Vamos a ver Variables, Tipos de de datos y operaciones básicas!

Python se creó en los 89 por Guido van Rossum, además es multiplataforma UNIX, LINUX; MacOS, WIndows. También es multiparadigma de alto nivel, permite programación orientada a objetos, programación estructurada y programación funcional

Sintaxis compacta, sencilla e intuitiva, con una curva de aprendizaje mínima y una potente liberia

Python no compila, sino que se ejecuta directamente (isa un intérprete). Eso permite hacer cosas que son imposibles en otros lenguajes como ejecutar instrucciones de manera interactiva, crear funciones al vuelo mientras un programa se ejecuta, interpretar un string como código python y ejecutarlo. etc

Variables: Espacios reservados de memoria que tiene asignado un identificador

en otros lenguajes las variables se DECLARAN e INICIALIZAN, en python esto es diferente porque no existe la declaración, simplemente las inicializamos
EJ PY:

En PYTHON no declaramos las variables.
Las variables se inicializan directamente

```
numero_entero = 42
numero_decimal = 12.5
texto = 'hola'
variable_logica = True
```

EJ otros lenguajes:

DECLARACIÓN

```
Definir sumaTotal Como Entero
Definir precio Como Real
Definir nota Como Texto
Definir terminado Como Logica
```

INICIALIZACIÓN

```
sumaTotal = 12
precio = 20.5
nota = "Hola"
terminado = Falso
```

Por este motivo es importante tener en cuenta de antemano los tipos de datos con los que vamos a trabajar.

Si queremos podemos realizar inicializaciones explícitas, es redundante pero al principio es útil

INICIALIZACIÓN EXPLÍCITA

```
numero_entero = int(42)
numero_decimal = float(12.5)
texto = str('hola')
variable_logica = bool(True)
```

SINTAXIS PERMITIDA AL NOMBRAR VARIABLES:

Los nombres pueden contener solo letras, números y barras bajas — my_variable_1 ✓, _my_variable_1 ✓

Pueden comenzar por una letra o una barra baja pero NUNCA por un número — 1_my_variable ✗

Los espacios no están permitidos, pero se pueden usar barras bajas para separar las palabras — my variable ✗

No se deben usar palabras que están asociadas ya a funciones internas de python — por ejemplo: *print* ✗

Los nombres de variables deben ser cortos pero descriptivos
- nombre >> n
- nombre_estudiante >> n_e
- tamaño_nombre >> tamaño_del_nombre_de_las_personas

Cuidado al usar la ele minúscula 'l', la i mayúscula 'I' y la letra o mayúscula 'O'. Al leer el código el usuario puede confundirlos con unos y ceros '1', '0'

```
x = y = z = 10

print(x,y,z)
```

```
10 10 10
```

```
x, y, z = 'texto 1', 'texto 2', 'texto2'

print(x,y,z)
```
✓ 0.2s

```
texto 1 texto 2 texto2
```

```
x, y, z = 10,20,30

print(x,y,z)
```
✓ 0.1s

```
10 20 30
```

```
x, y, z = 'texto 1', 120.3, 42

print(x,y,z)
```
✓ 0.2s

```
texto 1 120.3 42
```

Variables - Pedir valores por pantalla

La función input() permite obtener **texto** escrito por teclado.

```
print('¿Cómo te llamas?')
nombre = input()
print('Me alegro de conocerte', nombre)
```

```
¿Cómo te llamas?
Elena
Me alegro de conocerte Elena
```

```
nombre = input('¿Cómo te llamas? ')
print('Me alegro de conocerte', nombre)
```

```
¿Cómo te llamas? Elena
Me alegro de conocerte Elena
```

Recordemos que INPUT() solo acepta texto

La función input() permite obtener **texto** escrito por teclado.

```
numero = input('¿Cúantos años tienes? ')
print('Entonces has vivido aproximadamente', 365.0*numero, 'dias')
```

```
¿Cúantos años tienes? 25
Traceback (most recent call last):
  File "/Users/ tu_nombre_de_usuario /Desktop/Master Conquer Blocks/MODULOS/Python/Clase 1/teoria_clase1.py", line 15, in <module>
    print('Entonces has vivido aproximadamente', 365.0*numero)
TypeError: can't multiply sequence by non-int of type 'float'
```

Para solucionar esto debemos hacer que interprete el texto como un número!

```
numero = float(input('¿Cúantos años tienes? '))
print('Entonces has vivido aproximadamente', 365.0*numero, 'dias')
```

```
¿Cúantos años tienes? 25
Entonces has vivido aproximadamente 9125.0 dias
```

```
numero = int(input('¿Cúantos años tienes? '))
print('Entonces has vivido aproximadamente', 365*numero, 'dias')
```

```
¿Cúantos años tienes? 25
Entonces has vivido aproximadamente 9125 dias
```

VARIABLES - TIPOS

Las funciones INT(), FLOAT(), STR(), BOOL() son funciones de tipo. Pueden convertir un tipo de dato en otro tipo de dato.

cómo convertir el tipo de dato?

```
numero_entero = int(42)
numero_decimal = float(12.5)
texto = str('hola')
variable_logica = bool(True)
```

```
numero_entero = 42
numero_decimal = float(numero_entero)
print(numero_decimal)
```

✓ 0.1s

```
42.0
```

La función TYPE() nos devuelve el tipo de dato con el que estamos trabajando

```
numero_entero = 42
numero_decimal = float(numero_entero)
numero_texto = str(numero_entero)
print(type(numero_entero), type(numero_decimal), type(numero_texto))
```

✓ 0.3s

```
<class 'int'> <class 'float'> <class 'str'>
```

```
boolean_0 = bool(0)
boolean_1 = bool(1)
boolean_42 = bool(42)
boolean_float = bool(45.3)
boolean_texto = bool('hola')
boolean_texto_vacio = bool('')
print(boolean_0, boolean_1, boolean_42, boolean_float, boolean_texto)
print(boolean_texto, boolean_texto_vacio)
```

✓ 0.3s

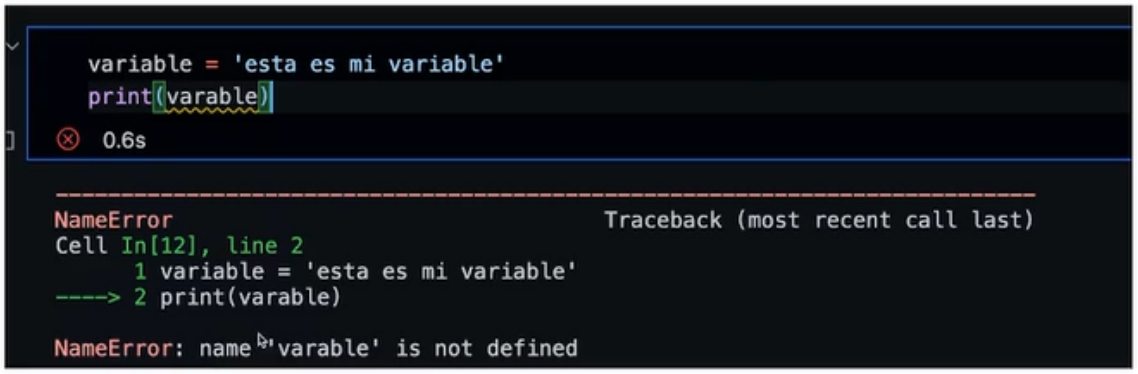
```
False True True True True
True False
```

puedo convertir un valor entero o decimal en un booleano.

el 0 es falso
el 1 es verdadero

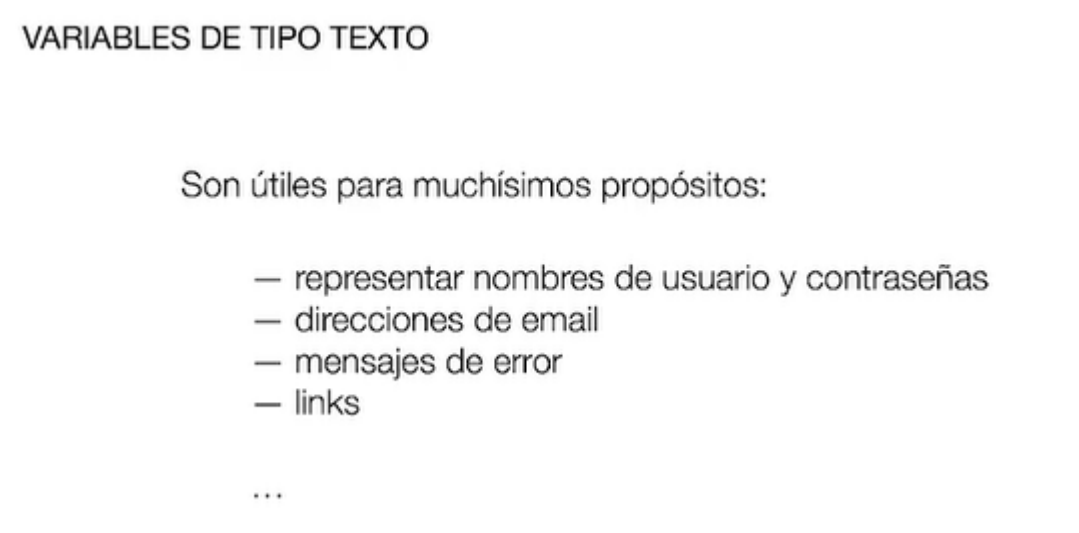
y cualquier valor superior a estos va a ser verdadero

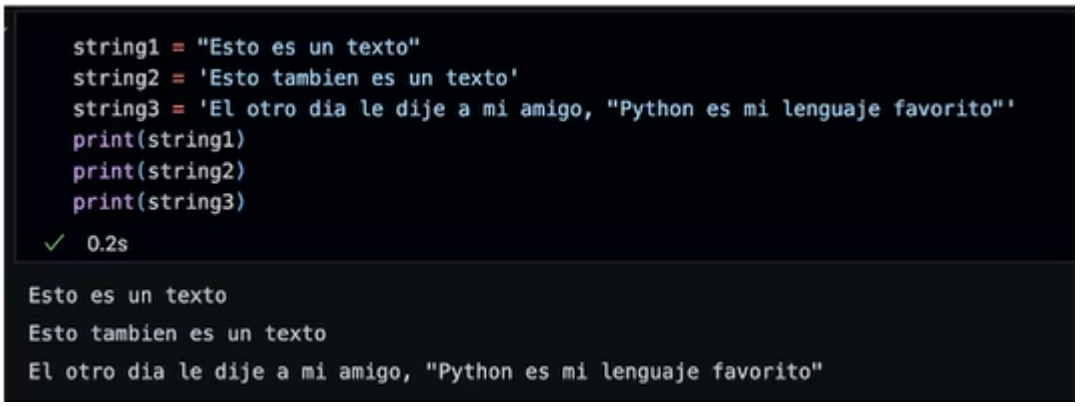
Hay algunos errores típicos con las variables, error de nombre:



CONSTANTE = VARIABLE

en casi todos los lenguajes las constantes tienen un valor que nunca cambia, pero en python las constantes no existen como tal, simplemente una constante es una variable que no variamos a lo largo de nuestro código





Existen **funciones internas** (o prefabricadas) en python para realizar acciones sobre variables o **objetos** que sean de **tipo** texto que puede sernos útiles.

Cambio de mayúscula a minúscula:

title()

```
nombre = 'juan gomez'
print(nombre.title())
```

✓ 0.6s

```
Juan Gomez
```

upper()

```
nombre = 'juan gomez'
print(nombre.upper())
```

✓ 0.6s

```
JUAN GOMEZ
```

lower()

```
nombre = 'jUAn goMeZ'
print(nombre.lower())
```

✓ 0.2s

```
juan gomez
```

Eliminar espacios en blanco:

right strip - rstrip():

```
>>> nombre = 'python '
>>> nombre
'python '
>>> nombre.rstrip()
'python'
>>> nombre
'python '
>>>
```

```
>>> nombre = nombre.rstrip()
>>> nombre
'python'
>>>
```

left strip - lstrip():

```
>>> nombre = ' python'
>>> nombre.lstrip()
'python'
>>>
```

strip():

```
>>> nombre = ' python '
>>> nombre
' python '
>>> nombre.strip()
'python'
```

Sustituir partes del string:

replace():

```
>>> string = 'Hola.Mundo'
>>> print(string.replace("."," "))
Hola Mundo
>>>
```

Encontrar un string dentro de otro string:

find():

```
>>> string = 'Hola Mundo'
>>> print(string.find('Hol'))
0
>>> print(string.find('do'))
8
>>> print(string.find('hey'))
-1
>>>
```

Combinar y concatenar texto:

```
nombre = 'juan'
apellido = 'gomez'
nombre_completo = nombre + " " + apellido
print(nombre_completo)
```

✓ 0.3s

```
juan gomez
```

```
nombre = 'juan'
apellido = 'gomez'
nombre_completo = nombre + " " + apellido
mensaje = "¡Hola, " + nombre_completo.title() + "!"
print(mensaje)
```

✓ 0.2s

```
¡Hola, Juan Gomez!
```

Tabs y saltos de linea:

Tab - \t :

```
>>> print("Python")
Python
>>> print("\tPython")
        Python
```

Salto de linea - \n :

```
>>> print("Lenguajes:\nPython\nJavaScript\nSolidity")
Lenguajes:
Python
JavaScript
Solidity
>>>
```

Podemos acceder a los componentes del string mediante sus índices:

```
>>> nombre = 'Juan'
>>> print(nombre[0])
J
>>>
```

Los índices comienzan en 0

O extraer partes del mismo:

de 0 a 4

```
>>> usuario = 'YoSoyJuan'
>>> print(usuario[0:5])
YoSoy
>>>
```

```
>>> usuario = 'YoSoyJuan'
>>> print(usuario[5:9])
Juan
>>>
```

Revertir un string:

```
>>> cadena = 'abcde'
>>> print(cadena[::-1])
edcba
>>>
```

Consultar el tamaño de un string:

```
>>> cadena = 'abcde'
>>> print(len(cadena))
5
>>>
```

```
>>> mensaje = "Jean le Rond d'Alembert fue un gran matematico"
>>> print(mensaje)
Jean le Rond d'Alembert fue un gran matematico
>>>
```

```
>>> mensaje = 'Jean le Rond d'Alembert fue un gran matematico'
  File "<stdin>", line 1
    mensaje = 'Jean le Rond d'Alembert fue un gran matematico'
                            ^
SyntaxError: invalid syntax
```

Syntax Error significa que el interprete no reconoce esta parte del código como código válido de python

Enteros o Integers:

```
>>> 2 + 3
5
>>> 3 - 2
1
>>> 2 * 3
6
>>> 3 / 2
1.5
>>>
```

Suma (+), resta(-), multiplicación (*), división (/)

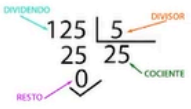
```
3 ↑ 2
```

```
>>> 3**2
9
>>> 3**3
27
>>> 10**6
1000000
>>>
```

potencia (**)

```
>>> 4 % 3
1
>>> 5 % 3
2
>>> 6 % 3
0
>>>
```

modulo o resto (%)




```
>>> 2 + 3*4
14
>>> (2 + 3) * 4
20
>>>
```

Python sigue el orden matemático de las operaciones:

1. contenido de los paréntesis
2. exponentes
3. multiplicación y división
4. suma y resta

Floats o decimales:

```
>>> 0.2 + 0.2
0.4
>>> 2 * 0.1
0.2
>>> 2 * 0.2
0.4
>>> 0.2 + 0.5
0.7
```

```
>>> 0.2 + 0.1
0.30000000000000004
>>> 3 * 0.1
0.30000000000000004
>>>
```

Python intenta darte tantos valores tras la coma como le es posible. Intenta trabajar con la mayor precisión posible.

El origen de esta imprecisión esta en como los ordenadores están forzados a representar los números de manera interna.

Ocurre en todos los lenguajes de programación.

Cómo combinar números y strings?

```
>>> numero_dias = 365
>>> mensaje = 'El año tiene ' + numero_dias + 'dias'
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not "int") to str
```

```
>>> numero_dias = 365
>>> mensaje = 'El año tiene ' + str(numero_dias) + ' dias'
>>> print(mensaje)
El año tiene 365 dias
>>>
```