Resumen Certamen 2

INF129

Autor: Eduardo Pino H.

22 de junio de 2025

Índice

1. Laboratorio 6: Listas

1.1. ¿Qué es una lista?

Una **lista** en Python es una colección ordenada, mutable y heterogénea (puede tener elementos de cualquier tipo).

Definición:

```
numeros = [1, 2, 3]
nombres = ["Ana", "Luis", "Tomas"]
mixta = [True, 3.14, "texto", 9]
```

¿Por qué usar listas?

- Permiten guardar varios valores en una sola variable.
- Son mutables: se pueden modificar, agregar o eliminar elementos.
- Se pueden recorrer fácilmente con ciclos.
- Pueden contener listas anidadas (listas de listas).

1.2. Acceso y modificación de elementos

```
print(nombres[1])     # Luis
print(nombres[-1])    # Tomas (ultimo)
numeros[0] = 10       # Modifica primer elemento: [10, 2, 3]
```

1.3. Crear listas vacías

```
lista1 = []
lista2 = list()
```

1.4. Slicing (rebanado)

```
sub = nombres[0:2]    # ['Ana', 'Luis']
```

1.5. Copiar listas

Incorrecto:

```
b = a         # b y a apuntan a la misma lista
```

Correcto:

```
b = a[:]      # Copia con slicing
b = list(a)   # Copia con list()
```

1.6. Operaciones básicas

- Concatenar: `a + b`
- Repetir: `a * 3`
- Eliminar por índice: `del a[2]`
- Verificar existencia: `"Ana" in nombres`

1.7. Principales métodos y funciones

```
lista.append(x)       # Agrega x al final
lista.remove(x)       # Elimina la primera aparicion de x
lista.insert(i, x)    # Inserta x en la posicion i
lista.count(x)        # Cuenta cuantas veces aparece x
lista.index(x)        # Primer indice de x
lista.sort()          # Ordena la lista
lista.reverse()       # Invierte el orden
len(lista)            # Largo de la lista
sum(lista)            # Suma (solo numericos)
min(lista), max(lista) # Minimo/maximo (solo numericos)
```

1.8. Recorriendo listas

Con for:

```
for nombre in nombres:
    print(nombre)
```

Con índice:

```
for i in range(len(nombres)):
    print(nombres[i])
```

Con while:

```
i = 0
while i < len(nombres):
    print(nombres[i])
    i += 1
```

1.9. Listas de listas (listas anidadas)

Definición:

```
matriz = [
    [1, 2, 3],
    [4, 5, 6],
```

```
    [7, 8, 9]
]
```

Acceso a elementos:

```
print(matriz[1][2])  # 6 (segunda fila, tercer columna)
```

Recorrido con for anidados:

```
for fila in matriz:
    for elem in fila:
        print(elem)
```

Recorrido por índice:

```
for i in range(len(matriz)):
    for j in range(len(matriz[i])):
        print(matriz[i][j])
```

2. Laboratorio 7: Diccionarios

2.1. ¿Qué es un diccionario?

Un **diccionario** en Python es una colección de pares **llave: valor** (key: value), desordenada y mutable. Permite asociar información a una clave, en vez de a un índice como las listas.

Definición:

```
notas = {
    "Sebastian": 97,
    "Camila": 100,
    "Victor": 89
}
```

¿Para qué sirven los diccionarios?

- Permiten acceder directamente a un valor usando una llave (nombre, código, etc).
- Son ideales para guardar datos que se identifican por algo único (por ejemplo: rut, nombre de usuario, código de producto).
- Son útiles para contar, clasificar o mapear información.

2.2. Diferencias entre listas y diccionarios

- **Listas:** Acceso por posición (índice). El orden importa.
- **Diccionarios:** Acceso por llave. El orden no importa. No puede haber llaves repetidas.

Ejemplo comparación:

```python
# Lista de precios (orden importa)
precios_lista = [1200, 1500, 2300]

# Diccionario de precios (llave = producto)
precios = {"pan": 1200, "leche": 1500, "queso": 2300}
```

2.3. Crear y acceder a diccionarios

```python
dicc1 = {}            # Diccionario vacio
dicc2 = dict()        # Otra forma

# Diccionario con valores iniciales
personas = {"Ana": 20, "Luis": 22}
print(personas["Ana"])   # 20
```

2.4. Agregar, modificar y eliminar elementos

```python
# Agregar o modificar
personas["Tomas"] = 19
personas["Ana"] = 21    # Modifica el valor de "Ana"

# Eliminar elemento
del personas["Luis"]
```

2.5. Funciones y operadores útiles

```python
len(diccionario)        # Cantidad de pares llave:valor
"Tomas" in personas     # True si existe la llave "Tomas"
```

2.6. Recorrer diccionarios

```python
# Recorrer llaves
for llave in personas:
    print(llave)
```

2.7. Ejemplo práctico

```python
# Crear un diccionario de pokemon por tipo
pokedex = {
    "agua":   ["Squirtle", "Psyduck", "Totodile", "Mudkip"],
    "fuego":  ["Charmander", "Vulpix", "Torchic", "Chimchar"],
```

```
    "planta": ["Bulbasaur", "Oddish", "Chikorita", "Treecko"]
}

print(pokedex["fuego"][2])    # "Torchic"
```

Notas

- Las llaves deben ser inmutables.
- No pueden haber llaves repetidas.
- Los valores pueden ser de cualquier tipo (incluyendo listas y otros diccionarios).

3. Laboratorio 8: Procesamiento de Texto y Archivos

3.1. Métodos avanzados para strings

replace:

```
original = "me gusta programar"
nuevo = original.replace(" ", "_")  # me_gusta_programar
```

split:

```
oracion = "Si, me gusta Python"
print(oracion.split())  # ['Si,', 'me', 'gusta', 'Python']

oracion = "Si, me gusta Python"
print(oracion.split(","))  # ['Si', ' me gusta Python']
```

join:

```
lista1 = ["iwi", "1", "3", "1"]
print("".join(lista1))  # iwi131

lista2 = ["1", "2", "3"]
print("->".join(lista2))  # 1->2->3
```

format:

```
s = "Soy {0} y vivo en {1}"
print(s.format("Ana", "Santiago"))  # Soy Ana y vivo en Santiago

s = "Soy {nombre} y vivo en {lugar}"
print(s.format(nombre="Ana", lugar="Santiago"))
print(s.format(lugar="Santiago", nombre="Ana"))
```

strip:

```
linea = "\nHola mundo "
print(linea.strip())  # Hola mundo
```

Caracteres especiales:

```
print("Hola\nMundo")

a = "1\t2\t3\t4"
len(a)  # 7

b = "1\n2\n3"
len(b)  # 5

print("Nombre:\tAna")
print("Edad:\t19")
```

3.2. Lectura y escritura de archivos

Lectura de archivos:

```
archivo = open("algun archivo.txt", "r")
for linea in archivo:
    linea_limpia = linea.strip()
    print(linea_limpia)
archivo.close()
```

Lectura y procesamiento de archivos .csv:

```
archivo = open("datos.csv", "r")
for linea in archivo:
    linea_limpia = linea.strip()
    lista = linea_limpia.split(",")
    print(lista)
archivo.close()
```

Construcción de un diccionario desde un .csv:

```
diccionario = {}
archivo = open("datos.csv", "r")
for linea in archivo:
    linea_limpia = linea.strip()
    lista = linea_limpia.split(",")
    # (logica para rellenar diccionario)
archivo.close()
```

Escritura de archivos:

```
archivo = open("resumen.txt", "w")
archivo.write("Resumen de compras\n")
archivo.write("Producto: Pan\n")
```

```
archivo.write("Cantidad: 3\n")
archivo.close()
```

Guardar datos de mascotas:

```
archivo = open("mascotas.txt", "w")
flag = True
while flag:
    nombre = input("Ingrese nombre de la mascota: ")
    if nombre == "salir":
        flag = False
    else:
        especie = input("Ingrese especie: ")
        edad = input("Ingrese edad: ")
        archivo.write("{},{},{}\n".format(nombre, especie, edad))
archivo.close()
```

Extraer datos de un .csv usando split con otro separador:

```
archivo = open("alumnos.csv", "r")
for linea in archivo:
    lista = linea.strip().split(";")
    print(lista)
archivo.close()
```

Notas:

- Siempre cerrar el archivo con `archivo.close()`.
- Si el archivo no existe y se abre en modo "r", arroja error.
- El modo "w" sobrescribe el archivo si ya existe.
- Para archivos .csv se pueden usar distintos separadores: `split(",")`, `split(";")`, etc.