	Carátula para entrega de prácticas	
Facultad de Ingeniería		Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Edgar Tista Garcia

Asignatura: Estructuras de Datos y Algoritmos I

Grupo: 2

No. de práctica(s): 9

Integrante(s): Álvarez López Carlos Manuel

No. de lista o brigada: 2

Semestre: 2023-2

Fecha de entrega: Jueves 11 de Mayo del 2023

Observaciones:

CALIFICACIÓN: _____

Objetivo:

Aplicar las bases del lenguaje de programación Python en el ambiente de Jupyter notebook.

Actividad 1

1. Variables

A diferencia de C, en Python se le puede asignar un mismo valor a varias v variables a la vez de la siguiente manera:

x y z = valor

A su vez, no es necesario especificar el tipo de dato que guarda la variable, además de que este puede cambiar a lo largo de la ejecución. También se notó que la función print(), agrega automáticamente un salto de línea, y para concatenar, se separa lo que se quiera imprimir con “,”.

2. Cadenas

Las cadenas pueden ser definidas usando tanto comillas simples (‘’) como con comillas dobles (“”). Cuando se está redefiniendo una variable, al igual que con C se puede usar el operador “+” para concatenar.

Una forma de imprimir variables con print() también se puede hacer con .format
Para esto a la variable que guarda la cadena a imprimir se le ponen corchetes donde vaya a ir otra cadena, para al final añadir “.format(cadena, cadena2) introduciendo las cadenas que se quieran imprimir respecto a los corchetes en la sección anterior. En los corchetes también pueden introducir el número de cadena del formato que se desea que esté en ese espacio.

3. Operadores

La principal diferencia que se encontró con los operadores con respecto a C, es que se añade “**” doble asterisco para exponentes, y doble diagonal “//” para división entera.

Por parte de las cadenas se pueden comparar cadenas con el operador “==”.

4. Listas

Las listas son varios elementos no necesariamente del mismo tipo, dentro de paréntesis cuadrados separados por comas “,”, son modificables y se puede acceder a sus elementos mediante índices, empezando por 0, hasta el número de elementos menos 1.
También se tiene una función para imprimir una lista ya implementada en el lenguaje.

5. Tupla

A diferencia de la lista, la tupla se declara con paréntesis “()” y esta no es mutable, por lo que sus elementos no pueden ser modificados. Aunque al igual que las listas son conjuntos de elementos no necesariamente del mismo tipo, separados por comas, pero con la ventaja de ocupar menos memoria para almacenarse que una lista. Al igual que con la lista se puede acceder a sus elementos mediante índices empezando en 0 hasta el número de elementos menos 1, y siguiendo el orden en que fueron definidos. También se tiene una función para imprimir una lista ya implementada en el lenguaje.

6. Tupla con nombre

Para trabajar con este elemento se debe de importar una biblioteca con la siguiente línea: “from collections import namedtuple”.

Cuenta con las características de una tupla, pero con la posibilidad que al definirla se le pueda introducir como primer propiedad un nombre para describirla. Este nombre debe de ser una cadena, y seguido de este separado por una coma se debe definir la tupla en cuestión.

nombreVariable = namedtuple(‘Nombre’,[elementos, elementos])

El lenguaje también tiene implementada su impresión en pantalla con la función print(), con la cual se pueden visualizar tanto el nombre de la tupla, como los elementos de la tupla. También se puede acceder a los elementos de la tupla con la función format(nombreDeLaTupla._fields).

7. Diccionarios

Se define encerrando sus elementos entre corchetes “{}” siendo que sus elementos se introducen primero la llave seguido de “:” y el valor asociado a esa llave, separando los distintos elementos con comas (,).

Cada elemento de este conjunto está formado de dos partes, una llave y su valor. En el caso de la llave esta es inmutable y del mismo tipo para todas las incluidas en el diccionario, además son irrepetibles por consiguiente no hay elementos repetidos en un diccionario. Es la forma en la cual acceder a un valor, ya que este conjunto no está ordenado. Por parte del valor estos su valor pueden ser distintos entre sí e ir cambiando con el tiempo, el valor de una llave puede ser otro diccionario.

Para acceder al valor se tiene que ocupar la llave del respectivo elemento como si fuera el índice de una lista o tupla. A su vez se pueden añadir elementos al diccionario con la función “diccionario.update({“llave”: valor})”.

Así como con las listas y las tuplas, el lenguaje trae implementado la impresión en pantalla de tanto todos los elementos de un diccionario, como de únicamente sus llaves, mediante la función print(), y las funciones diccionario.items() y diccionario.keys() para acceder a los elementos y a las llaves del diccionario respectivamente.

8. Funciones

Para declarar funciones se ocupa la palabra reservada “def” seguido del nombre de la variable y el nombre de los parámetros entre paréntesis. Las funciones en Python trabajan de manera similar que en C, con la diferencia que no hay que especificar el tipo de dato de los parámetros, y que en Python las funciones pueden retornar más de un valor, siendo que se pueden guardar cada uno de estos valores retornados se pueden guardar en varias variables de la siguiente manera: “val1,val2,val3 = func()”, siendo que func es una función que retorna 3 valores. En esta sintaxis se puede reemplazar una de las variables en donde se guardan datos por “_” para no guardar el dato que iría a esa variable.

También se pueden introducir parámetros con valores por defecto que se ocuparan en caso de no recibir un valor de parámetro al llamar a la función. Para esto al declarar la variable se define con un parámetro como se hace usualmente, pero a este se le asigna un valor con el operador “=”.

8.1 Variables globales

Para declarar variables globales declara la variable fuera de cualquier función, para que de este modo la variable pueda ser accesible desde cualquier ámbito(scope).

Los ámbitos en Python trabajan de manera similar que en C, siendo que dentro de una función se trabaja en un espacio local en el cual se pueden acceder a las variables globales y como si se tratare de paso por paso por valor, se puede trabajar dentro de la función con la variable global sin modificarla fuera del ámbito de la función; esto siempre y cuando no se cree una variable con el mismo nombre dentro

de la función, ya que esta reemplazaría a la variable global dentro de la función, y no se podría utilizar hasta su definición. Para esto también existe la posibilidad de ocupar la palabra reservada “global” para especificar que se está tratando de modificar la variable global siendo una especie de paso por referencia como en C.

- **Preguntas**

1. ¿Cuál es la diferencia entre una lista una tupla y un diccionario, en Python?

A pesar de que todas se podrían definir como conjuntos de elementos, difieren adicional a su forma de declaración, en que en el caso de la la lista es mutable por lo que sus elementos son dinámicos, a diferencia de la tupla, la cual no admite la modificación a sus elementos, aunque contando con la ventaja de ser más eficiente en cuestión de memoria que la lista; por otro lado el diccionario en sus elementos tiene una parte que es mutable y otra que no, ya que la llave no se puede modificar, como si lo puede hacer su valor.

Adicionalmente tanto como la lista así como la tupla siguen un orden, respetando el que se tiene al definirlas, aunado a permitir elementos duplicados; además de poder acceder a sus elementos mediante índices empezando por 0 hasta el número de elementos menos uno, o inclusive introducir números negativos para recorrer el conjunto de manera inversa. Por otro lado, el diccionario no tiene un orden, y la forma de acceder a sus elementos es mediante sus llaves, las cuales no se pueden repetir, por lo que no se puede tener elementos duplicados.

2. ¿Para qué sirve la estructura if-elif-else? Escribe un código de ejemplo sencillo

Sirve para evaluar condiciones lógicas y declaraciones booleanas, y realizar acciones con base en si resulta verdadero o falso. Si la condición evaluada en if es verdadera se ejecuta el código en la siguiente sangría; elif evalúa otra condición en caso de que la evaluada en if resulte falso, ejecutando las sentencias en su siguiente sangría; y finalmente else ejecuta otras líneas de código si no se cumplen ninguna de las condiciones anteriores.

En el código ejemplo se declara una variable “x” y se le asigna el valor de un entero. En seguida se encuentra una estructura if-elif-else, donde la condición de if evalúa si x es mayor que 20 para imprimir un mensaje en caso de verdadero; que en caso de que sea falso, elif evalúa si el número es menor que 10, imprimiendo un mensaje si llega a ser verdadero, pero si llega a ser falso se llega a else indicando que las condiciones anteriores fueron falsas e imprimiendo un mensaje indicando que el valor se encuentra entre 10 y 20

```python
x = 15

if x > 20:
    print("x es mayor de 20")
elif x < 10:
    print("x es menor que 10")
else:
    print("x esta entre 20 y 10")
```

```
x esta entre 20 y 10
```

3. Explica cómo funciona en Python el recorrido sobre listas “for _ in range()”

La estructura for ofrece una manera de recorrer los elementos de una lista, siendo “_” el espacio para una variable que representa el elemento de la lista en la que se está iterando. La función “range()” crea una lista, siendo que el parámetro introducido deberá ser un entero y será él último elemento de la lista, está empezara desde 0 e irá incrementando hasta llegar al límite impuesto en el parámetro; también se pueden introducir 2 parámetros, siendo el primero el límite inferior y el segundo el límite superior.

La estructura “for _ in range()” se puede ver también como un ciclo for de C, ya que el parámetro de la función range() serán las veces que se iterara, incrementando la variable en “_” con paso 1.

4. Crea una lista que contenga los siguientes elementos: México, China, Japón, Rusia, España, Argentina.

```
Paises en la lista:
Mexico
China
Rusia
España
Argentina
Colombia
```

Para añadir al final de la lista se usó el método append(), teniendo como parámetro la cadena “Colombia”. Para remover “Japon”, se ocupó el método remove(), el cual elimina el elemento cuyo valor coincida con el parámetro introducido, así que como parámetro se introdujo la cadena “Japon”.

4.3 Investiga al menos otras 5 funciones que se pueden aplicar a las listas, y anota que hace cada una

clear() - Elimina todos los elementos de una lista.

copy() - Retorna una copia de la lista.

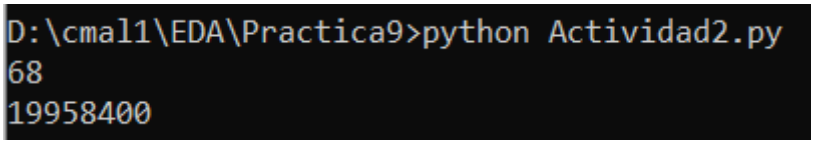
index() - Retorna el índice del primer elemento de la lista cuyo valor sea igual que aquel introducido como parámetro.

extend() - Añade elementos al final de la lista desde otro iterable, el cual se coloca como parámetro.

pop() - Elimina un elemento de la lista de una posición especificada, esta posición se introduce como parámetro, por lo que este debe ser un entero.

Actividad 2

a) Salidas del programa



Las salidas fueron en una línea 68, y en otra 19958400. Ambas salidas son de tipo entero.

b) Explica que hace el programa

En este programa se crea la función: funcion(); la cual inmediatamente después de su declaración y definición, se ejecuta, finalizando el programa.

La función crea una lista con valores enteros de manera creciente desde el 5 hasta el 12, incluyendolos. También crea dos funciones, valor1 y valor2, a las cuales se les iguala a 0 y a 1 respectivamente. Inmediatamente se ejecuta una estructura for la cual itera “i” a lo largo de la lista creada anteriormente, sumando cada elemento de la lista a la variable valor1. En seguida se ejecuta otra estructura for _ in, la cual de nuevo itera una variable a lo largo de la lista, pero multiplicando cada elemento de la lista a la variable valor2 y asignando ese nuevo valor a la variable. Finalmente se imprimen ambas variables, las cuales guardan la suma de todos los elementos de la lista, y el producto de todos los elementos de la lista.

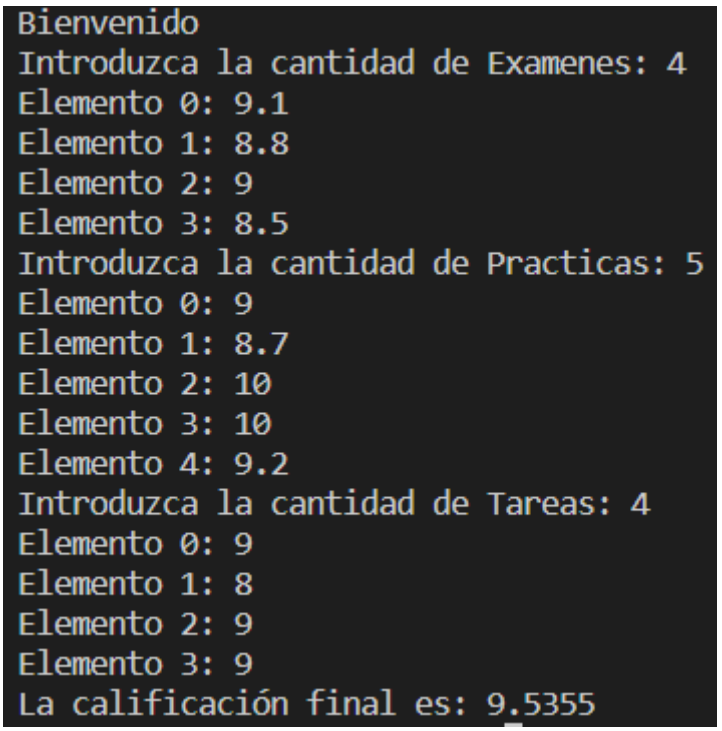
Actividad 3

Para el programa de este ejercicio primero se crea una función que servirá para recolectar las calificaciones de un conjunto de elementos en una lista. La función recolectarDatos() recibe como parámetros una lista en la que se guardarán las calificaciones y el número de calificaciones que se introducirán. La función ocupa un ciclo for y ocupa la función range() usando como parámetro el número de calificaciones a introducir; dentro de la estructura de repetición se guarda en una variable la calificación y se transforma en un flotante, para después usar el método append() de las listas para incluirlo al final de la lista.

También se creó una función que recibe como parámetro una lista, y devuelve el promedio de los elementos de la misma. En esta función se declara una variable, llamada suma la cual se hará de tipo flotante para guardar la suma de todas las calificaciones de la lista. Ocupando un ciclo for que itera en la lista que se recibe como parámetro de la función, se suma cada elemento de la lista a la variable suma. Finalmente se retorna el resultado de dividir suma entre lo que retorne la función len() introduciendo como parámetro la lista, siendo el número de elementos de la lista.

Ya declaradas las funciones se crearon 3 listas, para los exámenes, tareas, y prácticas respectivamente; además de una lista que contiene las listas antes mencionadas. Adicionalmente se creó una tupla que contiene las cadenas "Examenes", "Practicas" y "Tareas", que se ocupa enseguida para la impresión en pantalla. Después se ocupa un for ciclo in range(3) en el cual se ocupará la variable ciclo para iterar por listas ocupándose como índice. Dentro del for, se pregunta por la cantidad de calificaciones que se van a introducir, dato con el cual enseguida se llama a la función recolectarDatos() ocupando como parámetros la lista de listas usando como índice la variable ciclo, y la cantidad de calificaciones a introducir. Esto se repetirá 3 veces hasta llenar las calificaciones de exámenes, tareas y prácticas.

Después se calcula la calificación sin el extra de las tareas en la variable calificacion, siendo que el 65% equivale a los exámenes y el 35% a las prácticas. Después se calculara el extra de las tareas, llamando la función promedio() a las lista “tareas”; enseguida se calcula con una estructura if-elif-else si el promedio es mayor a 8.5 el extra será 0.5, si es menor o igual a 7 será -0.5, mientras que si no cumple las condiciones anteriores el extra será 0. Finalmente se suma el extra a la calificación calculada y se imprime en pantalla.



Conclusión

Tras haber hecho los ejercicios de la práctica, puedo afirmar que se cumplieron los objetivos propuestos en la misma, ya que se pudieron comprender y aplicar conocimientos del lenguaje Python en él personalmente nuevo entorno de Jupyter Notebook. La primera actividad es apropiada para familiarizarse y experimentar un poco con la sintaxis del lenguaje, así como con las funcionalidades del entorno de Jupyter Notebook, especialmente las celdas. Así mismo la segunda actividad me parece también fundamental para identificar la manera de ejecutar archivos desde la terminal. Finalmente el tercer ejercicio me gusto bastante ya que teniendo en cuenta las herramientas mencionadas en la primera actividad, brotan varias maneras de realizar la tarea solicitada en la tercera actividad. Una herramienta que me pareció interesante es y creo que se podría crear una actividad interesante con ella es la de los diccionarios, considero que la inclusión de una actividad que incite al alumno a ocuparlos podría estar bien.

Bibliografía

Python Tutorial. (n.d.). https://www.w3schools.com/python/