Examen Final de Informática

26/07/23

Nombre del Docente que corrige: _____ **Nota:** _____

- Se debe tener un total de 60% para poder aprobar en CADA uno de las tres partes.
- Tiempo de desarrollo 1 hora 30 minutos.

Nombre del Alumno: _____

Python (Parte 1)

Escribir un programa en Python que permita almacenar en un diccionario datos personas. Las claves del Diccionario serán los nombres de los alumnos y los valores para cada alumno que se almancenan en una lista son:

Edad (entero) , Peso (real) , Altura (real)

El programa presentará un menú invocando a la **función menú** que muestra:

1) Carga de Datos de una persona.

2) Calculo del promedio de altura de la personas.

3) Obtener el nombre de la persona mas pesada.

S o s) Salir del programa.

Función menú (10 puntos): No recibe argumento, y regresa SOLAMENTE 1,2,3,S o s .

Punto 1 (20 puntos): se invoca a la **función ingreso** que recibe como **argumento una clave válida** del diccionario. Desde el programa principal se debe pedir reingreso si la clave no existe, luego de ingresar una clave válida se envía esta como argumento a la función ingreso. En la función se cargan los valores de Edad (entero) , Peso (real) , Altura (real). Ver que todos los valores posibles SON POSITIVOS. En caso de no ser positivo el valor ingresado, **se debe solicitar el reingreso SOLO de ese valor.** La función ingreso deber retornar una variable booleana , que se utilizará como validación para habilitar las opciones 2, 3 y 4 del menú.

Punto 2 (30 puntos): se invoca a la **función promedio_altura** que **recibe como argumento el diccionario** y **retorna** al programa principal el promedio de altura de las presonas.

Tener en cuenta que si NO se cargó previamente (punto 1) NO se debe realizar la operación.

Punto 3 (30 puntos): Mostrar el nombre de la persona mas pesada usando invocando a la **función mas_pesado**, que recibe como argumento el Diccionario y regresa la clave que corresponde al nombre de la persona mas pesada.

Lógica, Consigna, General 10 puntos

Cantidad de funciones 4:

menú 10 puntos

ingreso 20 puntos

promedio_altura 30 puntos

mas_pesado 30 puntos

Calc (Parte 2)

Un mercado que cuenta con tres empleados quiere implementar un sistema para llevar las ventas de cada uno de ellos y así tener un seguimiento de los mismos. Para ello se plantea usar "LibreOffice Calc" para desarrollar una hoja de cálculo que lo permita hacer, la misma se muestra a continuación:

Python

Un centro de investigación biomédico necesita un programa que le permita registrar datos de tratamientos con los diferentes medicamentos que tiene bajo estudio. De cada Medicamento, se tiene

- una clave de identificación alfanumérica (una cadena que puede tener dígitos y caracteres).
- una descripción.(texto)
- el monto invertido en el proyecto ((0;∞)
- un número entre 1 y 30 que indica el tipo de medicamento (por ejemplo: 1: Cardíaco, 2: Inmunológico, etc.),

En base a lo anterior, desarrollar un programa completo que disponga el siguiente menú (**con función menú**) :

- 1. Carga
- 2. Muestra
- 3. Busca
- 4. Total
- s ó S Sale

1. Carga. Genera un diccionario que contenga los datos de todos los medicamentos en estudio. Cada vez que se llama a la **función carga,** se carga un medicamento. Realizar las validaciones que crea necesario. Esto es realizado por la función carga. la cual regresa un booleano que se utiliza para saber que al menos una vez se cargó algo.

2. Muestra: visualiza en pantalla el diccionario generado, a razón de un registro por línea.

3. Busca: llama a la **función buscar**, que determina si existe en el diccionario un medicamento en el que la clave del mismo (que recibe como argumento) coincida. Si existe, retorna la cadena contenida en el campo descripción y detiene la búsqueda. Si no existe, agregue un nuevo registro con esa clave al diccionario, con las validaciones que correspondan.

4. Total Determina el monto el promedio invocando a la **función promedio** que recibe el diccionario como argumento y regresa el promedio de inversión entre todos los proyectos.

S o S: sale del programa.

función menú regresa solamente a la función proncipal : 1,2,3,s ó S.

Cualquier va valor ingresado que no corresponda a la limitaciones del campo deben ser reingresados.

unciones:

enú (10 puntos),carga(30 puntos),buscar (30 puntos),promedio(20 puntos), codigo principal(10 puntos)

Examen Final de Informática Python 14/12/22

Nombre del Docente que corrige:_____ **Nota:** _____

- Se debe tener un total de 60% para poder aprobar en CADA una de las partes.
- Tiempo de desarrollo 1 hora 15 minutos.

Nombre del Alumno: Zang Gonzalo Ezequiel

Python (Parte 1)

Realizar un programa en Python que permita almacenar datos de 2 líneas de producción de una fábrica. Cada línea procesa el mismo tipo de producto, pero los valores de temperaturas y humedades distintas, es decir para cada Línea se almacenan varias temperatura y humedades.

Se propone crear un diccionario con una clave para cada línea de producción, y que cada clave almacene en dos listas los valores de temperaturas y humedades. A continuación, se muestra un ejemplo de la Estructura de datos propuesta:

Diccionario={"Linea1":[[temperaturas] [humedades]],"Linea2":[[temperaturas] [humedades]] }

Definir N que será una constante simbólica definida en el programa principal. N será la cantidad de pares de valores que se cargan en una línea, mas adelante te se explica el uso.

El programa presentará un menú invocando a la **Función menú** que muestra:

1) Carga de valores de Temp y Humedad para una línea

2) Calculo de cantidad de mediciones de cada línea.

3) Listado de los registros de temp y humedad para las dos líneas .

4) Promedio de temperatura y humedad para cada línea.

S o s) Salir del programa.

Función menú (10 puntos) : No recibe argumento, y regresa SOLAMENTE 1,2,3,4,S o s .

Punto 1 (30 puntos): se invoca a la **función ingreso** que recibe como **argumento una la clave válida** (Linea1 o Linea2 del diccionario) . Desde el programa principal se debe pedir reingreso si la Clave(Línea1 o Linea2) no es ingresada, luego de ingresar una clave válida se envía esta como argumento a la función ingreso. En la función se cargan **N** valores de Temperatura y luego **N** valores de Humedad para la línea que se recibe como argumento, que se agregan al los ya existentes en el diccionario/lista. **Recordar que N esta indicado como constante simbolica.** Las temperaturas deben encontrarse **(validar)** en [0;50] y la Humedad a [0;100). Si no están en ese rango **se debe solicitar reingreso.** La **función ingreso** deber retornar una variable booleana , que se utilizará para validar las opciones 2,3 y 4 que solo se puedan realizar si por lo menos se hizo una vez el punto 1. Esta **función ingreso** puede ser llamada las veces que sea, es decir TODAS las veces que se invoque se cargarían N pares de valores de temperatura y humedad.

Punto 2 (20 puntos) : se invoca a la **función cantidad** que **recibe como argumento el diccionario** y **retorna** al programa principal la **cantidad de valores cargados en la lista de cada línea**, este valor retornado se debe mostrar en la pantalla desde el programa principal. A modo de Ejemplo si el diccionario estuviera cargado con :

Diccionario={"Linea1":[[5,10] [80,90]],"Linea2":[[10,50, 45,38] [55,65,85,95]] }

Debería retornar 2 y 4 . 2 Valores cargados para línea 1 y 4 para la línea 2, en nuestro ejemplo N puede valer 1 o 2.

Tener presente que si NO se cargó previamente (punto 1) NO se puede realizar la operación.

Punto 3 (5 puntos): Se muestran los registros cargados al momento para cada línea.

Tener presente que si NO se cargó previamente (punto 1) NO se puede realizar la operación.

Punto 4 (20 puntos): Se debe llamar a la **función promedio**, que recibe como argumento el diccionario y regresa una lista con los valores de los **promedios de temperatura y humedad de cada línea (serían 4 valores en la lista)** y se muestran en el programa principal.

Tener presente que si NO se cargó previamente (punto 1) NO se puede realizar la operación.

Ejercicio de Python (Total 100 puntos se aprueba con 60%)

- Se descontarán puntos si no se respetan los nombres indicados en la consigna (funciones, variables, constantes) las mismas están en negritas.
- Se descontará el 50% del ítem si NO respeta la consigna Se descontará el 100% del Ítem si hay error conceptual.
- Puede escribir el programa en la IDE Spyder de Linux.

Consigna:

Un Meteorólogo tiene una formula experimental que quiere corroborar. Para ello lo que hace es cargar datos de un dd/mm (día/mes) y para ese día y mes carga Temperatura y presión atmosférica. El meteorólogo busca simplificar el resultado de un montón de ecuaciones de dinámica de fluidos y termodinámica que no son precisamente sencillas. Esas ecuaciones parten de los datos que generan los satélites o las estaciones meteorológicas para generar un modelo matemático que trata de reproducir el estado de la atmósfera en un momento dado. En nuestro caso el Meteorólogo postula:

"Si la presión atmosférica es menor a 800 y la temperatura superior a 25 grados debería llover."

- Valores válido de presión atmosférica: [760;960] mm de Hg (Mercúrio). Se reingresa si no está en ese rango.
- Valores válidos de Temperatura: [-10;55] grados centígrados.Se reingresa si no está en ese rango.
- Valores válidos de día: [1;30] enteros,Se reingresa si no está en ese rango.
- Valores válidos de mes: [1;12] enteros,Se reingresa si no está en ese rango.

Realizar un programa en Python que presente un menú (con **función menú**)como el siguiente:

1) Cargar datos de 1(un) día/mes (dd, mm, temperatura, presión)

2) Listar los valores cargados (1(un) día por línea)

3) Buscar predicción.

F ó f) Finalizar el programa.

Para el punto 1 del menú:

La carga se realiza con una **función carga** tendrá ciertas restricciones al cargar los valores. Estas ya fueron indicadas y se deben reingresar si no son correctas (ya se indicó esto, también). Lógicamente no pueden existir dos dd/mm iguales, esto se debe validar antes de cargar. Los datos cargados se almacenan en una varaible llamada: **Datos.**

Para el punto 2 del menú:

Lista los valores cargados, invocando a la **función listar**, que recibe como argumento la varaible **Datos** que contiene lo datos cargados por función carga.

Examen Final de Informática

27/07/22

Se debe tener total o parcialmente ***TODOS LOS TEMAS*** para que el docente corrija el Examen.

Ejercicio de Python (Total 100 puntos se aprueba con 60%)

- Se descontarán puntos si no se respetan los nombres indicados en la consigna (funciones, variables, constantes) las mismas están en negritas.
- Se descontará el 50% del ítem si NO respeta la consigna Se descontará el 100% del ítem si hay error conceptual.
- Puede escribir el programa en la IDE Spyder de Linux.

CONSIGNA:

Un docente desea automatizar la distribución de los alumnos de su cátedra en las diferentes comisiones y además desea poder llevar un control de los alumnos que regularizaron su materia y los que no.

Para ello se quiere realizar un programa en Python que presente el siguiente menú **(10 puntos)**

1) Cargar alumnos.

2) Cargar notas de parciales.

3) Ver datos.

S o s) Salir.

Se sale únicamente al elegir la opción de salir, en caso de seleccionar una opción incorrecta, se debe mostrar el siguiente mensaje: "**OPCIÓN INCORRECTA**" y volver a mostrar el menú completo nuevamente y pedir nuevamente el ingreso de una opción.

PUNTO 1 (Cargar alumnos)(40 puntos)

La carga se realiza con una **función carga** que recibe como **argumento la cantidad de alumnos** que se van a cargar, el cual se debe ingresar por teclado desde el programa principal (se debe validar que la misma sea mayor que cero, en caso contrario pedir reingreso). En la función se pide el ingreso del **N° de Legajo** (único para cada alumno) y el **apellido** y **nombre** del alumno (se debe verificar que el **N° de Legajo** no se encuentre cargado, en caso de estarlo, mostrar un mensaje diciendo: "**EL ALUMNO YA SE ENCUENTRA EN EL SISTEMA**" y pedir reingreso). Con estos datos el programa le debe asignar una comisión de forma automática siguiendo los siguientes criterios:

- Comisión 1: apellidos desde la letra **A** hasta la **N** inclusive.
- Comisión 2: apellidos desde la letra **Ñ** hasta la **Z** inclusive.

Almacenar los datos en una variable llamada **alumnos**.

La función **carga** retorna un **True** cuando termina. Esto se utiliza para validar que no se ejecuten los puntos 2 y 3 del menú sin antes haber ejecutado al menos una vez el punto 1 del menu.

PUNTO 2 (Cargar notas)(30 puntos)

Se cargan las notas de los alumnos, invocando a la función **cargarNotas**, que **SOLO** se puede ejecutar si ya se ejecutó el **punto 1**.

Esta función no recibe ningún argumento y en la misma se cargan las notas de los alumnos para lo cual el programa debe primero mostrar el legajo, el apellido y el nombre del alumno antes de pedir el ingreso de las notas del primer y segundo parcial de ese alumno, ambas notas deben estar entre [0;10], si no se pide reingreso de las NOTAS de ESE alumno, esto se debe realizar para cada alumno. Además, el programa debe cargar de forma automática la condición del mismo (**Regular:** Nota en los dos parciales mayor o igual a 6 y promedio mayor o igual a 7 o **Libre)**.

Las notas y la condición se deben almacenar en el diccionario **alumnos**.

La función **cargarNotas** retorna un **True** cuando termina.

PUNTO 3 (Ver datos)(20 puntos)

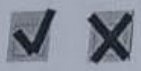
Esta opción del menú **SOLO** se puede ejecutar si la opción 1 y 2 del menú se ejecutaron con anterioridad. Antes de llamar a la función **debe preguntar al usuario si desea ver la versión resumida de los datos o todos los datos cargados** (en caso de ingresar una opción incorrecta se debe mostrar el mensaje "**OPCIÓN INCORRECTA**" y pedir el reingreso).

Esta función recibe como argumento la variable **alumnos** y la selección del usuario('r' de resumida, o 'c' de completa). En caso de querer ver la versión completa de los datos, la función muestra por pantalla los datos del diccionario (N° de Legajo, Apellido, Nombre, Comisión, Nota Primer Parcial, Nota Segundo Parcial, Condición) un renglón por cada alumno cargado.

En caso de elegir la opción resumida, la función debe mostrar en pantalla la cantidad total de alumnos, la cantidad de aprobados y el porcentaje de aprobados de cada comisión. Estos datos se deben presentar uno por cada renglón.

Ejercicio de CALC (Total 100 puntos se aprueba con 60%)

Nombre del Docente que corrige:_____

- Se debe tener un total de 60% para poder aprobar en CADA parte.
- Tiempo de desarrollo 1 hora 30 minutos.
- Dejar el Celular en Mudo y sobre la CPU.
- Ingresar en Linux y abrir la IDE que sea de preferencia del alumno.
- Recordar guardar en "DATOS" para evitar perdidas de código.
- Estas marcas son los check point que controla el docente al corregir. Se pide al alumno NO utilizar. El docente marcará con:
  - 

Nombre del Alumno:_____

Python_____

Una persona desea realizar un pequeño programa de ventas de una carniceria, y le pide a un estudiante de Ingenieria que le haga un script para hacer unas pruebas.

Se tiene declarada un diccionario **productos**, donde la clave es el identificador del producto, y como valores tiene una lista que contiene el nombre del corte y el precio por kilogramo.

A modo de ejemplo: productos={1:["vacío", 4321.9],2:["costilla",6784.56]}

Se debe plantear un script que permite presentar el siguiente menú (**con función menú**) :

1. Carga de una venta

2. Lista, Alta (agreagar o nuevo), B (Baja o borrar) o M (modificación) de los productos

3. Total de ventas.

f ó F Finaliza el programa

- función menú (10 puntos): no recibe nada, regresa solamente a la función principal : 1,2,3, s ó S. Cualquier valor de opción no válida muestra el cartel:"Opción NO válida" en la función menu y vuelve a mostrar el menú.

- script principal (10 puntos)

- 1. Carga de una venta (30 puntos): Si el usuario presiona "1. Carga una venta" llama a la **función vender** y pasa como argumento el diccionario productos. Dentro de la función se deben listar los productos, elegir uno de los mismos y luego ingresar el peso que se desea vender en Kg. (debe ser mayor que cero y menor a 1000), esto se hace para varios productos hasta que se ingresa un producto que no existe y allí se cierra la venta. Al cerrar esa venta, se guarda en una lista llamada **ventas** el monto total de esta venta, esta lista ventas se retorna al script principal. Recordar que cada vez que elija la opcion 1 puede vender varios productos con pesos distintos.

A modo de ejemplo: primera vez que se elige la opción 1 ventas=[2364.45], luego de la 2da vez que elige la opcion 1 ventas=[2364.45 , 123762,23]. El total acumulado de los ítems de la venta se almacena en la lista ventas que se regresa al script principal.

2. *Lista, Alta (agregar o nuevo), B (baja o borrar) o M (modificación) de los productos* (40 puntos): Si el usuario presiona 2, se llama a la **función labm,** que recibe como argumento el diccionario productos, muestra los elementos del diccionario por pantalla y muestra un menú *"A: Alta (agregar), B: Baja (borrar), M: modificar (precio por Kg)"* , realiza lo solicitado una vez y regresa True si se realiza con éxito la operación o False caso contrario. Recordar que el precio por Kilogramo DEBE ser positivo.

Nombre del Alumno:_____

Python (Parte 1)_____

Escribir un programa en Python que permita almacenar en un diccionario datos personas. Las claves del Diccionario serán los nombres de los alumnos y los valores para cada alumno que se almancenan en una lista son:

Edad (entero) , Peso (real) , Altura (real)

El programa presentará un menú invocando a la **función menú** que muestra:

1) Carga de Datos de una persona.

2) Calculo del promedio de altura de la personas.

3) Obtener el nombre de la persona mas pesada.

S o s) Salir del programa.

Función menú (10 puntos): No recibe argumento, y regresa SOLAMENTE 1,2,3,S o s .

Punto 1 (20 puntos): se invoca a la **función ingreso** que recibe como **argumento una clave válida** del diccionario. Desde el programa principal se debe pedir reingreso si la clave no existe, luego de ingresar una clave válida se envía esta como argumento a la función ingreso. En la función se cargan los valores de Edad (entero) , Peso (real) , Altura (real). Ver que todos los valores posibles SON POSITIVOS. En caso de no ser positivo el valor ingresado, **se debe solicitar el reingreso SOLO de ese valor.** La función ingreso deber retornar una variable booleana , que se utilizará como validación para habilitar las opciones 2, 3 y 4 del menú.

Punto 2 (30 puntos): se invoca a la **función promedio_altura** que **recibe como argumento el diccionario** y **retorna** al programa principal el promedio de altura de las presonas.

Tener en cuenta que si NO se cargó previamente (punto 1) NO se debe realizar la operación.

Punto 3 (30 puntos): Mostrar el nombre de la persona mas pesada usando invocando a la **función mas_pesado**, que recibe como argumento el Diccionario y regresa la clave que corresponde al nombre de la persona mas pesada.

Lógica, Consigna, General 10 puntos

Cantidad de funciones 4:

menú 10 puntos

ingreso 20 puntos

promedio_altura 30 puntos

mas_pesado 30 puntos

Calc (Parte 2)_____

Un mercado que cuenta con tres empleados quiere implementar un sistema para llevar las ventas de cada uno de

EXAMEN FINAL - REGULARES

NOTAS:
- Se debe tener total o parcialmente TODOS los temas para que el docente corrija el Examen.
- Se debe tener un total de 60% para poder aprobar en CADA uno de las tres partes.
- Tiempo de desarrollo 1 hora 30.

PARTE 1. Propuesto por Alicia.
El servicio de endocrinología de un hospital necesita de un programa para calcular el peso recomendado de una persona.
Escribir un programa en lenguaje Python que muestre el siguiente menú:

1.-Carga
2.- Búsqueda
3.- Listar IMC (Indice de masa corporal)
F-f Finalizar.

El menú se presenta con **función menú (10 puntos)**. Solamente saldrá del programa si ingresa f ó F. Los puntos 2,3 SOLO se pueden ejecutar si el Diccionario NO está vacío. Cuando el usuario elija alguna de las opciones el programa deberá llamar a la función correspondiente, o mostrar un cartel de **"Opción NO válida!"**

Si el usuario elige 1 (40 puntos) deberá llamar a una **función carga** que permita cargar un diccionario, donde cada clave es el apellido y nombre de un paciente y los valores son una lista con los siguientes datos : **altura** en metros [1 – 2.10] , **peso** en kg [30-200], **edad** [10,100]. Si ingresa un apellido y nombre que ya existe, debe solicitar reingreso, si ingresa un valor fuera del rango establecido (para :altura, peso o edad) debe reingresar solo el valor fuera de rango, no todos, solo el que está fuera de rango. La función deberá también calcular y guardar en la lista correspondiente a la clave, el cálculo del **peso recomendado** según la siguiente formula: **peso_recomendado= (altura en centímetros - 100 + 10% de la edad) *0.9**. La carga finaliza cuando se ingresa un valor de **altura negativo**.

Si el usuario elige 2 (20 puntos) deberá llamar a una **función rango_peso** que reciba como parámetro un rango de peso válido, **por ejemplo**: [50-60]. La función deberá buscar en el Diccionario todas aquellas personas que se encuentren en ese rango de peso y mostrar su nombre, edad y peso solamente, **si no existe**, deberá retornar False y desde la función principal se mostrará un mensaje: " **NO existen personas en ese rango**".
Si el diccionario está vació debería aparecer un cartel : **"No hay datos para Mostrar!".**

Si el usuario elige 3 (20 puntos) deberá llamar a una **función lista_imc** (listar índice de masa corporal) que calcula el imc y pero **muestra la categoría de peso según la tabla.** Se muestra los datos de cada par clave:valor y luego Obeso, Saludable según corresponda.

IMC	Categoría de peso

Escribir un programa en Python que permita almacenar productos con sus respectivas fechas de vencimiento mm aaaa

Debe presentar el siguiente menú:

1) Cargar (producto y la fecha de vencimiento)
2) Listar
3) N productos a vencerse
4) Buscar un producto
5) Finalizar

1) La funcion **cargar** ingresa el producto y las fechas de vencimiento, debe validar que mm = [1;12] y aaaa = [2000;2050]. Tambien debe validar si un producto ya se ha cargado.
2) La funcion **listar**, muestra todos los productos con sus respectivas fechas de vencimientos por pantalla.
3) La funcion **vencer** recibe como argumento un valor entero positivo ingresado por el usuario en el codigo principal, y busca los N primeros productos a vencerse
4) La funcion **buscar** recibe como argumento el nombre de un producto y muestra en pantalla el producto y la fecha de vencimiento. En caso de que no exista tal producto, debe retornar False al programa principal la cual deberá mostrar un cartel que diga "No existe el producto"
5) Finalizar el programa únicamente si se selecciona esta opción (Cualquier opción diferente, pedir, re-ingreso)

- Se debe tener total o parcialmente TODOS los temas para que el docente corrija el Examen.
- Para aprobar, se debe tener el 60% bien en CADA uno de las tres partes.
- **Tiempo de desarrollo 1 hora 30 minutos.**
- **Solo se puede tener abierto Calc, IDE de Python y Explorador de Archivos**. Si se observa que el alumno abre un navegador se considera que estaría copiando y finaliza su examen final con nota 1.
-**Guardar en Datos frecuentemente** por si existen cortes de energía.
-Por favor poner en el nombre de archivos .py y calc : **ApellidoNombre.py , ApellidoNombre.ods**
- Los check box ☐ son los puntos de control del docente son de uso del docente,se ruego no tachar los mismos.

Consiga de Python (60% para aprobar)

El alumno debe crear un script en Python que simule un sistema de gestión de contactos básico. El programa debe utilizar funciones, diccionarios y listas para almacenar y administrar la información de los contactos. ☐ Debe ofrecer un menú interactivo con diferentes opciones para que el usuario seleccione que quiere realizar, ☐ solo finaliza con S y ☐ debe indicar con un texto por pantalla si el ingreso es invalido, el menú se realizar invocando a la función menu, que no recibe nada y solo regresa 1,2,3,4,5 o S. Menú **(10 puntos)**

1. Agregar contacto **(30 puntos)**
2. Buscar contacto **(10 puntos)**
3. Alta de grupos **(10 puntos)**
4. Mostrar grupos **(20 puntos)**

S. Salir

Main script , lógica , etc **(20 puntos)**
☐ Definir un **diccionario vacío llamado agenda** para almacenar los contactos.
☐ Definir una lista llamada **grupos=["Trabajo"]**.
☐ Cada contacto se almacenará en el diccionario con los campos: **nombre** (clave del diccionario), **número** de teléfono y **grupo**.

☐ Los **nombres** propios se DEBEN formatear de manera que la primer letra sea mayúscula, se recomienda el método **.capitalize()**. Este Jorge es correcto, este jorge NO es correcto.
☐ El campo **número de Teléfono** debe ser numérico, si no lo es se debe solicitar reingreso.
☐ El campo **grupo**, se toma de los valores de una lista llamada grupos que **inicialmente tiene cargado solo un valor**. Esta lista puede tener por ejemplo "Familia", "Futbol", estos nombres se dan de alta con el punto 3 del menú.

El punto 1, permite agregar un contacto a la agenda llamando a la **función llamada agregar_contacto** que solicita al usuario ingresar ☐ **nombre,** ☐ **número de teléfono y** ☐ **grupo**, ☐ si el nombre ya existe en la agenda, se debe indicar con un cartel en la pantalla y presentar el menú nuevamente.
☐ Observación, el grupo se debe ser alguno de los cargados en la **lista grupos** y debe ser validado (si no existe pedir reingreso solo del grupo). ☐ La **función retorna una variable booleana** que se usar par evitar que no se ejecute el punto 2 y 4 sin haber cargado nada.

El punto 2, permite ☐ llamar a la **función buscar_contacto**, que recibe como argumento ☐ el nombre del contacto a buscar y ☐ muestra los datos: **nombre, teléfono, grupo**. ☐ Si no encuentra el contacto debe mostrar un cartel indicando que no hay coincidencias.

El punto 3, ☐ llama a la **función cargar_grupos**, esta función permite agregar a la lista grupos, lugares de entorno de los contactos, ☐ los elementos de esta lista no pueden repetirse. Ejemplo grupo=["Futbol", "Trabajo","Casa"]. ☐ Si se ingresa un valor repetido a la lista se debe solicitar reingreso son un cartel : "Ya existe ese grupo".

Punto 4, si el usuario selecciona esta opción, debe ingresar por teclado el nombre de un grupo de la lista **grupos** (☐ la lista **grupos** se debe mostrar por pantalla previo al pedido de carga) y validarlo (☐ si no se encuentra en la lista, debe pedir reingreso). ☐ Con este valor como argumento se invoca a la **función buscar_grupo**, la cual ☐ muestra los datos de todos los contactos que pertenezcan a ese grupo, por ejemplo si elige "Trabajo" muestra todos los datos de contacto de la agenda que son del entorno o grupo de trabajo.