VALUE - TYPE & VARIABLE

And the Assignment Operator

by

Lilian Blot

A value is one of the fundamental things that a program manipulates

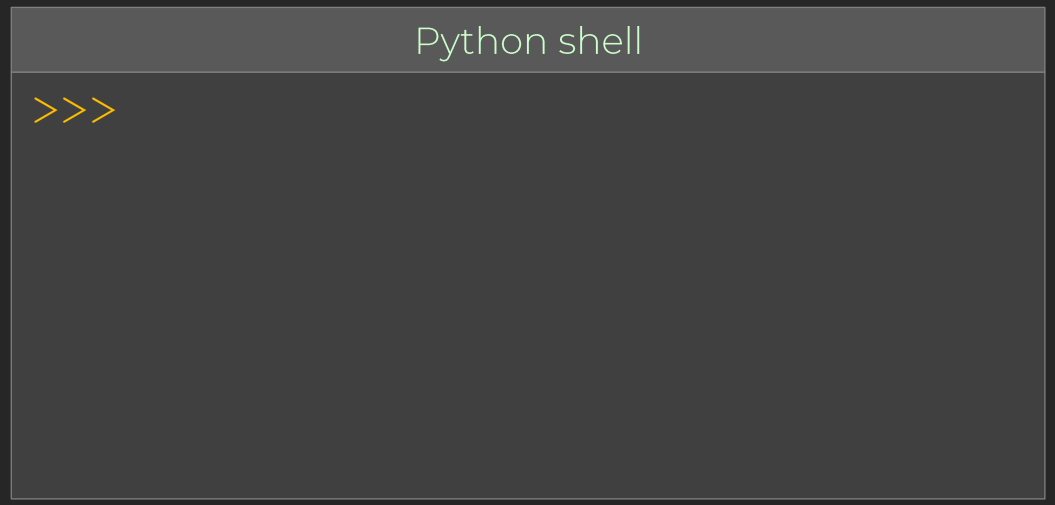
A value can be:

- a letter ‘L’
- a word ‘Lilian’
- a number 1, 3.14, 1.2e3 (i.e. 1200.0)
- and more complex data (seen later in the term)

Values have a type, for example 1 and 1.0 are not represented the same way in memory. They have a different type.

Words and letters are strings (‘str’ in Python)

‘lilian blot’ or “lilian blot”



Words and letters are strings (‘str’ in Python)

‘lilian blot’ or “lilian blot”

Python shell

```
>>> type(“Lilian Blot”)
<class 'str'>
>>>
```

Words and letters are strings (‘str’ in Python)

‘lilian blot’ or “lilian blot”

Numbers with a decimal point are called float (‘float’)

124.0, 124e2, 0.123

Python shell

```
>>> type(“Lilian Blot”)
<class 'str'>
>>>
```

Words and letters are strings (‘str’ in Python)

‘lilian blot’ or “lilian blot”

Numbers with a decimal point are called float (‘float’)

124.0, 124e2, 0.123

Python shell

```
>>> type(“Lilian Blot”)
<class 'str'>
>>> type(190.0)
<class 'float'>
>>>
```

Words and letters are strings (‘str’ in Python)

‘lilian blot’ or “lilian blot”

Numbers with a decimal point are called float (‘float’)

124.0, 124e2, 0.123

Integer (‘int’), are whole numbers

1, -124, 0

Python shell

```
>>> type(“Lilian Blot”)
<class 'str'>
>>> type(190.0)
<class 'float'>
>>>
```

Words and letters are strings (‘str’ in Python)

‘lilian blot’ or “lilian blot”,

Numbers with a decimal point are called float (‘float’)

124.0, 124e2, 0.123

Integer (‘int’), are whole numbers

1, -124, 0

Python shell

```
>>> type(“Lilian Blot”)
<class 'str'>
>>> type(190.0)
<class 'float'>
>>> type(190)
<class 'int'>
>>>
```

Words and letters are strings (‘str’ in Python)

‘lilian blot’ or “lilian blot”,

Numbers with a decimal point are called float (‘float’)

124.0, 124e2, 0.123

Integer (‘int’), are whole numbers

1, -124, 0

Python shell

```
>>> type(“Lilian Blot”)
<class 'str'>
>>> type(190.0)
<class 'float'>
>>> type(190)
<class 'int'>
>>>type('17')
<class 'str'>
```

A critical aspect of programming is the means it provides for using names to refer to computational objects.

We say that the name identifies a variable whose value is the object.

Python uses the assignment operator = to link a variable to a value.

Variables have type and the type of a variable is the type of the value it refers to.

Python shell

```
>>> name = 'Lilian Blot'
```

Python shell

```
>>> name = 'Lilian Blot'
```

Memory space

Python shell

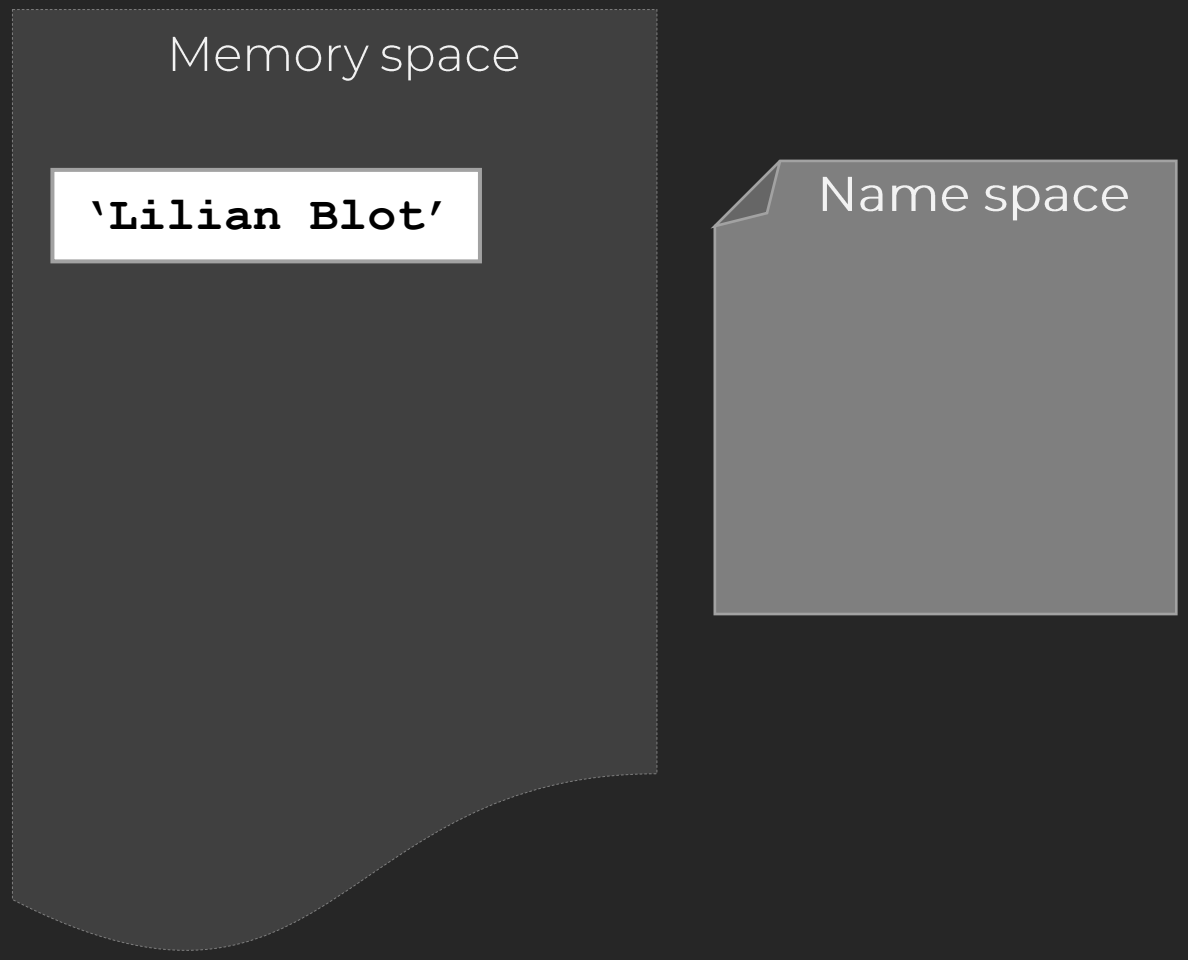
```
>>> name = 'Lilian Blot'
```

Memory space

Name space

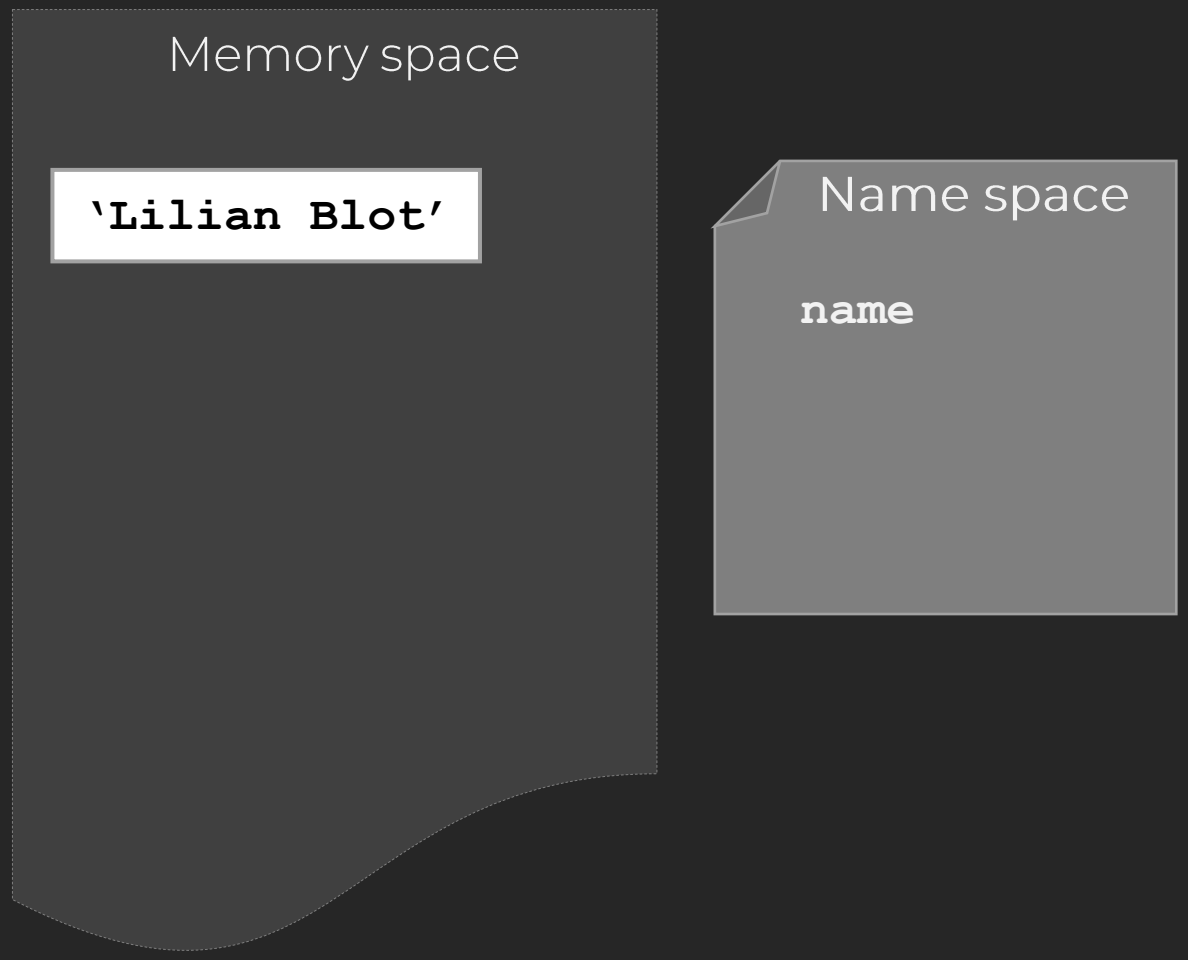
Python shell

```
>>> name = 'Lilian Blot'
>>>
```



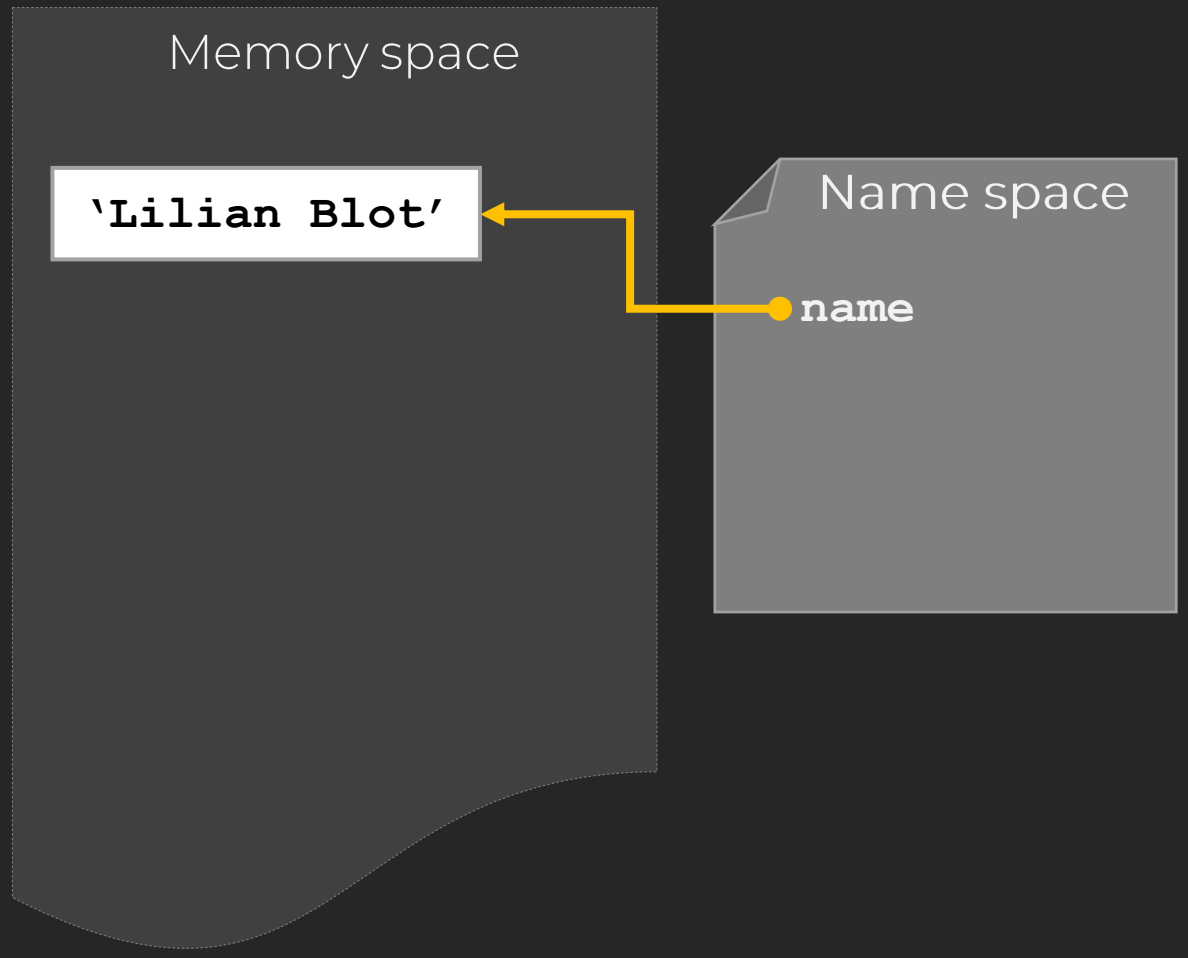
Python shell

```
>>> name = 'Lilian Blot'
>>>
```



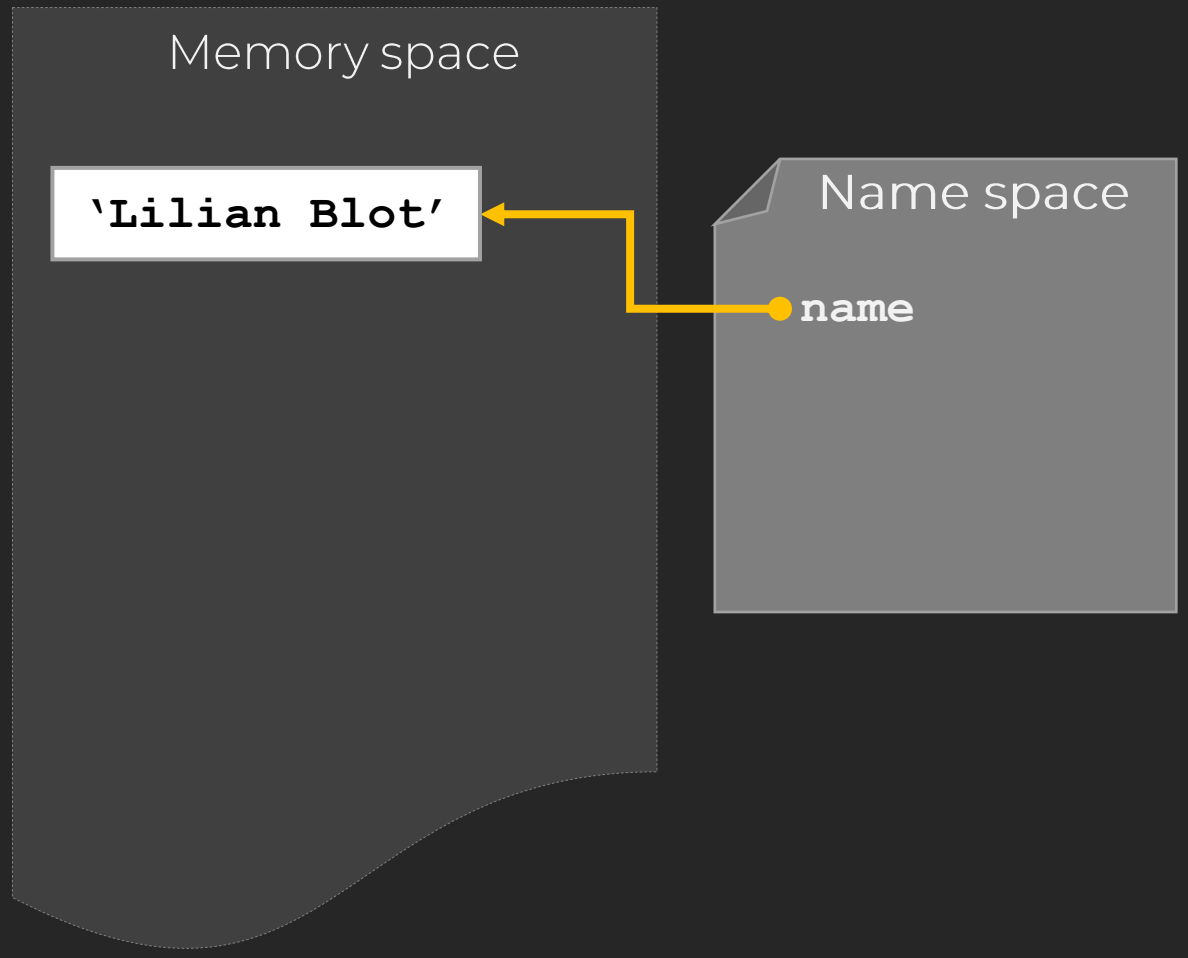
Python shell

```
>>> name = 'Lilian Blot'
>>>
```



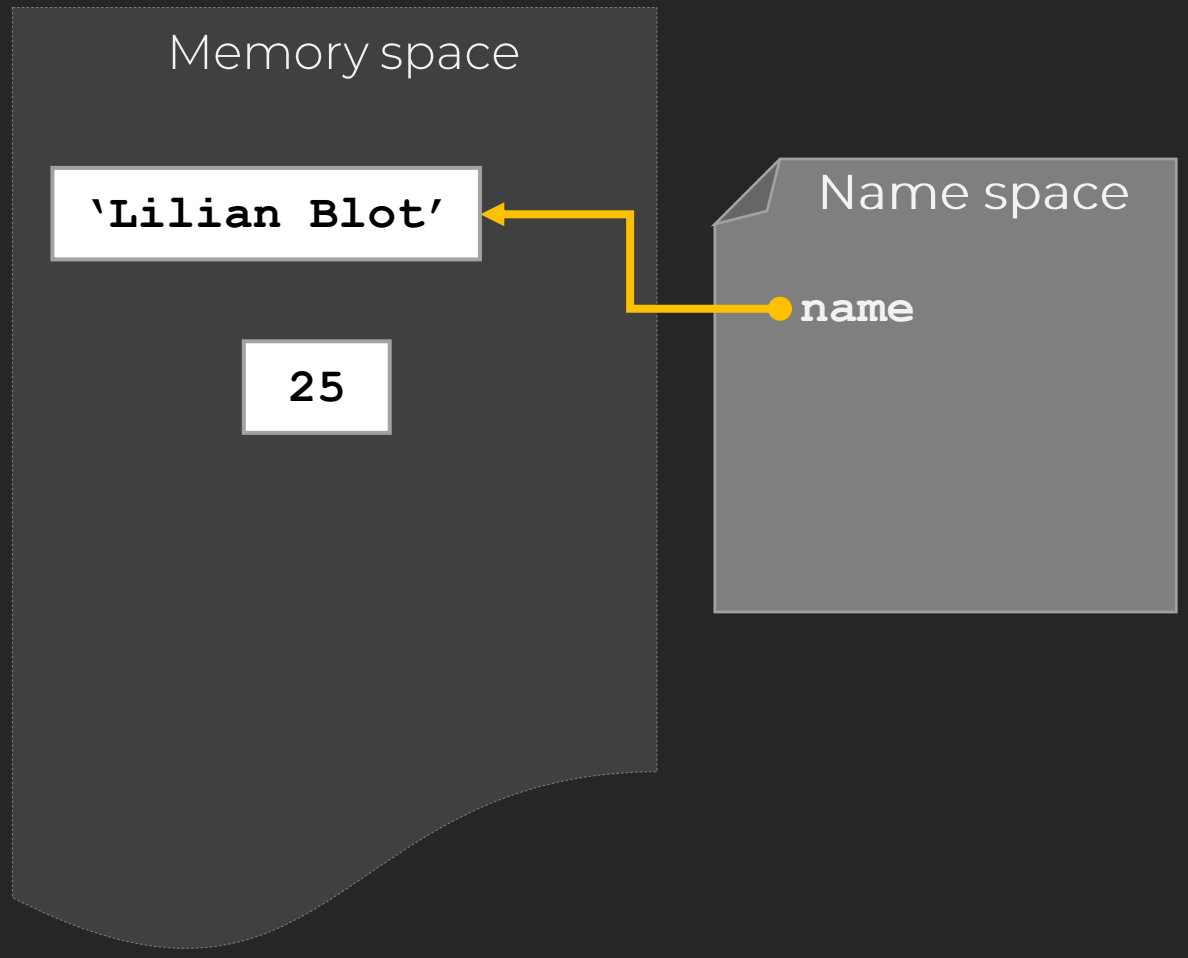
Python shell

```
>>> name = 'Lilian Blot'
>>> age = 25
```



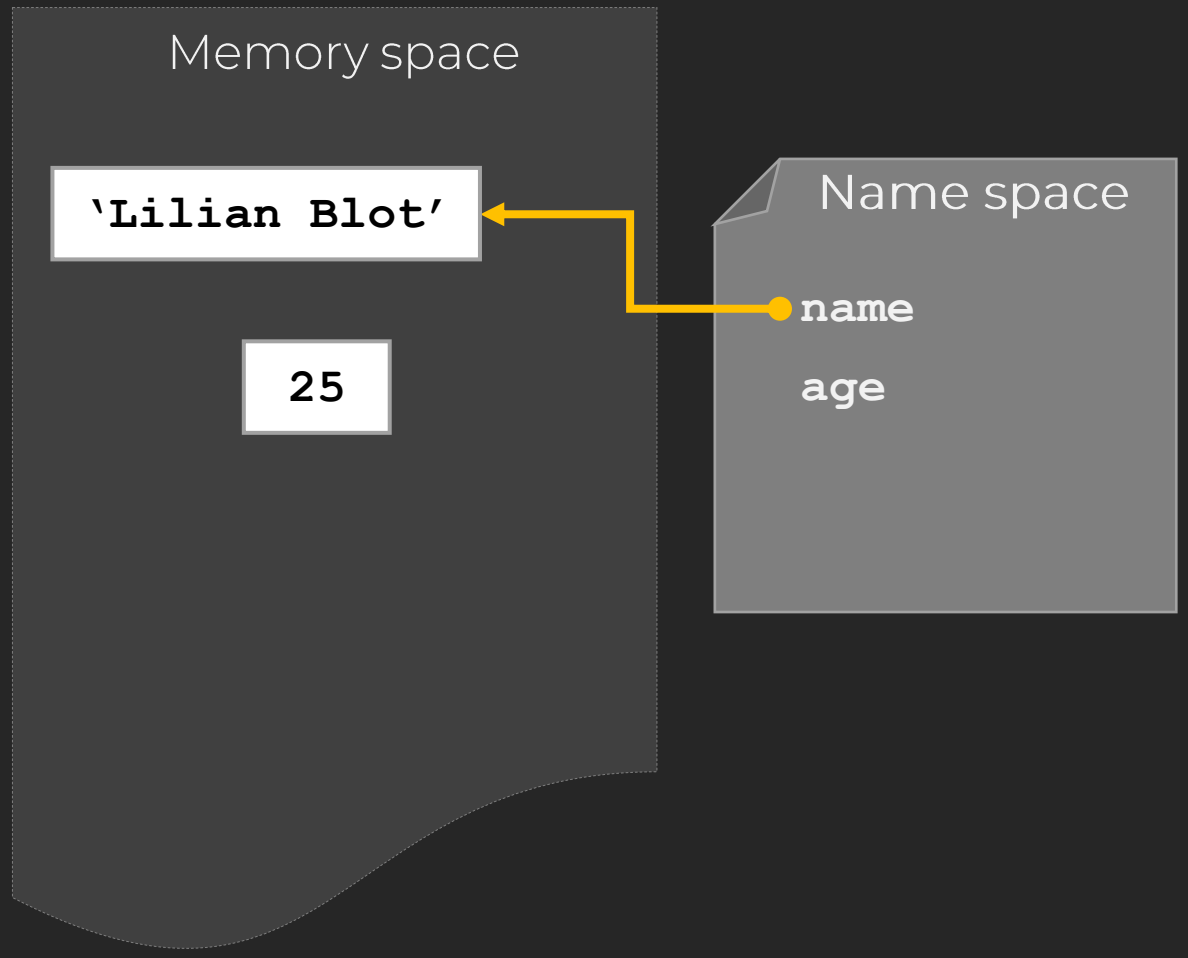
Python shell

```
>>> name = 'Lilian Blot'
>>> age = 25
>>>
```



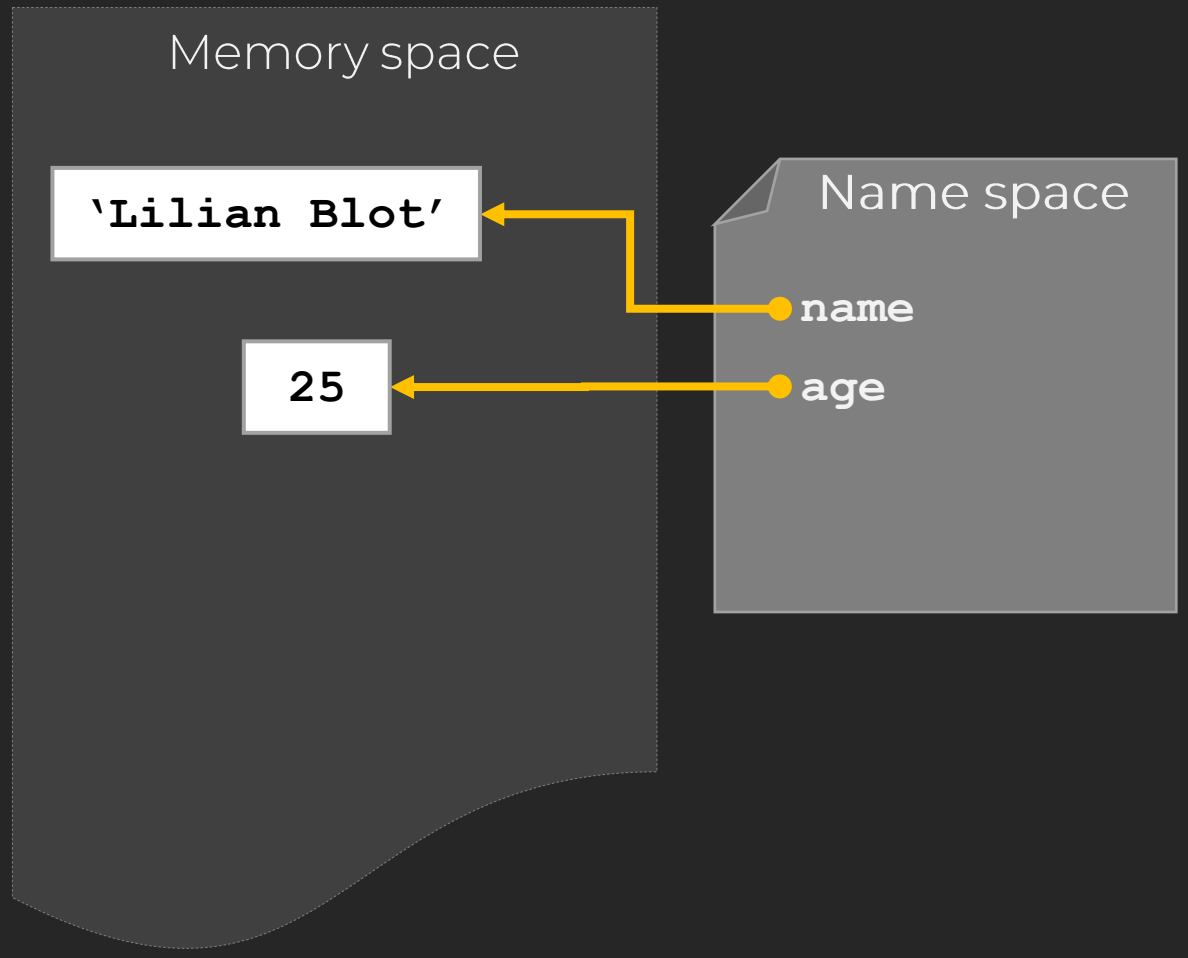
Python shell

```
>>> name = 'Lilian Blot'
>>> age = 25
>>>
```



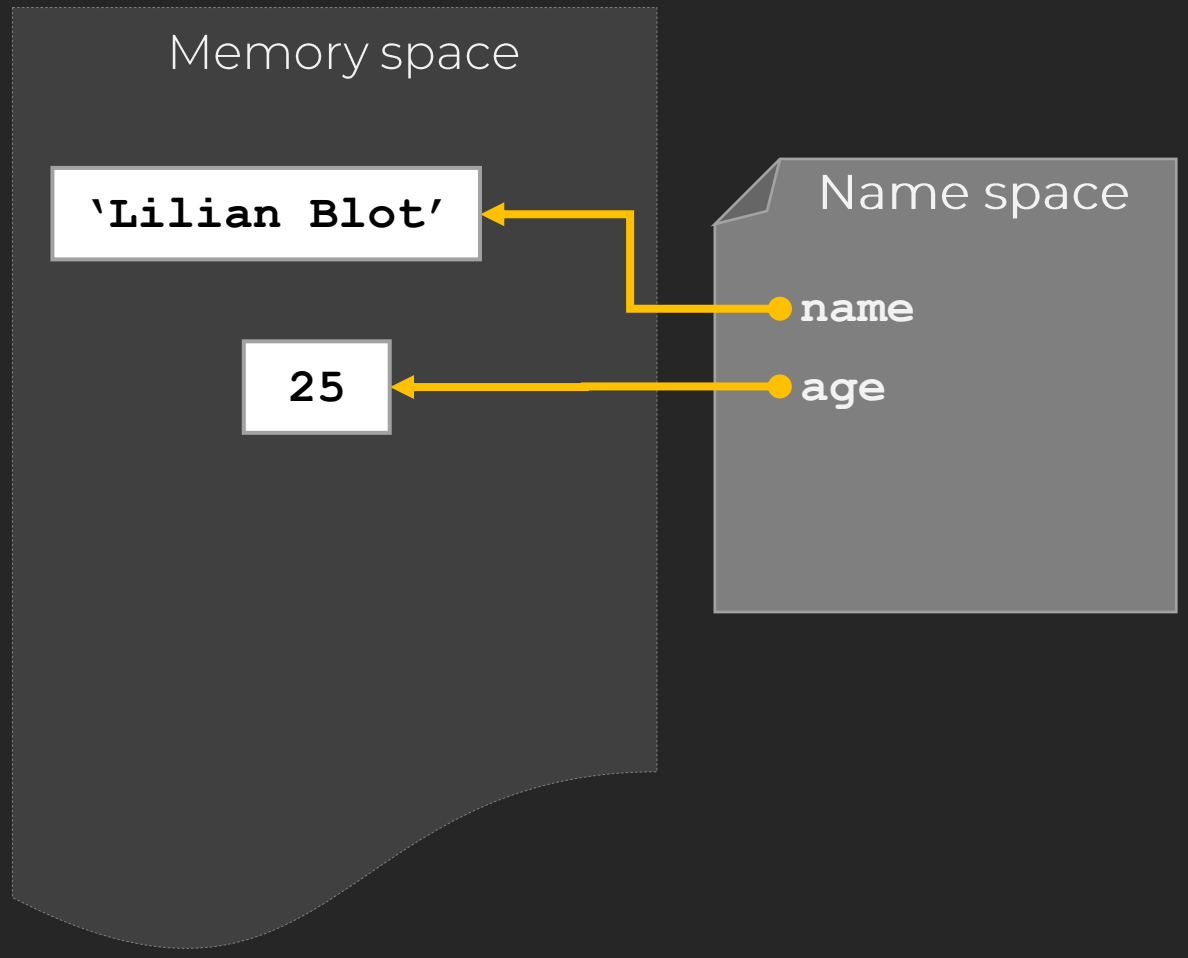
Python shell

```
>>> name = 'Lilian Blot'
>>> age = 25
>>>
```



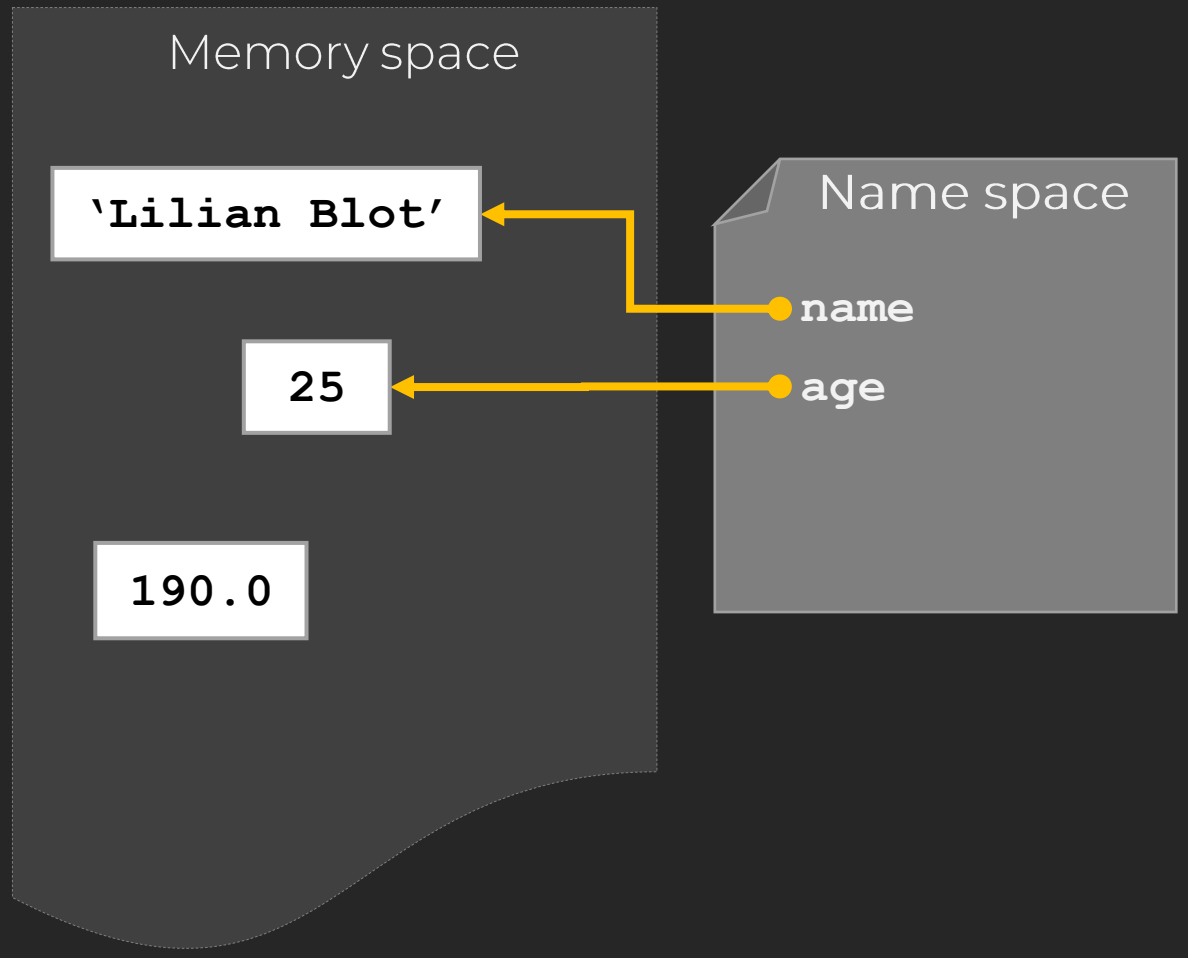
Python shell

```
>>> name = 'Lilian Blot'
>>> age = 25
>>> height_cm = 190.0
```



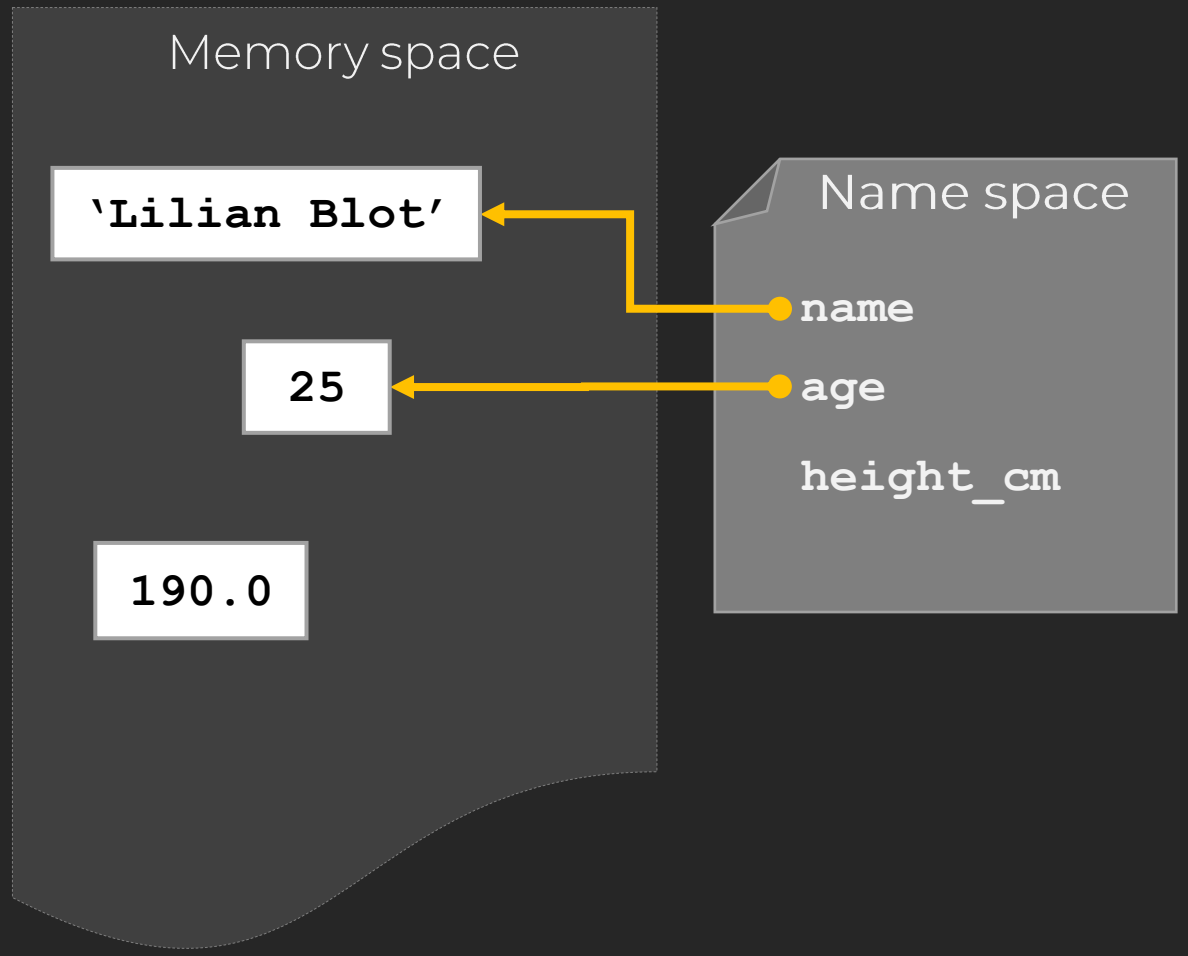
Python shell

```
>>> name = 'Lilian Blot'
>>> age = 25
>>> height_cm = 190.0
>>>
```



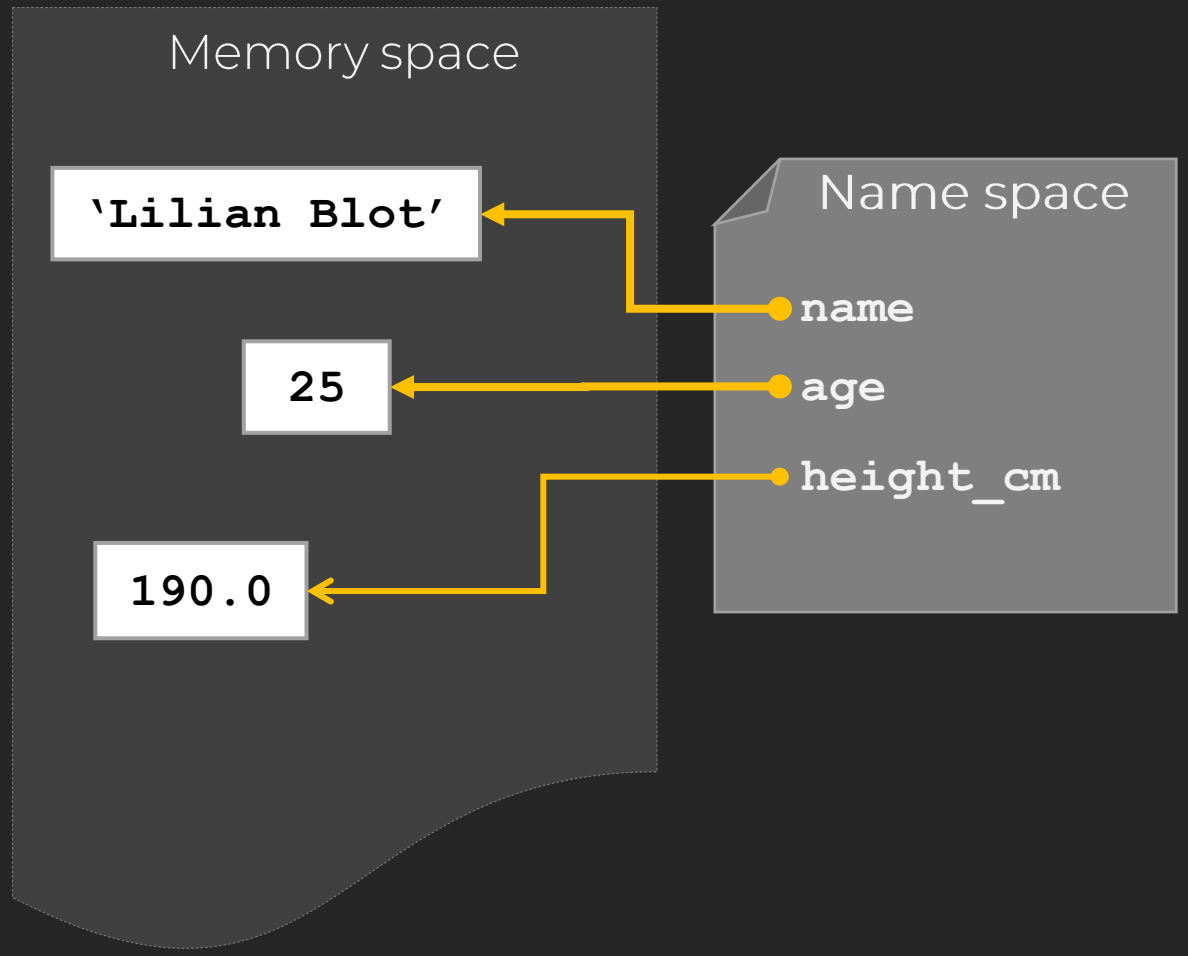
Python shell

```
>>> name = 'Lilian Blot'
>>> age = 25
>>> height_cm = 190.0
>>>
```



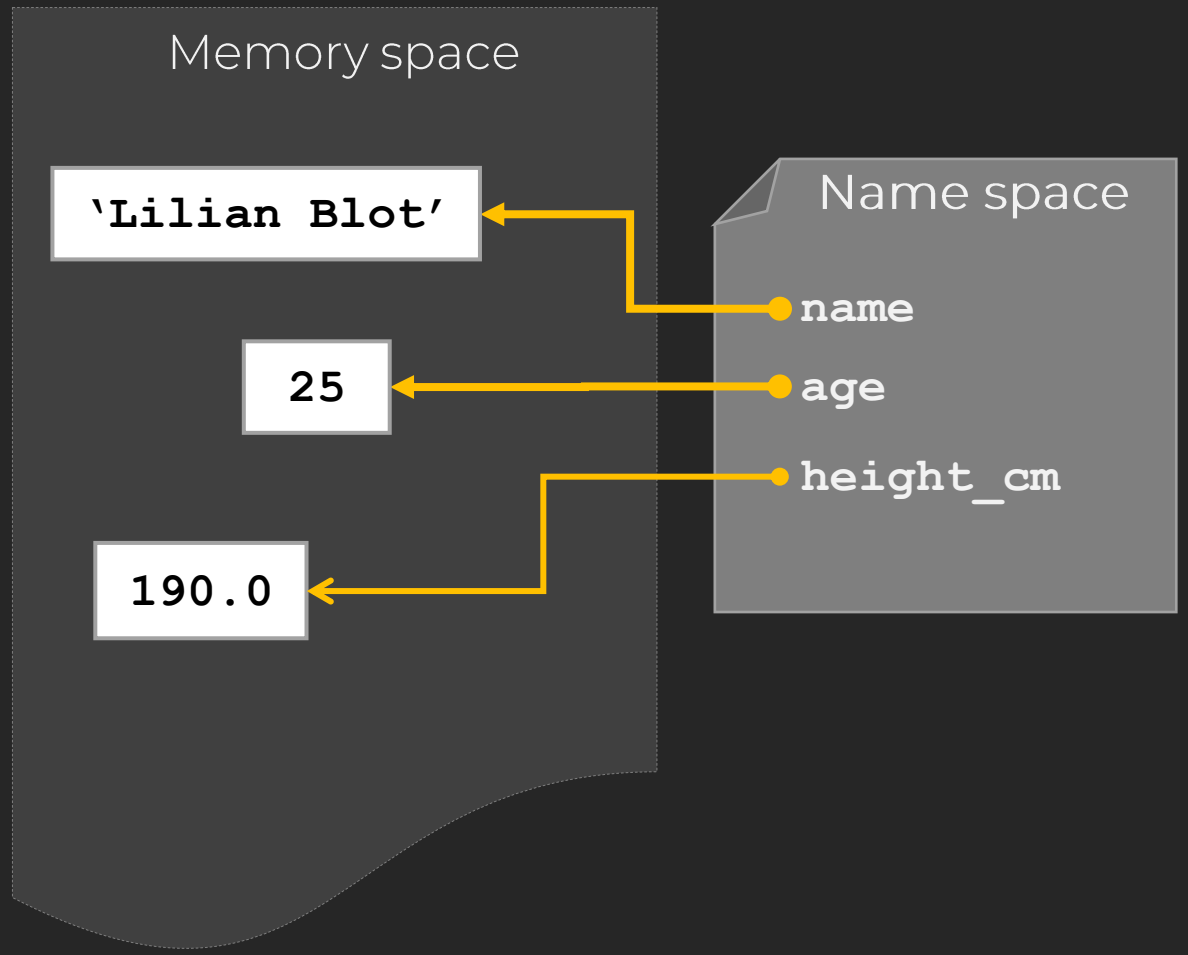
Python shell

```
>>> name = 'Lilian Blot'
>>> age = 25
>>> height_cm = 190.0
>>>
```



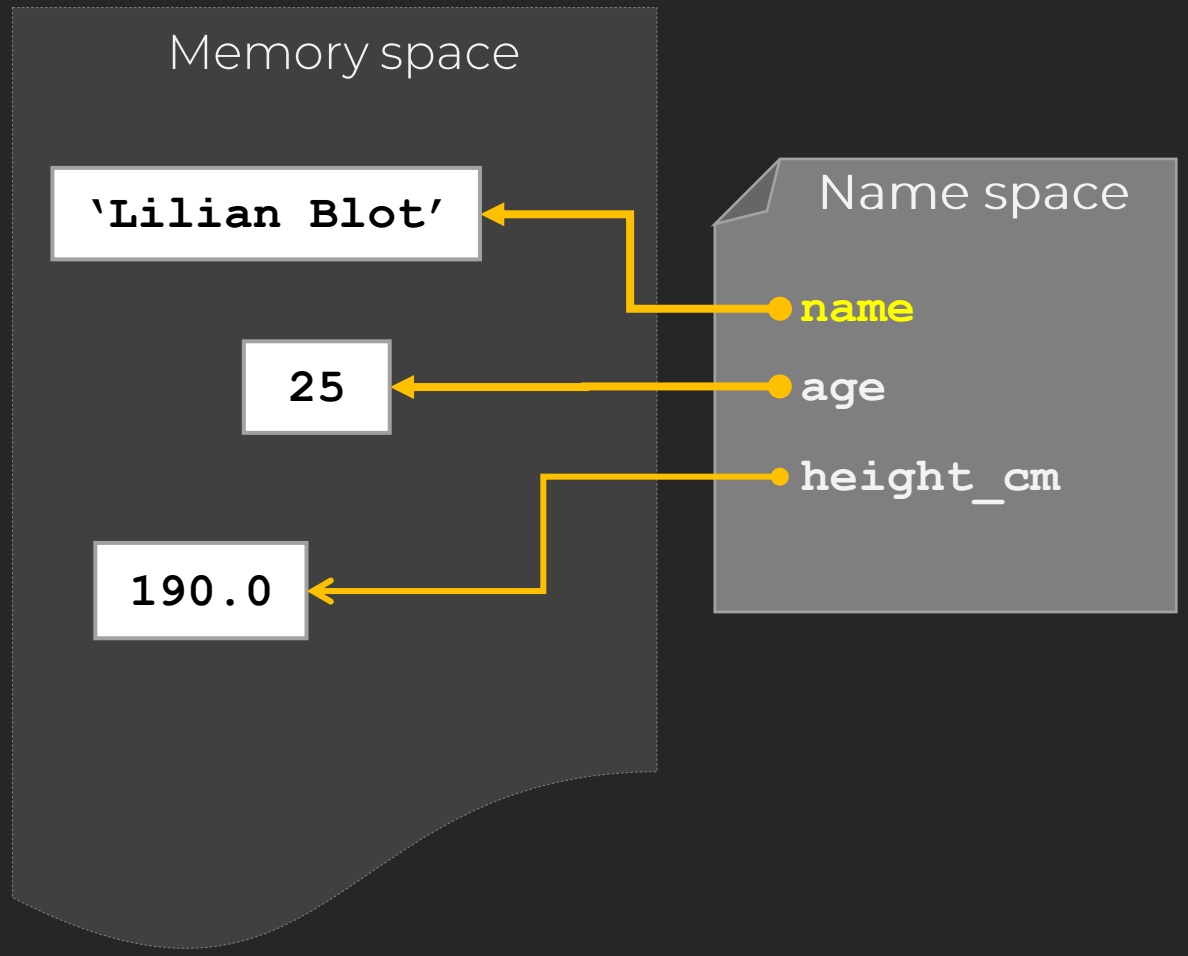
Python shell

```
>>> name = 'Lilian Blot'
>>> age = 25
>>> height_cm = 190.0
>>> print(name)
```



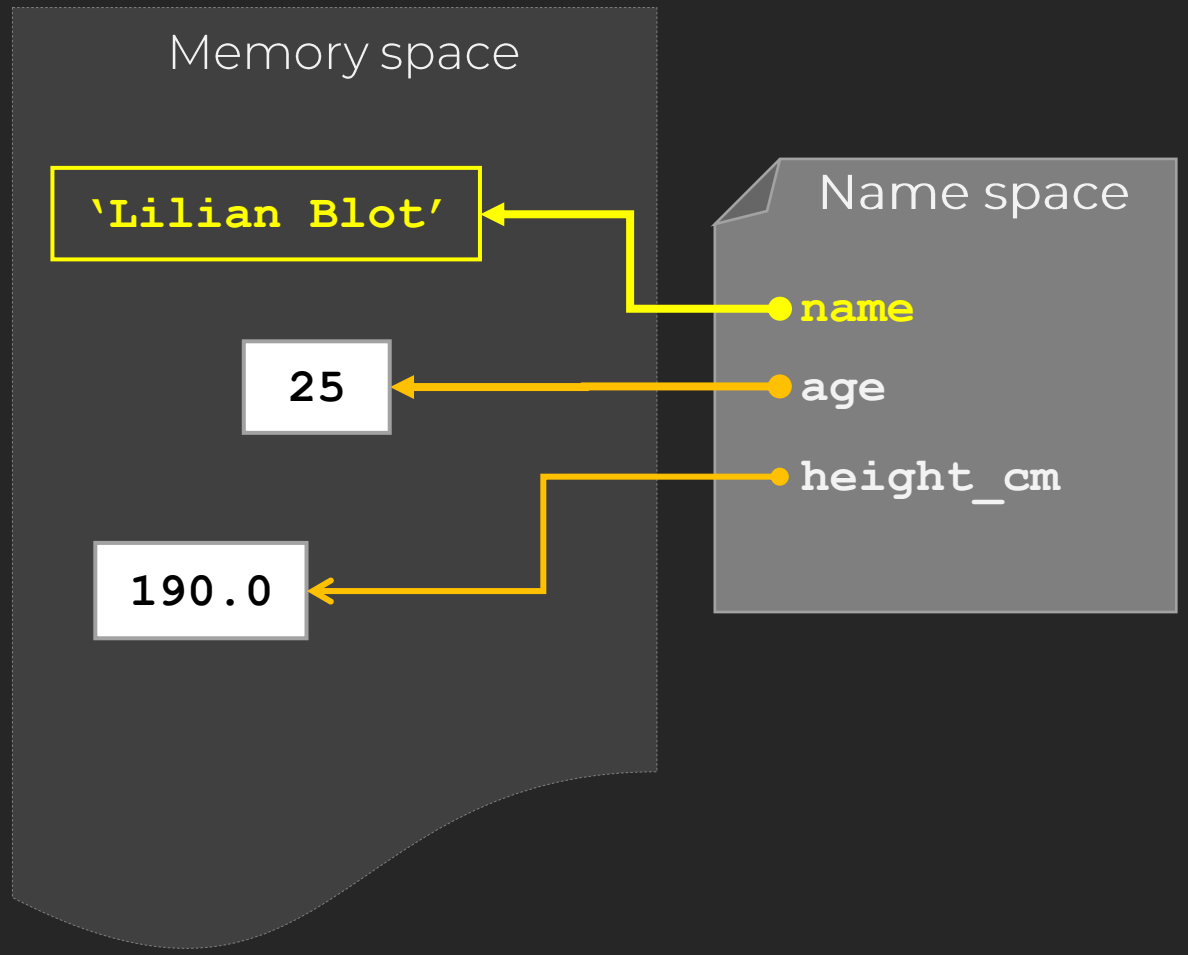
Python shell

```
>>> name = 'Lilian Blot'
>>> age = 25
>>> height_cm = 190.0
>>> print(name)
```



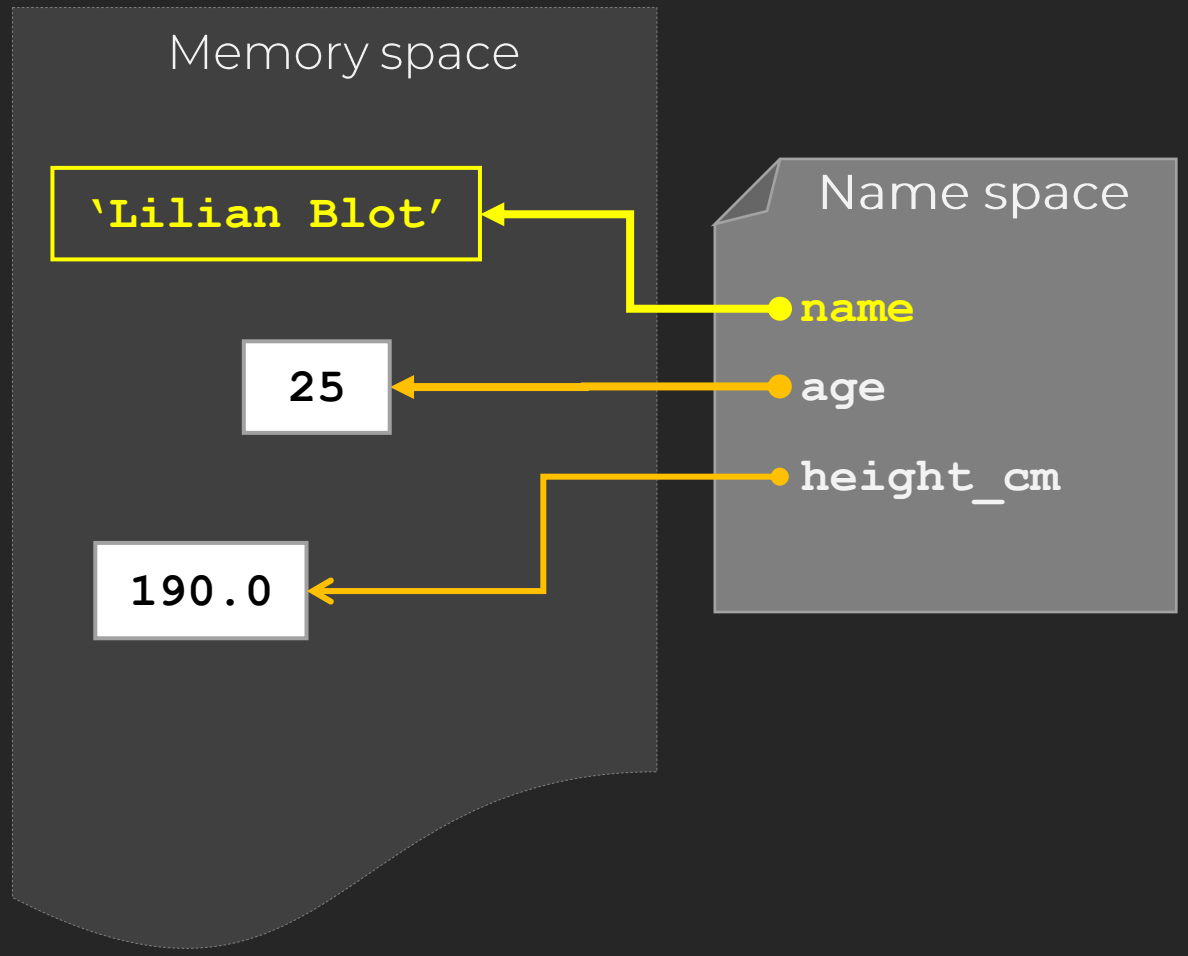
Python shell

```
>>> name = 'Lilian Blot'
>>> age = 25
>>> height_cm = 190.0
>>> print(name)
```



Python shell

```
>>> name = 'Lilian Blot'
>>> age = 25
>>> height_cm = 190.0
>>> print(name)
Lilian Blot
>>>
```



- variable names must start with a letter or an underscore
- variable names can contain letters, numbers and underscores
- variable names can be arbitrary long
- Python is case sensitive, so **`size`** and **`SIZE`** are two different variables
- by convention uppercase letters are not used
- multiple words names use underscore, e.g. book_title, book_price, ...

Language's rules and structures are defined by keywords. Such keywords cannot be used as variable names.

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

Python's 33 reserved keywords

We have seen how to create variables and assign them values of different types. In the next video, we will be looking at creating more complex expressions using values, variables and operators.