EXAMEN DE PYTHON

Ejercicio 1 [2 puntos]

Aquí tienes una tabla que muestra cuánto vale 1€ en varias monedas:

Dólar	1.42
Libra	0.87
Yen	113.86
Peseta	166.386

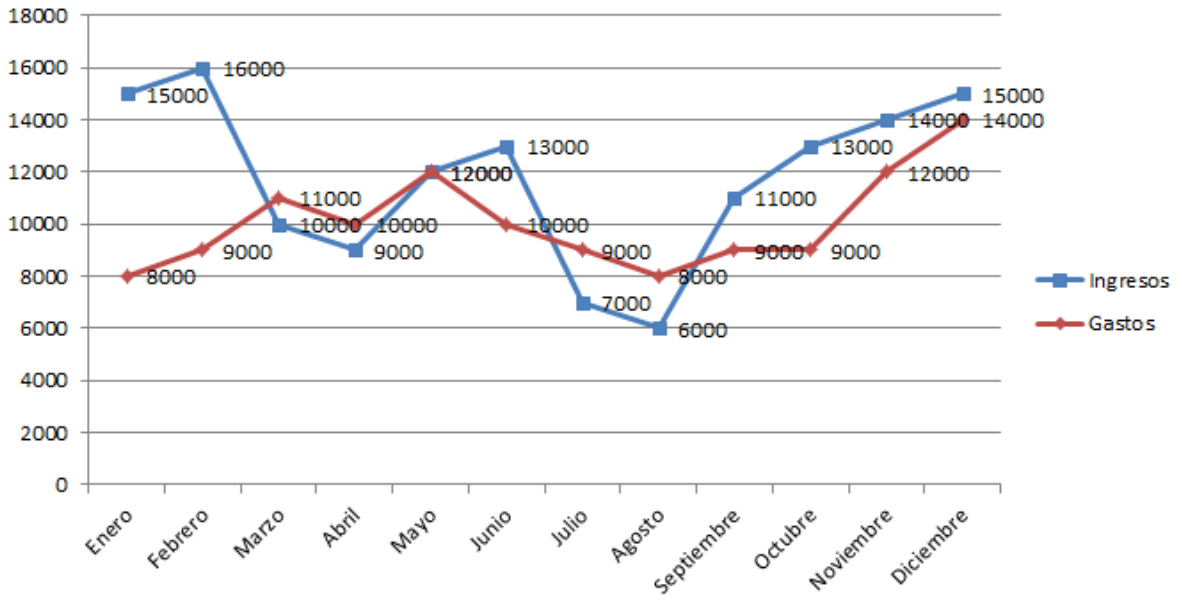
Haz un programa que pregunte por teclado el nombre de una moneda y cuántos euros quieres convertir a esa moneda. Si la cantidad no es un número, o es un número negativo deberá volver a preguntarla hasta que sea un número correcto. Si la moneda no está en la tabla, se mostrará “La moneda (*nombre de la moneda*) no está admitida”. Si la moneda está admitida, se mostrará el resultado de la conversión de euros a esa moneda.

Puedes hacer el ejercicio como quieras, pero para obtener la máxima puntuación:

- Usa un diccionario para almacenar la tabla.
- Haz una función que pida por teclado al usuario una cantidad de dinero. Si el número introducido es incorrecto o negativo, lo vuelve a pedir y así sucesivamente hasta que el número es correcto. La función devolverá el número introducido correctamente.
- Haz una función que reciba como parámetro una cantidad en euros y el nombre de una moneda. La función usará la tabla para calcular y devolver la cantidad de dinero correspondiente en la moneda. Si la moneda no está en la tabla lanzará un ValueError con el mensaje “La moneda (*nombre de la moneda*) no está admitida.
- Usa type hints en las funciones creadas

Ejercicio 2 [2 puntos]

La siguiente gráfica recoge los ingresos y gastos de una empresa en un año:



Haz un programa que cree dos listas para guardar los ingresos y los gastos de la empresa. A continuación, se mostrará en pantalla:

- Si cada mes ha tenido un balance positivo o negativo
- La media de ingresos y gastos anual
- El balance final anual y si ha sido positivo o no.

Puedes hacer el ejercicio como quieras, pero para obtener la máxima puntuación:

- Saca tuplas con los datos de las dos listas a la vez
- Programa funciones que te permitan reutilizar código
- Usa type hints en las funciones creadas

Ejercicio 3 [1.5 puntos]

Haz un programa que pregunte por teclado al usuario un día, mes y año y, en caso de que se trate de una fecha correcta, el programa mostrará el nombre del día de la semana (en español) que corresponde a esa fecha. Deberás utilizar la clase **date** que se encuentra en la librería **datetime** (esta librería ya viene con Python y solo hay que importarla):

date
+ date(año,mes,día) + int weekday()

- constructor: crea un objeto de la clase **date** que representa el día del año cuyo día, mes y año se pasan como parámetros. En caso de que la fecha sea incorrecta, se lanza un ValueError.
- weekday: devuelve un número con el día de la semana, de forma que el 0 es el lunes y el domingo es el 6

Puedes hacer el ejercicio como quieras, pero para obtener la máxima puntuación:

- Haz una función que reciba como parámetros un número de día, otro de mes y otro de año y nos devuelva el número del día de la semana que le corresponde a esa fecha en el calendario (0 = lunes, 6 = domingo). En caso de que la fecha sea incorrecta, se deberá lanzar un ValueError
- Haz una función que reciba como parámetro un número entre 0 y 6 y nos devuelva, en español, el nombre del día de la semana que le corresponde
- Usa type hints en las funciones creadas

Ejercicio 4 (2 puntos)

Programa la siguiente clase: (__ quiere decir dos guiones bajos. Los métodos que sale subrayado son “métodos estáticos”)

Cuenta_Corriente
+ __numero + __saldo
+ Cuenta_Corriente(numero,saldo) + add_dinero(cantidad) + retirar_dinero(cantidad) + hacer_transferencia(cuenta1,cuenta2,cantidad) + nueva_cuenta()

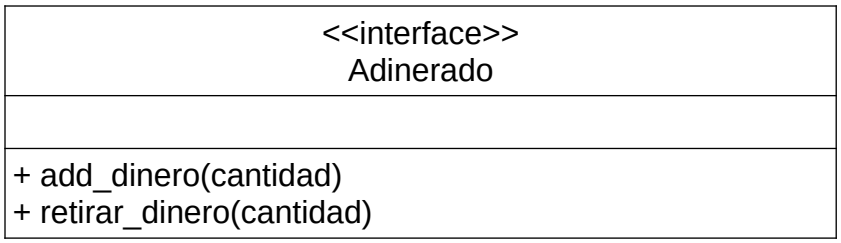
- La cuenta corriente tiene un número de cuenta y un saldo.
- El constructor inicializa ambas propiedades, pero si alguno de los dos números es negativo lanza un **ValueError** con un mensaje apropiado.
- **add_dinero** le añade la cantidad de dinero pasada como parámetro a la cuenta
- **retirar_dinero** retira de la cuenta la cantidad de dinero pasada como parámetro. Si la cantidad que se quiere retirar es negativa o mayor que el saldo, se lanzará un **ValueError** con un mensaje apropiado.
- **hacer_transferencia** transfiere la cantidad pasada como tercer parámetro de la primera cuenta a la segunda, pero solo si en la primera cuenta hay dinero suficiente. El método devuelve true si la transferencia se hace correctamente, y false si no se puede hacer.
- **nueva_cuenta:** Este método devuelve una nueva cuenta cuyo número es aleatorio entre 1000 y 9999 y su saldo es 0. Para sacar un número aleatorio usa la siguiente función:
    - librería: **random** (viene ya incluida en Python y solo hay que importarla)
    - función: **randint(inicio,fin)**
        - *Esta función devuelve un número aleatorio entre inicio (incluido) y fin (no incluido)*

Para obtener la máxima puntuación:

- Añade type hints a los métodos

Ejercicio 5 [0.5 puntos]

Programa la siguiente interfaz, y haz que la clase Cuenta_Corriente del ejercicio anterior la implemente.



Para obtener la máxima puntuación:

- Añade type hints a los métodos

Ejercicio 6 [2 puntos]

Crea un diccionario que almacene esta información sobre los clientes de un banco (debes usar la clase del ejercicio 4):

Nombre del cliente	Cuentas corrientes de ese cliente
Pedro López	n.º cuenta 1234, saldo 5000 n.º cuenta 5678, saldo 1200
Luisa Pérez	n.º cuenta 4492, saldo 800 n.º cuenta 2112, saldo 9000
Antonio González	n.º cuenta 1251, saldo 6000
John Pérez	n.º cuenta 7212, saldo 5200 n.º cuenta 4442, saldo 8900 n.º cuenta 9878, saldo 6000

Realiza un programa que pregunte el nombre de un cliente y nos muestre:

- Si el cliente está o no en el banco
- Cuánto dinero tiene el cliente
- Cuántas cuentas corrientes tiene el cliente.

Puedes hacer el ejercicio como quieras, pero para obtener la máxima puntuación:

- Crea funciones (con type hints) que organicen el programa en tareas independientes y lo hagan más legible.