Bitácora 5 del Taller de Herramientas Computacionales

Elías Jiménez Cruz, 409085596

18/01/2019

En la clase 10 empezamos a ver listas en Python, que son una estructura de datos bastante importante para este lenguaje de programación. Si ha de definirse en lenguaje coloquial de alguna manera, es un conjunto ordenado, el cual consta de elementos que tienen una posición en dicho conjunto. Para construirlo, basta con poner los elementos en el orden deseado, separados por comas, y encerrarlos dentro de corchetes. "[Esto, es, una, lista, de, 7, elementos]". Puede asignársele un nombre y guardarla como si se tratase de una variable cualquiera. Al fin y al cabo es un objeto, al igual de una cadena o un número entero, sólo que con propiedades diferentes. Ahora bien, divergiendo un poco en el tema, existe un comando llamado bool, el cual permite determinar si una variable de algún tipo determinado está vacía o guarda algún valor relacionado con nulidad, dependiendo de su tipo de variable. Si no es nulo, devuelve verdadero; si lo es devuelve falso. Entonces, en el caso de números enteros, será nulo su valor si es cero, en el caso de cadenas, será nulo si la cadena está vacía. En el caso de las listas, si efectivamente está vacía la lista, es decir, si es igual a []. Ahora bien, también se checó otro comando aplicable únicamente a las listas, el comando len, el cual devuelve el número de elementos de una lista en particular. Es decir, si ponemos len([a,b,c,d]), len será igual a 4. La anterior es la forma de invocar la instrucción. También vimos algunos métodos de las listas, el .append("valor"), que agrega un elemento a la cola de la lista, el .insert(posicion, valor), en

cual agrega en la posición deseada de la lista un valor dado, desplazando a la derecha el valor que ocupaba esa posición. También vimos .pop() y .pop(posicion) el cual quita, en el primer caso, de la lista el último elemento o, en el segundo caso, el elemento cuya posición se determina. Este valor quitado se puede guardar como variable. Se vio el método .extend("lista") en el cual a una lista dada se le pueden agregar los elementos de otra lista, que quedarían a la cola de la primera. Ahora, la manera en que se determinan las posiciones de los elementos de una lista es simple, el primer elemento será el elmento 0, y el último será el elemento len(lista)-1, lo que indica que todos los elementos se enumerarán del 0 al len(lista).1. Si se quiere imprimir o guardar un elemento determinado de una lista, basta con escribir lista[posicion], lo cual regresará el elemento solicitado. Ahora bien, es posible meter una lista dentro de otra, como si fuera un elemento cualquiera, por lo que si se quiere imprimir o guardar un elemento de esa lista anidada, basta con poner lista[posicionDeLaListaAnidada][Elemento], lo cual devolverá el valor sin problemas.//Ahora, se pasó a ver otro ciclo de Python, el ciclo for, el cual permite que se repita un proceso por cada elemento de una lista determinada. He aquí una transcrpición de los códigos vistos en clase:

```
{C = -20
iC = 5
gradosC = []
while C <= 40:
F = (9.0/5)*C +32
print C, F
#C = C + iC
gradosC.append(C)
C += iC
print '\n'

print '   c    F'
for grado in gradosC:
F=(9.0/5)*grado+32
print '%5d %5.1f' %(grado, F)
print '\n'
```

```
indice = 0
print '   c    F'
while indice < len(gradosC):
C = gradosC[indice]
F=(9.0/5)*C+32
print '%5d %5.1f' %(C, F)
indice += 1
print '\n'

gradosC=[]
for C in range(-20,45,5):
gradosC.append(C)
print gradosC
print '\n'

gradosC=[]
for i in range(0,21):
C=-20 + i*2.5
gradosC.append(C)
print gradosC
print '\n'}
```