Carátula para entrega de prácticas	
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	Edgar Tista García
Asignatura:	Estructuras de Datos y Algoritmos I
Grupo:	1
No. de Práctica(s):	9
Integrante(s):	Ugalde Velasco Armando
No. de Equipo de cómputo empleado:	
No. de Lista:	38
Semestre:	2020-2
Fecha de entrega:	26 de marzo de 2020
Observaciones:	

CALIFICACIÓN: _____

OBJETIVO

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

ACTIVIDAD 1

I) EJECUTA LA APLICACIÓN JUPYTER NOTEBOOK Y ABRE EL ARCHIVO CORRESPONDIENTE A LA PRÁCTICA 9. ESCRIBE COMENTARIOS GENERALES PARA CADA SECCIÓN RESPECTO A PYTHON Y SU DIFERENCIA CON EL LENGUAJE C.

1. Variables y tipos

- Las reglas para los identificadores son muy similares en ambos lenguajes.
- Al ser Python un lenguaje dinámico, no es necesario especificar el tipo de dato a utilizar, a diferencia de C.
- En Python, las indentaciones son necesarias para la correcta escritura del código.

2. Cadenas

- En C, las comillas simples se utilizan para expresar caracteres. En cambio, en Python los caracteres son considerados como cadenas, las cuales es posible expresar utilizando comillas simples o comillas dobles.
- Al igual que en el lenguaje C, en Python es posible “escapar” caracteres utilizando /.
- El método **format** tiene un funcionamiento parecido a los especificadores de formato en C.

3. Operadores

- Los operadores aritméticos son los mismos en ambos lenguajes, a excepción del operador **, que sirve para calcular un número elevado a una potencia.
- Los operadores booleanos, en Python, son las palabras **and**, **or** y **not**, lo cual hace más legibles y explícitas las operaciones realizadas.
- En Python, los valores Booleanos se encuentran predefinidos (**True** y **False**).

4. Listas

- Las listas cumplen con la misma funcionalidad que los arreglos en C. Sin embargo, éstas son mucho más flexibles, ya que es posible modificar su tamaño de forma indefinida. Además, sus métodos nos proveen de funcionalidades muy útiles.

5. Tuplas

- En C, es posible modificar los valores dentro de un arreglo. Sin embargo, no es posible realizar lo anterior en una tupla, ya que es inmutable.

- Los arreglos en C son estáticos, es decir, no cambian de tamaño. Lo mismo sucede con las tuplas.

6. Tupla con nombre

- El funcionamiento de las tuplas con nombre es parecido al de una estructura en C: se especifican los elementos que contendrá, y posteriormente se crean ejemplares con los valores respectivos. Sin embargo, las tuplas son inmutables, a diferencia de las estructuras en C.

7. Diccionario

- En C no existen diccionarios de forma nativa.

- Las llaves en los diccionarios tienen la misma función que un índice en un arreglo: localizar algún elemento, sin embargo, los arreglos son estructuras ordenadas, y los diccionarios no.

- Son muy flexibles a comparación de los arreglos, ya que es posible cambiar su tamaño de forma indefinida y utilizar cualquier tipo de dato para fungir como llave.

8. Funciones

- Las funciones en ambos lenguajes son muy similares. Sin embargo, en Python la sintaxis es un tanto distinta.

- En Python, no es necesario especificar el tipo de dato de salida ni los tipos de dato de los parámetros.

- En Python no se utilizan los corchetes para delimitar las declaraciones de funciones: se utilizan dos puntos (**:**) y se realiza la indentación pertinente.

- En Python es posible retornar más de un valor, utilizando una tupla, una lista o un objeto.

8.1 Variables Globales

- Al igual que en el lenguaje C, las variables globales se definen en el “bloque” más externo. Por ejemplo, en C es posible definir variables globales al inicio del programa, fuera de la función **main**.

- En Python, cualquier variable que no se encuentre definida dentro de un “espacio local” es una variable global.

II) GENERA LAS CELDAS CORRESPONDIENTES PARA LOS SIGUIENTES EJERCICIOS:

Escribir dos funciones que permitan:

1) Convertir millas a kilómetros y viceversa.

Para cumplir con el requerimiento establecido, se realizó la siguiente función:

```
def millasKm ():

    num = int(input("Valor: "))

    print("1) Millas a km")
    print("2) Km a millas")
    opc = int(input())

    while(opc != 1 and opc != 2):
        opc = int(input("Digite una opción válida: "))

    if (opc == 1):
        print("Kilometros: " + str(num*1.60934))
    else:
        print("Millas: " + str(num*0.621371))

millasKm()
```

```
Valor: 32
1) Millas a km
2) Km a millas
1
Kilometros: 51.49888
```

Función millas Km

Primero, se solicitó al usuario que ingresara el valor a convertir, utilizando la función input y convirtiendo el valor obtenido (cadena) a un entero.

Posteriormente, se solicitó la opción deseada, asegurando que ésta fuera un número válido (1 o 0). Finalmente, se imprimió el resultado deseado, realizando el cálculo pertinente dentro de la función **print**. Al final del fragmento de código podemos observar la expresión que ejecuta esta función.

Se muestra al final de la captura la salida de una ejecución del programa.

2) Convertir grados centígrados a Fahrenheit y viceversa.

Se realizó el mismo procedimiento que en la función anterior, a excepción de los cálculos realizados, que se muestran a continuación.

```
    if (opc == 1):
        print("Fahrenheit: " + str(num*1.8 + 32))
    else:
        print("Centigrados: " + str((num - 32) / 1.8))

centiFahr()
```

```
Grados: 0
1) Centígrados a Fahrenheit
2) Fahrenheit a Centígrados
2
Centigrados: -17.77777777777778
```

Cálculos realizados y salida del programa

En la captura de pantalla podemos observar que los cálculos realizados corresponden a la fórmula de conversión en cada caso. Al final, logramos observar la salida de una ejecución de programa, que muestra un resultado correcto.

ACTIVIDAD 2

```
(base) PS C:\Users\armau\OneDrive\Documentos\Escuela\EDA 1\Práctica 6> python -V
Python 3.7.4
(base) PS C:\Users\armau\OneDrive\Documentos\Escuela\EDA 1\Práctica 6>
```

Versión de Python

a) Indica la salida del programa

A continuación, se muestra la captura de pantalla de la salida del programa:

```
(base) PS C:\Users\armau\OneDrive\Documentos\Escuela\EDA 1\Práctica 6> python ejercicio.py
116
1764322560
```

b) Explica el funcionamiento del programa

En el programa se crea una lista con los números entre 11 a 18. Después, se declaran e inicializan dos variables que fungen como acumuladores: **valor1 = 0**, y **valor2 = 1**. Se ejecutan dos ciclos **for in**: el primero suma todos los números en la lista y los almacena en la variable **valor1**, y el segundo multiplica todos los valores en la lista y los almacena en **valor2**.

Finalmente, se imprimen los valores de las variables.

ACTIVIDAD 3

Análisis

Trabajar con **Jupyter Notebook** facilita en gran medida el desarrollo de programas en Python, ya que nos proporciona una interfaz amigable, clara y fácil de manipular. Podemos ejecutar los programas escritos con un simple clic, a diferencia de la línea de comandos, donde tenemos que ejecutar el comando Python y especificar la dirección el archivo cada que deseemos hacerlo.

Además, esta herramienta nos permite crear diferentes secciones para expresar más claramente nuestro código y modularizarlo, lo cual es un aspecto fundamental en la programación. También presenta nuestro código en diferentes colores, facilitando su legibilidad, y muestra sugerencias en él, como el autocompletar la escritura de un método.

Estas características nos facilitan sin duda alguna la escritura de nuestros programas.

CONCLUSIONES

En la práctica se obtuvieron las bases necesarias para realizar programas en el lenguaje Python. Se presentaron los conceptos fundamentales propios del lenguaje, es decir, sus variables, tipos, operadores, y sus principales estructuras de datos: listas, tuplas y diccionarios. El lenguaje presenta diferencias y similitudes con el lenguaje C, las cuales se analizaron a profundidad en la actividad 1. Algunas de ellas son:

- El lenguaje Python es interpretado y C es compilado.
- Python presenta tipado dinámico, a diferencia de C.
- Los operadores son parecidos, pero presentan diferencias notables.

- Python cuenta con estructuras de datos predefinidas, las cuales cuentan con métodos para su manipulación.

Para la ejecución de los programas se utilizó el ambiente de iPython Notebook, ahora llamado **Jupyter Notebook**, el cual facilita en gran medida la escritura de nuestro código. Proporciona funcionalidades muy útiles: nos permite ejecutar nuestros programas con un solo click y mejora en gran medida la legibilidad del código, proporcionando la capacidad de crear diferentes secciones, dando sugerencias para nuestro código, y resaltando de diferentes colores las expresiones de nuestros programas.

Las actividades realizadas en la práctica me parecieron adecuadas para introducir al alumno al lenguaje. Sin embargo, en mi opinión, una alternativa más cómoda y práctica podría ser realizar y ejecutar los programas en un IDE de forma nativa, como Spyder o IDLE, los cuales pienso que son más completos que Jupyter.