PRÁCTICO 2: PYTHON
Para cada ejercicio se pide:

a) **Cada ejercicio debe programarse en un proyecto.**
b) **Cada clase debe estar en un módulo separado, y el programa principal, que las invoca deberá importar todos los módulos que sean necesarios para resolver lo que solicite el enunciado.**
c) **Cada programa principal, deberá estar acompañado por una función de testing, que cree instancias de las clases con datos de prueba, que permita validar que las funcionalidades fueron testeadas en su totalidad, con datos correctos e incorrectos,**
d) **Dado que Python es un lenguaje de tipado dinámico, es imprescindible que cada vez que se reciba un parámetro formal en un método, se chequee el tipo de datos, para no hacer asignaciones incorrectas, por ejemplo, si se esperaba un string para inicializar un nombre de persona, que lo que se reciba pertenezca a la clase string.**

Ejercicio Nº 1

Defina una clase "Email" con los siguientes atributos: idCuenta, dominio, tipo de dominio y contraseña (todos estos datos se ingresan por teclado). Y los siguientes métodos:
a- El constructor.
b- Método "retornaEmail()" que construye una dirección e-mail con los valores de los atributos de Email. Por ejemplo:
idCuenta.: alumnopoo
dominio: gmail
tipoDominio: com
Resultado devuelto por retornaEmail: alumnopoo@gmail.com
c- Método "getDominio()", que retorna el dominio.
d- Método "crearCuenta(), crea una cuenta a partir de una dirección de correo que recibe como parámetro.

Implemente un programa que permita:

1- Ingresar el nombre de una persona y su dirección de e-mail (instancia de la clase Email) y luego imprima el siguiente mensaje:
Estimado <nombre> te enviaremos tus mensajes a la dirección <dirección de correo>.
2- Para la instancia creada en el ítem anterior, modificar la contraseña, teniendo en cuenta que se debe ingresar la contraseña actual, y ésta debe ser igual a la registrada en la instancia. Luego se debe ingresar la nueva contraseña y realizar la modificación correspondiente.
3- Crear un objeto de clase Email, a partir de una dirección de correo, como por ejemplo: informatica.fcefn@gmail.com, wicc2019@unsj-cuim.edu, juanLiendro1957@yahoo.com, etc.
4- Leer de un archivo separado por comas 10 direcciones de e-mail, ingresar un dominio e indicar cuántas de éstas tienen dominio igual al ingresado.

Ejercicio Nº 2 (Listas)

En una aerolínea se quiere implementar para sus viajeros frecuentes una promoción para acumular puntos de los viajes que realizan. Para ello, se necesita un sistema que sea capaz de gestionar el registro de viajeros y las millas. Defina una clase ViajeroFrecuente con los siguientes atributos: número de viajero, DNI, nombre, apellido y millas acumuladas. Para esta clase implemente los siguientes métodos:
a- El constructor.
b- "cantidadTotaldeMillas", retorna la cantidad de millas acumuladas.
c- "acumularMillas", recibe como parámetro la cantidad de millas recorridas en el último viaje.
d- "canjearMillas", recibe como parámetro la cantidad de millas a canjear. Para utilizar las millas debe verificarse que la cantidad que necesita sea menor o igual a la cantidad de millas acumuladas, caso contrario mostrar un cartel de error en la operación.

Implemente un programa que:

1- Leer de un archivo separado por comas 20 instancias de la clase ViajeroFrecuente, y almacenarlas en una lista.
2- Lea por teclado un número de viajero frecuente y presente un menú con las siguientes opciones:
a- Consultar Cantidad de Millas.
b- Acumular Millas.
c- Canjear Millas.
3- Represente el almacenamiento en memoria para la lista cargada con 4 viajeros.

Ejercicio Nº 3 (listas bidimensionales)
En una bodega se desea guardar la información de cada uno de los 20 camiones que utilizan para trasladar la producción durante la cosecha que transcurre en un lapso de 45 días corridos.
a- Defina la clase camión que posea como atributos: un identificador (un valor numérico de 1 a 20), nombre del conductor, patente del camión, marca del camión y tara (peso del camión vacío).
b- Defina una clase cosecha que contenga como atributo una lista bidimensional que permita registrar la cantidad de kilos por día que cada camión descarga en la bodega. El ingreso es por teclado, se ingresa identificador de camión, día, y el peso del camión cargado (el peso que informa la báscula de la bodega), para obtener los kilogramos de uva descargados se calcula la diferencia entre el peso del camión cargado y la tara. Un camión puede realizar varios viajes desde la finca a la bodega.
c- Implemente un programa que:
1- Lea los datos desde un archivo separado por comas y genere las instancias correspondientes de la clase “Camión”, almacenándolas en una lista.
2- Leer los datos de la cosecha desde un archivo separado por comas.
3- Presente un menú de opciones permita realizar las siguientes tareas:

3-1- Dado el número de identificador de un camión mostrar, la cantidad total de kilos descargados.
3-2- Dado un número correspondiente a un día mostrar un listado con el siguiente formato.

PATENTE	CONDUCTOR	CANTIDAD DE KILOS
********	**********	**********
*********	**********	**********

Ejercicio Nº 4 (Métodos y Constructores con valores por defecto)
a- Defina una clase FechaHora que registre: día, mes, año, hora, minutos y segundos, en el formato de 24 horas.
b- Implemente los métodos necesarios, para que pueda ejecutarse el Main que a continuación se propone.
En los métodos y constructor que se utilizan para fijar una determinada hora en el reloj deberá comprobarse que la fecha y hora son válidas.

```
if __name__ == '__main__':
    d=int(input("Ingrese Dia: "))
    mes=int(input("Ingrese Mes: "))
    a=int(input("Ingrese Año: "))
    h=int(input("Ingrese Hora: "))
    m=int(input("Ingrese Minutos: "))
    s=int(input("Ingrese Segundos: "))
    r = FechaHora () #  inicilizar día, mes, año con 1/1/2020, y hora, minutos y
                     #  segundos con 0h, 0m, 0s.
    r1 = FechaHora(d,mes,a); # inicializar con 0h 0m 0s
    r2= FechaHora(d,mes,a,h, m, s)
    r.Mostrar()
    r1.Mostrar()
    r2.Mostrar()
    input()
    r.PonerEnHora(5) # solamente la hora
```

```
r.Mostrar()
input()
r2.PonerEnHora(13,30) #hora y minutos
r2.Mostrar()
input()
r.PonerEnHora(14, 30, 25) #hora, minutos y segundos
r.Mostrar()
input()
r.AdelantarHora(3) #sumar 3 horas a la hora actual
r.Mostrar()
input()
r1.AdelantarHora(1,15) #sumar 1 hora y 15 minutos a la hora actual
r1.Mostrar()
input()
```

Nota: tener en cuenta que 60 segundos suman 1 minuto; 60 minutos suman 1hora; 24 horas suman 1 día. Las operaciones que adelanten el reloj, deben controlar:
a) Que la hora al llegar a 24, debe ponerse en 0 e incrementar un día.
b) Al incrementar un día, puede cambiar el mes.
c) Si el mes es diciembre, puede producirse el cambio de año.
d) El programa debe validar la cantidad de días de cada año.
e) El programa deberá validar años bisiestos (divisible por 4, y si es divisible por 100, debe ser divisible por 400), por ejemplo, 1900 es divisible por 4, pero no fue bisiesto porque no fue divisible por 400.

Ejercicio Nº 5 (Datos miembro estáticos y Funciones miembro estáticas)

Defina una clase “Alumno” que represente un alumno de escuela, y registre los siguientes datos: nombre del alumno, año y división a la que asiste, cantidad de inasistencias, cantidad máxima de inasistencias permitidas y cantidad total de clases (los últimos dos atributos son iguales para todos los alumnos de la escuela). Implemente un programa que:
1- Genere una lista que permita almacenar instancias de la clase “Alumno”. Los datos de los alumnos se leen desde un archivo separado por comas.
2- Presente un menú de opciones permita:
a- Ingresar un año y división, y liste nombre y porcentaje de inasistencias de los alumnos cuya cantidad de inasistencias supera la cantidad máxima de inasistencias permitidas. El listado debe tener el siguiente formato:

Alumno	Porcentaje
*******	**%
*******	**%

b- Modificar la cantidad máxima de inasistencias permitidas.

Ejercicio Nº 6 (Sobrecarga de operadores)

Dada la clase FechaHora, del ejercicio 4, presente un menú de opciones que permita lo siguiente:

1- Sumar hora, para ello sobrecargue el operador binario suma (+).
2- Restar hora, para ello sobrecargue el operador binario resta (-).
3- Distinguir entre dos horas cuál es mayor, para ello sobrecargue el operador relacional mayor (>).

Tener en cuenta la nota del ejercicio 4

Ejercicio Nº 7 (Sobrecarga por derecha)

a- Defina una clase Hora que registre: hora, minutos y segundos, en el formato de 24 horas.

b- Escriba los métodos necesarios, para que pueda ejecutarse el Main que a continuación se propone.

```
if __name__ == '__main__':
    d = int(input("Ingrese Dia: "))
    mes = int(input("Ingrese Mes: "))
    a = int(input("Ingrese Año: "))
    h = int(input("Ingrese Hora: "))
    m = int(input("Ingrese Minutos: "))
    s = int(input("Ingrese Segundos: "))
    f = FechaHora(d,mes,a,h, m, s)
    f.Mostrar()
    input()
    h1 = int(input("Ingrese Hora: "))
    m1 = int(input("Ingrese Minutos: "))
    s1 = int(input("Ingrese Segundos: "))
    r = Hora(h, m, s)
    r.Mostrar()
    input()
    f2 = f +r
    f2.Mostrar()
    input()
    f3 = r + f
    f3.Mostrar()
    input()
    f4 = f3 - 1 # Al restar un número entero a un objeto FechaHora se debe restar la cantidad de
                # días indicada por el número entero
    f4.Mostrar()
    f4 = 1 + f2 # suma un día a un objeto FechaHora
    input()
```

Tener en cuenta la nota del ejercicio 4

Ejercicio Nº 8 (Sobrecarga de operadores)

Defina una clase “Conjunto” que represente un conjunto matemático de números enteros. Implemente un programa que presente un menú de opciones que permita lo siguiente:
1- La unión de dos conjuntos, para ello sobrecargue el operador binario suma (+).
2- La diferencia de dos conjuntos, para ello sobrecargue el operador binario resta (-).
3- Verificar si dos conjuntos son iguales, para ello sobrecargue el operador “==” teniendo en cuenta que dos conjuntos se consideran iguales si tienen la misma cantidad de elementos y sus valores son iguales (sin importar el orden de los elementos).