GUÍA DE EJERCICIOS N°2

TEMA: **"Introducción a Python"**

Resolver los siguientes enunciados. Generar un archivo ".py" por cada ejercicio. De manera que se puedan ejecutar desde el intérprete de comandos de Linux (Mac.OS) o de Windows.
Por ejemplo: la resolución del ejercicio 3 de funciones en Python debería llamarse: eje1_3.py

1. **Funciones en Python**

1.1. Confeccionar una aplicación que muestre una presentación en pantalla del programa. Solicite la carga de dos valores y nos muestre la suma. Mostrar finalmente un mensaje de despedida del programa.
Implementar estas actividades en tres funciones.

1.2. Confeccionar una aplicación que solicite la carga de dos valores enteros y muestre su suma. Repetir la carga e impresión de la suma 5 veces. Mostrar una línea separadora después de cada vez que cargamos dos valores y su suma.

1.3. Confeccionar una función que reciba tres enteros y nos muestre el mayor de ellos. La carga de los valores hacerlo por teclado.

1.4. Desarrollar un programa que permita ingresar el lado de un cuadrado. Luego preguntar si quiere calcular y mostrar su perímetro o su superficie.

1.5. Confeccionar una función que le enviemos como parámetro un string y nos retorne la cantidad de caracteres que tiene. En el bloque principal solicitar la carga de dos nombres por teclado y llamar a la función dos veces. Imprimir en el bloque principal cual de las dos palabras tiene más caracteres.

1.6. Definir por asignación una lista de enteros en el bloque principal del programa. Elaborar tres funciones, la primera recibe la lista y retorna la suma de todos sus elementos, la segunda recibe la lista y retorna el mayor valor y la última recibe la lista y retorna el menor.

1.7. Confeccionar una función que cargue por teclado una lista de 5 enteros y la retorne. Una segunda función debe recibir una lista y mostrar todos los valores mayores a 10. Desde el bloque principal del programa llamar a ambas funciones.

1.8. Desarrollar un programa que permita cargar 5 nombres de personas y sus edades respectivas. Luego de realizar la carga por teclado de todos los datos imprimir los nombres de las personas mayores de edad (mayores o iguales a 18 años). Imprimir la edad promedio de las personas.

2. Tuplas y diccionarios en Python

2.1. Almacenar en una lista de 5 elementos tuplas que guarden el nombre de un pais y la cantidad de habitantes. Definir tres funciones, en la primera cargar la lista, en la segunda imprimirla y en la tercera mostrar el nombre del país con mayor cantidad de habitantes.

2.2. En el bloque principal del programa definir un diccionario que almacene los nombres de paises como clave y como valor la cantidad de habitantes. Implementar una función para mostrar cada clave y valor.

2.3. Crear un diccionario que permita almacenar 5 artículos, utilizar como clave el nombre de productos y como valor el precio del mismo.
Desarrollar además las funciones de:
1) Imprimir en forma completa el diccionario
2) Imprimir solo los artículos con precio superior a 100.

2.4. Desarrollar una aplicación que nos permita crear un diccionario ingles/castellano. La clave es la palabra en ingles y el valor es la palabra en castellano.
Crear las siguientes funciones:
1) Cargar el diccionario.
2) Listado completo del diccionario.
3) Ingresar por teclado una palabra en ingles y si existe en el diccionario mostrar su traducción.

2.5. Confeccionar un programa que permita cargar un código de producto como clave en un diccionario. Guardar para dicha clave el nombre del producto, su precio y cantidad en stock.
Implementar las siguientes actividades:
1) Carga de datos en el diccionario.
2) Listado completo de productos.
3) Consulta de un producto por su clave, mostrar el nombre, precio y stock.
4) Listado de todos los productos que tengan un stock con valor cero.

2.6. Confeccionar una agenda. Utilizar un diccionario cuya clave sea la fecha. Permitir almacenar distintas actividades para la misma fecha (se ingresa la hora y la actividad). Implementar las siguientes funciones:
1) Carga de datos en la agenda.
2) Listado completo de la agenda.
3) Consulta de una fecha.

2.7. Confeccionar una función que reciba una palabra y verifique si es capicúa (es decir que se lee igual de izquierda a derecha que de derecha a izquierda).

3. POO en Python

3.1. Implementar una clase llamada Alumno que tenga como atributos su nombre y su nota. Definir los métodos para inicializar sus atributos, imprimirlos y mostrar un mensaje si está regular (nota mayor o igual a 4)

3.2. Desarrollar un programa que cargue los lados de un triángulo e implemente los siguientes métodos: inicializar los atributos, imprimir el valor del lado mayor y otro método que muestre si es equilátero o no. El nombre de la clase llamarla Triangulo.

3.3. Confeccionar una clase que represente un empleado. Definir como atributos su nombre y su sueldo. En el método __init__ cargar los atributos por teclado y luego en otro método imprimir sus datos y por último uno que imprima un mensaje si debe pagar impuestos (si el sueldo supera a 3000).

3.4. Desarrollar una clase que represente un punto en el plano y tenga los siguientes métodos: inicializar los valores de x e y que llegan como parámetros, imprimir en que cuadrante se encuentra dicho punto (concepto matemático, primer cuadrante si x e y son positivas, si x<0 e y>0 segundo cuadrante, etc.).

3.5. Confeccionar una clase que administre una agenda personal. Se debe almacenar el nombre de la persona, teléfono y mail. Debe mostrar un menú con las siguientes opciones:
1- Carga de un contacto en la agenda.
2- Listado completo de la agenda.
3- Consulta ingresando el nombre de la persona.
4- Modificación de su teléfono y mail.
5- Finalizar programa.

3.6. Plantear un programa que permita **jugar a los dados**. Las reglas de juego son: se tiran tres dados si los tres salen con el mismo valor mostrar un mensaje que "gano", sino "perdió".

Pistas: Lo primero importar random. Despues lo que hacemos es identificar las clases: Dado y JuegoDeDados. Luego los atributos y los métodos de cada clase:

```
#Clase Dado :

 -atributos: valor

-métodos: tirar, imprimir, retornar_valor

#Clase JuegoDeDados:

-atributos:  3 Dado (3 objetos de la clase Dado)

-métodos: __init__, jugar
```

3.7. Supongamos que necesitamos implementar dos clases que llamaremos Suma y Resta. Cada clase tiene como atributo valor1, valor2 y resultado. Los métodos a definir son cargar1 (que inicializa el atributo valor1), carga2 (que inicializa el atributo valor2), operar (que en el caso de la clase "Suma" suma los dos atributos y en el caso de la clase "Resta" hace la diferencia entre valor1 y valor2), y otro método mostrar_resultado. Si analizamos ambas clases encontramos que muchos atributos y métodos son idénticos. En estos casos es bueno definir una clase padre que agrupe dichos atributos y responsabilidades comunes.