1. Cree una función llamada compara que retorne Verdadero si los elementos de dos arreglos pasados como parámetros son iguales, Falso de lo contrario. Un ejemplo de llamada es:

```
import numpy as np

a = [1, 2, 3, 4]
b = [1, 2, 3, 4]
c = [1, 2, 3, 5]

print(compara(a, b)) # True
print(compara(a, c)) # False
```

Una solución posible:

```
def compara(a, b):
  return np.all(a==b)
```

2. El siguiente código se usa para clasificar a personas según su índice de masa corporal (IMC).

```
import numpy as np

altura = [5.23, 7.17, 5.45, 5.98, 6.02]
peso = [135, 164, 137, 156, 161]

print(imc(altura, peso))
```

Implemente la función imc que recibe como parámetros una lista de alturas y otro de peso y retorna el índice de masa corporal para cada una de las personas. El imc se calcula como (peso en kilos)/(altura_en_metros^2).

Una solución posible:

```
def imc(altura, peso, sistema='metrico'):
  np_altura_m = np.array(altura)
  np_peso_kg = np.array(peso)

  indice = np_peso_kg / (np_altura_m ** 2)
  return indice
```

3. Al usar el código de la pregunta anterior, un nutricionista se dio cuenta que tenía datos tanto en el sistema métrico como en el imperial (como en el ejemplo). Dado eso, se les pide que agreguen un parámetro más a la función que diga que tipo de sistema de medición se usa, y si los datos están en el sistema imperial entonces los transforme a métrico. La altura se transforma multiplicando por 0.254 y el peso multiplicando por 0.453592. El código del ejemplo anterior quedaría de la siguiente forma:

```
import numpy as np

altura = [5.23, 7.17, 5.45, 5.98, 6.02]
peso = [135, 164, 137, 156, 161]

print(imc(altura, peso, 'imperial'))
```

Una solución posible:

```
def imc(altura, peso, sistema='metrico'):
  np_altura_m = np.array(altura)
  np_peso_kg = np.array(peso)
  if 'imperial' == sistema:
       np_altura_m = np_altura_m * 0.254
       np_peso_kg = np_peso_kg * 0.453592

  indice = np_peso_kg / (np_altura_m ** 2)

  return indice
```

4. El mismo nutricionista nos sugiere implementar nuestro programa de forma que entregue clasificado el IMC. La Organización Mundial de la Salud clasifica el IMC de la siguiente forma:

ÍNDICE MASA CORPORAL	CLASIFICACIÓN
<18.50	Infrapeso
18.50 - 24.99	Peso Normal
25.00 - 29.99	Sobrepeso
>=30.00	Obeso

Modifica tu código de forma que ahora entregue un arreglo donde la primera posición contenga todos los índices que estén en infrapeso, en la segunda los que estén con peso normal y así sucesivamente

Una solución posible:

```
def imc(altura, peso, sistema='metrico'):
  np_altura_m = np.array(altura)
  np_peso_kg = np.array(peso)
  if 'imperial' == sistema:
       np_altura_m = np_altura_m * 0.254
       np_peso_kg = np_peso_kg * 0.453592

  indice = np_peso_kg / (np_altura_m ** 2)

  # Creamos el arreglo de jugadores livianos
  infrapeso = indice < 18.50
  normal = np.logical_and(18.50 <= indice, indice < 25.00)
  sobrepeso = np.logical_and(25.00 <= indice, indice < 30.00)
  obeso = indice >= 30.00

  # Retornamos el resultado usando un truco numpy: posiciones prendidas con
  # la condición
  return np.array([indice[infrapeso], indice[normal], indice[sobrepeso], indice[obeso]])
```

5. El nutricionista nos sugiere otra mejora: implementar nuestro programa de forma que entregue los nombres de las personas clasificados, en lugar de los imc, ya que no sabe a quien corresponde cada valor.

Modifica tu código de forma que ahora reciba un listado de nombres y entregue un arreglo con los nombres donde la primera posición contenga todos los índices que estén en infrapeso, en la segunda los que estén con peso normal y así sucesivamente.

El código del ejemplo anterior quedaría de la siguiente forma:

```
import numpy as np

nombres = ['a', 'b', 'c', 'd', 'e']
altura = [5.23, 7.17, 5.45, 5.98, 6.02]
peso = [135, 164, 137, 156, 161]

print(imc(nombres, altura, peso, 'imperial'))
```

Una solución posible:

```
def imc(nombres, altura, peso, sistema='metrico'):
  np_nombres = np.array(nombres)
  np_altura_m = np.array(altura)
  np_peso_kg = np.array(peso)
  if 'imperial' == sistema:
       np_altura_m = np_altura_m * 0.254
       np_peso_kg = np_peso_kg * 0.453592

  indice = np_peso_kg / (np_altura_m ** 2)

  # Creamos el arreglo de jugadores livianos
  infrapeso = indice < 18.50
  normal = np.logical_and(18.50 <= indice, indice < 25.00)
  sobrepeso = np.logical_and(25.00 <= indice, indice < 30.00)
  obeso = indice >= 30.00

  # Retornamos el resultado usando un truco numpy: posiciones prendidas con la condición
  return np.array([np_nombres[infrapeso], np_nombres[normal], np_nombres[sobrepeso],
np_nombres[obeso]])
```

6. Se está acercando el término de semestre y queremos que los alumnos no reprueben. Para ello vamos a jugar un juego que se llama Todos contra el Profesor. La idea es simple: tanto el profesor como los alumnos parten con una cantidad de fichas, por ejemplo 50. En cada turno, tanto el profesor como cada alumno tiran el dado. El número obtenido por el profesor se compara con el de cada alumno y se hace lo siguiente: si el profesor saca un número mayor entonces le saca una ficha al alumno, si son iguales no pasa nada y si el alumno saca un número mayor entonces le saca una ficha al profesor. El juego continúa hasta que alguien queda con cero fichas. Si es el profesor entonces todos los alumnos ganan 5 décimas en el examen. De lo contrario pierden 1 décima.
Cree una función en Python llamada todos_contra_el_profesor, que reciba como parámetros el número de alumnos y la cantidad de fichas, simule el tiro de los datos con números aleatorios entre 1 y 6 y ejecute las operaciones vistas, retornando True si los alumnos ganan, False de lo contrario. ¡Use todo lo de numpy que se le ocurra para hacer esto más fácil de implementar!

Un ejemplo de llamado es el siguiente:

```
import numpy
import random

def todos_contra_el_profesor(alumnos, fichas):
      #aca va su implementación

num_alumnos = 46
num_fichas = 50
if (todos_contra_el_profesor(num_alumnos, num_fichas)):
  print('El resultado es que los alumnos ganan 5 décimas')
else:
  print('Los alumnos pierden 1 décima')
```

7. Imaginese un arreglo de tamaño **num** y hay una mosca en la posición **pos**. La mosca se mueve en forma aleatoria (al azar) a la izquierda, a la derecha, o se queda quieta. En caso que la mosca se salga del arreglo, la mosca es atrapada por una araña.
Para modelar el comportamiento de la mosca, ustede genera un número aleatorio entre 0 y 1, si el numero es menor a 0.33, entonces la mosca se mueve a la izquierda, en caso que sea un número entre 0.33 y 0.66, se mantiene quieta, y si es mayor a 0.66 se mueve a la derecha.
Cree una función que reciba **num** y **pos** e itere hasta que la mosca sea atrapada por alguna araña. Luego, muestre por pantalla el número de veces que la mosca estuvo en cada posición y el número total de movimientos realizados.

   Sea I = Izquierda, D = derecha, y Q = quieto. Un ejemplo, para un arreglo de tamaño 5, con la mosca en posición inicial 2 y realizó los siguientes movimientos: D I I I D Q Q I hubiera impreso por pantalla lo siguiente: 2 3 2 1 0

   El número total de movimientos fue: 8

```
import numpy
import random
def mosca(num,pos):
    arreglo=numpy.zeros(num)
    while (pos>=0) and (pos<len(arreglo)):
        arreglo[pos]=arreglo[pos]+1
        temp=random.random()
        if temp<0.33:
            pos=pos-1
        else:
            if temp>0.66:
                pos=pos+1
    print(arreglo)
    print("el numero de movimientos es ", arreglo.sum())

mosca(20,10)
```

8. Imagine el siguiente juego, dado un arreglo de números enteros, usted se encuentra en la primera posición. El valor de esa posición le indica el número de casillas que se moverá (en caso de ser negativo retrocede). Cree una función que reciba un arreglo y retorne 0 en caso que usted llegue a la última posición (aquí se acaba el juego), -1 en caso que se caiga por la izquierda y 1 en caso que se caiga por la derecha.

Ejemplo: [2,3,-1,1,4] retorna 0 (mi posición sería 0, 2, 1, 4, gané retorno 0)
Ejemplo: [2,-2,-1,1,4] retorna -1 (mi posición sería 0, 2, 1, -1, caí retorno -1)
Ejemplo: [2,5,-1,1,4] retorna 1 (mi posición sería 0, 2, 1, 6, caí retorno 1)

```
def juego(arreglo):
        flag=1
        pos=0
        while(flag==1):
                pos=pos+arreglo[pos]
                if pos<0:
                        return -1
                if pos>=len(arreglo):
                        return 1
                if pos==len(arreglo)-1:
                        return 0
```