

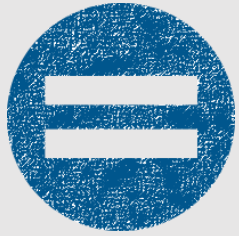
# NIVEL 1

---

VARIABLES, INSTRUCCIÓN DE ASIGNACIÓN  
& TIPADO DINÁMICO

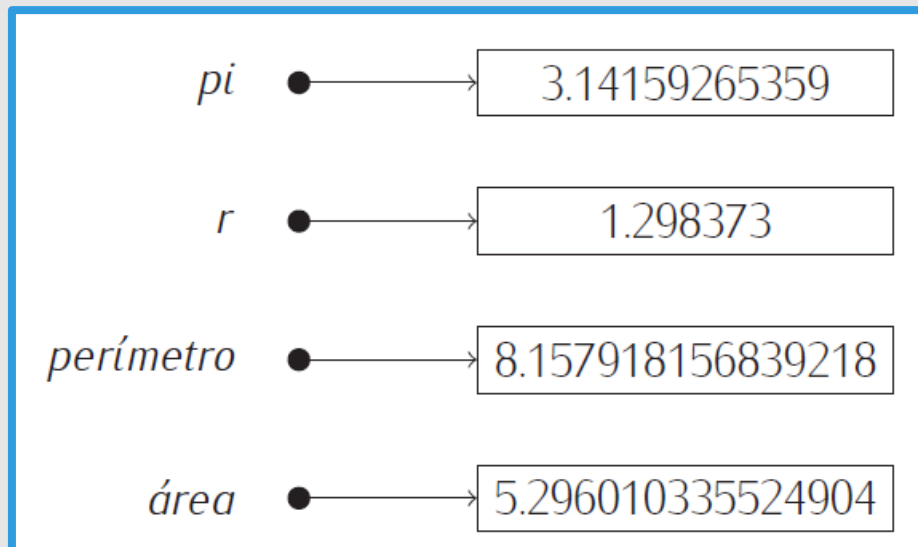


# VARIABLES E INSTRUCCIÓN DE ASIGNACIÓN



- Una de las características más poderosas de un lenguaje de programación es la capacidad de manipular variables
- Las variables se usan para **guardar valores** que se necesitarán más adelante en el programa
- La instrucción de **asignación** se usa para guardar un valor en una variable

Las asignaciones son “mudas”, no producen ninguna salida por pantalla

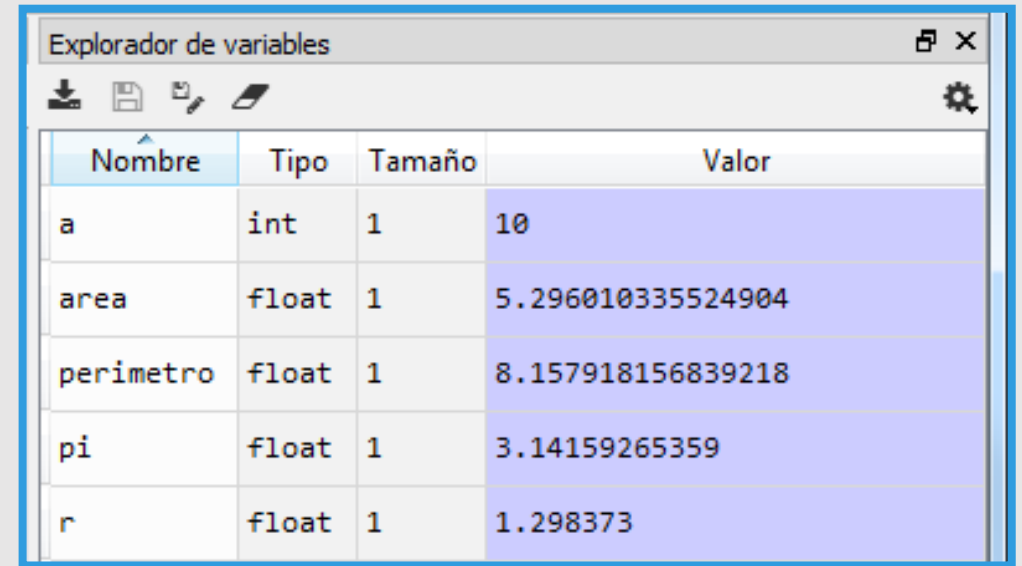


```
>>> pi = 3.14159265359↵  
>>> r = 1.298373↵  
>>> 2 * pi * r↵  
8.157918156839218  
>>> pi * r ** 2↵  
5.296010335524904
```

Se reserva memoria del computador para guardar el valor

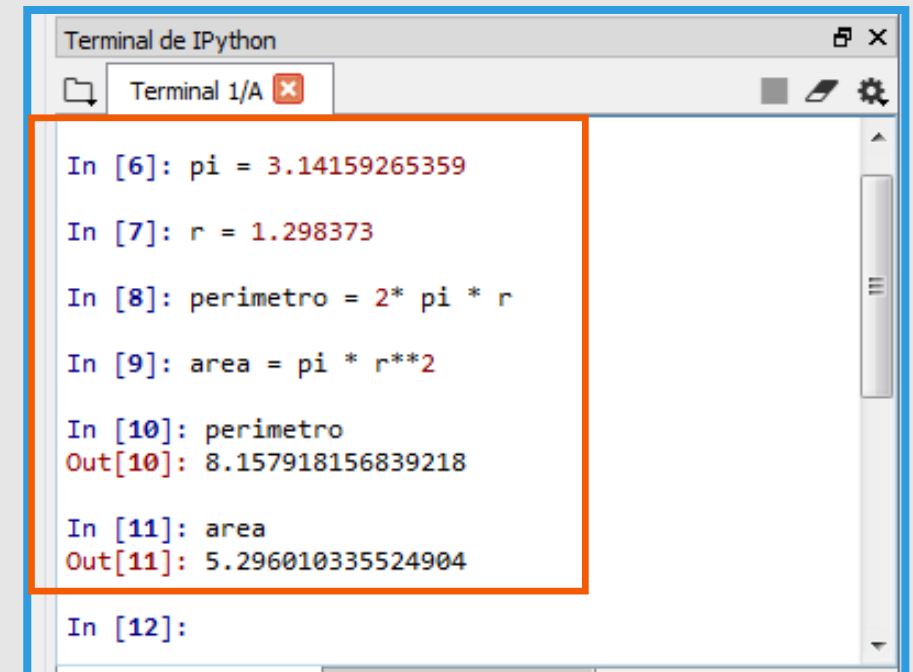
# VARIABLES E INSTRUCCIÓN DE ASIGNACIÓN

Si queremos ver cuánto vale una variable, basta con evaluar la expresión que la contiene (es decir escribirla en la terminal)



Explorador de variables

Nombre	Tipo	Tamaño	Valor
a	int	1	10
area	float	1	5.296010335524904
perimetro	float	1	8.157918156839218
pi	float	1	3.14159265359
r	float	1	1.298373



Terminal de IPython

```
In [6]: pi = 3.14159265359
In [7]: r = 1.298373
In [8]: perimetro = 2* pi * r
In [9]: area = pi * r**2
In [10]: perimetro
Out[10]: 8.157918156839218
In [11]: area
Out[11]: 5.296010335524904
In [12]:
```

# TIPADO DINÁMICO

También puede cambiar el tipo o **tipado dinámico** (Python es un lenguaje dinámicamente tipado)

```
In [18]: dia = "Jueves"

In [19]: dia
Out[19]: 'Jueves'

In [20]: dia = "Viernes"

In [21]: dia
Out[21]: 'Viernes'

In [22]: type (dia)
Out[22]: str

In [23]: dia = 21

In [24]: dia
Out[24]: 21

In [25]: type (dia)
Out[25]: int
```

Se puede asignar un valor a una misma variable cuantas veces se quiera. La variable solo «**recuerda**» el último valor asignado... hasta que se le asigne otro.

# NOMBRES DE VARIABLES Y PALABRAS RESERVADAS

- El nombre de una variable es su **identificador**
- Un identificador debe estar formado por letras minúsculas, mayúsculas, dígitos y/o el carácter de subrayado \_ (si el identificador tiene varias palabras)
- El primer carácter no puede ser un dígito
- Python distingue entre mayúsculas y minúsculas
- Un identificador no puede coincidir con una palabra reservada o palabra clave del lenguaje



and	as	assert	break	class	continue
def	del	elif	else	except	exec
finally	for	from	global	if	import
in	is	lambda	nonlocal	not	or
pass	raise	return	try	while	with
yield	True	False	None		

```

In [1]: area = 10
In [2]: area_figura = 100
In [3]: area_1 = 1000
In [4]: area1 = 10000

In [5]: area
Out[5]: 10

In [6]: area_figura
Out[6]: 100

In [7]: area_1
Out[7]: 1000

In [8]: area1
Out[8]: 10000

```



A continuación vamos a  
hablar de los errores más  
comunes que se  
comenten con la  
asignación

**¡PARA NO REPETIR!**

# ¡El orden es importante!

**SIEMPRE** variable = valor o expresión

```
In [13]: a=10

In [14]: a
Out[14]: 10

In [15]: 10 = a
File "<ipython-input-15-a1ebfc217b9f>", line 1
      10 = a
         ^
SyntaxError: can't assign to literal
```

variable = expresión **NO ES EQUIVALENTE** a  
expresión = variable

- Primero se evalúa la expresión a la derecha del símbolo =
- Luego se guarda el valor en la variable indicada a la izquierda del símbolo =



## Asignar (=) NO es comparar (==)

Muchas personas confunden la instrucción de **asignación** = con el operador de **comparación** ==

El primero se usa exclusivamente para asignar un valor a una variable. El segundo para comparar valores

```
In [17]: a=10
```

```
In [18]: a
Out[18]: 10
```

```
In [19]: a==1
Out[19]: False
```

```
In [20]: a
Out[20]: 10
```



# Una asignación NO es una ecuación

Las asignaciones no son ecuaciones matemáticas, por mucho que su aspecto nos recuerde a estas



```
In [27]: x = 3
```

Se asigna a la variable x el valor 3

```
In [28]: x
```

```
Out[28]: 3
```

```
In [29]: x = x + 1
```

Se evalúa la parte derecha de la asignación (sin tener en cuenta la parte izquierda). El valor de x es 3, que sumado a 1 da 4

```
In [30]: x
```

```
Out[30]: 4
```

¡Esto NO es una ecuación, es una asignación!

# Una asignación NO es una igualdad

```
In [1]: a = 10
```

Se asigna a la variable a el valor 10

```
In [2]: b = a
```

```
In [3]: a
```

```
Out[3]: 10
```

Se asigna a la variable b el valor de la variable a, pero son cajas diferentes

```
In [4]: b
```

```
Out[4]: 10
```

```
In [5]: a = 50
```

Se asigna a la variable a el valor 50

```
In [6]: a
```

```
Out[6]: 50
```

La variable b conserva su valor

```
In [7]: b
```

```
Out[7]: 10
```



# Un string NO es un identificador



Muchas personas confunden un string con un identificador de variable y viceversa. **No son la misma cosa**

Se asigna a la variable **a** el valor **1**. Como **a** es el nombre de una variable, es decir, un identificador, no va encerrado entre comillas

```
In [1]: a=1
```

```
In [2]: 'a'
```

```
Out[2]: 'a'
```

```
In [3]: a
```

```
Out[3]: 1
```

• Este es el string 'a'

• Esta es la variable a

Solo se puede asignar valores a variables, nunca a strings porque no son identificadores

```
In [4]: 'x' = 2
```

```
File "<ipython-input-4-171badc3e1dc>", line 1
```

```
'x' = 2
```

```
^
```

```
SyntaxError: can't assign to literal
```

# No inicializar variables

- En Python, la primera operación sobre una variable debe ser la asignación de un valor
- No se puede usar una variable a la que no se le ha asignado previamente un valor:

```
In [73]: a+2
```

```
Traceback (most recent call last):
```

```
File "<ipython-input-73-a917fe5125e3>", line 1, in <module>
    a+2
```

```
NameError: name 'a' is not defined
```