TIPOS ESTRUCTURADOS

LISTAS Y TUPLAS

Fundamentos de Informática

Grado en Ingeniería Química

Escuela Técnica Superior de Ingenieros Industriales y de Telecomunicación

UC | ETS de Ingenieros Industriales y de Telecomunicación

TIPOS ESTRUCTURADOS: Motivación

Hasta ahora hemos trabajado con variables en nuestros programas.

Por ejemplo, es posible calcular la media aritmética de un conjunto de números con una sola variable y un bucle! La desventaja es que esa variable sólo almacena un valor en cada iteración, los demás se pierden.

¿Cómo se pueden almacenar todos esos números en un programa? ¿Cómo se trabaja con grandes cantidades de datos en Python?

En matemáticas, los vectores y matrices permiten trabajar con grandes cantidades de datos.

En programación, se llaman arrays. Un array unidimensional es un vector, un array bidimensional es una matriz,…

En Python, se denominan listas, tuplas o cadenas.

Librería NumPy https://numpy.org/

TIPOS ESTRUCTURADOS: LISTAS I

Tipos de datos **escalares**: **int**, **float**, **complex**, **bool**.

Tipos de datos **estructurados**: **str**, **list**, **tuple**.

Una **lista** es una sucesión de elementos de cualquier tipo, incluida otra lista.

Una lista se define por los símbolos corchetes de apertura y cierre, que contienen a sus elementos separados por comas.

Sea una lista de **n** elementos:

[elemento_0**,** elemento_1**,…,** elemento_n-1**]**

[elemento_-n, elemento_-(n-1),…, elemento_-2, elemento_-1]

TIPOS ESTRUCTURADOS: LISTAS II

Para acceder a cada uno de sus elementos, se puede usar el operador de indexación. Hay dos opciones:

- Indexación positiva. El primer elemento de la lista por la izquierda ocupa la posición 0 en la lista, el siguiente ocupa la posición 1,…y el último elemento ocupa la posición n-1.
- Indexación negativa. El último elemento de la lista por la derecha ocupa la posición -1, el anterior ocupa la posición -2,…y el primer elemento ocupa la posición –n. Simplifica el acceso a los elementos del final de la lista.

TIPOS ESTRUCTURADOS: LISTAS III

Ejemplo. Sea una lista lista1 de 7 elementos, longitud 7, len(lista1)

>>> lista1= [10, 7.7, 3j, 'Hola', True, [2, 3, 4], (-6, -5)]

lista1	10	7.7	3j	'Hola'	True	[2,3,4]	(-6,-5)
Index positiva	0	1	2	3	4	5	6
Index negativa	-7	-6	-5	-4	-3	-2	-1

Es decir, lista1[0] es 10, …y lista1[6] es (-6, -5) o bien,

lista1[-1] es (-6, -5), lista1[-2] es [2, 3, 4],…y lista1[-7] es 10.

Precaución: el índice dentro del corchete no debe desbordar los límites de la lista, ni por la izquierda ni por la derecha!! Si desborda, Python dará un error del tipo: 'error de índice, índice de la lista fuera de rango'.

TIPOS ESTRUCTURADOS: LISTAS IV

¿Cómo acceder a los elementos de la sublista de lista1?

lista1	10	7.7	3j	'Hola'	True	[2,3,4]	(-6,-5)
Index positiva	0	1	2	3	4	5	6
Index negativa	-7	-6	-5	-4	-3	-2	-1

Añadir otro índice con la posiciones permitidas de la sublista.

Ejemplo:

lista1[5][0],…lista1[5][2] con indexación positiva.

lista1[-2][-3],…lista1[-2][-1] con indexación negativa.

TIPOS ESTRUCTURADOS: TUPLAS I

Una **tupla** es una sucesión de elementos de cualquier tipo, incluida otra tupla.

Una tupla se define por los símbolos paréntesis de apertura y cierre, que contienen a sus elementos separados por comas.

Sea una tupla de **n** elementos:

(elemento_0, elemento_1,…, elemento_n-1)

Con indexación positiva.

(elemento_-n, elemento_-(n-1),…, elemento_-2, elemento_-1)

Con indexación negativa.

TIPOS ESTRUCTURADOS: TUPLAS II

Ejemplo.

Sea una tupla tupla1 de 7 elementos, longitud 7, len(tupla1)

>>> tupla1= (10, 7.7, 3j, 'Hola', True, [2,3,4], (-6, -5))

tupla1[0] ó tupla1[-7] es 10, …y tupla1[6] ó tupla1[-1] es (-6, -5).

¿Qué ocurre si se intenta acceder a tupla1[10] o tupla1[-8] ?

Python dará un error: error de índice, índice de la tupla fuera de rango!!

¿Cómo acceder a los elementos de la lista o subtupla de la tupla1?

Añadir otro índice con la posiciones permitidas de la lista o subtupla.

Ejemplo:

tupla1[6][0] y tupla1[6][1] con indexación positiva.

tupla1[-1][-2] y tupla1[-1][-1] con indexación negativa.

TIPOS ESTRUCTURADOS: CADENAS I

Una **cadena** es una sucesión de caracteres.

Todo lo anterior sobre listas y tuplas se aplica también a las cadenas.

Ejemplo. Sea la cadena:

```
>>> cad1='Hola a todos'
```

Practica con el operador de indexación positivo o negativo sobre cad1.

```
>>>cad1[0];cad1[11]
```

¿Cómo acceder a una subcadena de una cadena?

OPERADOR DE CORTE : I

Es aplicable a cadenas, listas y tuplas.

Sintaxis general

cadena**[start:stop**[**:step**]]

start, stop y step pueden ser constantes, variables o expresiones enteras permitidas por la longitud de la cadena. Se admite indexación positiva o negativa.

start es la posición del carácter inicial, 0 por defecto.

stop es la posición del carácter final*, no incluído*, len(cadena) por defecto.

step indica el incremento/decremento. Puede ser positivo o negativo, pero NO CERO, 1 por defecto.

OPERADOR DE CORTE : II

cad1	H	o	l	a		a		t	o	d	o	s
Indexación positiva	0	1	2	3	4	5	6	7	8	9	10	11
Indexación negativa	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1

Prueba en la Shell de Python: cad1[:4], cad1[7:], cad1[::-1]

No recomendable	cad1[7:8]	cad1[:]	cad1[-12:-8]	cad1[7:]
Recomendable	cad1[7]	cad1	cad1[:4]	cad1[-5:]
Resultado	t	Hola a todos	Hola	todos

OPERADOR DE CORTE : III

¿Cómo acceder a los elementos que ocupan posiciones pares/impares en una lista/tupla?

Sea la lista l1= [-2, 8, 3, 55, 0, 15, 5.75]

l1[0: len(l1): 2] es [-2, 3, 0, 5.75]

l1[1: len(l1): 2] es [8, 55, 15]

Sea la tupla t1= (-2, 8, 3, 55, 0, 15, 5.75)

t1[0: len(t1): 2] es (-2, 3, 0, 5.75)

t1[1: len(t1): 2] es (8, 55, 15)

SECUENCIAS DE ESCAPE

Secuencias de escape. Son un par de caracteres, el primero \ y tienen un significado especial. Los dos caracteres denotan y ocupan el espacio de un único carácter. Por ejemplo, el salto de línea ocupa la posición 10 en el código Unicode.

\n Salto de línea

\t Tabulador horizontal

\v Tabulador vertical

\a Carácter de 'campana'

```
>>> print ('Hola\na todos')
```

¿LISTAS O TUPLAS?

Una lista admite modificación de cualquiera de sus elementos, pero una tupla NO!

Ejemplo. Sea la lista:

>>> lista1=[10, 7.7, 3j, 'Hola', True, [2,3,4], (-6, -5)]

>>> lista1[0]=0 está permitido, pero

>>> lista1[6][0]=0 da error! pues una tupla NO permite asignación de ítems.

Se dice que las listas son objetos mutables y las tuplas inmutables.

Las cadenas son objetos inmutables al igual que las tuplas.

Ejemplo. Sea la cadena cad1='Hola a todos'

>>> cad1[0]='h' da error!

OTROS OPERADORES I

Aplicables a listas/cadenas/tuplas.

+ Concatenación

*** Repetición**

Operadores de comparación **<** , **<=** , **>** , **>=** , **!=** , **==**

is

OTROS OPERADORES II

Ejemplos:

>>> 'hola'+' a todos' obtiene 'hola a todos'

>>> [0]*4 es [0,0,0,0]

Sean las listas

>>> a=[1,2,3]

>>> b=[1,2,3]

>>> a==b es True pues contiene los mismos valores en todos sus elementos.

>>> a is b es False pues son listas distintas que se almacenan en espacios de memoria distintos.

>>> a+b es [1,2,3,1,2,3]

BUCLE for-in CON LISTAS/TUPLAS/CADENAS

for elemento **in** serie**:**

bloque

serie puede ser: range(), pero también, lista, tupla o cadena.

Ejemplo.

```
lista=[7,8,9]
for i in range(len(lista)):
    print(lista[i])

for elemento in [7,8,9]:
    print(elemento)

lista=[7,8,9]
for elemento in lista:
    print(elemento)
```

```
tupla=(7,8,9)
for i in range(len(tupla)):
    print(tupla[i])

for elemento in (7,8,9):
    print(elemento)

tupla=(7,8,9)
for elemento in tupla:
    print(elemento)
```

```
cadena='Hola a todos'
for i in range(len(cadena)):
    print(cadena[i])

for elemento in 'Hola a todos':
    print(elemento)

cadena='Hola a todos'
for elemento in cadena:
    print(elemento)
```

MATRICES I

¿Cómo se define una matriz en Python?

Con una lista de listas.

Ejemplo1:

```
>>> mat1=[
        [0, 1, 2],
        [3, 4, 5]
]
```

define la matriz siguiente:

$$\begin{matrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{matrix}$$

```
>>> mat1
[[0, 1, 2], [3, 4, 5]]
```

MATRICES II

¿Cómo acceder a una fila de la matriz anterior mat1?

mat1[0], mat1[1]

¿Cómo acceder a una columna de la matriz anterior?

```
>>>for fila in mat1:
        print(fila[0])
```

Accede a la columna 0 de mat1, un elemento por iteración

Ejemplo2. Construir la matriz identidad 3x3:

```
>>> mat2 =[[1, 0, 0], [0, 1, 0],[0,0,1]]
```

$$\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}$$

INICIALIZACIÓN DE MATRICES POR COMPRENSIÓN (for implícito)

Inicializar una matriz de f filas x c columnas por comprensión (bucles implícitos) a un mismo valor v:

[[v for i in range(c)] for j in range(f)]

Ejemplo: Inicializar una matriz de 2x3 a cero por comprensión.

```
>>> mat1 =[[0 for i in range(3)] for j in range(2)]
>>> mat1
[[0, 0, 0], [0, 0, 0]]
```

0 0 0

0 0 0

¿Cómo acceder a una columna de mat1 con bucle implícito?

```
>>> [fila[0] for fila in mat1]
```

AÑADIR ELEMENTOS A UNA LISTA I

Hay dos formas de hacerlo:

1. Con el operador de concatenación +

>>> b=[1,2,3] crea una lista b [1,2,3]

>>> b=b+[4] crea una nueva lista b concatenando [1,2,3] y [4]
osea, [1,2,3,4]

2. Con el método **append.**

lista.**append(**elemento**)** Añade elemento a lista.

>>> b=[1,2,3] crea una lista b [1,2,3]

>>> b.append(4) añade a la lista b existente el 4 por la derecha de b
[1,2,3,4]

Diferencia: el operador de concatenación + **crea una nueva lista** con [1,2,3,4], mientras que el método append **añade a la lista existente** un nuevo elemento a la derecha, siendo por tanto, más eficiente.

AÑADIR ELEMENTOS A UNA LISTA II

¿Cómo leer una lista por teclado?

1. Crear una lista vacía [] y añadirla elementos con un bucle usando el método append, un elemento en cada iteración.

```
1 lista=[]
2 n=int(input("longitud de la lista "))
3 for i in range(n):
4     x=float(input("Dame un numero "))
5     lista.append(x)
6     print("lista ",lista)
```

2. Crear una lista de ceros con la longitud adecuada, y modificar sus valores en un bucle usando el operador de indexación, un valor en cada iteración.

```
 9 n=int(input("longitud de la lista "))
10 lista=[0]*n
11 for i in range(n):
12     x=float(input("Dame un numero "))
13     lista[i]=x
14     print("lista ",lista)
```

LEER/ESCRIBIR LISTA DE LISTAS I

1

```
1  nf=int(input("filas "))
2  nc=int(input("columnas "))
3  m=[] # crea lista vacia
4
5  # Lee matriz por teclado
6  for i in range(nf):
7      m.append([])
8      print(m)
9      for j in range(nc):
10         n=int(input('elemento ('+str(i)+','+str(j)+'): '))
11         m[i].append(n)
12         print(m)
13
14 # Escribe matriz en pantalla
15 print("\nMatriz")
16 for i in range(nf):     # Filas
17     for j in range(nc):  # Columnas
18         print(m[i][j], end=' ')
19     print()
```

En la práctica, una matriz NO se lee por teclado. Demasiado lento!!

LEER/ESCRIBIR LISTA DE LISTAS II

2

```
nf=int(input("filas "))
nc=int(input("columnas "))
# Crear matriz nula
m=[]
for i in range(nf):
    m.append([0]*nc)
    print(m)

# Leer por teclado matriz
for i in range(nf):
    for j in range(nc):
        m[i][j]=int(input('Elemento('+str(i)+','+str(j)+'): '))

# Escribe matriz en pantalla
print("Matriz")
for i in range(len(m)):    # Filas
    print(m[i])
```

En la práctica, una matriz NO se lee por teclado. Demasiado lento!!

BORRAR ELEMENTOS DE UNA LISTA

Sentencia **del**, del inglés DELETE. Permite eliminar un elemento de una lista, un trozo de la misma (usar el operador de corte :) o la lista completa.

Ejemplo:

```
>>>b=[1,2,3,4,5]   crea b [1, 2, 3, 4, 5]
>>>del b[2]        elimina el elemento de posición 2 quedando b [1, 2, 4, 5]
>>>del b[1:3]      elimina los elementos de posiciones 1 y 2 quedando b [1, 5]
>>>del b           elimina la lista b
>>>b               provoca el siguiente error. NameError: name 'b' is not defined
```

PERTENENCIA DE UN ELEMENTO A UNA SERIE

if elemento [**not**] **in** serie**:**

bloque

serie puede ser lista (simple, sin sublistas), tupla (simple) o cadena.

Ejemplo. Buscar un número x en una lista dada:

```
if x in lista:
   print('encontrado')
```

```
for elemento in lista:
    if x==elemento:
        print('encontrado')
```

```
if x not in lista:
    print('no encontrado')
```

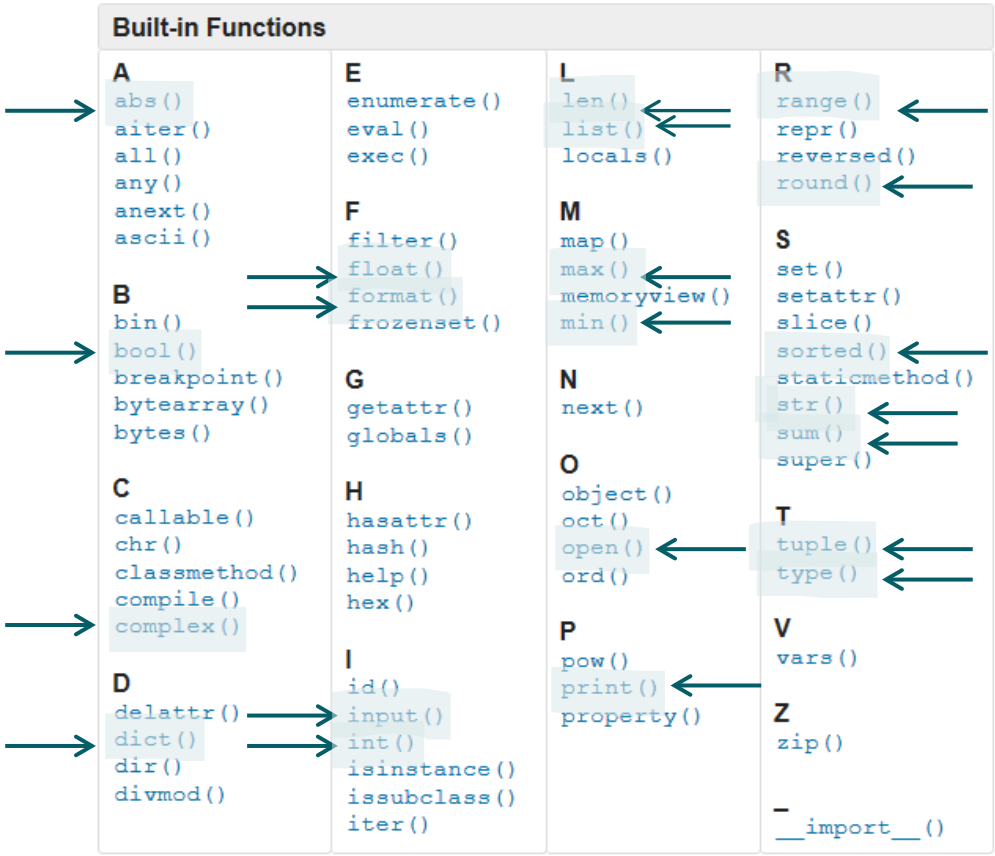
```
sw=0
for elemento in lista:
    if x==elemento:
        sw=1
        break
if sw==0: print ('no encontrado')
```

Funciones predefinidas en Python

Python » 3.10.7 Documentation » The Python Standard Library » Built-in Functions

Built-in Functions

The Python interpreter has a number of functions and types built into it that are always available. They are listed here in alphabetical order.



Ejercicio 1

Pedir cinco números por teclado y calcular su media usando un vector para almacenar los números. Escribir en pantalla los valores que superan la media y su posición en el vector.

cap4_1.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap4_Arrays\resueltos\cap4_1.py (3.10.7)

File Edit Format Run Options Window Help

```
1  #PB Oct 2022
2  #Forma 1
3  lista=[]
4  n=int(input("longitud de la lista "))
5  for i in range(n):
6      x=float(input("Dame un número "))
7      lista.append(x)
8      print("lista ",lista)
9  #Forma 2
10 n=int(input("longitud de la lista "))
11 lista=[0]*n
12 for i in range(n):
13     x=float(input("Dame un número "))
14     lista[i]=x
15     print("lista ",lista)
16 #Calculamos la media
17 media=0
18 for i in range(n):
19     media+=lista[i]
20 media=media/n
21 print("La media es ",media)
22 #Valores que superen la media y posición en lista
23 for i in range(n):
24     if lista[i]>media:
25         print(lista[i],'posición ',i)
```

Ejercicio 2

Calcular y presentar en pantalla los valores x_2 hasta x_{n-1} (con n como máximo 100) de la sucesión: $x_i = x_{i-1} + 3x_{i-2} + 4 / 3$. El programa leerá por teclado los valores de n, x_0 y x_1.

cap4_2.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap4_Arrays\resueltos\cap4_2.py (3.10.7)

File Edit Format Run Options Window Help

```
1  #PB Oct 2022
2  #Validacion de n
3  while True:
4      n=int(input("n° de términos de la sucesión "))
5      if n>2 and n<=100: break
6      print("Como mucho 100")
7      print("Vuelve a intentarlo")
8
9  lista=[0]*n     #usando el operador de repeticion
10 #Definir una lista de ceros con longitud n
11 lista[0]=float(input("Primer término "))
12 lista[1]=float(input("Segundo término "))
13 for i in range(2,n):
14     lista[i]=lista[i-1]+3*lista[i-2]+4/3
15     print("lista[",i,']:',lista[i])
```

Ejercicio 3a

Buscar un número en un vector desordenado. Inicializar el vector en una sentencia dentro del programa. Encontrar y escribir en pantalla todas las posiciones del vector donde se encuentra el número.

Usar un interruptor.

cap4_3a.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap4_Arrays\resueltos\cap4_3a.py (3.10.7)

File Edit Format Run Options Window Help

```
1  #PB Oct 2022
2  #Con switch
3  lista=[17,3,22,17,33,19]
4  x=int(input("Dame un número "))
5  encontrado=False
6  for i in range(len(lista)):
7      if lista[i]==x:
8          print("Encontrado en posición ",i)
9          encontrado=True
10 if not(encontrado):
11     print(x,"no está en la lista")
```

Ejercicio 3b

Usar el operador de pertenencia in.

cap4_3b.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap4_Arrays\resueltos\cap4_3b.py (3.10.7)

File Edit Format Run Options Window Help

```
1 #PB Oct 2022
2 lista=[17,3,22,17,33,19]
3 x=int(input("Dame un número "))
4 if x in lista:
5     print("Encontrado en la lista")
6     print('primera posición',lista.index(x))
7     print('cuantos',lista.count(x))
8 if x not in lista:
9     print(x," no está en la lista")
```

Ejercicio 4

Igual que el ejercicio anterior en la versión a, pero reducir la búsqueda a encontrar la primera posición de coincidencia en el vector. Si no existe el número en el vector, mostrar un mensaje en consecuencia.

cap4_4.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap4_Arrays\resueltos\cap4_4.py (3.10.7)

File Edit Format Run Options Window Help

```
1  #PB Oct 2022
2  lista=[17,3,22,17,33,19]
3  x=int(input("Dame un número "))
4  encontrado=False
5  for i in range(len(lista)):
6      if lista[i]==x:
7          print("Encontrado en posición ",i)
8          encontrado=True
9          break
10 if not(encontrado):
11     print(x," no está en la lista")
```

Ejercicio 5

Buscar un número en un vector ordenado ascendentemente. Inicializar el vector en una sentencia dentro del programa.

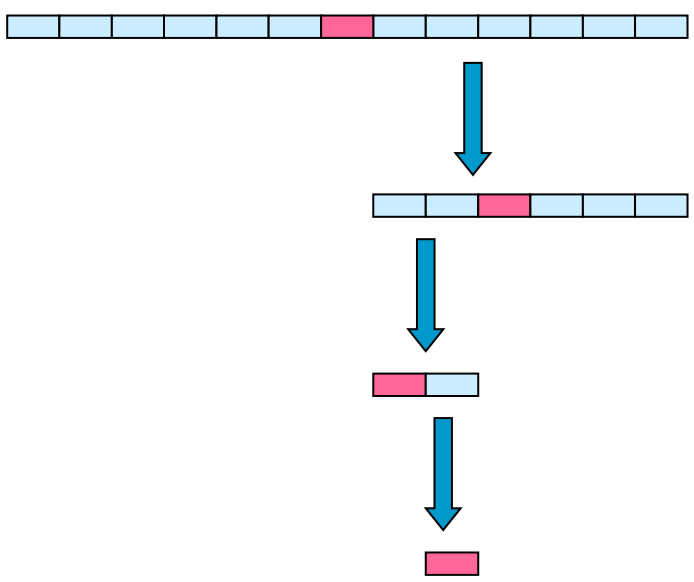
cap4_5.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap4_Arrays\resueltos\cap4_5.py (3.10.7)

File Edit Format Run Options Window Help

```
1  #PB Oct 2022
2  lista=[-1,0,3,3,7,15,32]
3  x=int(input("Dame un número "))
4  i=0
5  while x>lista[i] and i<len(lista)-1:
6      i+=1
7  if x==lista[i]:
8      print("Encontrado en posición ",i)
9  else:
10     print(x,"no está en la lista")
```

Ejercicio 6 Algoritmo

Buscar un número en un vector ordenado ascendentemente. Inicializar el vector en una sentencia dentro del programa. Usar el algoritmo de la búsqueda binaria o dicotómica.



La búsqueda dicotómica es un algoritmo más eficiente que el anterior.

La búsqueda se realiza siempre en la componente **central** de una lista cuya longitud se va reduciendo a la **mitad** izquierda o derecha del centro (según sea el número) sucesivamente hasta que, o bien encontramos el valor buscado, o bien el intervalo de búsqueda se ha reducido a una componente.

Ejercicio 6

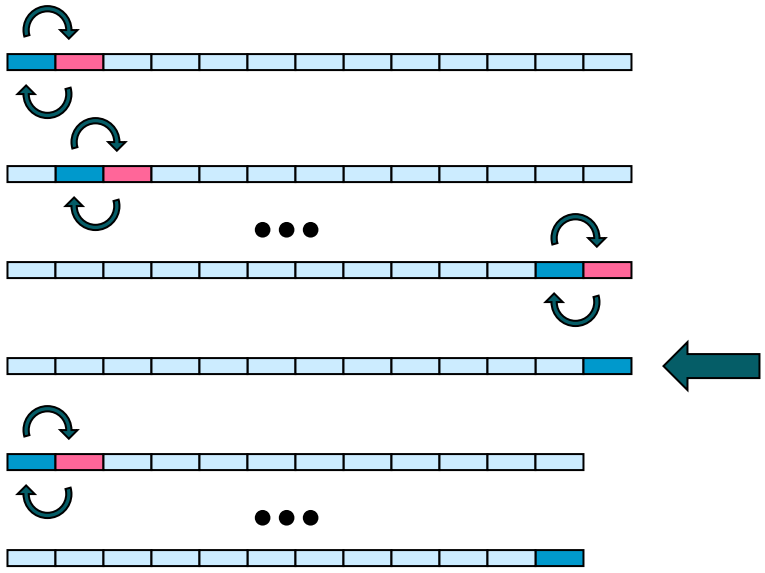
cap4_6.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap4_Arrays\resueltos\cap4_6.py (3.10.7)

File Edit Format Run Options Window Help

```
1  #PB Oct 2022
2  lista=[-1,0,3,3,7,15,32]
3  x=int(input("Dame un número "))
4  cen=len(lista)//2
5  izq=0
6  der=len(lista)-1
7  while x!=lista[cen] and izq<der:
8      if x>lista[cen]:
9          izq=cen+1
10     else:
11         der=cen-1
12     cen=(izq+der)//2
13
14 if x==lista[cen]:
15     print("Encontrado en posición ",cen)
16 else:
17     print(x," no está en la lista")
```

Ejercicio 7 Algoritmo

Ordenar ascendentemente/descendentemente un vector. Usar el método de la burbuja.



Se repite el proceso con el conjunto que queda y así sucesivamente

Ejercicio 7

Sea el vector [45, 26, 9, 3, 1]

Numero de parejas 4
[26, 9, 3, 1, 45]

Numero de parejas 3
[9, 3, 1, 26, 45]

Numero de parejas 2
[3, 1, 9, 26, 45]

Numero de parejas 1
[1, 3, 9, 26, 45]

cap4_7a.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap4_Arrays\resueltos\cap4_7a.py (3.10.7)

File Edit Format Run Options Window Help

```
1  #PB Oct 2022
2  v=[45,26,9,3,1]
3  np=len(v)-1
4  for i in range(np,0,-1):  #Número de parejas
5      print('Nº de parejas ',i)
6      for j in range(0,i): #pareja concreta a comparar
7          if v[j]>v[j+1]:
8                  v[j+1],v[j]=v[j],v[j+1]
9      print(v)
10 #print(v)
```

Para ordenar el vector descendentemente (de mayor a menor) basta cambiar el operador relacional “mayor que” (>) por “menor que” (<) en la condición.

Ejercicio 8

Leer por teclado una matriz de 2x3 y escribirla en pantalla.

cap4_8.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap4_Arrays\resueltos\cap4_8.py (3.10.7)

File Edit Format Run Options Window Help

```
1  #PB Oct 2022
2  nf=int(input("filas "))
3  nc=int(input("columnas "))
4  m=[] # crea lista vacia
5  # Lee matriz por teclado
6  for i in range(nf):
7      m.append([])
8      print(m)
9      for j in range(nc):
10         n=int(input('elemento ('+str(i)+','+str(j)+'): '))
11         m[i].append(n)
12         print(m)
13 # Escribe matriz en pantalla
14 print("\nMatriz")
15 for i in range(nf):     # Filas
16     for j in range(nc):  # Columnas
17         print(m[i][j], end=' ')
18     print()
```

Ejercicio 9

Inicializar una matriz en una sentencia dentro del programa y escribirla en pantalla.

cap4_9.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap4_Arrays\resueltos\cap4_9.py (3.10.7)

File Edit Format Run Options Window Help

```
1  #PB Oct 2022
2  m=[[1,2,3],[4,5,6]]     #de 2x3
3  # Escribe matriz en pantalla
4  print(" Matriz "+5*'*')
5  for i in range(len(m)):     # Filas
6      print(m[i])
7  # Escribe matriz en pantalla
8  print(" Matriz *****")
9  for i in range(len(m)):     # Filas
10     for j in range(len(m[i])): #Columnas
11         print(m[i][j], end=' ')
12     print()
```

Ejercicio 10c

Buscar un número en una matriz desordenada de 3x5, indicando todas las filas, si hay varias.

cap4_10c.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap4_Arrays\resueltos\cap4_10c.py (3.10.7)

File Edit Format Run Options Window Help

```
1  #PB Oct 2022
2  m=[[4,2,3,9,0],[7,5,7,4,4],[6,5,4,3,2]]
3  # Escribe matriz en pantalla
4  print("\tMatriz")
5  for i in range(len(m)):     # Filas
6      print(m[i])
7  x=int(input('Dame un número para buscar: '))
8  sw=0
9  for i in range(len(m)):
10     if x in m[i]:
11         print('Encontrado en fila ',i)
12         sw=1
13 if sw==0: print('No existe el valor buscado.')
```

Ejercicio 11

Sean 3 vectores de 4 componentes conocidas, calcular el vector suma y su módulo. Usar una lista para cada vector.

cap4_11.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap4_Arrays\resueltos\cap4_11.py (3.10.7)

File Edit Format Run Options Window Help

```
1  #PB Oct 2022
2  from math import sqrt
3  v1=[-1]*4;v2=[2]*4;v3=[-3]*4
4  s=[0]*4
5  for i in range(len(v1)):
6      s[i]=v1[i]+v2[i]+v3[i]
7      print('Componente ', i,' del vector suma: ',s[i])
8  print("Vector suma\n",s)
9  #Forma1..............
10 acum=0
11 for i in range(len(s)):
12     acum+=s[i]**2
13 modu=sqrt(acum)
14 print('Módulo del vector suma: ',modu)
15 #Forma2.............
16 acum=0
17 for elem in s:
18     acum+=elem**2
19 modu=sqrt(acum)
20 print('Módulo del vector suma: ',modu)
```

Ejercicio 12

Igual que el ejercicio anterior, pero usando una lista de listas para almacenar los tres vectores.

cap4_12a.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap4_Arrays\resueltos\cap4_12a.py (3.10.7)

File Edit Format Run Options Window Help

```
1  #PB Oct 2022
2  from math import sqrt
3  matriz=[[1,2,3,4],[-1,-2,-3,-4],[5,6,7,8]]
4  nf=len(matriz)
5  nc=len(matriz[0])
6  s=[0]*nc
7  #print("Vector suma inicializado a 0 ",s)
8  for fila in matriz:
9      c=0
10     for elem in fila:
11         s[c]+=elem
12         c+=1
13 #    print(s)
14 print("Vector suma :", s)
15 sumacuad=0
16 for elem in s:     # Columnas
17     sumacuad+=elem**2
18 #    print('Suma de cuadrados de s: ',sumacuad)
19 modu=sqrt(sumacuad)
20 print('Módulo del vector suma: ',modu)
```

Ejercicio 13

Sumar dos matrices, usando bucles for implícitos para inicializar la matriz suma

cap4_13.py - C:\DATOS\Curso 2022-2023\GIQ\LAB_PYTHON_22_23\cap4_Arrays\resueltos\cap4_13.py (3.10.7)

File Edit Format Run Options Window Help

```
1  #PB Oct 2022
2  a=[[2,1,3],[4,8,5]]
3  b=[[-2,-1,-3],[-4,-8,-5]]
4  c=[[0 for i in range(len(a[0]))] for j in range(len(a))]
5  #Suma de matrices
6  for i in range (len(a)):
7      for j in range (len(a[0])):
8          c[i][j]=a[i][j]+b[i][j]
9  #Escribir el resultado en pantalla.
10 print("Matriz suma\n",c)
```

Ejercicio 14. Fórmula

Multiplicar dos matrices conocidas de 2x3 y 3x4. Inicializar la matriz producto usando el operador *.

$$\begin{pmatrix} P_{11} & P_{12} & \dots & P_{1l} \\ P_{21} & P_{22} & \dots & P_{2l} \\ & \dots & P_{ij} & \dots \\ P_{n1} & P_{n2} & \dots & P_{nl} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} & \dots & A_{1m} \\ A_{21} & A_{22} & \dots & A_{2m} \\ & \dots & A_{ik} & \dots \\ A_{n1} & A_{n2} & \dots & A_{nm} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} & \dots & B_{1l} \\ B_{21} & B_{22} & \dots & B_{2l} \\ & \dots & B_{kj} & \dots \\ B_{m1} & B_{m2} & \dots & B_{ml} \end{pmatrix}$$

$$P_{ij} = \sum_{k=1}^{k=m} A_{ik} \times B_{kj} \quad , \forall \text{ i =1, …n y j = 1, …l}$$

Ejercicio 14

cap4_14.py - C:\DATOS\Curso 2024-2025\GIQ\LAB_PYTHON_24_25\cap4_Arrays\resueltos\cap4_14.py (3.12.6)

File Edit Format Run Options Window Help

```
#PB Oct 2024
a=[[3,2,1],[1,3,2]] #2x3
b=[[1,2,-1,3],[-4,-1,-2,-3],[1,2,1,-1]] #3x4
p=[]  #2x4
for i in range(len(a)):
    p.append([0]*len(b[0]))

#print('p inicialización',p)
for i in range(len(a)):
    for j in range(len(b[0])):
        for k in range(len(a[0])):
            p[i][j]+=a[i][k]*b[k][j]
#             print(i,j,k,a[i][k],b[k][j],p[i][j])
print("Matriz a")
for fila in a:
    for elem in fila:
        print(elem,end="    ")
    print()
print()
print("Matriz b")
for fila in b:
    for elem in fila:
        print(elem,end="    ")
    print()
print()
print("Matriz producto")
for fila in p:
    for elem in fila:
        print(elem,end="   ")
    print()
```