**Universidad Nacional Autónoma de México**

Facultad de Ingeniería

**Laboratorios de computación**

**salas A y B**

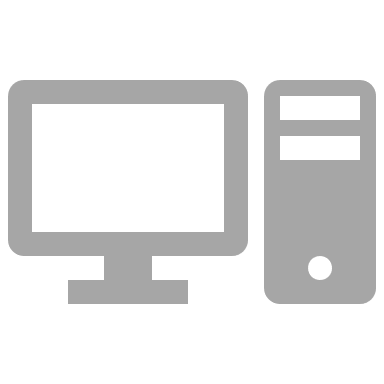
**PROFESOR:** M.I. Marco Antonio Martínez Quintana

**ASIGNATURA:** Estructura de Datos y Algoritmos I

**GRUPO:** 17

**NO DE PRÁCTICA:** 9

**NOMBRE:** Reyes Mendoza Miriam Guadalupe

**SEMESTRE:** 2020-2

**FECHA DE ENTREGA:** 31/03/2020

**OBSERVACIONES:**

**CALIFICACIÓN:**

**INTRODUCCIÓN A**

**PYTHON (I)**

**OBJETIVO**

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

**INTRODUCCIÓN**

**HISTORIA, ANTECEDENTES Y EVOLUCIÓN**

Python es un lenguaje de programación orientado a objetos. La programación orientada a objetos (OOP) se refiere a un tipo de programación de computadora en la que los programas definen no sólo el tipo de datos de una estructura, sino también los tipos de operaciones (*funciones*) que pueden aplicarse a la estructura de datos.

De esta manera la estructura de datos se convierte en un objeto qué incluye tanto datos como funciones. Además, los programadores pueden crear relaciones entre un objeto y otro, por ejemplo.

Con Python trabajaremos con dos de los múltiples paradigmas de programación: imperativa y funcional. La característica distintiva entre ellos es el concepto de estado. En un lenguaje ***imperativo*** el estado de la computación *se representa en los valores de las variables que tengamos en el programa*. Cada sentencia (instrucción) revisa un cambio definitivo del estado, añadiendo, cambiando o eliminando una variable. De forma ideal cada sentencia avanza el estado del cómputo desde un estado inicial hasta el resultado final deseado.

La programación ***funcional*** reemplaza el estado (valores cambiantes de las variables) por la noción simple de *evaluación de funciones*. Cada evaluación de Función crea un nuevo objeto u objeto a partir de objetos existentes. Dado que un programa funcional es una composición de funciones podemos diseñar funciones de nivel inferior factibles de entender, siendo esa composición más fácil de visualizar que una secuencia compleja de sentencias.

**VARIABLES Y TIPOS**

La denominación de las variables sigue el concepto más general de identificador, que es un nombre utilizado para identificar entidades (variables, funciones, clases, módulos u otros objetos).

* Pueden ser una combinación de letras en minúscula (**a - z**) o en mayúsculas (**A - Z**), dígitos (**0 - 9**) y el símbolo de guión bajo (“**\_**”).
* No puede comenzar con un dígito.
* No podemos usar símbolos especiales (“**@**”, “**!**”, “**#**”, “**$**”, “**%**”, …).
* No se necesita poner “**;**” al final de cada instrucción.
* Puede tener cualquier longitud.

Captura de pantalla de un celular

Descripción generada automáticamenteCaptura de pantalla de un celular

Descripción generada automáticamente

**CADENAS**

En su forma más sencilla son vectores de letras que definimos para formar un texto. Podemos utilizarlas para almacenar mensajes que nos sirvan para formar texto.

Las cadenas de texto pueden expresarse de diferentes maneras. Una forma de definirlas es encerrarlas entre comillas simples (**‘ ’**) o dobles (**“** **”**).

**Nota:** Existen caracteres especiales que no pueden ingresarse de manera directa en la cadena y para poder introducirlos se suele utilizar **\**.

Una captura de pantalla de una red social

Descripción generada automáticamente

**OPERADORES**

Un operador un Python es un símbolo especial que realiza un cálculo aritmético o lógico sobre uno o varios elementos. El operador se denomina *unario* su actúa sobre un único operando; si lo hace sobre dos se denomina *binario*. Una *expresión* es una construcción sintética compuesta por objetos junto a los operadores.

Existen varios tipos de operadores: aritméticos, de comparación, lógicos, a nivel de bit (bitwise), de asignación o especiales (de identidad y de membresía)

Captura de pantalla de un celular

Descripción generada automáticamente

**LISTAS**

Una lista es una *secuencia contenedora mutable.* Una secuencia es cualquier estructura de datos iterable, con un tamaño conocido, que permite el acceso a sus ítems vía índice entero base 0. Que sea contendora significa que almacena referencias de objetos en lugar del valor de sus ítems como lo hacen las secuencias planas. Ser mutable nos permitirá cambiar el valor de sus elementos que la componen. Las listas no tienen un tamaño determinado.

Podemos crear listas de varias maneras:

* Usando un par de corchetes para indicar una lista vacía: **[ ]**
* Usar un par de corchetes incluyendo los ítems separados por comas: **[2, 7, 8]**
* Usando el *constructor*: ***list (****‘Hola’****)***
* Usando las denominadas *listas de compresión*.

Captura de pantalla de un celular

Descripción generada automáticamente

Captura de pantalla de un celular

Descripción generada automáticamente

**TUPLAS**

Las tuplas son *secuencias contenedoras inmutables*. Al ser secuencias inmutables, en ellas no podremos cambiar el valor de sus elementos, y es ésa la diferencia principal respecto a las listas. La tuplas no tienen un tamaña no determinado.

Además, aportan mayor seguridad sobre la listas y tienen un ligero aumento de rendimiento al iterar sobre ellas.

Podemos crear tuplas de varias maneras:

* Usando un par de paréntesis para indicar una tupla vacía: **( )**
* Separando los ítems por comas: **( 2, 7, 8)**
* Usando el *constructor*: ***tuple (****‘Hola’****)***

En realidad, son la comas las que crean las tuplas, siendo opcionales los paréntesis (salvo en el caso de la tupla vacía en el que pueda haber algún tipo de ambigüedad).

Captura de pantalla de un celular

Descripción generada automáticamente

Una captura de pantalla de una red social

Descripción generada automáticamente

Una captura de pantalla de una red social

Descripción generada automáticamente

**DICCIONARIOS**

Los diccionarios son **mapas** de colecciones *contenedoras mutables desordenadas* de objetos cada uno de ellos en lugar de una posición relativa, mediante una clave.

Los diccionarios son contenedores, por lo que pueden albergar cualquier tipo de objeto y no tienen un tamaño determinado.

Los diccionarios son mutables como las listas, y al igual que ellas son una herramienta flexible para representar colecciones, sus claves más nemotécnicas se adaptan mejor cuando se nombran los elementos de una colección o tenemos campos etiquetados de un registro de base de datos, por ejemplo.

Las claves pueden ser cualquier objeto y no pueden estar duplicadas (se desaconseja el uso de números punto flotante como claves el diccionario ya que sabemos que se almacenan como aproximaciones).

Los diccionarios son objetos (instancias) de la clase ***dict***. Podemos crearlos de varias maneras:

* Usando un par de llaves para indicar un diccionario vacío: **{ }**
* Usando un par de llaves incluyendo los pares clave: valor separados por comas: **{ 1:** **‘Ana’ , 5:** **‘Pepe’ , 12:** **‘Eva’ }**
* Usando el constructor: ***dict (****x = 21****)***
* Usando los denominados *diccionarios de compresión*.

Una captura de pantalla de una red social

Descripción generada automáticamente

Una captura de pantalla de una red social

Descripción generada automáticamente

**FUNCIONES**

Las funciones sobre un dispositivo de estructuración de programas casi universal en los lenguajes de programación, donde podremos encontrarlas con el nombre de subrutinas o procedimientos. Las funciones sirven para maximizar la reutilización del código minimizar la redundancia. Además de ello nos permiten la descomposición procedural.

De programación procedural viviremos el programa en una serie de subprogramas, y usaremos una función para cada ello. Es más fácil llevar a cabo tareas más pequeñas de forma aislada que la ejecución de todo el proceso a la vez. Las funciones proporcionan una herramienta para dividir los programas en piezas bien definidas.

Al afrontar un problema complejo no intentaremos resolverlo directamente de forma total (con todos los detalles) sino que usaremos el diseño de arriba a abajo dividiendo, sucesivamente el problema original.

Captura de pantalla de un celular

Descripción generada automáticamente

Una captura de pantalla de una red social

Descripción generada automáticamente

**VARIABLES GLOBALES**

Captura de pantalla de un celular

Descripción generada automáticamenteSi se realizar una operación a nivel módulo (fuera de toda función) su ámbito será **global** es cierto todas las funciones incluidas en el módulo podrán acceder a ellas (pero no modificarlas).

Si se hace una función su ambito sera **local**, por lo que solo existira ahí. Cualquier modificación de la variable en el cuerpo de la función no tendrá ningún efecto en otras variables fuera de ella, incluso si tienen el mismo nombre.

Captura de pantalla de un celular

Descripción generada automáticamente

Una captura de pantalla de una red social

Descripción generada automáticamente

Captura de pantalla de un celular

Descripción generada automáticamente

**DESARROLLO Y RESULTADOS**

**JUPYTER NOTEBOOK**

Realizar un programa que calcule el **área** y el **perímetro** de las siguientes figuras:

**TRIÁNGULO**

Captura de pantalla de un celular

Descripción generada automáticamente

**CÍRCULO**

Captura de pantalla de un celular

Descripción generada automáticamente

**RECTÁNGULO**

Captura de pantalla de un celular

Descripción generada automáticamente

**TRAPECIO**

Captura de pantalla de un celular

Descripción generada automáticamente

**CONCLUSIONES**

Con esta práctica que es una breve introducción a Python pude observar que presenta muchas ventajas entre la cuales están su facilidad de uso y la legibilidad de código de una manera más fácil ara el usuario, es decir, es un lenguaje muy versátil el cual permite que pueda ser aplicado en diferentes ámbitos dentro de la programación.

Su sintaxis es muy sencilla, además de requiere menos líneas para realizar tareas básicas que si programamos en Java o C. Otra gran ventaja es que tiene una librería estándar, la cual permite ejecutar otras funciones y tareas más complejas con mayor facilidad que otros lenguajes.

Anteriormente no había tenido la oportunidad de trabajar con este lenguaje de programación, sin embargo, con esta práctica he podido darme cuenta de que es muy práctico y eficiente. Esto me hizo buscar información al respecto y darme cuenta de que realmente vale la pena aprender sus bases de forma precisa, por lo cual, me he dado a la tarea de buscar un libro que me ayude a dedicar parte de mi tiempo a conocer más acerca de Python, pues he leído que existen grandes compañías internacionales que lo utilizan para el *desarrollo de aplicaciones* y *sitios web*.

**BIBLIOGRAFÍA**

* Cuevas, A. (2019). *Programar con Python 3* (1ª edición). Madrid, España: RA-MA.
* Guagliano, C. (2019). *Programación en Python 1: Entorno de Programación- Sintaxis. Estructuras de Control* (1ª edición). Buenos Aires, Argentina: Six Ediciones.