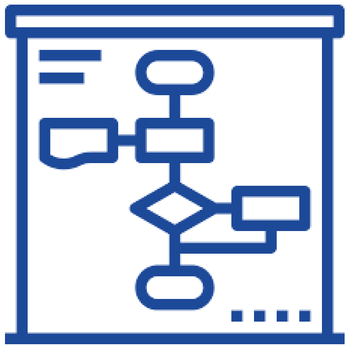
INTRODUCCIÓN A LA PROGRAMACIÓN



OTOÑO, 2021

UNIVERSIDAD TECNOLÓGICA DE CHILE
INSTITUTO PROFESIONAL
CENTRO DE FORMACIÓN TÉCNICA
nacap

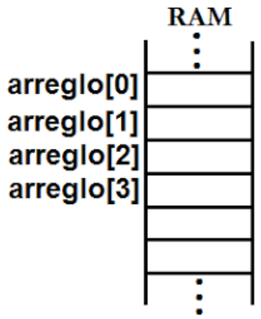
UNIDAD III

Arreglos, tuplas y diccionarios

[arreglos, listas]

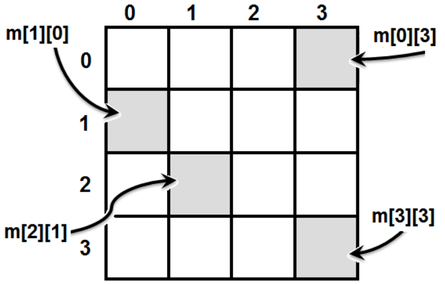
Conceptos Generales

¿Arreglos o Listas?



Python no tiene soporte integrado para arreglos (Arrays), pero se pueden usar Listas (Python Lists) en su lugar; es decir, se pueden trabajar listas como arreglos.

Nota: El módulo que provee las estructuras de datos y las funciones para trabajar con arreglos se llama **numpy**, no viene incluído con Python, por lo que hay que instalarlo por separado (para instalar NumPy, necesitas Python y pip en tu sistema).



Los arreglos tienen algunas similitudes con las listas:

- los elementos tienen un orden y se pueden acceder mediante su posición,
- los elementos se pueden recorrer usando un ciclo for,
- en la sintaxis intervienen los corchetes [ ],
- pueden contener muchos valores con un solo nombre y pueden acceder a los valores haciendo referencia a un número de índice (posición).

Sin embargo, también tienen algunas restricciones:

- todos los elementos del arreglo deben tener el mismo tipo,
- en general, el tamaño del arreglo es fijo (no van creciendo dinámicamente como las listas),
- se ocupan principalmente para almacenar datos numéricos.

¿Por qué necesitamos listas?

- Ante la necesidad de leer, almacenar, procesar y, finalmente, imprimir docenas, quizás cientos, tal vez incluso miles de números o datos (colección de elementos de distinto tipo).

- Nace la siguiente consulta… ¿Se necesita crear una variable separada para cada valor?...

```
var1 = int(input())
var2 = int(input())
var3 = int(input())
var4 = int(input())
var5 = int(input())
var6 = int(input())
…
varN = int(input())
```

Elementos de la lista

- Los elementos de la lista están **ordenados**, se pueden **modificar** y permiten valores **duplicados**.
- Los elementos de la lista están indexados, el primer elemento tiene índice [0], el segundo elemento tiene índice [1], etc.

Ordenado

Cuando decimos que las listas están ordenadas, significa que los artículos tienen un orden definido y ese orden no cambiará. Si agrega nuevos elementos a una lista, los nuevos elementos se colocarán al final de la lista.

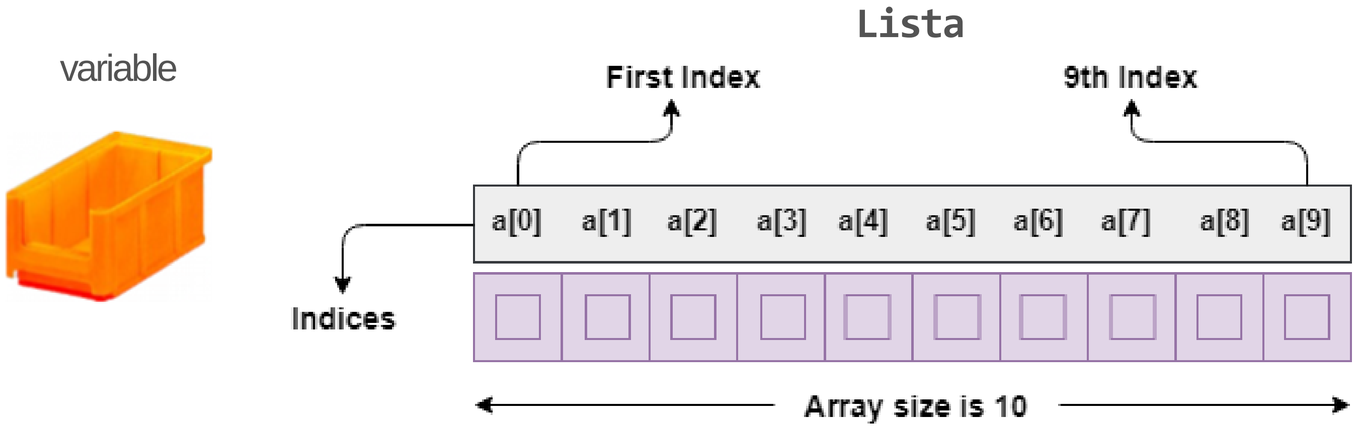
Modificable

La lista se puede cambiar, lo que significa que podemos cambiar, agregar y eliminar elementos en una lista después de que se haya creado.

Permite duplicados

Dado que las listas están indexadas, las listas pueden tener elementos con el mismo valor.

Paralelismo dimensiones físicas




```
#referencia a listas unidimensionales
y bidimensionales
num=[4,7,9,2]
print(num[3])
datos=[["uno", 2, 3],["a", "b", "c"]]
print(datos[1][2])
```

```
#referencia a arreglos unidimensionales
y bidimensionales
from numpy import array
temp=array([5,100,25,34,0,3])
print(temp[0])
print(temp[1])
print(temp[2])
print(temp[3])
print(temp[4])
print(temp[5])
temp1=array([[1,2,3],[4,5,6],[7,8,9]])
print(temp1[0,2)
```

Arreglo de una dimensión (un línea).

Arreglo 2 dimensiones, plano de ejes x e y (2 índices).

Creación de una lista

"list1" es una lista que consta de cinco valores, todos ellos números enteros.

La lista comienza con un corchete abierto y termina con un corchete cerrado.

Index: Posición de los elementos en una lista. Comienzan por el elemento 0, para el caso de list1, los index irían desde el 0 [1] al 4 [45].

```
list1 = [ 1, 7, 87, 9, 45]
```

Desde la perspectiva de Python, las listas se definen como objetos con el tipo de datos 'lista': <class 'list'>.

Elementos de una lista: en este caso list1 posee 5 elementos separados por una coma.

Imprimir una lista: (operador de rebanado)

```
print(list1[:])
print(list1[2:4])
```

El valor dentro de los corchetes que selecciona un elemento de la lista se llama índice y también permite acceder a un subconjunto de valores de la lista (rebanado).

Creación de una lista

Para crear una lista también puede utilizar la función **`list()`**, la cual es posible aplicarla sobre un iterable.

```
a=list("hola")
b=list(range(10))
print(a)
print(b)
```

```
['h', 'o', 'l', 'a']
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Operaciones sobre listas

Longitud de lista

Para determinar cuántos elementos tiene una lista, use la función **`len()`**:

```
colores=['azul', 'rojo', 'verde', 'amarillo']
print(len(colores))
```

Operaciones

`sum(x)` entrega la suma de los valores de la lista:

```
sum([1,2,1,-1,-2]) #resultado 1
sum([]) #resultado 0
```

Nota: se pueden utilizar también las operaciones + (para concatenar listas) y * (para repetir n veces).

```
list('hola') + [2, 3, 4]
[3.14, 6.28, 9.42] * 2
```

Acceder/Modificar valor i-enésimo

Es posible modificar el valor i-enésimo elemento, accediendo con el índice correspondiente:

```
colores=['azul', 'rojo', 'verde', 'amarillo']
colores[1]='negro'
#resultado =['azul', 'negro', 'verde', 'amarillo']
```

Nota: para acceder al último elemento de la lista, tenemos que usar **`-1`** como índice de la lista:

```
print(colores[-1])
```

Encontrar un elemento x

Para saber si un elemento x está en la lista L, se usa **`x in L`**.

```
r=list(range(0,20,2))
print(12 in r)
#resultado True
```

Operaciones sobre listas

l.count(x)

Cuenta las veces que el elemento x está en la lista:

```
letras = list('paralelepipedo')
print(letras.count('p’))
#resultado 3
```

l.append(x)

Agrega un elemento al final de la lista:

```
primos=[2, 3, 5, 7, 11]
primos.append(13)
primos.append(17)
print(primos)
#resultado [2, 3, 5, 7, 11, 13, 17]
```

l.remove(x)

Elimina un elemento x de lista por su valor:

```
l = [7, 0, 3, 9, 8, 2, 4]
l.remove(2)
print(l)
#resultado [7, 0, 3, 9, 8, 4]
```

l.insert(posición, nuevo elemento)

Agrega un elemento en una posición determinada:

```
semana=["lunes","miércoles","jueves","viernes”]
semana.insert(1,"martes")
print(semana)
#resultado ['lunes', 'martes', 'miércoles', 'jueves', 'viernes’]
```

l.pop(índice)

Elimina el elemento de la lista por su índice.

Nota: la instrucción pop() elimina el último elemento de la lista.

```
l = [7, 0, 3, 9, 8, 2, 4]
l.pop(4)
print(l)
#resultado [7, 0, 3, 9, 2, 4]
```

```
#elimina el i-enésimo elemento de la lista:
l = [7, 0, 3, 9, 8, 2, 4]
del l[4]
```

Operaciones sobre listas

l.reverse()

Invierte una lista:

```
l = [7, 0, 3, 9, 8, 2, 4]
l.reverse()
print(l)
#resultado [4, 2, 8, 9, 3, 0, 7]
```

l.sort()

Ordena la lista:

```
l = [7, 0, 3, 9, 8, 2, 4]
l.sort()
print(l)
#resultado [0, 2, 3, 4, 7, 8, 9]
```

Agregar elementos

Se puede iniciar la vida de una lista creándola vacía (esto se hace con un par de corchetes vacíos) y luego agregar nuevos elementos según sea necesario.

```
miLista=[] #creando una lista vacía
for i in range(0,6):
    miLista.insert(i,i)
print(miLista)
#resultado [0, 1, 2, 3, 4, 5]
```

Iteración sobre las listas

Una lista es un objeto **iterable**. Esto significa que sus valores se pueden recorrer usando un ciclo for:

```
valores = [6, 1, 7, 8, 9]
for i in valores:
    print(i**2,end=“ “)
#resultado 36 1 49 64 81 3
```

Ejercicios

1. Crear una lista (arreglo de datos) de n posiciones, llénela con nombres de personas y como salida debe mostrar los nombres almacenados y la posición correspondiente. Utilice ciclos para los procesos anteriores.
2. Llenar un lista de 10 posiciones con números aleatorios entre 1 y 100, como salida debe mostrar los números aleatorios del arreglo ordenados de menor a mayor.
3. Almacene nombres en una lista de n posiciones. Una vez completada la lista, implementar una opción que al ingresar una posición muestre el dato que contiene.
4. Dado una lista de 100 elementos numéricos enteros (1 al 100), generar el código que muestre todos los números de la lista y la suma de ellos.
5. Desarrollar un algoritmo que a partir de una lista lst_1=[1,2,3,4,5] genere una segunda lista lst_2 con los valores lst_2=[1,3,6,10,15].
6. Desarrollar un algoritmo que almacene 5 valores enteros aleatorios en una lista e imprima: a. El valor mayor; b. La suma de los números.; c. El promedio de los números.

Operaciones sobre Listas:

a. *Crear una lista de nombre list1 e Imprimir su longitud.*
b. *Eliminar el penúltimo elemento de la lista list1, verificar longitud de la nueva lista resultante.*
c. *Eliminar todos los elementos de la lista list1, (no eliminar lista), imprimir la longitud resultante y los elementos de la lista.*
d. *Crear la siguiente lista:*
   semana = ["lunes","miércoles,"jueves","viernes"], utilizando los métodos append y/o insert, según corresponda, agregar los días de la semana faltante. Imprimir la lista resultante además de su longitud.

inacap.cl