Fundamentos de Programación

ARREGLOS DE DATOS 1
Semana 11

OBJETIVO DEL LABORATORIO

Construye arreglos unidimensionales para la manipulación de datos para casos de negocio.

MARCO TEÓRICO

Hay ocasiones en donde necesitamos trabajar con conjuntos de datos y no con un dato individual, para ello necesitamos apoyarnos en los arreglos o listas como se conoce en Python.

Como fuente de información, se tiene la misma plataforma del Netacad. Tanto para realizar los ejemplos propuestos en la interfaz integrada, así como para repasar lo visto en clase.

RECURSOS

a. Hardware

- Pc Pentium IV a superior
- Conexión de red

b. Software

- Sistema Operativo Windows XP a superior
- Navegador Chrome o Firefox
- Edube Sandbox de Python desde Netacad

PROCEDIMIENTO

1. Listas unidimensionales:

Una lista es un conjunto ordenado de objetos. Por objetos entendemos cualquiera de los tipos de dato ya mencionados, incluso otras listas.

Para crear una lista, especificamos sus elementos entre corchetes y separados por comas.

```
>>> lenguajes = ["Python", "Java", "C", "C++"]
```

Dado que se trata de una colección ordenada, accedemos a cada uno de los elementos a partir de su posición, comenzando desde el 0.

```
>>> lenguajes[0]
'Python'
>>> lenguajes[1]
'Java'
```

(En Python, las cadenas de caracteres pueden estar formadas tanto por comillas dobles como por comillas simples).

La posición de un elemento, también conocida como índice, aumenta de izquierda a derecha cuando éste es positivo. Indicando un número negativo, los elementos se obtienen de derecha a izquierda.

```
>>> lenguajes[-1]
'C++'
```

```
>>> lenguajes[-2]
'C'
```

Para cambiar un elemento, combinamos la nomenclatura descrita con la asignación.

```
>>> lenguajes[1] = "Rust"
>>> lenguajes
['Python', 'Rust', 'C', 'C++']
```

Para remover un elemento, utilizamos la palabra reservada `del`.

```
>>> del lenguajes[1]
>>> lenguajes
['Python', 'C', 'C++']
```

Nótese que cuando un elemento es eliminado, a excepción del último, todos los que lo suceden se corren una posición hacia la izquierda.

La operación inversa es la de insertar un elemento en una posición determinada, desplazando el resto que le sucede una posición hacia la derecha, vía el método (volveremos sobre este concepto más adelante) `insert()`.

```
# Insertar la cadena "Elixir" en la posición 1.
>>> lenguajes.insert(1, "Elixir")
>>> lenguajes
['Python', 'Elixir', 'C', 'C++']
```

Por último, la operación más común es la de agregar un elemento al final de la lista. Para ello empleamos `append()`.

```
>>> lenguajes.append("Haskell")
>>> lenguajes
['Python', 'Elixir', 'C', 'C++', 'Haskell']
```

¡Genial! Baste por el momento con lo que se ha dicho sobre las listas.

CONCLUSIONES Y RECOMENDACIONES DE LA EXPERIENCIA
Al momento de construir listas, podemos trabajar con grupos de elementos que tienen la misma naturaleza; podemos realizar operaciones con los mismos, usando los temas vistos anteriormente.
Se recomienda usar el Edube SandBox para realizar nuestros ejercicios sin necesidad de instalar algún software adicional.

Así como en algún momento se solicita cálculos con números enteros, sería factible buscar realizar lo mismo; pero con números reales.

ACTIVIDAD VIRTUAL

1. Observa y analiza el siguiente video: https://www.youtube.com/watch?v=d9mcVq7r1YI, y responde las siguientes preguntas:

- ¿Cuáles son las listas iniciales que aparecen en el video?
- ¿Cuál es el valor y posición del elemento modificado en el video?
- Realiza ese ejemplo en Python, considerando como lista 1 las edades de ti y 6 compañeros más. Como lista 2 sus nombres. Modifica un par de nombres e imprime la longitud de cada lista.