

Variables en Python

Ing. Ezequiel González Busquin

Algoritmos y Programación I
FIUBA - 2ºC 2021

Asignación

```
>>> n = 10  
>>> n  
10  
>>> print(n)  
10  
>>> █
```

Asignación encadenada

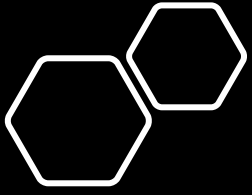
```
>>> a = b = c = 100
>>> a
100
>>> b
100
>>> c
100
>>> █
```

Tipo de la
variable

```
>>> x = 105.5
>>> print(x)
105.5
>>> x = "FM HIT"
>>> print(x)
FM HIT
>>> █
```

type(x)

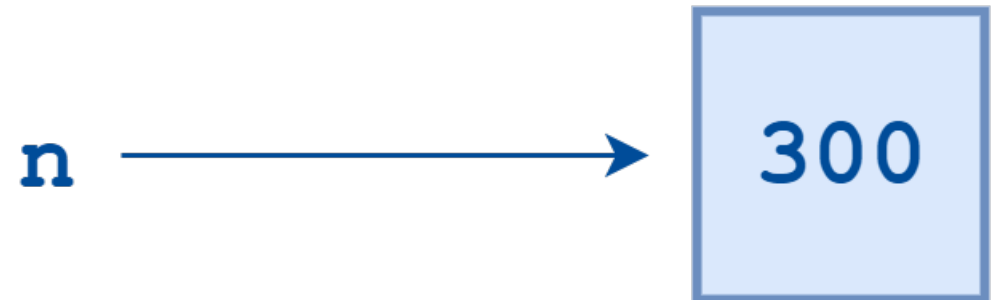
```
>>> x = 105.5
>>> type(x)
<class 'float'>
>>> x = "FM HIT"
>>> type(x)
<class 'str'>
>>> █
```



Referencias a los objetos

- Al inicializar una variable se crea un objeto, se le asigna el valor y se hace “apuntar” el identificador al objeto
- Se mantiene un **contador de referencias** a cada objeto
- Al inicializar por primera vez, el contador vale 1

```
>>> n = 300  
>>> 
```



Se puede ver que el
identificador n hace
referencia a un entero
con el valor 300

```
>>> print(n)
300
>>> type(n)
<class 'int'>
>>> █
```

Incrementando referencias

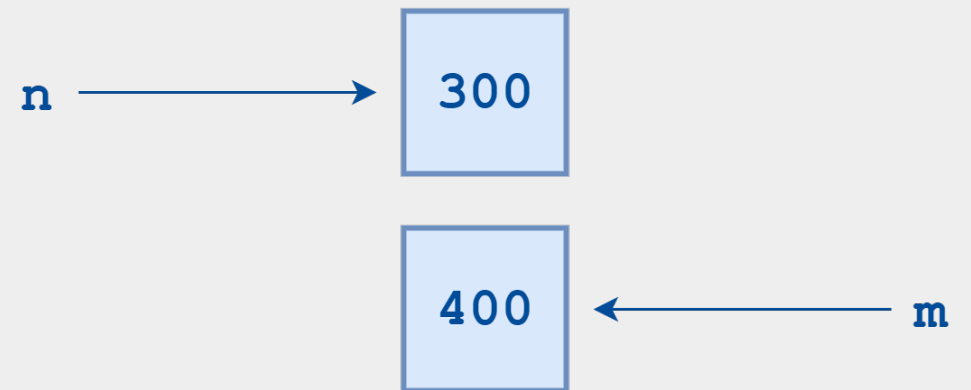
- Cuando se ejecuta el código, ambos identificadores hacen referencia a la misma posición de memoria, que es donde se almacena un objeto de tipo int y con el valor 300



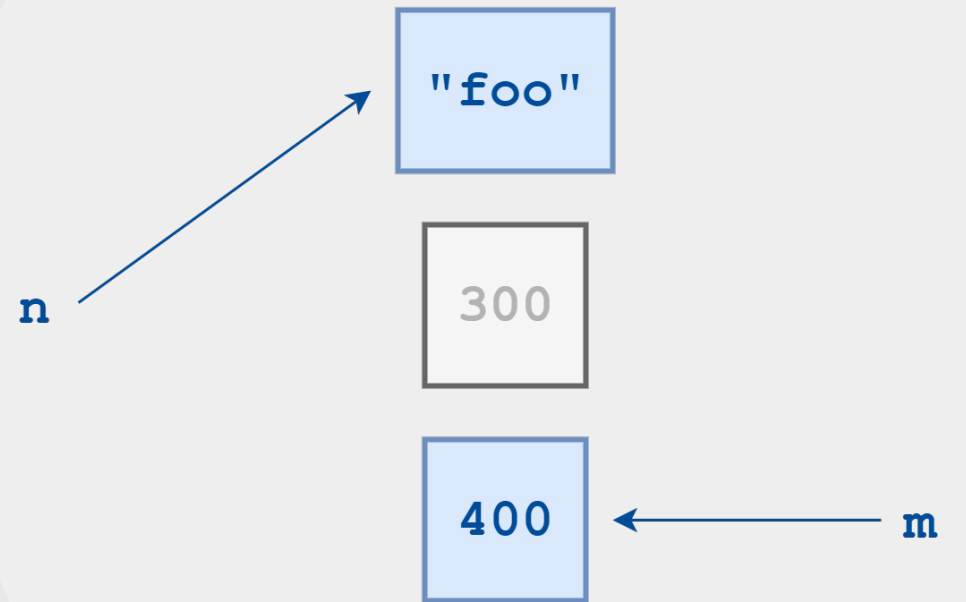
```
>>> m = n
>>> print(m, n)
300 300
>>> type(m)
<class 'int'>
>>>
```



```
>>> m = 400
>>> █
```



```
>>> n = "foo"  
>>> █
```



Tiempo de vida del objeto

Se dice que el objeto "nace" cuando se hace la primera referencia a él. Se reserva la memoria para almacenar su valor según el tipo de dato.

El objeto deja de existir cuando la cantidad de referencias que se le hacen cae a cero.

El intérprete de Python lleva cuenta de las referencias y cuando llegan a cero se ejecuta el recolector de memoria (*garbage collector*) para liberar la memoria que ya no se usa más.

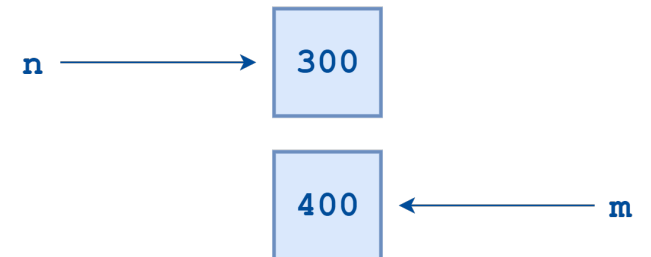
Se utiliza la función `id` para obtener un identificador único del objeto en memoria.

Identidad del objeto

```
>>> n = 300
>>> m = n
>>> id(n)
4301841136
>>> id(m)
4301841136
```



```
>>> m = 400
>>> id(m)
4301841424
>>>
```



Caché de enteros “chicos”

- Python crea un caché con los enteros entre $[-5, 256]$ para evitar crear objetos con los números más comunes

```
>>> n = 300
>>> m = 300
>>> id(n)
4301841136
>>> id(m)
4301841296
```

```
>>> n = 30
>>> m = 30
>>> id(n)
4299137696
>>> id(m)
4299137696
```

Nombres de las variables

Se permiten los caracteres alfabéticos, números y el guión bajo


Se diferencia entre mayúsculas y minúsculas (*case sensitive*)

No pueden empezar con número

No pueden llamarse como las palabras reservadas del lenguaje

Revisar el PEP8 y el PEP7540 (de la cátedra) para leer sobre las nomenclaturas camelCase, PascalCase, snake_case

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



Palabras reservadas de Python 3.6

- Fuente: <https://realpython.com/python-variables/>