

# P1

Programación 1

UNIDAD 05  
CLASE 06

## ESTRUCTURA DE DATOS TIPO LISTA PARTE 1



- | Estructura datos tipo lista
- | Función append
- | Mayores y menores en una lista
- | Listas paralelas



- Al final de esta clase ya podrás:
- | Introducirte al trabajo con variables de tipo lista para la resolución de problemas.
  - | Realizar sencillas bases de datos a través de listas y utilizar funciones para mostrar datos en pantalla

ISSD

**-Des-**  
Desarrollo de  
Software

**MÓDULO  
DIDÁCTICO**  
2023

# Primeros pasos en el uso de listas

¡Bienvenidos a la sexta clase de Programación 1!

Esta clase se estructura en torno a cuatro temas:

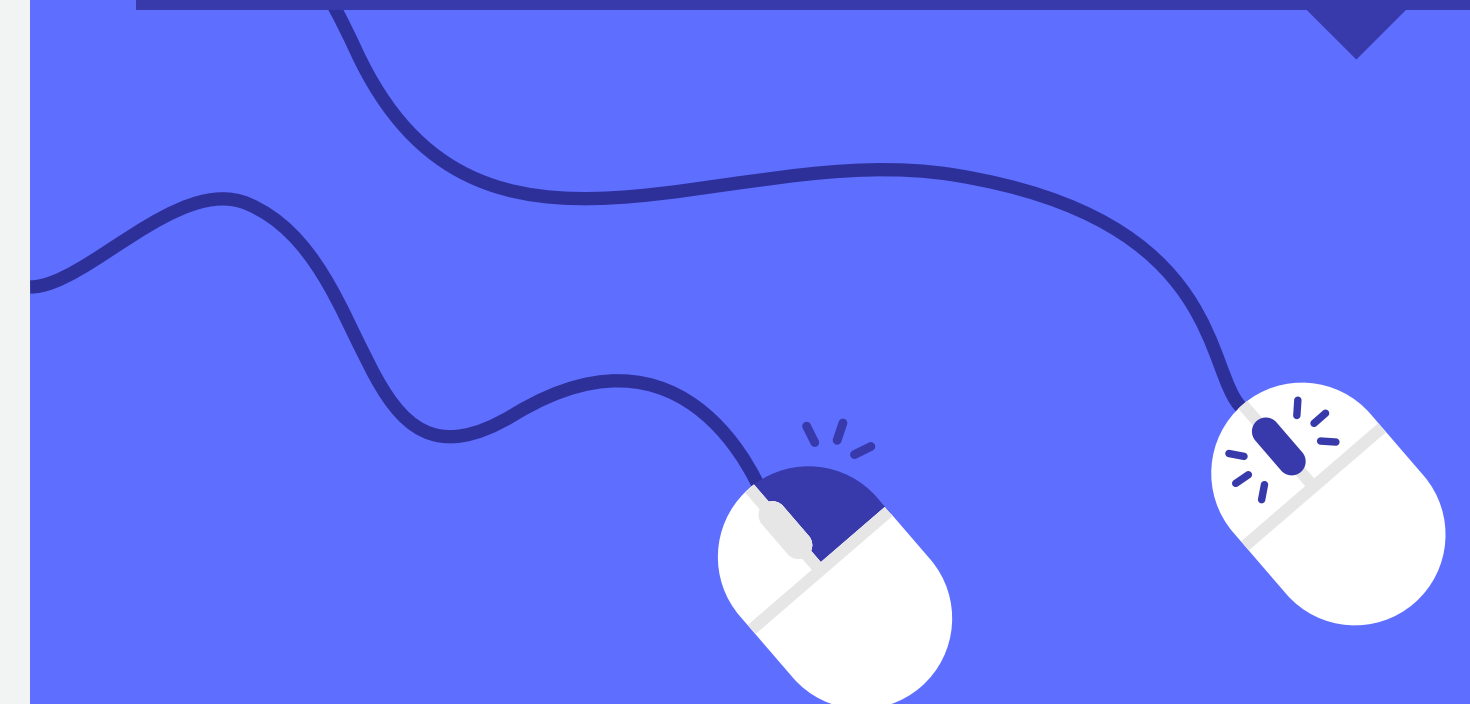
- | El primero de ellos te introducirá en estructuras de datos tipo lista.
- | En el segundo incorporaremos la función “**append**”.
- | En tercer lugar abordaremos los **métodos propios de las cadenas de caracteres**.
- | En cuarto y último lugar abordaremos las **listas pararelas** y su uso.

Como ya lo señalamos anteriormente, esta primera materia te introduce al mundo de la Programación, por eso lo más importante es que desarrolles la lógica, que seas capaz de identificar los elementos de un problema y que logres diseñar el algoritmo necesario para su resolución.

*¿Seguimos aprendiendo?*

Si querés realizar las pruebas del código en tu pc, te recomendamos hacer click en el botón ⬇. Se encuentra ubicado junto a cada código ejemplificado. De esta manera, podrás copiar y pegar el texto puro para que evites errores en la ejecución.

Si accedes al módulo desde un navegador, te recomendamos abrir los hipervínculos en una nueva pestaña presionando la rueda central del ratón, o bien usando el botón derecho y seleccionando la opción correspondiente.





Antes de comenzar con la clase, leé el siguiente artículo.

## Rol de Python en la exploración espacial

*Microsoft y la NASA se han unido para lanzar cursos en línea totalmente gratis sobre el uso y la aplicación de Python en problemas relacionados con la exploración espacial.*

De acuerdo a Stack Overflow, un sitio referente para programadores que cuenta con 40 millones de visitantes al mes, Python tiene un caso bastante solido para ser considerado el lenguaje de programación con mayor crecimiento actualmente.

El objetivo principal no es aprender Python en sí, sino comprender el papel que desempeña Python en las innovadoras soluciones que crea la NASA .

A través de la lente del descubrimiento espacial, *esta ruta de aprendizaje podría ser la chispa que prenda su pasión para aprender, detectar y crear de forma persistente, para que un día pueda ayudarnos a comprender un poco mejor el mundo más allá de la Tierra.*

## Potenciando a los jóvenes del mañana

Según un comunicado conjunto emitido por **Microsoft** y **NASA**, Python se ha convertido en un lenguaje de programación ampliamente utilizado en la comunidad científica y tecnológica debido a su facilidad de uso y su flexibilidad.

Su sintaxis legible y su amplia gama de bibliotecas y módulos lo convierten en una herramienta ideal para analizar datos y llevar a cabo tareas de programación complejas.

La iniciativa de capacitación, titulada "Introducción a Python en la exploración espacial de la NASA", está disponible en el sitio web de Microsoft Learn y ofrece una serie de cursos en línea gratuitos. Estos cursos están diseñados para brindar a los estudiantes y profesionales una base sólida en Python y su aplicación en la exploración del espacio. Los participantes aprenderán cómo utilizar Python para analizar datos de satélites, simular trayectorias de misiones espaciales y desarrollar algoritmos para diversas aplicaciones relacionadas con la exploración espacial.

## Unidos en la promoción de Python como lenguaje clave de exploración del espacio.

Según explica **Amy Hood**, vicepresidenta ejecutiva y directora financiera de Microsoft, *"Python es una herramienta esencial para la NASA y la comunidad espacial en general. Su flexibilidad y facilidad de uso lo convierten en un lenguaje de programación ideal para tareas complejas y análisis de datos en el ámbito de la exploración espacial"*.

Por su parte, **Jim Bridenstine**, administrador de la NASA, destacó la colaboración con Microsoft y afirmó: *"Estamos entusiasmados de asociarnos con Microsoft para fomentar el uso de Python en la comunidad espacial. Creemos que esta iniciativa permitirá a más personas adquirir las habilidades necesarias para participar en misiones espaciales y contribuir al futuro de la exploración del espacio"*.

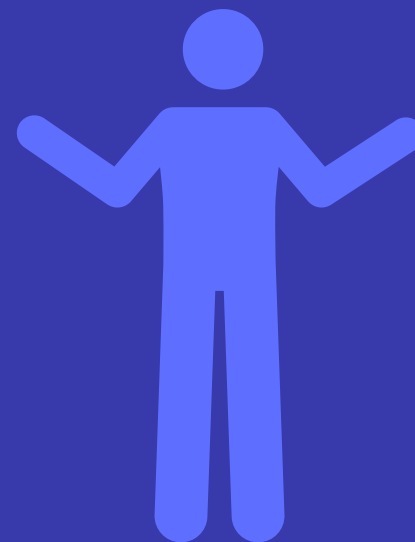
## De ayer a hoy... y al mañana

La utilización de Python en el contexto de la exploración espacial no solo beneficia a la NASA, sino que también abre oportunidades para la comunidad científica y tecnológica en general. Al ser un lenguaje de programación ampliamente utilizado y con una gran cantidad de recursos disponibles, Python permite a los desarrolladores e investigadores trabajar de manera más eficiente y colaborativa en proyectos relacionados con la exploración espacial.

En conclusión, Microsoft y la NASA están destacando la importancia de Python como lenguaje de programación en la exploración espacial. La iniciativa conjunta de capacitación tiene como objetivo fomentar el uso de Python y brindar a los interesados las habilidades necesarias para participar en misiones espaciales y contribuir al avance de la exploración del espacio. Con su facilidad de uso y su amplia gama de aplicaciones, Python continúa desempeñando un papel crucial en la innovación tecnológica

*Enseñamedeciencia.com - Marzo de 2023*

Python, el lenguaje de programación que estamos aprendiendo, ocupa un lugar privilegiado en la actualidad, no solo en las computadoras, sino en el mundo entero.



¿Te animás a seguir un paso más adelante?



# Estructura de datos tipo lista

Hasta ahora hemos trabajado con variables que permiten almacenar un único valor como podemos ver en los siguientes ejemplos:

```
edad=12
altura=1.79
nombre="juan"
```

*En Python existe un tipo de variable que permite almacenar una colección de datos y luego acceder por medio de un subíndice (similar a los string)*

## Creación de la lista por asignación

Para crear una lista por asignación debemos indicar sus elementos encerrados entre corchetes y separados por coma.

```
lista1=[10, 5, 3]           # lista de enteros
lista2=[1.78, 2.66, 1.55, 89,4] # lista de valores float
lista3=["lunes", "martes", "miercoles"] # lista de string
lista4=["juan", 45, 1.92]    # lista con elementos de distinto tipo
```

Si queremos conocer la cantidad de elementos de una lista podemos llamar a la función len:


```
lista1=[10, 5, 3]           # lista de enteros
print(len(lista1))          # imprime un 3
```



## Ejemplo 29

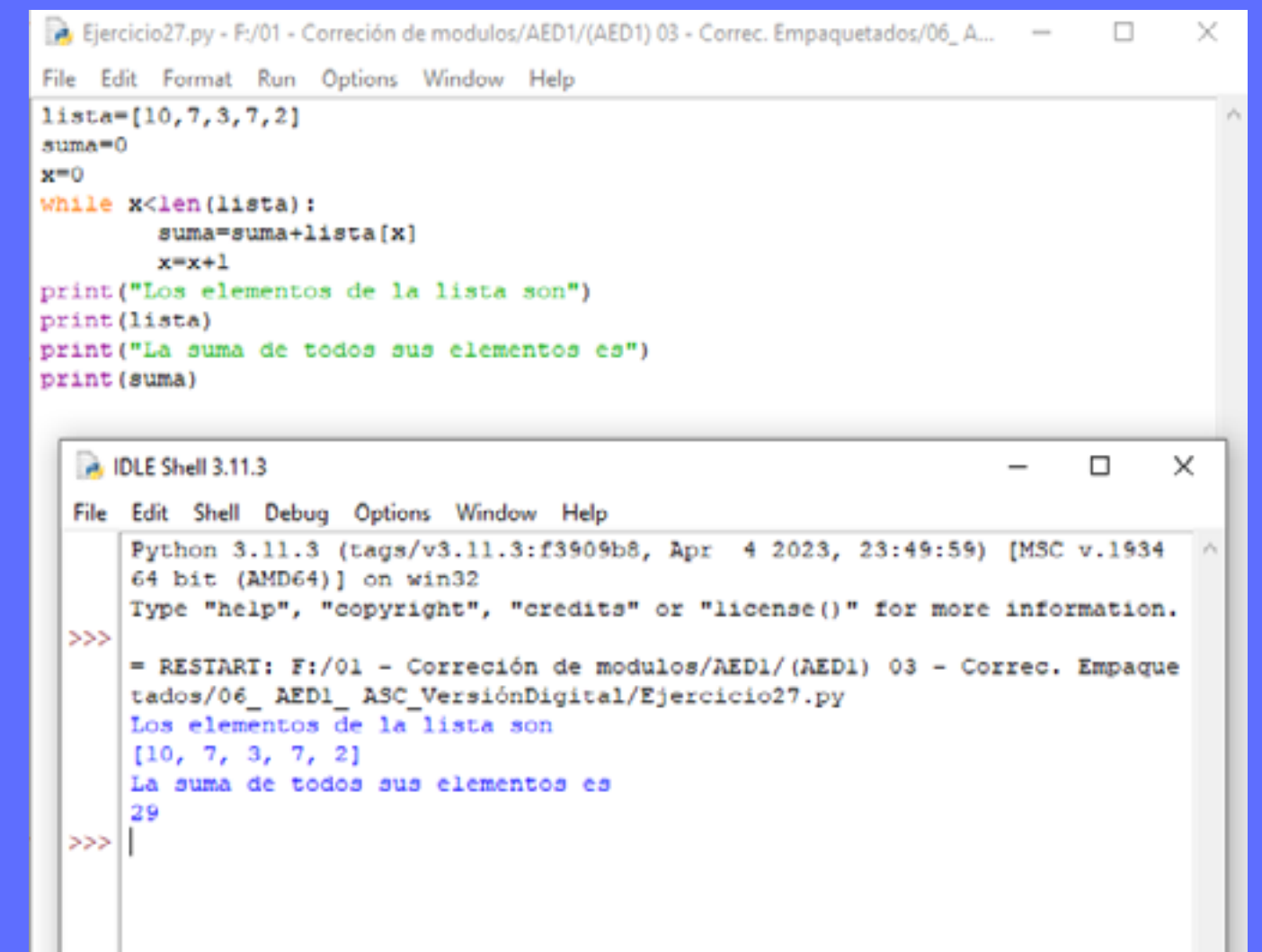
### Lista de elementos a sumar

Definir una lista que almacene 5 enteros. Sumar todos sus elementos y mostrar dicha suma.

| Programa: **|1|** 

```
lista=[10,7,3,7,2]
suma=0
x=0
while x<len(lista):
    suma=suma+lista[x]
    x=x+1
print("Los elementos de la lista son")
print(lista)
print("La suma de todos sus elementos es")
print(suma)
```

|1|



The screenshot shows a Python IDE window titled 'Ejercicio27.py'. The code in the editor is as follows:

```
lista=[10,7,3,7,2]
suma=0
x=0
while x<len(lista):
    suma=suma+lista[x]
    x=x+1
print("Los elementos de la lista son")
print(lista)
print("La suma de todos sus elementos es")
print(suma)
```

Below the editor is the IDLE Shell window, which shows the output of the program:

```
Python 3.11.3 (tags/v3.11.3:f3909b8, Apr  4 2023, 23:49:59) [MSC v.1934
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: F:/01 - Corrección de módulos/AED1/(AED1) 03 - Correc. Empaque
tados/06_AED1_ASC_VersiónDigital/Ejercicio27.py
Los elementos de la lista son
[10, 7, 3, 7, 2]
La suma de todos sus elementos es
29
>>>
```

| Primero definimos una lista por asignación con 5 elementos:

```
lista=[10,7,3,7,2]
```

| Definimos un acumulador para sumar los elementos de la lista y un contador para indicar qué posición de la lista accedemos:

```
suma=0  
x=0
```

| Mediante un ciclo while recorremos y sumamos cada elementos de la lista:

```
while x<len(lista):  
    suma=suma+lista[x]  
    x=x+1
```

| Cuando llamamos a la función print pasando como dato una lista luego se muestra en pantalla todos los elementos de la lista entre corches y separados por coma tal cual como lo definimos:

```
print("Los elementos de la lista son")  
print(lista)
```



| Finalmente mostramos el acumulador:

```
print("La suma de todos sus elementos es")  
print(suma)
```

## Ejemplo 30

### Lista de meses que devuelva el primer y último mes

**Definir una lista por asignación que almacene los nombres de los primeros cuatro meses del año. Mostrar el primer y último elemento de la lista solamente.**

| Programa: 

```
meses=["enero", "febrero", "marzo", "abril"]  
print(meses[0])           # se muestra enero  
print(meses[3])           # se muestra abril
```

Como queremos imprimir **solo el primer y último elemento** de la lista indicamos entre corchetes la posición de la lista del cual queremos rescatar el valor.

Si llamamos a print y pasamos solo el nombre de la lista se nos muestran todos los elementos:

```
print(meses)  
  
# se muestra ["enero", "febrero", "marzo", "abril"]
```



## Ejemplo 31

### Calculadora de promedios y alumnos utilizando listas

**Definir una lista por asignación que almacene en el primer componente de la lista el nombre de un alumno y en los dos siguientes sus notas.**

**Imprimir luego el nombre y el promedio de las dos notas.**

| Programa: 

```
lista=["ana", 7, 9]
print("Nombre del alumno:")
print(lista[0])
promedio=(lista[1]+lista[2])//2
print("Promedio de sus dos notas:")
print(promedio)
```

**Recordemos que el operador // se utiliza para dividir dos valores y retornar solo la parte entera.**

Como vemos en este problema los elementos de una lista pueden ser de distinto tipo, aquí tenemos el primer elemento de tipo string y los dos siguientes de tipo int.





### Desempeño 39

| Definir por asignación una lista con 8 elementos enteros. Contar cuántos de dichos valores almacenan un valor superior a 100.



### Desempeño 40

| Definir una lista por asignación con 5 enteros. Mostrar por pantalla solo los elementos con valores iguales o superiores a 7.



## Desempeño 41

Soluciones



| Definir una lista que almacene por asignación los nombres de 5 personas.  
Contar cuántos de esos nombres tienen 5 o más caracteres.

---



# Carga por teclado de sus elementos

Una lista en Python es una **estructura mutable** (es decir puede ir cambiando durante la ejecución del programa)

Hasta ahora hemos visto que podemos definir una lista por asignación indicando entre corchetes los valores a almacenar:

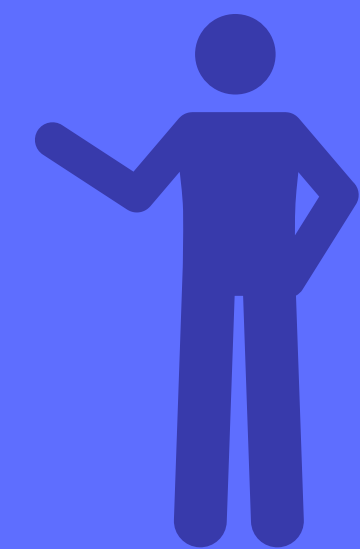
```
lista=[10, 20, 40]
```

Una lista, luego de definida, se puede modificar y podemos agregarle nuevos elementos al conjunto. La primera forma que veremos para incorporar elementos a nuestra lista es **el método append** a la lista y pasar como parámetro el nuevo elemento. A continuación te lo mostraremos en acción.

| Programa:

```
lista=[10, 20, 30]
print(len(lista))           # imprime un 3
lista.append(100)
print(len(lista))          # imprime un 4
print(lista[0])             # imprime un 10
print(lista[3])             # imprime un 100
```

Al aplicar la función `append(100)` incorporamos el elemento al final de la lista. Al final del programa, la lista queda así: `[10, 20, 30, 100]`



| Primero definimos una lista con tres elementos:

```
lista=[10, 20, 30]
```

| Imprimimos la cantidad de elementos que tiene la lista, en nuestro caso lo definimos con 3:

```
print(len(lista))          # imprime un 3
```

| Agregamos una nuevo elemento al final de la lista llamando al método append:

```
lista.append(100)
```

| Si llamamos nuevamente a la **función len** y le pasamos el nombre de nuestra lista ahora retornará un 4:

```
print(len(lista))          # imprime un 4
```

| Imprimimos el primer y cuarto elemento de la lista (recordá que se enumeran a partir del cero):

```
print(lista[0])            # imprime un 10  
print(lista[3])            # imprime un 100
```

## Ejemplo 32

### Cargar una lista de tamaño establecido

**Definir una lista vacía y luego solicitar la carga de 5 enteros por teclado y añadirlos a la lista. Imprimir la lista generada.**

| Programa: ⬇

```
lista=[]                                #definimos una lista vacia
#disponemos un ciclo de 5 vueltas
for x in range(5):
    valor=int(input("Ingrese un valor entero:"))
    lista.append(valor)
print(lista)                            #imprimimos la lista
```

| El algoritmo propuesto crea primero una lista vacía (debemos asignar los corchetes de apertura y cerrado sin contenido):

```
lista=[]
```

| Luego mediante un for (también podemos utilizar un while) solicitamos en forma sucesiva la carga de un entero por teclado y procedemos a agregarlo al final de la lista llamando al método append:

```
for x in range(5):  
    valor=int(input("Ingrese un valor entero:"))  
    lista.append(valor)
```

| Finalmente mostramos los elementos de la lista creada:

```
print(lista)
```



## Ejemplo 33

### Cargar una lista de tamaño variable

**Realizar la carga de valores enteros por teclado, almacenarlos en una lista. Finalizar la carga de enteros al ingresar el cero. Mostrar finalmente el tamaño de la lista.**

| Programa: ⬇

```
lista=[]
valor=int(input("Ingresar valor (0 para finalizar):"))
while valor!=0:
    lista.append(valor)
    valor=int(input("Ingresar valor (0 para finalizar):"))
print("Tamano de la lista:")
print(len(lista))
```

| En este problema la lista crecerá hasta que el operador ingrese el valor cero. La carga del primer valor se efectúa antes del ciclo while ya que la condición depende del valor ingresado:

```
valor=int(input("Ingresar valor (0 para finalizar:"))
```

| Luego dentro del ciclo while procedemos a agregar al final de la lista el valor ingresado y solicitar la carga del siguiente valor:

```
while valor!=0:  
    lista.append(valor)  
    valor=int(input("Ingresar valor (0 para finalizar:"))
```

| Cuando salimos del ciclo repetitivo procedemos a obtener el tamaño de la lista mediante la función len:

```
print(len(lista))
```



## Desempeño 42

| Almacenar en una lista los sueldos (valores float) de 5 operarios.  
Imprimir la lista y el promedio de sueldos.



## Desempeño 43

| Cargar por teclado y almacenar en una lista las alturas de 5 personas (valores float) Obtener el promedio de las mismas. Contar cuántas personas son más altas que el promedio y cuántas más bajas.



## Desempeño 44

Soluciones



| Una empresa tiene dos turnos (mañana y tarde) en los que trabajan 8 empleados (4 por la mañana y 4 por la tarde)  
Confeccionar un programa que permita almacenar los sueldos de los empleados agrupados en dos listas. Imprimir las dos listas de sueldos.



# Mayor y menor elemento de una lista

La búsqueda del mayor y menor elemento de una lista es una actividad muy común y práctica para resolver todo tipo de problemas al utilizar listas.

Para aplicar esta función es necesario que la lista tenga **valores del mismo tipo**; por ejemplo, números enteros o cadenas de caracteres. Podemos solicitar que se busque cual es mayor o menor alfabéticamente a una cadena de caracteres, pero no podemos buscar el mayor o menor de una lista si esta posee enteros y cadenas de caracteres al mismo tiempo.

A continuación te mostraremos su uso a través de un nuevo ejemplo.

## Ejemplo 34

### Encontrar el mayor valor de la lista

**Crear y cargar una lista con 5 enteros.**

**Implementar un algoritmo que identifique el mayor valor de la lista.**

| Programa: ⬇

```
lista=[]
for x in range(5):
    valor=int(input("Ingrese valor:"))
    lista.append(valor)
mayor=lista[0]
for x in range(1,5):
    if lista[x]>mayor:
        mayor=lista[x]
print("Lista completa")
print(lista)
print("Mayor de la lista")
print(mayor)
```

| En este problema comenzamos solicitando por teclado los 5 valores en la lista:

```
for x in range(5):  
    valor=int(input("Ingrese valor:"))  
    lista.append(valor)
```

| Para identificar el mayor de una lista primero tomamos como referencia el primer elemento, considerando a este en principio como el mayor de la lista:

```
mayor=lista[0]
```

| Seguidamente disponemos un for que se repita 4 veces esto debido a que no debemos controlar el primer elemento de la lista (recordar que la primer vuelta del for x toma el valor 1, luego el 2 y así sucesivamente hasta el 4):

```
for x in range(1,5):
```

| Dentro del for mediante una estructura condicional verificamos si el elemento de la posición x de la lista es mayor al que hemos considerado hasta este momento como mayor:

```
if lista[x]>mayor:  
    mayor=lista[x]:
```

| Como vemos en las dos líneas anteriores si el if se verifica verdadero actualizamos la variable mayor con el elemento de la lista que estamos analizando. Cuando finaliza el for, procedemos a mostrar el contenido de la variable "mayor" que tiene almacenado el mayor elemento de la lista:

```
print("Mayor de la lista")  
print(mayor)
```



## Ejemplo 35

### Encontrar el menor valor de la lista y su posición

**Crear y cargar una lista con 5 enteros cargados por teclado.**

**Implementar un algoritmo que identifique el menor valor de la lista y la posición donde se encuentra.**

| Programa: ⬇

```
lista=[]
for x in range(5):
    valor=int(input("Ingrese valor:"))
    lista.append(valor)
menor=lista[0]
posicion=0
for x in range(1,5):
    if lista[x]<menor:
        menor=lista[x]
        posicion=x
print("Lista completa")
print(lista)
print("Menor de la lista")
```



```
print(menor)
print("Posicion del menor en la lista")
print(posicion)
```

| Iniciamos una lista vacía y cargamos 5 elementos enteros:

```
lista=[]
for x in range(5):
    valor=int(input("Ingrese valor:"))
    lista.append(valor)
```

| Para identificar el menor y la posición de dicho valor en la lista definimos dos variables. La primera le cargamos el primer elemento de la lista, que es el que consideramos como menor hasta este momento:

```
menor=lista[0]
```

| Por otro lado la segunda variable almacena un cero que es la posición donde se encuentra el menor hasta este momento:

```
posicion=0
```

| Seguidamente disponemos un for que se repita 4 veces (que son la cantidad de elementos que nos faltan analizar de la lista):

```
for x in range(1,5):
```

| Dentro del for mediante un if verificamos si el elemento que estamos analizando de la lista es menor a la variable "menor":

```
if lista[x]<menor:
```

| En caso que sea menor a la que hemos considerado "menor" hasta este momento procedemos a actualizar la variable "menor" con el nuevo valor y también actualizamos la variable "posicion" con el valor del contador del for qué nos indica que posición de la lista estamos analizando:

```
menor=lista[x]  
posicion=x
```

| Cuando salimos del for mostramos la lista completa, el menor de la lista y la posición que tiene en la lista dicho menor:

```
print("Lista completa")  
print(lista)  
print("Menor de la lista")  
print(menor)  
print("Posicion del menor en la lista")  
print(posicion)
```



## Desempeño 45

| Ingresar por teclado los nombres de 5 personas y almacenarlos en una lista. Mostrar el nombre de persona menor en orden alfabético.



## Desempeño 46

Soluciones



| Cargar una lista con 5 elementos enteros.  
Imprimir el mayor y un mensaje si se repite dentro de la lista  
(es decir si dicho valor se encuentra en 2 o más posiciones en la lista)



# Lista paralelas

Podemos decir que **dos listas son paralelas cuando hay una relación entre las componentes de igual subíndice (misma posición) de una lista y otra**. Por ejemplo:

Nombres	Juan	Ana	Marcos	Pablo	Laura
Edades	12	21	27	14	21

Tenemos dos listas que ya hemos inicializado con 5 elementos cada una. En una se almacenan los nombres de personas y en la otra las edades de dichas personas.

Decimos que la lista nombres es paralela a la lista edades si en la componente 0 de cada lista se almacena información relacionada a una persona (Juan - 12 años).

**Es decir hay una relación entre cada componente de las dos listas.**

Esta relación la conoce únicamente el programador y se hace para facilitar el desarrollo de algoritmos que procesen los datos almacenados en las estructuras de datos.

## Ejemplo 36

### Lista de nombres y edades

**Desarrollar un programa que permita cargar 5 nombres de personas y sus edades respectivas. Luego de realizar la carga por teclado de todos los datos imprimir los nombres de las personas mayores de edad (mayores o iguales a 18 años)**

| Programa: ⬇

```
nombres=[]
edades=[]
for x in range(5):
    nom=input("Ingrese el nombre de la persona:")
    nombres.append(nom)
    ed=int(input("Ingrese la edad de dicha persona:"))
    edades.append(ed)
print("Nombre de las personas mayores de edad:")
for x in range(5):
    if edades[x]>=18:
        print(nombres[x])
```

| Definimos dos listas para almacenar los nombres y las edades de las personas respectivamente:

```
nombres=[]  
edades=[]
```

| Mediante un for cargamos en forma simultanea un elemento de cada lista, es decir un nombre de persona y la edad de dicha persona:

```
for x in range(5):  
    nom=input("Ingrese el nombre de la persona:")  
    nombres.append(nom)  
    ed=int(input("Ingrese la edad de dicha persona:"))  
    edades.append(ed)
```

| Para imprimir los nombres de la personas mayores de edad procedemos a analizar dentro de un for y mediante un if cada una de las edades almacenadas en la lista "edades", en el caso que su valor sea mayor o igual a 18 mostramos el elemento de la lista nombres de la misma posición:

```
edades.append(ed)  
for x in range(5):  
    if edades[x]>=18:  
        trint(nombres[x])
```



## Desempeño 47

| Crear y cargar dos listas con los nombres de 5 productos en una y sus respectivos precios en otra. Definir dos listas paralelas. Mostrar cuántos productos tienen un precio mayor al primer producto ingresado.



## Desempeño 48

| En un curso de 4 alumnos se registraron las notas de sus exámenes y se deben procesar de acuerdo a lo siguiente:

- a| Ingresar nombre y nota de cada alumno (almacenar los datos en dos listas paralelas)
- b| Realizar un listado que muestre los nombres, notas y condición del alumno. En la condición, colocar "Muy Bueno" si la nota es mayor o igual a 8, "Bueno" si la nota está entre 4 y 7, y colocar "Insuficiente" si la nota es inferior a 4.
- c| Imprimir cuantos alumnos tienen la leyenda “Muy Bueno”.



## Desempeño 49

Soluciones

| Realizar un programa que pida la carga de dos listas numéricas enteras de 4 elementos cada una. Generar una tercer lista que surja de la suma de los elementos de la misma posición de cada lista. Mostrar esta tercer lista.

Cuando te sientas listo, hacé click en el boton "Soluciones" para verificar tus respuestas





Durante la próxima clase seguiremos trabajando con listas y aprenderemos, entre otras cosas, cómo ordenar su contenido según diferentes criterios.

La complejidad de los algoritmos será mayor, pero a la vez veremos también cómo aumenta su utilidad y aplicación a diferentes problemas.

La potencia de la programación está en la gestión de grandes cantidades de datos, y las estructuras como las listas, nos posibilitan acceder a esta capacidad.

***¡Nos vemos!***

---

# Créditos

## Imágenes

Encabezado: Image by Steve Buissinne from Pixabay

<https://pixabay.com/photos/pawn-chess-pieces-strategy-chess-2430046/>

## Tipografía

Para este diseño se utilizó la tipografía *Source Sans Pro* diseñada por Paul D. Hunt.

Extraída de Google Fonts.

Si detectás un error del tipo que fuere (falta un punto, un acento, una palabra mal escrita, un error en código, etc.), por favor comunicate con nosotros a [correcciones@issd.edu.ar](mailto:correcciones@issd.edu.ar) e indicanos por cada error que detectes la página y el párrafo. Muchas gracias por tu aporte.

## Soluciones a los problemas

Te recomendamos utilizar esta sección luego de haber intentado por un largo tiempo la resolución y también para verificar tus soluciones.

### Solución al desempeño 39

| Programa: 

```
lista=[1000, 6000, 400, 23, 130, 400, 60, 2000]
cantidad=0
x=0
while x<len(lista):
    if lista[x]>100:
        cantidad=cantidad+1
    x=x+1

print("La lista esta constituida por los elementos:")
print(lista)
print("La cantidad de valores mayores a 100 en la lista son:")
print(cantidad)
```

## Solución al desempeño 40

| Programa: 

```
lista=[8,1,9,2,10]
x=0
print("Elementos de la lista con valores iguales o superiores a 7")
while x<len(lista):
    if lista[x]>=7:
        print(lista[x])
    x=x+1
```

## Solución al desempeño 41

| Programa: 

```
nombres=["juan", "ana", "marcos", "carlos", "luis"]
cantidad=0
x=0
while x<len(nombres):
    if len(nombres[x])>=5:
        cantidad=cantidad+1
    x=x+1
```

Volver a la clase



¿Terminaste de verificar  
tu código?  
Hacé click en el botón  
de arriba para continuar  
con el módulo



```
print("Todos los nombres son")
print(nombres)
print("Cantidad de nombres con 5 o mas caracteres")
print(cantidad)
```

## Solución al desempeño 42

| Programa: ⬇

```
sueldos=[]
suma=0
for x in range(5):
    valor=float(input("Ingrese el sueldo del operario:"))
    sueldos.append(valor)
    suma=suma+valor
print("Lista de sueldos")
print(sueldos)
promedio=suma/5
print("Promedio de sueldos")
print(promedio)
```

## Solución al desempeño 43

| Programa: 

```
alturas=[]
suma=0
for x in range(5):
    valor=float(input("Ingrese la altura:"))
    alturas.append(valor)
    suma=suma+valor
print("Las alturas ingresadas son")
print(alturas)
promedio=suma/5
print("El promedio de las alturas es")
print(promedio)
altas=0
bajas=0
for x in range(5):
    if alturas[x]>promedio:
        altas=altas+1
    else:
        if alturas[x]<promedio:
            bajas=bajas+1
```





```
print("La cantidad de personas mas bajas al promedio es")
print(bajas)
print("La cantidad de personas mas altas al promedio es")
print(alas)
```

## Solución al desempeño 44

| Programa: ⬇

```
sueldosman=[]
print("Sueldos turno manana")
for x in range(4):
    valor=float(input("Ingrese sueldo:"))
    sueldosman.append(valor)
sueldostar=[]
print("Sueldos turno tarde")
for x in range(4):
    valor=float(input("Ingrese sueldo:"))
    sueldostar.append(valor)
print("Turno manana")
print(sueldosman)
print("Turno tarde")
print(sueldostar)
```

[Volver a la clase](#)



## Solución al desempeño 45

| Programa: ⬇

```
nombres=[]
for x in range(5):
    nom=input("Ingrese nombre de persona:")
    nombres.append(nom)
nombremenor=nombres[0]
for x in range(1,5):
    if nombres[x]<nombremenor:
        nombremenor=nombres[x]
print("La lista completa de nombres ingresado son")
print(nombres)
print("El nombre menor en orden alfabetico es:")
print(nombremenor)
```

## Solución al desempeño 46

| Programa: 

```
lista=[]
for x in range(5):
    valor=int(input("Ingrese valor:"))
    lista.append(valor)
mayor=lista[0]
for x in range(1,5):
    if lista[x]>mayor:
        mayor=lista[x]
print("Lista completa")
print(lista)
print("Mayor de la lista")
print(mayor)
cantidad=0                # contamos cuantas veces se encuentra el mayor en la lista
for x in range(5):
    if lista[x]==mayor:
        cantidad=cantidad+1
if cantidad>1:
    print("El mayor se repite en la lista")
```

[Volver a la clase](#)



## Solución al desempeño 47

| Programa: ⬇

```
productos=[]
precios=[]
for x in range(5):
    nom=input("Ingrese el nombre del producto:")
    productos.append(nom)
    pre=int(input("Ingrese el precio de dicho producto:"))
    precios.append(pre)
cantidad=0
for x in range(1,5):
    if precios[x]>precios[0]:
        cantidad=cantidad+1
print("Cantidad de productos con un precio mayor al primer producto ingresado")
print(cantidad)
```

## Solución al desempeño 48

| Programa: 

```
nombres=[]
notas=[]
for x in range(4):
    nom=input("Ingrese nombre del alumno:")
    nombres.append(nom)
    no=int(input("Ingrese la nota de dicho alumno:"))
    notas.append(no)
cantidad=0
for x in range(4):
    print(nombres[x])
    print(notas[x])
    if notas[x]>=8:
        print("Muy Bueno")
        cantidad=cantidad+1
    else:
        if notas[x]>=4:
            print("Bueno")
        else:
            print("Insuficiente")
print("La cantidad de alumnos muy buenos son")
print(cantidad)
```

## Solución al desempeño 49

| Programa: 

```
lista1=[]
print("Carga de la primer lista")
for x in range(4):
    valor=int(input("Ingrese valor:"))
    lista1.append(valor)
lista2=[]
print("Carga de la segunda lista")
for x in range(4):
    valor=int(input("Ingrese valor:"))
    lista2.append(valor)
listasuma=[]
for x in range(4):
    suma=lista1[x]+lista2[x]
    listasuma.append(suma)
print("Lista resultante:")
print(listasuma)
```

[Volver a la clase](#)

