Tipos de datos complejos

- **Tuplas**
- **Listas**
- **Diccionarios**

Estos tres tipos, pueden almacenar colecciones de datos de diversos tipos y se diferencian por su sintaxis y por la forma en la cual los datos pueden ser manipulados.

Tuplas

- Una tupla es una variable que permite almacenar varios datos inmutables (no pueden ser modificados una vez creados) de tipos diferentes:

- **mi_tupla = ('cadena de texto', 15, 2.8, 'otro dato', 25)**

Se puede acceder a cada uno de los datos mediante su índice correspondiente, siendo 0 (cero), el índice del primer elemento:

print mi_tupla[1]

 # Salida: 15

Otra forma de acceder a la tupla de forma inversa (de atrás hacia adelante), es colocando un índice negativo:

```
print mi_tupla[-1]
# Salida: 25
print mi_tupla[-2]
# Salida: otro dato
```

También se puede acceder a una porción de la tupla, indicando (opcionalmente) desde el índice de inicio hasta el índice de fin:

print mi_tupla[1:4]

Devuelve: (15, 2.8, 'otro dato')

print mi_tupla[3:]

Devuelve: ('otro dato', 25)

print mi_tupla[:2]

Devuelve: ('cadena de texto', 15)

Importante

- No se puede modificar una tupla
- No se puede añadir nuevos elementos
- No se pueden borrar elementos

- Debes saber algo más a la hora de crear tuplas. Para crear una tupla vacía, sin elementos, abrimos y cerramos paréntesis sin más:

  mi_tupla=()

- Si no te crees que esto es una tupla, prueba con la función type() para saber de qué objeto estamos hablando:

  mi_tupla=()

  print type(mi_tupla)

- Para crear una tupla con un único elemento, debemos finalizar la enumeración con una coma:

```
mi_tupla=(1,)
print mi_tupla[0]
```

- El método count() nos devuelve cuántos elementos existen con una valor determinado.

mi_tupla=(1,)

print mi_tupla.count(1)

- El método index() nos devuelve el índice de la primera ocurrencia que haya del valor buscado.

mi_tupla=(1,)

print mi_tupla.index(1)

Listas

Una lista es similar a una tupla con la diferencia fundamental de que permite modificar los datos una vez creados

mi_lista = ['cadena de texto', 15, 2.8, 'otro dato', 25]

A las listas se accede igual que a las tuplas, por su número de índice:

print mi_lista[1]

Salida: 15

print mi_lista[1:4]

Devuelve: [15, 2.8, 'otro dato']

print mi_lista[-2]

Salida: otro dato

- Las lista NO son inmutables: permiten modificar los datos una vez creados:

  mi_lista[2] = 3.8

  # el tercer elemento ahora es 3.8

- Las listas, a diferencia de las tuplas, permiten agregar nuevos valores:

  mi_lista.append('Nuevo Dato')

- Con insert podemos añadir nuevos elementos en la posición deseada

```
mi_lista=[0,1,2,3]
mi_lista.insert(2, 0)
print mi_lista[2]
```

- Con insert podemos añadir nuevos elementos en la posición deseada

mi_lista=[0,1,2,3]

mi_lista.remove(2)

print mi_lista[0:3]

- Para buscar si existe un elemento en una lista, usamos **in**

mi_lista=[0,1,2,3]

print 4 in mi_lista

#false

print 1 in mi_lista

#true

- Para buscar un elemento en la lista

  mi_lista=[0,1,2,3]

  print mi_lista.index(0)

- Si no existe el valor, devuelve error

- Si esta duplicado, devuelve el primero

  mi_lista=[0,1,2,0]

  print mi_lista.index(0)

- Para iterar sobre todos los elementos de la lista

```
mi_lista = [0,1,2,3,4,5]
for x in mi_lista:
    print x
```

- Numero de elementos

  mi_lista = [0,1,2,3,4,5]

  print len(mi_lista)

- Menor de los elementos

  mi_lista = [0,1,2,3,4,5]

  print min(mi_lista)

- Mayor de los elementos

  mi_lista = [0,1,2,3,4,5]

  print max(mi_lista)

Concatenación simple

```
lista1 = [1, 2, 3, 4]
lista2 = [3, 4, 5, 6, 7, 8]
lista3 = lista1 + lista2
print lista3


tupla1 = (1, 2, 3, 4, 5)
tupla2 = (4, 6, 8, 10)
tupla3 = (3, 5, 7, 9)
tupla4 = tupla1 + tupla2 + tupla3
print tupla4
```

Diccionarios

- Mientras que a las listas y tuplas se accede solo y únicamente por un número de índice, los diccionarios permiten utilizar una clave para declarar y acceder a un valor:

  mi_diccionario = {'clave_1': valor_1, 'clave_2': valor_2, 'clave_7': valor_7}

  print mi_diccionario['clave_2'] # Salida: valor_2

- Un diccionario permite eliminar cualquier entrada:

  del(mi_diccionario['clave_2'])

- Al igual que las listas, el diccionario permite modificar los valores

  mi_diccionario['clave_1'] = 'Nuevo Valor'

Recorrer un diccionario

```
materias = {}
materias["lunes"] = [1, 2]
materias["martes"] = [3]
materias["miercoles"] = [4, 5]
materias["jueves"] = []
materias["viernes"] = [6]

for dia in materias:
    print dia, ":", materias[dia]
```

Pero, y si pasamos un valor que no existe?

print materias.get("domingo")

- Nos devuelve

  None

Es posible, también, obtener los valores como tuplas donde el primer elemento es la clave y el segundo el valor.

```
for dia, codigos in materias.items():
    print dia, ":", codigos
```

Comprobar si existe un valor

```
dias = {'lunes': 1, 'martes': 2}

if dias.has_key('lunes'):
    print dias['lunes']
# Imprime 12

if dias.has_key('jueves'):
    print dias['jueves']
# No se ejecuta
```

```
if 'martes' in dias:
    print dias['martes']

# Imprime 7
```

Obtener las claves y valores de un diccionario

```
dias = {'lunes': 1, 'martes': 2, 'miercoles': 3}

for clave, valor in dias.iteritems():

    print "El valor de la clave %s es %s" % (clave, valor)
```