Trabajo Práctico 2 - NumPy

1. Definir un script que pida al usuario ingresar 4 valores $a,\ b,\ c,\ d$. Generar con éstos dos vectores $\vec{u} = (a, b)$ y $\vec{v} = (c, d)$. El script debe imprimir una matriz con los resultados de las operaciones:
   a. $\vec{v} - \vec{u}$
   b. $-\vec{v} - \vec{u}$
   c. $2\vec{u} - \vec{v}$
   d. $\frac{1}{2}\vec{u} - 2\vec{v}$

2. Una tienda de electrodomésticos escribía a mano en el archivo de texto "stock.txt" los productos que vende, junto con el respectivo precio y stock restante. Quieren guardar esta información en bloques de datos más seguros.
   Implementar un script que lea los *n* datos el archivo y genere tres arreglos unidimensionales con los tres tipos de datos del mismo.
   Ayuda: Reorganizarlos en una matriz de tres columnas y *n* filas, y luego dividirla en tres vectores y finalmente arreglar la inconsistencia de tipos.

A partir del inciso anterior:

3. Escribir un script que pida un string al usuario con el nombre de un producto, busqué si se encuentra en el vector, y en cuyo caso imprima su precio y cantidad disponible.
4. Escribir una función `comprar(producto)` que reciba un string con el nombre de un producto, verifique si existe o hay disponibilidad, y en dicho caso disminuya la cantidad disponible y retornará *Verdadero*, en cualquier otro caso deberá devolver *Falso*.
5. Escribir una función `Agregar(producto, precio, cantidad)` y `Reponer(producto, cantidad)`. La primera añadirá un nuevo elemento a los tres vectores (en caso de que no exista en la misma), la segunda aumentará la cantidad disponible de dicho producto, en caso de que el mismo exista. Ambas retornarán *Verdadero* o *Falso* según corresponda.

6. Se desea implementar un sistema de lotería virtual. A continuación se detallan los pasos a seguir para desarrollar un prototipo del proyecto. Se detallan los pasos a seguir acompañado de observaciones a tener en cuenta para cuando deban preparar sus propios proyectos.

   a. Primero generar la combinación del billete ganador en un vector. El billete deberá contener cuatro (4) números aleatorios en el rango [0 ; 5].
   Obs: El rango de los números aleatorios es muy acotado. La razón de esto se explicará en el **inciso c.**

b. Debemos generar los billetes de los participantes. Crearemos una matriz a cuyas filas asignaremos 10 billetes aleatorios (con las mismas restricciones del billete ganador).
Obs: Reutilizar el código de generación del billete ganador para generar las filas. Puede incluso crear una función que lo contenga y devuelva un billete aleatorio cada vez que se le invoca. La reutilización y modificación de código que ya poseemos es un aspecto importante a la hora de preparar un proyecto.
Obs 2: Al final de la consigna hay dos ideas comunes para esto. Primero implementar su propia idea y luego leer las sugerencias para ganar una nueva perspectiva.

c. Los ganadores serán aquellos que posean coincidencias del **mismo número** en la **misma posición** que el billete ganador. Escribir un código que devuelva un arreglo con la cantidad de coincidencias de cada participante.
Ejemplo:
Billete ganador: [3 2 0 5 4 3 1]
Billete participante 1: [2 4 1 5 2 3 1] 1 coincidencias.
Billete participante 2: [1 5 1 4 3 1 3] 0 coincidencias.
Billete participante 2: [4 1 0 5 2 3 0] 3 coincidencias.
Arreglo devuelto: [1 0 3]
Obs: La razón de elegir un rango de números aleatorios pequeño en el **inciso a** es para poder comprobar las coincidencias. Si eligiéramos el rango de [0 ; 99] sería difícil que en esta pequeña prueba tuviéramos muchas coincidencias, sin embargo al elegir un rango más pequeño seguramente podremos comprobar el funcionamiento de nuestro programa. Al verificar que el funcionamiento es correcto podremos simplemente cambiar los parámetros del código para tener rango aleatorio más grande, distinta cantidad de números por billete y cantidad de participantes más grande.

d. Opcional 1: Luego de tener el programa listo, crear una función que tome como parámetros cantidad de números por billete, rango de números aleatorios, y cantidad de participantes. La función debe devolver por un lado el billete ganador y por otro la matriz con los billetes de los participantes.

e. Opcional 2: Crear una función que tome como parámetros el billete ganador y la matriz de billetes de participantes, y devuelva el número del participante que tenga el billete ganador (en caso que lo haya), o **[-1]** si ningún participante ganó. Ejemplo de encabezado:
```
Comprobar_ganador (billete_ganador, participantes)
```

Sugerencia: Para comprobar, pueden “sembrar” la combinación ganadora, es decir, reemplazar alguna fila de los participantes por la combinación del ticket ganador. De esta forma se aseguran que exista la combinación y la función devuelva un resultado.

IDEAS PARA LA GENERACIÓN DE BILLETES DE PARTICIPANTES:
Reiteramos que, si bien estas ideas son simples, lo mejor es intentar resolver el ejercicio antes de leerlas.
La primera idea es generar una matriz e ir recorriendo cada fila asignando un billete aleatorio con la función que habremos generado del inciso 1.
La segunda idea es crear primero un vector de boleto para el primer participante. Luego generar un nuevo vector para el participante 2 y **concatenar verticalmente** con el vector del participante anterior. Esto nos dará una matriz de profundidad 2 (por ser dos participantes). Luego generar el vector del tercer participante y **concatenar verticalmente** con la matriz anterior. El resultado será una matriz de profundidad 3 con cada fila conteniendo un vector. Repetir hasta finalizar la cantidad solicitada.

7. En toda comunicación digital los datos viajan en formato **binario**, es decir, cadenas de unos y ceros. Para poder interpretar esta información los datos se envían en "paquetes", es decir, grupos de tamaño finito.
   A continuación se presenta la señal recibida en una comunicación digital. Los paquetes respetan el formato ASCII, por lo tanto cada paquete es de 8 bits (cada 8 números se tiene un carácter).
   Ejemplo: El paquete '**01000001'** corresponde al carácter "A", mientras que el **'010101010'** es el carácter 'a'. La señal :'**01000001010101010'** es una señal de dos paquetes que contiene el mensaje "Aa".

   La comunicación que debe descifrar contiene nueve (9) paquetes de ocho (8) bits cada uno. Separar cada uno de los paquetes y mostrar el mensaje descifrado.
   Para convertir cada paquete en su carácter ASCII usar la siguiente expresión:
   ```
   chr(int('010101010',2))
   ```
   Donde la secuencia insertada entre comillas es la combinación ASCII del carácter.
   Ejemplo de uso:
   ```
   x = chr(int('010101010',2))
   print(x)
   ```

   Señal a decodificar:
   010010000110111101101100011000010100110101110101011011100110010001101111

   Tabla de código ASCII básico (de referencia): http://sticksandstones.kstrom.com/appen.html