**UNIVERSIDAD DE GUADALAJARA**

**CUCEI**

**DIVISIÓN DE ELECTRÓNICA Y COMPUTACIÓN**

**DEPARTAMENTO DE CIENCIAS COMPUTACIONALES**

**TAREA No. 4**

*TEMA***: LISTAS**

***EQUIPO No. 4***

*INTEGRANTE*: **ROBLES PULIDO EFRAIN**

*NOMBRE DE LA MATERIA:* **FUNDAMENTOS FILOSOFICOS DE LA COMPUTACION**

*SECCIÓN:* **D13**  *CALENDARIO:* **2021A**

*NOMBRE DEL PROFESOR:* **MARISCAL LUGO LUIS FELIPE**





**Marco Teórico**

**8.1 Secuencias**

Una ***secuencia*** es un objeto que contiene varios elementos de datos, almacenados uno tras otro. Puede realizar operaciones en una secuencia para examinar y manipular los elementos almacenados en ella.

Python proporciona varias formas de realizar operaciones en los elementos que se almacenan en una secuencia. Hay varios tipos diferentes de objetos de secuencia en Python. Como las listas y tuplas.

Tanto las listas como las tuplas son secuencias que pueden contener varios tipos de datos. La diferencia entre listas y tuplas es simple: una lista es mutable, lo que significa que un programa puede cambiar su contenido, pero una tupla es inmutable, lo que significa que, una vez creado, su contenido no se puede modificar.

**8.2 Introducción a las listas**

Una **lista** es un objeto que contiene varios elementos de datos. Las listas son mutables, lo que significa que su contenido se puede cambiar durante la ejecución de un programa, se pueden agregar o eliminar elementos de ellas. Puede utilizar indexación, segmentación y varios métodos para trabajar con listas en un programa.

Números\_pares = [2, 4, 6, 8, 10]

Los elementos que están entre corchetes y separados por comas son los elementos de la lista. Después de que se ejecute esta declaración, la variable *números\_pares* hará referencia a la lista.



números\_pares

Podemos almacenar una gran variedad de valores sin importar el tipo de datos en las listas como nombres y números a la vez.

También en Python también tiene una función list () incorporada que puede convertir ciertos tipos de objetos en listas. Como la función de rango devuelve un iterable del ciclo for, que es un objeto que contiene una serie de valores sobre los que se puede iterar. Entonces se puede usar una declaración como la siguiente para convertir el objeto iterable de la función de rango en una lista:



Cuando se ejecuta esta declaración, suceden las siguientes cosas:

• Se llama a la función de rango con 5 pasado como argumento. La función devuelve un iterable que contiene los valores 0, 1, 2, 3, 4.

• El iterable se pasa como argumento a la función list (). La función list () devuelve la lista [0, 1, 2, 3, 4].

• La lista [0, 1, 2, 3, 4] se asigna a la variable de números.

Otro ejemplo:



Esta declaración se asignará la lista [1, 3, 5, 7, 9] a la variable de números.

**Operador de repetición**

Cuando el operador en el lado izquierdo del símbolo \* es una secuencia (como una lista) y el operando en el lado derecho es un número entero, se convierte en el *operador de repetición.* El operador de repetición hace varias copias de una lista y las une a todas. Aquí está el formato general:

*list \* n*

Por ejemplo:

Texto

Descripción generada automáticamente con confianza media

* En la línea 1, la expresión [0] \* 5 hace cinco copias de la lista [0] y las une a todas en una sola lista. La lista resultante se asigna a la variable de números.
* En la línea 2, la variable de números se pasa a la función de impresión. La salida de la función se muestra en la línea 3.

**Iterando sobre una lista con el bucle for**

Existen técnicas para acceder a los caracteres individuales de una cadena. Muchas de las mismas técnicas de programación también se aplican a las listas. Por ejemplo, puede iterar sobre una lista con el bucle for, como se muestra a continuación:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Este código mostrará lo siguiente:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

**Indexación**

Otra forma de acceder a los elementos individuales de una lista es con un índice. Cada elemento de una lista tiene un índice que especifica su posición en la lista. La indexación comienza en 0, por lo que el índice del primer elemento es 0, el índice del segundo elemento es 1, y así sucesivamente. El índice del último elemento de una lista es 1 menos que el número de elementos de la lista.

También se puede utilizar índices negativos con listas para identificar las posiciones de los elementos en relación con el final de la lista. El intérprete de Python agrega índices negativos a la longitud de la lista para determinar la posición del elemento. El índice -1 identifica el último elemento de una lista, -2 identifica el penúltimo elemento, y así sucesivamente.

Se generará una excepción IndexError si usa un índice no válido con una lista. Por ejemplo, mira el siguiente código:



Texto

Descripción generada automáticamente

La última vez que este bucle itera, a la variable de índice se le asignará el valor 5, que es un índice no válido para la lista. Como resultado, la declaración que llama a la función de impresión provocará que se genere una excepción IndexError.

**Función len**

Python tiene una función incorporada llamada len que devuelve la longitud de una secuencia, como una lista. El siguiente código demuestra:



La primera instrucción asigna la lista [10, 20, 30, 40] a la variable my\_list. La segunda instrucción llama a la función len, pasando la variable my\_list como argumento. La función devuelve el valor 4, que es el número de elementos de la lista. Este valor se asigna a la variable de tamaño. La función len se puede utilizar para evitar una excepción IndexError cuando se itera sobre una lista con un bucle.

**Las listas son mutables**

Las listas en Python son mutables, lo que significa que sus elementos se pueden cambiar. En consecuencia, una expresión en la lista de formularios [índice] puede aparecer en el lado izquierdo de un operador de asignación. El siguiente código muestra un ejemplo:

Texto

Descripción generada automáticamente

La declaración en la línea 2 mostrará: 

El enunciado de la línea 3 asigna 99 a los números [0]. Esto cambia el primer valor de la lista a 99. Cuando se ejecute la instrucción en la línea 4, mostrará: 

Cuando usa una expresión de indexación para asignar un valor a un elemento de la lista, se debe de usar un índice válido para un elemento existente o se producirá una excepción IndexError.

**Listas de concatenación**

Concatenar significa unir dos cosas. Puede utilizar el operador + para concatenar dos listas. Aquí hay un ejemplo:

Pantalla con letras y números

Descripción generada automáticamente con confianza media



También puede usar el operador de asignación aumentado + = para concatenar una lista a otra.

Un conjunto de letras negras en un fondo blanco

Descripción generada automáticamente con confianza media

La última declaración agrega list2 a list1. Después de que se ejecuta este código, list2 permanece sin cambios, pero list1 hace referencia a la siguiente lista:



**8.3 Rebanar lista**

Una expresión de corte selecciona un rango de elementos de una secuencia. En Python, puede escribir expresiones que seleccionen subsecciones de una secuencia, conocidas como rebanadas.

Un segmento es un grupo de elementos que se toman de una secuencia. Cuando toma un segmento de una lista, obtiene un grupo de elementos dentro de la lista. Para obtener una porción de una lista, escribe una expresión en el siguiente formato general:



En el formato general, inicio es el índice del primer elemento del sector y final es el índice que marca el final del sector. La expresión devuelve una lista que contiene una copia de los elementos desde el inicio hasta el final (pero sin incluirlos). Por ejemplo, supongamos que creamos la siguiente lista:



La siguiente declaración usa una expresión de corte para obtener los elementos de los índices 2 hasta, pero sin incluir, 5:

  
Después de que se ejecuta esta declaración, la variable mid\_days hace referencia a la siguiente lista:



Si omite el índice de inicio en una expresión de corte, Python usa 0 como índice de inicio. La siguiente sesión de modo interactivo muestra un ejemplo:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

La línea 4 envía los números de segmento [: 3] como argumento a la función de impresión. Debido a que se omitió el índice inicial, el segmento contiene los elementos desde el índice 0 hasta el 3.

Si se omite el índice final en una expresión de corte, Python usa la longitud de la lista como índice final. La siguiente sesión de modo interactivo muestra un ejemplo:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En la línea 4 se envía los números de segmento [2:] como argumento a la función de impresión. Debido a que se omitió el índice final, el segmento contiene los elementos desde el índice 2 hasta el final de la lista.

Y si se omite tanto el índice inicial como el final en una expresión de corte, obtendrá una copia de la lista completa. La siguiente sesión de modo interactivo muestra un ejemplo:

Texto

Descripción generada automáticamente

Los ejemplos anteriores de corte se obtienen cortes de elementos consecutivos de listas. Las expresiones de corte también pueden tener un valor de paso, lo que puede hacer que se omitan elementos en la lista. La siguiente sesión de modo interactivo muestra un ejemplo de una expresión de corte con un valor de paso:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

En la expresión de corte en la línea 4, el tercer número dentro de los corchetes es el valor del paso. Un valor de paso de 2, como se usa en este ejemplo, hace que el segmento contenga cada segundo elemento del rango especificado en la lista.

También puede utilizar números negativos como índices en expresiones de corte para hacer referencia a posiciones relativas al final de la lista. Python agrega un índice negativo a la longitud de una lista para obtener la posición a la que hace referencia ese índice. La siguiente sesión de modo interactivo muestra un ejemplo:

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

**8.4 Búsqueda de elementos en listas con el operador in**

Puedes buscar un elemento en una lista utilizando el operador in. En Python, puede usar el operador in para determinar si un elemento está contenido en una lista.

Aquí está el formato general de una expresión escrita con el operador in para buscar un elemento en una lista:



La expresión devuelve verdadero si el elemento se encuentra en la lista o falso en caso contrario.

También se puede utilizar el operador not in para determinar si un elemento no está en una lista. Como:

Texto

Descripción generada automáticamente

**8.5 Métodos de lista y funciones integradas útiles**

Las listas tienen numerosos métodos que le permiten trabajar con los elementos que contienen. Python también proporciona algunas funciones integradas que son útiles para trabajar con listas.

Las listas tienen numerosos métodos que le permiten agregar elementos, eliminar elementos, cambiar el orden de los elementos, etc. Como lo muestra la siguiente tabla:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* **El método de adición**

El método de agregar se usa comúnmente para agregar elementos a una lista. El elemento que se pasa como argumento se agrega al final de los elementos existentes de la lista. Como lo muestra el siguiente código:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Dentro del ciclo, se llama al método append para construir la lista. La primera vez que se llama al método, el argumento que se le pasa se convierte en el elemento 0. La segunda vez que se llama al método, el argumento que se le pasa se convierte en el elemento 1. Esto continúa hasta que el usuario sale del bucle.

* **El método de índice**

Anteriormente, vio cómo se puede usar el operador *in* para determinar si un elemento está en una lista. A veces es necesario saber no solo si un elemento está en una lista, sino también dónde se encuentra.

El método de índice es útil en estos casos. Pasas un argumento al método de índice y devuelve el índice del primer elemento de la lista que contiene ese elemento. Si el elemento no se encuentra en la lista, el método genera una excepción ValueError. Por ejemplo:

Texto, Carta

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Los elementos de la lista de alimentos se muestran en la línea 11, y en la línea 14 se pregunta al usuario qué artículo desea cambiar. La línea 18 llama al método de índice para obtener el índice del artículo. La línea 21 obtiene el nuevo valor del usuario y la línea 24 asigna el nuevo valor al elemento que contiene el valor anterior.

* **El método de inserción**

El método de inserción le permite insertar un elemento en una lista en una posición específica. Pasas dos argumentos al método de inserción: un índice que especifica dónde se debe insertar el elemento y el elemento que deseas insertar. Como se muestra a continuación:



* **El método de clasificación**

El método de clasificación reorganiza los elementos de una lista para que aparezcan en orden ascendente (del valor más bajo al valor más alto). Aquí hay un ejemplo:

Texto

Descripción generada automáticamente con confianza media

Cuando este código se ejecute, mostrará lo siguiente:



* **El método de eliminación**

El método de eliminación elimina un elemento de la lista. Pasas un elemento al método como argumento y el primer elemento que contiene ese elemento se elimina. Esto reduce el tamaño de la lista en un elemento. Todos los elementos posteriores al elemento eliminado se desplazan una posición hacia el principio de la lista. Se genera una excepción ValueError si el elemento no se encuentra en la lista. Por ejemplo:Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamenteInterfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

* **El método inverso**

El método inverso simplemente invierte el orden de los elementos de la lista. Aquí hay un ejemplo:

Texto, Carta

Descripción generada automáticamente

Este código mostrará lo siguiente:



**La declaración *del***

El método de *eliminación* que vio anteriormente elimina un elemento específico de una lista, si ese elemento está en la lista. Algunas situaciones pueden requerir que elimine un elemento de un índice específico, independientemente del elemento que esté almacenado en ese índice. Esto se puede lograr con la declaración *del*. Por ejemplo:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Este código mostrará lo siguiente:



**Las funciones *min* y *max***

La función *min* acepta una secuencia, como una lista, como argumento y devuelve el elemento que tiene el valor más bajo en la secuencia. Aquí hay un ejemplo:



Este código mostrará lo siguiente:



La función *max* acepta una secuencia, como una lista, como argumento y devuelve el elemento que tiene el valor más alto en la secuencia. Aquí hay un ejemplo:



Este código mostrará lo siguiente:



# **Bibliografía**

Gaddis, T. (2021). *Starting out with Python* (tercera ed.). Pearson. Recuperado el 13 de Marzo de 2021, de https://fdocuments.in/document/starting-out-with-python-3rd-edition-pdf-firebase-python-programming-2nd-edition.html

**5.- Conclusiones personales**

Gracias a las secuencias podemos tener nuestros datos en forma de lista para hacer múltiples acciones que nos permita utilizar nuestra información de forma organizada para nuestro propósito deseado. Se puede hacer diversas acciones como agregar, eliminar, acomodar o buscar dentro de la lista en donde tendremos datos sea un numero o un carácter para así después utilizar esa información. Si lo utilizamos con los temas anteriores podremos tener programas para ir organizando datos en forma de lista y poder hacer las modificaciones necesarias para tener actualizaciones de la lista.