# Taller Integrador - Python.

### Clase 1 – viernes 27/03/2020.

# Unidad 1.

### Introducción - Algoritmo.

En términos generales, la computación, también llamada Informática, es la disciplina que busca establecer una base científica para resolver problemas mediante el uso de computadoras.

Un algoritmoes un conjunto finito de instrucciones ejecutables, no ambiguas, que dirige a una actividad que termina o resuelve un problema en tiempo limitado. Por ejemplo, hay algoritmos para construir aeroplanos a escala (expresados en forma de hojas de instrucciones), para tocar música (en forma de partituras) para preparar platos de cocina (en forma de recetas), etc.

En el ámbito de los equipos de computación, los algoritmos se representan como programas dentro de las computadoras, que reciben colectivamente el nombre de software (a diferencia del equipo en sí, que constituye el hardware).

Para que una máquina efectúe una tarea, primero debemos diseñar un algoritmo para llevarla a cabo y luego programarlo en la máquina.

Una de las principales labores en el campo de la Computación consiste en el desarrollo de algoritmos, por lo que una parte importante de esta ciencia se ocupa de cuestiones relacionadas con esta tarea.

El estudio de los algoritmos comenzó como un tema de matemáticas, mucho antes de que se inventaran las computadoras actuales, y su principal objetivo era descubrir un conjunto único de instrucciones para resolver cualquier problema de un cierto tipo; así, una vez descubierto un algoritmo para efectuar una tarea, la realización de ésta ya no requeriría entender los principios en que se basa el algoritmo, sino tan sólo seguir las instrucciones.

En cierto sentido, la inteligencia requerida para llevar a cabo la tarea está codificada en el algoritmo.

### Alcance de la Computación

Se puede apreciar el alcance de la Computación o Informática, considerando algunos de sus objetos de estudio desde el punto de vista de los algoritmos, descubrir un algoritmo para resolver un problema equivale, en lo esencial, a descubrir una solución para ese problema). Los estudios de esta rama de la informática, denominada Algoritmia, se apoyan en áreas como la psicología humana de la resolución de problemas y las teorías de la educación.

Una vez descubierto un algoritmo para resolver un problema, se debe representar de manera que se lo pueda comunicar a una máquina o a otras personas. Se debe transformar el algoritmo conceptual en un conjunto claro de instrucciones y representar estas últimas sin ambigüedad. Los estudios que se ocupan de esto se basan en conocimientos del lenguaje y la gramática, y han dado lugar a un gran número de esquemas para representar algoritmos, llamados lenguajes de programación, fundados en diversos enfoques del proceso de programación, llamados paradigmas de programación(v.g. programación imperativa, programación funcional, programación lógica, programación orientada a objetos).

En general, la estructura algorítmica de los grandes sistemas automatizados se debe guiar por un proceso de ingeniería similar al de las máquinas mismas. Por otra parte, en muchos casos, como en los entornos comerciales, en el desarrollo y elaboración de programas no hace falta descubrir algoritmos radicalmente nuevos; más bien, el problema principal consiste en identificar qué sistemas automatizados se necesitan y cómo estos nuevos sistemas van a interactuar con los ya existentes. La combinación de estas dos problemáticas ha dado origen a la rama de la informática conocida como Ingeniería de Software.

Otra rama importante de la informática se ocupa del diseño y la construcción de máquinas, pero no con el objetivo de dominar los detalles de la realización de las máquinas actuales con circuitos electrónicos, puesto que esto es responsabilidad de la ingeniería electrónica, sino para comprender lo suficiente la tecnología de hoy de manera de apreciar sus ramificaciones y su influencia sobre el desarrollo de la informática.

De esto se ocupa una rama de la informática conocida generalmente bajo la denominación de Arquitectura de Computadoras.

Algo íntimamente relacionado con el diseño del equipo de cómputo es el diseño de la interfaz entre una máquina y el mundo exterior: la forma en que se insertan los algoritmos en una máquina, la forma en que se le indica a la máquina qué algoritmo debe ejecutar, etc.

La resolución de tales problemas en un entorno en el que se espera que la máquina suministre muy diversos servicios implican la coordinación de actividades y la asignación de recursos, temas que se estudian en la rama de la informática dedicada a los Sistemas Operativos.

La comunicación entre máquinas y los problemas inherentes se estudian en una rama a menudo considerada en conjunto con la anterior, denominada Redes de Teleprocesamiento y Comunicación de Datos.

Un tema de singular importancia en el campo de la informática tiene que ver con el almacenamiento y recuperación de los datos en los sistemas de cómputo, y la combinación de técnicas para obtener sistemas de almacenamiento masivos centralizados que puedan dar la apariencia de tener una gran cantidad de organizaciones apropiadas para varias aplicaciones. Este tipo de problemas se estudian en la rama de la informática que se ha dado en llamar Bases de Datos.

Las tareas que se exigen a las máquinas requieren cada vez más inteligencia, y es por ello que la informática ha recurrido al estudio de la inteligencia humana como guía. La rama de la informática, conocida como Inteligencia Artificial, se basa en gran medida en investigaciones de campos como la psicología, la biología y la lingüística para diseñar algoritmos que imiten el razonamiento y la percepción de la mente humana.

### Lenguajes de programación.

Todas las computadoras tienen un lenguaje nativo, es el lenguaje de máquina.

Los lenguajes de máquina son diferentes para cada una dependiendo del fabricante de su unidad central de proceso (CPU).

Los lenguajes de máquina son instrucciones y direcciones de memoria expresadas en binario (cadenas de ceros y unos). Debido a la dificultad de programar con este tipo de instrucciones, se comenzaron a traducir esas cadenas binarias a numeración hexadecimal, lo que hacía la tarea de codificar, un poco más amigable.

Con el paso del tiempo comenzaron a surgir los primeros programas ensambladores, que permitían al programador usar una serie de instrucciones más comprensibles y fáciles de leer para el ser humano.

La programación en lenguaje ensamblador se considera de bajo nivel, esto quiere decir que los programadores piensen a un nivel cercano al de la máquina para la cual programan y sus códigos deben incluir un gran nivel de detalle.

Los lenguajes de alto nivel están más cerca del lenguaje humano que del lenguaje de máquina. Se crearon a principios de los años 50 para simplificar y optimizar la tarea de escribir programas para computadoras.

Los lenguajes de alto nivel ocultan al programador los detalles de la arquitectura de la máquina para la cual programan y permiten que el programador utilice solo el lenguaje de alto nivel evitándole la tediosa tarea de traducir a lenguaje de máquina los algoritmos que escriba.

Además, los programas escritos en este tipo de lenguajes, tiene la particularidad de ser transportables entre máquinas y podrán ejecutarse en cualquier equipo que tenga el intérprete para dicho lenguaje.

Podemos resumir que el calificativo de alto nivel, señala la independencia respecto a una computadora específica en contraposición a los lenguajes de bajo nivel.

### Traducción del código fuente.

Vimos que un programa escrito en un lenguaje de alto nivel, requiere ser traducido a lenguaje de máquina para que esta pueda entender las instrucciones y ejecutarlas.

Esta traducción pude hacerse de dos formas.

La primera es hacerlo como un COMPILADOR, traduciendo todo el programa escrito a un lenguaje que sea comprensible por la máquina.

El resultado es un programa que se puede ejecutar tantas veces como sea necesario directamente en el equipo sin necesidad de ningún otro proceso, lo que hace que la ejecución sea relativamente rápida.

La segunda manera es hacerlo mediante la interpretación línea a línea.

El proceso de traducción viene de la mano del proceso de ejecución.

### Paradigmas de la programación.

Existen diferentes formas de representar los algoritmos, dependiendo de donde serán ejecutados y de qué tipo de acciones llevará a cabo. Estas representaciones serán en forma de programa y en el lenguaje que más se adapte a la solución que necesitamos desarrollar.

También existen diferentes convenciones para la escritura de nuestros programas, estas convenciones son los paradigmas de programación.

Al día de hoy, los paradigmas utilizados para el desarrollo de programas son los siguientes:

* Programación Imperativa: lista o enunciado de órdenes.
* Programación funcional: diseño en base al uso de funciones.
* Programación orientada a objetos: se basa en la interacción de subprogramas llamados objetos que encierran datos y algoritmos para su utilización.
* Programación orientada a eventos: se basa normalmente en programación orientada a objetos, pero en términos de diseño gráfico de interfaces de usuario asociadas a eventos relacionados por ejemplo con la digitalización de teclas o movimientos del ratón.

### El lenguaje Python.

Python es especialmente recomendado para la actividad profesional pero también para el aprendizaje por su simplicidad.

Los programas en este lenguaje suelen ser bastante compactos, por lo general suelen ser más cortos que el mismo programa escrito en otros lenguajes, como por ejemplo C.

Es un lenguaje interpretado, multiplataforma y orientado a objetos.

En realidad, podemos decir que Python es multiparadigma, ya que permite la codificación orientada a objetos, en forma imperativa y en menor medida, la programación funcional.

Otro detalle no menor es la gratuidad de su intérprete que tiene versiones para casi cualquier plataforma.

Existen dos formas de ejecutar el código:

* Escribir el código en un archivo de texto y ejecutarlo.
* Escribir el código en el intérprete y obtener una respuesta línea a línea.

Para nuestro curso vamos a utilizar la primera, para lo cual nos vamos a valer del IDLE  (Integrated DeveLopment Environment) que viene con la instalación de Python.

Python tiene muchas aplicaciones gracias a su simplicidad de escritura.

Por ejemplo, en el campo de la inteligencia artificial, ya que permite plasmar ideas complejas mediante un código simple y sencillo de escribir y mantener. Es muy utilizado para los desarrollos de learning machine.

Otra rama que hace uso de las bondades de Python es Big Data.

Hay muchas librerías ya desarrolladas que permiten a los profesionales el análisis de grandes volúmenes de información y facilita las tareas de manipulación de los mismos.

También se destaca en el proceso de datos matriciales mediante el uso de gráficos y funciones de estadística. La ciencia del dato, cada vez más difundida y utilizada en grandes empresas que manejan un volumen considerable de información.

En el campo de las aplicaciones web, Python permite desarrollar mucho más con menos líneas de código. Se crean prototipos funcionales de forma más eficiente.

El framework Django, por ejemplo, permite crear aplicaciones web dinámicas y muy seguras.

También se destacan el desarrollo de juegos y la visualización de datos, en general, las aplicaciones prácticas de Python son numerosas.

### Descarga e instalación.

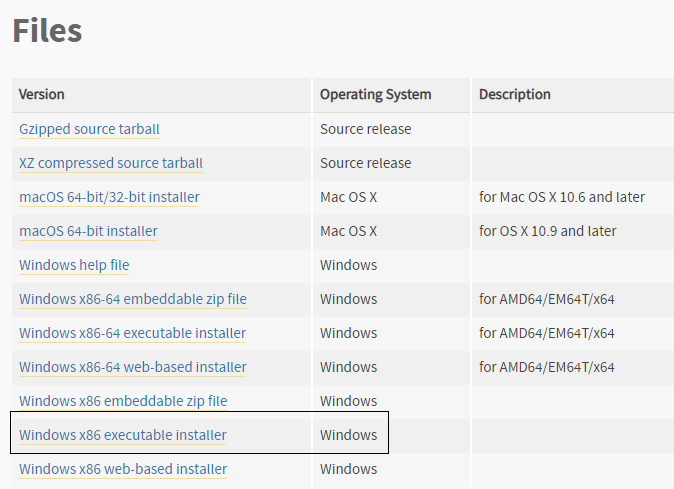
Desde la web oficial <https://www.python.org/> ingresamos en la sección de descargas y seleccionamos el archivo que queremos instalar.

Para el curso vamos a utilizar la versión 3.7.2



Una vez que seleccionamos la versión, veremos los archivos disponibles para descargar.

En nuestro caso seleccionaremos el ejecutable que corresponda a la versión del sistema operativo que tengamos.



Una vez descargado, lo ejecutamos y seguimos los pasos de instalación.

### Estructura de un programa en Python.

La siguiente, es una estructura de programa sugerida y muy utilizada.

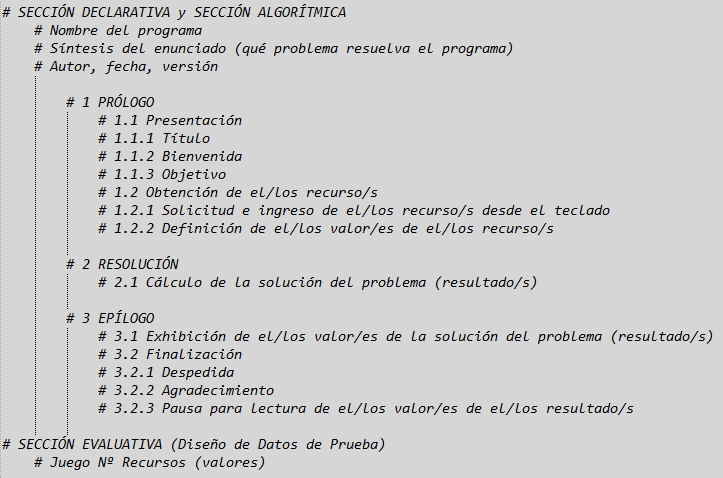
El hecho de que sea sugerida, hace que no sea obligatorio su uso ni afecte el normal desarrollo de nuestro programa, pero es necesario conocerla ya que la mayoría de los desarrollos en equipo a los que tengamos acceso suelen utilizarla y de conocer su existencia, se nos haría muy difícil editar el código o participar en algún punto del ciclo de vida del desarrollo.

En Python, se considera un comentario a todo lo que coloquemos a continuación del carácter # y hasta el final de la línea.

Básicamente dividimos nuestro programa en dos secciones, la Declarativa-Algorítmica y la Evaluativa.

En la primera definimos los recursos y desarrollamos el algoritmo y en la segunda, definimos y declaramos los valores de prueba del programa.

A su vez, la primera sección la dividiremos en Prologo (definimos recursos y pedimos datos al usuario), Resolución (contiene el núcleo del algoritmo, la solución en sí del problema) y el Epílogo (por lo general se usa para presentar los resultados al usuario).

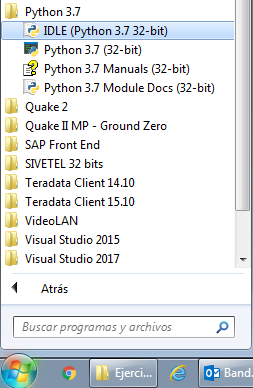


### Hola mundo.

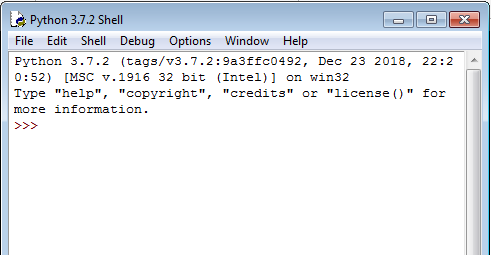
Vamos a escribir el conocido “Hola mundo” para entender el funcionamiento básico del editor que viene en la instalación de Python.

Para simplificar la lectura de los códigos de ejemplo, no vamos a utilizar la estructura sugerida en nuestros ejemplos.

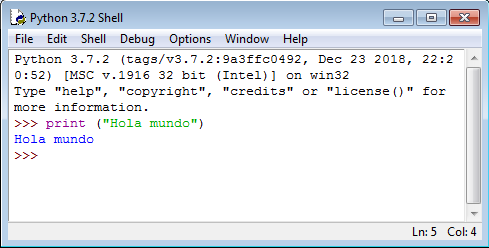
Desde el botón de inicio de Windows, buscamos el ejecutable de nuestro editor:



Una vez seleccionado el editor, se nos va a abrir la ventana de trabajo:



Podemos escribir y probar nuestras sentencias a continuación del símbolo >>> (ENTER para ejecutar)



O podemos crear un nuevo archivo y guardar nuestro código para reutilizarlo más tarde.

FILE > NEW FILE

Se nos va a abrir otra ventana en la cual vamos a escribir y guardar nuestro código.

Podemos cerrar nuestro archivo y volver a abrirlo o podemos ejecutarlo directamente pulsando F5.

Este es el funcionamiento básico del editor, a partir de acá comenzaremos a escribir y guardar nuestros programas mientras aprendemos la sintaxis de Python.

### Tipos de datos.

Todos los valores de datos en Python son objetos, y cada uno de estos objetos tiene un tipo.

También existe el tipo de dato “objeto” que acepta cualquier objeto como argumento y devuelve el mismo tipo de objeto cargado.

También se pueden crear tipos de datos definidos por el usuario.

Se dividen en dos grupos:

* Mutable: su contenido puede cambiar en tiempo de ejecución.
* Inmutable: su contenido no puede cambiar en tiempo de ejecución.

### Números enteros.

Los números enteros son aquellos que no tienen decimales, pueden ser positivos o negativos, además del cero. Pueden ser enteros comunes o largos, dependiendo de qué tan grande sea.

En Python no es necesario declarar el tipo de dato en la variable, el mismo intérprete se encarga de hacerlo al asignarles nosotros un valor.

A diferencia de Java, el tamaño del valor que puedo almacenar, depende de la plataforma, es decir que podemos almacenar un entero más grande en plataformas de 64 bits que el entero que almacenamos en plataformas de 32 bits.

Veamos un ejemplo de cómo almacenar un número entero:

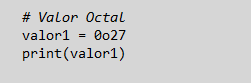


Hasta la versión 2 de Python, teníamos también el tipo de número, entero largo, el cual se debía indicar agregando la letra L al final del valor.

En la versión 3 ese tipo desapareció y para todo, usamos el tipo entero.

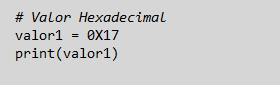
Además de trabajar con números decimales (base 10), también podemos trabajar con octales (base 8) y valores hexadecimales.

Para indicarle al intérprete que vamos a cargar un valor en octal, debemos anteponer “0o” o “0O” (en ambos casos es cero-letra o).



Lo que veremos por pantalla es el equivalente decimal del octal 27 (es el número 23).

De manera similar procedemos para trabajar con valores hexadecimales, anteponiendo “0x” o “0X” al valor que deseamos cargar (en ambos casos es cero-letra x).

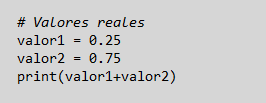


Al ejecutarlo, nos devuelve el valor 23, que es el equivalente decimal al valor hexadecimal 17.

### Coma flotante.

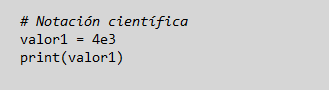
Los números reales son los que tienen una parte entera y otra parte decimal, en Python se usa el punto para separar ambas partes.

Ejemplo:



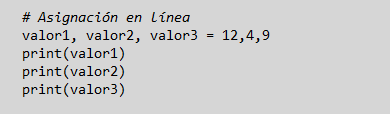
Al ejecutar el programa, veremos por pantalla el resultado de la suma, 1.0 (también en formato coma flotante).

También se puede utilizar notación científica, añadiendo una e (de exponente) para indicar un exponente en base 10, por ejemplo:



Lo que quisimos representa es 4x10 elevado al cubo, al ejecutar el programa, veremos el resultado 4000.

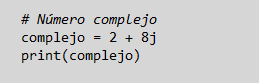
Podemos asignar valor a más de una variable en una sola línea:



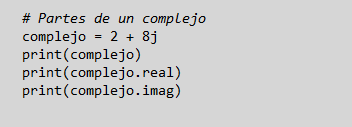
### Números complejos

Python nos permite trabajar con números complejos, que son aquellos que tienen una parte real y otra imaginaria.

Internamente se almacenan en formato de coma flotante, esta es la sintaxis:



También podemos trabajar por separado con la parte real o con la imaginaria:



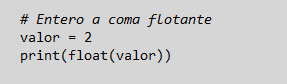
### Conversiones.

Podemos realizar conversiones entre los diferentes tipos de números que vimos hasta ahora. Por ejemplo, si tengo un número entero y lo quiero pasar a complejo.

Para esto, vamos a utilizar las funciones int( ), float( ) y complex( ).

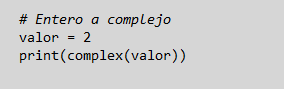
Veamos algunos ejemplos:

Ejemplo 1)



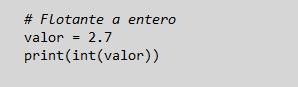
En pantalla, luego de ejecutar, veremos el resultado de la conversión, en este caso, el valor 2.0

Ejemplo 2)



En pantalla veremos el valor (2 + 0j)

Ejemplo 3)



En el ejemplo 3, lo que hacemos es convertir el número con coma flotante a número entero.

Básicamente, lo que hace Python es eliminar la parte decimal.

### Booleanos.

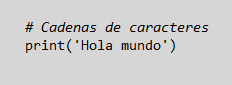
El tipo booleano solo puede tener dos valores, TRUE o FALSE.

Es muy utilizado en las estructuras de decisión para comprobar la veracidad o no de una condición.

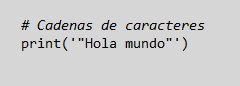
Los vamos a ver más en detalle cuando veamos este tipo de estructuras.

Cadena de caracteres (str).

Son secuencias inmutables que contienen caracteres encerrados entre comillas que pueden ser simples o dobles.

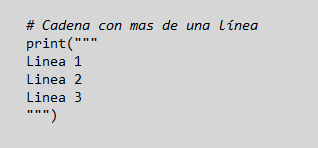


Si queremos agregar comillas a nuestro texto, lo podemos hacer utilizando el otro tipo de comillas. Por ejemplo, si queremos ponerle comillas a nuestro Hola mundo (que lo escribimos con comillas simples) lo que deberemos hacer es usar comillas dobles de la siguiente manera:

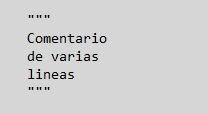


Y veremos por pantalla el mensaje: “Hola mundo”.

También podemos mostrar por pantalla, cadenas con más de una línea de la siguiente manera:

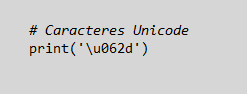


También podemos utilizar las comillas triples para hacer comentarios de varias líneas:



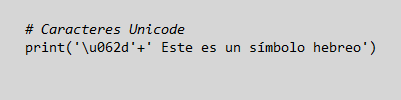
### Caracteres Unicode.

Hasta ahora vimos cadenas con caracteres ASCII, pero también podemos mostrar por pantalla caracteres Unicode escribiendo \u y el código del carácter, por ejemplo:



Podemos ver la lista de caracteres Unicode en: <https://unicode-table.com/es/>

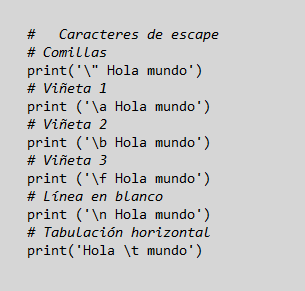
Podemos combinar estos caracteres con los caracteres tradicionales:



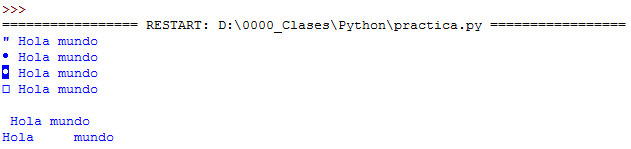
Caracteres de escape.

Al igual que en la mayoría de los lenguajes de programación, en Python también disponemos de caracteres especiales.

Algunos ejemplos:



Si escribimos y ejecutamos el código anterior, veremos el siguiente resultado:



Para el primer caso, pudimos introducir una comilla doble dentro de la cadena, anteriormente habíamos visto otra forma de hacerlo. Esto también sirve para introducir comillas simples.

Luego vienen tres ejemplos de cómo hacer viñetas y por último, como dejar una línea en blanco sobre nuestra cadena de caracteres y como aplicar la tabulación.

Operadores aritméticos.

Estos son los operadores aritméticos que tenemos disponibles en Python.

Podemos probar los ejemplos directamente en el editor.

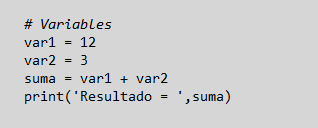
|  |  |  |  |
| --- | --- | --- | --- |
| **Operador** | **Nombre** | **Ejemplo** | **Resultado** |
| + | Suma | 15 + 20 | 35 |
| - | Resta | 15 - 7 | 8 |
| - | Negativo | -5 | -5 |
| \* | Multiplicación | 8 \* 4 | 32 |
| \*\* | Exponente | 3 \*\* 2 | 9 |
| / | División | 24 / 3 | 8 |
| // | División entera | 17 // 2 | 7 |
| % | Resto | 29 % 3 | 2 |

Variables.

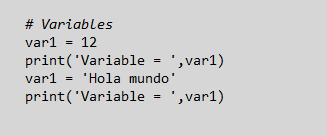
Para definir una variable, debemos asignarle un nombre (identificador) y un valor.

No es necesario declarar el tipo de variable que queremos crear, esto se define al momento de asignarle un valor.

Cuando declaramos una variable, lo que hace Python es reservar una porción de memoria y relacionarlo con el identificador de la variable que acabamos de declarar.



Podemos, dentro de un mismo programa, cambiar el valor de una variable e incluso el tipo de dato que se le carga.



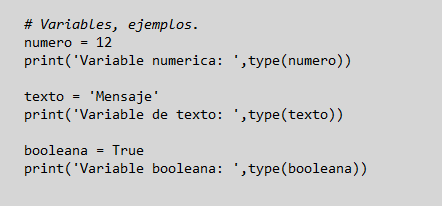
Aunque esto está permitido, se recomienda siempre dar a las variables, un nombre que nos ayude a recordar a simple vista que valor estamos guardando en ella.

Las variables en Python son, por defecto, variables locales.

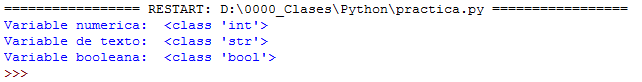
Esto quiere decir que si definimos una variable dentro de una porción de código, como por ejemplo, una función, esta solo existirá dentro de la función y no serán accesibles desde otra parte del programa.

De igual manera, toda variable declarada fuera de una función, no será visible dentro de la misma.

Podemos ver el tipo de variable o tipo de dato, mediante la función TYPE. Veamos como declarar y asignar valores a diferentes tipos de variable y mostrar por pantalla que tipo es.



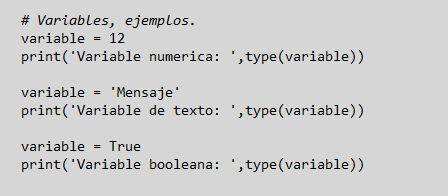
Este es el resultado que deberíamos ver por pantalla:



Para el caso de las variable booleanas, ya sea que le asignemos el valor True o False, debemos tener en cuenta que la primer letra siempre va en mayúscula.

¿Podemos asignarle un tipo diferente de dato a la misma variable durante la ejecución del programa?

La respuesta es SI.

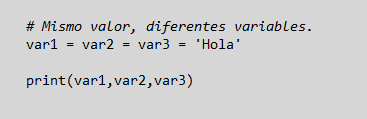


En este caso obtendremos el mismo resultado que con el ejemplo anterior, esta vez utilizando la misma variable.

Asignación simultánea.

Podemos, en una misma línea de código, asignar el mismo valor a diferentes variables o diferentes valores a diferentes variables.

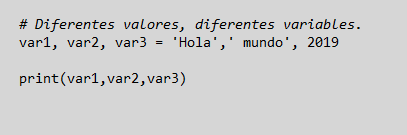
Vemos como se hace:



Para este primer caso, este es el resultado que veremos por pantalla al mostrar el contenido de las tres variables:



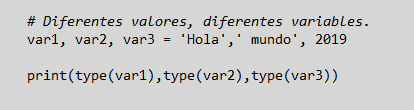
Veamos ahora como asignar diferentes valores a diferentes variables en una sola línea de código:



Esto es lo que deberíamos ver por pantalla al ejecutar el código:



También podemos ver el tipo asignado a cada variable, como ya vimos, mediante el uso de TYPE:



Identificadores.

Los nombres que damos a las variables se denominan identificadores. El nombre de una variable, es su identificador.

Python, al igual que la mayoría de los lenguajes de programación, tiene reglas específicas respecto a los identificadores.

Un identificador puede comenzar con una letra (mayúscula o minúscula) o con un guion bajo y luego continua con una serie de caracteres (letras mayúsculas o minúsculas con y sin tilde, números y guiones bajos).

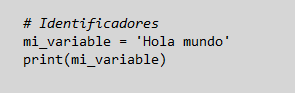
Los espacios no están permitidos, y el intérprete distingue entre mayúsculas y minúsculas.

Por ejemplo, las identificadores Nombre, nombre y NOMBRE, son diferentes para Python.

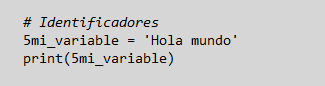
Tampoco se permiten signos de exclamación o caracteres especiales como @.

En general, los identificadores que comienzan con uno o dos guiones bajos, se reservan para tipos especiales que veremos más adelante.

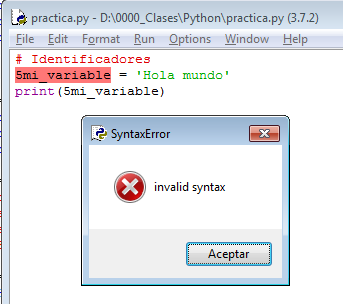
Veamos un ejemplo de identificador válido:



Ahora, un ejemplo de identificador no válido:



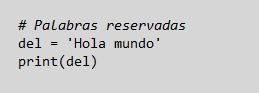
Si ejecutamos el código del ejemplo anterior, el intérprete Python nos dará un mensaje de error:



Palabras reservadas.

Python tiene palabras que, a pesar de ser identificadores válidos, no pueden ser usadas.

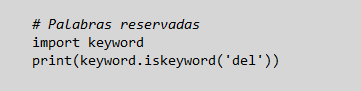
Son palabras reservadas para uso del intérprete, por ejemplo DEL.



Si ejecutamos el ejemplo anterior, el intérprete nos dará un mensaje de error.

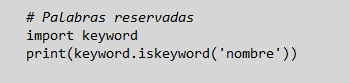
Podemos verificar si una palabra es reservada o no, importando, es decir, incorporando a nuestro programa, el módulo KEYWORD que viene con la instalación de Python.

Por ejemplo, si queremos consultar acerca de si la palabra DEL es reservada o no, procedemos de la siguiente forma:



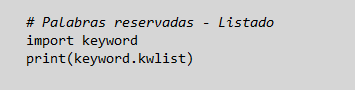
Como DEL es una palabra reservada, por pantalla veremos impresa la palabra TRUE.

Probemos ahora con una palabra que no sea reservada, por ejemplo, NOMBRE:



En este caso, veremos por pantalla el resultado FALSE ya que la palabra NOMBRE, no es una palabra reservada.

También podemos ver la lista completa de palabras reservadas de nuestra versión de Python:



Al ejecutar el código del ejemplo, veremos por pantalla el listado completo de palabras reservadas de nuestra versión de Python.

|  |
| --- |
| Mi primer programa. |

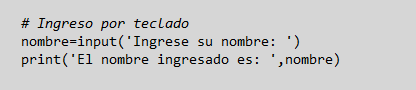
Vamos a hacer un programa simple para reforzar los conceptos vistos hasta ahora.

Ejercicio a) Desarrollar un programa que solicite a un usuario el ingreso por teclado el valor del radio de un círculo y muestre por pantalla la superficie de dicho círculo (PI por radio al cuadrado).

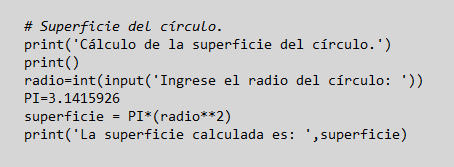
Considerar que el usuario no comete errores al ingresar los datos.

Solución.

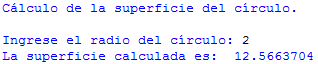
La lectura de datos por teclado se hace mediante INPUT con la siguiente sintaxis de ejemplo:



Una solución simple para el ejercicio propuesto es la siguiente:



Al ejecutarlo veremos por pantalla lo siguiente:



En el input siempre tendremos una cadena de texto, por más que el valor ingresado sea un número, por lo tanto, antes de asignar dicho valor a una variable, deberemos convertirlo, en este caso, lo pasamos a entero (tipo INT).

\*\*\* Modificar el código para que el programa acepte que el usuario ingrese un número real en lugar de un número entero.

Nota: si queremos que luego de mostrar una frase por pantalla, el cursor no cambie de línea, deberemos agregar, dentro de la misma sentencia PRINT, separado por coma, lo siguiente: end=’’