Clase 4 Listas II y Matrices

TEMAS: Listas. Rebanadas. Concepto de iterable. Instrucción for. Listas por comprensión. Matrices. Implementación en Python.

Rebanadas

Una **rebanada**(slice) es una manera de referirse a un grupo

de elementos perteneciente a una lista.

En lugar de usar un solo subíndice se utilizan dos o tres , separados por “dos puntos”.

```
lista = [7,8,9,10,11,12]
sublista = lista[2:5]
'''Los dos subindices indican el inicio y el fin de la rebana
da'''
'''El subindice final no esta incluido'''
"-----------------------------------------------"
'''Dejar en blanco alguno de los subindices hace que se consi
dere el extremo de la lista'''
lista1 = [7,8,9,10,11,12]
lista2 = lista1[3:] #[10,11,12]
lista3 = lista[ :3] #[7,8,9]
```

Cuando se usan tres subindices, el tercer actúa como **incremento**.

```
lista = [7,8,9,10,11,12,13,14]
sublista = lista[1:6:2]
print(sublista) #[8,10,12]

'''Un incremento negativo toma los elementos de atras hacia a
delante.'''
original = [1,2,3,4,5]
invertida = original[ : :-1]
print(invertida) #[5,4,3,2,1]
```

> reverse in situ → cambia la lista en si // [::-1] crea una copia de la lista. la original queda igual

```
lista = [1,2,3,4,5,6,7,8]
a = 3
b = 5
sublista = lista[a:b]
print(sublista) #[4,5]
```

Las rebanadas funcionan con variables

→ Una rebanada nula es una rebanada que no contiene ningún elemento.

Se crean utilizando el mismo subíndice para inicio y fin

Recordar que el subindices final no esta incluido.

```
lista = ['a','b','c','d']
lista[2:2] = ['X','Y']  #prim
er nro es la posición
print(lista) #
['a','b','X','Y','c','d']
```

Se utilizan para insertar elementos en masa en una lista.

EJEMPLO 1

Obtener sublistas con los primeros y últimos N elementos de una lista

```
lista = list(range(10))
n = int(input("Cuantos elementos desea tomar?"))
comienzo = lista[:n] #N elementos del comienzo
final = lista[-n:] #N elementos del final
extremos = comienzo+final
print()
print("Lista original" , lista)
print("Comienzo:" , comienzo)
print("Final:" , final)
print("Comienzo+Final:",extremos)

#Ejemplo de ejecucion
#Cuantos elementos desea tomar? 3
#Lista original [0,1,2,3,4,5,6,7,8,9]
#Comienzo [0,1,2]
#Final [7,8,9]
#Comienzo+Final [0,1,2,7,8,9]
```

Comparación de listas

Las listas pueden ser comparadas como cualquier otra variable.

La comparación se realiza elemento a elemento.

```
[2,3] > [1,4] #True
[2,3] > [2,4] #False ambiguedad de 2 > 2 -- 3
> 4 no. False
[2,4,6] > [2,4] #True 6 > que nada . True
```

Copia de listas

Copiar una lista solo copia la referencia al objeto.

```
lista1 = [1,2,3]
lista2 = lista1
lista2.append(4)
print(lista1) #[1,2,3,4]
#Las variables lista1 y lista2 apuntan al mismo objeto en mem
oria.

'''Se puede verificar mediante la funcion id(<objeto>), que d
evuelve  un nro que representa la identidad de un objeto y es
equivalente a su direccion de memoria.'''

lista1 = [1,2,3]
lista2 = lista1
print(id(lista1), id(lista2))
#Ej   188989899   188989899

#TÉCNICAS DE ID

'''Podemos realizar una copia a traves de rebanadas. MÁS PERF
ORMANCE'''
lista = [1,2,3]
lista2 = lista1[ : ] ---> abarca todos los elementos. una nue
va lista copia y pega
lista2.append(4)
print(lista1) #[1,2,3]

'''Utilizar la funcion list()'''
lista = [1,2,3]
lista2 = list(lista1) ---> copio y pego, una lista
lista2.append(4)
print(lista1) #[1,2,3]

'''Crear una lista nueva concatenando una lista vacia. truch
a'''
lista = [1,2,3]
```

```
lista2 = lista1 + [] ---> concateneo una lista vacia
lista2.append(4)
print(lista1) #[1,2,3]

'''Utilizar el metodo copy()  MÉTODO NUEVO. MÁS PERFORMANC
E'''
lista = [1,2,3]
lista2 = lista1.copy
lista2.append(4)
print(lista1) #[1,2,3]
```

Uso de for con listas

La instrucción **for** puede utilizarse para recorrer listas sin necesidad de range().

En todo caso la variable usada en el for recoge todo el elemento de la lista, y no su subíndice.

```
vocales = ['a','e','i','o','u']
for letra in vocales:
    print(letras, end=" ") #a e i o u

'''Tambien puede usarse una rebanada para recorrer la lista p
arcialmente'''
vocales = ['a','e','i','o','u']
for letra in vocales[1:4]:
    print(letras, end=" ") # e i o

'''Ademas del elemento se requiere su subindice, puede usarse
la
funcion enumerate().'''
vocales = ['a','e','i','o','u']
for i,letra in enumerate(vocales):
    print(f"{i} {letra} ", end= " ") #0a,1e,2i...

'''enumerate() devuelve una pareja de valores formada por el
```

```
subindice
y el elemento correspondiente'''
(<subindice> , <elemento>)
'''Esta pareja de valores recibe el nombre de tupla, y se des
empaqueta
en dos variables'''
```

Función pass

- no hace nada
- situaciones especiales
- por lo general se usa mal

```
def calccularsalario(empleado):
   pass #aún no implementado. a = 0 es lo mim
smo

"""ejemplo de utilización """
if nota <4:
   recuperan.append(nombre)
```

Función map

```
<lista2> = list(map(<funcion>,<lista1>))
#la funcio se le aplica a cada uno de los ele
mentos de la lista
#resultados se convierte en una lista anueva

#Ejemplo
numeros = [1,2,3,4]
raices = list(map(lambda x: x**(1/2), numero
s)) #genera una secuencia de valores
#Equivalente a
numeros = [1,2,3,4]
raices = []
for i in numeros:
    raices.append(i**(1/2))
```

- La función map aplica una función cualquiera a todos los elementos de una lista.
- devuelve una lista del msimo largo que la orginal, pero los valores no lo son

```
'''Las funciones lambda son ideales para esto
s casos.'''
'''La funcion list() es necesaria para conver
tir a formato de lista el objeto devuelto por
map().'''
```

Función filter()

La función filter selecciona algunos elementos de una lista para crear una nueva lista con ellos.

BOOLEANA → True or False

Los elementos de la lista original que se añaden a la nueva lista son aquellos que devuelven True al aplicarles una función.

```
#Ejemplo
numeros = [0,1,2,3,4,5]
impares = list(filter(lambda x: x%2!=0, n
umeros))
print(impares) #[1,3,5]
```

Descarta todos los valores que le dio False y se queda con los que le dio True

> map → lista del mismo largo de la lista original y los valores transformados que se usan en el print

> filter → lista más corta o el mismo largo que la original y la lista tiene los valores de la lista original sin alterarlos

Listas por comprensión

- teoría de conjuntos
  - dos maneras de describirlos
  - por extensión: nombrar todos los elementos que contienen
  - por comprensión: describir las carac que deben tener los integrantes del conjunto

Las listas por comprensión son una manera matemática para crear listas.

Los corchetes son necesario para crear la lista.

La función list() también sirve.

```
<lista> = [<expr> for <elem> in <secuencia>]

'''La expresion <expr> representa alguna oper
acion que se aplica a cada elemento<elem> de
<secuencia>.
El resultado de esta expresion se agregara a
<lista>.'''

#ejemplo
cuadrado = [x**2 for x in range(6)]
print(cuadrado)= [0,1,4,9,16,25]
```

```
cubospares = [i**3 for i in range(11) if
i**3 % 2==0]
print(cubospares)
#[0,8,64,216,512,1000]
```

Se puede agregar un if para seleccionar elementos y rechazar otros

> En muchos casos las función map y filter pueden ser reemplazadas por listas por comprensión.

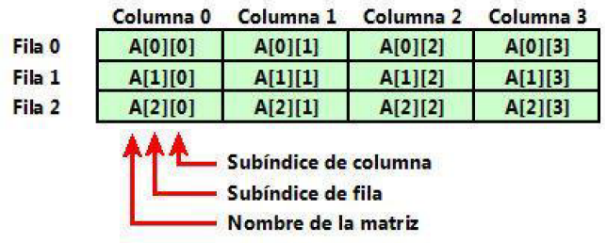
Matrices → de dos dimensiones

Estructura de datos formadas por filas y columnas.

Se vera solo matrices rectangulares o cuadradas.

- ▼ Se las simula construyendo una lista de listas.

  es una lista que es un si interior contiene lsitas



Ambos comienzan desde 0

Forma estática o dinámica

Se necesitan dos subindices, **el primero se refiere a las filas y el segundo a las columnas.**

```
'''Alternativa 1 : Crear la matriz como una a lista de listas
en forma estatica
----> hay que saber el tamaño ya de entrad'''
matriz = [ [0,0,0,0],
           [0,0,0,0],
           [0,0,0,0] ]

matriz = [ [0,0,0,0], [0,0,0,0] , [0,0,0,0] ]

for i in range(1,5):
    for j in range(1,5):
        print(0, end=" ")
    print( )
```

```
#0 0 0 0
#0 0 0 0
#0 0 0 0
#0 0 0 0

#---------Fin creacion de la matriz--------

'''Alternativa 2 : Crear la matriz como una lista de listas e
n forma dinamica'''
filas = 3 #int(input())
columnas = 4 #int(input())
matriz = [] #!!!!!!!!!!!!!!!!!!!!!
for f in range(filas):
    matriz.append([]) ----> fila vacia por el momento
    for c in range(columnas): ---> iteración lo que indica co
lumnas
        matriz[f].append(0) ---> le agrefamos a matriz sub f.
la fila que agregue vacía


#[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
#---------Fin creacion de la matriz--------


'''Alternativa 3: Usando poder de replicacion'''
filas = 3
columnas = 4
matriz = []
for f in range(filas):
    matriz.append([0]*columnas) ---> fila ya armada . repito
la cantidad de elementos en la lista

#[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
#---------Fin creacion de la matriz--------

'''Alternativa 4: Usando replicacion y listas por comprension
```

```
|  FAVORITA DE THOMPSON'''
filas = 3
columnas = 4
matriz = [[0] * columnas for i in range(filas)]

#[[0, 0, 0, 0], [0, 0, 0, 0], [0, 0, 0, 0]]
#---------Fin creacion de la matriz--------
```

EJERCICIO 2: RELLENAR UNA MATRIZ

Una vez creada la matriz se rellena con números. Podemos ingresarlos por teclado. Luego se imprimirá por pantalla.

Ambas tareas a traves de funciones.

```
'''Lectura de datos'''
def rellenarmatriz(matriz):
#autodetectamos el tamaño de la matriz
filas = len(matriz)
columnas = len(matriz[0])
for f in range(filas):
    for c in range(columnas):
        n = int(input("Ingrese un numero: "))
        matriz[f][c] = n

#no hay return. las listas son mutables

'''Impresion de la matriz'''
def imprimirmatriz(matriz):
#autodetectamos el tamaño de la matriz
    filas = len(matriz)
    columnas = len(matriz[0])
    for f in range(filas):
    for c in range(columnas):
        print("%3d" %matriz[f][c], end="")
```

```
    print() #avanzar a la prox fila de la pantalla para que l
a otra aparezca abajo

#Programa Principal
filas = 3
columnas = 4
matriz = []
for i in range(filas):
    matriz.append([0]* columnas)

#---------Fin creacion de la matriz--------
rellenarmatriz(matriz)
imprimirmatriz(matriz)
```