.

Revisen este video. Word lo hace por ustedes.

https://www.youtube.com/watch?v=kpxPddiNs3Q

Ramírez, F. (2000). *Cognotécnicas, herramientas para pensar más y mejor.* México D.F., México : AlfaOmega.

Ramírez, F. (12 de 05 de 2020). *AprendaEnLínea*. Obtenido de http://www.aprendaenlinea.mx/p/cognotecnicas

Real Python. (02 de 05 de 2020). *Real Python Tutorials*. Obtenido de Real Python: https://realpython.com/

W3Schools. (02 de 05 de 2020). *W3Shools - Python tutorial*. Obtenido de W3Schools: https://www.w3schools.com/python/default.asp