



FUNDAMENTOS DE INFORMATICA

INGENIERA SILVIA PATRICIA BARDELLI

PROFESORA ING. SILVIA PATRICIA BARDELLI

CLASE NRO 6

Temas:

- § Arreglos: Listas
- § Agregado y eliminación de elementos
- § Subíndices
- § Búsqueda de máximos y mínimos

INTRODUCCIÓN

- § Supongamos que se nos plantea el siguiente problema:
- § *Leer tres números enteros e imprimirlos en orden inverso.*
- § ¿Sabemos resolverlo?

INTRODUCCIÓN

```
a = int(input("Ingrese un número: "))  
b = int(input("Ingrese otro número: "))  
c = int(input("Y otro más: "))  
print(c, b, a)
```

INTRODUCCIÓN

- § Ahora que conocemos la estrategia de resolución, vamos a ampliar el problema a **100 números** en lugar de 3.
- § ¿Podemos utilizar la misma estrategia?

INTRODUCCIÓN

- § Evidentemente es necesario implementar otro tipo de solución, que utilice una **estructura de datos**, es decir a un conjunto de datos agrupados de alguna manera.
- § A esta estructura de datos se la denomina ***arreglo***.

ARREGLOS

Un *arreglo* es un conjunto de variables agrupadas bajo un solo nombre.



vec

ARREGLOS

§ Dentro de los arreglos existen los **vectores** y las **matrices**.

§ En este curso trabajaremos exclusivamente con vectores. Matrices se tratará en la materia Programación I.

ARREGLOS

Dado que un vector es un conjunto de variables, se hace necesario poder **identificar** a cada una de esas variables para poder trabajar con ellas.

4	7	9	2	1	8	6	0	5	3
---	---	---	---	---	---	---	---	---	---

vec

ARREGLOS

Para eso se utiliza un *subíndice*, que identifica la posición de cada variable. Equivale al *número de orden* de la variable dentro del arreglo.

4	7	9	2	1	8	6	0	5	3
0	1	2	3	4	5	6	7	8	9
vec									

SUBÍNDICES

- § Los subíndices deben ser números enteros.
- § Siempre comienzan a partir del 0. Y terminan uno antes del tamaño del vector.

SUBÍNDICES

- § Se escriben luego del nombre del arreglo, encerrados entre corchetes.
- § Pueden usarse constantes, variables y expresiones.

SUBÍNDICES

4	7	9	2	1	8	6	0	5	3
0	1	2	3	4	5	6	7	8	9
vec									

`print(vec[2])` *# Constante*

`a = 4`

`print(vec[a])` *# Variable*

`print(vec[a+3])` *# Expresión*

LISTAS

- § En Python los vectores se implementan a través de **listas**.
- § Las listas pueden crecer y reducirse, algo que no ocurre con los vectores tradicionales en otros lenguajes de programación.

RESOLUCIÓN DEL EJEMPLO NRO 1

Leer 100 números e imprimirlos en orden inverso

```
vec = [ ]
```

```
i = 0
```

```
while i < 100:
```

```
    n = int(input("Ingrese un número: "))
```

```
    vec.append(n)
```

```
    i = i + 1
```

Impresión

```
i = 99
```

```
while i >= 0:
```

```
    print(vec[i])
```

```
    i = i - 1
```

NOVEDADES DEL EJEMPLO NRO 1

Declaración de listas

Al inicializar una variable con dos corchetes sin escribir nada entre ellos, estamos creando una *lista vacía*.

Método append()

Permite agregar nuevos elementos al final de una lista, como si se tratara de nuevos vagones de un ferrocarril.

EJEMPLO NRO 2

Objetivo:

Imprimir por pantalla una lista definida dentro del programa.

RESOLUCIÓN DEL EJEMPLO NRO 2

Imprimir una lista por pantalla

`lista = [4, 7, 2, 9, 5]`

`x = 0`

`while x < 5:`

`print(lista[x], end=" ")`

`x = x + 1`

NOVEDADES DEL EJEMPLO NRO 2

Las listas pueden declararse con elementos en su interior, es decir no vacías.

Estos elementos pueden ser constantes o variables:

```
lista1 = [4, 7, 2, 9, 5]  
lista2 = [a, num, 3, x]
```

NOVEDADES DEL EJEMPLO NRO 2

end= " " (dentro del print)

Indica que el renglón no terminó, y que la siguiente impresión debe aparecer al lado de la anterior.

EJEMPLO NRO 3

Objetivo:

Leer 50 números enteros, calcular su promedio e imprimir aquellos valores leídos que sean mayores al promedio obtenido.

EJEMPLO NRO 3 – PRIMERA PARTE

Primera parte: Lectura de datos y cálculo del promedio

ELEMENTOS = 50

v = []

i = 0

suma = 0

while i < ELEMENTOS:

 n = int(input("Ingrese un numero: "))

 v.append(n)

 suma = suma + n

 i = i + 1

prom = suma/ELEMENTOS

print("El promedio es", prom)

EJEMPLO NRO 3 – SEGUNDA PARTE

Segunda parte:
Imprimir aquellos elementos leídos
que sean mayores al promedio

`i = 0`

`while i < ELEMENTOS:`

`if v[i] > prom:`

`print(v[i])`

`i = i + 1`

NOVEDADES DEL EJEMPLO NRO 3

Si en alguna oportunidad fuera necesario modificar la cantidad de números a ingresar, el único cambio requerido será el valor de la variable ELEMENTOS, sin necesidad de ajustar ningún otro aspecto del programa.

Escribir el nombre de la variable en mayúsculas es una convención utilizada para definir constantes.

ATENCIÓN

4	7	9	2	1	8	6	0	5	3
0	1	2	3	4	5	6	7	8	9

vec

$a = 5$

$\text{vec}[a] = \text{vec}[a] + 1$ *# El 8 se convierte en 9*

$b = 7$

$\text{vec}[b] = \text{vec}[b + 1]$ *# El 0 se convierte en 5*

EJEMPLO NRO 4

Objetivo:

Leer un conjunto de números y guardarlos en una lista, finalizando la carga con -1.

Luego buscar el mayor elemento leído, mostrarlo y eliminarlo del arreglo.

Imprimir por pantalla la lista antes y después del borrado.

EJEMPLO NRO 4 – PRIMERA PARTE

Primera parte: Lectura de datos

```
v = [ ]
```

```
n = int(input("Ingrese un número o -1 para terminar: "))
```

```
while n != -1:
```

```
    v.append(n)
```

```
    n = int(input("Ingrese un número o -1 para terminar: "))
```

EJEMPLO NRO 4 – SEGUNDA PARTE

Segunda parte: Cálculo de la cantidad de elementos

`largo = len(v)` *# len() devuelve la longitud de la lista*

`if largo == 0:`

`print("No se ingresaron valores")`

`else:`

EJEMPLO NRO 4 – TERCERA PARTE

Tercera parte: Búsqueda del máximo

```
mayor = v[0]
```

```
pos = 0
```

```
for i in range(largo):
```

```
    if v[i] > mayor:
```

```
        mayor = v[i]
```

```
        pos = i
```

```
imprimirlista(v)
```

```
print("El máximo es", mayor, "y se encontró en la  
posición", pos)
```

EJEMPLO NRO 4 – CUARTA PARTE

Cuarta parte: Eliminación del máximo

print("Borrando el", mayor)

del v[pos]

imprimirlista(v)

EJEMPLO NRO 4 – QUINTA PARTE

Quinta parte: Función de impresión

```
def imprimirlista(vec):  
    largo = len(vec)  
    for i in range(largo):  
        print(vec[i], end=" ")  
    print()
```

EJEMPLO NRO 4 – PROGRAMA COMPLETO

```
def imprimirlista(vec):  
    largo = len(vec)  
    for i in range(largo):  
        print(vec[i],end=" ")  
    print()
```

Programa principal

```
v = []  
n = int(input("Ingrese un numero o -1 para terminar: "))  
while n != -1:  
    v.append(n)  
    n = int(input("Ingrese un numero o -1 para terminar: "))  
largo = len(v)  
if largo == 0:  
    print("No se ingresaron valores")  
else:  
    mayor = v[0]  
    pos = 0  
    for i in range(largo):  
        if v[i] > mayor:  
            mayor = v[i]  
            pos = i  
    imprimirlista(v)  
    print("El máximo es",mayor,"y se encontró en la posición",pos)  
    print("Borrando el",mayor)  
    del v[pos]  
    imprimirlista(v)
```


NOVEDADES DEL EJEMPLO NRO 4

Función len()

Devuelve la longitud de una lista, es decir su cantidad de elementos.

Instrucción for

Es una alternativa a la instrucción while en ciclos que estén controlados por un contador. Pueden ser utilizados con listas o sin ellas.

NOVEDADES DEL EJEMPLO NRO 4

Función range()

Genera una secuencia de números enteros entre 0 y el valor del parámetro suministrado. Este valor final **nunca** está incluido.

range(5) è 0 1 2 3 4

range(10) è 0 1 2 3 4 5 6 7 8 9

NOVEDADES DEL EJEMPLO NRO 4

Instrucción del:

Permite borrar elementos de una lista.
También sirve para eliminar variables o listas completas.

del x *# Borra la variable x*

del lista[4] *# Borra el elemento de la
posición 4*

del lista *# Borra la lista completa*

EJEMPLO NRO 5

Objetivo:

Escribir una función para ingresar números enteros en una lista y devolverla como valor de retorno.

La cantidad de valores a leer se recibe como parámetro.

EJEMPLO NRO 5 – PRIMERA PARTE

```
def cargarlista(cuantos):
```

```
    lista = [ ]
```

```
    for elemento in range(cuantos):
```

```
        n = int(input("Ingrese un número entero: "))
```

```
        lista.append(n)
```

```
    return lista
```

EJEMPLO NRO 5 – SEGUNDA PARTE

Programa principal

```
cant = int(input("Ingrese la cantidad de  
elementos: "))
```

```
print( )
```

```
milista = cargarlista(cant)
```

```
print( )
```

```
print(milista)
```

EJEMPLO NRO 5 – SEGUNDA PARTE

Ingrese la cantidad de elementos: 4

Ingrese un número entero: 2

Ingrese un número entero: 7

Ingrese un número entero: 9

Ingrese un número entero: 1

[2, 7, 9, 1]

NOVEDADES DEL EJEMPLO NRO 5

Una función puede devolver una lista como valor de retorno.

Es posible imprimir directamente una lista sin necesidad de escribir un ciclo.

EJEMPLO NRO 6:

Uso de for y range()

```
# Imprimir los números del 1 al 100
```

```
for numero in range(1, 101):  
    print(numero, end=" ")
```

Cuando *range()* se usa con dos parámetros el primero indica el inicio de la secuencia, mientras que el segundo señala el final de la misma. El valor final no está incluido.

EJEMPLO NRO 7

range() con incremento:

```
# Imprimir los números impares del 1 al 100
```

```
for impar in range(1, 101, 2):  
    print(impar, end=" ")
```

Cuando *range()* se usa con tres parámetros, el tercero actúa como *incremento*. Con sólo dos parámetros el incremento es 1.

EJEMPLO NRO 8: INCREMENTO NEGATIVO

Imprimir los números del 100 al 1

```
for i in range(100, 0, -1):  
    print(i, end=" ")
```

Cuando el incremento es negativo el valor inicial debe ser mayor que el valor final.

EJERCITACIÓN

- Práctica 6:
Ejercicios 1 a 5