



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: TISTA GARCÍA EDGAR

Asignatura: ESTRUCTURA DE DATOS Y ALGORITMOS I

Grupo: 1

No de Práctica(s): Guía práctica de estudio 09: Introducción a Python (I).

Integrante(s): MURRIETA VILLEGAS ALFONSO

Semestre: 2018 - 2

Fecha de entrega: 16 DE ABRIL DE 2018

Observaciones:

CALIFICACIÓN: _____

INTRODUCCIÓN A PYTHON

INTRODUCCIÓN

Dentro de los lenguajes de programación existen distintas clasificaciones ya sea por la generación o por el paradigma, en este caso, Python a diferencia del lenguaje C es un lenguaje interpretado donde al momento de escribir líneas de código de un programa todas estas son “interpretadas” por un intérprete que es un programa capaz de analizar y ejecutar otros programas, por otro lado, el lenguaje C es un lenguaje compilado, donde una vez creado el código fuente mediante un compilador se crea un código objeto el cual posteriormente pasa a ser un programa ejecutable.

Python es un lenguaje de alto nivel, caracterizado por ser multi-paradigma, además, este lenguaje de programación que se caracteriza por ser interpretado y cuya filosofía hace hincapié en una sintaxis donde se favorece al código legible.

OBJETIVOS DE LA PRÁCTICA

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

DESARROLLO

1. Antecedentes

Python fue creado a finales de 1980 por el programador holandés Guido van Rossum en el CWI (Centrum Wiskunde & Informatica), en los Países Bajos, como un sucesor del lenguaje de programación ABC.

Desde su creación hasta hoy en día han existido 2 versiones sobresalientes de este lenguaje de programación, el primero publicado alrededor del año 2000, la versión 2 de Python que se caracterizó por ser la versión que se dio a conocer en el mundo (Incluyó un recolector de basura y la administración de memoria), sin embargo, uno de los eventos más importantes y controversiales fue el salto que dio entre la versión 2 a la versión 3 en el año 2009, donde bajo el nombre de “Py3k” realizó bastantes cambios incluso en la parte sintáctica en la que se escribía el lenguaje.

2. Análisis de actividades de la práctica

Para poder hacer un mejor análisis de toda la parte teórica del manual de prácticas, se decidió separar en distintos sub-apartados todo lo mencionado en la guía práctica de estudio 9.

3.1 Python Notebook

3.1.1 Análisis entre Python y el lenguaje C

1. VARIABLES Y TIPOS

Las variables en Python tienen la característica de ser alfanuméricas, con características como:

- 1) No se especifican el tipo de valor que va a contener la variable debido a que esta información se encuentra de manera implícita al momento de asignar valores.
- 2) Se omite el uso de “ ; ” al final de cada instrucción.

NOMBRES RESERVADOS

Al igual que otros lenguajes de programación, existen palabras reservadas dentro de Python, a continuación, se muestran todas las palabras:

and, as, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while, with, yield.

CARACTERÍSTICAS EXTRAS

Algunas otras características son:

- 1) La escritura de cadenas o conjunto de caracteres es más sencillo algunos ejemplos del uso de estas son:

```
#Inicializando variables
x = 10      #variable de tipo entero
print(x)    #función para imprimir los valores de las variables

#Se puede utilizar comillas dobles o simples para crear una cadena
cadena = "Hola Mundo"      #variable de tipo cadena
print(cadena)
```

10
Hola Mundo

Donde podemos ver como la variable cadenas guarda todo el conjunto de caracteres “Hola mundo”.

- 2) También podemos ver que el uso de “type” como una función empleada para conocer el tipo de variable que estamos usando, ya sea una cadena (Str=string) o un entero (Int=entire).

- 3) Por otro lado, cuando una variable tiene un valor constante, el nombre se escribe en mayúsculas.

```
SEGUNDOS_POR_DIA = 60 * 60 * 24  
PI = 3.14
```

COMPARACIÓN ENTRE LENGUAJE C Y PYTHON

En este apartado realmente podemos notar grandes diferencias con respecto al lenguaje C, lo primero es que no necesitamos declarar el tipo de variable que vamos a utilizar ya que de eso en cierta forma se encarga el lenguaje y el intérprete (Realmente notamos que está implícito esta información), otra gran diferencia es que aquí ya podemos manejar cadenas de caracteres y no arreglos de caracteres como en C.

Por último, también podemos notar que dentro de la parte sintáctica no hay tantos cambios salvo la excepción del “;”.

2. CADENAS

En el caso de Python cuando guardamos una podemos utilizar ya sea las comillas simples o dobles, y una de las características importantes de estas es que las cadenas son inmutables ya que los caracteres que tienen no se pueden cambiar.

A continuación, se muestra un caso simple del uso de cadenas dentro de Python.

```
#Inicializando cadenas  
cadena1 = 'Hola '  
cadena2 = "Mundo"  
print(cadena1)  
print(cadena2)  
concat_cadenas = cadena1 + cadena2 #Concatenación de cadenas  
print(concat_cadenas)
```



```
Hola  
Mundo  
Hola Mundo
```

Lo primero que podemos ver es el uso de variables de tipo *string* encargadas de guardar una palabra entera, por otro lado, notamos algo realmente interesante y es la forma en cómo podemos emplear una tercera variable que a través del operador de suma simplemente estamos conjuntando o concadenando dos palabras

Sin embargo, debido a que poco a poco se van agregando nuevas implementaciones en cada versión de Python, realmente existen muchísimas funciones dentro del lenguaje o en bibliotecas que dan la oportunidad de hacer una infinidad de cosas a las cadenas, por ejemplo, el uso de la función “format()” es sin duda el más claro ejemplo dentro de la concatenación de cadenas, a continuación se muestra el uso de esta.

```
In [3]: cadena1= "Hola"
cadena2= "todos"
ConCade = "Texto total: {} {} {}".format(cadena1, cadena2, 3)
print(ConCade)

Texto total: Hola todos 3
```

COMPARACIÓN ENTRE LENGUAJE C Y PYTHON

El uso de cadenas es una de las facilidades que han traído las nuevas generaciones de los lenguajes de programación, Python es un lenguaje que realmente facilita al usuario para poder hacer un sinfín de acciones sobre las cadenas, el más claro ejemplo es la enorme cantidad de funciones que hacen acciones como volver todos elementos en mayúsculas, etc. C realmente es un lenguaje que pese tiene mucho tiempo desde su invención, también puede manejar cadenas*, pero no le da tanta comodidad al programador.

Las principales diferencias las notamos realmente en el manejo de estas (La declaración realmente no es muy diferente).

3. OPERADORES

Al igual que la mayoría de los lenguajes de programación los operadores aritméticos son los mismos, donde la suma se hace a través de “+”, la resta mediante “-”, la multiplicación “*” y la división con “/”.

A continuación, se muestra un pequeño código encargado de mostrar cómo se utilizaría algunos operadores extras dentro de Python

```
In [5]: a=10
b=3

#division natural resultado = 10/3 que es aprox 3.333
resultado = a / b
print(resultado)

#division con floor/piso, el resultado se redondea hacia abajo
resultado = a // b #resultado = 3, no 3.3333
print(resultado)

#modulo/residuo, el residuo de una division
resultado = a % b #resultado es 1
print(resultado)

#exponencial, eleva a la potencia de
resultado = a **3 #10 elevado a 3 = 1000
print(resultado)

3.333333333333333
3
1
1000
```

Lo primero que podemos ver es que la salida de la primera división donde el resultado tiene como tipo de valor un float el cual a su vez puede notarse al último un número 5 que como vimos en capítulos anteriores se debe a la forma en que las computadoras asumen la conversión de entero a decimales (Punto flotante), por otro lado, también vemos como en la segunda división solamente se toma la parte entera de la división mediante el operador “//”.

Como operadores extras vemos el modulo que al igual que C se utiliza el operador “%” y por último vemos como el uso de ** sirve para hacer la función de potencias.

COMPARACIÓN ENTRE LENGUAJE C Y PYTHON

Al igual que la gran mayoría de lenguajes de programación, los operadores aritméticos son prácticamente los mismo, en este apartado lo que realmente podemos ver es que no hay grandes diferencias, salvo tal vez la manera en que se realizan algunas operaciones como el manejo de potencias o el “truncar”.

3.1.2 Función Millas - Metros

Para este apartado lo que se pidió fue que se creara un programa donde a través del uso de funciones se pudiera calcular la cantidad de metros en una distancia dada en millas y pies, y la cantidad de millas dada una distancia de metros y centímetros.

Lo primero que se tuvo que hacer fue buscar los valores de conversión entre las distintas unidades, las cuales se pueden ver en la siguiente imagen (Dentro de los óvalos rojos).

```
def metro_milla():#Función de conversión de metros a millas
    metros= eval(input("Ingrese la cantidad de metros:"))
    millas = metros * 0.000621371
    pies = metros * 3.28084
    print("La cantidad de millas son: ", millas)
    print("La cantidad de pies son: ", pies)

def milla.metro():#Función de conversión de millas a metros
    milla= eval(input("Ingrese la cantidad de millas:"))
    metro = milla * 1609.34
    cm = metro*100
    print("La cantidad de millas son: ", metro)
    print("La cantidad de pies son: ", cm)
```

Posteriormente, se tuvieron que crear 2 funciones las cuales se encargaran de cada una de las correspondientes correcciones, cabe destacar que estas funciones se encargan tanto de pedir los valores al usuario, como de transformarlos y posteriormente imprimirlos.

Sin embargo, para que el usuario tuviera la decisión de escoger cuál de las conversiones quiere se creó otra función principal la cual tiene como principal objetivo desplegar un pequeño menú de opciones que a través de condiciones if se encarga de llamar a la función correspondiente. A continuación, se muestra la salida de nuestro programa.

```

Terminal de IPython
Terminal 1/A
Programación/Python/EDA-1/1_practice/1_ejercicio(VF).py', 
wdir='C:/Users/pon_c/Downloads/Documents/Programación/
Python/EDA-1/1_practice')
CONVERSOR DE UNIDADES
1) Metros a Millas/Pies
2) Millas a Metros/Centimetros

Ingrese el número de la conversion quiere hacer: 1

Ingrese la cantidad de metros:10
La cantidad de millas son:  0.00621371
La cantidad de pies son:  32.8084

In [8]: 
```

Salida del programa empleando el IDE Spyder

```

if opcion == 2: #Llamamos a la función de conversión de milla a metro
    milla_metro()

main()

#RECORDATORIOS

#1) No se cierran las funciones extras o secundarias
#2) Eval sirve solamente para números no para cadenas

CONVERSOR DE UNIDADES
1) Metros a Millas/Pies
2) Millas a Metros/Centimetros
Ingrese el número de la conversion quiere hacer: 1
Ingrese la cantidad de metros:10
La cantidad de millas son:  0.00621371
La cantidad de pies son:  32.8084

```

Salida del programa empleando el programa Jupyter Notebook

Lo primero se debe destacar es que para una mayor comodidad y facilidad al momento de programar se decidió programar en Spyder, por otro lado, realmente durante el tiempo en el de realización de este programa, hubo bastantes problemas al momento de realizar conversiones, el error que daba era que no se podían asociar enteros con flotantes, para ello y a través de la búsqueda en el libro “Python 3 Curso Práctico” (Ver en referencias) mediante el uso de la función eval() simplemente metíamos la parte de asignación de valores y fácilmente omitíamos el anterior (Ver imagen inferior).

```
metros= eval(input("Ingrese la cantidad de metros: " ))
```

Una de las características o condiciones de este problema era resolverlo a través del uso de programa o ambiente de trabajo “Jupyter Notebook” el cual realmente resulta amistoso para programar programas sobre todo a través del uso de celdas.

NOTA: Un pequeño detalle de este programa es que, como resultado final tenemos un archivo ejecutable con la extensión .ipynb que es un documento el cual está destinado a trabajarse en el ambiente de trabajo “Jupyter Notebook”, a diferencia del .py que es una extensión más universal de archivos escritos en Python.

3.2 Manejo de archivos y ejecución por líneas de comando

Para esta actividad lo primero que se pidió es que no se utilizara Jupyter Notebook, se dio la posibilidad de poder escribir el código en cualquier editor de texto, en mi caso, empleé “Visual Studio Code”, además de que se pidió que a través de la consola (Al ser Windows nos referimos a MS-Dos) se ejecutara el programa utilizando las líneas de comando “*Python “Nombre.extensión”*”, sin embargo, debido a que en visual studio code anteriormente se instalaron plugins con la finalidad de que se pudiera ejecutar los programas de manera automática, no hubo necesidad de que tuviéramos que escribir líneas en la consola para abrir nuestro programa. A continuación, se muestra la salida del programa (Véase en la imagen inferior).

```
[Running] python "c:\Users\pon_c\Downloads\Documents\Programación\Python\EDA-1\1_practice\actividad_2.py"
116
1764322560

[Done] exited with code=0 in 0.121 seconds
```

Como se muestra en la imagen anterior, como resultados tenemos la impresión de 2 valores, el primero es 116 y el segundo es 1764322560.

Lo primero que podemos ver en nuestro código son 2 principales ciclos *for* donde en el primer ciclo *for* lo que estamos haciendo es ir sumando valor por valor los elementos de “lista”, es por ello que inicializamos la variable *valor1* en 0 ya que solamente queremos saber el valor de la suma de todos los integrantes de “lista”.

Por otro lado, lo que tenemos en el segundo ciclo *for* es la multiplicación de todos los elementos de “lista”, es por ello que en este caso la variable “valor 2” está inicializada en 1 (Recordemos que si multiplicamos por 0 a cualquier valor nos dará como resultado 0).

Por último, solamente tenemos la parte de la salida o impresión que se encarga de imprimir o dar a conocer los valores finales de ambos ciclos al usuario.

3.3 Diferencias entre Jupyter Notebook y Ejecución en consola

Realmente este es un apartado interesante, el pensar que hace no más de 25 años la gran mayoría de programadores desarrollaban, y ejecutaban sus programas en consola, solamente me hace reflexionar en que tan fácil ahora resulta utilizar un editor de texto o un IDE para el desarrollo de nuestros proyectos.

La verdad, el trabajar en Jupyter Notebook es mucho más fácil que el trabajar en la consola y sobretodo porque no tenemos que estar buscando el archivo que queremos correr con el intérprete, además que visualmente es más atractivo y cómodo

Otro caso que quería mencionar es que por ejemplo para el apartado 2 utilicé Visual Studio Code el cual al instalarle algunos plugins me facilitó la forma en que puedo ejecutar los programas escritos en Python.

4. Relación entre ejercicios y tema

Los ejercicios que se realizaron a lo largo de esta práctica tuvieron como principal característica conocer a un nuevo lenguaje, por ejemplo, en el segundo ejercicio sin duda alguna demuestra como los ciclos (En ese caso el ciclo for) cambia bastante la forma sintáctica con respecto al lenguaje C, sin embargo, tienen la misma finalidad y trabaja de la misma forma, salvo que en este caso el contador no debemos declarar previamente.

Otro detalle del segundo problema fue como mediante el uso de distintos operadores podemos ir realizando distintas acciones, por ejemplo “+” solo suma los elementos mientras que “*” los va multiplicando.

Por último, en la primera actividad la creación del programa a través de funciones para hacerlo modular y más ordenado fue realmente fácil, salvo el hecho de los problemas que se tuvieron al realizar las conversiones, pero sobretodo este ejercicio creo que tenía la finalidad de demostrarnos que hacer programas modulares es sin duda algo que mejorará nuestro desempeño como programadores.

CONCLUSIONES

Esta fue la primera práctica donde se empleó otro lenguaje de programación que no fuese C, realmente el dar un salto entre lenguajes de programación implica un cambio realmente importante, sobre todo porque hemos cambiado realmente de un lenguaje compilado a un lenguaje interpretado, y también, porque Python es un lenguaje moderno, con una visión que está destinada a los programadores quienes quieren hacer códigos realmente fáciles de entender.

Los principales problemas a los que me tuve que afrontar fue el cómo se manejaban los valores dentro del programa sobre todo porque a diferencia del lenguaje C en Python no se declaran las variables, lo cual provocó muchos errores al momento de realizar operaciones aritméticas.

Por otro lado, en esta práctica realmente me fue interesante trabajar y ejecutar mis programas en diferentes editores o IDE's sobre todo porque al inicio pese empecé utilizando Jupyter Notebook realmente no me pareció la más cómodo y peor aun haciéndolo desde el mismo intérprete de Python, al final fui conociendo otros ambientes como fue Visual Studio Code y por último y realmente agradable, Spyder.

Python sin duda es un lenguaje de programación que en verdad me tiene intrigado, sobre todo por la forma en que se trabaja, en cierta forma lo veo realmente agradable porque si se tiene un mínimo conocimiento de inglés realmente pareciera que es un texto escrito, sin embargo, también algo interesante es como existen prácticamente una infinidad de funciones para realizar cualquier cosa.

Por último, como se vio en esta práctica, la gran diversidad de editores de textos e IDE's nos proporcionan la facilidad de poder programar de manera más cómoda, sin embargo, un aspecto importante radica en la forma en la que salen nuestros programas, puesto que por ejemplo en el caso del ambiente “Jupyter Notebook” notamos que la salida o extensión de nuestros programas es **.ipynb** mientras que en los IDE's o editores de texto como son Visual Studio Code o Spyder la salida es directamente un archivo de Python con la extensión **.py** .

REFERENCIA

The Python Language Reference. Recuperado el 20 de abril de 2018, de <https://docs.python.org/3/reference/index.html>

Mark J. Guzdial, Barbara Ericson. (2015). *Introducción a la computación y programación con Python*. 3ra Edición. Madrid España.

Tutorial de Python. Recuperado el 20 de abril de 2018, de <http://docs.python.org.ar/tutorial/pdfs/TutorialPython2.pdf>

Solano Jorge A. (2017). *Manual de Prácticas del laboratorio de Estructuras de Datos y algoritmos 1*. UNAM, Facultad de Ingeniería.

Cuevas Álvarez. (2016). Python 3 Curso Práctico. Ra – Ma.