Introducción a la Programación

Práctica 5 – Listas

Versión del 4 de febrero de 2022

1. Listas

Muchas veces necesitamos trabajar con un conjunto de números, por ejemplo, imaginemos la situación siguiente: queremos calcular el promedio de cinco números. Una primera aproximación a este problema podría ser crear la cantidad necesaria de variables individuales; leemos 5 valores desde el teclado, los asignamos a distintas variables, calculamos el promedio de estos 5 valores y mostramos el resultado. Por ejemplo, la función para calcular dicho promedio sería la siguiente:

```
promedio(a0, a1, a2, a3, a4):
    return (a0+a1+a2+a3+a4)/5
```

No es una función muy complicada. Sólo faltaría escribir el programa principal que llame a dicha función. Debe leer los 5 valores y mostrar el promedio calculado. Ahora bien, ¿qué sucede si queremos calcular el promedio de 100 números ó de 1000 números? La solución es usar un conjunto de variables de la forma a[i], donde i es una variable entera que especifica un elemento en particular del conjunto. Al valor entero entre corchetes [ ] que se usa para referirse a cada elemento de un arreglo se lo conoce como el índice de la lista. Las listas son secuencias ordenadas de valores, permiten guardar una cantidad ilimitada de elementos, accediéndolos por su posición (sin olvidar que la primer posición es la 0). Se puede imprimir un elemento de la lista (print($a[2]$) o toda la lista completa (print(a)).
También se puede cambiar el valor de un elemnto de la lista ($a[1] = 5$), en este caso si la lista llamada a contenía $[3, 6, 7, 9]$ como se modificó lo que contenía en la posición 1 la lista ahora tendrá estos valores $[3, 5, 7, 9]$.

1.1. Funciones para trabajar con listas

Existen funciones que nos facilitan el uso de las listas, la función *len* permite conocer la cantidad de elementos de una lista. Por ejemplo, si la lista es $lista = [7, 3, 4, 6, 10]$ y $longitud = len(lista)$ ¿Qué valor se asignará a *longitud*? En *longitud* se asignará 5 que es la cantidad de elementos que posee *longitud*, donde sus índices serán $0, 1, 2, 3, 4$.
La función *append* permite agregar un elemento al final de la lista, se usa de la siguiente manera (*lista* es la lista que nombramos recien): $lista.append(2)$ de esta manera si mostramos que contiene *lista*, haciendo $print(lista)$ obtendremos $[7, 3, 4, 6, 10, 2]$ y si hacemos $print(len(lista))$ ahora obtendremos 6.
Como es de esperar también tenemos una función que nos permite quitar elementos de la lista (*pop*) pero... no podemos decirle a la función que elemento queremos quitar sino que le decimos que posición ocupa el elemento que deseamos quitar. Veamos un ejemplo con la lista anterior:

```
lista=[7,3,4,6,10,2]
lista.pop(3)
print(lista)
```

imprimirá en pantalla $[7, 3, 4, 10, 2]$ Existen en python muchísmas otras funciones para trabajar con listas como (*count*,*del*,*reverse*) pero que no se pueden usar en las prácticas ni en los parciales ya que la idea de esta materia es que ustedes sean capaces de hacer sus propias funciones.

1.2. Recorrer listas

Podemos recorrer listas con while y con for:

```
animales = [‘‘gato’’, ‘‘perro’’, ‘‘raton’’]
i = 0
while (i < len(animales)):
    print(animales[i])
    i = i + 1

for i in range(len(animales)):
    print (animales[i])

for elemento in animales:
    print (elemento)
```

Los primeros dos ciclos permiten recorrer la lista por índice, la variable i vale 0 inicialmente para poder acceder al elemento que tiene la lista *animales* en su primer posición y en cada iteración incrementa su valor hasta 2 para obtener el último elemento de la lista que es *animales*[2], claramente es más cómodo utilizar ciclos *for* en estos casos pues uno imagina que la función *range* fue diseñada para este trabajo. En el último ciclo se recorre la lista por elemento, en este caso la variable *elemento* va cambiando en cada iteración, en la primera será *gato*, luego *perro* y en la última iteración *raton* ignorando cual es la posición que ocupa cada elemento en la lista.

Notas preliminares

- Los ejercicios marcados con el símbolo ★ constituyen un subconjunto mínimo de ejercitación. Sin embargo, aconsejamos fuertemente hacer todos los ejercicios.
- Todos los ejercicios de esta práctica que involucren definir funciones deben ser probados en un programa aparte que no pida datos ingresados por el usuario sino que directamente imprima en pantalla muchas llamadas a dichas funciones.

Ejercicio 1 ★

Hacer un programa que guarde la siguiente lista en una variable: `["elefante", "jirafa", "mono"]`, luego pida el nombre de otro animal, lo agregue a la lista e imprima en pantalla el cuarto animal de la lista.

Ejercicio 2 ★

Definir una función llamada `mostrarEnUnaLinea` que tome una lista de enteros y muestre todos sus elementos en una linea separados por espacios.

Ejercicio 3 ★

Definir una función llamada `divisores` que tome un entero y devuelva la lista de divisores de ese entero.

Ejercicio 4 ★

Definir una función llamada `laMasCorta` que tome dos listas y devuelva la que tenga menos elementos. Si tienen igual cantidad, deberá devolver la primera.

Ejercicio 5 ★

Definir una función llamada `sonFactores` que tome un entero `n` y una lista de enteros, y devuelva `True` si los números de la lista son factores de `n` (es decir, si `n` es divisible por todos ellos).

Ejercicio 6 ★

Definir una funcion que tome una lista y devuelva `True` si tiene al menos un elemento repetido, y `False` en caso contrario.

Ejercicio 7 ★

Definir una función llamada `dondeAparece` que tome una lista de enteros y un entero llamado `blanco` como parámetros, y devuelva el primer índice donde `blanco` aparece en el arreglo, si lo hace, y -1 en caso contrario.

Ejercicio 8 ★

Hacer una función que tome una lista de números decimales y devuelva el promedio de los elementos.

Ejercicio 9 ★

Escribir una función llamada `maximo` que tome una lista de números y devuelva el valor del máximo elemento.

Ejercicio 10 ★

Escribir una función llamada `maximoIndice` que tome una lista de números y devuelva el índice del máximo elemento.

Ejercicio 11 ★

Escribir una función llamada `maximoEntre` que tome una lista de números y dos enteros `a` y `b` menores que la longitud de la lista y devuelva el índice del máximo elemento considerando solo los que están entre el índice `a` y el índice `b`.

Ejercicio 12 ★

Escribir una función llamada `intercambiar` que tome una lista de números `s` y dos enteros positivos `a` y `b` menores que la longitud de la lista y cambie el elemento ubicado en `s[a]` por el elemento ubicado en `s[b]`. Ojo, esta función no debe devolver una lista, sino modificar la que toma como parámetro.

Ejercicio 13 ★

Escribir una función llamada `frecuencia` que tome una lista de enteros `s` y otro entero `n` como parametros y devuelva la cantidad de veces que aparece `n` en `s`.

Ejercicio 14 ★

Definir una función llamada `interseccion` que tome dos listas sin repetidos y devuelva una nueva lista que contenga sólamente aquellos elementos que estén ambas listas.

Ejercicio 15 ★

Definir una función llamada `union` que tome dos listas sin repetidos y devuelva una nueva lista que contenga los elementos de ambas listas. Ojo, la lista de retorno debe no tener repetidos.

Ejercicio 16 ★

Definir una función llamada `diferencia` que tome dos listas sin repetidos y devuelva una nueva lista que contenga los elementos la primera que no estén en la segunda.

Ejercicio 17 ★

Definir una función llamada `mcd` que tome dos enteros positivos y devuelva el máximo común divisor usando funciones de los ejercicios anteriores.

Ejercicio 18 ★

Definir una función que tome un entero `n` y devuelva los primeros `n` primos.

Ejercicio 19 ★

Definir una función que tome un entero `n` y devuelva la lista de los primos que aparecen al factorear `n`. Ejemplo, para 24, la lista debe ser: `[2, 2, 2, 3]`

Ejercicio 20 ★

a) Definir una función que tome una lista de números `s` y un número decimal x y devuelva la cantidad de elementos de `s` que sean menores que x.
b) Si se pone como condición que `s` siempre esté ordenada de mayor a menor, ¿cómo podría modificarse el programa para que haga menos iteraciones?

Ejercicio 21 ★

Definir una función que tome una lista de números `s` y un número decimal x y cambie todos los elementos menores que x por 0.

Ej:

Para

```
s = [1, 4.1, 6.3, 2, 3.2, 8]
x = 3
```

el la lista debe pasar a ser:

```
s = [0, 4.1, 6.3, 0, 3.2, 8]
```

Ejercicio 22 ★

Escribir un programa que pida al usuario una cadena, y luego escriba en pantalla la cantidad de veces que aparece cada letra (sin mostrar las que no aparecen). Ej:

```
Palabra ingresada: "conocido"

c : 2
o : 3
n : 1
i : 1
d : 1
```

Ejercicio 23 ★

Modificar el programa del ejercicio 1.2 para que muestre visualmente los resultados, repitiendo asteriscos. Ej:

```
Palabra ingresada: "conocido"

c : **
o : ***
n : *
i : *
d : *
```

Ejercicio 24 ★

Auxilio Mecánico

En una empresa de auxilio mecánico existen dos coberturas para los clientes, *Oro* y *Plata*. Los clientes con cobertura *Oro* tienen más beneficios que los de cobertura *Plata* y pagan mensualmente un abono mayor. Por ejemplo los clientes *Oro* tienen usos ilimitados y los clientes *Plata* sólo hasta 5. El sistema ya está funcionando y tenemos que implementar una función particular que devuelva el costo para un cliente de un servicio particular.
Se cuenta con las siguientes funciones y sus valores de retorno:

`cobertura(cliente)`: retorna un string con los valores "Oro" o "Plata", correspondiente al tipo de cobertura del cliente.
`usados(cliente)`: retorna un entero que representa la cantidad de servicios que ya utilizó el cliente.
`radioDeCobertura(cliente, localidad)`: devuelve True si el cliente se encuentra dentro del radio de cobertura cubierto por la empresa.

Al recibir un llamado el operador telefónico solicita el número de cliente, la localidad para la que solicita la asistencia y le informa el costo del servicio teniendo en cuenta que el servicio no tendrá costo para los clientes *Oro* que estén dentro del área de cobertura y para los clientes *Plata* que les quedaran servicios para usar y estuvieran dentro dicha área. Pagarán $50, los clientes *Plata* dentro del área de cobertura pero ya sin servicios gratis. Pagarán $30 extra los clientes que estén fuera del área de cobertura. Observación, los dos últimos montos pueden ser acumulativos. Hacer una función llamada *pagara* que dado un cliente y una localidad devuelva el costo del servicio para el cliente.

Ejercicio 25 ★

Hacer una función que automatice el control vehicular en rutas nacionales. Hacer el control para la Ruta Nacional 8 durante un día completo, se debe controlar que los automóviles no superen 100 km/h y en caso de hacerlo se les enviará una multa a sus hogares, si es reincidente la multa se duplica. Para ello cuenta con las siguientes funciones.
`darPatentes(ruta)`: Dada una ruta nacional devuelve una lista con todas las patentes de los autos que pasaron en el día.
`controlVelocidad(patente)`: Recibe un número de patente y devuelve la velocidad a la que cruzó el radar dicho automóvil.
`reincidente(patente)`: Devuelve True en caso de que la patente ya tenga multas por exceso de velocidad.
`costoActual()`: No recibe parámetros y devuelve el costo por superar la velocidad permitida.
`enviarMulta(patente, costo)`: Manda una multa al domicilio del propietario del automóvil con el costo.