EJERCICIO 1

Transformar el siguiente código de manera que no sea necesario usar la estructura **if**:

```
opcion = input("¿Que tipo de comida quieres?");

if opcion.lower()=="fruta":
    res = "Manzana, platano o pera";
elif opcion.lower()=="verdura":
    res = "Tomate, lechuga o zanahoria";
elif opcion.lower()=="carne":
    res = "Cerdo, ternera o pollo";
elif opcion.lower()=="pescado":
    res = "Sardinas, caballa ó salmonete";
else:
    res = "No tenemos de ese tipo de comida";

print(res)
```

EJERCICIO 2

Crear un código Python que transforme de forma equivalente el código que aparece en la captura eliminando la estructura condicional sustituyéndolo por llamadas a funciones:

```
opcion=input("¿Quieres mostrar,modificar o longitud?")
lista=[4,8,14,22]

if opcion=="mostrar":
    for i in range(0,len(lista)):
        print("Posicion",i,":",lista[i])
elif opcion=="modificar":
    for i in range(0,len(lista)):
        lista[i]*=2
    print(lista)
elif opcion=="longitud":
    print(len(lista))
else:
    print("Error")
```

EJERCICIO 3

Hacer un programa que no utilice estructuras condicionales que permita convertir un número a octal, a binario o a hexadecimal según se quiera. (Mirar la funciones hex, bin y oct, solo la parte numérica sin prefijos 0x, 0b 0o)

EJERCICIO 4

Partiendo del siguiente diccionario:

```
persona = {
    "nombre": "daniel garcía peraltes",
    "edad": 29,
    "peso": 87,
    "altura": 183
}
```

Conseguir el siguiente resultado por pantalla mediante un bucle y sin usar if:

```
La persona: DANIEL GARCÍA PERALTES
29 años
87 Kilos
183 centimetros
```

EJERCICIO 5

Partiendo del siguiente diccionario modificarlo mediante un bucle y sin usar if para transformar cada valor string a su tipo de datos correspondiente:

```
#ANTES
producto={
    "nombre":"Solomillo a la pimienta verde",
    "precio":"18.95798",
    "calorias":"1050",
    "vegano":"False",
}
```

```
#DESPUES
producto={
    "nombre":"Solomillo a la pimienta verde",
    "precio":18.96,
    "calorias":1050,
    "vegano":False,
}
```

EJERCICIO 6

Partiendo del fichero CSV llamado **juegos.csv** que tiene un formato como el siguiente:

	A	B	C
1	Juego	Precio	Genero
2	Super Mario E	30.0	Plataformas
3	Silent Hill	12.0	Terror
4	Resident Evil	45.0	Terror
5	Halo 5	20.0	Shooter
6	Final Fantasy '	19.90	JRPG
7	PES 2021	9.99	Deportivo
8	FIFA 2022	59.99	Deportivo

Leer su contenido y trasladarlo una estructura Python tipo lista de diccionarios. El fichero está disponible a través de la siguiente URL de internet:

https://firebasestorage.googleapis.com/v0/b/chat-7d403.appspot.com/o/juegos.csv?alt=media&token=17762748-ca38-45b7-a5de-3caea749de59

EJERCICIO 7

Repetir el ejercicio anterior, pero trasladarlo a un diccionario de diccionarios donde la clave del diccionario principal sea el nombre del juego.

EJERCICIO 8

Partiendo de los resultados del **EJERCICIO 6 y DEL EJERCICIO 7** guárdalo en un fichero llamado juegos2.csv con formato similar al de su origen.

EJERCICIO 9

Partiendo del fichero CSV llamado **agenda.csv** que se encuentra en la siguiente URL:

https://firebasestorage.googleapis.com/v0/b/chat-7d403.appspot.com/o/agenda.csv?alt=media&token=166c52a7-08d8-49e7-af6b-9f5d9cef0e1f

Tiene un formato como el siguiente:

	A	B	C	D	E	F
1	DNI	NOMBRE	APELLIDOS	AÑO	PUESTO	SUELDO
2	12345678D	Juan	Rodriguez Pc	2001	Camarero	1200.0
3	98765432F	Maria	Lopez Serran	2003	Profesora	1500
4	31543265G	Carmen	Fernandez Sa	1996	Actriz	2000
5	98812334V	Alfonso	Ribera Salva	1999	Camionero	789
6	54376574B	Carlos	Saez Aguilar	2010	Vendedor	1500.0

Leer su contenido y trasladarlo una estructura Python como la siguiente:

```
[{“DNI”:”12345678D”,
”NOMBRE”:”Juan”,
”APELLIDOS:”Rodriguez Polo”,
”AÑO”:2001,
”PUESTO”:”Camarero”,
”SUELDO”:1200.0},…]
```

EJERCICIO 10

Repetir el ejercicio anterior, pero trasladarlo a una estructura como la siguiente:

```
{“12345678D”:{“DNI”:”12345678D”,
                ”NOMBRE”:”Juan”,
                  ”APELLIDOS:”Rodriguez Polo”,
                  ”AÑO”:2001,
                  ”PUESTO”:”Camarero”,
                  ”SUELDO”:1200.0},
 “98765432F”:{“DNI”:”98765432F”,
                  “NOMBRE”:”Juan”,
                  “AÑO”:….
```

EJERCICIO 11

Partiendo de los resultados del **EJERCICIO 9** guárdalo en un fichero llamado **agenda2.csv** con similar formato e intenta abrirlo con Excel o similar. Haz lo mismo con el **EJERCICIO 10.**

EJERCICIO 12

Crear una **clase Coche**, a través de la cual se puedan crear objetos que almacenen el color del coche, la marca, el modelo, el número de caballos, el número de puertas y la matrícula. Crear el metodo __init__, el método __str__. Crear también el método __eq__ que compare cuando dos objetos son iguales **(tienen la misma matricula)**.

EJERCICIO 13

Crea una clase llamada **Libro** que guarde la información acerca de un libro de una biblioteca. La clase debe guardar el título del libro, ISBN, autor, número de ejemplares totales del libro y número de ejemplares prestados. La clase contendrá los siguientes métodos:

- Constructor__init__.
- Método __str__.
- Método __eq__ donde dos libros son iguales si tienen el mismo ISBN.
- Método llamado ***préstamo*** que incremente el atributo correspondiente cada vez que se realice un préstamo del libro. No se podrán prestar libros de los que no queden ejemplares disponibles para prestar.
- Método ***devolución*** que produce la devolución de un libro. Si no se ha prestado ningún libro no se puede devolver.

EJERCICIO 14

Crear la clase **Cerveza** que tiene los siguientes atributos:

- Código de identificación (alfanumérico)
- Nombre.
- Tipo (rubia, tostada, roja y negra).
- Elaboración artesanal o no.
- Precio.
- Existencias

Además de los siguientes métodos:

- Constructor con todos los parámetros menos el id que tiene el prefijo CERV-X , donde X es un número que se controla desde la clase y que se incrementa en 1 cada vez que se crea una instancia de la clase Cerveza.
- **servir_cerveza(int)**. Resta de las existencias y si no hay suficientes sirve las que pueda.
- **reponer_cerveza(int)**. Añade existencias.
- Método __**eq**__ donde una cerveza es igual a otra si coinciden los códigos identificativos.
- Método __**str**__.

EJERCICIO 15

Crear la clase **Joya** que tiene los siguientes atributos:

- Código de identificación (alfanumérico)
- Nombre.
- Marca.
- Tipo (collar, anillo, brazalete, diadema o broche).

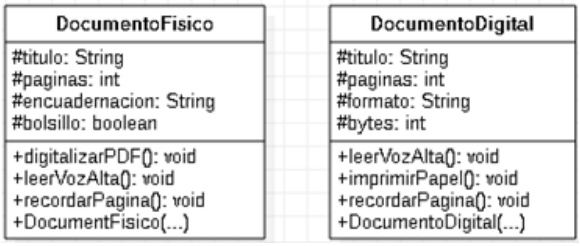
- Precio.
- Peso en gramos.
- Quilates.

Además de los siguientes métodos:

- Constructor con todos los parámetros menos el id que tiene el prefijo JOYA-X , donde X es un número que se controla desde la clase y que se incrementa en 1 cada vez que se crea una instancia de la clase Joya.
- Método __eq__ donde una joya es igual a otra si coinciden los códigos identificativos.
- Método __str__.
- Método **pureza()** que devuelve un número con la pureza de oro en función del peso y los quilates (1quilate=aprox 4,167% de oro)

EJERCICIO 16

Dado el siguiente diagrama de clases, **escribir clases de código Python refactorizando** para simplificar su mantenimiento **usando herencia**:



- **En las clases incluir únicamente** los atributos y métodos descritos en el diagrama con las superclases y subclases necesarias.
- El **constructor** además del método __str__ y __eq__ (dos documentos son iguales si tienen el mismo título).
- **Los métodos void digitalizarPDF, leerVozAlta, imprimir y recordarPagina** solo sacan por pantalla un mensaje con la acción que realizan.

EJERCICIO 17

Crear la aplicación para la gestión de empleados de una empresa que está compuesta por varias clases:

- La clase **Empleado** tiene cuatro atributos**: nombre, dni** como cadena de texto, **sueldo_base y años en la empresa**. Implementar el método **__init__, y __eq__** (mismo dni mismo empleado), además del método **sueldo_neto** que devuelve un float con el sueldo deduciéndole un 15% de impuestos. Debe implementarse también un método **__str__** que debe mostrar todos los atributos y el **sueldoFinal** debe aparecer dentro del string que devuelve el **__str__**.

- La clase **Vendedor** que hereda de **Empleado** y tiene como atributos propios las ventas totales realizadas por el vendedor y la comisión que se lleva por venta realizada. Hay que redefinir el método __**str**__ con los nuevos datos y también **sueldo_neto** añadiendo la comisión total por ventas realizadas al cálculo que realiza la clase **Empleado**.
- Por último, la clase **Vendedor** tiene un método llamado **vender** que lanzar un mensaje por pantalla informativo.
- La **clase Empresa** que tiene 2 atributos que son el nombre de la empresa y una lista que está inicialmente vacía. El nombre de la empresa se le pasa al constructor.
- La clase **Empresa** tendrá un método para añadir objetos de las clases anteriores a la lista, llamado **contratar_trabajador**.
- Por último, la clase **Empresa** tendrá un método llamado **imprimir_resumen** que recorre la lista completamente y para cada objeto **(**que puede ser **Empleado o Vendedor)** hace un print de lo que devuelve el método __**str**__ correspondiente, donde además en caso de ser un objeto de la clase **Vendedor** ejecutar también el método vender.

EJERCICIO 18

Crear la clase **Producto** que tiene los atributos nombre, identificador (código alfanumérico) y precio. Tiene que tener constructor, __**str**__ **y** __**eq**__**,** donde un producto es igual a otro cuando tienen el mismo identificador. También tendrá el método **comprar** que tiene como parámetro las unidades a llevarse. El método **comprar** devuelve un **float** con el precio multiplicado por la unidades que se quiere llevar. Si al comprar se lleva más de **50 unidades** el precio final se reduce a la mitad.

Crear la clase **Perecedero** que hereda de **Producto**. Se añade un atributo más que es los días para **caducar**. Hay que hacer el método __**str**__. El método **comprar** es similar al de producto solo que le reduce el precio calculado en la clase **Producto** según los días que le queden para caducar:

- Si le queda un día para caducar, el precio final es la cuarta parte del que devuelve comprar de la clase **Producto**.
- Si le quedan 2 días para caducar, el precio final es la tercera parte.
- Si le quedan 3 días para caducar, el precio final es la mitad.
- Si le quedan 4 o más días no se modifica el precio original.

- Crear la **clase Tienda** como se describe a continuación:
  - La clase tiene 1 atributo que es una lista que está inicialmente vacía.
  - La clase tendrá un método para añadir objetos de las clases anteriores llamado **nuevo_producto** a la lista anteriormente mencionada. El método recibe objetos de la clase Producto y Perecedero.

- La clase Tienda también tendrá un método __**str**__ uniendo todos los datos de todos los productos que haya en la lista (**Producto o Perecedero)**, llamando al método __**str**__ correspondiente.

EJERCICIO 19

Partiendo de la siguiente API en formato JSON:

https://api.coindesk.com/v1/bpi/currentprice.json

De donde se obtienen las cotizaciones actuales del bitcoin en DOLARES, EUROS Y LIBRAS. Hacer scripts de Python que muestren lo siguiente:

- Mostar la descripción, el valor de divisa y el símbolo en cada una de las divisas, además de la fecha de última actualización del bitcoin incluido en el json
- Pedir al usuario que introduzca DOLARES, EUROS Y LIBRAS y se le muestre el valor en la divisa elegida.

EJERCICIO 20

Partiendo de la carpeta de la API OMDB realice una búsqueda completando el parámetro **s** pidiendo datos al usuario.

- La información a mostrar es el título de la película, el año y el URL del poster.
- Hay que mostrar los resultados de más a menos reciente.
- Por ejemplo una petición mediante esta URL nos mostraria las pelicula cuyo titulo incluya indiana **http://www.omdbapi.com/?s=indiana&apikey=b76b385c&page=1&type=movie&Content-Type=application/json**.
- Apikey es un campo importante ya que sin él no se permite el acceso a la API.

EJERCICIO 21

Regístrate en la web de https://currencylayer.com/ para obtener acceso al conversor de divisar y realiza una aplicación que pida una divisa origen, otra destino y una cantidad, obteniendo el resultado de la conversión y lo muestre por pantalla. El registro es necesario para obtener una API Key propia. El programa debe usar el **endpoint** siguiente:

https://api.apilayer.com/currency_data/convert?to=destino&from=origen&amount=dinero&apikey=TUAPIKEY