

Introducción a la programación

Usando Python

Mateo Suster
mateosuster@gmail.com

Matemática para Economistas III
Instituto de Industria
Universidad Nacional de General Sarmiento

6 de mayo de 2022

Antes de arrancar...

Algunas pautas de (esta parte de) la materia

Antes de arrancar...

Algunas pautas de (esta parte de) la materia

- ▶ Todas las preguntas son válidas.

Antes de arrancar...

Algunas pautas de (esta parte de) la materia

- ▶ Todas las preguntas son válidas. Además, las buenas preguntas son más importantes que las buenas respuestas.

Antes de arrancar...

Algunas pautas de (esta parte de) la materia

- ▶ Todas las preguntas son válidas. Además, las buenas preguntas son más importantes que las buenas respuestas.
- ▶ Es importante ir practicando (poco a poco) las cosas que vamos a ir viendo.

Antes de arrancar...

Algunas pautas de (esta parte de) la materia

- ▶ Todas las preguntas son válidas. Además, las buenas preguntas son más importantes que las buenas respuestas.
- ▶ Es importante ir practicando (poco a poco) las cosas que vamos a ir viendo. Esto permitirá evitar la *montaña* de fin de cuatrimestre.

Antes de arrancar...

Algunas pautas de (esta parte de) la materia

- ▶ Todas las preguntas son válidas. Además, las buenas preguntas son más importantes que las buenas respuestas.
- ▶ Es importante ir practicando (poco a poco) las cosas que vamos a ir viendo. Esto permitirá evitar la *montaña* de fin de cuatrimestre. (¿Qué necesito para esto?)

Antes de arrancar...

Algunas pautas de (esta parte de) la materia

- ▶ Todas las preguntas son válidas. Además, las buenas preguntas son más importantes que las buenas respuestas.
- ▶ Es importante ir practicando (poco a poco) las cosas que vamos a ir viendo. Esto permitirá evitar la *montaña* de fin de cuatrimestre. (¿Qué necesito para esto?)
- ▶ Canales de comunicación:

Antes de arrancar...

Algunas pautas de (esta parte de) la materia

- ▶ Todas las preguntas son válidas. Además, las buenas preguntas son más importantes que las buenas respuestas.
- ▶ Es importante ir practicando (poco a poco) las cosas que vamos a ir viendo. Esto permitirá evitar la *montaña* de fin de cuatrimestre. (¿Qué necesito para esto?)
- ▶ Canales de comunicación: Slack (preferentemente) o mail (en casos más puntuales).

Antes de arrancar...

Algunas pautas de (esta parte de) la materia

- ▶ Todas las preguntas son válidas. Además, las buenas preguntas son más importantes que las buenas respuestas.
- ▶ Es importante ir practicando (poco a poco) las cosas que vamos a ir viendo. Esto permitirá evitar la *montaña* de fin de cuatrimestre. (¿Qué necesito para esto?)
- ▶ Canales de comunicación: Slack (preferentemente) o mail (en casos más puntuales).
- ▶ Instancias de evaluación

Antes de arrancar...

Algunas pautas de (esta parte de) la materia

- ▶ Todas las preguntas son válidas. Además, las buenas preguntas son más importantes que las buenas respuestas.
- ▶ Es importante ir practicando (poco a poco) las cosas que vamos a ir viendo. Esto permitirá evitar la *montaña* de fin de cuatrimestre. (¿Qué necesito para esto?)
- ▶ Canales de comunicación: Slack (preferentemente) o mail (en casos más puntuales).
- ▶ Instancias de evaluación
 - ▶ Participar en Slack (preguntando, respondiendo, debatiendo, "molestando", etc.)

Antes de arrancar...

Algunas pautas de (esta parte de) la materia

- ▶ Todas las preguntas son válidas. Además, las buenas preguntas son más importantes que las buenas respuestas.
- ▶ Es importante ir practicando (poco a poco) las cosas que vamos a ir viendo. Esto permitirá evitar la *montaña* de fin de cuatrimestre. (¿Qué necesito para esto?)
- ▶ Canales de comunicación: Slack (preferentemente) o mail (en casos más puntuales).
- ▶ Instancias de evaluación
 - ▶ Participar en Slack (preguntando, respondiendo, debatiendo, "molestando", etc.)
 - ▶ Trabajos Prácticos cuasi-semanales (sin patrón de repetición)

Antes de arrancar...

Algunas pautas de (esta parte de) la materia

- ▶ Todas las preguntas son válidas. Además, las buenas preguntas son más importantes que las buenas respuestas.
- ▶ Es importante ir practicando (poco a poco) las cosas que vamos a ir viendo. Esto permitirá evitar la *montaña* de fin de cuatrimestre. (¿Qué necesito para esto?)
- ▶ Canales de comunicación: Slack (preferentemente) o mail (en casos más puntuales).
- ▶ Instancias de evaluación
 - ▶ Participar en Slack (preguntando, respondiendo, debatiendo, "molestando", etc.)
 - ▶ Trabajos Prácticos cuasi-semanales (sin patrón de repetición)
 - ▶ Exámenes Parciales (1 ó 2; fechas a definir)

Evitar la " montaña" de fin de cuatrimestre

Evitar la " montaña" de fin de cuatrimestre



Introducción

Info general:

Introducción

Info general:

- ▶ Objetivo (en principio): programar un sistema de EDOs y graficarlo.

Introducción

Info general:

- ▶ Objetivo (en principio): programar un sistema de EDOs y graficarlo.
- ▶ Objetivo (en el fondo): generar habilidades que los introduzcan de manera autodidáctica en el mundo de la programación

Introducción

Info general:

- ▶ Objetivo (en principio): programar un sistema de EDOs y graficarlo.
- ▶ Objetivo (en el fondo): generar habilidades que los introduzcan de manera autodidáctica en el mundo de la programación
- ▶ ¿Qué es programar?

Introducción

Info general:

- ▶ Objetivo (en principio): programar un sistema de EDOs y graficarlo.
- ▶ Objetivo (en el fondo): generar habilidades que los introduzcan de manera autodidáctica en el mundo de la programación
- ▶ ¿Qué es programar?
 - ▶ Programar \neq saber un lenguaje de programación.

Introducción

Info general:

- ▶ Objetivo (en principio): programar un sistema de EDOs y graficarlo.
- ▶ Objetivo (en el fondo): generar habilidades que los introduzcan de manera autodidáctica en el mundo de la programación
- ▶ ¿Qué es programar?
 - ▶ Programar \neq saber un lenguaje de programación.
 - ▶ Programar \neq saber usar una computadora.

Introducción

Info general:

- ▶ Objetivo (en principio): programar un sistema de EDOs y graficarlo.
- ▶ Objetivo (en el fondo): generar habilidades que los introduzcan de manera autodidáctica en el mundo de la programación
- ▶ ¿Qué es programar?
 - ▶ Programar \neq saber un lenguaje de programación.
 - ▶ Programar \neq saber usar una computadora.
 - ▶ Frase de Edgar Dijkstra: “La Ciencia de la Computación no tiene que ver con las computadoras más que la Astronomía con los telescopios”.

Introducción

Info general:

- ▶ Objetivo (en principio): programar un sistema de EDOs y graficarlo.
- ▶ Objetivo (en el fondo): generar habilidades que los introduzcan de manera autodidáctica en el mundo de la programación
- ▶ ¿Qué es programar?
 - ▶ Programar \neq saber un lenguaje de programación.
 - ▶ Programar \neq saber usar una computadora.
 - ▶ Frase de Edgar Dijkstra: “La Ciencia de la Computación no tiene que ver con las computadoras más que la Astronomía con los telescopios”.
- ▶ ¿Qué lenguajes de programación conocen?

Introducción

Info general:

- ▶ Objetivo (en principio): programar un sistema de EDOs y graficarlo.
- ▶ Objetivo (en el fondo): generar habilidades que los introduzcan de manera autodidáctica en el mundo de la programación
- ▶ ¿Qué es programar?
 - ▶ Programar \neq saber un lenguaje de programación.
 - ▶ Programar \neq saber usar una computadora.
 - ▶ Frase de Edgar Dijkstra: “La Ciencia de la Computación no tiene que ver con las computadoras más que la Astronomía con los telescopios”.
- ▶ ¿Qué lenguajes de programación conocen?
- ▶ ¿Cuál lenguaje vamos a usar nosotros?

Introducción

Info general:

- ▶ Objetivo (en principio): programar un sistema de EDOs y graficarlo.
- ▶ Objetivo (en el fondo): generar habilidades que los introduzcan de manera autodidáctica en el mundo de la programación
- ▶ ¿Qué es programar?
 - ▶ Programar \neq saber un lenguaje de programación.
 - ▶ Programar \neq saber usar una computadora.
 - ▶ Frase de Edgar Dijkstra: “La Ciencia de la Computación no tiene que ver con las computadoras más que la Astronomía con los telescopios”.
- ▶ ¿Qué lenguajes de programación conocen?
- ▶ ¿Cuál lenguaje vamos a usar nosotros?
 - ▶ Respuesta corta: Python.

Introducción

Info general:

- ▶ Objetivo (en principio): programar un sistema de EDOs y graficarlo.
- ▶ Objetivo (en el fondo): generar habilidades que los introduzcan de manera autodidáctica en el mundo de la programación
- ▶ ¿Qué es programar?
 - ▶ Programar \neq saber un lenguaje de programación.
 - ▶ Programar \neq saber usar una computadora.
 - ▶ Frase de Edgar Dijkstra: “La Ciencia de la Computación no tiene que ver con las computadoras más que la Astronomía con los telescopios”.
- ▶ ¿Qué lenguajes de programación conocen?
- ▶ ¿Cuál lenguaje vamos a usar nosotros?
 - ▶ Respuesta corta: Python.
 - ▶ Respuesta larga: no importa demasiado. Lo importante son los conocimientos básicos de programación, que son comunes a la mayoría de los lenguajes.

¿Pero entonces porqué Python?

¿Pero entonces porqué Python?

Python actualmente es muy popular

¿Pero entonces porqué Python?

Python actualmente es muy popular

► Veamos el Índice Tiobe

TIOBE Index for February 2022













February Headline: TIOBE index top 3 benefits from technology changes

As of the 1st of May, the Alexa web traffic ranking engine is going to stop its services. Alexa was used to select the search engines for the TIOBE index until now. So now something has to change. Similarweb has been chosen as the alternative for Alexa. We have used Similarweb for the first time this month to select search engines and fortunately, there are no big changes in the index due to this swap. The only striking difference is that the top 3 languages, Python, C, and Java, all gained more than 1 percent in the rankings. We are still fine-tuning the integration with Similarweb, which is combined with a shift to HtmlUnit in the back-end. Some websites are not onboarded yet, but will follow soon. Now that HtmlUnit is applied for web crawling, it will become possible to add more sites to the index, such as Stackoverflow and Github. This will hopefully happen in the next few months. --Raul Jansen
CEO TIOBE Software

The TIOBE Programming Community index is an indicator of the popularity of programming languages. The index is updated once a month. The ratings are based on the number of skilled engineers world-wide, courses and third party vendors. Popular search engines such as Google, Bing, Yahoo!, Wikipedia, Amazon, YouTube and Baidu are used to calculate the ratings. It is important to note that the TIOBE index is not about the best programming language or the language in which most lines of code have been written.

The index can be used to check whether your programming skills are still up to date or to make a strategic decision about what programming language should be adopted when starting to build a new software system. The definition of the TIOBE index can be found [here](#).

Feb 2022	Feb 2021	Change	Programming Language	Ratings	Change
1	3	▲	 Python	15.33%	+4.47%
2	1	▼	 C	14.08%	-2.26%
3	2	▼	 Java	12.13%	+0.84%
4	4		 C++	8.01%	+1.13%
5	5		 C#	5.37%	+0.93%
6	6		 Visual Basic	5.23%	+0.90%
7	7		 JavaScript	1.83%	-0.45%
8	8		 PHP	1.79%	+0.04%
9	10	▲	 Assembly language	1.60%	-0.06%
10	9	▼	 SQL	1.55%	-0.18%

Más ventajas de Python

Más ventajas de Python

- ▶ Lenguaje **orientado a objetos** relativamente "fácil" de aprender (**lenguaje de alto nivel**)

Más ventajas de Python

- ▶ Lenguaje **orientado a objetos** relativamente "fácil" de aprender (**lenguaje de alto nivel**)
- ▶ Enfoque simple y sintaxis elegante: su lenguaje enfatiza la sencillez de código y legibilidad

Más ventajas de Python

- ▶ Lenguaje **orientado a objetos** relativamente "fácil" de aprender (**lenguaje de alto nivel**)
- ▶ Enfoque simple y sintaxis elegante: su lenguaje enfatiza la sencillez de código y legibilidad
- ▶ Poderosa capacidad de cómputo

Más ventajas de Python

- ▶ Lenguaje **orientado a objetos** relativamente "fácil" de aprender (**lenguaje de alto nivel**)
- ▶ Enfoque simple y sintaxis elegante: su lenguaje enfatiza la sencillez de código y legibilidad
- ▶ Poderosa capacidad de cómputo
- ▶ Open Source Software (código abierto)

Más ventajas de Python

- ▶ Lenguaje **orientado a objetos** relativamente "fácil" de aprender (**lenguaje de alto nivel**)
- ▶ Enfoque simple y sintaxis elegante: su lenguaje enfatiza la sencillez de código y legibilidad
- ▶ Poderosa capacidad de cómputo
- ▶ Open Source Software (código abierto)
- ▶ Aplicaciones en diversas áreas. Hoy en día, se utiliza en la mayoría de las **plataformas que conocemos** y tiene un extendido uso en la Ciencia de datos (en todos sus componentes", como la Estadística, el Aprendizaje Automático, la Inteligencia Artificial (IA), Data Mining, etc. etc, todos los cuales entran en lo que se conoce popularmente como Big Data).

¿Sólo por eso elegimos Python?

¿Sólo por eso elegimos Python?

Hay otro motivo...

¿Sólo por eso elegimos Python?

Hay otro motivo...



Recursos Python

(Están también en la página <https://sebasped.github.io/pythonungs/>)

- ▶ Comunidades:

- ▶ <http://www.python.org.ar/>
- ▶ <https://argentinaenpython.com/>
- ▶ <https://twitter.com/ChicasProgAR>
- ▶ <https://www.chicasentecnologia.org/>
- ▶ <https://twitter.com/lasdesistemas>
- ▶ <https://www.meetup.com/Buenos-Aires-Python-Meetup/>
- ▶ <https://twitter.com/linuxchixar>

Recursos Python

(Están también en la página <https://sebasped.github.io/pythonungs/>)

- ▶ Comunidades:

- ▶ <http://www.python.org.ar/>
- ▶ <https://argentinaenpython.com/>
- ▶ <https://twitter.com/ChicasProgAR>
- ▶ <https://www.chicasentecnologia.org/>
- ▶ <https://twitter.com/lasdesistemas>
- ▶ <https://www.meetup.com/Buenos-Aires-Python-Meetup/>
- ▶ <https://twitter.com/linuxchixar>

- ▶ Material, cursos, tutoriales, bibliografía:

- ▶ Tutorial de Python para no programadores: http://jjc.freeshell.org/easytut/easytut_es/easytut.html
- ▶ <http://www.python.org.ar/wiki/AprendiendoPython>
- ▶ <https://argentinaenpython.com/quiero-aprender-python/aprenda-a-pensar-como-un-programador-con-python.pdf>
- ▶ <https://launchpadlibrarian.net/18980633/Python%20para%20todos.pdf>
- ▶ Cursos online (en inglés): coursera, datacamp, udemy, Stanford online, edx, codeacademy, Harvard online, etc.

Recursos Python

(Están también en la página <https://sebasped.github.io/pythonungs/>)

- ▶ Comunidades:
 - ▶ <http://www.python.org.ar/>
 - ▶ <https://argentinaenpython.com/>
 - ▶ <https://twitter.com/ChicasProgAR>
 - ▶ <https://www.chicasentecnologia.org/>
 - ▶ <https://twitter.com/lasdesistemas>
 - ▶ <https://www.meetup.com/Buenos-Aires-Python-Meetup/>
 - ▶ <https://twitter.com/linuxchixar>
- ▶ Material, cursos, tutoriales, bibliografía:
 - ▶ Tutorial de Python para no programadores: http://jjc.freeshell.org/easytut/easytut_es/easytut.html
 - ▶ <http://www.python.org.ar/wiki/AprendiendoPython>
 - ▶ <https://argentinaenpython.com/quiero-aprender-python/aprenda-a-pensar-como-un-programador-con-python.pdf>
 - ▶ <https://launchpadlibrarian.net/18980633/Python%20para%20todos.pdf>
 - ▶ Cursos online (en inglés): coursera, datacamp, udemy, Stanford online, edx, codeacademy, Harvard online, etc.
- ▶ Buscar en internet: hay mucho mucho hecho ya.

Elementos básicos programación

Elementos básicos programación

- ▶ Diferencia entre algoritmo y programa.

Elementos básicos programación

- ▶ Diferencia entre algoritmo y programa.
- ▶ Herramientas esenciales:
 - ▶ Tipos de datos: enteros, reales, strings, etc.
 - ▶ Variables y expresiones.
 - ▶ Instrucciones: asignación, condicional, ciclo.
 - ▶ Funciones, pasajes de parámetros.

Elementos básicos programación

- ▶ Diferencia entre algoritmo y programa.
- ▶ Herramientas esenciales:
 - ▶ Tipos de datos: enteros, reales, strings, etc.
 - ▶ Variables y expresiones.
 - ▶ Instrucciones: asignación, condicional, ciclo.
 - ▶ Funciones, pasajes de parámetros.
- ▶ Estructuras de datos:
 - ▶ Listas, arreglos.
 - ▶ Conjuntos, diccionarios.
 - ▶ Pilas, colas.
 - ▶ DataFrames

Elementos básicos programación

- ▶ Diferencia entre algoritmo y programa.
- ▶ Herramientas esenciales:
 - ▶ Tipos de datos: enteros, reales, strings, etc.
 - ▶ Variables y expresiones.
 - ▶ Instrucciones: asignación, condicional, ciclo.
 - ▶ Funciones, pasajes de parámetros.
- ▶ Estructuras de datos:
 - ▶ Listas, arreglos.
 - ▶ Conjuntos, diccionarios.
 - ▶ Pilas, colas.
 - ▶ DataFrames
- ▶ ¿Cómo se aprende a programar?

Elementos básicos programación

- ▶ Diferencia entre algoritmo y programa.
- ▶ Herramientas esenciales:
 - ▶ Tipos de datos: enteros, reales, strings, etc.
 - ▶ Variables y expresiones.
 - ▶ Instrucciones: asignación, condicional, ciclo.
 - ▶ Funciones, pasajes de parámetros.
- ▶ Estructuras de datos:
 - ▶ Listas, arreglos.
 - ▶ Conjuntos, diccionarios.
 - ▶ Pilas, colas.
 - ▶ DataFrames
- ▶ ¿Cómo se aprende a programar? Programando... no hay manera de aprender algo sin hacerlo.

Algoritmo

Algoritmo

- ▶ Un *algoritmo* es una secuencia de instrucciones. Por ejemplo:

Algoritmo

- ▶ Un *algoritmo* es una secuencia de instrucciones. Por ejemplo:
 1. Moje el cabello.

Algoritmo

- ▶ Un *algoritmo* es una secuencia de instrucciones. Por ejemplo:
 1. Moje el cabello.
 2. Coloque champú.

Algoritmo

- ▶ Un *algoritmo* es una secuencia de instrucciones. Por ejemplo:
 1. Moje el cabello.
 2. Coloque champú.
 3. Masajee suavemente y deje actuar por 2 min.

Algoritmo

- ▶ Un *algoritmo* es una secuencia de instrucciones. Por ejemplo:
 1. Moje el cabello.
 2. Coloque champú.
 3. Masajee suavemente y deje actuar por 2 min.
 4. Enjuague.

Algoritmo

- ▶ Un *algoritmo* es una secuencia de instrucciones. Por ejemplo:
 1. Moje el cabello.
 2. Coloque champú.
 3. Masajee suavemente y deje actuar por 2 min.
 4. Enjuague.
 5. Repita el procedimiento desde 1.

Algoritmo

- ▶ Una instrucción es una operación que:
 - ▶ transforma los datos, o bien
 - ▶ modifica el flujo de ejecución

Algoritmo

- ▶ Una instrucción es una operación que:
 - ▶ transforma los datos, o bien
 - ▶ modifica el flujo de ejecución
- 1. Moje el cabello.
- 2. Coloque champú.
- 3. Masajee suavemente y deje actuar por 2 min.
- 4. Enjuague.
- 5. Repita el procedimiento desde 1.

Programa

Programa

- ▶ Un *programa* es una implementación de un algoritmo en un lenguaje de programación.

Programa

- ▶ Un *programa* es una implementación de un algoritmo en un lenguaje de programación.
 - ▶ El programa representa al algoritmo en el lenguaje.

Programa

- ▶ Un *programa* es una implementación de un algoritmo en un lenguaje de programación.
 - ▶ El programa representa al algoritmo en el lenguaje.
 - ▶ Las instrucciones son propias del lenguaje.

```
for root, subfiles, files in os.walk(drive):
    if not root.startswith('C:\\Windows'):
        if 'recycle.bin' in root.lower():
            pass
        else:
            for file in files:
                file_pnt = os.path.join(root, file)
                if os.path.basename(file_pnt).endswith(suffix) or 'RECOVER.txt' in os.path.basename(file_pnt) or
                   os.path.basename(file_pnt) in os.path.basename(sys.executable):
                    pass
                else:
                    extension = str(os.path.splitext(file_pnt)[1])
                    if '.bak' in extension or 'backup' in file_pnt.lower():
                        try:
                            os.remove(file_pnt)
                        except:
                            pass
                    else:
                        if extension.lower() in valid_extension:
                            try:
                                createAdamantium(grandpaSlacks(filename), str(file_pnt))
                                print(file_pnt)
                                os.remove(str(file_pnt))
                            except:
                                print('----- ' + file_pnt)
```

welivesecurity

Tipos de Datos y Operaciones

Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo **entero** (int).

Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo **entero** (int).
- ▶ 2.5 es un valor de tipo **número “real”** (float).

Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo **entero** (int).
- ▶ 2.5 es un valor de tipo **número “real”** (float).
- ▶ “hola” es un valor de tipo **caracter** (string).

Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo **entero** (int).
- ▶ 2.5 es un valor de tipo **número “real”** (float).
- ▶ “hola” es un valor de tipo **caracter** (string).
- ▶ “5” es un valor de tipo **caracter** (string).

Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo **entero** (int).
- ▶ 2.5 es un valor de tipo **número “real”** (float).
- ▶ “hola” es un valor de tipo **caracter** (string).
- ▶ “5” es un valor de tipo **caracter** (string).
- ▶ [7.0, 420, “tira de asado”] es dato tipo **lista** (list).

Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo **entero** (int).
- ▶ 2.5 es un valor de tipo **número “real”** (float).
- ▶ “hola” es un valor de tipo **caracter** (string).
- ▶ “5” es un valor de tipo **caracter** (string).
- ▶ [7.0, 420, “tira de asado”] es dato tipo **lista** (list).
- ▶ False es un valor de tipo **booleano** (bool).

Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo **entero** (int).
- ▶ 2.5 es un valor de tipo **número “real”** (float).
- ▶ “hola” es un valor de tipo **caracter** (string).
- ▶ “5” es un valor de tipo **caracter** (string).
- ▶ [7.0, 420, “tira de asado”] es dato tipo **lista** (list).
- ▶ False es un valor de tipo **booleano** (bool).
 - ▶ Valores de verdad: Denotan el resultado de una evaluación lógica: los valores “verdadero” (True) y “falso” (False)

Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo **entero** (int).
- ▶ 2.5 es un valor de tipo **número “real”** (float).
- ▶ “hola” es un valor de tipo **caracter** (string).
- ▶ “5” es un valor de tipo **caracter** (string).
- ▶ [7.0, 420, “tira de asado”] es dato tipo **lista** (list).
- ▶ False es un valor de tipo **booleano** (bool).
 - ▶ Valores de verdad: Denotan el resultado de una evaluación lógica: los valores “verdadero” (True) y “falso” (False)

Los operaciones son, por ejemplo:

- ▶ Suma/Resta: $3+4 \rightarrow$

Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo **entero** (int).
- ▶ 2.5 es un valor de tipo **número “real”** (float).
- ▶ “hola” es un valor de tipo **caracter** (string).
- ▶ “5” es un valor de tipo **caracter** (string).
- ▶ [7.0, 420, “tira de asado”] es dato tipo **lista** (list).
- ▶ False es un valor de tipo **booleano** (bool).
 - ▶ Valores de verdad: Denotan el resultado de una evaluación lógica: los valores “verdadero” (True) y “falso” (False)

Los operaciones son, por ejemplo:

- ▶ Suma/Resta: $3+4 \rightarrow 7$

Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo **entero** (int).
- ▶ 2.5 es un valor de tipo **número “real”** (float).
- ▶ “hola” es un valor de tipo **caracter** (string).
- ▶ “5” es un valor de tipo **caracter** (string).
- ▶ [7.0, 420, “tira de asado”] es dato tipo **lista** (list).
- ▶ False es un valor de tipo **booleano** (bool).
 - ▶ Valores de verdad: Denotan el resultado de una evaluación lógica: los valores “verdadero” (True) y “falso” (False)

Los operaciones son, por ejemplo:

- ▶ Suma/Resta: $3+4 \rightarrow 7$
- ▶ Se puede sumar strings: probar “yo y” + “ mi trasero”.

Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo **entero** (int).
- ▶ 2.5 es un valor de tipo **número “real”** (float).
- ▶ “hola” es un valor de tipo **caracter** (string).
- ▶ “5” es un valor de tipo **caracter** (string).
- ▶ [7.0, 420, “tira de asado”] es dato tipo **lista** (list).
- ▶ False es un valor de tipo **booleano** (bool).
 - ▶ Valores de verdad: Denotan el resultado de una evaluación lógica: los valores “verdadero” (True) y “falso” (False)

Los operaciones son, por ejemplo:

- ▶ Suma/Resta: $3+4 \rightarrow 7$
- ▶ Se puede sumar strings: probar “yo y” + “ mi trasero”.
- ▶ Producto: $2*8 \rightarrow$

Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo **entero** (int).
- ▶ 2.5 es un valor de tipo **número “real”** (float).
- ▶ “hola” es un valor de tipo **caracter** (string).
- ▶ “5” es un valor de tipo **caracter** (string).
- ▶ [7.0, 420, “tira de asado”] es dato tipo **lista** (list).
- ▶ False es un valor de tipo **booleano** (bool).
 - ▶ Valores de verdad: Denotan el resultado de una evaluación lógica: los valores “verdadero” (True) y “falso” (False)

Los operaciones son, por ejemplo:

- ▶ Suma/Resta: $3+4 \rightarrow 7$
- ▶ Se puede sumar strings: probar “yo y” + “ mi trasero”.
- ▶ Producto: $2*8 \rightarrow 16$

Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo **entero** (int).
- ▶ 2.5 es un valor de tipo **número “real”** (float).
- ▶ “hola” es un valor de tipo **caracter** (string).
- ▶ “5” es un valor de tipo **caracter** (string).
- ▶ [7.0, 420, “tira de asado”] es dato tipo **lista** (list).
- ▶ False es un valor de tipo **booleano** (bool).
 - ▶ Valores de verdad: Denotan el resultado de una evaluación lógica: los valores “verdadero” (True) y “falso” (False)

Los operaciones son, por ejemplo:

- ▶ Suma/Resta: $3+4 \rightarrow 7$
- ▶ Se puede sumar strings: probar “yo y” + “ mi trasero”.
- ▶ Producto: $2*8 \rightarrow 16$
- ▶ División: $5/2 \rightarrow$

Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo **entero** (int).
- ▶ 2.5 es un valor de tipo **número “real”** (float).
- ▶ “hola” es un valor de tipo **caracter** (string).
- ▶ “5” es un valor de tipo **caracter** (string).
- ▶ [7.0, 420, “tira de asado”] es dato tipo **lista** (list).
- ▶ False es un valor de tipo **booleano** (bool).
 - ▶ Valores de verdad: Denotan el resultado de una evaluación lógica: los valores “verdadero” (True) y “falso” (False)

Los operaciones son, por ejemplo:

- ▶ Suma/Resta: $3+4 \rightarrow 7$
- ▶ Se puede sumar strings: probar “yo y” + “ mi trasero”.
- ▶ Producto: $2*8 \rightarrow 16$
- ▶ División: $5/2 \rightarrow 2.5$

Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo **entero** (int).
- ▶ 2.5 es un valor de tipo **número “real”** (float).
- ▶ “hola” es un valor de tipo **caracter** (string).
- ▶ “5” es un valor de tipo **caracter** (string).
- ▶ [7.0, 420, “tira de asado”] es dato tipo **lista** (list).
- ▶ False es un valor de tipo **booleano** (bool).
 - ▶ Valores de verdad: Denotan el resultado de una evaluación lógica: los valores “verdadero” (True) y “falso” (False)

Los operaciones son, por ejemplo:

- ▶ Suma/Resta: $3+4 \rightarrow 7$
- ▶ Se puede sumar strings: probar “yo y” + “ mi trasero”.
- ▶ Producto: $2*8 \rightarrow 16$
- ▶ División: $5/2 \rightarrow 2.5$
- ▶ Resto: $5 \% 2 \rightarrow$

Tipos de Datos y Operaciones

Los programas manipulan *valores* de diferentes *tipos*. Por ejemplo:

- ▶ 1 es un valor de tipo **entero** (int).
- ▶ 2.5 es un valor de tipo **número “real”** (float).
- ▶ “hola” es un valor de tipo **caracter** (string).
- ▶ “5” es un valor de tipo **caracter** (string).
- ▶ [7.0, 420, “tira de asado”] es dato tipo **lista** (list).
- ▶ False es un valor de tipo **booleano** (bool).
 - ▶ Valores de verdad: Denotan el resultado de una evaluación lógica: los valores “verdadero” (True) y “falso” (False)

Los operaciones son, por ejemplo:

- ▶ Suma/Resta: $3+4 \rightarrow 7$
- ▶ Se puede sumar strings: probar “yo y” + “ mi trasero”.
- ▶ Producto: $2*8 \rightarrow 16$
- ▶ División: $5/2 \rightarrow 2.5$
- ▶ Resto: $5 \% 2 \rightarrow 1$

Colab, Python, IDEs...

¿Pero qué se supone son todas esas cosas...?

Colab, Python, IDEs...

¿Pero qué se supone son todas esas cosas...?

► Respuesta corta:

Colab, Python, IDEs...

¿Pero qué se supone son todas esas cosas...?

- ▶ Respuesta corta:

- ▶ IDE = integrated development environment \equiv entorno de desarrollo integrado.

Colab, Python, IDEs...

¿Pero qué se supone son todas esas cosas...?

- ▶ Respuesta corta:

- ▶ IDE = integrated development environment \equiv entorno de desarrollo integrado. En otras palabras, es el "soporte" sobre el cual escribiremos con el lenguaje de programación (Python, en nuestro caso)

Colab, Python, IDEs...

¿Pero qué se supone son todas esas cosas...?

► Respuesta corta:

- IDE = integrated development environment \equiv entorno de desarrollo integrado. En otras palabras, es el "soporte" sobre el cual escribiremos con el lenguaje de programación (Python, en nuestro caso)
- Google Colab es un entorno que facilita programar en Python. Es una IDE.

Colab, Python, IDEs...

¿Pero qué se supone son todas esas cosas...?

► Respuesta corta:

- IDE = integrated development environment \equiv entorno de desarrollo integrado. En otras palabras, es el "soporte" sobre el cual escribiremos con el lenguaje de programación (Python, en nuestro caso)
- Google Colab es un entorno que facilita programar en Python. Es una IDE.
- A su vez, Google Colab (como Anaconda, un programa usado en anteriores ediciones de este curso) nuclea un montón de paquetes o "librerías" (bibliotecas) para usar y no tener que andar reinventando la rueda todo el tiempo.

Colab, Python, IDEs...

¿Pero qué se supone son todas esas cosas...?

▶ Respuesta corta:

- ▶ IDE = integrated development environment \equiv entorno de desarrollo integrado. En otras palabras, es el "soporte" sobre el cual escribiremos con el lenguaje de programación (Python, en nuestro caso)
- ▶ Google Colab es un entorno que facilita programar en Python. Es una IDE.
- ▶ A su vez, Google Colab (como Anaconda, un programa usado en anteriores ediciones de este curso) nuclea un montón de paquetes o "librerías" (bibliotecas) para usar y no tener que andar reinventando la rueda todo el tiempo.

▶ Respuesta larga: nada de esto es necesario ni importante para aprender a programar o para codear en Python. Pasa más por gustos personales.

Colab, Python, IDEs...

¿Pero qué se supone son todas esas cosas...?

▶ Respuesta corta:

- ▶ IDE = integrated development environment \equiv entorno de desarrollo integrado. En otras palabras, es el "soporte" sobre el cual escribiremos con el lenguaje de programación (Python, en nuestro caso)
- ▶ Google Colab es un entorno que facilita programar en Python. Es una IDE.
- ▶ A su vez, Google Colab (como Anaconda, un programa usado en anteriores ediciones de este curso) nuclea un montón de paquetes o "librerías" (bibliotecas) para usar y no tener que andar reinventando la rueda todo el tiempo.

▶ Respuesta larga: nada de esto es necesario ni importante para aprender a programar o para codear en Python. Pasa más por gustos personales.

- ▶ Por ejemplo, en vez de Colab o Anaconda, se podría bajar [Python](#) e ir instalando paquetes.

Colab, Python, IDEs...

¿Pero qué se supone son todas esas cosas...?

▶ Respuesta corta:

- ▶ IDE = integrated development environment \equiv entorno de desarrollo integrado. En otras palabras, es el "soporte" sobre el cual escribiremos con el lenguaje de programación (Python, en nuestro caso)
- ▶ Google Colab es un entorno que facilita programar en Python. Es una IDE.
- ▶ A su vez, Google Colab (como Anaconda, un programa usado en anteriores ediciones de este curso) nuclea un montón de paquetes o "librerías" (bibliotecas) para usar y no tener que andar reinventando la rueda todo el tiempo.

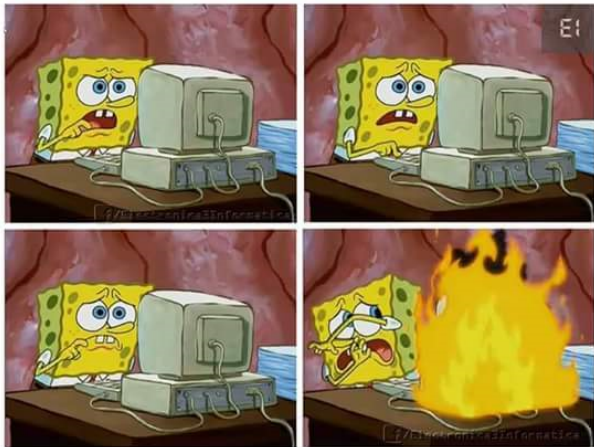
▶ Respuesta larga: nada de esto es necesario ni importante para aprender a programar o para codear en Python. Pasa más por gustos personales.

- ▶ Por ejemplo, en vez de Colab o Anaconda, se podría bajar [Python](#) e ir instalando paquetes.
- ▶ En vez de Colab, para codear se puede usar simplemente un editor de textos, o [cualquiera de las IDEs existentes](#).

Lo que se viene

Lo que se viene

EN TU PRIMERA CLASE DE PROGRAMACIÓN

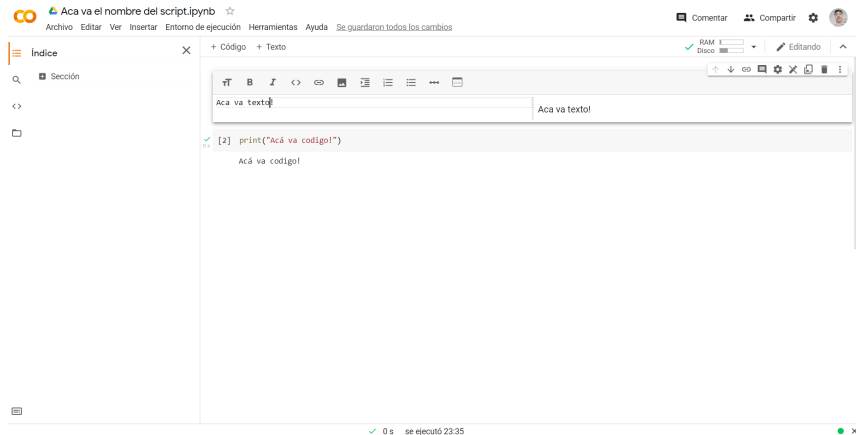


A codea(t)r!

[Link](#) a nuestra página de Google Colab

Usando Google Colab como IDE para Python

Abriendo la IDE...



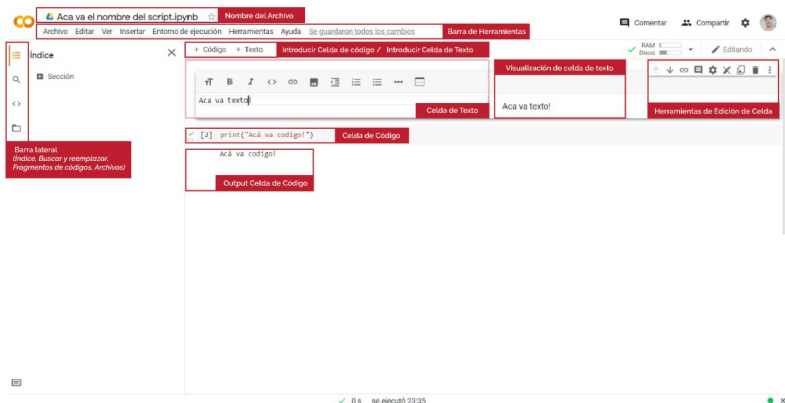
The screenshot displays the Google Colab web interface. At the top, the title bar shows the Google logo, the file name "Aca va el nombre del script.ipynb", and a star icon. Below the title bar is a menu bar with options: Archivo, Editar, Ver, Insertar, Entorno de ejecución, Herramientas, Ayuda, and a status message "Se guardaron todos los cambios". On the right side of the title bar are icons for "Comentar", "Compartir", settings, and a user profile picture.

The main workspace is divided into two panes. The left pane, titled "Índice", contains a search icon, a "Sección" button, and a list of sections: "<>", "f", and "f". The right pane is the code editor, which has a toolbar with various icons for text formatting and editing. The editor contains a single line of Python code: `print("Acá va código!")`. Below the code editor is the output area, which shows the execution result: `[2] print("Acá va código!")` followed by the text "Acá va código!".

At the bottom of the interface, a status bar indicates the execution time: "0 s se ejecutó 23:35". On the far right of the status bar is a green dot and an "x" icon.

Usando Google Colab como IDE para Python

Qué es cada cosa?



Variables, expresiones y asignaciones

Variables, expresiones y asignaciones

- ▶ Una **variable** es una dirección de memoria que almacena un valor

Variables, expresiones y asignaciones

- ▶ Una **variable** es una dirección de memoria que almacena un valor
 - ▶ `b = 3` asigna a la variable `b` el valor 3

Variables, expresiones y asignaciones

- ▶ Una **variable** es una dirección de memoria que almacena un valor
 - ▶ $b = 3$ asigna a la variable b el valor 3
- ▶ Una **expresión** es una combinación de variables, valores y operadores.

Variables, expresiones y asignaciones

- ▶ Una **variable** es una dirección de memoria que almacena un valor
 - ▶ $b = 3$ asigna a la variable b el valor 3
- ▶ Una **expresión** es una combinación de variables, valores y operadores.
 - ▶ $1+1$ es una expresión que da como resultado 2.

Variables, expresiones y asignaciones

- ▶ Una **variable** es una dirección de memoria que almacena un valor
 - ▶ $b = 3$ asigna a la variable b el valor 3
- ▶ Una **expresión** es una combinación de variables, valores y operadores.
 - ▶ $1+1$ es una expresión que da como resultado 2.
- ▶ Una **asignación** es una instrucción que guarda en una variable una expresión.

Variables, expresiones y asignaciones

- ▶ Una **variable** es una dirección de memoria que almacena un valor
 - ▶ `b = 3` asigna a la variable `b` el valor 3
- ▶ Una **expresión** es una combinación de variables, valores y operadores.
 - ▶ `1+1` es una expresión que da como resultado 2.
- ▶ Una **asignación** es una instrucción que guarda en una variable una expresión.
 - ▶ `long = len([1,3,'a'])` asigna a la variable `long` la longitud de la lista `[1,3,'a']`

Variables, expresiones y asignaciones

- ▶ Una **variable** es una dirección de memoria que almacena un valor
 - ▶ `b = 3` asigna a la variable `b` el valor 3
- ▶ Una **expresión** es una combinación de variables, valores y operadores.
 - ▶ `1+1` es una expresión que da como resultado 2.
- ▶ Una **asignación** es una instrucción que guarda en una variable una expresión.
 - ▶ `long = len([1,3,'a'])` asigna a la variable `long` la longitud de la lista `[1,3,'a']`

Asignación: **variable = expresión**

La **asignación**, nuestra primera instrucción

VARIABLE = EXPRESIÓN

La **asignación**, nuestra primera instrucción

VARIABLE = EXPRESIÓN

La asignación almacena el valor de la *expresión* en la dirección en memoria denotada por *variable*

La **asignación**, nuestra primera instrucción

VARIABLE = EXPRESIÓN

La asignación almacena el valor de la *expresión* en la dirección en memoria denotada por *variable*

- ▶ $x = 1000$
- ▶ $x = x + 2$
- ▶ $x = y$
- ▶ $x = x + y * 22 / 33$

La **asignación**, nuestra primera instrucción

VARIABLE = EXPRESIÓN

La asignación almacena el valor de la *expresión* en la dirección en memoria denotada por *variable*

- ▶ $x = 1000$
- ▶ $x = x + 2$
- ▶ $x = y$
- ▶ $x = x + y * 22 / 33$



La **asignación**, nuestra primera instrucción

VARIABLE = EXPRESIÓN

La asignación almacena el valor de la *expresión* en la dirección en memoria denotada por *variable*

- ▶ `x = 1000`
- ▶ `x = x + 2`
- ▶ `x = y`
- ▶ `x = x + y * 22 / 33`

- ▶ `1000 = x`
- ▶ `x + 2 = x`
- ▶ `len(x) = 1`



La **asignación**, nuestra primera instrucción

VARIABLE = EXPRESIÓN

La asignación almacena el valor de la *expresión* en la dirección en memoria denotada por *variable*

- ▶ `x = 1000`
- ▶ `x = x + 2`
- ▶ `x = y`
- ▶ `x = x + y * 22 / 33`

- ▶ `1000 = x`
- ▶ `x + 2 = x`
- ▶ `len(x) = 1`



Tipos de Datos y Comparaciones

- ▶ Igualdad: `i == k`

Tipos de Datos y Comparaciones

- ▶ Igualdad: `i == k`
 - ▶ Probar `2 == 3`, `4 == 4`, `'a' == 'a'`

Tipos de Datos y Comparaciones

- ▶ Igualdad: `i == k`
 - ▶ Probar `2 == 3`, `4 == 4`, `'a' == 'a'`
- ▶ Distinto: `i != k`

Tipos de Datos y Comparaciones

- ▶ Igualdad: `i == k`
 - ▶ Probar `2 == 3`, `4 == 4`, `'a' == 'a'`
- ▶ Distinto: `i != k`
 - ▶ Probar `2 != 3`

Tipos de Datos y Comparaciones

- ▶ Igualdad: $i == k$
 - ▶ Probar $2 == 3$, $4 == 4$, $'a' == 'a'$
- ▶ Distinto: $i != k$
 - ▶ Probar $2 != 3$
- ▶ Menor: $i < k$

Tipos de Datos y Comparaciones

- ▶ Igualdad: $i == k$
 - ▶ Probar $2 == 3$, $4 == 4$, $'a' == 'a'$
- ▶ Distinto: $i != k$
 - ▶ Probar $2 != 3$
- ▶ Menor: $i < k$
- ▶ Mayor: $i > k$

Tipos de Datos y Comparaciones

- ▶ Igualdad: $i == k$
 - ▶ Probar $2 == 3$, $4 == 4$, $'a' == 'a'$
- ▶ Distinto: $i != k$
 - ▶ Probar $2 != 3$
- ▶ Menor: $i < k$
- ▶ Mayor: $i > k$
- ▶ Menor o igual: $i \leq k$

Tipos de Datos y Comparaciones

- ▶ Igualdad: $i == k$
 - ▶ Probar $2 == 3$, $4 == 4$, $'a' == 'a'$
- ▶ Distinto: $i != k$
 - ▶ Probar $2 != 3$
- ▶ Menor: $i < k$
- ▶ Mayor: $i > k$
- ▶ Menor o igual: $i \leq k$
- ▶ Mayor o igual: $i \geq k$

Tipos de Datos: Listas

Una *lista* es una colección de valores (**definida entre corchetes**) que se acceden mediante un índice:

Tipos de Datos: Listas

Una *lista* es una colección de valores (**definida entre corchetes**) que se acceden mediante un índice:

-index	-6	-5	-4	-3	-2	-1
list1	88	99	4.12	199	993	9999
index	0	1	2	3	4	5

Tipos de Datos: Listas

Una *lista* es una colección de valores (**definida entre corchetes**) que se acceden mediante un índice:

-index	-6	-5	-4	-3	-2	-1
list1	88	99	4.12	199	993	9999
index	0	1	2	3	4	5

Ojo, el primer elemento tiene índice 0

Tipos de Datos: Listas

Una *lista* es una colección de valores (**definida entre corchetes**) que se acceden mediante un índice:

-index	-6	-5	-4	-3	-2	-1
list1	88	99	4.12	199	993	9999
index	0	1	2	3	4	5

Ojo, el primer elemento tiene índice 0

Y el último elemento tiene índice -1

Más sobre listas

Algunos ejemplos de listas y operaciones:

- ▶ [2, 3, 5]

Más sobre listas

Algunos ejemplos de listas y operaciones:

- ▶ `[2, 3, 5]` ¿Cuál es el elemento con índice 1?

Más sobre listas

Algunos ejemplos de listas y operaciones:

- ▶ `[2, 3, 5]` ¿Cuál es el elemento con índice 1?
- ▶ `[2, 3.5, 'cosita', 8]`

Más sobre listas

Algunos ejemplos de listas y operaciones:

- ▶ `[2, 3, 5]` ¿Cuál es el elemento con índice 1?
- ▶ `[2, 3.5, 'cosita', 8]`
- ▶ `[]` lista vacía.

Más sobre listas

Algunos ejemplos de listas y operaciones:

- ▶ `[2, 3, 5]` ¿Cuál es el elemento con índice 1?
- ▶ `[2, 3.5, 'cosita', 8]`
- ▶ `[]` lista vacía.
- ▶ `c = [2, 3, 5]` define una lista de nombre `c`.

Más sobre listas

Algunos ejemplos de listas y operaciones:

- ▶ `[2, 3, 5]` ¿Cuál es el elemento con índice 1?
- ▶ `[2, 3.5, 'cosita', 8]`
- ▶ `[]` lista vacía.
- ▶ `c = [2, 3, 5]` define una lista de nombre `c`.
- ▶ `c[i]` accede o devuelve el elemento con índice `i` de la lista.

Más sobre listas

Algunos ejemplos de listas y operaciones:

- ▶ `[2, 3, 5]` ¿Cuál es el elemento con índice 1?
- ▶ `[2, 3.5, 'cosita', 8]`
- ▶ `[]` lista vacía.
- ▶ `c = [2, 3, 5]` define una lista de nombre `c`.
- ▶ `c[i]` accede o devuelve el elemento con índice `i` de la lista.
- ▶ `len(c)` devuelve la longitud de la lista.

Más sobre listas

Algunos ejemplos de listas y operaciones:

- ▶ `[2, 3, 5]` ¿Cuál es el elemento con índice 1?
- ▶ `[2, 3.5, 'cosita', 8]`
- ▶ `[]` lista vacía.
- ▶ `c = [2, 3, 5]` define una lista de nombre `c`.
- ▶ `c[i]` accede o devuelve el elemento con índice `i` de la lista.
- ▶ `len(c)` devuelve la longitud de la lista.
- ▶ `c.append(x)` agrega el elemento `x` al final de la lista `c`.
- ▶ Y muchas operaciones más que iremos viendo...

En resumen, ¿qué es una lista?

En resumen, ¿qué es una lista?

PYnative.com

List in Python

```
L = [ 20, 'Jessa', 35.75, [30, 60, 90] ]
```

↑
L[0]

↑
L[1]

↑
L[2]

↑
L[3]

- ✓ **Ordered**: Maintain the order of the data insertion.
- ✓ **Changeable**: List is mutable and we can modify items.
- ✓ **Heterogeneous**: List can contain data of different types
- ✓ **Contains duplicate**: Allows duplicates data