Python

Basics

A simple interpreted language

A simple interpreted language

no separate compilation step

A simple interpreted language

no separate compilation step

```
$ python
>>>
```

A simple interpreted language

no separate compilation step

```
$ python
>>> print 1 + 2
3
>>>
```

A simple interpreted language

no separate compilation step

```
$ python
>>> print 1 + 2
3
>>> print 'charles' + 'darwin'
charlesdarwin
>>>
```

Put commands in a file and execute that

Put commands in a file and execute that

```
$ nano very-simple.py
```

Put commands in a file and execute that

```
$ nano very-simple.py
```

```
print 1 + 2
print 'charles' + 'darwin'
```

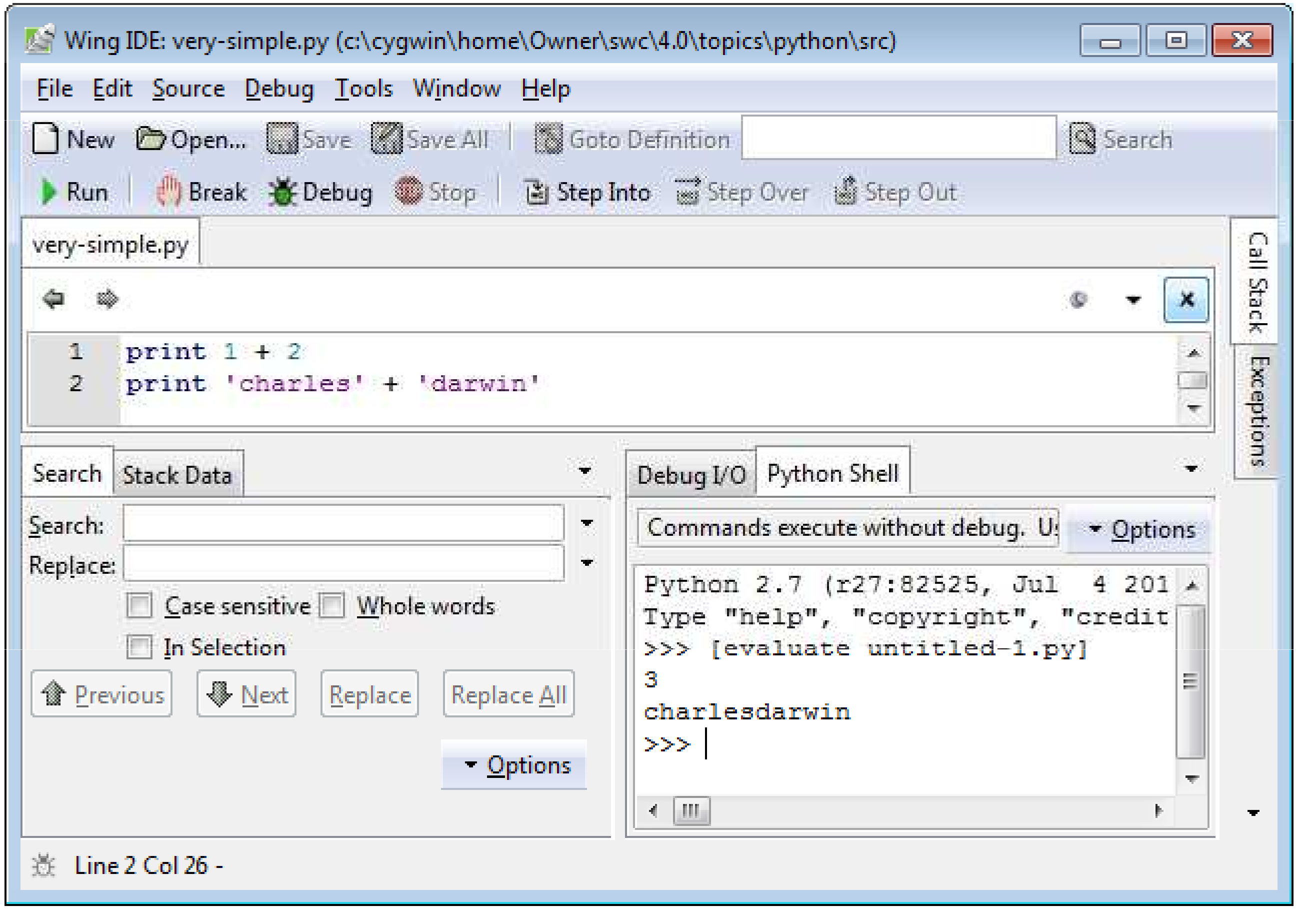
Put commands in a file and execute that

```
$ nano very-simple.py
```

```
print 1 + 2
print 'charles' + 'darwin'
```

```
$ python very-simple.py
3
charlesdarwin
$
```

Use an *integrated development environment* (IDE)



Wing IDE: very-simple.py (c:\cygwin\home\Owner\swc\4.0\topics\python\src)
File Edit Source Debug Tools Window Help
New Open... Save Save All Goto Definition Search
Run Break Debug Stop Step Into Step Over Step Out
very-simple.py
Call Stack
Exceptions
Search Stack Data
Search:
Replace:
□ Case sensitive □ Whole words
□ In Selection
Previous Next Replace Replace All
Options
Debug I/O Python Shell
Options

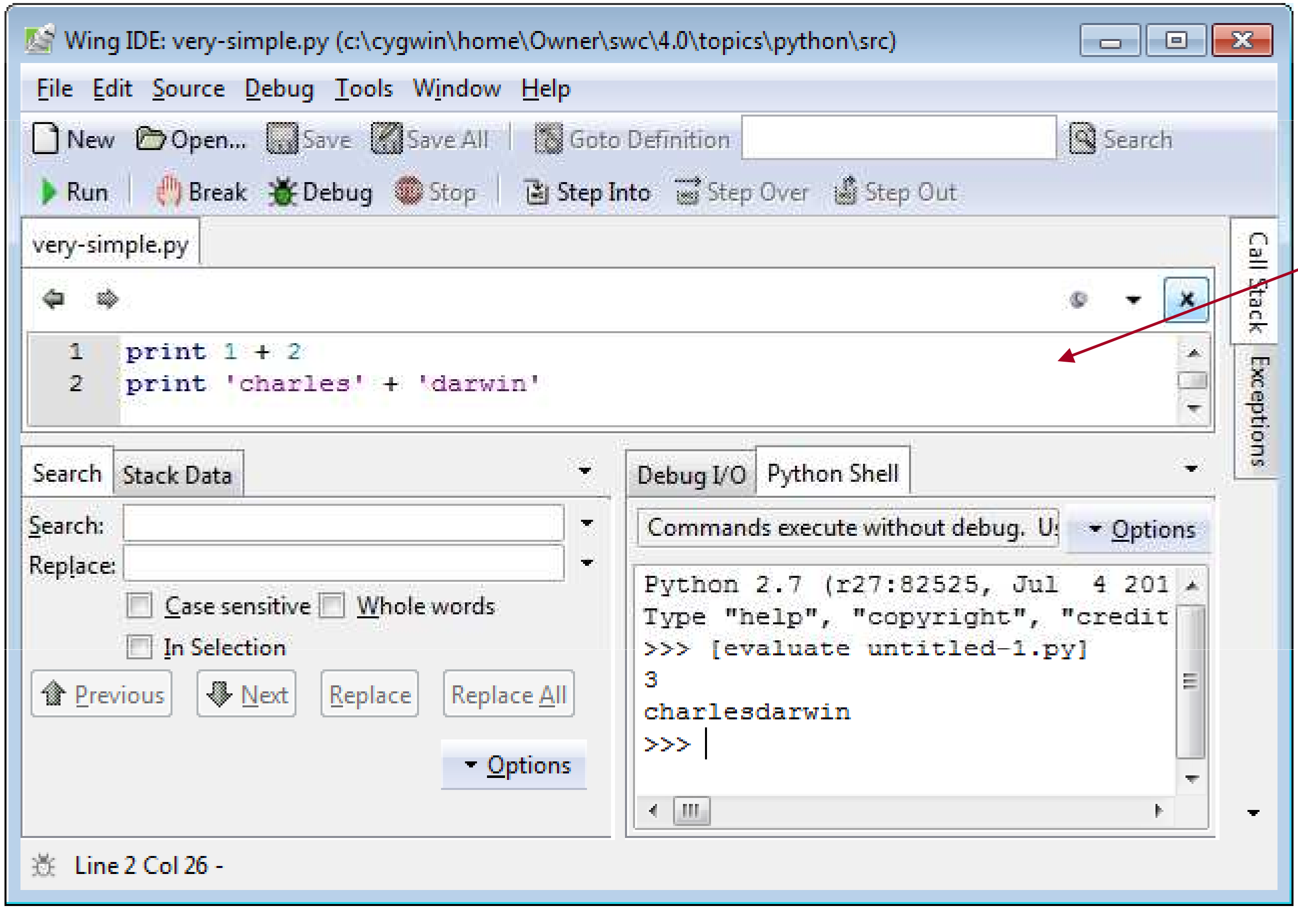
```
print 1 + 2
print 'charles' + 'darwin'
```

Commands execute without debug. U:

```
Python 2.7 (r27:82525, Jul  4 201
Type "help", "copyright", "credit
>>> [evaluate untitled-1.py]
3
charlesdarwin
>>> |
```

Line 2 Col 26 -

Use an *integrated development environment* (IDE)


Wing IDE: very-simple.py (c:\cygwin\home\Owner\swc\4.0\topics\python\src)
File Edit Source Debug Tools Window Help
New Open... Save Save All Goto Definition Search
Run Break Debug Stop Step Into Step Over Step Out
very-simple.py
Call Stack
Exceptions
Search Stack Data
Search:
Replace:
□ Case sensitive □ Whole words
□ In Selection
Previous Next Replace Replace All
Options
Debug I/O Python Shell
Commands execute without debug. U:
Options
Line 2 Col 26 -

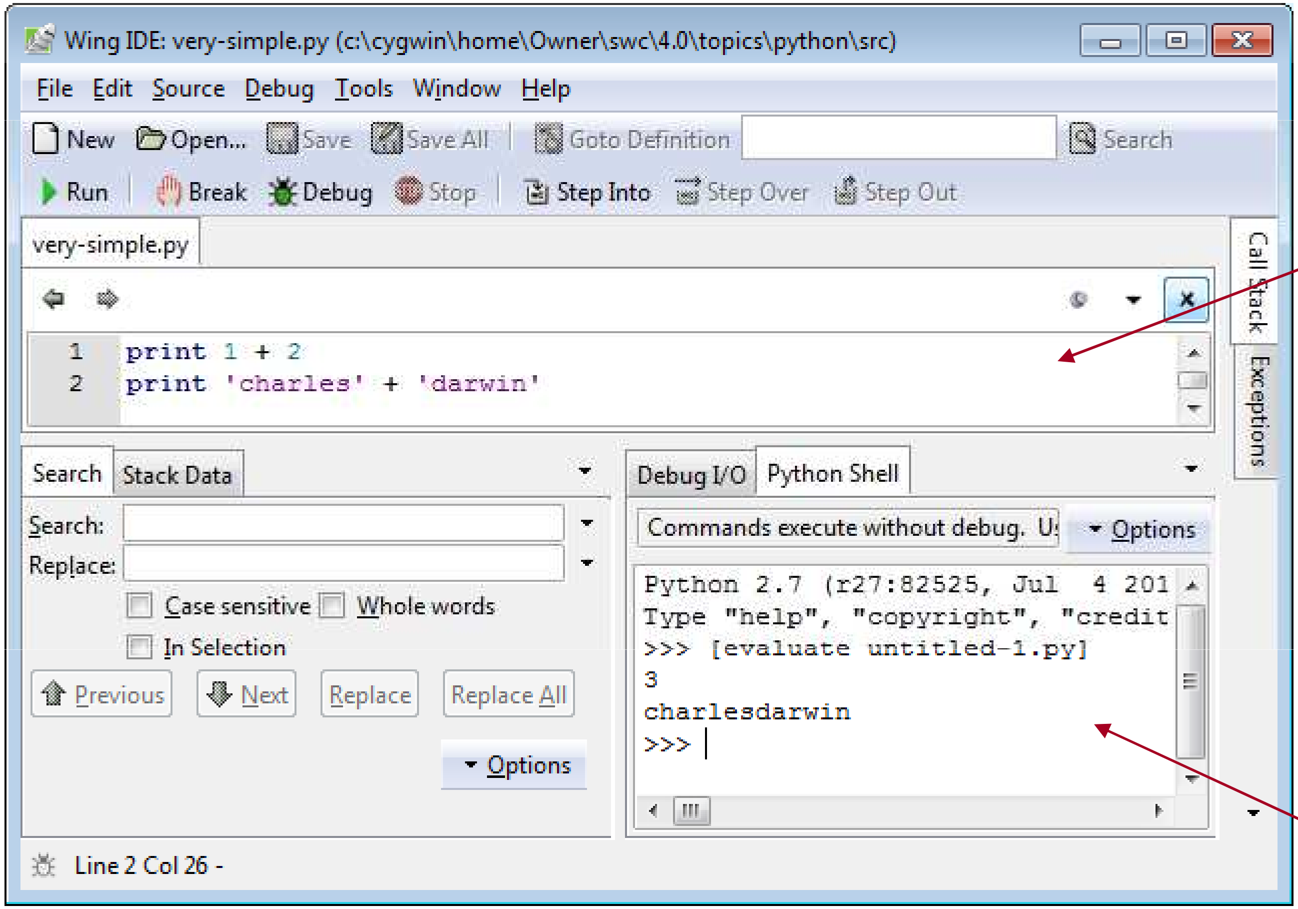
```
print 1 + 2
print 'charles' + 'darwin'
```

```
Python 2.7 (r27:82525, Jul  4 201
Type "help", "copyright", "credit
>>> [evaluate untitled-1.py]
3
charlesdarwin
>>> |
```

Source

file

Use an *integrated development environment* (IDE)


Wing IDE: very-simple.py (c:\cygwin\home\Owner\swc\4.0\topics\python\src)
File Edit Source Debug Tools Window Help
New Open... Save Save All Goto Definition Search
Run Break Debug Stop Step Into Step Over Step Out
very-simple.py
Call Stack
Exceptions
Search Stack Data
Search:
Replace:
□ Case sensitive □ Whole words
□ In Selection
Previous Next Replace Replace All
Options
Debug I/O Python Shell
Commands execute without debug. U:
Options
Line 2 Col 26 -

```
print 1 + 2
print 'charles' + 'darwin'
```

```
Python 2.7 (r27:82525, Jul  4 201
Type "help", "copyright", "credit
>>> [evaluate untitled-1.py]
3
charlesdarwin
>>> 
```

Source file

Execution shell

Variables are names for values

Variables are names for values

Created by use

Variables are names for values

Created by use: no declaration necessary

Variables are names for values

Created by use: no declaration necessary

```
>>> planet = 'Pluto'
>>>
```

Variables are names for values

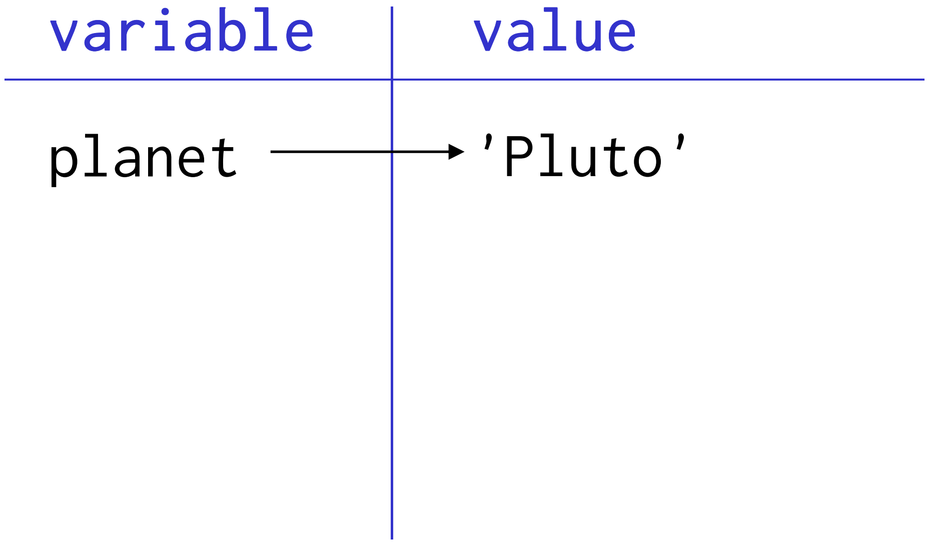
Created by use: no declaration necessary

```
>>> planet = 'Pluto'
>>> print planet
Pluto
>>>
```

Variables are names for values

Created by use: no declaration necessary

```
>>> planet = 'Pluto'
>>> print planet
Pluto
>>>
```



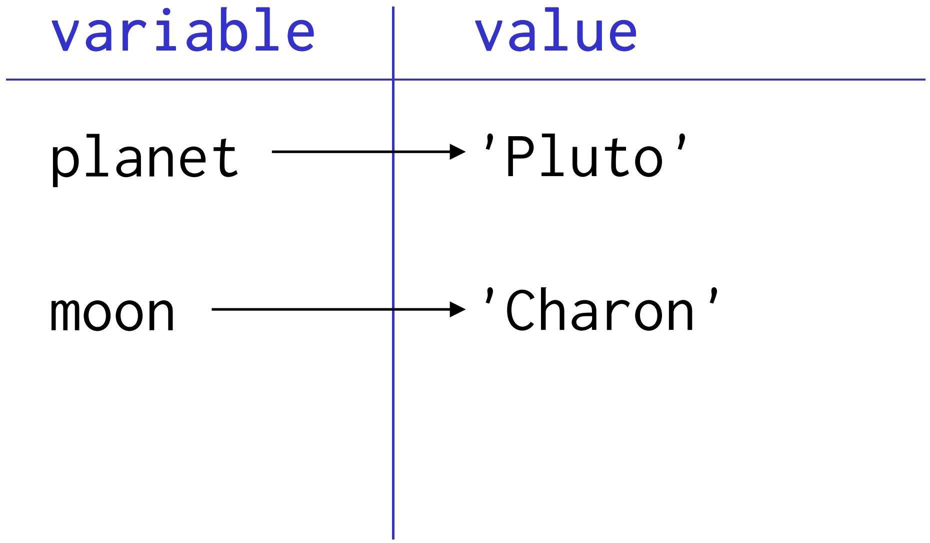
variable value

planet → 'Pluto'

Variables are names for values

Created by use: no declaration necessary

```
>>> planet = 'Pluto'
>>> print planet
Pluto
>>> moon = 'Charon'
>>>
```

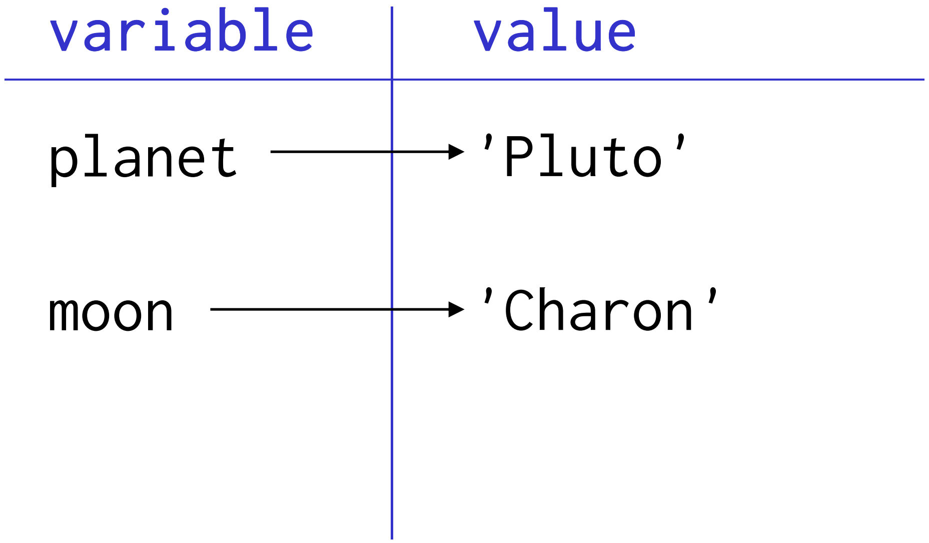


variable	value
planet	'Pluto'
moon	'Charon'

Variables are names for values

Created by use: no declaration necessary

```
>>> planet = 'Pluto'
>>> print planet
Pluto
>>> moon = 'Charon'
>>> p = planet
>>>
```

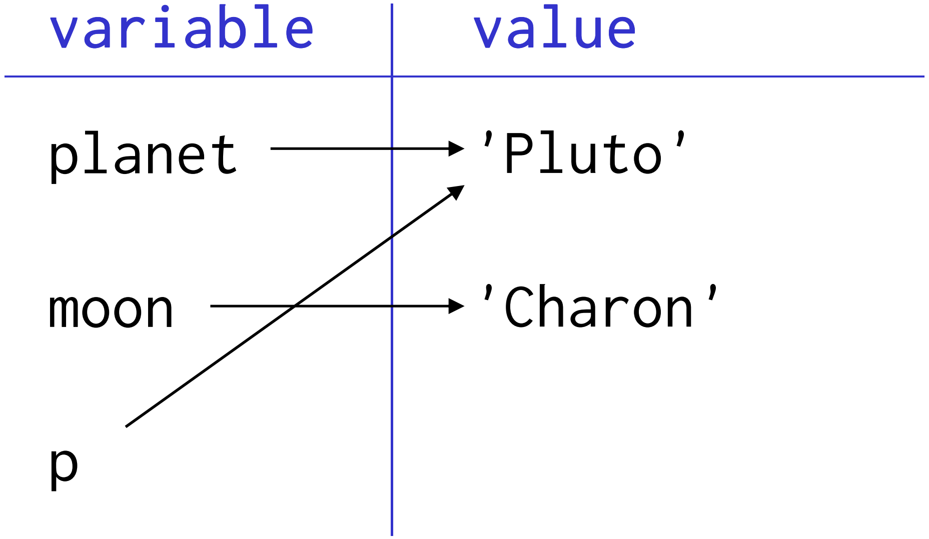


variable	value
planet →	'Pluto'
moon →	'Charon'

Variables are names for values

Created by use: no declaration necessary

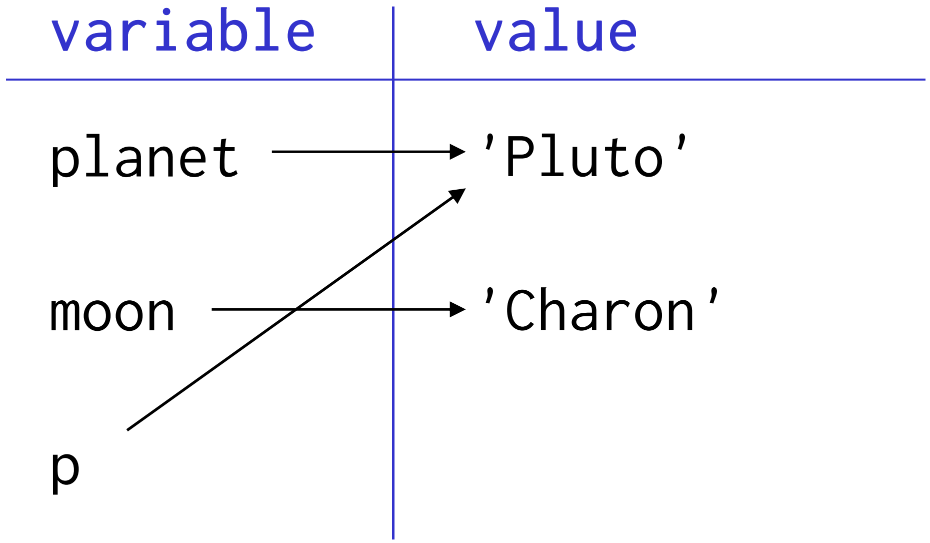
```
>>> planet = 'Pluto'
>>> print planet
Pluto
>>> moon = 'Charon'
>>> p = planet
>>>
```



Variables are names for values

Created by use: no declaration necessary

```
>>> planet = 'Pluto'
>>> print planet
Pluto
>>> moon = 'Charon'
>>> p = planet
>>> print p
Pluto
>>>
```



variable | value

planet 'Pluto'

moon 'Charon'

p

A variable is just a name

A variable is just a name

Does not have a type

A variable is just a name

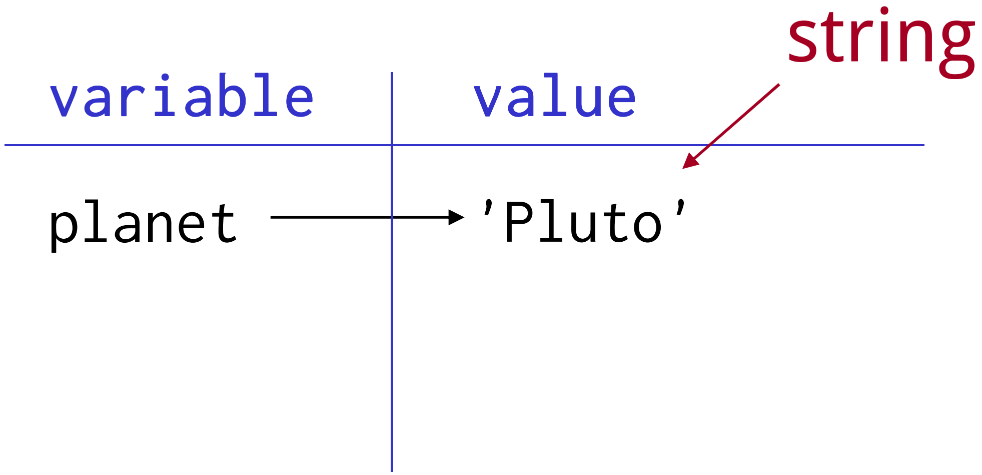
Does not have a type

```
>>> planet = ’Pluto’
>>>
```

A variable is just a name

Does not have a type

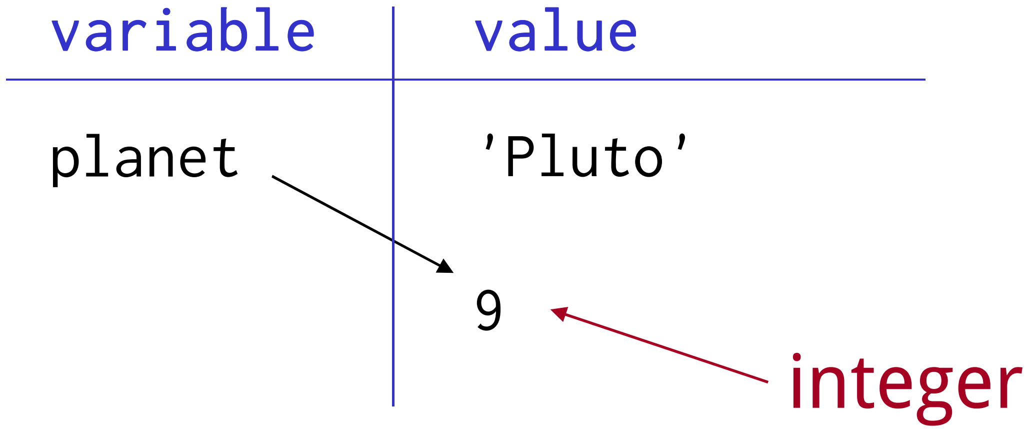
```
>>> planet = 'Pluto'
>>>
```



A variable is just a name

Does not have a type

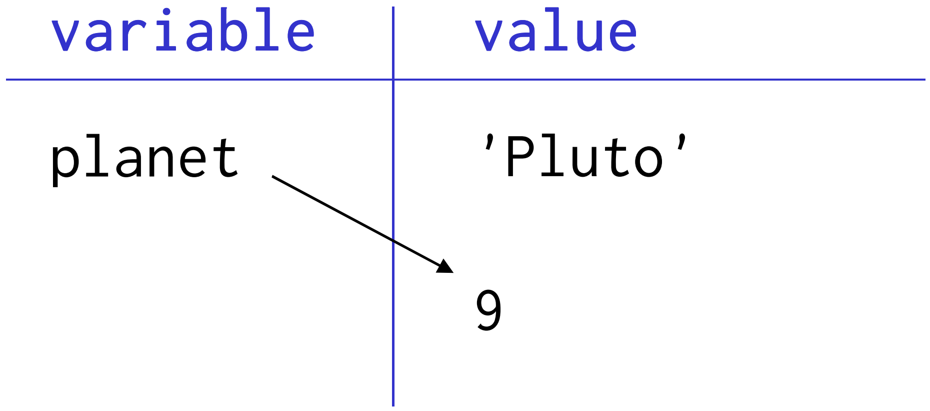
```
>>> planet = 'Pluto'
>>> planet = 9
>>>
```



A variable is just a name

Does not have a type

```
>>> planet = 'Pluto'
>>> planet = 9
>>>
```



variable | value

planet

'Pluto'

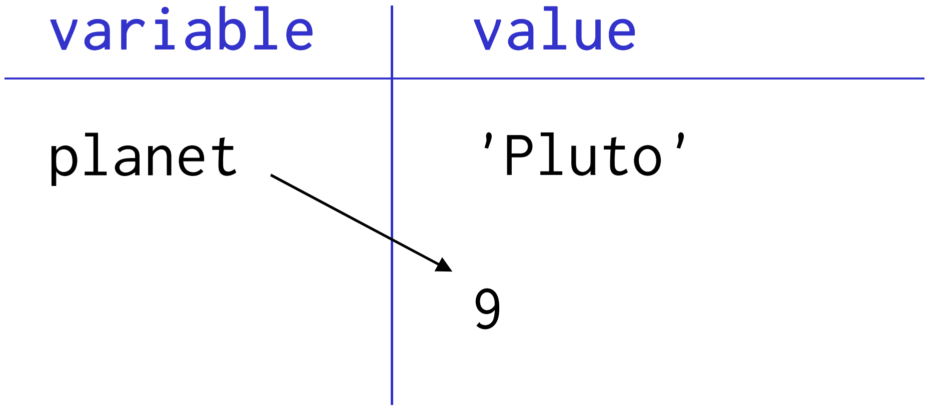
9

Values are *garbage collected*

A variable is just a name

Does not have a type

```
>>> planet = 'Pluto'
>>> planet = 9
>>>
```



variable | value

planet

'Pluto'

9

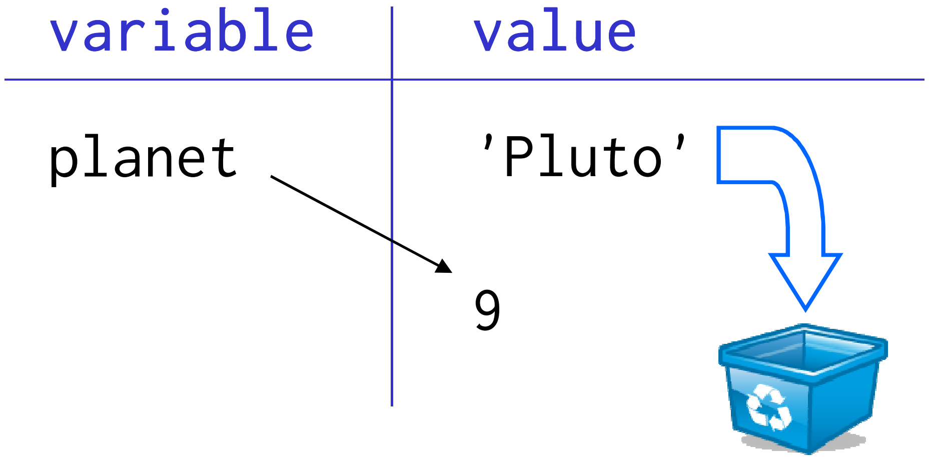
Values are *garbage collected*

If nothing refers to data any longer, it can be recycled

A variable is just a name

Does not have a type

```
>>> planet = 'Pluto'
>>> planet = 9
>>>
```



variable	value
planet	'Pluto' 9

Values are *garbage collected*

If nothing refers to data any longer, it can be recycled

Must assign value to variable before using it

Must assign value to variable before using it

```
>>> planet = 'Sedna'
>>>
```

Must assign value to variable before using it

```
>>> planet = 'Sedna'
>>> print plant        # note the deliberate misspelling
```

Must assign value to variable before using it

```
>>> planet = 'Sedna'
>>> print plant          # note the deliberate misspelling
Traceback (most recent call last):
    print plant
NameError: name 'plant' is not defined
>>>
```

Must assign value to variable before using it

```
>>> planet = 'Sedna'
>>> print plant          # note the deliberate misspelling
Traceback (most recent call last):
    print plant
NameError: name ’plant’ is not defined
>>>
```

Python does not assume default values for variables

Must assign value to variable before using it

```
>>> planet = 'Sedna'
>>> print plant        # note the deliberate misspelling
Traceback (most recent call last):
    print plant
NameError: name ’plant’ is not defined
>>>
```

Python does not assume default values for variables

Doing so can mask many errors

Must assign value to variable before using it

```
>>> planet = 'Sedna'
>>> print plant          # note the deliberate misspelling
Traceback (most recent call last):
    print plant
NameError: name 'plant' is not defined
>>>
```

Python does not assume default values for variables

Doing so can mask many errors

Anything from # to the end of the line is a comment

Values *do* have types

Values *do* have types

```
>>> string = "two"
>>> number = 3
>>> print string * number    # repeated concatenation
twotwotwo
>>>
```

Values *do* have types

```
>>> string = "two"
>>> number = 3
>>> print string * number    # repeated concatenation
twotwotwo
>>> print string + number
Traceback (most recent call last)
    number + string
TypeError: cannot concatenate ’str’ and ’int’ objects
>>>
```

Values *do* have types

```
>>> string = "two"
>>> number = 3
>>> print string * number    # repeated concatenation
twotwotwo
>>> print string + number
Traceback (most recent call last)
    number + string
TypeError: cannot concatenate ’str’ and ’int’ objects
>>>
```

Would probably be safe here to produce `'two3'`

Values *do* have types

```
>>> string = ″two”
>>> number = 3
>>> print string * number    # repeated concatenation
twotwotwo
>>> print string + number
Traceback (most recent call last)
    number + string
TypeError: cannot concatenate ’str’ and ’int’ objects
>>>
```

Would probably be safe here to produce `'two3'`

But then what should `'2'+'3'` be?

Values *do* have types

```
>>> string = "two"
>>> number = 3
>>> print string * number    # repeated concatenation
twotwotwo
>>> print string + number
Traceback (most recent call last)
    number + string
TypeError: cannot concatenate 'str' and 'int' objects
>>>
```

Would probably be safe here to produce `'two3'`

But then what should `'2'+'3'` be?

Doing too much is as bad as doing too little...

Use functions to convert between types

Use functions to convert between types

```
>>> print int('2') + 3
5
>>>
```

Use functions to convert between types

```
>>> print int(’2’) + 3
5
>>> print 2 + str(3)
23
>>>
```

Numbers

Numbers

14	32-bit integer (on most machines)

Numbers

`14`	32-bit integer (on most machines)
`14.0`	64-bit float (ditto)

Numbers

`14`	32-bit integer (on most machines)
`14.0`	64-bit float (ditto)
`1+4j`	complex number (two 64-bit floats)

Numbers

`14`	32-bit integer (on most machines)
`14.0`	64-bit float (ditto)
`1+4j`	complex number (two 64-bit floats)
`x.real, x.imag`	real and imaginary parts of complex number

Arithmetic

Arithmetic

Addition	+	35 + 22	57

Arithmetic

Addition	+	`35 + 22`	`57`
		`'Py' + 'thon'`	`'Python'`

Arithmetic

Addition	`+`	`35 + 22`	`57`
		`'Py' + 'thon'`	`'Python'`
Subtraction	`-`	`35 - 22`	`13`

Arithmetic

Addition	`+`	`35 + 22`	`57`
		`'Py' + 'thon'`	`'Python'`
Subtraction	`-`	`35 - 22`	`13`
Multiplication	`*`	`3 * 2`	`6`

Arithmetic

Addition	+	35 + 22	57
		'Py' + 'thon'	'Python'
Subtraction	-	35 - 22	13
Multiplication	*	3 * 2	6
		'Py' * 2	'PyPy'

Arithmetic

Addition	+	35 + 22	57
		'Py' + 'thon'	'Python'
Subtraction	-	35 - 22	13
Multiplication	*	3 * 2	6
		'Py' * 2	'PyPy'
Division	/	3.0 / 2	1.5

Arithmetic

Addition	+	35 + 22	57
		'Py' + 'thon'	'Python'
Subtraction	-	35 - 22	13
Multiplication	*	3 * 2	6
		'Py' * 2	'PyPy'
Division	/	3.0 / 2	1.5
		3 / 2	1

Arithmetic

Addition	+	35 + 22	57
		'Py' + 'thon'	'Python'
Subtraction	-	35 - 22	13
Multiplication	*	3 * 2	6
		'Py' * 2	'PyPy'
Division	/	3.0 / 2	1.5
		3 / 2	1
Exponentiation	**	2 ** 0.5	1.41421356...

Arithmetic

Addition	+	`35 + 22`	`57`
		`'Py' + 'thon'`	`'Python'`
Subtraction	-	`35 - 22`	`13`
Multiplication	*	`3 * 2`	`6`
		`'Py' * 2`	`'PyPy'`
Division	/	`3.0 / 2`	`1.5`
		`3 / 2`	`1`
Exponentiation	**	`2 ** 0.5`	`1.41421356...`
Remainder	%	`13 % 5`	`3`

Prefer *in-place* forms of binary operators

Prefer *in-place* forms of binary operators

```
>>> years = 500
>>>
```

Prefer *in-place* forms of binary operators

```
>>> years = 500
>>> years += 1
>>>
```

Prefer *in-place* forms of binary operators

```
>>> years = 500
>>> years += 1
>>>
```

The same as `years = years + 1`

Prefer *in-place* forms of binary operators

```
>>> years = 500
>>> years += 1
>>> print years
501
>>>
```

Prefer *in-place* forms of binary operators

```
>>> years = 500
>>> years += 1
>>> print years
501
>>> years %= 10
>>>
```

Prefer *in-place* forms of binary operators

```
>>> years = 500
>>> years += 1
>>> print years
501
>>> years %= 10
>>>
```

The same as `years = years % 10`

Prefer *in-place* forms of binary operators

```
>>> years = 500
>>> years += 1
>>> print years
501
>>> years %= 10
>>> print years
5
>>>
```

Comparisons

Comparisons

```
3 < 5                   True
```

Comparisons

3 < 5	True
3 != 5	True

Comparisons

`3 < 5`	`True`
`3 != 5`	`True`
`3 == 5`	`False`

Comparisons

`3 < 5`	`True`
`3 != 5`	`True`
`3 == 5`	`False`

←

Single = is assignment

Double == is equality

Comparisons

`3 < 5`	`True`
`3 != 5`	`True`
`3 == 5`	`False`
`3 >= 5`	`False`

Comparisons

`3 < 5`	`True`
`3 != 5`	`True`
`3 == 5`	`False`
`3 >= 5`	`False`
`1 < 3 < 5`	`True`

Comparisons

`3 < 5`	`True`
`3 != 5`	`True`
`3 == 5`	`False`
`3 >= 5`	`False`
`1 < 3 < 5`	`True`
`1 < 5 > 3`	`True`

← But please don't do this

Comparisons

`3 < 5`	`True`
`3 != 5`	`True`
`3 == 5`	`False`
`3 >= 5`	`False`
`1 < 3 < 5`	`True`
`1 < 5 > 3`	`True`
`3+2j < 5`	*error*

software carpentry

created by

Greg Wilson

October 2010