

Instituto de Formación Técnica Superior N° 18

Tecnicatura Superior en Desarrollo de Software

Desarrollo de Sistemas Orientado a Objetos

- Clase nro 2 -

Nombre y apellido del Profesor: Lic. Eduardo Iberti

1er Cuatrimestre 2025

Desarrollo de Sistemas Orientado a Objetos

- 1 - Python. Características del lenguaje.
- 2 - Software necesario. Descarga e instalación. IDEs.
- 3 - IDLE Python.
- 4 - Otros IDEs. Sublime Text. PyCharm. Visual Studio Code.
- 5 - Variables y Tipos de datos.
- 6 - Expresiones y operadores.
- 8 - Ingresar datos por teclado.
- 9 - Ejercicios Parte 1.
- 10 - Estructuras de Decisión. If else. Match case.
- 11 - Estructuras de Iteración. While. For.
- 12 - Ejercicios Parte 2.

Python. Características del lenguaje

- **Simple:** Python se destaca por su sintaxis clara y concisa, lo que lo hace fácil de aprender y entender para los principiantes. El lenguaje está diseñado para minimizar la cantidad de código necesario para realizar tareas comunes.
- **Orientado a objetos:** es un lenguaje orientado a objetos. Los objetos son fundamentales en Python, lo que significa que todo en Python es un objeto. Esto permite a los programadores crear objetos y trabajar con ellos de manera fácil y eficiente.
- **De Alto nivel:** Python es un lenguaje de programación de alto nivel que facilita su aprendizaje. Python no requiere que comprendas los detalles de la computadora para poder desarrollar programas de manera eficiente.
- **Interpretado:** es un lenguaje interpretado, lo que significa que no es necesario compilar el código antes de ejecutarlo. Esto hace que el proceso de desarrollo sea más rápido y eficiente, ya que el código se puede probar inmediatamente sin la necesidad de compilarlo primero.

Python. Características del lenguaje

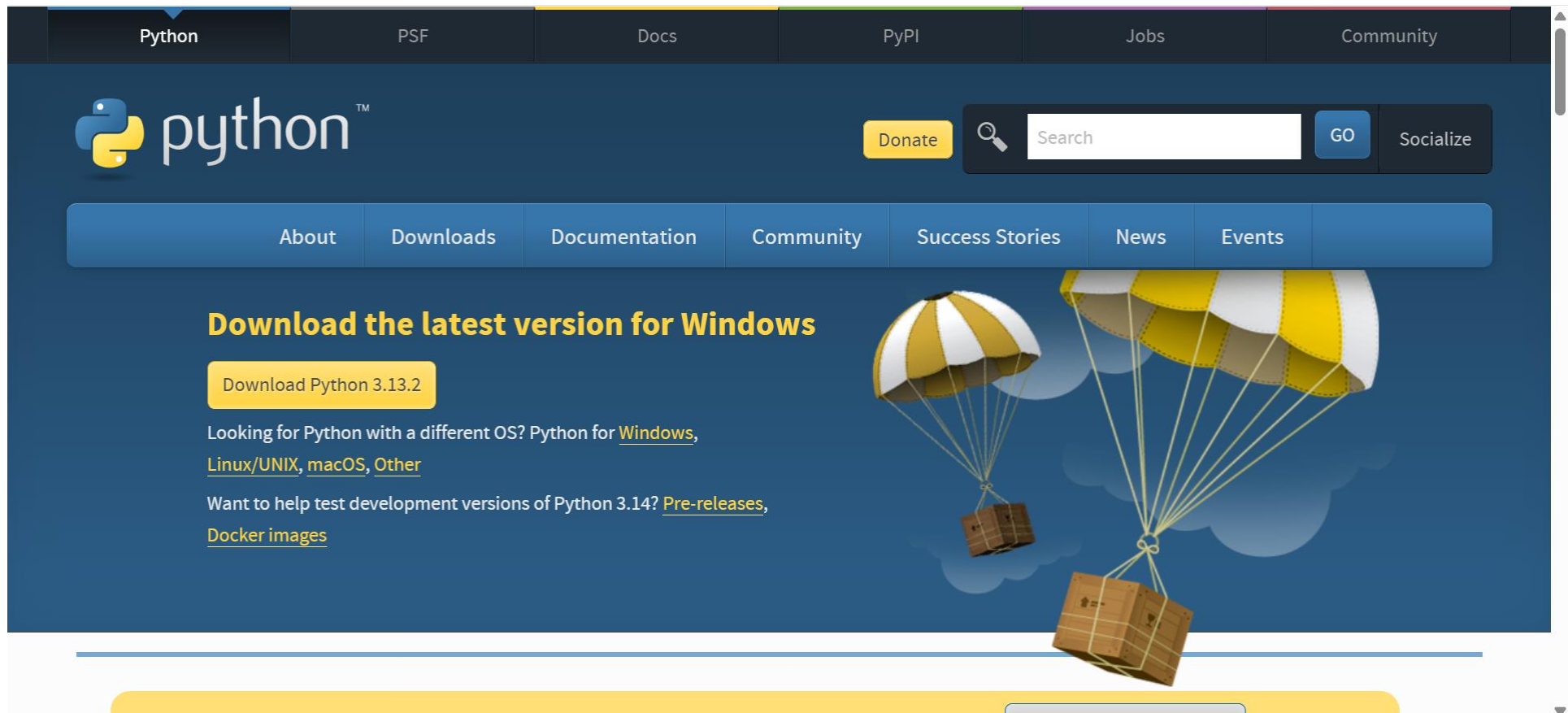
- **Multiparadigma:** admite varios paradigmas de programación, lo que significa que los programadores pueden escribir código en diferentes estilos, según sus preferencias y necesidades. Python admite programación imperativa, programación orientada a objetos y programación funcional.
- **Multiplataforma:** se puede ejecutar en diferentes plataformas, incluyendo Windows, Mac y Linux. Esto significa que los programas escritos en Python pueden ejecutarse en diferentes sistemas operativos sin necesidad de cambiar el código.
- **Tipado dinámico:** es un lenguaje de tipado dinámico, lo que significa que el tipo de una variable no necesita ser declarado explícitamente. Python determina el tipo de una variable en tiempo de ejecución. Esto hace que la programación en Python sea más fácil y rápida, ya que los programadores no necesitan preocuparse por el tipo de datos que se están utilizando en cada momento.

Python. Características del lenguaje

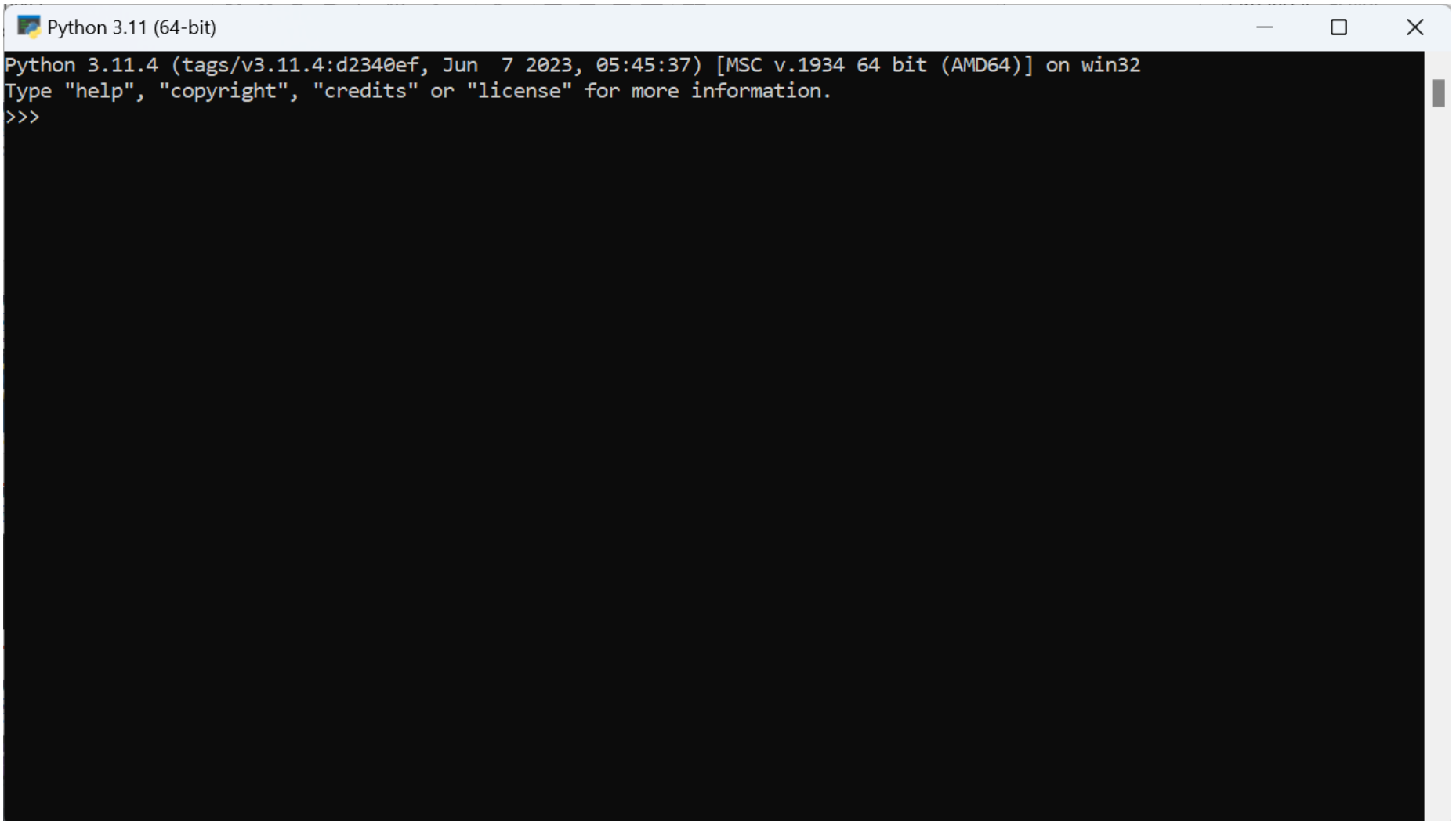
- **De Propósito general:** Python es un lenguaje de propósito general. Significa que se puede usar Python en varios dominios, entre los que se incluyen:
 - **Desktop software**
 - **Mobile apps**
 - **Web applications**
 - **Big data applications**
 - **Testing**
 - **Automation**
 - **Data science, machine learning, AI**

Software necesario. Descarga e instalación

<https://www.python.org/downloads/>



IDLE Python

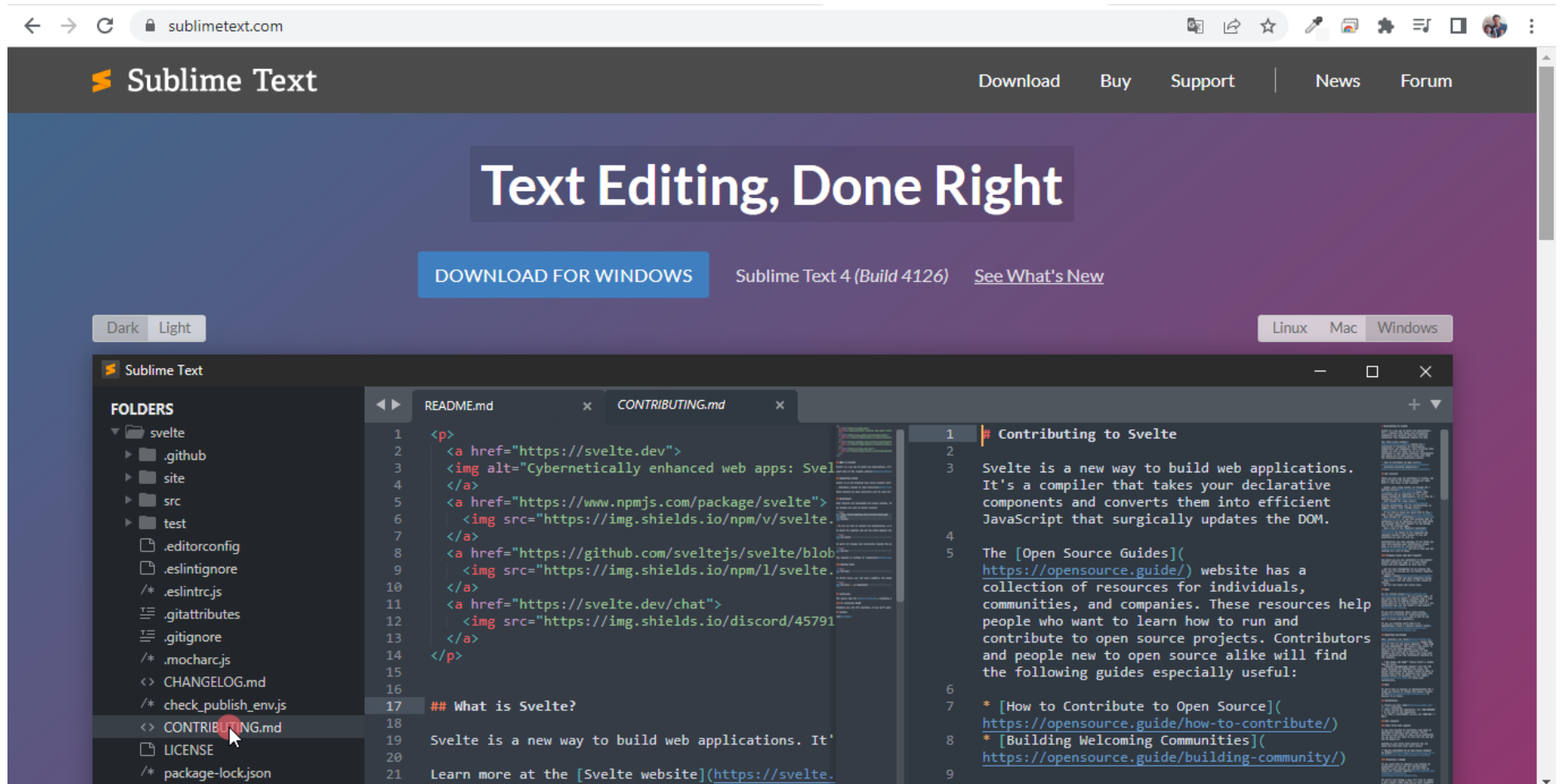


The image shows a screenshot of the Python 3.11.4 (64-bit) IDLE shell window. The window has a title bar with the text "Python 3.11 (64-bit)" and standard Windows window controls (minimize, maximize, close). The main area is a black terminal with white text. The text displayed is: "Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32", followed by "Type 'help', 'copyright', 'credits' or 'license' for more information.", and then the prompt ">>>".

```
Python 3.11 (64-bit)  
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license" for more information.  
>>>
```

Otros IDEs. Sublime Text

<https://www.sublimetext.com/download>



Otros IDEs. PyCharm

<https://www.jetbrains.com/es-es/pycharm/download>



The screenshot shows the PyCharm website homepage. At the top is a dark navigation bar with the 'JET BRAINS' logo on the left and links for 'Para desarrolladores', 'Para equipos', 'Para aprender', 'Soluciones', 'Asistencia', and 'Tienda' in the center. On the right of the bar are icons for search, user profile, shopping cart, and a flag. Below the navigation bar, the word 'PyCharm' is on the left, and on the right are links for 'Novedades', 'Funcionalidades', 'Aprender', a 'Tarifas' button, and a blue 'Descargar' button. The main content area features a large graphic with overlapping yellow, green, and teal geometric shapes. Overlaid on this is the PyCharm logo (a black square with 'PC' and a horizontal line) followed by the text 'PyCharm' in a large, bold font. Below this, it says 'IDE de Python para desarrolladores profesionales'. A black 'DESCARGAR' button is positioned below the text. At the bottom, a small line of text reads 'Versión Professional completa o versión gratis Community'. On the far right edge, there is a vertical 'Sugerencias' (Suggestions) button.

JET BRAINS

Para desarrolladores Para equipos Para aprender Soluciones Asistencia Tienda

PyCharm

Novedades Funcionalidades Aprender Tarifas Descargar

PC PyCharm

IDE de Python
para desarrolladores profesionales

DESCARGAR

Versión Professional completa o versión gratis Community

Sugerencias

Otros IDEs. Visual Studio Code

<https://code.visualstudio.com/download>

The screenshot shows the Visual Studio Code website homepage. The browser address bar displays `code.visualstudio.com`. The website header includes the Visual Studio Code logo, navigation links (Docs, Updates, Blog, API, Extensions, FAQ, Learn), a search bar, and a blue 'Download' button. A banner below the header states: 'Version 1.66 is now available! Read about the new features and fixes from March.'

The main content area features the text 'Code editing. Redefined.' followed by 'Free. Built on open source. Runs everywhere.' Below this is a blue 'Download for Windows' button with a dropdown arrow, and a link for 'Other platforms and Insiders Edition'. A small disclaimer at the bottom left reads: 'By using VS Code, you agree to its license and privacy statement.'

The background of the main content area is a dark-themed screenshot of the Visual Studio Code IDE interface. The IDE shows the 'EXTENSIONS: MARKETPLACE' sidebar on the left with a list of installed and available extensions like Python, GitLens, C/C++, ESLint, and others. The main editor area displays a JavaScript file named 'serviceWorker.js' with code for a service worker. The bottom panel shows a terminal window with the output of a command, indicating that a create-react-app is running on localhost:3000.

Variables

En Python toda variable declarada ha de tener su identificador, pero su tipo de datos es dinámico, y va a tomar el tipo según el valor asignado, por ello, no se pueden tener variables sin inicializar. A continuación, pasamos a describir los tipos de datos primitivos soportados en Python. Cabe destacar que el número de bites asignados a los tipos de datos en Python dependerá de la arquitectura en la que se esté ejecutando y de la longitud de los mismos. Por tanto, los tipos de datos no tienen una longitud fija y contiene mucha más información que el propio valor del dato, ya que todos ellos son objetos, como veremos más adelante.

Para crear una variable utilizamos la instrucción `<nombre> = <expresión>`.

```
>>> x = 5
```

```
>>> nombre = "Juan"
```

Tipos de datos

Int

Almacenan como su propio nombre indica números enteros, sin parte decimal.

Float

Almacenan número reales, es decir números con parte fraccionaria.

Bool

Se trata de un tipo de dato que solo puede tomar dos valores: **"True"** y **"False"**. Es un tipo de dato bastante útil a la hora de realizar chequeos sobre condiciones. En C, hasta C99, no había un dato equivalente y para suplir su ausencia muchas veces se emplean enteros con valor 1 si "true" y 0 si "false". Otros lenguajes como Pascal sí tienen este tipo de dato.

Tipos de datos

Strings

Secuencias de caracteres.

```
| myString = "I am a string"
```

List

Colecciones **ordenadas y mutables** de datos de **diferentes** tipos. Se declara de la siguiente forma:

```
| numeros = [1, 2, 3]
```

Métodos built-in: min, max, len, append, insert, remove, pop, reverse, sort, list, tuple.

Operadores: Concatenación, repeticion, slice, range slice, in, not in.

Tipos de datos

Tuple

Colecciones **ordenadas e inmutables** de datos de **diferentes** tipos.

```
| numeros = (1, 2, 3)
```

Métodos built-in: min, max, len.

Operadores: Concatenación, repeticion, slice, range slice, in, not in.

Set

Colecciones **desordenadas y mutables** de datos de **diferentes** tipos **inmutables**. Esto quiere decir que los objetos mutables (listas o diccionarios) no pueden formar parte de un conjunto. Representa un conjunto matemático

```
| numeros = {1, 2, 3}
```

Métodos built-in: add, update, clear, copy, discard, remove

Operadores: Union(|), intersection(&), difference(-), Symmetric_difference(^),

Tipos de datos

Dictionary

Colecciones **desordenadas** de **parejas clave-valor**.

```
| dict = { key1:value1, key2:value2, ...keyN:valueN }
```

La clave debe ser un objeto **inmutable**. Esto quiere decir que una lista u otro diccionario no pueden ser la clave de un elemento del diccionario. La misma clave no puede aparecer dos veces en un mismo diccionario. Si se intenta insertar una pareja clave-valor en un diccionario que ya contiene dicha clave, se actualizará el valor de la clave y se perderá el valor antiguo.

Métodos built-in: min, max, len, pop, clear, items, keys, values, update.

Operadores: ninguno.

Expresiones y operadores

Operadores lógicos

- **not:** negación unaria.
- **and:** condicional.
- **Or:** condicional.

Los operadores lógicos and y or no evalúan los operadores subsecuentes una vez que el resultado puede ser determinado basado en los anteriores. Por ejemplo:

```
fall = False  
fal2 = False  
tru1 = True  
tru2 = True  
tru1 or fall
```

Esta expression solo evalúa la primera condición y, cómo se cumple, no evalúa la segunda.

Expresiones y operadores

Operadores de igualdad

- **is**: misma identidad. Los dos identificadores apuntan al **mismo** objeto.
- **is not**: diferente identidad. Los dos identificadores apuntan a **diferentes** objetos.
- **==** equivalente. Los dos identificadores apuntan a **objetos con el mismo valor**
- **!=** no equivalente. Los dos identificadores apuntan a **objetos con diferente valor**.

Operadores de comparación

- **<** menor que
- **<=** menor o igual que
- **>** mayor que
- **>=** mayor o igual que.

Se producen excepciones cuando se comparan tipos incompatibles (e.g. un string y un double).

Expresiones y operadores

Operadores aritméticos

- + suma
- - resta
- * multiplicación
- / división
- // división entera
- % módulo.

La multiplicación, división, división entera y módulo tienen preferencia sobre la suma y la resta, tal y como ocurre en las matemáticas. Si se desea una preferencia distinta se debe especificar mediante el uso de paréntesis.

Ingresar datos por teclado

```
name = input("Ingresa tu nombre: ")
```

```
print("Bienvenido" , name)
```

```
entero = int(input("Ingresa tu edad: "))
```

```
print("Tu edad es", entero)
```

Desarrollar Ejercicios 1 y 2 del Classroom

Ejercicio 1

Escribir un programa que pida al usuario que ingrese los tamaños de la base y la altura de un rectángulo y muestre:

1. El perímetro del rectángulo
2. El área del rectángulo

Ejercicio 2

Implementar un programa que pida al usuario que ingrese dos números y muestre la suma, resta, división y multiplicación de ambos.

Control de flujo en Python

El modo de ejecución de un programa en Python en ausencia de elementos de control de flujo es secuencial, es decir una instrucción se ejecuta detrás de otra y sólo se ejecuta una vez. Esto nos permite hacer programas muy limitados; para evitarlo se introducen estructuras de control de flujo. Las estructuras de control de flujo de Python son las típicas de cualquier lenguaje de programación

Sentencias condicionales. If then else

Ejecutan un código u otro en función de que se cumpla o no una determinada condición. Pasemos a ver sus principales tipos.

if then else

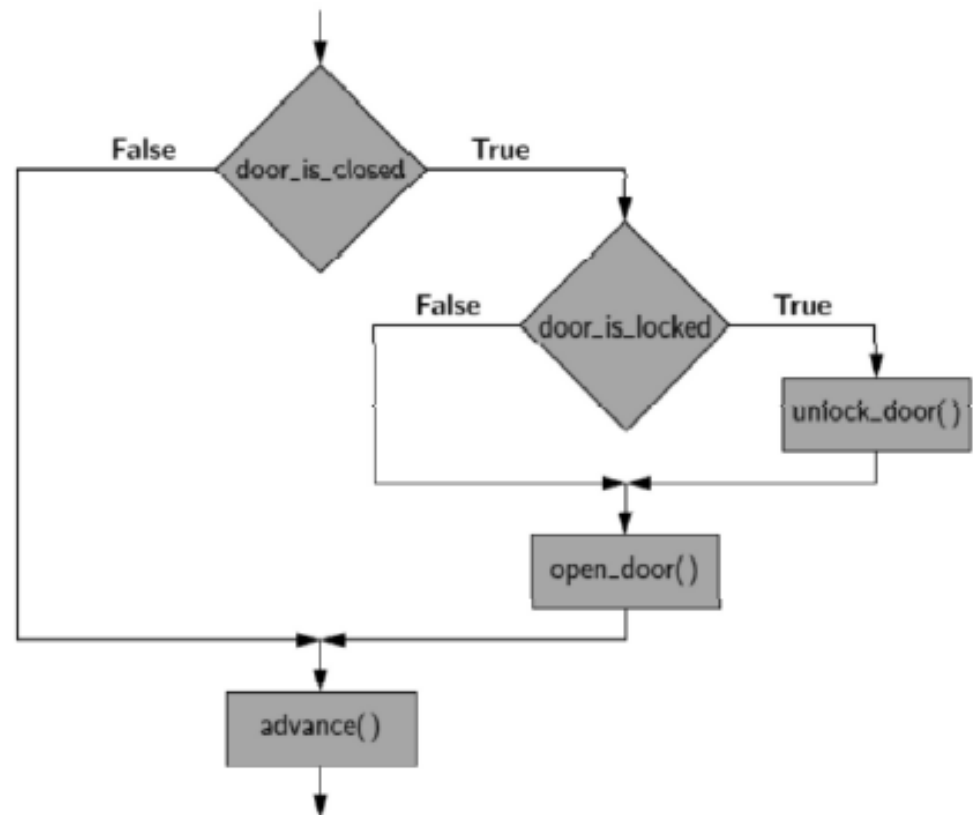
Su modo más simple de empleo es:

```
if first_condition:
    first_body
elif second_condition:
    second_body
...
else:
    last_body
```

Sentencias condicionales. If then else

Cada condición es un valor tipo boolean, y cada cuerpo contiene una o más sentencias para ser ejecutadas condicionalmente. El grupo de sentencias se ejecuta solo si la condición toma un valor true. En caso contrario se sigue ejecutando ignorando el grupo de sentencias.

```
if door_is_closed:  
    if door_is_locked:  
        unlock_door()  
    open_door()  
advance()
```



Sentencias condicionales. Sentencia Match case

Desde la versión 3.10 en adelante, Python añadió la posibilidad de utilizar el condicional switch (**match**) al estilo “switch case” usados en otros lenguajes de programación. Se puede implementar esta función utilizando las palabras clave **match** y **case**.

Ejemplo:

```
lang = input("What's the programming language you want to learn? ")

match lang:
    case "JavaScript":
        print("You can become a web developer.")

    case "Python":
        print("You can become a Data Scientist")

    case "PHP":
        print("You can become a backend developer")

    case "Solidity":
        print("You can become a Blockchain developer")

    case "Java":
        print("You can become a mobile app developer")

    case _:
        print("The language doesn't matter, what matters is solving problems.")
```


Sentencias condicionales. Sentencia Match case

Se puede apreciar que es una sintaxis mucho más limpia que usar múltiples declaraciones elif.

Es importante observar que no se agrega una palabra clave “break” a cada uno de los casos, como se hace en otros lenguajes de programación.

Bucles

Son instrucciones que nos permiten repetir un bloque de código mientras se cumpla una determinada condición. Pasemos a ver sus tipos.

Bucle while

Cuando en la ejecución de un código se llega a un bucle while se comprueba si se verifica su condición, si se verifica se continúa ejecutando el código del bucle hasta que esta deje de verificarse. Su sintaxis es:

```
while condition:  
    body
```

```
data = "String that will be read until a X character  
appears"  
j = 0  
while j < len(data) and data[j] != 'X' :  
    j += 1
```

Bucle for

Su formato es el siguiente:

```
| for element in iterable:  
|     body
```

Veamos un ejemplo:

```
| suma = 0  
| for number in list_numbers:  
|     suma = suma + number
```

Bucle for

El bucle for en Python es muy simple. Sin embargo, este bucle no permite conocer el orden de los elementos en una secuencia. En algunas aplicaciones, conocer el índice de un elemento es necesario, como por ejemplo buscando el máximo elemento de una lista.

En lugar de iterar directamente sobre los elementos, se puede iterar sobre los índices del elemento y acceder a ellos. Por ejemplo:

```
big_index = 0
for j in len(list_numbers):
    if(list_numbers[j] > list_numbers[big_index]):
        big_index=j
```

Break y continue

No se tratan de un bucle, pero sí de sentencias íntimamente relacionadas con estos. El encontrarse una sentencia `break` en el cuerpo de cualquier bucle detiene la ejecución del cuerpo del bucle y sale de este, continuándose ejecutando el código que hay tras el bucle.

Esta sentencia también se puede usar para forzar la salida del bloque de ejecución de una instrucción condicional.

```
found = False
for item in data:
    if item == target:
        found = True
        break
```

`Continue` también detiene la ejecución del cuerpo del bucle, pero en esta ocasión no se sale del bucle, sino que se pasa a la siguiente iteración de este. Se recomienda el uso muy esporádico de estas instrucciones, pero aún hay situaciones donde estos comandos pueden ser efectivos y eficientes para evitar introducir condiciones lógicas complejas.

Desarrollar Ejercicios 3 y 4 del Classroom

Ejercicio 3

Escribe un programa en Python que solicite al usuario 5 números enteros. Luego imprimir el máximo y el mínimo de los valores ingresados. El programa deberá permitir el ingreso de valores iguales.

Ejercicio 4

Escribe un programa en Python que solicite 5 números enteros al usuario. El mismo debe indicar si entre dichos valores hay números duplicados o no, imprimiendo por pantalla “HAY DUPLICADOS” o “SON TODOS DISTINTOS”.

Clase 2

Consultas, dudas sobre
la clase?

