Clase 17: Estructuras Indexadas (parte II)

Funciones para listas

Notas:

- L se considera un alias del argumento
- si se modifica parámetro L dentro de la función afecta también al argumento (lista) utilizado en la invocación

Ejemplo: Sumar valores de una lista

```
#suma: list (num) -> num
#sumar numeros de una lista
#ej: suma([20,30,10])->60

def suma(L):
    assert type(L)==list
    total=0
    for valor in L:
        total = total + valor # total+=valor
    return total

assert suma([20,30,10])==60
```

```
# usando range
def suma(L):
    assert type(L)==list
    total=0
    for i in range(len(L)): # [0,1,..,len(L)-1]
        total = total + L[i]
    return total

assert suma([20,30,10])==60
```

Ej. Indicar el índice de una lista donde se encuentra cierto valor

```
#indice: any list(any) -> int
#indice (desde 0) de 1ª aparición de x en L
#-1 si no está
#ej: indice(30,[20,30,10])->1
```

```
def indice(x,L):
    assert type(L)==list
    for i in range(len(L)): #[0,1,..,len(L)-1]->[0,1,2]
        if L[i]==x: return i
    return -1

assert indice(30,[20,30,10]) == 1
assert indice(40,[20,30,10]) == -1
```

¿Cómo imprimir los valores de una lista?

```
L = [10, 20, 30, 40, 50,60,70]
for valor in L:
  print(valor)
```

```
10
20
30
40
50
60
70
```

Ej. Realizar una copia 'efectiva' de una lista

```
#copia: list -> list
def copia(L):
    assert type(L)==list
    duplicado=[]
    for valor in L:
      duplicado.append(valor)
    return duplicado


assert copia([8,9,7,9])==[8,9,7,9]
```

Instrucción 'while'

Esta instrucción es más general que la instrucción `for` y permite iterar sobre un bloque de instrucciónes, su sintaxis es:

```
while condicion: # condición es de tipo boolean
    # bloque de instrucciones
```

las instrucciones del ciclo **`while`** se ejecutarán mientras **`condición`** sea igual a **`True`**.

```
lista = [10,20 ,30, 40, 50]
indice = 0
while indice < len(lista):
    print (lista[indice])
    indice = indice +1
```

```
10
20
30
40
50
```

```
#suma: list(int) -> int
def suma(L):
    assert type(L)==list
    suma = 0
    indice = 0
    while indice < len(L):
        suma+= L[indice]
        indice+=1
    return suma

assert suma([8,9,7,9])==33
```

Strings son listas inmutables

Los strings operan como una estructura indexada de caracteres. Es decir, una lista de caracteres inmutable:

```
palabra = "algoritmo"
palabra[2]
```

```
'g'
```

```
otrapalabra = "ALGORITMO"
palabra+otrapalabra
```

```
'algoritmoALGORITMO'
```

```
len(palabra)
```

```
9
```

```
palabra < otrapalabra
```

```
False
```

```
'a'>'A'
```

```
True
```

```
palabra[2:len(palabra)]
```

```
'goritmo'
```

```
'a' in palabra
```

```
True
```

```
palabra = "algoritmo"
for caracter in palabra:
    print (caracter)
```

```
a
l
g
o
r
i
t
m
o
```

```
def inverso(x):
    s = ''
    for i in  range(len(x)-1,-1, -1):
        s = s + x[i]
    return s
assert inverso("roma")=="amor"
```

```
inverso('hola')
```

```
'aloh'
```

```
def inverso(x):
    s =''
    for c in x:
      s=c+s
    return s

assert inverso("roma")=="amor"
```

```
def capicua(x):
    return x==inverso(x)
assert capicua("reconocer")
```

otras funciones de python para strings

```
x = "operaciones de strings en python"
```

```
x.find("s")
```

```
10
```

```
x.count("s")
```

```
3
```

```
x.isalpha()
```

```
False
```

```
x.islower()
```

```
True
```

```
x.isupper()
```

```
False
```

```
x.upper()
```

```
'OPERACIONES DE STRINGS EN PYTHON'
```

```
x.lower()
```

```
'operaciones de strings en python'
```

```
y = " " + x + " "
y
```

```
' operaciones de strings en python '
```

```
y.strip()
```

```
'operaciones de strings en python'
```

```
x.replace("o","s")
```

```
'speracisnes de strings en pythsn'
```

Ej. ¿Cómo ver si una frase es capicúa?

```
def fraseCapicua(frase):
    frase=frase.replace(" ", "") #eliminar espacios
    frase=frase.lower()  #todo a minusculas
    return capicua(frase)
assert fraseCapicua("Anita lava la tina")
```

Lista de listas (lista de 2 dimensiones o tabla o matriz)

```
T=[["gabriela","jose","rosa","matias"],\
   [62,48,56,49],\
   [45,52,35,48]]
#promedios por pregunta (por filas)
for i in range(1,len(T)):
    print ("promedio pregunta",i,float(sum(T[i]))/len(T[i]))

#promedios por alumno (por columnas)
for j in range(len(T[0])):
    s=0.0
    for i in range(1,len(T)):
        s+=T[i][j]
    print ("promedio: ",T[0][j],s/(len(T)-1))
```

```
promedio pregunta 1 53.75
promedio pregunta 2 45.0
promedio:  gabriela 53.5
```

```
promedio:  jose 50.0
promedio:  rosa 45.5
promedio:  matias 48.5
```