Fundamentos de Programación

TEMA: ARREGLO DE DATOS 1 Y 2 Y MANEJO DE FUNCIONES 1 Y 2
Semana 15

OBJETIVO DEL LABORATORIO

Construye arreglos unidimensionales, bidimensionales para la manipulación de datos para casos de negocio.
Realiza ordenamiento de los elementos en un arreglo.
Crea procedimientos o funciones para casos de negocio, con parámetros y con retorno.

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.
Así como podemos usar las listas unidimensionales, podemos usarlas como matriz; ósea bidimensionales. Nos permite con ello trabajar con tablas.
A su vez, los elementos de una lista; los podemos ordenar en base a algún criterio que tengamos.
A través de los diversos temas vistos en PYTHON, hemos usado funciones predeterminadas; sin embargo, nosotros podemos crear la función que queramos en base a un escenario en particular.
Crear funciones implica que podamos tenerlas con o sin parámetros, así como con retorno.
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

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']
```

2. Funciones con argumentos

En Python los argumentos de las funciones tienen un comportamiento particular a diferencia de otros lenguajes. Tomemos como ejemplo la siguiente función.

```
def sumar(a, b):
    return a + b
```

De ella decimos que tiene dos argumentos posicionales. Llevan ese nombre por cuanto al invocar `sumar(7, 5)`, `7` se corresponde con `a` por ser el primer argumento, y `5` se corresponde con `b` por ser el segundo. Si lo invertimos, de modo que llamemos a `sumar(5, 7)`, entonces ahora `a == 5` y `b == 7`.

Además, al invocar a una función con argumentos posicionales, estos siempren deben tener un valor. Por ejemplo, las siguientes llamadas arrojan un error.

```
# Falta especificar el argumento b.
sumar(7)
# Sobra un argumento.
sumar(7, 5, 3)
```

Ahora bien, una función puede tener argumentos con valores por defecto, de modo que, al llamarla, si no hemos especificado un valor concreto por argumento, éste toma automáticamente el que la definición le ha asignado por defecto.

```
def sumar(a, b=5):
    return a + b


# ¡Perfecto! Imprime 12.
print(sumar(7))
# En este caso, b == 10.
print(sumar(5, 10))
```

Agreguemos, para avanzar un poco más, un tercer argumento `c` con un valor por defecto.

```
def sumar(a, b=5, c=10):
    return a + b + c


# Imprime 22 (7 + 5 + 10).
print(sumar(7))
```

¿Qué ocurre si quiero indicar un valor para el argumento `c` mientras que `b` mantenga su valor por defecto? En ese caso, en lugar de pasar el argumento por posición, lo voy a pasar por nombre.

```
# Imprime 32 (7 + 5 + 20).
print(sumar(7, c=20))
```

La posibilidad de pasar argumentos por su nombre en la llamada a una función solo puede darse en aquellos que tengan un valor por defecto. De ahí que a estos se los conozca como argumentos por nombre. En estos casos, la posición de los argumentos es indistinta, de modo que el siguiente código es perfectamente válido.

```
# El orden es indistinto en los argumentos por nombre.
print(sumar(7, c=20, b=10))
```

(Nótese que, por convención, cuando especificamos el valor de un argumento por nombre no se ubican espacios alrededor del signo igual).

La única restricción es que los argumentos posicionales deben preceder a los argumentos por nombre, tanto en la definición de una función como en la llamada. El siguiente ejemplo no es un código válido de Python.

```
# ¡¡¡Inválido!!!
def sumar(a=5, b):
    return a + b
```

Python provee otras herramientas interesantes para trabajar con los argumentos. Por ejemplo, la posibilidad de crear funciones con argumentos infinitos.

CONCLUSIONES Y RECOMENDACIONES DE LA EXPERIENCIA

- Al construir nuestras funciones, hace posible que podamos trabajarlas en función de algunos parámetros, para su posterior procesamiento.
- 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=fys83EGuayY, y responde las siguientes preguntas:

- ¿Cuáles son las listas que aparecen en el video?
- Realiza esos ejemplos en Python

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

- ¿Cuáles son las funciones que aparecen en el video?
- Realiza esos ejemplos en Python