Practical Computing for Scientists

Armin Sobhani

CSCI 2000U

UOIT – Fall 2015

UNIVERSITY OF ONTARIO
INSTITUTE OF TECHNOLOGY

Python
Introduction

UNIVERSITY OF ONTARIO
INSTITUTE OF TECHNOLOGY

Time to solution is determined by:

Time to solution is determined by:

how long it takes to write a program

Time to solution is determined by:

how long it takes to write a program

how long it takes that program to run

Time to solution is determined by:

how long it takes to write a program

human time

how long it takes that program to run

Time to solution is determined by:

how long it takes to write a program	how long it takes that program to run
human time	machine time

Time to solution is determined by:

how long it takes to write a program

how long it takes that program to run

human time

machine time

Every language makes a tradeoff

between these

Time to solution is determined by:

how long it takes to write a program

how long it takes that program to run

human time

machine time

Every language makes a tradeoff

between these

Python

Time to solution is determined by:

how long it takes to write a program

human time

how long it takes that program to run

machine time

Every language makes a tradeoff

between these

Python

MATLAB

Time to solution is determined by:

how long it takes to write a program

how long it takes that program to run

human time

machine time

Every language makes a tradeoff

between these

Python

MATLAB

Fortran

Time to solution is determined by:

how long it takes to write a program	how long it takes that program to run
human time	machine time

Every language makes a tradeoff

between these

Python	Fortran
MATLAB	C/C++

Time to solution is determined by:

how long it takes to write a program

how long it takes that program to run

human time

machine time

Every language makes a tradeoff

between these

Python

MATLAB

Java

Fortran

C/C++

Time to solution is determined by:

how long it takes to write a program	how long it takes that program to run
human time	machine time

Every language makes a tradeoff

between these

Python	Java	Fortran
MATLAB	C#	C/C++

Time to solution is determined by:

how long it takes to write a program

how long it takes that program to run

human time

machine time

Every language makes a tradeoff

between these

Python

Java

Fortran

MATLAB

C#

C/C++

Why  python™ ?

Why python™ ?

- easier to pick up than alternatives

Why python™?

- easier to pick up than alternatives
- free

Why python™?

- easier to pick up than alternatives
- free
- cross-platform

Why python™?

- easier to pick up than alternatives
- free
- cross-platform
- widely used

Why python™?

- easier to pick up than alternatives
- free
- cross-platform
- widely used
- well documented

Why python™?

- easier to pick up than alternatives
- free
- cross-platform
- widely used
- well documented
- well supported

Setting Up Python

Setting Up Python

SciPy Stack

Setting Up Python

SciPy Stack

- NumPy

Setting Up Python

SciPy Stack

- NumPy
- SciPy Library

Setting Up Python

SciPy Stack

- NumPy
- SciPy Library
- Matplotlib

Setting Up Python

SciPy Stack

- NumPy
- SciPy Library
- Matplotlib
- IPython

Setting Up Python

SciPy Stack

- NumPy
- SciPy Library
- Matplotlib
- IPython
- Sympy

Setting Up Python

SciPy Stack

- NumPy
- SciPy Library
- Matplotlib
- IPython
- Sympy
- pandas

Setting Up Python

Scientific Python Distributions

Setting Up Python

Scientific Python Distributions

- Anaconda
- Enthought Canopy
- Python(x,y)
- WinPython
- Pyzo

Setting Up Python

Scientific Python Distributions

- Anaconda
- Enthought Canopy
- Python(x,y)
- WinPython
- Pyzo

Anaconda

Installing Anaconda

- Download Anaconda:
  - Blackboard > Course Content > Week 5 (Oct. 13–16) > Wednesday Oct. 14 > Download Anaconda

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Uninstalling Anaconda

```
$ _
```

Uninstalling Anaconda

```
$ rm -rf ~/anaconda3
$ _
```

Python 3 vs. 2

Python 3 vs. 2

`print` is a function rather than a statement

```
>>> _
```

Python 3 vs. 2

`print` is a function rather than a statement

```
>>> print(3.14159)
3.14159
>>> _
```

Python 3 vs. 2

`print` is a function rather than a statement

```
>>> print(3.14159)
3.14159
>>> print 3.14159
  File "<stdin>", line 1
    print 3.14159
                ^
SyntaxError: Missing parentheses in call to 'print'
>>> _
```

Changing Python Version in Anaconda

```
$ _
```

Changing Python Version in Anaconda

```
$ conda create -n python2 python=2.7 anaconda
…
$ _
```

Changing Python Version in Anaconda

```
$ conda create -n python2 python=2.7 anaconda
...
$ _
```

Changing Python Version in Anaconda

```
$ conda create -n python2 python=2.7 anaconda
…
$ _
```

Changing Python Version in Anaconda

```
$ conda create -n python2 python=2.7 anaconda
…
$ source activate python2
$ _
```

Changing Python Version in Anaconda

```
$ conda create -n python2 python=2.7 anaconda
…
$ source activate python2
$ conda env list
# conda environments:
#
python2                  *  /home/asobhani/anaconda3/envs/python2
root                        /home/asobhani/anaconda3

$ _
```

Changing Python Version in Anaconda

```
$ conda create -n python2 python=2.7 anaconda
…
$ source activate python2
$ conda env list
# conda environments:
#
python2                  *  /home/asobhani/anaconda3/envs/python2
root                        /home/asobhani/anaconda3

$ _
```

omit `source` in Windows

Changing Python Version in Anaconda

```
$ conda create -n python2 python=2.7 anaconda
…
$ source activate python2
$ conda env list
# conda environments:
#
python2                  *  /home/asobhani/anaconda3/envs/python2
root                        /home/asobhani/anaconda3

$ source deactivate
# conda environments:
#
python2                     /home/asobhani/anaconda3/envs/python2
root                     *  /home/asobhani/anaconda3
$ _
```

Changing Python Version in Anaconda

```
$ conda create -n python2 python=2.7 anaconda
…
$ source activate python2
$ conda env list
# conda environments:
#
python2                  *  /home/asobhani/anaconda3/envs/python2
root                        /home/asobhani/anaconda3

$ source deactivate
# conda environments:
#
python2                     /home/asobhani/anaconda3/envs/python2
root                     *  /home/asobhani/anaconda3
$ _
```

omit `source` in Windows

Checkpoint 11

- Please complete the *How does science model reality Results*:
  - Blackboard > Course Content > Week 5 (Oct. 13–16) > Wednesday Oct. 14 > Checkpoint 11

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

Python
Basics

UNIVERSITY OF ONTARIO INSTITUTE OF TECHNOLOGY

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

```
$ ipython
In [1]: _
```

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
$
```

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

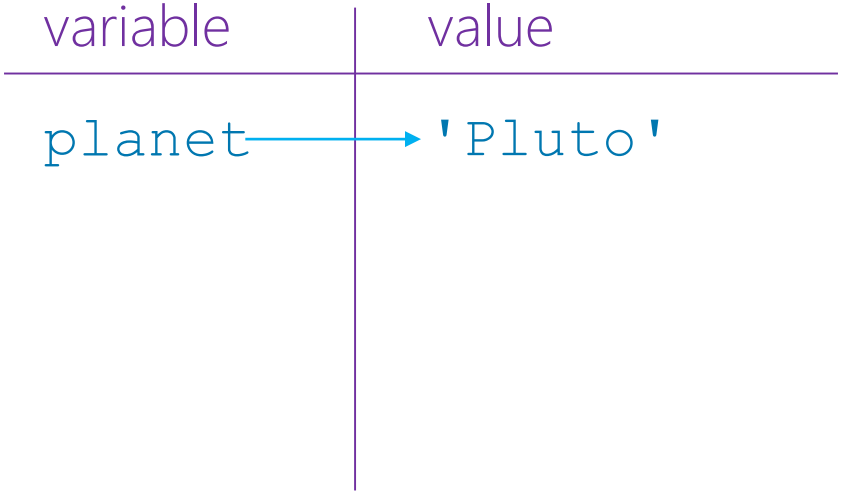
Created by use: no declaration necessary

```
>>> planet = 'Pluto'
>>> print(planet)
Pluto
>>>
```

Variables are names for values

Created by use: no declaration necessary

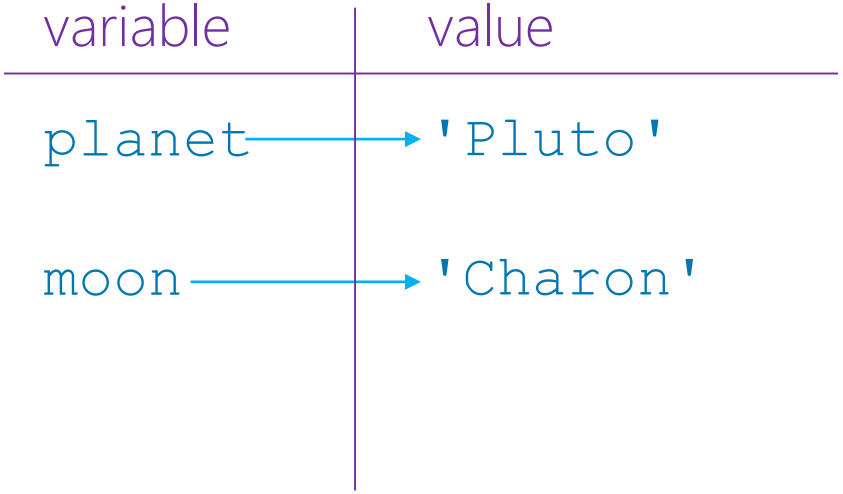
```
>>> planet = 'Pluto'
>>> print(planet)
Pluto
>>>
```



Variables are names for values

Created by use: no declaration necessary

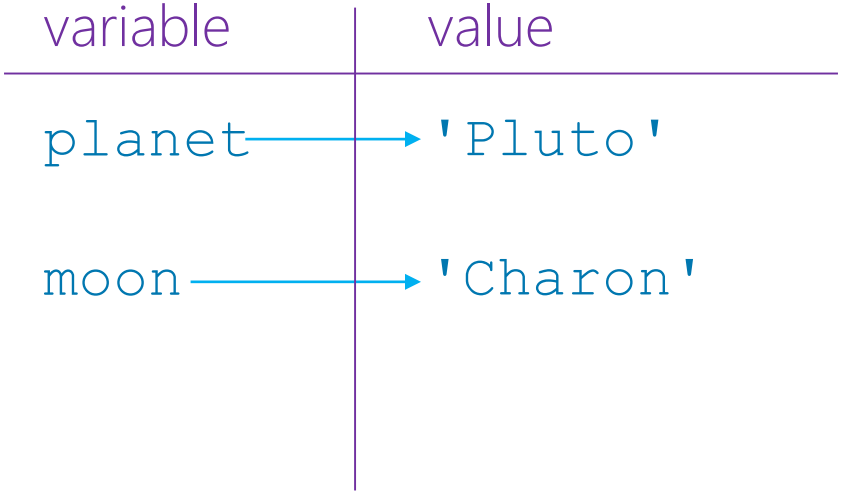
```
>>> planet = 'Pluto'
>>> print(planet)
Pluto
>>> moon = 'Charon'
>>>
```



Variables are names for values

Created by use: no declaration necessary

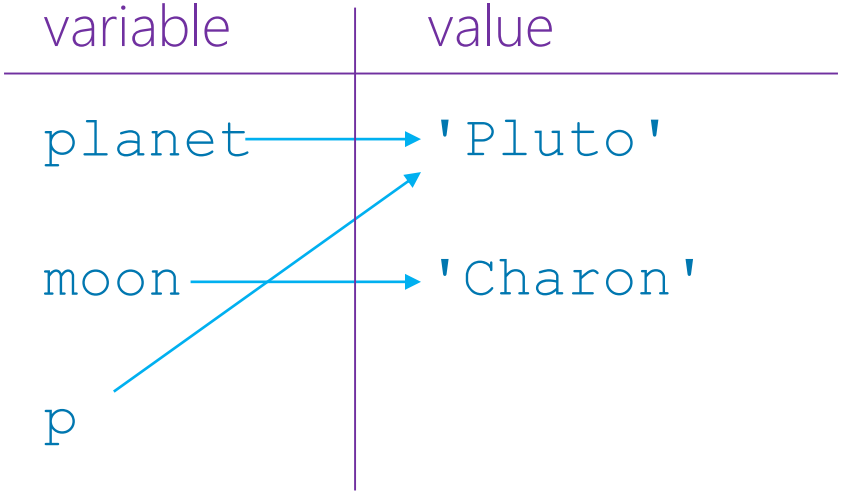
```
>>> planet = 'Pluto'
>>> print(planet)
Pluto
>>> moon = 'Charon'
>>> p = planet
>>>
```



Variables are names for values

Created by use: no declaration necessary

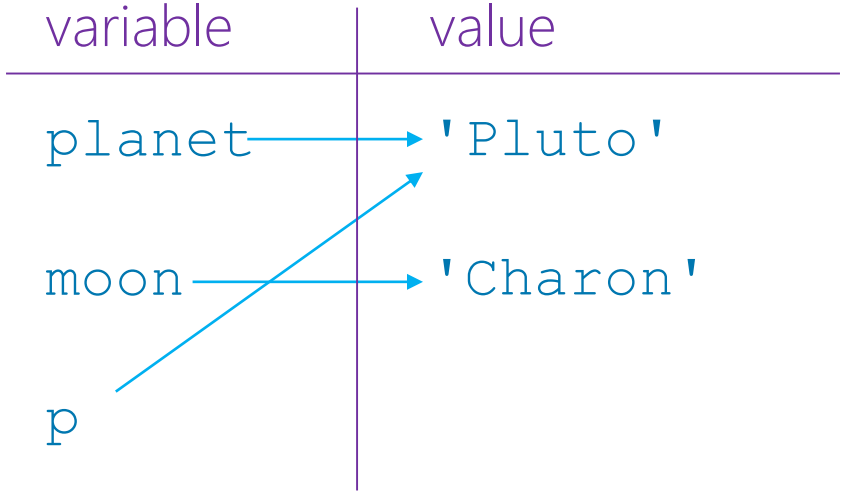
```
>>> planet = 'Pluto'
>>> print(planet)
Pluto
>>> moon = 'Charon'
>>> p = planet
>>>
```



Variables are names for values

Created by use: no declaration necessary

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



A variable is just a name

A variable is just a name

Does not have a type

A variable is just a name

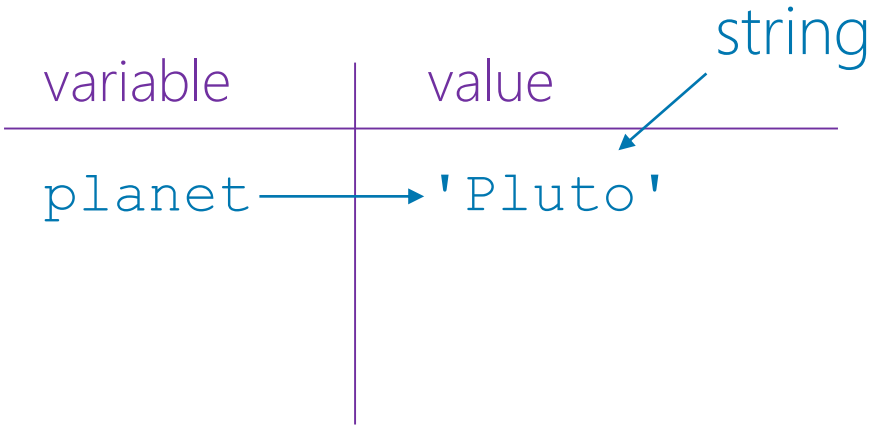
Does not have a type

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

A variable is just a name

Does not have a type

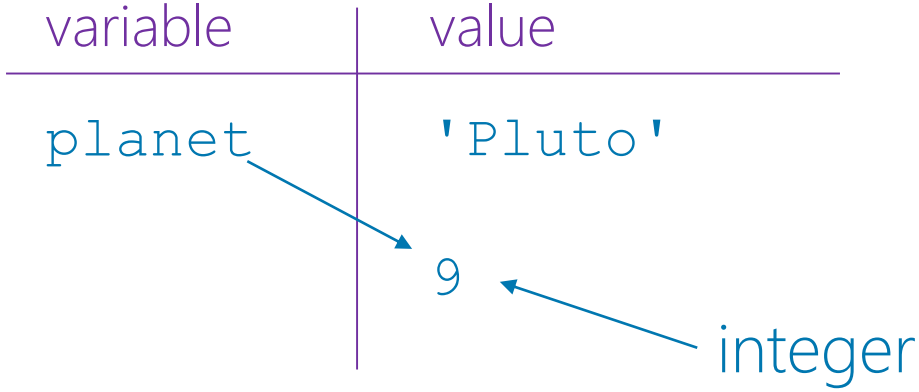
```
>>> planet = 'Pluto'
>>>
```



A variable is just a name

Does not have a type

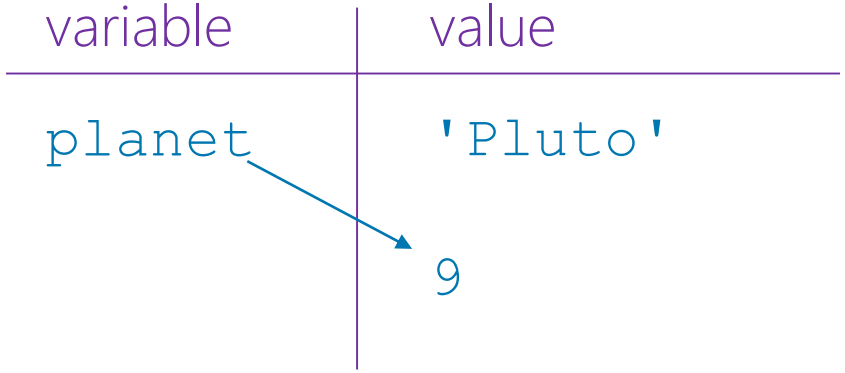
```
>>> planet = 'Pluto'
>>> planet = 9
>>>
```



A variable is just a name

Does not have a type

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

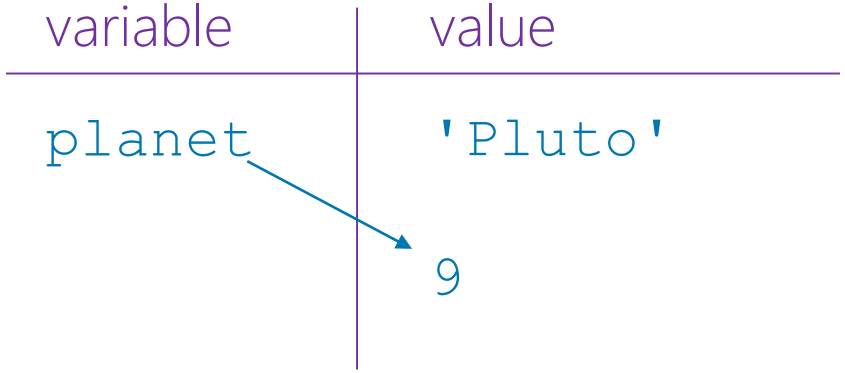


Values are *garbage collected*

A variable is just a name

Does not have a type

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



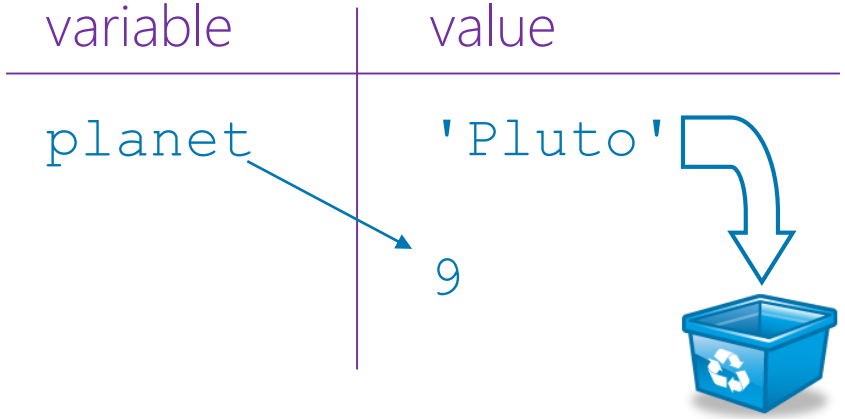
Values are *garbage collected*

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
>>>
```



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
>>>
```