«Talento Tech»

Iniciación a la Programación con Python

CLASE 2

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Buenos Aires Ciudad

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Clase N° 2 | “Hola mundo”

Temario:

- Instalación y configuración del entorno de desarrollo (Visual Studio Code + Python).
- Interfaz de Visual Studio Code.
- "Hola Mundo" en Python.
- Sintaxis básica de Python: variables y tipos de datos simples

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Python

¿Por qué elegimos Python?

Python es una excelente opción para quienes están dando sus primeros pasos en la programación debido a su facilidad de aprendizaje. Su sintaxis clara y comprensible se asemeja al lenguaje natural, lo que permite concentrarse en entender los conceptos clave de la programación sin tener que preocuparse por reglas complejas o una sintaxis confusa. Al usar Python, las y los estudiantes pueden enfocarse en aprender a pensar como programadores, ya que el lenguaje se encarga automáticamente de aspectos técnicos como la gestión de la memoria. Esto facilita la escritura de código, haciendo que el proceso se sienta más como dar instrucciones lógicas en lugar de enfrentar líneas de código difíciles de descifrar.

Además de su facilidad de aprendizaje, Python es un lenguaje muy versátil, lo que lo convierte en una herramienta ideal para explorar diversas áreas de la tecnología. Los principiantes pueden utilizar Python para crear sitios web, analizar datos, desarrollar juegos, automatizar tareas y mucho más. Esta versatilidad significa que, independientemente de cuáles sean sus intereses o metas, Python tiene algo que ofrecerles. Lo que aprenden hoy pueden aplicarlo en una amplia variedad de campos, permitiéndoles descubrir y profundizar en aquellos que más les interesen.

Python es un lenguaje muy versátil.

Un aspecto clave de Python es su comunidad global, diversa y acogedora. Quienes sean principiantes encontrarán una gran cantidad de recursos de aprendizaje disponibles, como tutoriales en línea, cursos, libros y guías interactivas. Además, existen foros (como Stack Overflow o Reddit) donde pueden hacer preguntas y obtener respuestas de personas experimentadas. Esta red de apoyo no sólo facilita el aprendizaje de Python, sino que también fomenta la participación activa del estudiantado en la comunidad, animándoles a compartir sus propios conocimientos a medida que avanzan.

Dominar Python también abre muchas puertas en el ámbito profesional. El conocimiento de Python es muy valorado en campos en rápido crecimiento, como la ciencia de datos, el desarrollo web, el análisis de seguridad, el aprendizaje automático y la inteligencia artificial. La demanda de profesionales que sepan Python sigue en aumento, ya que muchas de las empresas más innovadoras del mundo utilizan este lenguaje para desarrollar sus productos y servicios.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Instalación de Python

- Necesitamos instalar el intérprete Python desde su página oficial.
- Ingresamos a Downloads y hacemos clic en el botón Download Python.
- Asegúrate de descargar la última versión (o como mínimo la versión 3.8.x).

🔗 **Página oficial de Python**

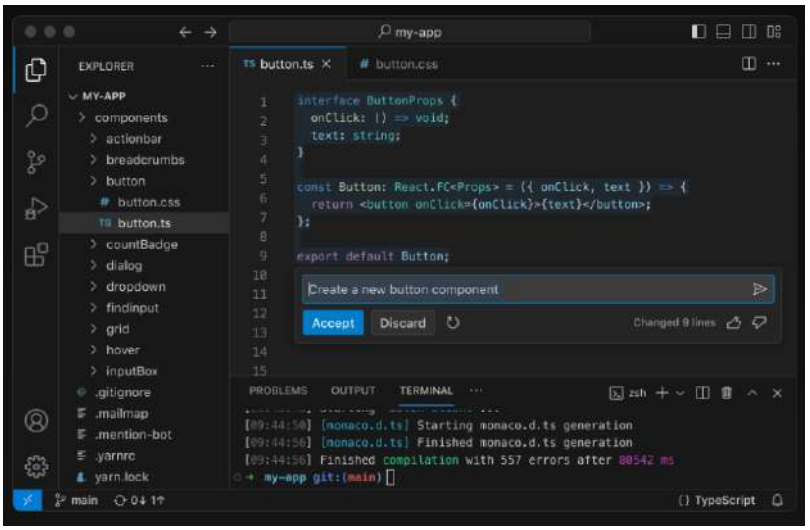
Procedemos con la instalación realizando los pasos típicos para instalar aplicaciones en nuestro sistema operativo.

💡 **Nota:** Si usás Windows, recordá instalar seleccionando la opción *"Add python.exe to PATH".*

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Instalación de VSCode

Para escribir código utilizaremos un editor de texto llamado **Visual Studio Code**.
Lo descargamos desde su página oficial y seguimos los pasos necesarios para su instalación, asegurándonos de crear un ícono de acceso en el escritorio o menú de nuestra computadora.



[Página oficial de VSCode]

La pantalla principal de VSCode

La pantalla principal de VSCode

Visual Studio Code (VSCode) es una herramienta que te ayuda a escribir y gestionar código. Su diseño es bastante simple y está dividido en cinco áreas principales:

Editor: Esta es la parte principal donde escribes tu código. Puedes abrir varios archivos al mismo tiempo y verlos en diferentes secciones, tanto vertical como horizontalmente. Se parece bastante a la ventana de un navegador web y su sistema de pestañas. Puedes borrarlas, moverlas, abrir nuevas, etcétera.

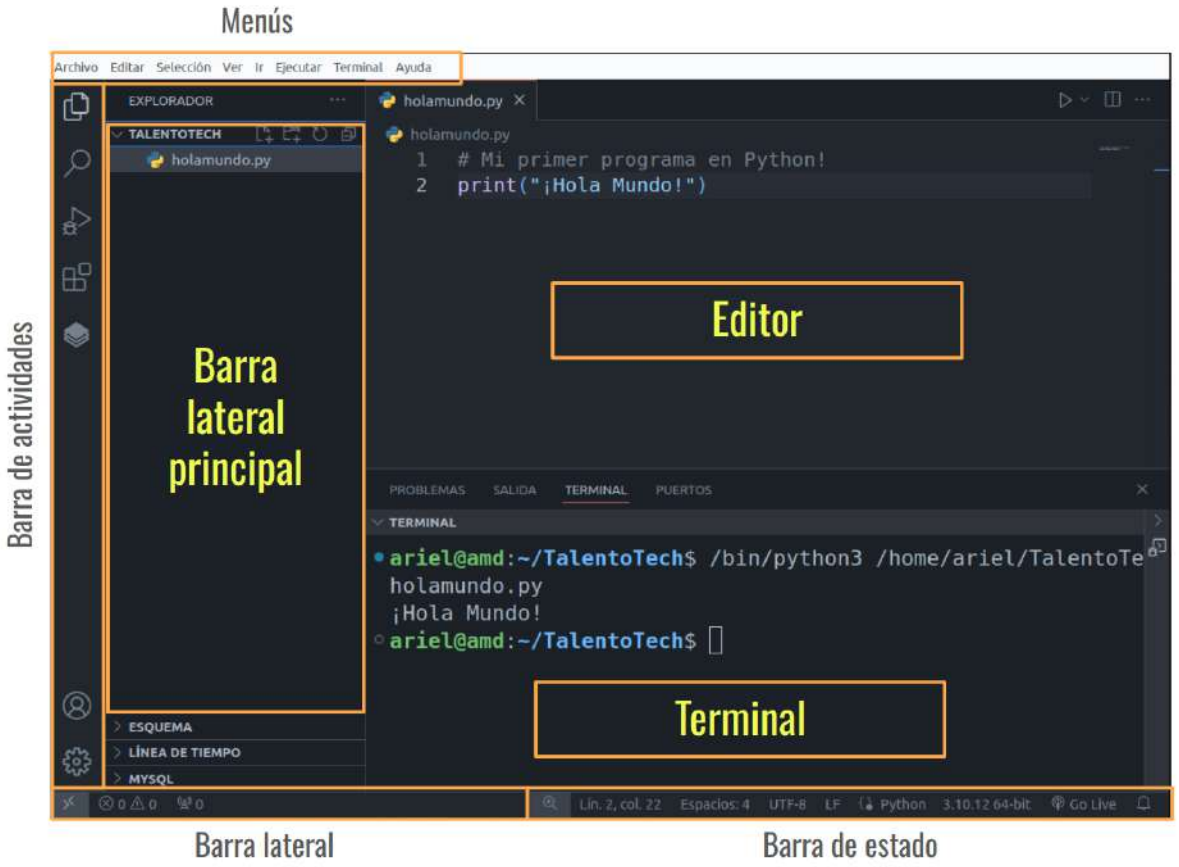
Barra lateral principal: Aquí encontrarás varias herramientas que te ayudan a trabajar en tu proyecto, como un explorador de archivos que te permite ver y acceder a todos los archivos de tu proyecto. Este explorador es similar al explorador de archivos que posee tu sistema operativo, pero está integrado en el editor.

Barra de estado: En la parte inferior de la ventana, una barra de estado muestra información sobre tu proyecto y los archivos que estás editando, como el nombre del archivo y el estado de guardado.No la pierdas de vista: siempre hay información útil en esa zona de VSCode.

Barra lateral: Está ubicada en el lado izquierdo. Te permite cambiar entre diferentes vistas y te muestra indicadores adicionales, como el número de cambios pendientes, si tenemos compartido algún puerto de comunicaciones, etcétera. Quizás no nos sea demasiado útil hasta dentro de un tiempo.

Terminal: Este panel se encuentra en la parte inferior de la ventana, debajo del editor. Muestra información adicional como resultados de pruebas, errores y advertencias, y una terminal integrada donde puedes escribir comandos y ver el resultado de la ejecución de tus programas Python. Puedes mover este panel a los lados izquierdo o derecho si prefieres más espacio vertical en el editor.

Buenos Aires aprende
Agencia de Habilidades para el Futuro



Escribir una línea de código en VSCode

Escribir una línea de código en VSCode

Escribir una línea de código en VSCode es muy parecido a escribir texto en un procesador de textos, como Word o Google Docs. Primero, necesitas abrir VS Code y, dentro de él, abrir un archivo donde quieras escribir tu código. Es como abrir un documento en Word o Docs.

Una vez que tienes tu archivo abierto en VS Code, puedes empezar a escribir código en el área principal del editor. Nuevamente, es igual a empezar a escribir texto en tu documento de Word o Google Docs. Simplemente haz clic en el área donde quieres escribir y comienza

a tipear.

Después de escribir tu código, es importante guardarlo para no perder tu trabajo. En VSCode, puedes hacer esto haciendo clic en el icono de guardar (parece un disquete) en la parte superior de la ventana, o usando el atajo de teclado (Ctrl+S en Windows o Cmd+S en Mac). También puedes activar la opción “Autoguardado” que aparece en el menú “Archivo”, y VSCode se encargará de guardar por vos cada modificación que hagas en el código.

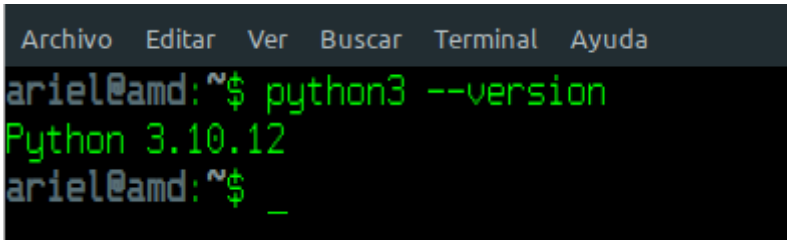
“Hola Mundo”

“Hola Mundo”

Este ejercicio es una excelente manera de comenzar a aprender programación porque te introduce a la estructura básica de un programa en Python y te familiariza con la sintaxis del lenguaje.

Paso 1: Entender qué es un programa
Un programa de computadora es básicamente un conjunto de instrucciones que le dice a la computadora qué hacer. Estas instrucciones están escritas en un lenguaje de programación, que la computadora puede entender y ejecutar. Python es uno de estos lenguajes y es especialmente famoso por ser claro y fácil de leer.

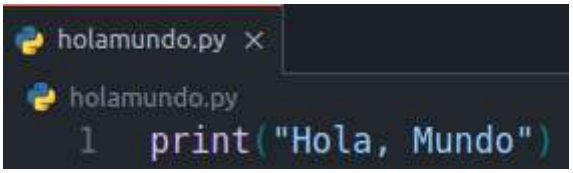
Paso 2: Preparar el ambiente de desarrollo
Antes de escribir tu primer programa, te recomendamos asegurarte de tener Python correctamente instalado. Es usual que algunos ordenadores tengan Python preinstalado, por lo que te sugerimos verificar su instalación abriendo una terminal o línea de comandos y escribiendo lo siguiente:
`python --version` (en Windows)
`python --version` (en Linux).
Si ves un número de versión ya estás en condiciones de comenzar.



En caso de que lo explicado con anterioridad no fuera suficiente, dentro de esta misma unidad en el aula virtual, vas a poder encontrar un pequeño tutorial en video que te explica cómo instalar Python.

Paso 3: Escribir el programa "Hola, mundo"
Para escribir tu programa utilizaremos nuestro entorno de desarrollo integrado (IDE), mejor conocido como **Visual Studio Code.**
En nuestro editor podremos ver cómo se ubican las distintas líneas de código numeradas una abajo de la otra. Nos ubicamos sobre la primera y utilizaremos la palabra reservada **“print”**. Al tipearla, pasará a ser de otro color y se separará de los paréntesis donde escribiremos “Hola, Mundo”.

De esta manera se verá lo descrito anteriormente:



Paso 4: Explicación del código
print(): Esta es una función incorporada en Python que se utiliza para enviar datos a la salida estándar (generalmente, tu pantalla). Aquello que queramos que se “imprima en pantalla” se coloca entre paréntesis y comillas.

"Hola, mundo": Este es el texto que queremos que aparezca en la pantalla. El texto debe estar entre comillas (pueden ser simples - **'** - o dobles - **"** -) para que Python lo reconozca como una cadena de caracteres. Esta concatenación de carácteres alfabéticos es lo que se denomina *string* y, además, es el tipo de dato que Python utiliza para almacenar texto.

Paso 5: Ejecutamos el programa
Para ejecutar tu programa tendrás que guardar el archivo con la extensión **.py**, por ejemplo de la siguiente manera: **holamundo.py**. Luego, presioná el ícono “play” (▸) en Visual Studio Code y deberías ver el mensaje Hola, Mundo en la pantalla. Así:

```
Hola, Mundo
```

¿Por qué es importante este programa?

¿Por qué es importante este programa?
"Hola, Mundo" es un rito de iniciación en el aprendizaje de la programación porque es sencillo pero a su vez demuestra el ciclo de edición, compilación (interpretación en el caso de Python) y ejecución de un programa. A pesar de su simplicidad, te enseña conceptos fundamentales como la sintaxis básica de un lenguaje (en este caso el uso de las comillas o paréntesis), cómo las funciones realizan tareas específicas (print) y cómo se produce la salida de un programa (la impresión en pantalla del mensaje deseado).

Mostrar “Hola mundo” en pantalla exitosamente representa un primer gran logro y también te motiva a seguir aprendiendo. Es el primer paso hacia la creación de programas más complejos y el desarrollo de habilidades de programación más avanzadas.

La función print()

La función print()
La función print() en Python es esencial para la salida de datos, permitiendo a los programadores mostrar información en la terminal de forma legible. Su flexibilidad y versatilidad hacen de print() una herramienta poderosa tanto para la depuración como para la interacción con el usuario. A continuación, profundizaremos en su funcionamiento, opciones de formato y ejemplos prácticos para ilustrar su uso efectivo.

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Podemos pensar en print() como una herramienta especial de Python. Cada vez que escribimos print() y colocamos algo entre los paréntesis, esa herramienta realiza una tarea específica: mostrar en pantalla el texto o los números que le hemos proporcionado. Esta herramienta, que llamamos función, es como un pequeño programa dentro de tu programa. Python tiene muchas de estas herramientas, cada una con un propósito específico.

Una función es simplemente un conjunto de instrucciones que Python ya conoce y que puede ejecutar cuando se lo pedimos. Cuando utilizamos print(), estamos diciéndole a Python que ejecute la función print. Podemos pensar en una función como una receta. Cuando seguimos una receta, estamos realizando una serie de pasos que nos llevan a un resultado, como preparar un plato de comida. En Python, las funciones hacen lo mismo: siguen una serie de pasos para realizar una tarea específica.

Aunque aún no conocemos todos los detalles, la idea principal es que una función es algo que puedes usar para realizar tareas en tu programa de manera sencilla. Dentro de poco aprenderemos a crear nuestras propias funciones, pero por ahora, piensa en print() como un ejemplo de cómo una función puede hacer que los programas sean útiles y poderosos.

Entrada / Salida de datos: La función print()

Entrada / Salida de datos: La función print()

La función print() es utilizada para imprimir datos en la terminal. Acepta múltiples parámetros, los cuales pueden ser variables, literales (textos directos), o una combinación de ambos separados por comas. Podés delimitar las cadenas de texto con comillas simples (') o dobles ("), según prefieras. Por defecto, después de imprimir, print() añadí un salto de línea, moviendo el cursor a la línea siguiente. Si se invoca print() sin pasarle argumentos, simplemente imprimirá una línea en blanco. Veamos las distintas posibilidades y cuáles serían sus resultantes o “salidas”:

```
nombre = "Mundo"
print("Hola,", nombre)
```

```
# Salida: Hola, Mundo
```

La línea de código de color verde, al final del código anterior, es un **comentario.** En Python, los comentarios se crean utilizando el símbolo #. Cuando escribís "#" al comienzo de una línea o en cualquier parte de ella, todo lo que sigue a la derecha del símbolo se considerará un comentario. Python ignora cualquier texto que esté precedido por #, lo que significa que esos comentarios no afectarán en absoluto el funcionamiento de tu código. Los comentarios son útiles para dejar notas, explicar lo que hace una parte del código, o para desactivar temporalmente una línea de código sin eliminarla. Por ejemplo, si deseás explicar qué hace una sección específica de tu programa podés agregar un comentario justo encima de esa sección o al final de la línea de código correspondiente. Es importante usar comentarios de manera clara para que otras personas (o vos mismo en el futuro) puedan entender fácilmente lo que el código está haciendo y el propósito detrás de ciertas decisiones. En este caso, estamos usando los comentarios para indicar qué se mostrará en la terminal al ejecutar el ejemplo propuesto.

Controlando el final de la línea con end

El parámetro end de print() permite especificar qué se debe imprimir al final de la cadena. Por defecto, **end='\n'**, lo que significa que cada llamada a print() termina con un salto de línea. Modificar este comportamiento puede ser útil para imprimir en la misma línea o añadir un delimitador específico.

```
print("Hola,", end=' ')
print("Mundo")
# Salida: Hola, Mundo
```

Insertando nueva línea y tabulaciones

\n (Nueva línea): Inserta un salto de línea en el punto especificado de la cadena.

```
print("Hola\nMundo")
# Salida:
# Hola
# Mundo
```

\t (Tabulación): Añade una tabulación o un conjunto fijo de espacios en blanco, útil para alinear texto.

```
print("Nombre:\tJuan\nEdad:\t30")
# Salida:
# Nombre: Juan
# Edad:   30
```

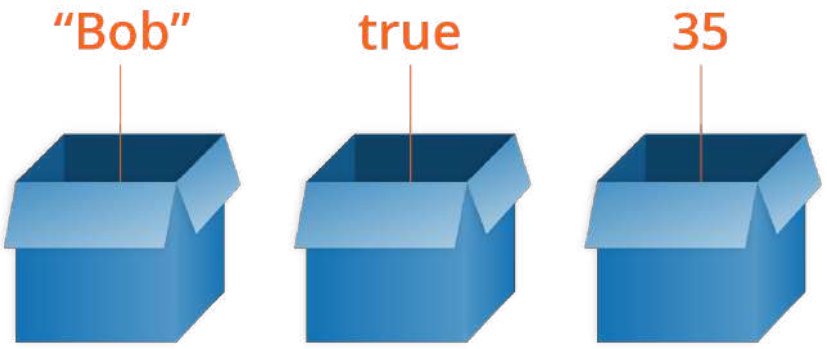
Consideraciones al usar \t
La tabulación (\t) inserta un número fijo de espacios, que puede variar según el entorno donde se visualice el texto pero comúnmente equivale a 4 u 8 espacios. Su efectividad para alinear texto depende de la longitud de las cadenas que lo preceden o siguen. Si las cadenas son más largas o más cortas de lo esperado la alineación puede no ser la deseada.

```
print("Producto\tPrecio")
print("Manzanas\t$ 1.00")
print("Peras\t\t$ 1.50")
# Salida:
# Producto        Precio
# Manzanas        $ 1.00
```

```
# Peras             $ 1.50
```

La función print() es una de las herramientas más básicas y poderosas en Python, esencial para la salida de nuestros datos. A través de sus parámetros como **end** y la utilización de caracteres especiales como **\n** y **\t**, permite un control detallado sobre cómo se formatea y presenta la información en la terminal.

Variables en Python



Variables en Python

El concepto de variable es uno de los pilares de la programación y es común a todos los lenguajes por lo que entenderlo será fundamental para todas las personas que deseen iniciarse en el desarrollo de software. Las mismas son centrales ya que nos permiten que los programas sean dinámicos y flexibles. ¡Comencemos!

¿Qué es una variable?

En programación, una variable es como una "caja" donde podremos ingresar datos de distinto tipo y origen. Es decir, se tratará de un contenedor en la memoria de tu computadora donde almacenaremos datos temporales que tu programa pueda usar y modificar mientras se ejecuta. Una variable es en resumidas cuentas un contenedor que tiene una etiqueta con un nombre único; esta etiqueta te permite acceder al contenido de la

caja o cambiarlo.

Por ejemplo, si yo estoy desarrollando la programación de un videojuego crearé o *"declararé"* (en términos técnicos) la variable **"puntos"**, donde mi programa guardará los puntos que cada equipo vaya acumulando. Esta misma variable (que guardará un tipo de dato numérico) me servirá para, por ejemplo, finalizar el juego. Imaginemos que nuestro videojuego es de volleyball, el pseudocódigo para ganar el set podría ser algo como: *"si puntos es igual a 25, entonces finalizar juego…"*

¿Cómo funcionan las variables en Python?

¿Cómo funcionan las variables en Python?

Python es un lenguaje de programación de **tipado dinámico**, lo que significa que no necesitás declarar explícitamente el tipo de dato que una variable va a almacenar (como números, texto, listas, etc.). Esto hace que trabajar con variables en Python sea especialmente sencillo.

Veamos un ejemplo:

```
mensaje = "Hola, Mundo"
print(mensaje)
```

En este ejemplo, **"mensaje"** es una variable que contiene el texto *"Hola, Mundo"*. Dentro de mi contenedor llamado **mensaje** almacenaré aquello que deseo imprimir en pantalla, la cadena de texto *"Hola, Mundo"*. La **función print()** entonces se utiliza para mostrar el contenido de la variable mensaje en la pantalla.

El **operador de asignación** en Python y en muchos otros lenguajes de programación es el símbolo =: se utiliza para *asignar un valor a una variable.* La forma en que funciona es bastante sencilla pero fundamental para casi cualquier programa. La estructura básica de una asignación es:

```
variable = valor
```

Donde variable es el nombre de la variable que estás creando o actualizando, y valor es el dato que deseas almacenar en dicha variable.

Formas de asignación

Formas de asignación

Asignación única: Cada vez que usás el operador de asignación podés establecer o cambiar el valor de una variable. Si la variable ya tiene un valor determinado, el mismo será reemplazado por el nuevo valor asignado.

En el primer ejemplo, el número **30** se asigna a la variable **edad.** En el segundo ejemplo, la cadena de caracteres **"Juan"** se asigna a la variable **nombre:**

```
edad = 30
nombre = "Juan"
```

Asignaciones múltiples: Python también permite la asignación múltiple, lo que te permite asignar valores a varias variables en una sola línea de código. Veamos cómo se vería el código de las asignaciones múltiples de varias variables:

```
x, y, z = 1, 2, 3
```

Aquí, **x** recibe el valor **1**, **y** el valor **2**, y **z** el valor **3**.

Asignación a variables existentes: También podés utilizar el valor actual de una variable para cambiarlo. ¿Cómo? En el ejemplo siguiente, el contador es igual a “0”. En la siguiente línea de código **contador** es igual a **contador (“0”) más 1**, por lo que el valor final de la variable será **1**. Esto es útil para actualizar el contenido de una variable basándose en su valor anterior:

```
contador = 0
contador = contador + 1 # Actualiza contador a 1
```

Cambiando el valor de una variable

Cambiando el valor de una variable

Una de las características de las variables es que el dato que contienen puede cambiar o variar a lo largo del tiempo en un programa. Por ejemplo:

```
numero = 10
print(numero) # Imprime 10

numero = numero + 5
print(numero) # Imprime 15
```

Inicialmente, la variable número contiene el valor 10. Luego, le sumamos 5, y el nuevo valor de número se convierte en 15.

El operador de asignación es crucial porque permite:

- **Almacenar valores:** Permite guardar valores para usarlos más tarde en tu programa.
- **Actualizar valores:** Da lugar a cambios en los valores de las variables a medida que tu programa se ejecuta, lo que es esencial para la lógica de programación dinámica y los cálculos.

El operador de asignación **=** es uno de los aspectos más básicos y esenciales de Python, permitiendo la manipulación y gestión de datos dentro de un programa. Su comprensión y uso correcto son fundamentales para que la programación sea eficaz.

En Python, una variable puede comenzar almacenando un tipo de dato, como un número, y luego cambiar para almacenar un tipo de dato diferente, como una cadena de texto o una lista. Esto es posible gracias al **tipado dinámico** de Python. Por ejemplo:

```
dato = 100
print(dato) # Imprime 100

dato = "Cien"
print(dato) # Imprime Cien
```

En este ejemplo, “dato” inicialmente almacena un valor numérico (100), pero luego cambia para almacenar una cadena de texto ("Cien").

Buenos Aires aprende
Agencia de Habilidades para el Futuro

Nombres de las variables

Nombres de las variables

Elegir nombres adecuados para las variables es crucial en la programación porque mejora la legibilidad del código y facilita su mantenimiento. Al nombrar variables en Python, hay varias reglas y buenas prácticas que te aconsejamos seguir para asegurarte de que tu código sea claro y fácil de entender.

1. **Caracteres permitidos:** Los nombres de las variables pueden incluir letras (mayúsculas y minúsculas), números y guiones bajos (_). Sin embargo, deben comenzar con una letra o un guión bajo. Por ejemplo, _miVariable o miVariable son válidos, pero 1miVariable no lo es.
2. **Palabras reservadas:** Python reserva un conjunto de palabras para su sintaxis y estructura de lenguaje, conocidas como palabras reservadas. Estas no pueden ser utilizadas como nombres de variables. Algunas de estas palabras incluyen *if, for, class, return, global,* entre otras. Intentar usar una palabra reservada como nombre de variable resultará en un error de sintaxis.
3. **Sensibilidad a mayúsculas y minúsculas:** Python diferencia entre mayúsculas y minúsculas. Esto significa que **variable**, **Variable** y **VARIABLE** serían consideradas tres variables distintas. Aunque técnicamente es posible tener variables con nombres tan similares, esto no es recomendable ya que puede prestarse a la confusión.

Buenas prácticas a la hora de nombrar tus variables:

Buenas prácticas a la hora de nombrar tus variables:

Elegir nombres adecuados para las variables es muy importante para escribir código que sea claro y fácil de mantener. Una de las prácticas más recomendadas es **usar nombres significativos y descriptivos**, ya que estos deben reflejar de manera clara los datos que contienen. Por ejemplo, optar por edad o numero_de_edad en lugar de simplemente e o n hace que el código sea más legible y comprensible, no solo para quien lo escribe, sino también para cualquier otra persona que trabaje con él en el futuro.

Cuando se trata de nombres largos, es común utilizar minúsculas y guiones bajos para separar las palabras. Esta convención, conocida como *snake_case*, es la recomendada por la **guía de estilo de Python, PEP 8.** Nombres como nombre_usuario o contador_intentos son ejemplos de cómo esta práctica mejora la legibilidad del código.

Es igualmente importante evitar nombres demasiado generales o ambiguos. Aunque es posible usar nombres cortos como “x” o “n", estos no proporcionan suficiente información sobre el propósito o el tipo de datos que la variable contiene, lo que puede dificultar la comprensión del código.

Además, en Python, el uso de un guión bajo (_) al principio del nombre de una variable, como en _privado, tiene un significado especial. Esto refiere al alcance de la variable (o método): significa que ésta es privada y está destinada para uso interno dentro de una clase o módulo. Aunque Python no aplica esta privacidad de manera estricta, es una convención ampliamente respetada entre las personas que desarrollan software.

Seguir estas reglas y buenas prácticas en la elección de nombres de variables no sólo contribuye a la claridad del código, sino que también facilita su mantenimiento y colaboración con colegas. Un código bien nombrado es más fácil de entender, depurar y ampliar, lo que resulta en un trabajo más eficiente y efectivo para todas las partes involucradas.

Tipos de datos simples en Python

Tipos de datos simples en Python

Entender los tipos de datos es fundamental para manejar variables y realizar operaciones eficientemente. Los tipos de datos determinan **la clase de valor** que una variable puede almacenar, cómo se almacena ese valor en la memoria y las operaciones que son posibles realizar con esos valores. Vamos a profundizar en los tipos de datos básicos en Python y dejaremos para más adelante algunos tipos de datos más complejos.

1. Número Entero (int)
Los enteros son números sin parte decimal, que pueden ser positivos o negativos. Python permite trabajar con ellos para realizar operaciones matemáticas básicas como suma, resta, multiplicación y división, entre otras.

Ejemplos:

```
edad = 25
numero_de_estrellas = -100
años = 2023
```

Estos ejemplos muestran cómo asignar un número entero a una variable. Se pueden realizar operaciones matemáticas como edad + 5 para obtener 30.

2. Número Decimal (float)
Los números decimales o flotantes contienen una parte fraccionaria separada por un punto. Son útiles para representar valores que requieren mayor precisión, como medidas o cantidades financieras.

Ejemplos:

```
altura = 1.75
temperatura = -20.3
precio = 19.99
```

Cada ejemplo asigna un número decimal a una variable. Las operaciones con flotantes siguen las reglas de la aritmética de punto flotante.

3. Cadena de Texto (str)
Las cadenas de texto son secuencias de caracteres usadas para almacenar datos textuales. Python las maneja con mucha flexibilidad, permitiendo operaciones como concatenación, multiplicación y acceso a elementos individuales.

Ejemplos:

```
nombre = "Juan"
```

```
mensaje = "Hola, Mundo"
letra = 'A'
```

Aquí, nombre y mensaje son variables que almacenan cadenas de texto. Se pueden combinar cadenas con el operador +, como en "Hola " + "Mundo" para obtener "Hola Mundo".

4. Booleano (bool)
Los booleanos representan dos valores: Verdadero (True) y Falso (False). Son muy utilizados en estructuras de control para tomar decisiones en función de condiciones.

Ejemplos:

```
esMayorDeEdad = True
examenAprobado = False
```

Veremos que estos valores se utilizan frecuentemente en expresiones condicionales, para ejecutar un bloque de código si la condición es verdadera.

Los tipos de datos en Python son la base sobre la cual se construyen las estructuras de datos y se realizan operaciones. Comprender los tipos de datos básicos es esencial para cualquier programador de Python, ya que facilita la manipulación de datos y la implementación de lógica compleja en los programas. A medida que avances, trabajaremos con tipos de datos más complejos (como las listas y diccionarios) y sus métodos asociados, lo que te permitirá construir programas más potentes y eficientes.

Ejercicio Práctico #1:

Lista de la compra

Usando la instrucción **print()** y lo visto en la clase, escribir un programa que muestre en la terminal la lista de productos que necesitamos comprar en el supermercado.

💡**Tips:**

- Podés usar tabuladores para ordenar la lista.

- Recordá que si necesitas una línea en blanco, puedes usar “print()”.

Ejercicio Práctico #2

Ingreso promedio

Escribir un programa que guarde en variables el monto del ingreso de cada uno de los primeros seis meses del año.
Luego, calcular y guardar en otra variable el promedio de esos valores.
Por último, mostrar una leyenda que diga “El ingreso promedio en el semestre es de xxxxx” donde “xxxxx” es el valor calculado.


Buenos Aires
aprende
Agencia de Habilidades para el Futuro
Buenos
Aires
Ciudad