OPTATIVO 1 – PYTHON I

Prof. Ing. Aaron Zárate

UNIVERSIDAD AMERICANA

Colecciones en Python

Una colección es un conjunto de datos. En Python tenemos varios tipos que podemos utilizar con el objetivo de almacenar, organizar y manipular múltiples conjuntos de datos, por ello también se les conoce como estructura de datos. Los tipos que vamos a estudiar son:

- Listas
- Tuplas
- Set (conjunto)
- Diccionarios

Estos son los tipos mas comunes y mas utilizados.

Listas en Python

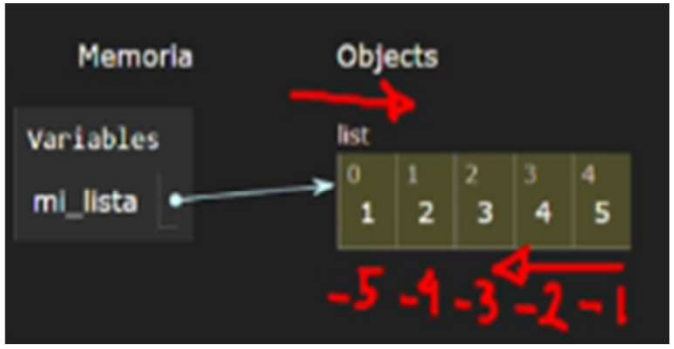
Las listas en Python son colecciones ordenadas y mutables de elementos que pueden cambiar de tamaño, podemos añadir, modificar o eliminar elementos

```
# Sintaxis listas
mi_lista = [elemento1, elemento2, elemento3]
```

```
# Ejemplos de listas
numeros = [1, 2, 3, 4, 5]
frutas = ["manzana", "banana", "cereza"]
mixta = [1, "dos", 3.0, [4, 5]]
```

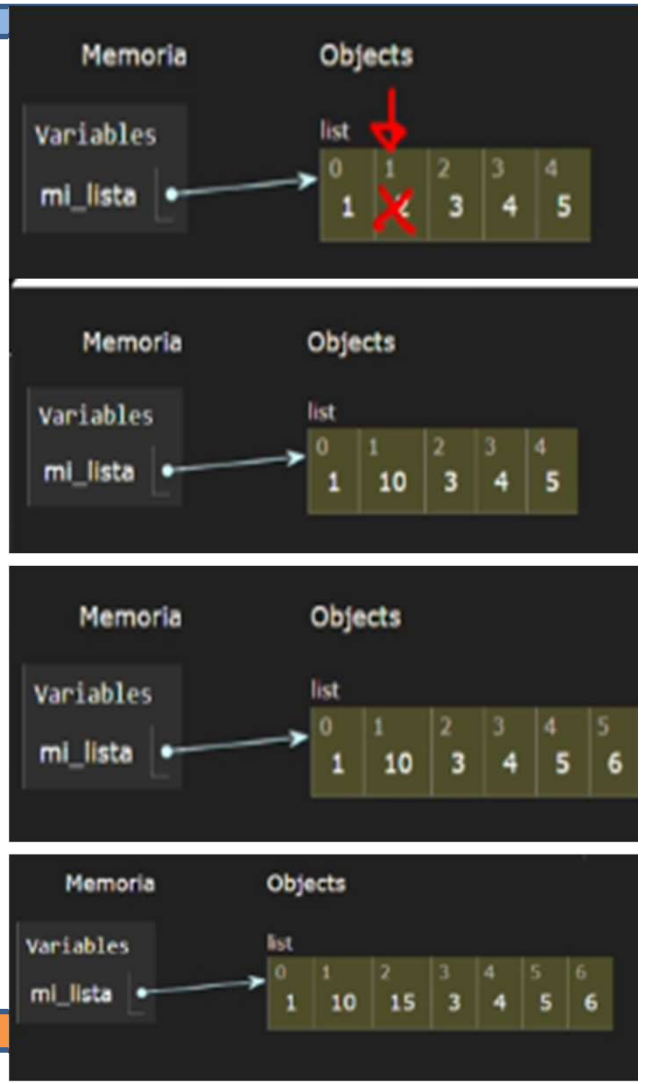
Ejemplo de listas en Python

```python
print("*** Manejo de Listas ***")
mi_lista = [1 , 2 , 3 , 4 , 5]
print(f"{mi_lista} --> Lista original")

#Largo de una lista
print(f"Largo de la lista: {len(mi_lista)}")

#Acceder a los elementos de la lista por indice
print(f"Accedemos al valor del indice 4: {mi_lista[4]}")
print(f"Accedemos al ultimo indice: {mi_lista[-1]}")
```



Agregar elementos a una lista

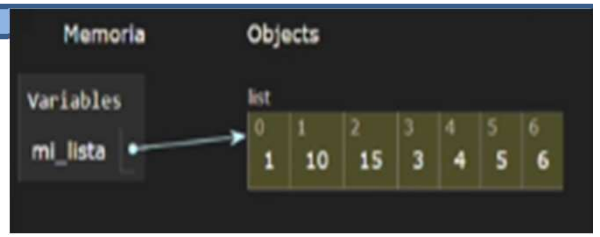
```python
#Modificar los Elementos de una lista
mi_lista[1] = 10
print(f"Modificamos el valor del indice 1:{mi_lista[1]}")

#Agregar un nuevo elemento al final de la lista
mi_lista.append(6)
print(f"{mi_lista} --> Se agrego el elemento 6")

#Añadir un nuevo elementoe n un indice especifico
mi_lista.insert(2,15)
print(f"{mi_lista} -> Se añadio el valor de 15 en indice 2")
```

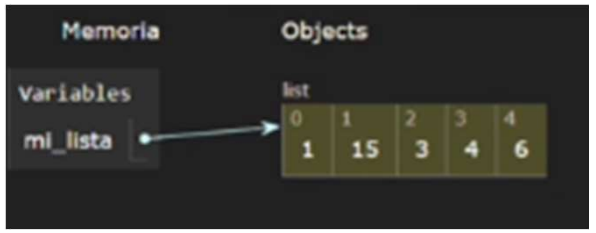


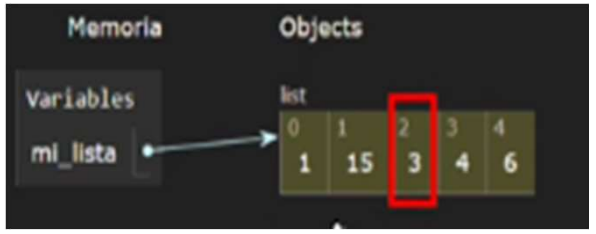
UNIVERSIDAD AMERICANA

Eliminar elementos de una lista

```python
#Eliminar elementos en una lista
#usando el metodo remove
mi_lista.remove(5)
print(f"{mi_lista} Se removio el valor 5 ")

#Removemos por indice con el metodo pop
mi_lista.pop(1) # remueve elemento del indice 1
print(f"{mi_lista} -> Se elimino el indice 1")

#Eliminar usando la palabra del
del mi_lista[2]
print(f"{mi_lista} -> Se elimino el indice 2")

#Obtener sublistas
#Genera un sublista  del indice 1 al 2 (3 no se incluye)
sublista = mi_lista[1:3]
print(f"Sublista [1:3]-> {sublista}")
```

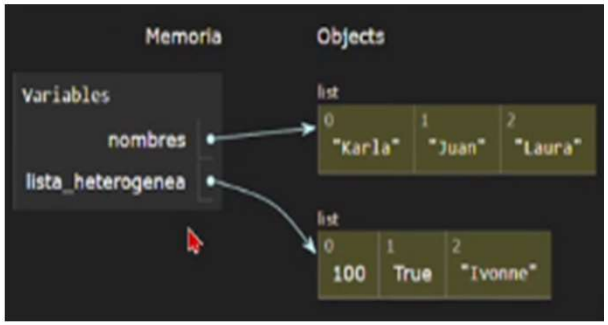






Iterar elementos de una lista

```
print(f"*** Iterar listas  ***")

nombres = ["Karla", "Juan", "Laura"]
for nombre in nombres:
    print(nombre)

lista_heterogenea = [100, True, "Ivonne"]
print()
for elemento in lista_heterogenea:
    print(elemento)
```



Ejemplo Playlist

Crea un programa para administrar una lista de canciones. Debes solicitar al usuario cuantas canciones desea agregar a la lista y posteriormente ir solicitando cada canción que desea agregar a la lista.

Finalmente debe desplegar la lista de canciones en orden alfabético.

Ejemplo 1 Playlist

```
print(f"***  Playlist de canciones ***")
#Creamos una lista vacia
lista_reproduccion = []
#Empezamos a agregar canciones
lista_reproduccion.append("Hotel California - Eagleas")
lista_reproduccion.append("Staying Alive - Bee Gees")
lista_reproduccion.append("Dream on - Aerosmith")
#Ordenar la lsita en orden alfabetico sort
lista_reproduccion.sort()# reverse=True
#Mostrar la lista de canciones
print(f"\nLista de reproduccion en orden alfabetico")
print(lista_reproduccion)
print(f"\nIteramos el Playlist")
#Mostrar la lista iterando sus elementos
for cancion in lista_reproduccion:
    print(f"- {cancion }")
```

Ejemplo 2 Playlist

```python
print('*** Playlist de Canciones ***')

# Creamos la lista vacia
lista_reproduccion = []

numero_canciones = int(input('Cuantas canciones deseas agregar? '))

# iteramos cada elemento de la lista para agregar un nuevo elemento
for indice in range(numero_canciones):
    cancion = input(f'Proporciona la cancion {indice + 1}: ')
    lista_reproduccion.append(cancion)

# Ordenar la lista en orden alfabetico. sort
#lista_reproduccion.sort(reverse=True)
lista_reproduccion.sort()

# Mostar la lista lista iterando sus elementos
print('\nIteramos el playlist')
for cancion in lista_reproduccion:
    print(f'- {cancion}')
```

Promedio de calificaciones

Crea un programa para realizar el calculo promedio de calificaciones

El programa debe solicitar el numero de calificaciones a utilizar para obtener el promedio. Posteriormente, se debe solicitar cada calificación al usuario.

Posteriormente realizar la suma de todas las calificaciones y finalmente mandar a imprimir el promedio.

Tuplas en Python

Las tuplas son similares a las listas, ya que también son una colección de datos ordenados, pero las tuplas son inmutables, lo que significa que una vez creada una tupla, no es posible modificar su tamaño, ni podemos agregar mas elementos, ni modificarlos, ni eliminarlos.

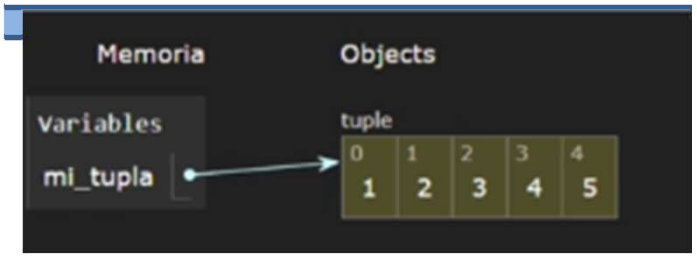
Las tuplas suelen utilizarse para crear colecciones de datos que no deben cambiar con el tiempo

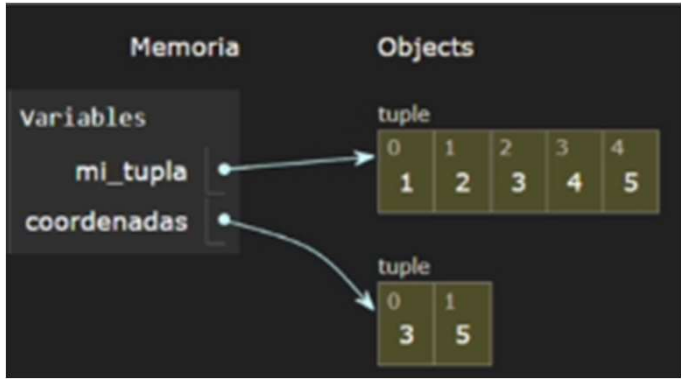
```
# Sintaxis de una tupla
mi_tupla = (elemento1, elemento2, elemento3)
mi_tupla_sin_parentesis = elemento1, elemento2, elemento3
```

```
# Ejemplos de tuplas
tupla_numeros = (1, 2, 3, 4, 5)
tupla_mixta = ("manzana", 10, 3.14, [1, 2, 3])
tupla_sin_parentesis = 'Juan', 'Karla'
tupla_un_elemento = 10,  # La coma no es opcional
```

Ejemplo de Tuplas en Python

```python
print('*** Manejo de Tuplas ***')
mi_tupla = (1, 2, 3, 4, 5)
print(mi_tupla)
# No podemos modificar una tupla
#mi_tupla[0] = 10
#mi_tupla.append(6)
# Iteramos los elementos de una tupla
for elemento in mi_tupla:
    print(elemento, end=' ')
# Crear una tupla para una coordenada x,y
coordenadas = (3, 5)
# Accedemos a cada elemento de la tupla
print(f'\nCoordenada en el eje x: {coordenadas[0]}')
print(f'Coordenada en el eje y: {coordenadas[1]}')
# Crear una tupla unitaria
tupla_un_elemento = 10,
print(f'Tupla de un elemento: {tupla_un_elemento}')
# Tupla anidada
tuplas_anidada = (1, (2,3), (4,5))
print(f'Segundo elemento tupla anidada: {tuplas_anidada[1]}')
```

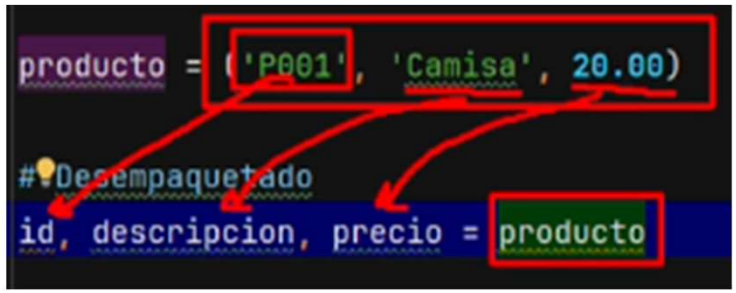




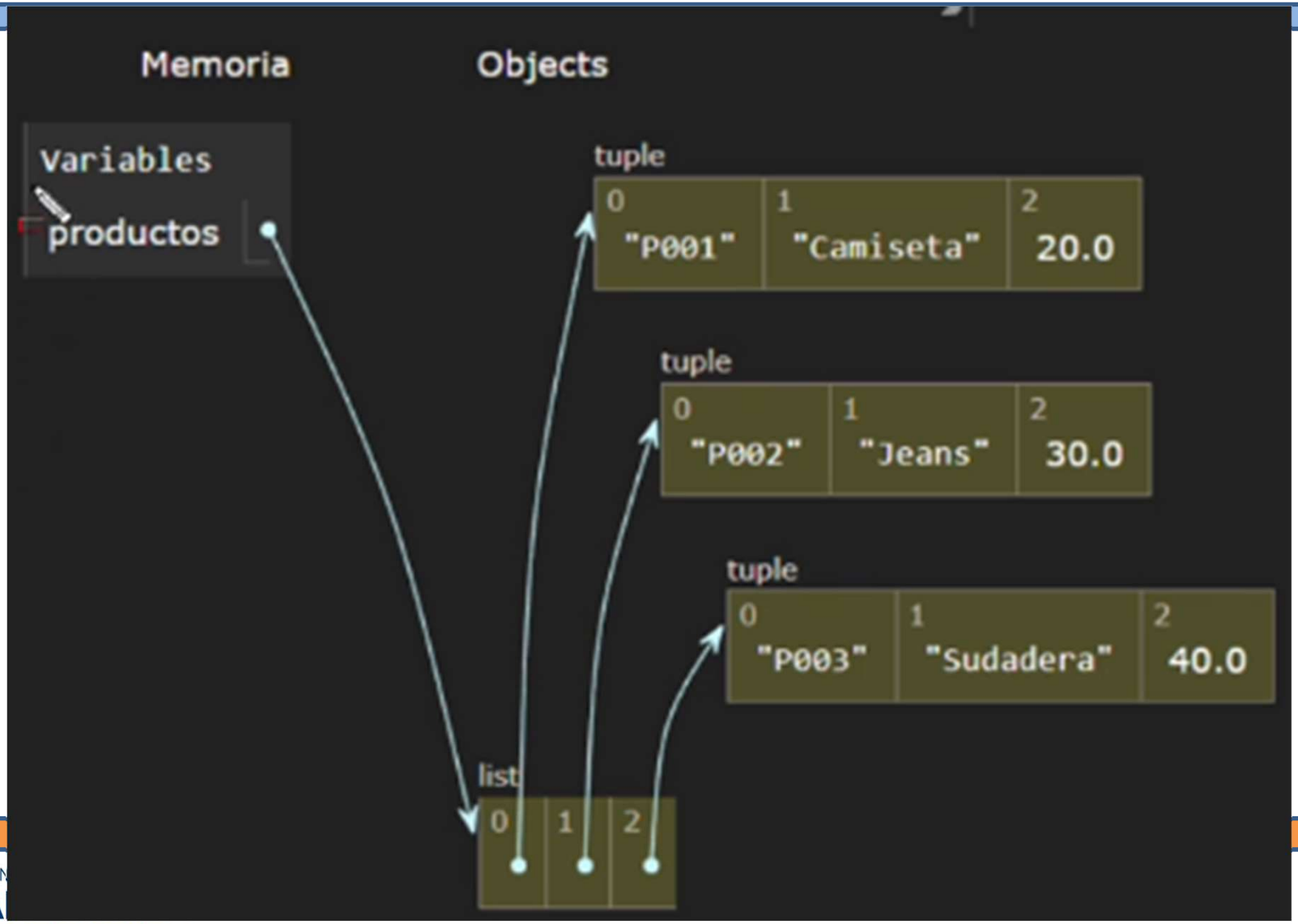
UNIVERSIDAD AMERICANA

Desempaquetado de Tuplas

```python
print('*** Desempaquetado de Tuplas ***')  # unpacking

producto = ('P001', 'Camisa', 20.00)

# Desempaquetado
id, descripcion, precio = producto

#Imprimir los valores
print(f'Tupla completa: {producto}')
# Valores independientes ya desempaquetados
print(f'Producto: id = {id}, descripcion = {descripcion}, precio = {precio}')
```



Combinación de listas y tuplas



Combinación de listas y tuplas

```python
print('*** Combinación de Listas y Tuplas ***')

# definir una lista que almacena tuplas de productos
productos = [
    ('P001', 'Camiseta', 20.00),
    ('P002', 'Jeans', 30.00),
    ('P003', 'Sudadera', 40.00)
]
# Imprimir la informacion de cada producto
# y ademas calculamos el precio total
precio_total = 0

print('Información de los productos: ')
for producto in productos:
    #print(producto)
    id, descripcion, precio = producto  # unpacking
    print(f'Producto: id = {id}, descripcion = {descripcion}, precio = ${precio}')
    precio_total += precio # producto[2]
print(f'Precio total de los productos: ${precio_total}')
```

AMERICANA

Sets en Python

Los set (conjuntos) son colecciones de datos no ordenados de elementos únicos. Son muy útiles cuando debemos asegurarnos de que no haya elementos duplicados en nuestra colección.

```
# Sintaxis de un set
mi_set = {elemento1, elemento2, elemento3}
```

```
# Ejemplos de sets
set_a = {1, 2, 3, 4}
set_b = {3, 'Juan', True, 6.5}
numeros = {1, 2, 2, 3, 4}
print(numeros)  # Salida: {1, 2, 3, 4}
```

Ejemplos de Sets en Python

```python
print('*** Manejo de Sets ***')
# Crear un conjunto
mi_set = {1, 2, 3, 4, 5, 4}
print(f'Mi set: {mi_set}')
# Agregar elementos al set
mi_set.add(6)
mi_set.add(7)
# Intentamos agregar un elemento duplicado
mi_set.add(3)
# Eliminar un elemento del conjunto
mi_set.remove(4)
print(f'Mi set modificado: {mi_set}')
# Iterar los elementos del set
for elemento in mi_set:
    print(elemento, end=' ')
# Comprobar si existe un elemento en el set
print(f'\nExiste el valor de 1 en el set? {1 in mi_set}')
# Obtener la longitud del set
print(f'Longitud del conjunto: {len(mi_set)}')
```