**Clase 02. Introducción a la programación con Python**

# Definición de programa. Lenguaje interpretado vs. compilado. Python como lenguaje.

## ¿Qué es la programación? ¿Qué es un programa?

La programación como concepto surge de otro más antiguo, que es el de algoritmo. Un algoritmo es una secuencia de pasos lógicos para llegar a un resultado. Un código de programación es un algoritmo. Una receta de cocina también es un algoritmo. Si el algoritmo se sigue de forma taxativa, se llega al resultado deseado, ya sea un software con un propósito específico, una app para el celular, o una pizza.

La programación surge como complemento necesario para el ordenador o computadora. La computadora en sus orígenes es concebida como un artefacto para acelerar determinado tipo de cálculos. Los cálculos alcanzados por la computadora fueron creciendo y complejizándose cada vez más, en conjunto con los desarrollos que hicieron que las computadoras funcionen más rápidamente, en un espacio más reducido y consumiendo menos recursos.

Un programa es por lo tanto una forma de que la computadora entienda el funcionamiento de un algoritmo. En su forma más básica, una computadora sólo entiende secuencias de ceros y unos. Pero nosotros no hablamos ese idioma. Es entonces necesario buscar una forma de transmitir los algoritmos que diseñamos los humanos (en nuestra forma de comunicarnos) al lenguaje que entiende la computadora, que por tener los dos símbolos 0 y 1 se denomina lenguaje binario.

## Programación y lenguajes

Hoy en día existen muchos lenguajes de programación. Pero, ¿por qué tienen que ser tantos? La respuesta es simple: “no existe la navaja suiza para programación”. Algunos lenguajes se comportan mejor para sistemas empresariales; otros se comportan muy bien para desarrollo Web; los sistemas de tiempo real tienen lenguajes que se adaptan mejor a sus necesidades; y, finalmente, existe un puñado de lenguajes que funcionan muy bien para Data Science. Entre ellos podemos mencionar a Python, R, Julia y Scala. Si bien tienen diferencias más o menos sutiles, todos cumplen con su propósito de ayudar al Data Scientist. Además, estos cuatro lenguajes aparecen entre los más populares en las búsquedas laborales en LinkedIn. En nuestra clase vamos a trabajar con Python, porque lo consideramos como un lenguaje que no puede faltar en la preparación de un buen CV / Portfolio, y además tiene una curva de aprendizaje no tan empinada si lo que queremos es hacer Data Science.

## Lenguaje interpretado vs. compilado

Para entender cómo vamos a utilizar Python, veamos antes algunas de sus cualidades. En primer lugar Python es un lenguaje interpretado. Esto significa que en todo momento tenemos un programa “intérprete” que traduce en tiempo casi real cada uno de nuestros comandos, línea por línea, al código binario de la computadora. Esto no siempre es así. Otros lenguajes (que se denominan compilados) se traducen a binario solamente al final, una vez que se escribieron por completo.

Pensemos por un segundo en esta diferencia: los lenguajes interpretados permiten un uso interactivo. Esto significa que podemos probar el código “de a pedacitos”, sin tener que verificar que esté completo.

## Python como lenguaje

Veamos ahora a Python con un poco más de detalle. Python es un lenguaje open source, lo que significa que podemos trabajar libremente con él. Una de sus características principales es su simplicidad. Que un lenguaje sea simple significa que es fácil de compartir con otros programadores, y de esta forma lograr trabajar de forma colaborativa.

Para tener una instalación completa de Python con la que podamos trabajar, necesitamos al menos tres componentes:

* El intérprete de Python. Este es un programa que tiene que estar funcionando para que podamos ejecutar el código
* Un editor de texto o IDE, necesario para que podamos escribir efectivamente el código. Veremos esto en la sección siguiente.
* Paquetes para uso específico. Los paquetes son conjuntos de funciones que vienen pre-armadas para solucionar problemas habituales.

A continuación veremos en detalle la instalación de Python para trabajar en Data Science.

# Instalación de Python vía Miniconda

De acuerdo a los componentes mencionados en la sección anterior, hay varias formas de instalar Python. Mencionaremos aquí una de las más simples que se utilizan habitualmente en Data Science: Miniconda.

Miniconda es un instalador que ofrece el intérprete de Python, junto con una herramienta llamada conda, muy útil para instalar paquetes adicionales. Puede descargarse a partir de [este enlace](https://docs.conda.io/en/latest/miniconda.html). Tener en cuenta que hay dos grandes versiones de Python que coexisten: La versión 2 y la versión 3. En este curso utilizaremos la versión 3.

Una vez instalado, Miniconda puede utilizarse de varias maneras. La más simple es iniciarlo como programa, y abrirá una terminal. Aquí escribimos python y entramos directamente al intérprete, donde podemos trabajar sin problemas.

## Python tradicional

Tradicionalmente, Python puede desarrollarse en [Anaconda](https://www.anaconda.com/products/individual), o en su versión simplificada, [Miniconda](https://docs.conda.io/en/latest/miniconda.html). A su vez, puede utilizarse de **varias formas más:**

* La forma más básica es escribiendo python en la terminal, lo que abre un entorno de trabajo dentro de la misma terminal.  
    
   📢No es la forma más cómoda, ni la más utilizada.
* Otra forma más útil es usando Python interactivo (IPython). Puede accederse escribiendo ipython en la terminal.  
    
  📢 No aporta muchas mejoras si se usa de esa forma.

**¡No siempre es la mejor forma!**

## Python interactivo

Otra manera de utilizar Python es a través de su versión interactiva, llamada IPython. La diferencia con la forma “tradicional” es que esta IPython está mucho más orientada al uso de pequeños bloques de código, permitiendo realizar pequeñas pruebas y facilitando la colaboración entre programadores. No es que una forma sea mejor que otra, cada una tiene sus ventajas. Más allá de esto, la forma interactiva es una de las más populares para Data Science. En este curso utilizaremos la forma interactiva.

Para iniciar IPython, abramos Miniconda y escribamos ipython. Aquí seguimos en la terminal, pero cada línea de código está marcada con un número. Esta es una de las diferencias de la forma interactiva.

## IPython y notebooks

Hasta aquí seguimos trabajando en la terminal, que no es la forma más cómoda de programar. Veamos esto: acabamos de instalar el intérprete, y hasta el momento tenemos dos formas de accederlo y utilizarlo. Probemos con una tercera forma, que es la más utilizada para Data Science. Sigue siendo IPython, pero veremos que la forma de trabajar con el código es mucho mejor. Esta forma incluye la utilización de gráficos de una forma muy práctica e intuitiva.

Abramos Miniconda y ejecutemos el siguiente código:

| conda install -c conda-forge jupyterlab notebook |
| --- |

Veremos el progreso de la instalación en la terminal.

A continuación, escribamos en Miniconda

| conda install -c conda-forge jupyterlab notebook |
| --- |

Veremos que se abre una pestaña de nuestro navegador, mostrándonos una lista de archivos.

Esta forma de trabajo se denomina programación con “notebooks”. Nuevamente, sigue siendo IPython, pero aquí en vez de trabajar mediante una terminal estamos usando el navegador. Desde el navegador podremos escribir código, que pasará por el intérprete (que ya tenemos instalado en la computadora gracias a Miniconda), y finalmente podremos ver los resultados en el mismo navegador.

Cada conjunto de líneas de código se escribe en una caja de texto. Puede trabajarse con varias cajas de texto al mismo tiempo. Estas cajas de texto pueden ejecutarse al modo interactivo de una o todas juntas. El conjunto completo de cajas de texto constituyen una notebook.

En resumen, el proceso de instalación de Python funciona de la siguiente manera:

* Instalación de Miniconda con Python 3
* Instalación de los programas Jupyter y Notebook desde Miniconda
* Utilización de IPython en el navegador por medio de notebooks

Esta es una de las configuraciones más utilizadas y más solicitadas en las búsquedas laborales.

## Una opción más con una variante híbrida

En casos donde queremos probar algo rápido, o plantear una idea en ámbitos más didácticos que empresariales, Google ofrece la interfaz Colaboratory en <https://colab.research.google.com/> . Sólo hay que contar con una cuenta de Google. Una vez que accedemos a la página, tenemos a disposición una notebook en cuestión de segundos. Por defecto, esta opción aloja toda la información en la nube. Esta forma de trabajo es muy práctica, pero puede tener limitaciones a la hora de trabajar profesionalmente.

Finalmente, con algunas configuraciones adicionales, puede utilizarse la interfaz de Colaboratory con el entorno de ejecución local. Esta opción híbrida puede servir para los que no se terminan de acostumbrar al entorno de Jupyter notebooks.

# ¿Cómo usar Google Collab?

Permite trabajar en un entorno no local y la creación de Notebooks 🚀

Es un producto de Google Research. Está especialmente adecuado para tareas de aprendizaje automático, análisis de datos y educación.

Jupyter es el proyecto de código abierto en el que se basa Colab.

Nos permite compartir notebooks sin la necesidad de descargar ningún software extra.

El código se ejecuta en una máquina virtual dedicada a tu cuenta y pueden eliminarse luego de cierto tiempo.

# IDE’s

Son **aplicaciones de software** que permiten a programadores **desarrollar código** en diferentes lenguajes.

Consta, usualmente, de:

1. Editor de código.
2. Depuradores (Debuggers) que permiten encontrar errores en el código
3. Herramientas automáticas

IDE’s para desarrollo de Python

Permite trabajar en un entorno no local y la creación de Notebooks 🚀

Las herramientas que mostramos anteriormente no son las únicas en donde compilar código de Python…  


# Nociones básicas de programación

## Variable, asignación, expresiones

En esta sección veremos los bloques conceptuales más básicos de la programación con Python.

Pensemos en la siguiente expresión matemática: x + y = z

En esta expresión queremos representar que la suma de los números x e y dará como resultado el número z.

Pensemos ahora esto en términos de programación: si quisiéramos hacer un programa que realice una suma, podemos plantearlo de la siguiente manera:

* Ingresar un número, que cumplirá el rol de la x
* Ingresar otro número, que cumplirá el rol de la y
* Decirle a la computadora que efectúe la suma de los dos números
* Al resultado lo haremos cumplir el rol de la z
* Mostrar al usuario el valor de z, esto es, el resultado de sumar los dos números ingresados x e y.

Todo este proceso está pensado detalladamente para programación.

Toda vez que tenemos un símbolo al que le damos un valor concreto (por ejemplo x = 2), estamos haciendo una asignación a una variable. El símbolo x es una variable porque puede tomar cualquier valor. En este caso particular le damos (mediante el proceso de asignación) el valor 2.

En Python podemos trabajarlo de la siguiente manera:

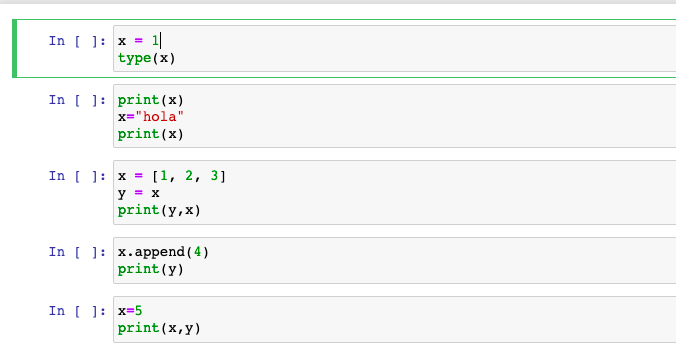
| x = 2 y = 3 z = x + y z 5 |
| --- |

La primera línea asigna a la variable x el valor 2. La segunda línea asigna a la variable y el valor 3. La tercera línea suma x e y. La cuarta línea muestra el resultado.

Cada parte del código “compuesta” que represente un valor en concreto (por ejemplo x + y) es una expresión. En principio, podemos hacer la analogía con las expresiones matemáticas sin inconvenientes.

### Ejercicios

¿Cuáles son las salidas de los siguientes bloques de código?



## Objetos y punteros

En Python todo es un objeto. Un objeto, en términos de programación, es una entidad que tiene datos y además tiene metadatos (atributos o propiedades) y funcionalidad propia (métodos). Consideremos el siguiente caso:

| x=1.0 x.is\_integer()  x=1.4 x.is\_integer() |
| --- |

En estos casos, is\_integer() es una funcionalidad de x que dice si es entero o no

| print(x.real,x.imag) |
| --- |

Aquí, real e imag son propiedades de x que muestran su parte real y su parte imaginaria.

Por otra parte, el ejemplo de la sección anterior nos muestra la forma de trabajar con punteros (pointers). A diferencia de otros lenguajes donde las variables se comportan como contenedores de datos, las variables en Python son punteros. Esto significa que no contienen un valor, sino que apuntan a un dato que se encuentra en la memoria. De esta forma, si asignamos una variable a otra (y = x), simplemente le estamos diciendo a ambas que apunten al mismo dato (la lista [1, 2, 3]). En su forma más específica, la operación “=” realiza la conexión entre el puntero y el dato. En cambio, la aplicación de métodos (por ejemplo append()) cambia el dato de destino.

Esta diferencia es sutil y en la práctica no afecta el trabajo de Data Science. No obstante trae mejoras sustanciales en la performance y es importante tenerla en cuenta para no cometer errores en la programación.

## Expresiones

Una expresión es una combinación de operadores y operandos que se interpreta para producir algún otro valor.

En cualquier lenguaje de programación, una expresión se evalúa según la precedencia de sus operadores.

* Expresiones constantes: son las expresiones que solo tienen valores constantes.
* Expresiones aritméticas: una expresión aritmética es una combinación de valores numéricos, operadores y, a veces, paréntesis.
* Expresiones integrales: este es el tipo de expresiones que producen solo resultados enteros después de todos los cálculos.
* Expresiones flotantes: este es el tipo de expresiones que producen números de punto flotante como resultado de todos los cálculos

## Operaciones

Python define una serie de operaciones básicas. Estas operaciones se muestran a continuación:

Las operaciones aritméticas son las más básicas. Prestar atención a algunas específicas que pueden ser de mucha utilidad.

| Aritméticas |  |
| --- | --- |
| a + b | Suma |
| a - b | Resta |
| a \* b | Multiplicación |
| a / b | División |
| a // b | División entera (resultado sin decimal) |
| a % b | Módulo (resto de la división entera) |
| a \*\* b | Exponenciación |
| -a | Negativo |

Además del operador de asignación “=”, existen los siguientes, que son equivalentes a las operaciones de la columna de la derecha, y nos pueden hacer ahorrar mucho tiempo.

| Asignaciones |  |
| --- | --- |
| a += b | a = a + b |
| a -= b | a = a - b |
| a \*= b | a = a \* b |
| a /= b | a = a / b |
| a //= b | a = a // b |
| a %= b | a = a % b |
| a \*\*= b | a = a \*\* b |

Los comparadores sirven para, precisamente, comparar objetos. Se comportan de manera análoga a los correspondientes símbolos matemáticos que representan.

| Comparadores |  |
| --- | --- |
| a == b | a igual a b |
| a != b | a distinto de b |
| a < b | a menor a b |
| a > b | a mayor a b |
| a <= b | a menor o igual que b |
| a >= b | a mayor o igual que b |

Los operadores de identidad van más al fondo del detalle de los objetos y punteros. Si dos objetos son el mismo (operador “is”), significa que apuntan al mismo dato. Por otra parte, si queremos saber si un dato está contenido en otro, no importa en qué parte, nos será de mucha utilidad el operador “in”. Esto último se usa frecuentemente en las listas de datos.

| **Identidad y pertenencia** |  |
| --- | --- |
| a is b | a es el mismo objeto que b |
| a is not b | a no es el mismo objeto que b |
| a in b | a está contenido en b |
| a not in b | a no está contenido en b |

Glosario

**Programación**: formas de ejecutar un algoritmo (recetas)

**Lenguajes**: herramientas computacionales que permiten resolver problemas con estructuras de código. En Data Science existen varios comunes: Python , R , Java, Julia, C, C++

**Lenguaje interpretado**: cualquier lenguaje de programación que se ejecute línea a línea y que convierta las órdenes a formato binario (e.g Python , R)

**IDE**: aplicaciones donde escribimos el código de un lenguaje particular (e.g Spyder, Kite, Visual Studio, Atom)

**Variable**: Cualquier estructura que permita almacenar información para su manipulación

**Asignación**: Proceso mediante el cual se le asigna un valor particular a una variable

**Punteros**: herramientas que nos permiten conectar a las variables con sus valores respectivos

**Expresiones**: combinaciones de operadores y operandos que dan como resultado un valor particular

**Operadores**: son los que permiten trabajar sobre las variables, pueden ser de 4 tipos (aritméticos, relacionales, de asignación y lógicos)