



Universidad
Tecmilenio®



Visualización y programación en Python

Tema 1. Introducción a la programación con Python





Actualmente en todo tipo de industria se quiere de algún software para apoyar alguna de las funciones, tareas, procesos o incluso productos.

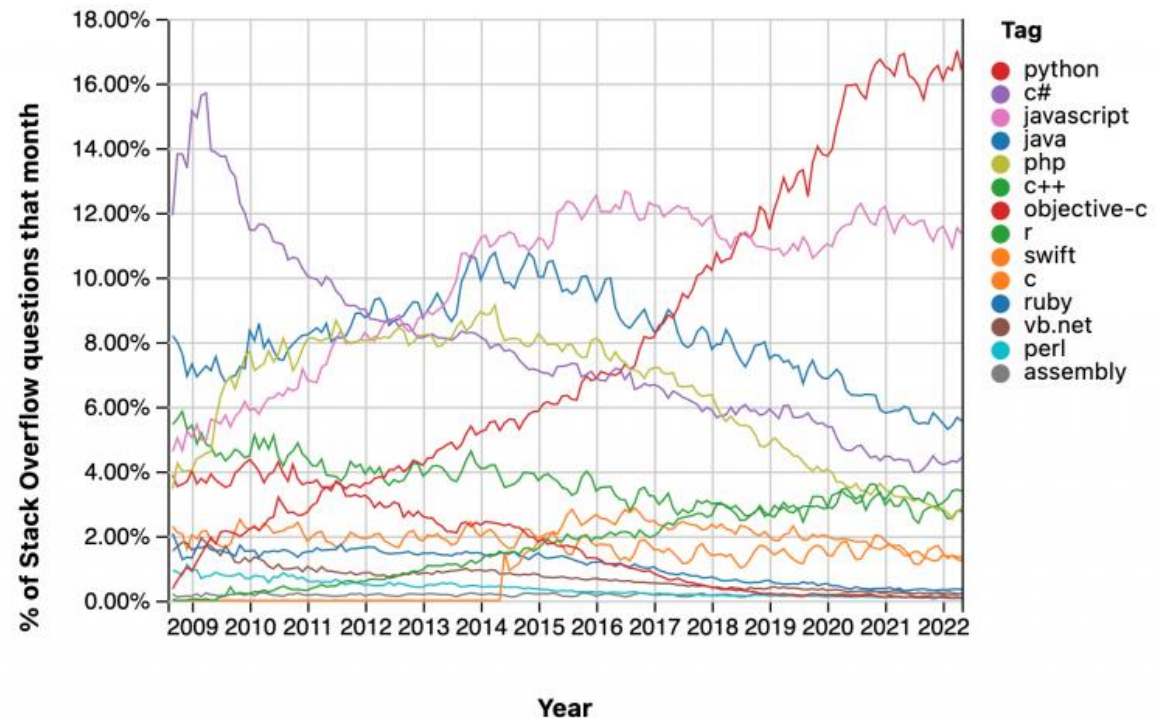
Así que pensar en un lenguaje de programación que nos permita al mismo tiempo hacer un análisis de datos, automatización de tareas, *web scraping*, *machine learning*, *data science*, desarrollo de videojuegos, interfaces gráficos o finanzas, ahora es posible gracias a **Python**, su versatilidad y sencillez.



Su inicio se remonta a los años 90's. Desarrollado por el investigador holandés **Guido Van Rossum** mientras trabajaba en el centro de investigación CWI (Centrum Wiskunde & Informatica) de Ámsterdam.

Python es uno de los lenguajes más usados y populares en la actualidad

Figura 1. Lenguajes de los que se tienen más preguntas



Fuente: Stack Overflow. (2022). *Stack Overflow Trends*. Recuperado de <https://insights.stackoverflow.com/trends?tags=python%2Cjavascript%2Cjava%2Cc%23%2Cphp%2Cc%2B%2B>.

El **lenguaje Python** se refiere a **19 principios** que influyen en el diseño del software y que buscan enmarcar lo que fue importante para el creador de lenguaje al momento de ir concibiéndolo.

1. Bello es mejor que feo.
2. Explícito es mejor que implícito.
3. Simple es mejor que complejo.
4. Complejo es mejor que complicado.
5. Plano es mejor que anidado.
6. Espaciado es mejor que denso.
7. La legibilidad es importante.
8. Los casos especiales no son lo suficientemente especiales como para romper las reglas.
9. Sin embargo, la practicidad gana a la pureza.
10. Los errores nunca deben pasar silenciosamente.
11. A menos que se silencien explícitamente.
12. Frente a la ambigüedad, evitar la tentación de adivinar.
13. Debería haber una, y preferiblemente solo una, manera obvia de hacerlo.
14. A pesar de que esa manera no sea obvia a menos que seas holandés (el creador lo era).
15. Ahora es mejor que nunca.
16. A pesar de que nunca es muchas veces mejor que ahora mismo.
17. Si la implementación es difícil de explicar, es una mala idea.
18. Si la implementación es fácil de explicar, puede que sea una buena idea.
19. Los *namespaces* son una gran idea, ¡tenemos más de esos!

Características particulares de Python

- **Es un lenguaje interpretado**, no compilado.
- **Usa tipado dinámico**, lo que significa que una variable puede tomar valores de distinto tipo.
- **Es fuertemente tipado**, lo que significa que el tipo no cambia de manera repentina.
- **Es multiplataforma.**



Atributos de Python:

Lenguaje de alto nivel.

Propósito general.

Alta gestión de memoria.

Sintaxis sencilla.

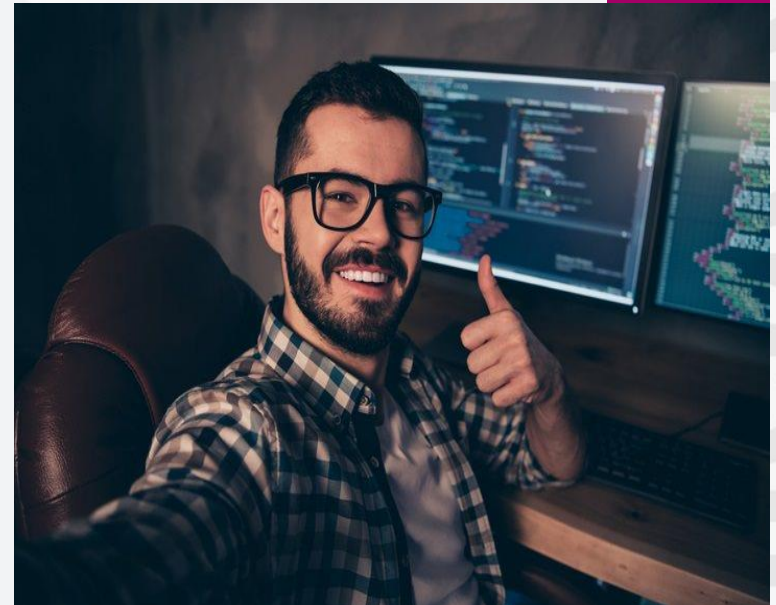
Semántica dinámica.

Orientado a objetos.



Características que hacen **popular** a Python:

- Es fácil de aprender.
- Tiene una comunidad activa y solidaria.
- Es flexible.
- Ofrece soluciones de desarrollo Web.
- Se adapta a la analítica y ciencia de datos.
- Es eficiente, rápido y confiable.
- Se usa con tecnología IoT.
- Potencia la automatización.
- Es un lenguaje académico.



Explicación

Empresas que usan Python

Pinterest	Python cumple una función sumamente importante para la aplicación, ya que es el motor de la plataforma. Un dato curioso es que ha utilizado el lenguaje de programación de Python desde el primer día de su existencia. Además de que Pinterest es un ejemplo del uso de Python, también utiliza Django para el uso de frameworks y darle mejoras progresivas al código de la aplicación.
Panda 3D	Es el motor de videojuegos de Walt Disney Company. Este programa de software libre u open source, fue lanzado en el 2002 y hasta el día de hoy se sigue utilizando para el desarrollo de juegos. Este programa en Python refleja el gran potencial que tiene este lenguaje de programación para el desarrollo de juegos en 3D. En este sistema se muestra cómo se puede combinar con otros lenguajes de programación como el C++ para ofrecer una funcionalidad más completa. La ventaja es que los programadores obtienen mayores ventajas de desarrollo rápido y la administración avanzada de memoria para desarrollar y actualizar Panda3D.
Dropbox	Está principalmente soportado con códigos de Python. En 2012 Dropbox contrató al creador de Python, Guido van Rossum, retirando su talento de Google y permaneciendo ahí hasta 2019. Dropbox es un programa de Python. Tanto la aplicación como el cliente desktop están en su totalidad en este lenguaje. Aunque se va sustituyendo con el lenguaje Go, cabe resaltar que también los drivers de web se implementaron con Python permitiendo que funcionen eficientemente bajo cualquier sistema operativo.
Spotify	Su página esta construida con WordPress y la aplicación con Python. Esto le permite realizar análisis de datos y algunos servicios de backend. Se calcula que más de 6000 procesos están en operación en Spotify con códigos en Python. Esto refleja que alrededor del 80% de estos servicios están escritos en Python.

Netflix	Es uno de los lenguajes preferidos en Netflix sobre todo para el análisis de datos, específicamente ayuda a soportar el entrenamiento para aprendizaje automático y para sus principales algoritmos como de recomendación, para la elección del arte que muestra, de marketing y más. Se usa el código en Python por su mayor diversidad, sintaxis concisa y limpia, su enorme comunidad de desarrolladores y la calidad de las librerías externas para resolver problemas de programación.
Uber	Lo que hacen es que los programas en Python se combinen con los otros lenguajes de programación como Node.js, Go y Java para un mayor desarrollo de funciones en su aplicación móvil. También reconoce que Python le ha servido para desarrollar mayor tecnología avanzada en relación con el análisis de Big data, ya que este lenguaje de programación es utilizado para construir estas funciones nuevas.
Instagram	Los desarrolladores de Instagram mencionan que Python, es de fácil desarrollo, muy simple de gestionar y perfecto para soportar el gran volumen de visitas diarias que Instagram recibe. La facilidad de programar en Python apoya al equipo enfocarse en la experiencia al usuario y ofrecer constantes mejoras en de manera ágil y segura.
Reddit	Originalmente fue desarrollado en Lisp, pero se reescribió en Python para obtener una mayor flexibilidad de desarrollo y acceso a una gran cantidad de bibliotecas de código.
Google	Python es el idioma oficial desde sus inicios, las razones fueron su flexibilidad, rápido desarrollo, escalabilidad y excelente rendimiento. Google App Engines es su ejemplo más concreto, este programa permite habilitar herramientas de administración de sistemas.
YouTube	El uso de Python es para diferentes propósitos en toda la plataforma; sin embargo, la que destaca más para la marca es la velocidad que permite el desarrollo de funciones en un tiempo récord.





Mapa mental

Objetivo: Describir el lenguaje de programación Python, sus características, ventajas, aplicaciones y alcances.

Instrucciones:

- Para realizar esta actividad es importante que consultes el contenido del tema.
- Con base en lo aprendido elabora un **mapa mental** que incluya la historia, el concepto, características y ventajas del lenguaje de programación.
- Incluye una reflexión sobre el impacto que ha tenido en el área laboral actual y ejemplifica brevemente con un caso de éxito.
- Utiliza colores, elementos gráficos y tipografías a través de una plataforma digital que permita la mayor comprensión de lo expuesto.
- Máximo 2 cuartillas.



Existen una gran variedad de extensiones para **Python**, que te permiten usar librerías y funciones anteriormente desarrolladas, por ejemplo Selenium que pueden facilitar un simulador de carga u operaciones vía web con Python desde tu computadora, pudiendo hacer un ambiente de pruebas o bien automatizando acciones rutinarias y aburridas en las que no quieres invertir tiempo.

¿Qué uso te gustaría darle?



Stack Overflow. (2022). *Stack Overflow Trends*. Recuperado de <https://insights.stackoverflow.com/trends?tags=python%2Cjavascript%2Cjava%2C%23%2Cphp%2C%2B%2B>.

Referencias



Visualización y programación en Python

Tema 2. Condiciones.





Generalmente un programa computacional, en cualquier lenguaje de programación, sigue las instrucciones que el programador le da de manera secuencial, una detrás de otra; entonces surge la necesidad de poder elegir cuándo ejecutar ciertas instrucciones bajo **condiciones** específicas.

Para elegir la **secuencialidad de las instrucciones** se pueden usar estas condiciones para lograr lo que se quiere con el código y darle un flujo dependiendo de las variables o valores que necesites que contemple para dar secuencia a esas instrucciones.





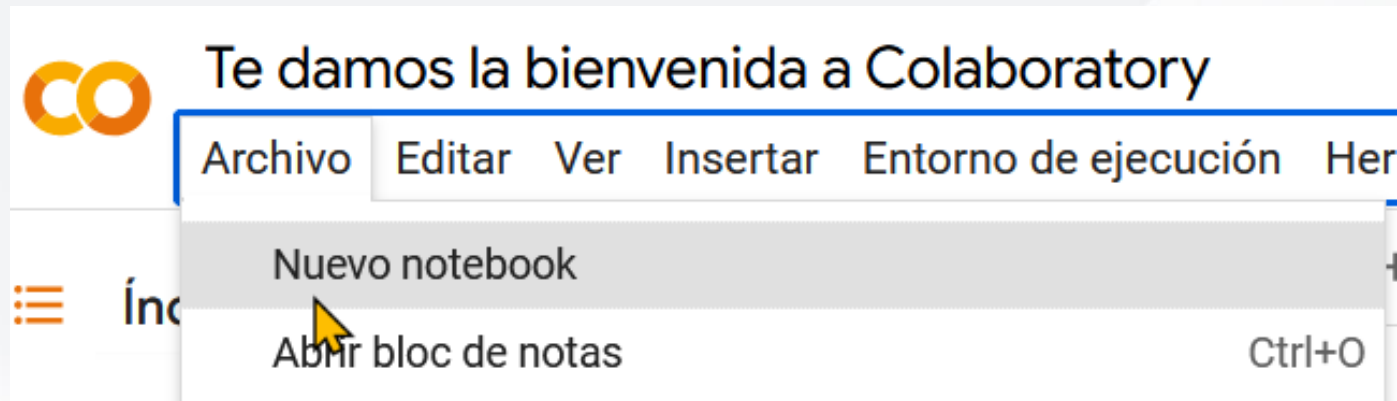
Figura. Diagrama de un **condicional**.



Comenzar tu primer programa con Python:

Ingresa a colab: <https://colab.research.google.com/>
Descárgalo en: <https://www.python.org/downloads/>
O desde esta página:
<https://www.anaconda.com/products/distribution>

**En colab de Google crea un nuevo notebook, con el
menú Archivo – Nuevo notebook:**



El siguiente paso es nombrar tu notebook y comenzar con tu código, escribe el tradicional **“hola mundo”** con esta primera línea:

```
Print ("hola mundo")
```

Y **ejecútala** con el botón de la izquierda. Ahora **declara** algunas variables asignándoles un valor e imprimiendo sus valores, según se muestra a continuación:

```
#Definiendo una variable entera con valor de 3
entero = 3
#Definiendo una variable con decimales con valor de 10.245
decimales = 10.245
#Definiendo una variable con decimales con valor de ciudad
texto = "ciudad"
#Imprimiendo los valores de las variables separándolas por una línea
print ("Entero: ", entero, "\nCon decimales: ", decimales, "\nTexto: ", texto)
```



En **Python** con solo declarar la variable y asignarle un valor ya se define también su tipo. **Las condiciones para nombrar una variable son:**

- El nombre debe comenzar con letra o `_`, no con número.
- No se permite que use guiones – ni espacios.
- No usar palabras reservadas, ya usadas para ciertos usos y funciones

Ahora en la instrucción **print** permite mostrar el valor de cualquier tipo de variable y texto con que se quiera acompañar.



Una **variable** puede ser

- **Global:** es decir, que se use en todo el programa o bien.
- **Local:** que solo mantenga su valor mientras se ejecute una función que la contenga.

#Se define variable global

```
v_global = 10
```

#Se define una función que muestra un valor de una variable local
def mostrarvalor():

```
    v_local = 5
```

```
    #Muestra valor de variable local porque esta dentro de la función  
    print ("Variable Local:", v_local)
```

#Ejecuta la función y muestra el valor que tuvo la variable local
mostrarvalor()

#Muestra el valor que tiene la variable global

```
print ("Variable Global:", v_global)
```

#Marcara error pues la variable local no se esta usando fuera de la función

```
print("Variable Local" v_local)
```



Es muy común dentro del **código dividir o enfocar** el flujo del programa de acuerdo con ciertas **condiciones**, hay varios operadores relacionales que podemos tener para realizar estas condiciones:

Operador	Ejemplos	Explicación
>	variable > 10 a > b	Compara que el operador de la izquierda sea mayor que el de la derecha. Por lo que pueden ser estos operadores variables del programa.
<	variable < 9 c < d	Compara que el operador de la izquierda sea menor que el de la derecha. Por lo que pueden ser estos operadores variables del programa.
==	variable == 8 e == f	Compara que el operador de la izquierda sea igual el de la derecha. Por lo que pueden ser estos operadores variables del programa.
>=	variable >= 10 a >= b	Compara que el operador de la izquierda sea mayor o igual que el de la derecha. Por lo que pueden ser estos operadores variables.
<=	variable <= 9 c <= d	Compara que el operador de la izquierda sea menor o igual que el de la derecha. Por lo que pueden ser estos operadores variables.

Cada uno de estas **condiciones o comparaciones**, dan como resultado un **valor booleano**, que puede ser **verdadero (true)**, cuando sí se cumple o bien **falso (false)**, cuando no se cumple.

Hay ocasiones en que se requiere más de una condición para que se siga un **bloque de código**, por lo que se pueden usar los siguientes **operadores lógicos**:

Operador	Ejemplos	Descripción
<i>and</i>	<code>a > b and c < 10</code>	Es verdadero (True), cuando AMBAS condiciones se cumplen . conocimiento o los pocos deseos de cambio.
<i>or</i>	<code>a > b or c < 10</code>	Es verdadero (True) cuando UNA de las condiciones se cumple .
<i>not</i>	<code>not (variable == 8)</code> <code>not (e == f)</code>	Niega la comparación que contiene, es decir si el resultado de la comparación es verdadero, True, lo cambia a falso y viceversa.

Para poner en práctica los operadores de condicionales, tanto los relacionales como los lógicos se usa la sentencia **If**

La traducción de **if es si**, la de **else es si no** y la de **elif es si no si**, es decir, esta última es para cuando no se cumple la sentencia y se espera que cumpla otra.

Ejemplo:

#Función para la categoría de antigüedad, que reciba de entrada los años y de salida nos entregue la categoría

```
def cat_anios(anios):
```

```
    if anios<3:
```

```
        #Verifica si es menor que 3
```

```
        cat_a="Novato"
```

```
    elif anios<=6:
```

```
        #Verifica si es menor o igual que 6
```

```
        cat_a="Comprometido"
```

```
    else:
```

```
        #Si no cumplió con las anteriores quiere decir que es mayor que 6
```

```
        cat_a="Socio"
```

```
    return cat_a
```

#Prueba de función

#Imprime con valor de 2 que corresponde a Novato

```
cat_an = cat_anios(2)
```

```
print(cat_an)
```

#Imprime con valor de 10 que corresponde a Socio

```
cat_an = cat_anios(10)
```

```
print(cat_an)
```

#Función para la categoría del sueldo, que reciba de entrada la cantidad y de salida nos entregue la categoría

def cat_sueldo(cant):

if cant>20000:

#Verifica si es mayor que 20000

cat_s="Alto"

elif cant>=10000:

#Verifica si es mayor o igual que 10000

cat_s="Medio"

else:

#Si no cumple con las anteriores quiere decir que es menor que 10000

cat_s="Bajo"

return cat_s

#Prueba de función

#Imprime con valor de 2 que corresponde a Novato

cat_s = cat_sueldo(10000)

print(cat_s)

#Imprime con valor de 10 que corresponde a Socio

cat_s = cat_sueldo(25000)

print(cat_s)

En instrucciones como ésta se pueden añadir tantas condiciones como sean necesarias, con la intención de cumplir con aquello que se nos está pidiendo.

Hay que tomar en cuenta el orden al momento de poner un **or o un and**, el **or** es para **cuando se cumpla cualquiera de las dos** y el **and** para **cuando se cumplan ambas**.

Tomará primero la condición con and así:

```
cat_an=="Socio" and cat_s=="Alto"
```

Si esa se cumple, ejecuta el comando o si se cumple ésta otra:

```
cat_s=="Medio"
```

también es como si con paréntesis definiéramos:

```
(cat_an=="Socio" and cat_s=="Alto") or (cat_s=="Medio"):
```

Lo cual dará un resultado diferente a:

```
cat_an=="Socio" and (cat_s=="Alto" or cat_s=="Medio")
```

Estas mismas condiciones y operadores se pueden usar dentro de comandos en una sola línea de código en Python:

```
# [código si se cumple] if [condición] else [código si no se cumple]
```

En este notebook de Google colab puedes ver todo el código desarrollado en el tema:

https://colab.research.google.com/drive/1782ijHe_niK7154YwwC1OvLbMsk3epet?usp=sharing







Cuadro sinóptico

Objetivo: Realizar un cuadro sinóptico donde se incluyan los diferentes **tipos de operadores condicionales y ternarios** del lenguaje de programación Python.

Instrucciones:

- Con base en lo aprendido elabora un cuadro sinóptico donde incluyas todos los operadores, condicionales y ternarios vistos.
- Incluye un ejemplo de un código que utilice este tipo de operadores y explica el funcionamiento que tiene en la línea de código indicada.
- Utiliza colores, elementos gráficos y tipografías a través de una plataforma digital que permita la mayor comprensión de lo expuesto.
- Máximo 2 cuartillas. Guardalas en formato .pdf y comparte.





Se tomaron en cuenta los principales operadores tanto relacionales como lógicos. Python cuenta con más por lo que es recomendable que puedas investigar:

¿Qué otros operadores pueden ser útiles para el código que tu podrías desarrollar? ¿Qué otros elementos tienen Python para controlar este flujo del programa? ¿Qué aplicaciones sencillas pudieras desarrollar con estos conocimientos que te apoyen en tu día a día?



Bustamante, S. (2021). Operadores Básicos en Python con ejemplos. Recuperado de <https://www.freecodecamp.org/espanol/news/operadores-basicos-en-python-con-ejemplos/>

Torres, A. (2021). Ejemplos de funciones en Python: Como declarar y llamar con parámetros. Recuperado de <https://www.freecodecamp.org/espanol/news/ejemplos-de-funciones-de-python-como-declarar-y-llamar-con-parametros/>



La obra presentada es propiedad de ENSEÑANZA E INVESTIGACIÓN SUPERIOR A.C. (UNIVERSIDAD TECMILENIO), protegida por la Ley Federal de Derecho de Autor; la alteración o deformación de una obra, así como su reproducción, exhibición o ejecución pública sin el consentimiento de su autor y titular de los derechos correspondientes es constitutivo de un delito tipificado en la Ley Federal de Derechos de Autor, así como en las Leyes Internacionales de Derecho de Autor.

El uso de imágenes, fragmentos de videos, fragmentos de eventos culturales, programas y demás material que sea objeto de protección de los derechos de autor, es exclusivamente para fines educativos e informativos, y cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por UNIVERSIDAD TECMILENIO.

Queda prohibido copiar, reproducir, distribuir, publicar, transmitir, difundir, o en cualquier modo explotar cualquier parte de esta obra sin la autorización previa por escrito de UNIVERSIDAD TECMILENIO. Sin embargo, usted podrá bajar material a su computadora personal para uso exclusivamente personal o educacional y no comercial limitado a una copia por página. No se podrá remover o alterar de la copia ninguna leyenda de Derechos de Autor o la que manifieste la autoría del material.

