OPTATIVO 1 – PYTHON I

Prof. Ing. Aaron Zárate

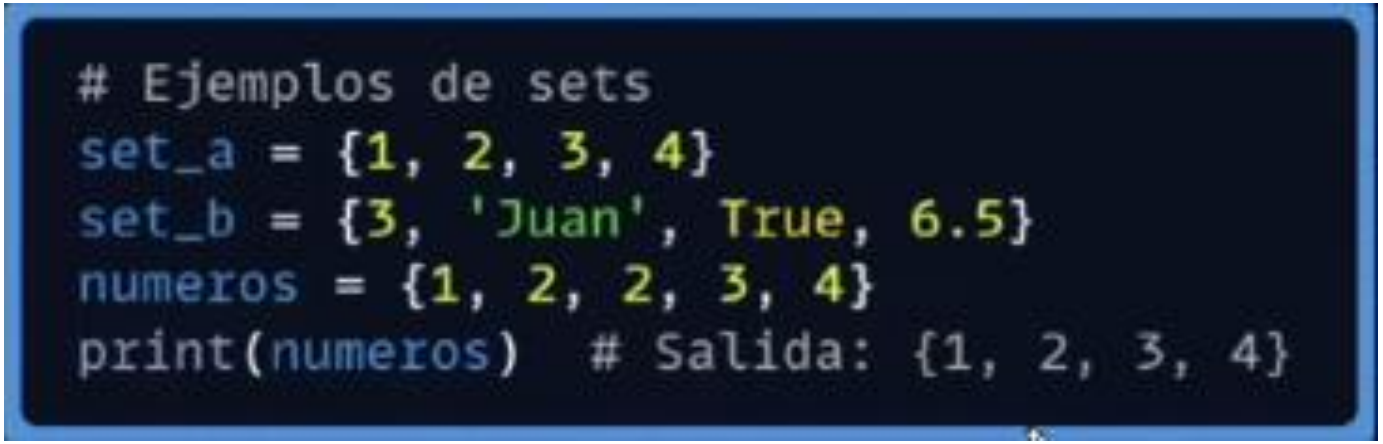
UNIVERSIDAD AMERICANA

Set - Conjutos

Un Set (conjunto) es una colección desordenada de elementos únicos e inmutables. Los sets son útiles cuando se necesita almacenar múltiples elementos, pero no deben ser duplicados y no importa el orden de los elementos.

Características de los sets

- **Elementos únicos:** Un set no permite elementos duplicados. Si añades un elemento que ya está presente, simplemente se ignorará.
- **Colección desordenada:** No mantiene el orden de inserción de los elementos.
- **Mutables:** Puedes añadir y eliminar elementos después de su creación.
- **Tipos inmutables:** Los elementos dentro de un set deben ser de un tipo inmutable (números, strings, tuplas, etc.)..



Operaciones con Sets

Crear un conjunto

```
>>> mi_set = {1,2,3,4}
```

Crear un conjunto a partir de una lista

```
>>> lista = ['a','b','c','d']
>>> set2 = set(lista)
>>> print(set2)
{'b', 'c', 'a', 'd'}
```

Crear un conjunto vacio

```
>>> mi_set = set()
```

```
>>> print(type(mi_set))
<class 'set'>
```

Operaciones con Sets - II

Añadir elementos

```
>>> numeros = {1, 2, 3}
>>> numeros.add(4)
>>> print(numeros)
{1, 2, 3, 4}
```

Eliminar elementos
remove(): Lanza un error si el elemento no existe.

```
>>> numeros.remove(3)
>>> print(numeros)
{1, 2, 4}
>>>
```

discard(): No lanza un error si el elemento no existe

```
>>> numeros.discard(5)   # No pasa nada
...
```

#crear un conjunto vacio

Operaciones con Sets - III

Verificar si un elemento está presente:

```
>>> print(2 in numeros)
True
>>> print(10 in numeros)
False
```

Recorrer un conjunto

```
>>> for elemento in mi_set:
...     print(elemento, end=' ')
...
1 3 4 5 6 -1
```

Operaciones entre Sets

Unión (union() o |): Combina elementos de ambos sets.

```
>>> A = {1, 2, 3}
>>> B = {3, 4, 5}
>>> union = A | B
>>> print(union)
{1, 2, 3, 4, 5}
```

O también

```
>>> A = {1, 2, 3}
>>> B = {3, 4, 5}
>>> A.union(B)
{1, 2, 3, 4, 5}
```

Operaciones entre Sets

Intersección (intersection() o &): Elementos comunes a ambos sets.

```
>>> A = {1, 2, 3, 4}
>>> B = {3, 4, 5, 6}
>>> interseccion = A & B
>>> print(interseccion)
{3, 4}
```

Diferencia (difference() o -): Elementos en A que no están en B.

```
>>> print(diferencia)
{1, 2}
```

Ejemplo Lista de Suscriptores

Crear un programa para administrar una lista de suscriptores utilizando su email. Suponer que una persona se suscribe al boletín informativo utilizando su email. A medida que la lista crece, hay que asegurarse que no tengamos suscriptores duplicados. También debemos poder agregar y eliminar suscriptores.

```
print('*** Lista de Suscriptores ***')

suscriptores = {'luisa@mail.com', 'marcos@mail.com', 'elena@mail.com'}
print(f'Lista de suscriptores inicial: {suscriptores}')

# Verifica si un nuevo suscriptor ya está en la lista
nuevo_suscriptor = 'karla@mail.com'
if nuevo_suscriptor in suscriptores:
    print(f'El nuevo suscriptor ya está en la lista {nuevo_suscriptor}')
else:
    suscriptores.add(nuevo_suscriptor)
    print(f'El nuevo suscriptor se ha agregado a la lista {nuevo_suscript
print(f'Lista de suscriptores : {suscriptores}')
```

UNIVERSIDAD AMERICANA

Ejemplo Lista de Suscriptores - II

Crear un programa para administrar una lista de suscriptores utilizando su email. Suponer que una persona se suscribe al boletín informativo utilizando su email. A medida que la lista crece, hay que asegurarse que no tengamos suscriptores duplicados. También debemos poder agregar y eliminar suscriptores.

```
# Eliminamos un suscriptor
suscriptor_eliminar = 'elena@mail.com'
suscriptores.remove(suscriptor_eliminar)
print(f'El suscriptor {suscriptor_eliminar} ha sido eliminado de la lista')
print(f'Lista de suscriptores: {suscriptores}')

# Verificamos la cantidad total de suscriptores
print(f'Cantidad total suscriptores: {len(suscriptores)}')

# Mostramos todos los suscriptores
print(f'--- Lista de Suscriptores ---')
for suscriptor in suscriptores:
    print(f'- {suscriptor}')
```

Diccionarios en Python

Un diccionario es una colección ordenada de pares clave-valor. Los diccionarios son muy útiles cuando necesitas almacenar datos en forma de pares, donde cada clave (key) se asocia a un valor (value). A diferencia de las listas y los sets, los elementos de un diccionario se acceden mediante sus claves y no por su posición.

```
# Sintaxis de un diccionario
mi_diccionario = {clave1: valor1, clave2: valor2}
```

```
# Ejemplo de un diccionario
persona = {'nombre': 'Pedro', 'edad': 30, 'es_casado': True}
```

Ejemplo de Diccionarios en Python

#Modificar un valor del diccionario

```
persona['edad'] = 35
print(f'Diccionario de persona: {persona}')
```

#Agregar un nuevo elemento

```
persona['profesion'] = 'Ingeniero'
print(f'Diccionario de persona: {persona}')
```

#Eliminar un elemento

```
# Eliminar un elemento
del persona['ciudad']
print(f'Diccionario de persona: {persona}')
```

Ejemplo de Diccionarios en Python

#Iterar los elementos de un dict(llave, valor)

```
# Iterar los elementos de un dict (llave, valor)
for llave, valor in persona.items():
    print(f'Llave: {llave}, Valor: {valor}')
```

#Obtener los valores

```
print(f'\nValores del diccionario: ')
for valor in persona.values():
    print(f'- Valor: {valor}')
```

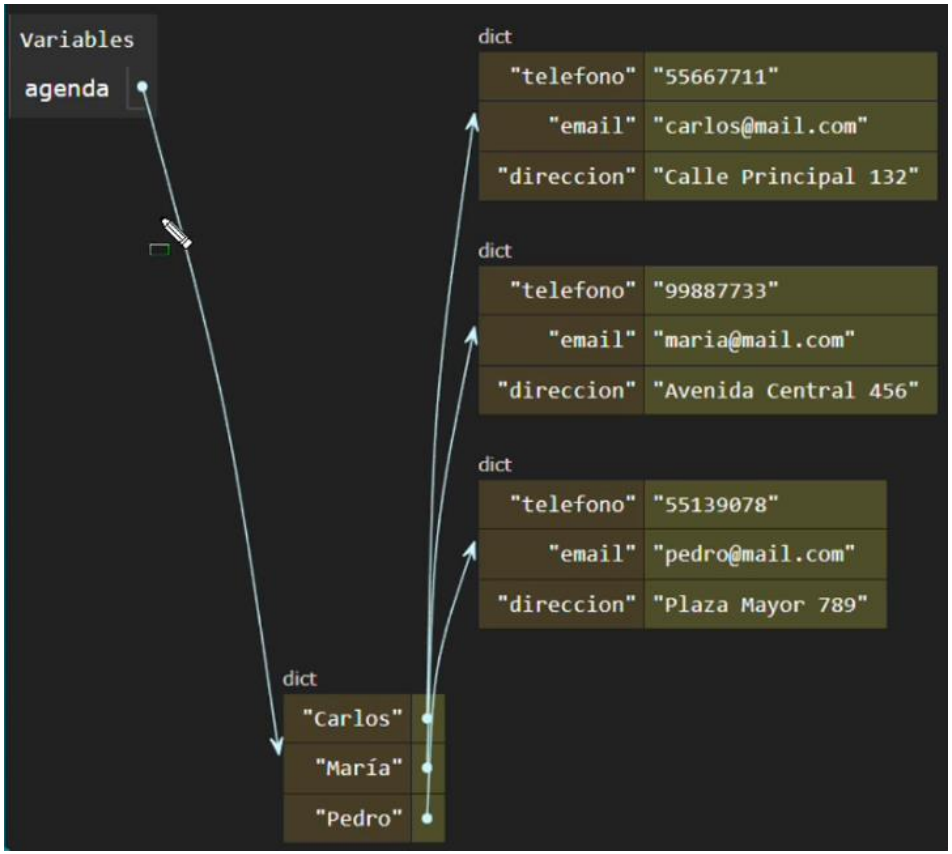
#Obtener las llaves

```
print(f'Impresión de las llaves del diccionario:')
for llave in persona.keys():
    print(f'- {llave}')
```

Ejemplo – Agenda de Contactos

Crear una agenda de contactos utilizando un diccionario de Python con la siguiente estructura

```
agenda{
        nombre {
                telefono
                email
                dirección
        }
}
```



Agenda de Contactos – Creación de la estructura

```
print('*** Agenda de Contactos ***')

agenda = {
    'Carlos': {
        'telefono': '55667711',
        'email': 'carlos@mail.com',
        'direccion': 'Calle Principal 132'
    },
    'María': {
        'telefono': '99887711',
        'email': 'maria@mail.com',
        'direccion': 'Avenida Central 456'
    }

}
```

Agenda de Contactos – Información de un Contacto

#Acceder a la información de un contacto específico

```
print(f'''Información del contacto de María:
    Teléfono: {agenda['María']['telefono']}
    Email: {agenda.get('María').get('email')}
    Dirección: {agenda.get('María').get('direccion')}
''')
```

Agenda de Contactos – Agregar o Eliminar Contacto

#Agregar un nuevo contacto

```
agenda['Ana'] = {
    'telefono': '55678392',
    'email': 'ana@mail.com',
    'direccion':'Calle Salvador Diaz 321'
}
```

#Eliminar un contacto existente

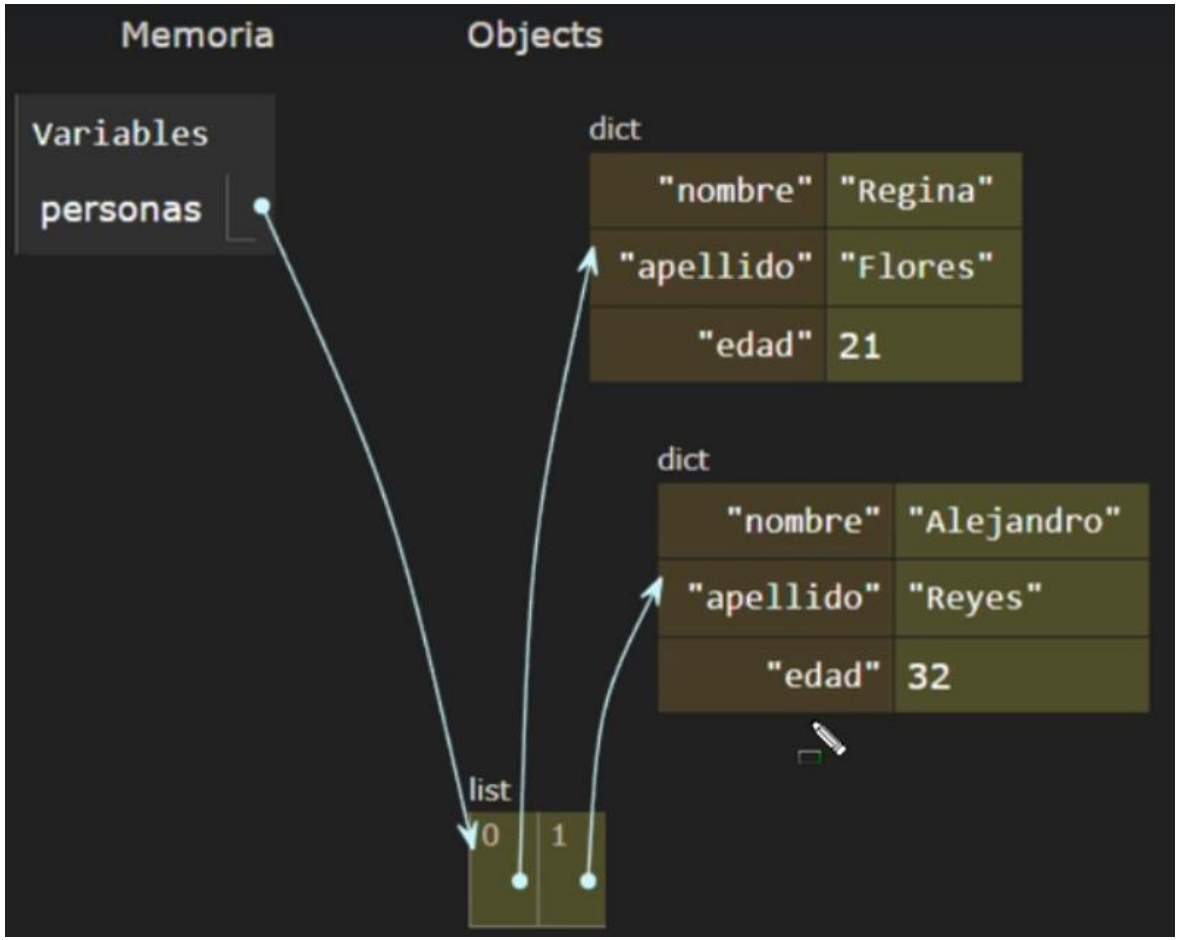
```
agenda.pop('Pedro')
#del agenda['Pedro']
print(agenda)
```

Agenda de Contactos – Mostrar Contactos

#Visualizar todos los contactos

```
print('\nContactos en la Agenda')
for nombre, detalles in agenda.items():
    print(f'''Nombre: {nombre}
    Teléfono: {detalles.get('telefono')}
    Email: {detalles.get('email')}
    Dirección: {detalles.get('direccion')}
''')
```

Combinar Listas con Diccionarios



Combinar Listas con Diccionarios

#Primero se crea la lista que va contener los diccionarios

```
print('*** Listas y Diccionarios ***')

personas = [
    {
        'nombre': 'Regina',
        'apellido': 'Flores',
        'edad': 21
    },
    {
        'nombre': 'Alejandro',
        'apellido': 'Reyes',
        'edad': 32
    }
]
```

Combinar Listas con Diccionarios

#Acceder a un diccionario desde una lista

```
print(f'''Nombre: {personas[0].get('nombre')}
''')
```

#Recorrer los elementos de la lista

```
print()
for contador, persona in enumerate(personas):
    print(f'{contador} - Persona: {persona}')
```

Compresión de Listas

La comprensión de listas en una forma concisa y eficiente de crear listas a partir de otros iterables(listas, tuplas, set o diccionarios)

Permite filtrar elementos y aplicar expresiones a cada elemento de un iterable de manera muy legible y en una sola línea de código.

```
# Sintaxis comprension de listas
[nueva_expresion for elemento in iterable if condicion]
```

nueva_expresion: Es la Expresión que define cómo se modifica o procesa cada elemento del iterable

elemento: Variable que representa cada elemento del iterable original

iterable: La secuencia o colección sobre la cual se itera

condicion: (Opcional) Es una condición para filtrar los elementos de iterable

Ejemplo de compresión de Listas

#Determinar los números pares en la lista

```
# Ejemplo comprension de listas
numeros = [1, 2, 3, 4, 5, 6]
pares = [x for x in numeros if x % 2 == 0]
print(pares)  # Salida: [2, 4, 6]
```

#Hallar el cuadrado de los números en la lista

```
# Ejemplo comprension de listas
numeros = [1, 2, 3, 4, 5]
cuadrados = [x**2 for x in numeros]
print(cuadrados)  # Salida: [1, 4, 9, 16, 25]
```