Tipos básicos de datos en Python

Tipo de datos list (listas o arreglos)

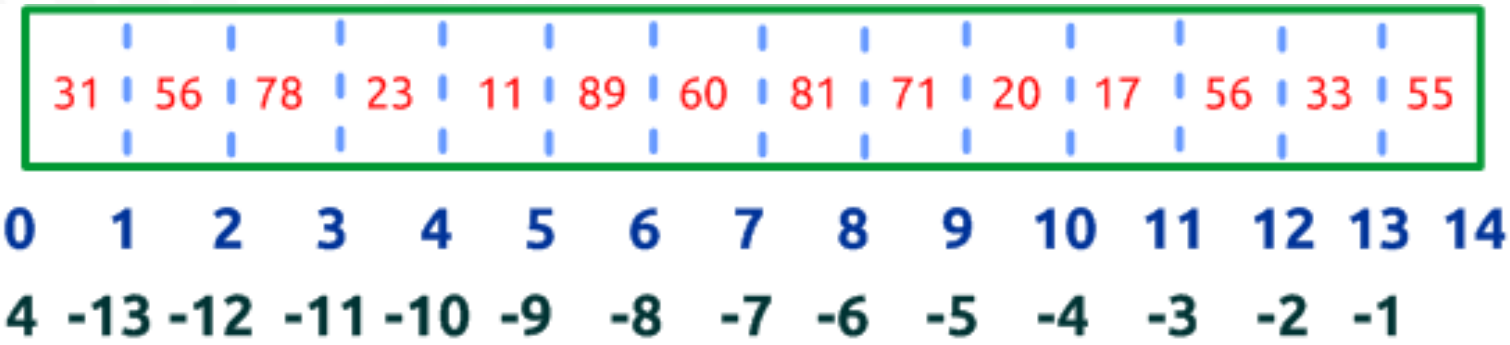
En Python una **`lista`** es una variable de arreglos donde donde los datos tendrán una posición determinada en el arreglo.

En una lista puedes asignar una serie de valores o ítems **`separándolos por una coma`**; la serie de valores deberá estar **`delimitada por corchetes`**. No se requiere que todos los ítems sean del mismo tipo

Tipos básicos de datos en Python

Tipo de datos list

- Los valores o ítems dentro de una lista se conocen como elementos.
- A cada elemento se le asocia un índice en la lista.
  - El primer elemento de una lista, siempre tendrá el índice 0.



Tipos básicos de datos en Python

Tipo de datos list

- Los elementos se pueden borrar de la lista o se pueden agregar nuevos elementos. Usando la notación punto objeto podemos ingresar nuevos elementos a la lista, ejemplo: `list.append(v)` Agrega el elemento “v” al final de la lista. `list.insert(i,v)` Agrega el elemento “v” en la posición “i” de la lista.
- Con la función `len(list)` conocemos el número de elementos que tiene el arreglo.

Tipos básicos de datos en Python

Tipo de datos list

Usando IPython, visualiza el resultado de los siguientes ejemplos:

```
a = [ 31, 56, 78, 23, 11, 89, 60, 81, 71, 20, 17, 56, 33, 55]
b = [ 452, 389, 'year', 'temperatura', 67] # Lista de strings e
    integers
c = [ 789, [1000, 2000], 0.34, 0.27] # Lista Anidada
d = []  # Lista vacia
a[0]    # Verificando el valor del 1er elemento de la lista a
a[4]    # Valor del 5to elemento de a
a[7]    # Valor del 8vo elemento de a
c[1][1] # Valor del 2do elemento de c y 2do. elemento de la lista
     anidada
a[-1]   # Valor del ultimo elemento de a
```

Tipos básicos de datos en Python

Tipo de datos list

Usando IPython, explica qué sucede en los siguientes ejemplos:

```
a[-13]  # Valor de 13vo elemento del ultimo elemento hacia la iz.
a[0:3]  # Visualizando del 1ero al 3er elemento.
a[:-5]  # Visualizando la lista a sin considerar los ultimos 5
    elementos
a[5:]   # Visualizando la lista a sin considerar los primeros 5
    elementos
a[:]    # Visualizando toda la lista
a[1:10:2] # Del 2do al 10mo elemento, visualizando con "saltos"
    de una posicion
a[-1:-4:-2] # Del ultimo al 4to elemento a la izquierda con "
    saltos" de una posicion
a[-1:-len(a):-1]    # Observa que la sintaxis es lista[
    indice_inicio:indice_final:salto]
```

Tipos básicos de datos en Python

Tipo de datos list

```
a[-1:-len(a):-1]    # Observa que la sintaxis es lista[
    indice_inicio:indice_final:salto]
a[-1:-len(a)-1:-1]  # el valor del indice_final no se visualiza.
    Estos 2 ejemplos invierten la lista
a.append(350)  # Agregando un elemento al final de la lista
a.insert(0,500) # Insertando en un valor al inicio de la lista
len(a) # Conociendo el numero de elementos de la lista
aa = bb = a[0:5] # Copiando de manera simultanea en dos listas,
    del 1er al 4to elemento de a
aw = a[:]  # Visualizando toda la lista aw
del aw[1]  # Borrando el segundo elemento de la lista aw
del aw[0:3] # Borrrando desde el 1er has ta
del bb    # Borrando el contenido de la lista bb
aa.remove(78) # remove elimina el VALOR de un elemento
aa.pop(1)  # pop elimina un elemento por su INDICE, en este caso
    se elimina el 2do elemento
```

Tipos básicos de datos en Python

Tipo de datos list

```
aa.pop()  # pop() elimina el ultimo de la lista
a[2:3] = [] # Elimina el 3er elemento
s = ['AB','CD','EF']
w = ['GH','IJ','KL']
s.extend(w) # Extendiendo la lista s con la lista w
s.index('EF') # Encontrando el INDICE de 'EF'
s.append('AB')  # Agregando 'AB' al final de la lista s
s.count('AB') # Conociendo la frecuencia que tiene 'AB' en s
s.sort()  # Ordenando la lista
sorted(s, reverse=True) # Ordenando la lista pero en sentido
    inverso
s.reverse() # Invirtiendo el contenido de la lista
del s[:] # Eliminando el contenido de la lista s
```

Tipos básicos de datos en Python

Tipo de datos tuplas

Las **tuplas** son similares a las listas, pero la principal diferencia entre ellas radica en que **los ítems no pueden ser modificados** después de la creación de la tupla.

Al momento de establecer una tupla, debemos considerar las mismas reglas que se usan en una lista, exceptuando que la serie de valores en la tupla se **delimita por paréntesis**. Una tupla puede identificarse como una lista de constantes. Ejemplo:

Tipos básicos de datos en Python

Ventajas en el uso de tuplas

Usar tuplas tiene las siguientes ventajas:

- Nosotros podemos usar tuplas para diferentes tipos de datos y una lista para datos similares.
- Debido a que las tuplas son inmutables, iterar a través de la tupla es más rápido que una lista. Existe un ligera mejora en el desempeño del procesador de la computadora.
- Las tuplas al tener elementos que no cambian, pueden usarse como claves para un diccionario. Con las listas ésto no es posible.

Tipos básicos de datos en Python

Ventajas en el uso de tuplas

- Si tenemos datos que no cambian, almacenarlos como una tupla nos garantiza que los datos están protegidos ante una sobre escritura.

Ejemplos:

```
tup = ()   # Tupla vacia
tup = (23, 45, 78, 90)  # Tupla con un mismo tipo de dato
tupl = (67, "Windows", 3.1416)  # Tupla con diferentes tipos de
    datos
tup = (10, "Hola", (34,56,23), [9.81,3.1416])  # Tupla anidada
tup1 = 2, 5.6, "Linux", [6.7, 4.5] # Creando una tupla sin
    parentesis
```

Tipos básicos de datos en Python

Tipo de datos tuplas

```
tupl = (34,) # Creando una tupla de un elemento
tupl = 456, # Usar una coma la final se crea una tupla
tup[0]  # Visualizando el 1er elemento de la tupla
tup[2]  # Visualizando el 3er elemento de la tupla
tup[-1] # Visualizando el ultimo elemento de la tupla
tup[2][1] # Visualizando el 3er elemento de la tupla y el 2do
    elemento de la tupla anidada
tup[2][:] # Visualizando el 3er elemento de la tupla y todo el
    contenido de la tupla anidada
```

Tipos básicos de datos en Python

Tipo de datos tuplas

```
tup[3][1] # Visualizando el 4to elemento de la tupla y el 2do.
    elemento de la lista
t = ((2,4,6,8,10)+(3,6,9,12,15)+(4,8,12,16,20)) # Concatenando 3
    tuplas
t = ((2,4,6,8,10)+(3,6,9,12,15)*3+(4,8,12,16,20)) # Concatenando
    3 tuplas y repitiendo la 2da. tres veces
del tup1[2]  #Borrando el tercer elemento de la tupla
del tup1  # Eliminando la tupla
3 in t   # Evaluando si el valor de 3 se encuentra en la tupla t
q = list(t) # Convirtiendo la tupla t a lista
tt = tuple(q) # Convirtiendo la lista q a tupla
```

Tipos básicos de datos en Python

Tipo de datos diccionario

Los **diccionarios** son estructuras de datos donde podemos almacenar cualquier tipo de valor, ya sean números enteros, cadenas de carácteres, lista e incluso funciones. La peculiaridad de un diccionario es que cada elemento se identifica por un clave (key)

Para crear un diccionario, los valores deben **delimitarse por llaves** y cada valor es separado por comas; la clave y el valor se separan con dos puntos. Ejemplo:

```
# Estableciendo un diccionario vacio
huracan = {}
# Almacenando valores en un diccionario
huracan= {'Alice' : 1953, 'Alma' : 1970, 'Barry' : 2013 , \
        'Beta' : 2005, 'mismo_year' : ['Gert', 'Bert']}
```

Tipos básicos de datos en Python

Tipo de datos diccionario

```
# Visualizando los elementos de un diccionario
print huracan['Alice']
print huracan['Alma']
print huracan['mismo_year']
# Visualizando los elementos de la lista en el diccionario
print huracan['mismo_year'][0]
print huracan['mismo_year'][1]
# Visualizando todo el diccionario
for i in huracan:
        print i, ":", huracan[i]
```

Tipos básicos de datos en Python

Tipo de datos diccionario

```
# El metodo dict: a partir de una representacion de un
# diccionario devuelve un diccionario de datos
tide = dict(Compo='M2',Periodo=12.42, \
      Descripcion='Principal lunar')
# Zip recibe dos elementos iterables (mismas dimensiones) y
# devuelve un diccionario relacionado con el i-esimo
# elemento de cada uno de los iterables.
tides = dict(zip(['M2', 'S2','N2','K2'], \
      [12.42,12.00,12.66,11.97]))
asd = dict(zip('hola',[9,8,7,6]))
```

Tipos básicos de datos en Python

Tipo de datos diccionario

```
# Items devuelve una lista de tuplas, los 1eros elementos son
# la clave el segundo elemento es su valor
items = tides.items()
# keys regresa una lista de las etiquetas del diccionario
eti = tides.keys()
# values() regresa una lista de los valores del diccionario
val = tides.values()
# clear() elimina todos los elementos del diccionario
asd.clear()
```

Tipos básicos de datos en Python

Tipo de datos diccionario

```
# get() a partir de una clave, devuelve el valor de la clave
tides.get('N2')
# update() recibe otro diccionario y si alguna clave se repite
# se "borra" el valor antiguo y se agrega el valor nuevo
tides = dict(zip(['M2', 'S2','N2','K2'], \
      [12.42,12.00,12.66,11.97]))
tides2 = dict(zip(['01', 'S2','P1','Q1'], \
      [25.82,0,24.07,26.87]))
tides.update(tides2)
```