La **lista en Python** son variables que almacenan arrays, internamente cada posición puede ser un tipo de datos distinto.

```
>>> factura = ['pan', 'huevos', 100, 1234]
>>> factura
['pan', 'huevos', 100, 1234]
```

Las listas en Python son:

- heterogéneas: pueden estar conformadas por elementos de distintos tipo, incluidos otras listas.
- mutables: sus elementos pueden modificarse.

Una lista en Python es una estructura de datos formada por una secuencia ordenada de objetos.

Los elementos de una lista pueden accederse mediante su índice, siendo 0 el índice del primer elemento.

```
>>> factura[0]
'pan'
>>> factura[3]
1234
```

La función len() devuelve la longitud de la lista (su cantidad de elementos).

```
>>> len(factura)
4
```

Los índices de una lista inicia entonces de 0 hasta el tamaño de la lista menos uno (len(factura) - 1):

```
>>> len(factura) - 1
3
```

append()
Este método agrega un elemento al final de una lista.

```
>>> versiones_plone = [2.5, 3.6, 4, 5]
>>> print versiones_plone
[2.5, 3.6, 4, 5]
>>> versiones_plone.append(6)
>>> print versiones_plone
[2.5, 3.6, 4, 5, 6]
```

count()
Este método recibe un elemento como argumento, y cuenta la cantidad de veces que aparece en la lista.

```
>>> versiones_plone = [2.1, 2.5, 3.6, 4, 5, 6]
>>> print "6 ->", versiones_plone.count(6)
6 -> 1
>>> print "5 ->", versiones_plone.count(5)
5 -> 1
>>> print "2.5 ->", versiones_plone.count(2.5)
2.5 -> 1
```

extend()
Este método extiende una lista agregando un iterable al final.

```
>>> versiones_plone = [2.1, 2.5, 3.6]
>>> print versiones_plone
[2.1, 2.5, 3.6]
>>> versiones_plone.extend([4])
>>> print versiones_plone
[2.1, 2.5, 3.6, 4]
>>> versiones_plone.extend(range(5,7))
>>> print versiones_plone
[2.1, 2.5, 3.6, 4, 5, 6]
```

index()
Este método recibe un elemento como argumento, y devuelve el índice de su primera aparición en la lista.

```
>>> versiones_plone = [2.1, 2.5, 3.6, 4, 5, 6, 4]
>>> print versiones_plone.index(4)
3
```

insert()
Este método inserta el elemento x en la lista, en el índice i.

```
>>> versiones_plone = [2.1, 2.5, 3.6, 4, 5, 6]
>>> print versiones_plone
[2.1, 2.5, 3.6, 4, 5, 6]
>>> versiones_plone.insert(2, 3.7)
>>> print versiones_plone
[2.1, 2.5, 3.7, 3.6, 4, 5, 6]
```

Tipo tuplas

Las tuplas son objetos de tipo secuencia, específicamente es un tipo de dato lista inmutable. Esta no puede modificarse de ningún modo después de su creación.

count()
Este método recibe un elemento como argumento, y cuenta la cantidad de veces que aparece en la tupla.

```
>>> valores = ("Python", True, "Zope", 5)
>>> print "True ->", valores.count(True)
True -> 1
>>> print "'Zope' ->", valores.count('Zope')
'Zope' -> 1
>>> print "5 ->", valores.count(5)
5 -> 1
```

index()
Comparte el mismo método index() del tipo lista. Este método recibe un elemento como argumento, y devuelve el índice de su primera aparición en la tupla.

```
>>> valores = ("Python", True, "Zope", 5)
>>> print valores.index(True)
1
>>> print valores.index(5)
3
```

Tipo diccionarios

El diccionario, define una relación uno a uno entre claves y valores.

Clase	Tipo	Notas	Ejemplo
dict	Mapeos	Mutable, sin orden.	{'cms':"Plone", 'version':5}

```
diccionario = {
...     "clave1":234,
...     "clave2":True,
...     "clave3":"Valor 1",
...     "clave4":[1,2,3,4]
... }
>>> print diccionario, type(diccionario)
{'clave4': [1, 2, 3, 4], 'clave1': 234,
'clave3': 'Valor 1', 'clave2': True} <type 'dict'>
```

Usted puede acceder a los valores del diccionario usando cada su clave, se presenta unos ejemplos a continuación:

```
>>> diccionario['clave1']
234
>>> diccionario['clave2']
True
>>> diccionario['clave3']
'Valor 1'
>>> diccionario['clave4']
[1, 2, 3, 4]
```

Asignar valor a clave
Esta operación le permite asignar el valor especifico del diccionario mediante su clave.

```
>>> versiones = {'python': 2.7, 'zope': 2.13, 'plone': None}
>>> versiones['plone']
>>> versiones['plone'] = 5.1
>>> versiones
{'python': 2.7, 'zope': 2.13, 'plone': 5.1}
>>> versiones['plone']
5.1
```

clear()
Este método remueve todos los elementos desde el diccionario.

```
>>> versiones = dict(python=2.7, zope=2.13, plone=5.1)
>>> print versiones
{'zope': 2.13, 'python': 2.7, 'plone': 5.1}
>>> versiones.clear()
>>> print versiones
{}
```

copy()
Este método devuelve una copia superficial del tipo diccionario:

```
>>> versiones = dict(python=2.7, zope=2.13, plone=5.1)
>>> otro_versiones = versiones.copy()
>>> versiones == otro_versiones
True
```

get()
Este método devuelve el valor en base a una coincidencia de búsqueda en un diccionario mediante una clave, de lo contrario devuelve el objeto None.

```
>>> versiones = dict(python=2.7, zope=2.13, plone=5.1)
>>> versiones.get('plone')
5.1
>>> versiones.get('php')
>>>
```

Ejercicio 1
Escriba una programa que tome una lista de números y devuelva la suma acumulada, es decir, una nueva lista donde el primer elemento es el mismo, el segundo elemento es la suma del primero con el segundo, el tercer elemento es la suma del resultado anterior con el siguiente elemento y así sucesivamente. Por ejemplo, la suma acumulada de [1,2,3] es [1, 3, 6].

Ejercicio 2
Escribe una programa llamada "elimina" que tome una lista y elimine el primer y último elemento de la lista y cree una nueva lista con los elementos que no fueron eliminados.
Luego escribe una función que se llame "media" que tome una lista y devuelva una nueva lista que contenga todos los elementos de la lista anterior menos el primero y el último.

Ejercicio 3
Escribe una programa "ordenada" que tome una lista como parámetro y devuelva True si la lista está ordenada en orden ascendente y devuelva False en caso contrario.
Por ejemplo, ordenada([1, 2, 3]) retorna True y ordenada([b, a]) retorna False.

Crea una tupla con los meses del año, pide números al usuario, si el numero esta entre 1 y la longitud máxima de la tupla, muestra el contenido de esa posición sino muestra un mensaje de error.

Crea una tupla con números e indica el numero con mayor valor y el que menor tenga.

Ejercicios de diccionarios

Ejercicio 1
Escribe un programa python que pida un número por teclado y que cree un diccionario cuyas claves sean desde el número 1 hasta el número indicado, y los valores sean los cuadrados de las claves.

Ejercicio 2
Escribe un programa que lea una cadena y devuelva un diccionario con la cantidad de apariciones de cada carácter en la cadena.

Ejercicio 3

Vamos a crear un programa en python donde vamos a declarar un diccionario para guardar los precios de las distintas frutas. El programa pedirá el nombre de la fruta y la cantidad que se ha vendido y nos mostrará el precio final de la fruta a partir de los datos guardados en el diccionario. Si la fruta no existe nos dará un error. Tras cada consulta el programa nos preguntará si queremos hacer otra consulta.

Ejercicio 4
Codifica un programa en python que nos permita guardar los nombres de los alumnos de una clase y las notas que han obtenido. Cada alumno puede tener distinta cantidad de notas. Guarda la información en un diccionario cuya claves serán los nombres de los alumnos y los valores serán listas con las notas de cada alumno.

El programa pedirá el número de alumnos que vamos a introducir, pedirá su nombre e irá pidiendo sus notas hasta que introduzcamos un número negativo. Al final el programa nos mostrará la lista de alumnos y la nota media obtenida por cada uno de ellos. Nota: si se introduce el nombre de un alumno que ya existe el programa nos dará un error.