

Universidad Nacional de Jujuy Facultad de Ingeniería

Introducción a la Programación Programación I

Listas

Contenido

- Introducción
- Tipos de datos
- Conceptos. Listas simples, paralelas, anidadas
- Lista de listas de igual longitud ("tablas")
- Rebanadas - slicing
- Métodos internos - operaciones: agregar, buscar, modificar, insertar, eliminar.

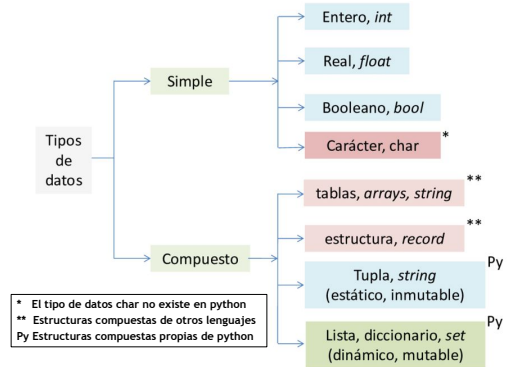
Introducción

Muchos programas trabajan con una gran colección de información similar

- Palabras en un documento
- Datos de un experimento
- Clientes de un negocio
- Representación gráfica de la información
- Estudiantes en un curso
- Obtener el promedio de las notas de un curso. Mostrar por pantalla dicho promedio y las notas ingresadas que sean mayores que él.

Programa = algoritmo + estructura de datos

Tipos de datos



Listas: concepto

Una lista es una secuencia dinámica y ordenada de ítems de cualquier tipo (heterogéneo), encerrada entre corchetes. Las listas son mutables. Se puede acceder a los ítems de una lista por medio de un índice (indexar). Un índice positivo significa contar hacia adelante desde el principio de la lista, un índice negativo significa contar hacia atrás desde el final de la lista.

- Las listas pueden ser:
 - uni-dimensional, listas simples (vector), se indexan por un solo índice
 - bi-dimensional (tabla-matriz), se indexan mediante dos índices
 - multi-dimensionales
 - paralelas, anidadas

mi_lista = [1, ['a', 'e', 'i', 'o', 'u'], 8.9, 'hola', True]

Listas: representación gráfica

nombre		nota		listado		
índice		índice		índice	0	1
0	Mary	0	70.5	0	Mary	70.5
1	Anna	1	63.8	1	Anna	63.8
2	Juan	2	30.3	2	Juan	30.3


```

nombre = ['Mary', 'Anna', 'Juan']
nota = [70.5, 63.8, 30.3]
print(nombre[2], nota[2]) #Juan, 30.3
print(nombre[-1], nota[-1]) #Juan, 30.3
  
```

```

listado = [['Mary', 70.5], ['Anna', 63.8], ['Juan', 30.3]]
print(tabla[2][0], tabla[2][1]) #Juan, 30.3
  
```

Operaciones: recorrido

- Recorrido 
- Creación
- Asignación de valores
- Concatenación
- Modificación de elementos
- Búsqueda

```
for i in range(len(nombre)):
    print(i, nombre[i], nota[i])
```

```
0 Mary 70.5
1 Anna 63.8
2 Juan 30.3
```

```
for i, item in enumerate(nombre): #(start=1)
    print(i, item)
```

```
0 Mary
1 Anna
2 Juan
```

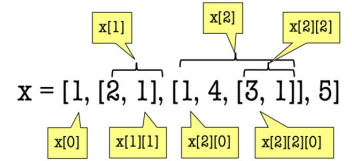
Listas anidadas: recorrido

```
b = [3, 1]
```

```
c = [1, 4, b]
```

```
a = [2, 1]
```

```
x = [1, a, c, 5]
```

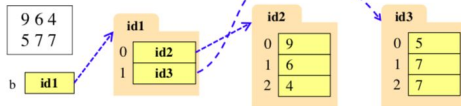


```
for pos, elem in enumerate(x):
    print(pos, elem)
```

```
0 1
1 [2, 1]
2 [1, 4, [3, 1]]
3 5
```

Listas de listas de igual longitud

```
b = [[9, 6, 4], [5, 7, 7]]
```



```
for i in range(len(b)):
    for j in range(len(b[0])):
        print(b[i][j], end=' ')
    print()
```

```
for elem in b:
    print(elem)
```

Listas de listas: tablas

```
d = [[5,4,7,3],[4,8,9,7],[5,1,2,3],[4,1,2,9],[6,7,8,0]]
```

```
for i in range(len(d)):
    for j in range(len(d[0])):
        print(d[i][j], end=' ')
    print()
```

	j	0	1	2	3
i	0	5	4	7	3
1	4	8	9	7	
2	5	1	2	3	
3	4	1	2	9	
4	6	7	8	0	

```
d[3][2] = 8
```

```
for pos, elem in enumerate(d):
    print(pos, elem)
```

```
0 [5, 4, 7, 3]
1 [4, 8, 9, 7]
2 [5, 1, 2, 3]
3 [4, 1, 8, 9]
4 [6, 7, 8, 0]
```

Rebanadas: slicing (sub - string)

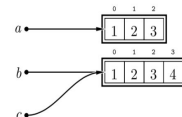
Las rebanadas pueden tener 1, 2 o 3 parámetros `lst[i:f:p]`, donde `lst` es la lista, el parámetro `i` es el inicio, `f` es el fin-1 y `p` es el paso. Dada la lista `vocales = ['a','e','i','o','u']`

vocales	'a'	'e'	'i'	'o'	'u'
índice	0	1	2	3	4
índice	-5	-4	-3	-2	-1

```
print(vocales[1]) #e
print(vocales[-5]) #a
print(vocales[1:3]) #['e','i']
print(vocales[1:-1]) #['e','i','o']
print(vocales[:3]) #['a','e','i']
print(vocales[2:]) #['i','o','u']
print(vocales[:1]) #['a']
print(vocales[:-1]) #['a','e','i','o']
print(vocales[::2]) #['a','i','u']
print(vocales[1::2]) #['e','o']
print(vocales[::-1]) #['u','o','i','e','a']
```

Crear, asignar, modificar, concatenar

```
lst1 = [] #lista vacía
lst2 = list() #lista vacía
lst3 = list(range(6,2,-2)) #[6, 4]
lst4 = list('aeiou') #['a','e','i','o','u']
lst5 = [3, 5, 7]
lst5.append(10) #[3, 5, 7, 10]
lst5[len(lst5) - 1] = 11 #[3, 5, 7, 11]
lst5.insert(0, 2) #[2, 3, 5, 7, 11]
#El operador + y el * generan nuevas listas
lst6 = lst3 * 2 #[6, 4, 6, 4]
a = [1,2,3]
b = a + [4]
c = b
print(a) #[1, 2, 3]
print(c) #[1, 2, 3, 4]
```



Listas por comprensión

Las listas definidas por comprensión son aquellas donde sus elementos se describen a través de las propiedades que tienen en común.

lista = [<expresión> for <item> in <iterable> if <condición>]

```
lst1 = [3, 5, 6, 4, 1]
lst2 = [4, 0, 3, 9, 2]
```

```
res1 = [x for x in lst1 if x > 3] #[5, 6, 4]
res2 = [lst1[i] + lst2[i] for i in range(1, len(lst1), 2)]#[5, 13]
res3 = [w for w in lst1 for z in lst2 if w == z] #[3, 4]
res4 = [x + 1 if x < 5 else x for x in lst1] #[4, 5, 6, 5, 2]
res5 = [w*z for w,z in zip(lst1, lst2)] #[12, 0, 18, 36, 2]
```

Operador is, in

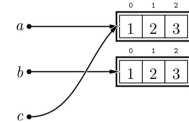
Dos variables pueden apuntar a la misma zona de memoria o bien a diferentes zonas. Por ejemplo:

```
a = [1,2,3]
b = [1,2,3]
c = a

print(a==b) #True
print(b > c) #False
print(2 in a) #True
```

El operador **is** devuelve True si dos objetos son en realidad el mismo objeto, es decir, si residen ambos en la misma zona de memoria, y False en caso contrario

```
print(a is b) #False
print(a is c) #True
```



Métodos internos

Método	Función o significado
lista.append(x)	Agrega x al final de lista
lista.extend(otraLista)	Agrega la otraLista al final de lista
lista.insert(i, x)	Inserta x en la posición i de lista y mueve los otros ítems a la derecha
lista.remove(x)	Elimina la primera ocurrencia de x en lista
del lista[i]	Elimina el elemento de la posición i de lista
Item = lista.pop(i)	Elimina el elemento de la posición i de lista y retorna su valor
lista.sort()	Ordena los elementos de lista
lista.reverse()	Invierte los elementos de lista
lista.index(x)	Retorna el índice de la primera ocurrencia de x
lista.count(x)	Retorna la cantidad de ocurrencias de x

Ejemplos

```
lst = [3, 5, 2, 2, 7, 11, 5, 13]
print(len(lst), max(lst), min(lst), sum(lst))
lst.append(2) # agrega al final
print(lst.count(2)) #3
print(lst.index(5)) #1
print(lst.index(5, 2)) #6
lst_copia = lst.copy() #copia lst
lst_copia.reverse() #invierte lst
del lst_copia[2:4]
print(lst_copia) #[2, 13, 7, 2, 2, 5, 3]
```

Ejemplos

```
lst_nueva = [17, 19, 23]
lst.extend(lst_nueva) #agrega lst_nueva a lst
lst.insert(len(lst)//2, 0) # inserta 0 al medio
lst.remove(11) #elimina primer 11 sino hay 11 da error
lst.pop(3) #elimina el valor en posición 3
#si no hay parámetro elimina el último
print(lst) #[3, 5, 2, 7, 0, 5, 13, 2, 17, 19, 23]
lst.clear() #borra la lista, del lst[:]
```

Cargar lista

lista	'd'	'c'	'e'	'a'	'c'	'f'	...
índice	0	1	2	3	4	5	...



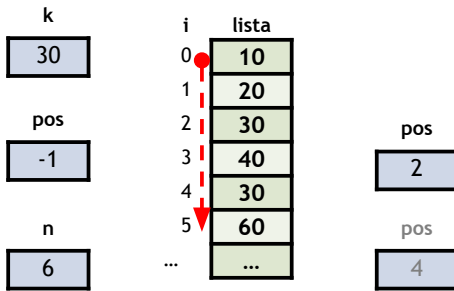
```
def cargar1():
    lista = []
    n = int(input('cantidad:'))
    for i in range(n):
        item = input('item:')
        lista.append(item)
    return lista

lst1 = cargar1()
```

```
def cargar2():
    lista = []
    item = input('item (") fin:')
    while (item != ""):
        lista.append(item)
        item = input('item (") fin:')
    return lista

lst2 = cargar2()
```

Buscar



Búsqueda de k

```
def buscar1(lista,k):
    pos = -1
    n = len(lista)
    for i in range(n):
        if lista[i] == k:
            pos = i
    return pos
```

```
def buscar3(lista,k):
    pos = -1
    for i,item in enumerate(lista):
        if item == k:
            pos = i
            break
    return pos
```

```
def buscar2(lista,k):
    pos,i = -1,0
    n = len(lista)
    while (i < n) and (pos == -1):
        if lista[i] == k:
            pos = i
        else:
            i += 1
    return pos
```

Promedio de la clase

Legajo	Nombre	Nota
88	Mary	70.5
29	Anna	63.8
53	Juan	30.3

```
lista = [[88,'Mary',70.5],[29,'Anna',63.8],[53,'Juan',30.3]]
```

Promedio de la clase

```
def leer():
    n = int(input('Ingrese nro estudiantes: '))
    lista = []
    for i in range(n):
        legajo = int(input('Legajo: '))
        nombre = input('Nombre: ')
        nota = float(input('Nota: '))
        lista.append([legajo,nombre,nota])
    return lista
```

```
def mostrar(lista):
    for pos, elem in enumerate(lista):
        print(pos,elem)
```

```
def suma(lista):
    suma = 0
    for elem in lista:
        suma += elem[2]
    return suma
```

```
#principal
estudiantes = leer()
mostrar(estudiantes)
print('Promedio:',promedio(estudiantes))

def promedio(lista):
    return suma(lista)/len(lista)
```

Listas anidadas: métodos internos

```
lista = [[88,'Mary',70.5],[29,'Anna',63.8],[53,'Juan',30.3]]
nuevo1 = [88,'Pedro',41.5]
nuevo2 = [60,'Tom',35.7]
```

```
lista.append(nuevo1)
0 [88, 'Mary', 70.5]
1 [29, 'Anna', 63.8]
2 [53, 'Juan', 30.3]
3 [88, 'Pedro', 41.5]
```

```
lista.insert(2,nuevo2)
0 [88, 'Mary', 70.5]
1 [29, 'Anna', 63.8]
2 [60, 'Tom', 35.7]
3 [53, 'Juan', 30.3]
4 [88, 'Pedro', 41.5]
```

```
lista.pop(3)
0 [88, 'Mary', 70.5]
1 [29, 'Anna', 63.8]
2 [60, 'Tom', 35.7]
3 [88, 'Pedro', 41.5]
```

Bibliografía

- https://www.w3schools.com/python/python_lists.asp
- <https://www.askpython.com/python/list/iterate-through-list-in-python>
- <https://j2logo.com/python/tutorial/tipo-list-python/>
- Libro: Introducción a la Programación con Python. Capítulo 5
- <https://docs.python.org/es/3/>