



Qué es **Python**



En palabras de su propio creador, Guido van Rossum, Python es un:

**“El lenguaje de programación de alto nivel, y su filosofía de diseño central, tiene que ver con la legibilidad del código y una sintaxis que permite a los programadores expresar conceptos en unas pocas líneas de código”**



## Por qué aprender Python?

Python es un lenguaje de programación muy potente, flexible y fácil de aprender.

Podremos crear lo que se nos ocurra, desde simples Scripts hasta juegos y aplicaciones muy grandes.

Python es de código abierto (libre), gratuito y multiplataforma.

Python es un lenguaje de propósito general, ejemplo de ello es la lectura/escritura de archivos, sitios web, bases de datos, interfaz gráfica de usuario, etc. Es por eso que es utilizado por muchas personas y tiene una gran comunidad de usuarios activos.



## Ambiente de Desarrollo

El IDE (Integrated Development Environment)

ONLINE

Utilizaremos la plataforma Online **REPL.IT**, la cual te permite crear tu código y compartirlo

### Ventajas

- Gratis y no necesitas instalar nada.
- Sencillo, interactivo y te permite compartir tú código con un enlace.
- Soporta más de 40 lenguajes entre ellos Python, Java, etc.  
IDE Online, crear códigos, proyectos y compartirlos.



## Ambiente de Desarrollo

Local

\* Solo para quienes deseen tener una alternativa instalada en sus equipos

1. Interprete de Python. Para que el computador pueda identificar y comprender el código, una especie de traductor.

Para descargar Python ir a la página oficial  
<http://www.python.org>

Actualmente está en la versión 3.9.4

2. Editor de código (Visual Studio Code). Para facilitar la escritura de nuestro código.

[code.visualstudio.com](https://code.visualstudio.com)

*Recomendamos video de instalación*

<https://www.youtube.com/watch?reload=9&v=qw0ae-GTuck&feature=youtu.be>

IDE

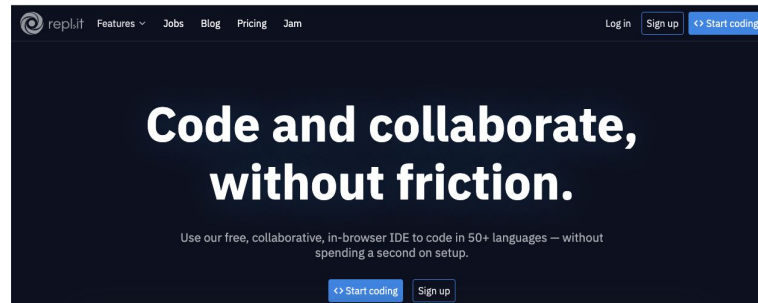


repl.it

—




# Registro en REPL.IT

- Ingresar a [www.repl.it](https://www.repl.it)



- Registrarse

A registration form on the Repl.it website. It features three social login options at the top: Google, GitHub, and Facebook. Below these are three input fields for "username", "email", and "password". There is a checkbox labeled "I'm a teacher" and a link "or log in". At the bottom is a large red button labeled "Sign up".

username

email

password

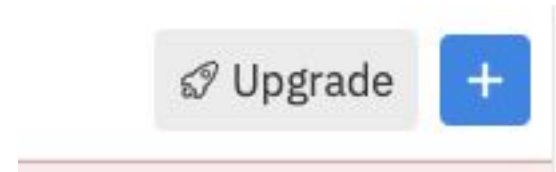
☐ I'm a teacher [or log in](#)

Sign up

\* (sugerido usuario y correo Uninorte)

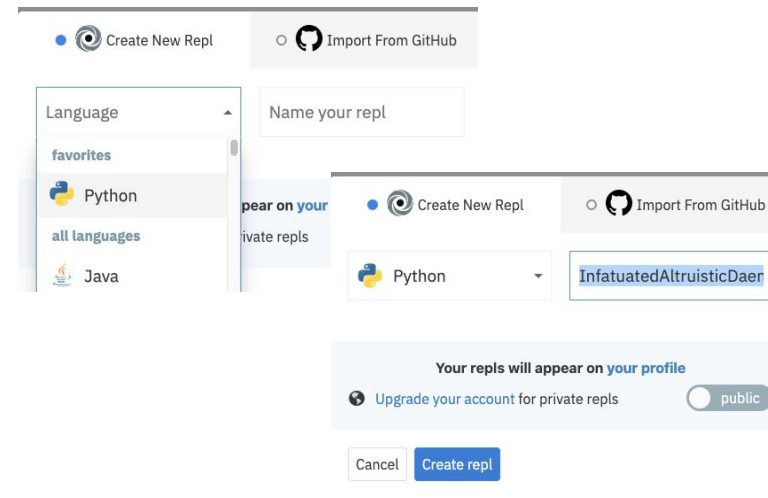
# Creación de un proyecto Python

- Dar clic en el botón + en color azul (Esquina superior derecha)



- Seleccione Python en la lista de Lenguajes, y dar clic en el botón “Crear Repl”

Nota: Esta plataforma te permite programar en otros lenguajes de programación incluido Java)



# Ambiente de trabajo Repl.it

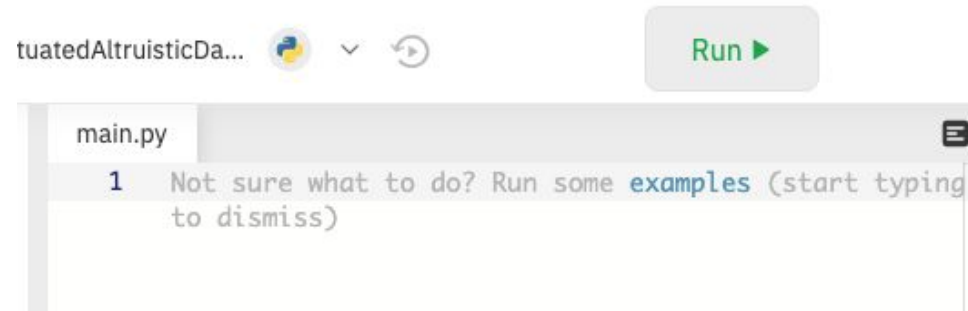
Se encuentran 3 secciones, la central (color blanco) es donde desarrollan el código, la de la derecha (color negro) permite mostrar la ejecución y realizar operaciones directamente (tipo consola), y la sección más a la izquierda muestra todos los proyectos Python (la extensión de los proyectos es .py)





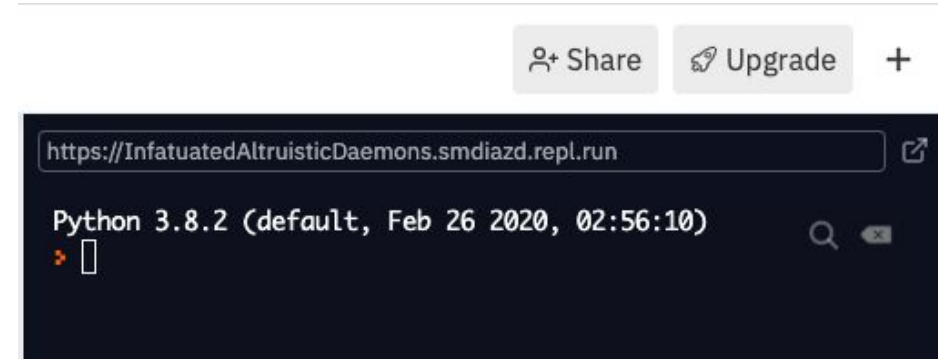
## Ejecutar

- Sección central, esquina superior, dar clic en el botón “Run” para ejecutar



## Compartir

- Para compartir el código con otra persona, e inclusive modificarlo simultáneamente en línea- En la sección de Ejecución del código, en la parte superior, dar clic en el botón “Share”, obtener el link y envíalo por chat o correo.



---

# Sobre el lenguaje de programación





## Comentarios en el código

Los comentarios son **textos informativos** que ayudan a entender el código (ya sea para nosotros u otros programadores que vean nuestro código). Los comentarios comienzan con el símbolo **#** y todo lo que sea agregado después del símbolo **NO será interpretado como código** en nuestro programa.

Ejemplos:

```
#Esto es un comentario
```

```
a + b #Esto es una suma
```

Para escribir más de una línea de comentario se utilizan tres comillas simples'''

```
''' Esto
```

```
también
```

```
es
```

```
un comentario '''
```

# OPERADORES

---



# Operadores Aritméticos

La siguiente tabla muestra **todos los operadores aritméticos** soportados por Python. Supongamos que tenemos una variable **a = 10** y otra **b = 20**, entonces:

Operador	Descripción	Ejemplo
+	Suma	$a + b = 30$
-	Resta	$a - b = -10$
*	Multiplicación	$a * b = 200$
/	División	$b / a = 2$
%	Módulo – Devuelve el resto de la división	$b \% a = 0$
**	Exponente – Realiza exponencial	$a ** b = 10 \text{ a la } 20$
//	División baja - Devuelve el entero de la división	$9 // 2 = 4$ y $9,0 // 2,0 = 4,0$



# Operadores comparativos

Supongamos que la variable **a = 10** y la **variable b = 20**:

Operador	Descripción	Ejemplo
==	Si los valores de los 2 operadores <b>son iguales</b> la condición es True	(a == b) no es True
!=	Si los valores de los 2 operadores <b>no son iguales</b> la condición es True	(a != b) es True
>	Si el valor del operador de la izquierda es <b>mayor que</b> el operador de la derecha la condición es True	(a > b) no es True
<	Si el valor del operador de la izquierda es <b>menor que</b> el valor del operador de la derecha la condición es True	(a < b) es True
>=	Si el valor del operador de la izquierda es <b>mayor o igual que</b> el valor del operador de la derecha la condición es True	(a >= b) no es True
<=	Si el valor del operador de la izquierda es <b>menor o igual que</b> el valor del operador de la derecha la condición es True	(a <= b) es True



## Operadores Lógicos

Pseudocódigo	Python
Inicio ...  Fin	No hay instrucción asignada
Y, And	and
O, Or	or
No, Not	!
= (igual)	==
<>	!=



# Variantes de operador de asignación

## Operadores de asignación

Los **operadores de asignación en python** son utilizados para asignarle valor a las variables. Por ejemplo: `a = 5`, el signo `=` asigna el valor **5** a la variable **a**.

Supongamos que **a = 10** y **b = 20**:

Operador	Ejemplo
<code>=</code>	<code>c = a + b</code> (se asigna el valor de <code>a + b</code> en <code>c</code> )
<code>+=</code>	<code>c += a</code> es lo mismo que <code>c = c + a</code>
<code>-=</code>	<code>c -= a</code> es lo mismo que <code>c = c - a</code>
<code>*=</code>	<code>c *= a</code> es lo mismo que <code>c = c * a</code>
<code>/=</code>	<code>c /= a</code> es lo mismo que <code>c = c / a</code>
<code>%=</code>	<code>c %= a</code> es lo mismo que <code>c = c % a</code>
<code>**=</code>	<code>c **= a</code> es lo mismo que <code>c = c ** a</code>
<code>//=</code>	<code>c //= a</code> es lo mismo que <code>c = c // a</code>





# Operadores Especiales

## Operadores Especiales

Existen otros operadores en el lenguaje python:

### Ejemplos de ellos:

--> **is** - Es True si los operadores son idénticos

--> **is not** - Es True si los operadores no son idénticos

--> **in** - Es True si el valor o variable se encuentra en la secuencia

--> **not in** - Es True si el valor o variable no se encuentra en la secuencia

Primero declaramos 2 variables (**a = 10** y **b = 10**).

-->> **a is b** - Es True porque son iguales

-->> **a is not b** - Es False porque son iguales

Luego cambiamos el valor de las variables (**a = 1**, **b = 6** y **c = [1,2,3,4,5]**)

--> **a in c** - Es True porque **a** se encuentra en **c**

--> **b in c** - Es False porque **b** no se encuentra en **c**

# VARIABLES

---



# Nombres de variables

- Python es CASE SENSITIVE.

Por ejemplo la variable Nota y nota las toma como dos variables diferentes, aunque solo difieran en que una tiene la primera letra en mayúscula y la otra en minúscula.

- No pueden iniciar con número. Por ejemplo: 2PARCIAL

Nota: Buenas practicas en general para la programación

Convenciones al nombrar variables (estilos)

Snake Case : nota\_parcial (separador guión bajo)

Camel Case : notaParcial (Iniciales primera minúscula luego mayúscula)

Pascal Case : NotaParcial (Iniciales todas en mayúsculas)



## Tipos de variables

Pseudocódigo	Python
Entero	<code>int</code>
Real	<code>float</code>
Lógico	<code>boolean</code>
Cadena	<code>Str</code>

- NO se declarar variables  
Por defecto todas son tipo Cadena, por ello es importante convertirlas a número cuando así lo requieran
- Python es un lenguaje de tipado dinámico:, es decir una variable puede tomar valores de distinto tipo. Por ejemplo  
a=5  
a="hola"

# PRIMITIVAS

---

# Lectura de variables (datos de entrada por pantalla)

Pseudocódigo	Python
Lectura de variables (Lea o Leer)	<code>input()</code>
Lea variableEntera  Ejemplo: Entero: numero Escriba "Digite el primer número" Lea numero	<code>variableEntera=int(input("Msg guía"))</code>  Ejemplo <code>numero = int(input("Digite el primer número"))</code> * Por defecto los datos al leerlos son cadenas, por eso se convierten a int
Lea variableReal  Ejemplo: Entero: primer_parcial Escriba "Digite nota del primer parcial" Lea primer_parcial	<code>VariableReal=float(input("Msg guía")):</code>  Ejemplo <code>Primer_parcial = float(input("Digite nota del primer parcial"))</code>  * Por defecto los datos al leerlos son cadenas, por eso se convierten a float
Lea variableCadena Ejemplo Cadena: nombre Escriba "Digite su nombre completo" Lea nombre	<code>variableCadena=input("Msg guía"):</code> Ejemplo <code>nombre=(input("Digite su nombre completo"))</code>  * Por defecto los datos son cadena por tanto no requiere conversión



## Salida por pantalla

Pseudocódigo	Python
Escribir o Escriba	<b>print()</b>
Escriba “Bienvenidos a clase”	print(“Bienvenidos a clase”)  *Puede utilizar comillas dobles o sencillas.
Cadena: nombre nombre □ “Sandra” Escriba “Bienvenida a clase “,nombre	nombre=“Sandra” Print(“Bienvenida a clase “+nombre)
Entero: aaaa aaaa □ 2020 Escriba “Bienvenido año “,aaaa	aaaa=2020 Print(“Bienvenida a clase “+str(aaaa))  * Para CONCATENAR (pegar) un dato, este debe ser cadena, por tanto es necesario convertirlo cuando es un dato numérico

---

# Proyectos ejemplo



# Proyecto Ejemplo (main.py)

```
main.py
1 a=float(input("Digite la nota del primer parcial "))
2 b=float(input("Digite la nota del segundo parcial "))
3 c=float(input("Digite la notas del tercer parcial "))
4 x=(a+b+c)/3
5 print("La nota definitiva es "+str(x))
```

```
https://InfatuatedAltruisticDaemons.smdiazd.repl.run
Digite la nota del primer parcial 3.5
Digite la nota del segundo parcial 4
Digite la notas del tercer parcial 3
La nota definitiva es 3.5
>
```

Sacar nota definitiva de una asignatura, donde se tomaron 3 notas y todas tienen el mismo peso.

# CONDICIONALES

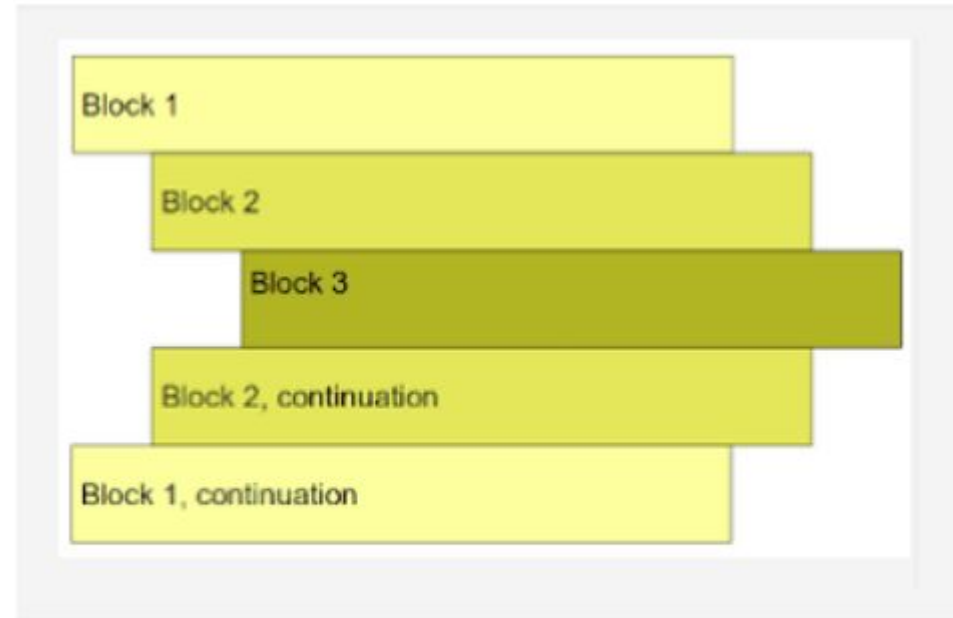
---

# Condicionales

Pseudocódigo	Python
<pre>Si(condición)Entonces     ... Sino     ... FinSi</pre>	<pre>if(condición): #el entonces es :     ... else:     ... #el fin si no existe</pre>
<pre>DependiendoDe(variable)     Caso 1     ...     Caso n     Sino     ... FinDD</pre>	<p>El Dependiendo De no está implementado en Python</p>

# Condicional SI

En este tema de los **condicionales** aparecen dos nuevo conceptos: los **bloques** y la **indentación** (sangrado, sangría, etc). **¿Como funcionan?**: Todas las declaraciones de código con la misma distancia a la derecha pertenecen al **mismo bloque de código** (el bloque termina en una línea con menos sangría o al final del código). Los bloques se pueden anidar agregando más sangrías a la derecha. La **indentación** se utiliza para que los códigos sean más legibles, comprensibles para los programadores. Como se puede ver, el código que sigue al **condicional if y else** comienza con una indentación de 4 espacios (un tabulador).



**Bloques e indentación en python**

# Ejemplo: Si Independientes

main.py

```
1  #para dos números diferentes, indica cuál es el
   número mayor
2  a=float(input("Digite primer número "))
3  b=float(input("Digite segundo número "))
4  if a > b:
5
6      print ("el primero número es mayor que el
           segundo")
7
8  if a < b:
9
10     print ("el segundo número es mayor que el
           primero")
```

<https://InfatuatedAltruisticDaemons.smdiazd.repl.run>

```
Digite primer número 45
Digite segundo número 2
el primero número es mayor que el segundo
>
```

# Ejemplo: Si - Sino

main.py

```
1 semaforo=(input("Digite el color del semaforo:  
rojo o verde "))  
2 if semaforo == "verde":  
3     print ("Cruzar la calle")  
4 else:  
5     print ("Esperar")  
6
```

<https://InfatuatedAltruisticDaemons.smdiazd.repl.run>

```
Digite el color del semaforo: rojo o verde verde  
Cruzar la calle  
> f
```

# Ejemplo If-elif

main.py

```
1 compra=float(input("Digite el valor de la compra "))
2 )
3 if compra <= 100:
4     print ("Pago en efectivo")
5 elif compra > 100 and compra < 300:
6     print ("Pago con tarjeta de débito")
7 else:
8     print ("Pago con tarjeta de crédito")
```

<https://InfatuatedAltruisticDaemons.smdiazd.repl.run>

Digite el valor de la compra 20

Pago en efectivo



CICLOS

—



# Ciclos

Pseudocódigo	Python
<pre>MientrasQue()Haga ... FinMientrasQue</pre>	<pre>while(): ...</pre>
<pre>Haga ... FinHH(condición)</pre>	No hay instrucción asignada
<pre>Para(inicio;condición;incremento) ... FinPara</pre>	<pre>n = 10 for k in range(1,n+1,1):     print(""*k)</pre>

# Ejemplo Mientras Que

```
anio = 2001
while anio <= 2012:
    print "Informes del Año", str(anio)
    anio += 1
```

La iteración anterior, generará la siguiente salida:

A terminal window with a dark background and light gray text. It displays the output of the Python code, showing the phrase "Informes del año" followed by each year from 2001 to 2012, one line per year.

```
Informes del año 2001
Informes del año 2002
Informes del año 2003
Informes del año 2004
Informes del año 2005
Informes del año 2006
Informes del año 2007
Informes del año 2008
Informes del año 2009
Informes del año 2010
Informes del año 2011
Informes del año 2012
```


# Para

Por cada año en el rango 2001 a 2013, imprimir la frase *"Informes del Año año"*:

```
for anio in range(2001, 2013):  
  
    print "Informes del Año", str(anio)
```

# Funciones

—




En Python, la definición de funciones se realiza mediante la instrucción `def` más un nombre de función descriptivo

```
def mi_funcion():  
    # aquí el algoritmo
```

Una función, no es ejecutada hasta tanto no sea invocada. Para invocar una función, simplemente se la llama por su nombre:

```
def mi_funcion():  
    print "Hola Mundo"
```

```
mi_funcion()
```



Cuando una función, haga un retorno de datos, éstos, pueden ser asignados a una variable:

```
def funcion():  
    return "Hola Mundo"
```

```
frase = funcion()
```

```
print frase
```



## Ejemplo:

```
def sumar(number1, number2):  
    Resultado= number1 + number2  
    Return Resultado  
  
number1=int(input("Digite el primer número"))  
number2=int(input("Digite el Segundo número"))  
  
Valor_Final=sumar()  
  
Print(Valor_Final)
```



## Ejemplo 2:

```
def es_par(numero):  
    if numero % 2 == 0:  
        return True  
    else:  
        return False
```

```
numero=int(input("Digite un numero"))  
resultado=es_par()  
print(resultado)
```

```
>>>(2)  
True  
>>>(5)  
False
```



# Tabla de equivalencias

Pseudocódigo	Python
Entero Vector [100]	<pre>import numpy as np Vector =np.array([])</pre>
Para i=1;n;1 hacer leer Vector[i] Fin para	<pre>for i in range (1,n+1,1):     Vector[i]=input():</pre>



# Ambiente de Desarrollo

Local

\* Solo para quienes deseen tener una alternativa instalada en sus equipos

1. Interprete de Python. Para que el computador pueda identificar y comprender el código, una especie de traductor.

Para descargar Python ir a la página oficial

<http://www.python.org>

Actualmente está en la versión 3.8.5

2. Editor de código (Visual Studio Code). Para facilitar la escritura de nuestro código. [code.visualstudio.com](https://code.visualstudio.com)

*Recomiendo video de instalación*

<https://www.youtube.com/watch?reload=9&v=qw0ae-GTuck&feature=youtu.be>