**Contenido**

Clase 1

# Modulo 1: Introducción a Python

## Preparación

### ¿Qué es un lenguaje de programación?

Un lenguaje de programación es un conjunto de reglas (sintaxis específicamente) en la cual se les dan instrucciones a dispositivos electrónicos (celulares, maquinas, computadoras, etc.)

Por su manera de funcionar, hay principalmente 2 tipos de lenguaje de programación

1. Compilado
   1. Este tipo de lenguaje es previamente “traducido” a código de máquina.
   2. Tiene un menor tiempo de ejecución\*
      1. Esta brecha se ha disminuido
   3. Dan mayor prioridad a la eficiencia que a soporte de plataformas cruzadas
   4. Provee un control más fino del CPU y uso de memoria "Lenguaje de programación compilado - Wikipedia, la enciclopedia libre." <https://es.wikipedia.org/wiki/Lenguaje_de_programaci%C3%B3n_compilado>.
2. Interpretado
   1. Ejecuta las instrucciones directamente. Es decir “traduce” en tiempo real
   2. Son más fáciles de evolucionar durante el desarrollo y la ejecución.
   3. Es independiente de la plataforma utilizada
   4. Es más sencillo realizar la depuración "Intérprete (informática) - Wikipedia, la enciclopedia libre." <https://es.wikipedia.org/wiki/Int%C3%A9rprete_(inform%C3%A1tica)>.

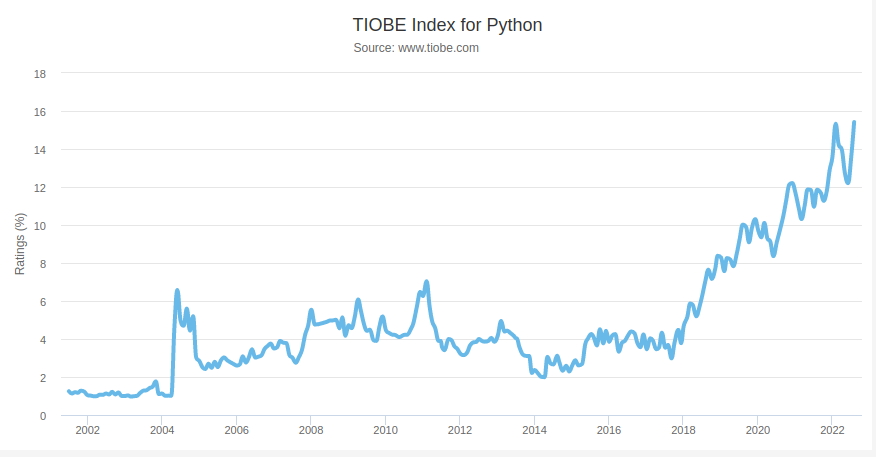
En este curso específicamente se trabajará utilizaron Python 3.

Historia:

1. Este lenguaje fue creado a finales de 1980s y su primera implementación fue en diciembre de 1989 por Guido Van Rossum
2. El nombre “Python” viene del programa televisivo BBC “*Monty Python’s Flying Circus”*
3. Python 3.0 fue lanzada el 3 de diciembre de 2008.
   1. NO Es compatible con versiones anteriores. Ósea que código programado en Python 1.0 o 2.0 no corre en Python 3.0 y viceversa
   2. Fue diseñado con la filosofía de: reducir la duplicación de características eliminando viejas formas de hacer las cosas.
   3. Características importantes de esta versión
      1. Print se hizo una función
      2. Las entradas en input, son automáticamente alfanuméricas
      3. Introducción de división entera
4. (al 18 de agosto del 2022) la última versión es Python3.10.6

("History of Python - Wikipedia." <https://en.wikipedia.org/wiki/History_of_Python>.)

De acuerdo con el TIOBE index, en agosto del 2022, Python ocupa 15.42% del mercado. Fue considerado el lenguaje del año en el 2007, 2010, 2018, 2020, 2021.

"Python Language of Year 2021 - TIOBE." 09 Febrero. 2022, <https://www.tiobe.com/news/item/python-language-of-year-2021/>.

### Similitudes y diferencias con otros lenguajes de programación

|  |  |
| --- | --- |
| Similitudes | Diferencias |
| * Es ingles * Puede ser orientado para múltiples aplicaciones   + Búsqueda de información   + Programación de páginas web, aplicaciones móviles,   + Generación de graficas   + Control de equipo y maquinaria\* | * Es un lenguaje interpretado   + Es más lento que un lenguaje compilado * Es un lenguaje tabulado   + No necesitas separadores especiales ({}/ ;)   + No necesita la creación de espacios especiales     - Secciones de setup o loop |

### Instalación de Python para windows

Descarga del lenguaje de Python: <https://www.python.org/downloads/>

Durante la descarga de Python, seleccionar la casilla de “add to Path” en la parte inferior de la ventana.

Descarga del compilador: <https://code.visualstudio.com/download>

Después de finalizada la descarga e instalación, agregar la extensión de Python

Presentar partes de vscode

1. Editor de texto
2. Consola
3. Panel izquierdo
   1. Explorador de archivos
      1. Esta sección muestra los demás programas, archivos (de texto o imágenes) que están presentes en el directorio actual de trabajo.
   2. Búsqueda
      1. Un panel donde se busca palabras o frases en todos los archivos dentro del directorio actual de trabajo
   3. Control de versiones (git)
      1. En conjunto con un tercer programa. Funciona para almacenar versiones anteriores, realizar colaboraciones con otros desarrolladores y poder implementar versiones de programas tanto para produccion como desarrollo
   4. Panel de depuración
      1. Funciona para detectar errores en el programa. En el se muestra en tiempo real el valor de las variables, comportamientos de las mismas y tipo de dato almacenado, entre otros.
   5. Extensiones
      1. Se encuentran subprogramas desarrollados y hábiles para combinar con VScode y mejorar la funcionalidad y personalización del mismo.
   6. Perfil
      1. Permite la creación o enlace de una cuenta personal donde se guardan preferencias de atajos de teclado, cuentas asociadas, personalización de VScode entre otros, esta información se guarda en la nube.
   7. Ajustes
4. Panel inferior
   1. Programa/compilador
   2. Notificaciones
5. Sub-ventana
   1. Problemas
      1. En esta se describen los problemas y advertencias que impiden el funcionamiento correcto del programa
         1. Da una corta descripcion del problema y la posicion [fila y col] donde se encuentra
   2. Salida
      1. Este funciona como un registro de los eventos manejados por VSCode y las extensiones instaladas
   3. Consola de depuración
      1. Esta es la ventana con la cual se hace la depuración e interacción del programa con el usuario mediante entradas y salidas.
   4. Consola/terminal
      1. Esta se conecta con la consola de la computadora, por lo cual se pueden utilizar comandos para interactuar con los diversos elementos del pc. También se pueden hacer comandos para ejecutar acciones como creación de archivos, apagar y reiniciar el equipo
      2. Cuando corremos el programa en modo normal, es la manera de interacción con el mismo.

## Sintaxis

Conjunto de reglas que definen las secuencias correctas de los elementos de un lenguaje de programación. ("sintaxis | Definición | Diccionario de la lengua española | RAE - ASALE." <https://dle.rae.es/sintaxis>.)

Python es un lenguaje “case-sensitive”. Quiere decir que SI distingue entre mayúsculas y minúsculas. Para el programa no es igual “A” que “a”

### Código programable

Las reglas de programación en Python 3 son:

* Las palabras reservadas siempre son en ingles
  + Importe de librerías
  + Declaración de Funciones
  + Lectura y escritura de datos en consola, pantalla y/o archivos

|  |  |
| --- | --- |
| Palabra reservada | función |
| input | Solicita datos al usuario. Siempre genera valores alfanuméricos |
| print | Muestra datos al usuario mediante la consola/terminal |
| for | Un tipo de bucle que corre un determinado número de veces |
| while | Un tipo de bucle que corre en basado en condiciones |
| from ….. import  import | Importa librerías en el programa |
| def | Utilizada para crear una función |
| break | Interrumpe forzadamente el ciclo o bucle |
| continue | Reinicia/salta a la siguiente interacción del bucle, ignorando todo lo que está por debajo |
| if  else | Evalúa condiciones y realiza acciones dependiendo del resultado. Da “rutas” u “opciones” |
| int | Función para transformar datos alfanuméricos a datos numéricos enteros |
| float | Función para transformar datos alfanuméricos a datos numéricos flotante (decimales) |
| str | Función para transformar cualquier dato a datos alfanumérico |

* Para dependencias (dentro de ciclos for, while o funciones) se tabula el código inmediatamente debajo
* Los comentarios siempre deben llevar por delante el “#”

### Tipos de datos

En Python, se pueden agrupar los datos en función de su uso (aritmética o solo información), en los siguientes

* Alfanuméricos o cadena de caracteres (*string)*
  + Son letras, números y caracteres especiales que se utilizan para dar información
  + Este tipo de dato siempre debe estar entre comillas (pueden ser ambas sencillas o dobles) [no se permite combinar comillas]. Se abre y cierra con la misma comilla
* Binarios o lógicos
  + Tiene única y exclusivamente 2 valores
    - Cierto (True, 1) o Falso (False, 0)
    - Son el resultado de una comparación, utilizando los otros 2 tipos de datos (alfanuméricos y numéricos)
* Numéricos
  + Son números (enteros, decimales, racionales, irracionales, fracciones, etc..) que se utilizan para realizar operaciones matemáticas, tales como, suma, resta, multiplicación, división, etc.
  + Esta categoría se puede subdividir en
    - Enteros
      * Solo contempla los números antes del punto decimal
    - Flotantes (decimales)
      * Considera ambos números antes y después del punto decimal

### Operaciones y operadores

#### Alfanuméricos

|  |  |  |
| --- | --- | --- |
| Operación | Símbolo | ejemplo |
| Adición o concatenación | + | Hola + mundo= Hola mundo |
| Extracción | [] | Hola[0]=H  Hola[-1]=a  Hola[0:1]=Ho |

#### Lógicos

|  |  |  |  |
| --- | --- | --- | --- |
| Operación | Palabra reservada o símbolo | Ejemplo | |
| “Multiplicación” | and | |  |  |  | | --- | --- | --- | | A | B | A and B | | 0 | 0 | 0 | | 0 | 1 | 0 | | 1 | 0 | 0 | | 1 | 1 | 1 | | |
| “Adición” | or | |  |  |  | | --- | --- | --- | | A | B | A or B | | 0 | 0 | 0 | | 0 | 1 | 1 | | 1 | 0 | 1 | | 1 | 1 | 1 | | |
| Negación | not | |  |  | | --- | --- | | A | Not A | | 0 | 1 | | 1 | 0 | | |
| Comparación igual | == | “hola”== “mundo”  False | 10==10  True |
| Comparación diferente | != | “a” != “b”  True | 5!=5  False |
| Mayor que | > | 10>5  True | |
| Menor que | < | 10<5  False | |
| Mayor o igual que | >= | 4>=0  True | |
| Menor o igual que | <= | 5<=0  False | |

#### Numéricos

|  |  |  |
| --- | --- | --- |
| Operación | Símbolo | ejemplo |
| Adición | + | 1+2=3 |
| Sustracción | - | 8-1=7 |
| Multiplicación | \* | 2\*3=6 |
| División | / | 8/2=4 |
| División entera | // | 8//3=2 |
| Modulo (residuo) | % | 7%4=3 |
| Potencia (exponenciación) | \*\* | 2\*\*3=8 |

### Tabulaciones y espaciados

A diferencia de otros lenguajes de programación, en Python el código se agrupa en funciones de las tabulaciones y/o espaciados utilizados.

El código “medular” va alineado completamente a la izquierda del compilador. Para las secciones “hijas” del código, como pudieran ser opciones para *if*, secuencias dentro del *for y while*, este debe ser espaciado/tabulado. No hay límite de tabulación, pero se debe mantener uniformidad a lo largo del mismo.

Quiere decir que, si la primera línea “hija” tiene un espacio, todas las demás obligatoriamente deben conservar 1 espacio. Si se agregan espacios adicionales en líneas subsecuentes, estas serán consideradas “hijas” de las “hijas” o nuevos subprocesos o pueden ocasionar comportamientos no deseados en nuestro código.

### Comentarios

Estos no son parte fundamental del código ni tampoco afectan en ninguna manera la ejecución de este. Pero funcionan para documentar, o en momentos de pruebas de funcionamiento, saltar líneas de códigos que no se desean utilizar en el momento sin tener que borrarlas.

## Lectura y escritura de datos

Una de las interacciones y usos principales de los programas es que requiere interactuar con datos. Pero al momento de escribir nuestro programa, no se almacena en el código todos los posibles datos a utilizar, sino que depende de los datos guardados en base de datos, archivos, el internet, o simplemente aquellos provistos por los usuarios u otros programas.

La lectura de datos es aquella que recibe información que posteriormente será utilizada por el programa para hacer cálculos o procesamientos de datos.

La escritura de datos por otra parte es la generación o presentación de información después de haber sido procesada.

Ambos procesos de lectura y escritura pueden ser

* Interacciones con el usuario
* Con archivos
  + Para este curso serán específicamente de texto (.txt) y de valores (.csv)

# Modulo 2: Almacenamientos en memoria

## Variables y constantes

En programación, una variable es un elemento de datos cuyo valor puede cambiar durante el curso de la ejecución de un programa. El uso de variables es indispensable para la programación debido a que son estas las cargan la información que se utilizara en el programa. Al leer los datos, ya sea desde la consola (por el usuario) o de un archivo, estos se deben cargar en una variable u otro tipo de elemento de memoria (lista, tuplas o diccionarios) para que se puedan utilizar durante la ejecución del programa.

La única diferencia entre variables y constantes es su funcionamiento durante la ejecución del programa. Las variables si pueden cambiar su valor durante su ejecución, las constantes no pueden. A diferencia de otros lenguajes de programación, Python NO identifica/diferencias constantes.

Partes de la variable

* Identificador
  + Corresponde al nombre que utilizaremos para manejar la variable. Este identificador es el que se usa para las diversas operaciones y manejo de información a lo largo del programa.
  + Reglas:
    - El identificador tiene que iniciar con una letra (del alfabeto ingles)
    - No puede tener espacios intermedios
    - Solo puede estar compuesta de caracteres letras, números y guion bajo (a-z, 0-9, \_) "Python Variables - W3Schools." <https://www.w3schools.com/python/python_variables.asp>.
* Valor
  + Corresponde a la información que guarda la variable.

### Tipos de variables

Las variables manejan todos los tipos de datos previamente descritos y los operadores pertinentes.

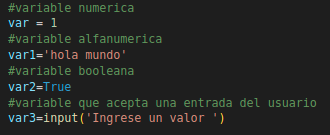
Tipos de datos

* Alfanumérico
* Numérico
* Booleano

### Declaración/Creación de variables

Para crear una variable, se hace de izquierda a derecha, colocando el identificador, seguido de un signo de igual (=), seguido del valor (alfanumérico, numérico o booleano) o de expresión (palabra reservada, ecuación, función o proceso)

Ejemplo



### Lectura de variables

En Python, la lectura se realiza con un simple llamado a la variable utilizando su identificador.

Ejemplo

### Modificación de variables

Para modificar los valores de las variables, primeramente, hay que realizar el llamado del identificador, seguido de un igual, acompañado de una expresión que modificara su valor

Ejemplo



En este ejemplo se agrega 5 unidades a la variable “var”

Algunas maneras de abreviar la expresión, hacerla más compacta es utilizando los siguientes operadores

Operadores de variables opcionales

|  |  |  |
| --- | --- | --- |
| Operador | función | ejemplo |
| += | Suma y asigna | X+=1 |
| -= | Resta y asigna | x-=1 |
| \*= | Multiplica y asigna | X\*=1 |
| /= | Divide y asigna | X/=1 |

### Operaciones de variables

Las operaciones dependerán del tipo de datos, y siguen los mismos lineamientos previamente explicados

Actividad de variables

1. Solicitar datos al usuario
   1. Nombre
   2. Cedula
   3. Año de nacimiento
      1. Se asume que todos nacieron el 1 de enero

### Conversión entre tipos de variables

Al utilizar la función *input* para solicitar datos de parte el usuario, notaremos que estos son automáticamente transformados a tipo alfanumérico. Pero en algunos casos vamos a necesitar trabajar con datos numéricos.

Esta transformación se hace mediante

Fin de clase 1

--------------------------------------

Clase 2

## Arreglos

Hasta el momento hemos guardado datos por variables separadas, pero en la mayoría de los casos requerimos utilizar datos en conjunto o que tengan una manera ordenada. Para esta finalidad utilizamos arreglos.

Definimos arreglos como: una colección ordenada de datos Los arreglos (matrices) se emplean para almacenar múltiples valores en una sola variable, frente a las variables que sólo pueden almacenar un valor. [1]

Python tiene 3 principales tipos de arreglos. Estos son:

### Listas

* 1. Utiliza corchetes []
  2. Son mutables/cambiantes
  3. Permite duplicación de valores
  4. Se llaman a sus elementos por medio de índices, iniciando a partir del #0.
  5. Permite almacenar cualquier tipo de datos
     1. Numérico
     2. Alfanumérico
     3. Booleano

### Tuplas

* 1. Utiliza paréntesis ()
  2. Son inmutables
     1. Su valor no puede ser alterado, agregado o eliminado después que haya sido creado
  3. Se llaman a sus elementos por índices, iniciando a partir del #0
  4. Permite duplicación de valores
  5. Puede almacenar cualquier tipo de datos
  6. Para creación
     1. Abre paréntesis
     2. Incluir elemento
     3. Terminar con una coma “,” si es 1 elemento. 2 o más elementos, enumerar normalmente.
        1. Ejemplos
           1. 1 elemento

A=("apple”,)

* + - * 1. 2 elementos

A=("apple”, “pear”)

### Diccionarios

* 1. Utiliza llaves {}
  2. Guarda datos por medio de “llaves” y “valores”
  3. Los “valores” del diccionario son ordenados y mutables/cambiantes
  4. No permite duplicación de valores
     1. 1 “llave” no puede tener 2 “valores”
        1. Si se asigna un nuevo “valor” a la “llave”, se borrará el “valor” anterior

Tabla de características

|  |  |  |  |
| --- | --- | --- | --- |
|  | Mutable | Ordenado | Llamado |
| Listas | Si | Si | Por índices |
| Tuplas | No | Si | Por índices |
| Diccionarios | Si | Si | Por llaves |

### Uso recomendado de cada uno de los arreglos

Listas

* Para operaciones de arreglo
  + Vectores
  + Matrices
* Utilizando para base de datos

Tuplas

* Para data heterogénea (diferente)
* Puede ser utilizada como

Diccionarios

* Para crear esquemas de datos
  + En conjunto con listas
* Utilizados en JSON

Fin de clase 2

--------------

Clase 3

# Modulo 3: Depuración, decisiones y bucles

Al momento de escribir el programa, es de esperar que se produzcan errores, que nuestro programa tenga distintos comportamientos en función de las entradas recibidas o que un proceso especifico sea repetitivo, por lo que veremos maneras de corregir y/o mitigar estos errores, crear condiciones para evaluar y maneras de hacer acciones repetidas

## Depuración

Consiste en desarrollar una acción para detectar y solucionar problemas en la programación [2]

Iniciaremos primeramente con la detección y corrección de errores,

## Expresiones condicionales

Una expresión condicional es un tipo de expresión que comprueba los datos para ver si cumple una condición y, a continuación, realiza una acción en función del resultado. [3]

### Comparación simple

Son aquellas comparaciones donde tenemos 2 opciones.

If (condición )

Else

### Comparación anidada

Es un tipo de comparación donde se comparan múltiples condiciones

If

Elif (los necesarios)

Else

### Comparación múltiple

En utilizada cuando una condición tiene múltiples opciones.

Match

## Bucles

es una secuencia de instrucciones de código que se ejecuta repetidas veces, hasta que la condición asignada a dicho bucle deja de cumplirse [4]

### Bucle for

es una estructura de control en programación en la que se puede indicar de antemano el número máximo de iteraciones [5]

### Bucle while

El propósito es repetir un bloque de código mientras una condición se mantenga verdadera. [6]