Funciones Generales de Python
Modo de uso *nombrefuncion*(*parametros*)

len(x) : cantidad de elementos (para cadena caracteres, listas)
sum(x): devuelve la suma de elementos de una lista numérica
min(x): devuelve el valor mínimo
max(x): devuelve el valor máximo
str(x): convierte la variable a string
int(x): convierte la variable a tipo de dato entero
list(x): convierte la variable a una lista
tuple(x): convierte la variable a una tupla
set(x): convierte la variable a un conjunto.

in: operador para verificar existencia en una cadena o colección
Ej 8 **in** lista

Funciones de cadenas de caracteres
Modo de uso: variable.*nombrefuncion*(*parametros*)

find(x): devuelve el índice donde se encuentra x. Si no se encuentra devuelve -1
index(x): devuelve el índice donde se encuentra x. Si no se encuentra devuelve error
strip(): elimina espacios en blanco al inicio y final de la cadena
split(sep): separa una cadena en una lista de acuerdo al separador
lower(): devuelve una nueva cadena con los caracteres en minúsculas
upper(): devuelve una nueva cadena con los caracteres en mayúsculas
isdigit(): devuelve True si todos los caracteres de la cadena son números
isalpha(): devuelve True si todos los caracteres de la cadena son letras
startswith(x): devuelve True si la cadena inicia con la subcadena x
endswith(x): devuelve True si la cadena termina con la subcadena x
count(sub): devuelve el número de veces que sub se encuentra en la cadena
replace(old,new): devuelve una nueva cadena en la que se han sustituido todas las apariciones de *old* con *new*
join(lista): Devuelve una cadena formada por la concatenación de todos los elementos de la lista. Ej `"".join(lista)`

Funciones de listas
Modo de uso: variable.*nombrefuncion*(*parametros*)

append(elem): agrega un elemento al final de la lista
sort(): ordena los elementos de la lista de menor a mayor
reverse(): invierte el orden de la lista
index(x): devuelve el índice donde se encuentra x. Si no se encuentra devuelve error

count(elem): devuelve el número de veces que elem se encuentra en la lista
pop(): elimina el último elemento de la lista y lo devuelve
pop(i): elimina el elemento de la lista en el índice i y lo devuelve
remove(elem): elimina elem de la lista

Funciones de random
Importar primero el módulo con
import random as rd

Modo de uso: rd.*nombrefuncion*(*parametros*)

randint(a,b): devuelve un número aleatorio entre a y b(ambos incluidos)
randrange(a,b): devuelve un número aleatorio entre a y b-1
randrange(b): devuelve un número aleatorio entre 0 y b-1
choice(secuencia):selecciona aleatoriamente un elemento de la secuencia(secuencia numérica, cadena o lista). Retorna un valor de acuerdo al tipo de dato de la secuencia.
sample(secuencia,k):selecciona aleatoriamente k elementos de la secuencia(secuencia numérica, cadena o lista) y retorna una lista.

Operadores
>
<
==
!=
**
^
|
&

Numpy

Propiedades

Modo de uso: nombrevariable.**propiedad**
No lleva paréntesis

ndim:Devuelve el número de dimensiones del arreglo.
shape:Devuelve una tupla con el número de elementos por eje (o axis).
dtype:Devuelve el tipo de datos de los elementos del arreglo
size:Devuelve el número de elementos en el arreglo

Funciones para crear vectores

Modo de uso : np.**nombrefuncion**(parametros)

```
#zeros Crea un arreglo lleno de ceros.
v = np.zeros((3,), int)

#ones Crea un arreglo lleno de unos.
v = np.ones((4,),int)
#full Crea un arreglo lleno con el valor pasado como
parámetro.
v = np.full((4,),5) #vector con 4 números 5
#arange Análoga a la función range() , pero devuelve un
arreglo de numpy de una dimensión con una secuencia numérica.
v7 = np.arange(5)
v8 = np.arange(1,5,dtype=float)
v9 = np.arange(1,6,2,dtype=float)
```

Funciones para crear vectores con valores aleatorios

```
#random.rand: Devuelve un arreglo lleno de números aleatorios
del 0 al 1 sin incluir.
v10 = np.random.random(3)

#random.randint: Devuelve un arreglo con números aleatorios
enteros dentro de un rango especificado [desde,hasta).
v11 = np.random.randint(1,20,10)

v12 = np.random.randint(10, size=25) #si solo se incluye un
numero, empieza en 0
```

Otras funciones

```
#copy: Devuelve un nuevo arreglo que es la copia del original.
a = np.array([1, 2, 3, 4], float)
b = a.copy()
#tolist(): Devuelve una lista a partir de un arreglo
c = a.tolist()
#astype Devuelve un nuevo arreglo con el tipo de dato cambiado
d = a.astype(int)
```

REDUCCIONES BÁSICAS

```
#funcion sum
x = np.array([1, 2, 3, 4])
```

```
c = np.sum(x)
print(c) #salida 10
d = x.sum()
print(d) #salida 10
#funciones estadisticas
x = np.array([1, 2, 3, 1])
c = x.mean()#1.75
print(c)
d = np.median(x)#1.5
print(d)

#valores extremos
#min y max

x = np.array([1, 3, 2])
print(x.min())#salida 1
print(x.max())#salida 3

#indices de valores min y max
i1 = x.argmin()  # índice del valor mínimo es 0
print(i1)
i2= x.argmax()  # índice del valor máximo es 1
print(i2)

#INDEXACION BOOLEANA
arr = np.arange(7)
#Recordemos que la comparación con un escalar #devuelve una
matriz booleana:
mb = arr > 4
print(mb) #[False False False False False  True  True]
may = arr[arr>4]
print(may) #[5 6]

#usando indexacion booleana en vectores paralelos
val1= np.arange(10)
exp1= val1**2
#recuperar elementos de un vector usando una condición en el
otro
res = exp1[val1 % 2 == 0]
print(res)
```

INDEXACIÓN CON VECTORES

WHERE

Ejemplo con vectores

```
>>> a = np.arange(5,10)
>>> np.where(a < 8)  # tell me where in a, entries are < 8
(array([0, 1, 2]),)  # answer: entries indexed by 0, 1, 2
```

Para los vectores(arreglos 1D) la tupla solo tiene un elemento con el vector que tiene los índices donde se cumple la condición

El resultado del where también puede ser utilizado para obtener las entradas del arreglo que satisfacen la condición:

```
>>> a[np.where(a < 8)]
array([5, 6, 7])   # selects from a entries 0, 1, 2
```