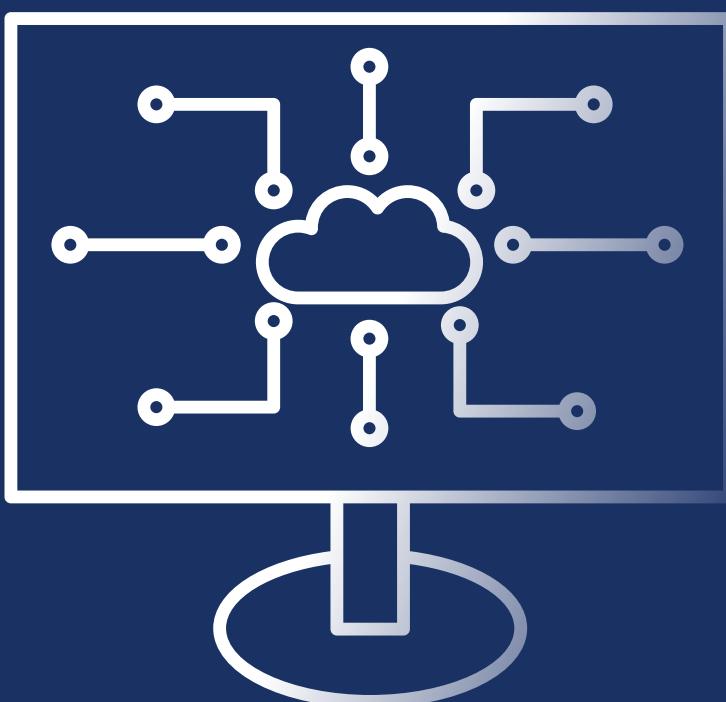


Programación I



Lic. Julia Monasterio
marmonasterio@uade.com.ar



Unidades

-  **Unidad 1:** Repaso de matrices, estructura, operaciones y uso en funciones
-  **Unidad 2:** Implementación y gestión en proyectos en Git
-  **Unidad 3:** Listas avanzadas, cadena de caracteres y expresiones regulares.
-  **Unidad 4:** Manipulación avanzada de diccionarios, tuplas y conjuntos.
-  **Unidad 5:** Excepciones y pruebas unitarias
-  **Unidad 6:** Archivos
-  **Unidad 7:** Recursividad

Clase N°3

TEMAS

- Uso de for con listas
- Practica de listas
- Ejercitación de matrices
- Listas por comprension
- Introducción a cadena de caracteres

Comparación de listas

Comparación de listas

- Las listas pueden ser comparadas como cualquier otra variables
- La comparación se realiza **elemento a elemento**
- **Python** compara las listas siguiendo un enfoque **lexicografico** (similar a las palabras de un diccionario) . Por ejemplo si quiero saber si una lista es mayor a otra lista, toma el primer elemento de ambas, y si ese es mayor, **la primera lista se considera mayor y no se compara el resto.**

Comparación de listas

```
print([2,3]>[1,4])
```

True

```
print([2,3]>[2,4])
```

False

```
print([2,4,6]>[2,4])
```

True

- En el ultimo ejemplo, cuando Python se queda sin elementos en una de las listas y la otra lista todavía tiene elementos, considera que la lista mas larga es “mayor”

Uso de foren listas

Uso de for con listas

- Ejemplo:

```
vocales = ['a', 'e', 'i', 'o', 'u']

for letra in vocales:
    print(letra, end=" ")
```

En cada ciclo **letra** va a tomar un valor de la lista de vocales.

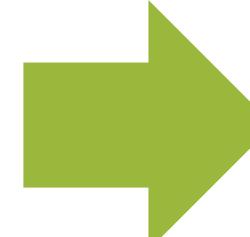
En otros lenguajes esto se llama **for each**

Uso de for con listas

- Pueden usarse una rebanada para recorrer la lista parcialmente

```
vocales = ['a','e','i','o','u']

for letra in vocales[1:4]:
    print(letra, end=" ")
```



```
e i o
```

Recordar que las rebanadas nunca contienen el
ultimo subíndice

Uso de `for` con listas - Función enumerate

- Función **enumerate** cuando requerimos obtener tanto el índice como el valor de los elementos
- Se utilizan cuando se necesita tanto el **índice** como el **elemento**
- Devuelve una tupla (ya lo veremos mas adelante) con el indice y el elemento en cada iteración (**<subindice>,<elemento>**)

Uso de for con listas - Función enumerate

- Ejemplo:

Tenemos una lista de notas, y queremos sumar cinco puntos extra a todas las notas que tienen un valor por debajo de 70.

```
notas=[75,80,60,90,45]
```

```
PUNTOS_EXTRA=5
```

```
for i, nota in enumerate(notas):  
    if nota<70:  
        notas[i]= nota+PUNTOS_EXTRA
```

```
Notas originales [75, 80, 60, 90, 45]  
Notas actualizadas [75, 80, 65, 90, 50]
```

Instrucción pass

Instrucción pass

- La instrucción **pass** **no hace nada**
- **Puede usarse en situaciones especiales o para representar código aun no escrito, es decir no implementado**
- Por ejemplo en su TPO si tienen funciones, o bloques que aun no han implementado pueden utilizar la instrucción **pass**

Instrucción pass

- Ejemplo:

```
def calcularSueldoNeto(sueldoBruto):  
    pass
```

Instrucción pass

- No hay que hacer un uso abusivo de la instrucción **pass**

```
nota=10

if nota>=4:
    pass
else:
    print("Recuperas")
```



Como evitarián el uso de **pass**?

Práctica en clase

Crear un programa que solicite al usuario ingresar un número y busque la cantidad de ocurrencias que existen sobre una lista ya cargada



Resolución ejercicio

```
def contarOcurrencias(lista,valorBuscado):  
  
    if valorBuscado in lista:  
        return lista.count(valorBuscado)  
    else:  
        -1
```

```
#Programa Principal  
  
lista=[45,56,76,44,55,78,34,43,31,43]  
  
valorBuscado=43  
  
print(f"La cantidad de veces que aparece {valorBuscado} en la lista es: {contarOcurrencias(lista,  
valorBuscado)}")
```

Práctica en clase

Crear un programa que intercambie el primer y último elemento de una lista precargada



Resolución ejercicio

```
def intercambiarValorLista(lista):
    aux = lista[0]

    lista[0]=lista[len(lista)-1]
    lista[len(lista)-1]=aux
```

```
#Programa Principal
lista=[1,4,5,6]
print("Lista original ",lista)
intercambiarValorLista(lista)
print("Lista intercambiada",lista)
```

Matrices

Matrices

- Una **matriz** es una estructura de datos formada por filas y columnas

	Columna 0	Columna 1	Columna 2	Columna 3
Fila 0	$A[0][0]$	$A[0][1]$	$A[0][2]$	$A[0][3]$
Fila 1	$A[1][0]$	$A[1][1]$	$A[1][2]$	$A[1][3]$
Fila 2	$A[2][0]$	$A[2][1]$	$A[2][2]$	$A[2][3]$

$A[1][0]$

A: nombre de matriz **[1]** subíndice de fila **[0]:** subíndice de columna

Instrucción pass

- A diferencia de la mayoría de los lenguajes de programación.
Python no tiene soporte para matrices
- **Se simulan construyendo lista de listas**, es decir es una lista donde sus elementos, son listas
- Se necesitan dos subíndices, el primero se refiere a las filas y el segundo a las columnas. **Ambos comienzan en 0**

Matrices

- Creación de matrices

1. Forma estática:

```
matriz=[[0,0,0],  
        [0,0,0],  
        [0,0,0]]
```

```
print(matriz)
```

```
[[0, 0, 0], [0, 0, 0], [0, 0, 0]]
```

Matrices

- Creación de matrices

2. Forma dinámica:

```
filas=3
columnas=4
matriz=[]

for f in range (filas):
    matriz.append([])
    for c in range(columnas):
        matriz[f].append(0)

print(matriz)
```

```
[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
```

Matrices

- Creación de matrices

2. Forma dinámica utilizando el replicar:

```
filas=3
columnas=4
matriz=[]

for f in range (filas):
    matriz.append([0]*columnas)

print(matriz) | [[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
```

Recorrer una matriz

- Generalmente el recorrido de una matriz se realiza mediante el uso de un ciclo for anidado, que vaya accediendo uno por uno a los elementos que queremos acceder de la misma, tanto para la fila como para la columna.
- Al ser listas con listas en su interior, seguiremos teniendo acceso a las mismas funciones para **agregar, modificar y eliminar elementos como hemos visto anteriormente cuando trabajamos con listas de forma individual.**

Recorrer una matriz

```
for fila in range(0, len(matriz)):  
    for columna in range(0, len(matriz[fila])):  
        print("[", fila, "][", columna, "]", end=' ')  
        print("->", matriz[fila][columna])
```

[0][0] -> 0
[0][1] -> 0
[0][2] -> 0
[0][3] -> 0
[1][0] -> 0
[1][1] -> 0
[1][2] -> 0
[1][3] -> 0
[2][0] -> 0
[2][1] -> 0
[2][2] -> 0
[2][3] -> 0

Práctica en clase

Escribe una función que tome dos números enteros como argumentos (filas y columnas) y devuelva una matriz completa con números aleatorios entre 0 y 10



Resolución ejercicio

```
import random  
def crearMatriz(filas,columnas):  
    matriz=[]  
    for i in range(filas):  
        fila=[]  
        for j in range(columnas):  
            numero= random.randint(1,10)  
            fila.append(numero)  
        matriz.append(fila)  
    return matriz
```

```
#Programa Principal  
filas = int(input("Ingrese la cantidad de filas que quiere que tenga su matriz\n"))  
columnas = int(input("Ingrese la cantidad de columnas que quiere que tenga su matriz\n"))  
  
print(crearMatriz(filas,columnas))
```

Práctica en clase

Crear una función que tome una matriz numérica y devuelva una nueva matriz donde cada elemento se eleve al cuadrado



Resolución ejercicio

```
def elevarElementosAlCuadrado(matriz):  
    nuevaMatriz=[]  
  
    for i in range(len(matriz)):  
        fila=[]  
        for j in range(len(matriz[i])):  
            calculo=matriz[i][j]**2  
            fila.append(calculo)  
        nuevaMatriz.append(fila)  
    return nuevaMatriz
```

Práctica en clase

Crea un programa que pida un número al usuario un número de mes (por ejemplo, el 4) y diga cuántos días tiene (por ejemplo, 30).

Debes usar una matriz para su parametrización y una función para la recuperación del dato.

Asumir que febrero tiene 28



Resolución ejercicio - Función obtener meses

```
def obtenerDiasMes(mes, matrizMeses):
    for i in range(len(matrizMeses)):
        if matrizMeses[i][0]==mes:
            return matrizMeses[i][1]
```

Resolución ejercicio - Creación de matriz

```
matrizMeses=[[1,31],  
             [2,28],  
             [3, 31],  
             [4, 30],  
             [5, 31],  
             [6, 30],  
             [7, 31],  
             [8, 31],  
             [9, 30],  
             [10, 31],  
             [11, 30],  
             [12, 31]]
```

Resolución ejercicio - Invocación función

```
#Programa Principal
mes= int(input("Ingrese el mes del cual desea saber la cantidad de dias\n"))

if mes<0 or mes>12:
    print("Mes invalido")
else:
    print(f"La cantidad de dias que tiene el mes {mes} es de {obtenerDiasMes(mes,matrixMeses)}")
```

Práctica en clase

Crea una matriz con un tamaño que el usuario le indique por teclado (puede ser 6×4 , 7×2 , etc.) pero como máximo podrá contener 10×10 valores y como mínimo 2×2 .

Crear una función para la cargar de los valores y, por último, otro procedimiento para visualizar los resultados.

Los valores para cargar deberán ser números positivos entre 0 y 100, siendo éstos generados al azar



Resolución ejercicio - Función para cargar matriz

```
import random

def cargarMatriz(filas,columnas):
    matriz=[]

    for fila in range(filas):
        fila=[]
        for columna in range(columnas):
            numero= random.randint(0,100)
            fila.append(numero)
        matriz.append(fila)
    return matriz
```

Resolución ejercicio - Función imprimir matriz

```
def mostrarDatosMatriz(matriz):
    for fila in range(len(matriz)):
        for columna in range(len(matriz[fila])):
            print(f"[Fila] {fila} [Columna] {columna} -> {matriz[fila][columna]}")

#Solo imprimir valores
def visualizarMatriz(matriz):
    for fila in matriz:
        for columna in fila:
            print(columna, end="\t")
    print()
```

Resolución ejercicio - Programa Principal

```
#Programa Principal
```

```
filas= int(input("Ingrese la cantidad de filas que desea que tenga la matriz\n"))
columnas= int(input("Ingrese la cantidad de filas que desea que tenga la matriz\n"))

if filas >= 2 and filas <= 10 and columnas >= 2 and columnas <= 10:
    matrizGenerada = cargarMatriz(filas, columnas)
    print("\nMatriz generada:")
    mostrarDatosMatriz(matrizGenerada)
    visualizarMatriz(matrizGenerada)
else:
    print("El tamaño de la matriz debe estar entre 2x2 y 10x10.")
```

Listas por comprehensión

Listas por comprensión

- La comprensión es una construcción sintáctica para crear listas a partir de los elementos de otros elementos iterables (como otras listas)
- Se escribe de una forma mas concisa

Listas por comprensión - Sintaxis de construcción

<lista resultante> = [**<expr>** for **<elem>** in **<colección>**] **if <condicion>**

- **<expr>** : representa alguna operación que se aplica a cada elemento **<elem>** de **<colección>**. Transformación del elemento. El resultado de esta expresión se agregara a lista resultante.
- Los corchetes son necesarios para crear la lista. También puede utilizarse **list()**
- **if<condicion>** es opcional

Ejemplo

- Crear una lista con una secuencia de 1 a 100.

Sin listas por comprensión

```
lista=[]

for i in range(1,101):
    lista.append(i)

print(lista)
```

Con listas por comprensión

```
lista=[elemento for elemento in range(1,101)]

print(lista)
```

Ejemplo 2

- Definir una lista con 5 valores enteros, luego a partir de la primer lista generar una segunda lista con los valores elevados al cuadrado

Sin listas por comprensión

```
lista=[5,6,7,8,2]
lista2=[]
for i in range(len(lista)):
    lista2.append(lista[i]**2)

print(f"Lista original: {lista}")
print(f"Lista con valores al cuadrado {lista2}")
```

Con listas por comprensión

```
lista=[5,6,7,8,2]
lista2= [elemento**2 for elemento in lista]

print(f"Lista original: {lista}")
print(f"Lista con valores al cuadrado {lista2}")
```

Ejemplo 3

- Generar una lista con todos los valores multiplos de 8 comprendidos entre 1 y 500

Sin listas por comprensión

```
lista=[]

for i in range(1,501):
    if i%8==0:
        lista.append(i)

print(f"Lista multiplos de 8: {lista}")
```

Con listas por comprensión

```
lista=[elemento for elemento in range(1,501) if elemento%8==0]

print(f"Lista multiplos de 8 entre 1 y 500: {lista}")
```

Práctica en clase

Desarrollar cada una de las siguientes funciones y escribir un programa que permita verificar su funcionamiento imprimiendo la lista luego de invocar a cada función:

- a. Cargar una lista con números al azar de cuatro dígitos. La cantidad de elementos también será un número al azar de dos dígitos. Realice la composición de la lista por comprensión y de la forma habitual (tendrá dos funciones distintas).



Práctica en clase

- b. Calcular y devolver la sumatoria de todos los elementos de la lista anterior.
- c. Eliminar todas las apariciones de un valor en la lista anterior. El valor a eliminar se ingresa desde el teclado y la función lo recibe como parámetro. Utilice comprensión de listas para resolverlo.



Práctica en clase

d. Determinar si el contenido de una lista cualquiera es capicúa, sin usar listas auxiliares. Un ejemplo de lista capicúa es [50,17,91,17,50]



Función para cargar lista sin comprehension

```
import random

def cargarListaSinComprehension():
    lista=[]
    cantElementos= random.randint(10,99)

    for i in range(0,cantElementos):
        numero= random.randint(1000,9999)
        lista.append(numero)

    return lista
```

Función para cargar lista con comprensión

```
def cargarListaConComprension():
    lista=[random.randint(1000,9999) for i in range(random.randint(10,99))]
    return lista
```

Función para sumar los valores de una lista

b. Calcular y devolver la sumatoria de todos los elementos de la lista anterior.

```
def sumarValoresLista(lista):  
    return sum(lista)
```

Función para eliminar valor de una lista

c. Eliminar todas las apariciones de un valor en la lista anterior. El valor a eliminar se ingresa desde el teclado y la función lo recibe como parámetro. Utilice comprensión de listas para resolverlo.

```
def eliminarValorLista(lista,elementoAeliminar):  
    lista=[elemento for elemento in lista if elemento!=elementoAeliminar]  
    return lista
```

Función para determinar si lista es capicua

d. Determinar si el contenido de una lista cualquiera es capicúa, sin usar listas auxiliares. Un ejemplo de lista capicúa es [50,17,91,17,50]

```
def esCapicua(lista):  
    return lista==lista[::-1]
```

Se esta utilizando slice o rebanadas, recordemos que las rebanadas tenían **inicio:fin:paso**, en este caso no se esta utilizando ni inicio, ni fin, solamente paso.

Al colocar -1 la lista se recorre de atrás hacia delante

Cadenas de Caracteres

Cadena de caracteres

- Una cadena de caracteres (string) es un conjunto de 0 o mas caracteres
- Una **cadena vacía** es que tiene **cero** caracteres
- Python utiliza la codificación UTF-8 para representar caracteres.

Cadena de caracteres - UTF-8

- Es un estándar de codificación de caracteres que permite representar texto en casi cualquier sistema de escritura del mundo utilizando secuencia de bytes.
- UTF-8 soporta caracteres regionales como la ñ, las vocales con tilde, etc.

Cadena de caracteres

- Las cadenas de caracteres se tienen que encerrar entre comillas dobles o simples (indistintas cualquiera de las dos)
- Es importante que si comenzamos con un **tipo de comilla terminemos con ese tipo de comilla**
- Esto permite utilizar el otro tipo de comillas dentro de la cadena

Cadena de caracteres

```
cadena1="Lunes"  
cadena2='Martes'
```

```
marca="Papa John's"
```

```
frase='Juan dijo "VENI PARA ACA" y se puso rojo'
```

Cadena de caracteres

- Una cadena extensa puede ser distribuidas en varias líneas **al momento de escribirse**, es decir para evitar que el editor se desplace hacia la derecha se coloca \ una barra invertida

```
poema="Este poema lo escribio Borges \
quien nacio en Argentina"
```

```
print(poema)
```

```
Este poema lo escribio Borges quien nacio en Argentina
```

Cadena de caracteres

- Se utilizan tres juegos de comillas para que la cadena retenga el formato dado por el programador y puede extenderse por múltiples líneas

```
cancion= '''Soy un viajante, un caminante  
Con la vida por delante  
Y llevo tanto, tanto amor dentro de mí'''
```

```
print(cancion)
```

```
Soy un viajante, un caminante  
Con la vida por delante  
Y llevo tanto, tanto amor dentro de mí
```

Cadena de caracteres

- El especificador de conversión de una cadena es **%s**

```
nombre=input("Ingrese su nombre\n")
print("%s es su nombre!"%nombre)
```

```
Ingrese su nombre
Juan
Juan es su nombre!
```

Cadena de caracteres

Secuencias de escape

- Las secuencias de escape se utilizan para representar caracteres con significados especiales
- Toda secuencia de escape comienza con una barra invertida \
- **El carácter que se escriba después de ella establece el significado de la secuencia de escape**
- **Cuando se imprime una secuencia de escape lo que vamos a visualizar es el efecto que esa secuencia de escape provoca**

Cadena de caracteres

Secuencias de escape

- Ejemplos:
 - \n : Salto de linea
 - \t : tabulador
 - \' : comilla simple
 - \" : comilla doble
 - \\ : barra invertida

Cadena de caracteres

Secuencias de escape

- Ejemplos:
 - \n : Salto de linea
 - \t : tabulador
 - \' : comilla simple
 - \" : comilla doble
 - \\ : barra invertida

Cadena de caracteres

Ejemplo - Secuencia de escape

```
print("Lunes \n Martes \t Miercoles \"")
```

Lunes

Martes

Miercoles

Cadena de caracteres

Cadenas crudas

- Una cadena cruda (**raw**) se crea escribiendo la letra **r** antes de abrir las comillas.
- Es una cadena donde **no se interpretan las secuencias de escape**
- **Se utilizan basicamente para rutas o paths**

```
ruta=r"C:\Users\nuevo\UADE\Segundo cuatrimestre\Programación 1"  
print(ruta)
```

Resumen de la clase

- Comparación de listas
- Uso de for en listas
- Instrucción pass
- Matrices
- Listas por comprensión
- Cadena de caracteres
- Secuencias de escape

Muchas gracias!

Consultas?

