Introduction to Programming languages and to Python

Rémi Marchal
Inorganic Theoretical Chemistry group, ISCR, Rennes
remi.marchal@univ-rennes1.fr
February, 05^{th} 2018

Introduction to the programming Python language

Outline

Outline

Definition

Wikipedia says :

Definition

Wikipedia says :

- A programming language is a formal language that can be uses to produce various kind of output.

Definition

Wikipedia says :

- A programming language is a formal language that can be uses to produce various kind of output.
- Programming languages generally consist of instructions for a computer.

Definition

Wikipedia says :

- A programming language is a formal language that can be uses to produce various kind of output.
- Programming languages generally consist of instructions for a computer.
- Programming languages can be used to create programs that implement specific algorithms.

Outline

A bit of history

The different generation of Programming Languages

Since the discovery of computers, several generation of programming languages were created.

- The first-generation programming languages
- The second-generation programming languages
- The third-generation programming languages

Outline

The first-generation programming languages

The frist programming languages generation

The first-generation programming languages

The frist programming languages generation

- The earliest computers were often programmed without the help of a programming language, by writing programs in absolute machine language.

The first-generation programming languages

The frist programming languages generation

- The earliest computers were often programmed without the help of a programming language, by writing programs in absolute machine language.
- The programs, in decimal or binary form, were read in from punched cards or magnetic tape or toggled in on switches on the front panel of the computer.

The first-generation programming languages

The frist programming languages generation

- The earliest computers were often programmed without the help of a programming language, by writing programs in absolute machine language.
- The programs, in decimal or binary form, were read in from punched cards or magnetic tape or toggled in on switches on the front panel of the computer.
- Absolute machine languages were later termed first-generation programming languages (1GL).

Outline

The second-generation programming languages

The second programming languages generation

The second-generation programming languages

The second programming languages generation

- This is the so-called assembly languages, which were still closely tied to the instruction set architecture of the specific computer.

The second-generation programming languages

The second programming languages generation

- This is the so-called assembly languages, which were still closely tied to the instruction set architecture of the specific computer.
- These served to make the program much more human-readable and relieved the programmer of tedious and error-prone address calculations.

Outline

The third-generation programming languages

The third programming languages generation

The third-generation programming languages

The third programming languages generation

- They are known as High-level programming languages

Outline

Outline

Compiled languages

What is it ?

Compiled languages

What is it ?

- They are written in an human-readable language

Compiled languages

What is it ?

- They are written in an human-readable language
- They need need to be compiled by a compiler

Compiled languages

What is it ?

- They are written in an human-readable language
- They need need to be compiled by a compiler

A compiler : What is it ?

A compiler is a software that generates machine code from source code. The machine code contains the instruction to the computer.

Compiled languages

What is it ?

- They are written in an human-readable language
- They need need to be compiled by a compiler

A compiler : What is it ?

A compiler is a software that generates machine code from source code. The machine code contains the instruction to the computer.

Widely used languages

Fortran

C

C++

Outline

Interpreted languages

What is it ?

Interpreted languages

What is it ?

- Unlike compiled ones, no need to be compiled

Interpreted languages

What is it ?

- Unlike compiled ones, no need to be compiled
- Execute each command each one after each other

Interpreted languages

What is it ?

- Unlike compiled ones, no need to be compiled
- Execute each command each one after each other

Widely used languages

Fortran C C++

Outline

Compiled languages VS interpreted languages

Reasons for using compiled languages

Reasons for using interpreted languages

Compiled languages VS interpreted languages

Reasons for using compiled languages

- They are compiled so faster than interpreted

Reasons for using interpreted languages

Compiled languages VS interpreted languages

Reasons for using compiled languages

- They are compiled so faster than interpreted
- Highly efficient external libraries for FFT, matrix diagonalization, ...

Reasons for using interpreted languages

Compiled languages VS interpreted languages

Reasons for using compiled languages

- They are compiled so faster than interpreted
- Highly efficient external libraries for FFT, matrix diagonalization, ...

Reasons for using interpreted languages

- Interpreted so sometime easier to build.

Compiled languages VS interpreted languages

Reasons for using compiled languages

- They are compiled so faster than interpreted
- Highly efficient external libraries for FFT, matrix diagonalization, ...

Reasons for using interpreted languages

- Interpreted so sometime easier to build.
- In python, extremely good ploting libraries

Introduction to the programming Python language

Outline

Introduction

History
What is python ?
Why learning and using Python ?
Some basic rules

First hints

Importing libraries
Print a variable
Storing a data sequence
Storing a data sequence
Unix in Python
IOS
Conditional
loops
Conditional loops
Conditional loops

Outline

A bit of history

1991 : Guido Van Rossum Start to develop the Python programming language.
2001 : Creation of the Python Software Foundation, a non-profit organization aiming at foster development of the Python community.
2009 : Creation of Python3.

Outline

Introduction

History

What is python ?

Why learning and using Python ?

Some basic rules

First hints

Importing libraries

Print a variable

Storing a data sequence

Storing a data sequence

Unix in Python

IOS

Conditional

loops

Conditional loops

Conditional loops

What is python ?

Quesaco ?

What is python ?

Quesaco ?

- Python is an Object-oriented language. Is allows you to create and manipulate easily some objects

What is python ?

Quesaco ?

- Python is an Object-oriented language. Is allows you to create and manipulate easily some objects
- It is an interpreted language, so no compilation needed

Outline

Introduction
History
What is python ?
Why learning and using Python ?
Some basic rules
First hints
Importing libraries
Print a variable
Storing a data sequence
Storing a data sequence
Unix in Python
IOS
Conditional
loops
Conditional loops
Conditional loops

Why Python ?

Advantages

There is several advantages in using Python

Why Python ?

Advantages

There is several advantages in using Python

- Strong developer community so a lot of libraries ready to be used

Why Python ?

Advantages

There is several advantages in using Python

- Strong developer community so a lot of libraries ready to be used
- A large user community so a lot of forum and tutorial available on the web

Why Python ?

Advantages

There is several advantages in using Python

- Strong developer community so a lot of libraries ready to be used
- A large user community so a lot of forum and tutorial available on the web
- Several graphical libraries available

2 ways to write program in python

Through the interpreter

```
pr075014:examples rmarchal$ python
Python 2.7.14 (default, Sep 22 2017, 00:06:07)
[GCC 4.2.1 Compatible Apple LLVM 8.1.0 (clang-802.0.42)
] on darwin
Type "help", "copyright", "credits" or "license" for mo
re information.
>>> a=3.5
>>> b=2.0
>>> c=a*b
>>> print c
7.0
>>> quit()
pr075014:examples rmarchal$
```

2 ways to write program in python

Through a script

```
pr075014:examples rmarchal$ cat test.py
a=3.5
b=2.0
c=a*b
print c
pr075014:examples rmarchal$ python test.py
7.0
pr075014:examples rmarchal$
```

2 ways to write program in python

Which way to choose ?

2 ways to write program in python

Which way to choose ?

It depends what you want to do.

2 ways to write program in python

Which way to choose ?

It depends what you want to do.

- If you just want to do a quite small calculation, you should use the interpreter

2 ways to write program in python

Which way to choose ?

It depends what you want to do.

- If you just want to do a quite small calculation, you should use the interpreter
- If you want to write a quite long program, use the script

2 ways to write program in python

Which way to choose ?

It depends what you want to do.

- If you just want to do a quite small calculation, you should use the interpreter
- If you want to write a quite long program, use the script
- If you want to write a program and execute it several times, use the script

Outline

Basics rules

What to avoid ?

When writing a program in python, there is some basics rules that you should follow :

- Take care about the indentation
- Avoid infinite loops
- THINK BEFORE CODING

Outline

Outline

Import Libraries

Import Libraries

Why ?

Python developers have developed several different kind of libraries for various purposes such as IOs (writing and reading files), mathematical operation (exponential, logarithm,...), matrix operations (diagonalisation, inversino, transpose, ...) ...

$\hookrightarrow$ It can be useful to call them

Import Libraries

Why ?

Python developers have developed several different kind of libraries for various purposes such as IOs (writing and reading files), mathematical operation (exponential, logarithm,...), matrix operations (diagonalisation, inversino, transpose, ...) ...
↪ It can be useful to call them

How to call them ?

Either from the interpreter or the scripts, libraries are called using the following line (in the example, we will import the ≪ math ≫ library)

- import math (this is for the whole library)
- from math import exp (for importing only the ≪ sqrt ≫ routine of the ≪ math ≫ library)

Outline

Printing a variable

Printing a variable

Why ?

At the end of the program, you want the result to be displayed.

Printing a variable

Why ?

At the end of the program, you want the result to be displayed.

How ?

This can be done by using the print python function.

Printing a variable

Why ?

At the end of the program, you want the result to be displayed.

How ?

This can be done by using the print python function.

An example

```
pr075014:examples rmarchal$ python
Python 2.7.14 (default, Sep 22 2017, 00:06:07)
[GCC 4.2.1 Compatible Apple LLVM 8.1.0 (clang-802.0.42)
] on darwin
Type "help", "copyright", "credits" or "license" for mo
re information.
>>> a=3.5
>>> b=2.0
>>> c=a*b
>>> print c
7.0
>>> quit()
pr075014:examples rmarchal$
```

Outline

Storing data in lists

Storing data in lists

What is a list

A list is a sequence of variable.
Ex : $[2.4, 3.5, 6.0]$

Storing data in lists

What is a list

A list is a sequence of variable.
Ex : $[2.4, 3.5, 6.0]$

How to create a list

This is done like this :
listname=[]

Storing data in lists

What is a list

A list is a sequence of variable.
Ex : $[2.4, 3.5, 6.0]$

How to create a list

This is done like this :
listname=[]

How to store data in a created list

To do it, you should « append » an existing list. For example :
listname.append(3.0)

Outline

Introduction
History
What is python ?
Why learning and using Python ?
Some basic rules
First hints
Importing libraries
Print a variable
Storing a data sequence
Storing a data sequence
Unix in Python
IOS
Conditional
loops
Conditional loops
Conditional loops

Storing data in lists

An example

```
Python 2.7.14 (default, Sep 22 2017, 00:06:07)
[GCC 4.2.1 Compatible Apple LLVM 8.1.0 (clang-802.0.42)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> list=[]
>>> print list
[]
>>> list.append(3.0)
>>> print list
[3.0]
>>> list.append([2.3,2.7])
>>> print list
[3.0, [2.3, 2.7]]
>>> quit()
pr075014:examples rmarchal$
```

Outline

Redirecting Unix command into a Python variable

Redirecting Unix command into a Python variable

Why ?

It can be useful for storing your actual path, redirecting some ≪ grep ≫ command in python lists, reading files (see later), ...

Redirecting Unix command into a Python variable

Why ?

It can be useful for storing your actual path, redirecting some ≪ grep ≫ command in python lists, reading files (see later), ...

How ?

This is possible thanks to the ≪ os ≫ python library. For this, you should use the ≪ popen ≫ function of this library

Redirecting Unix command into a Python variable

Redirecting Unix command into a Python variable

An example

```
pr075014:examples rmarchal$ pwd
/Users/rmarchal/Documents/cours-formation-chimie-theo/2018/Python/cours/examples
pr075014:examples rmarchal$ python
Python 2.7.14 (default, Sep 22 2017, 00:06:07)
[GCC 4.2.1 Compatible Apple LLVM 8.1.0 (clang-802.0.42)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import os
>>> f=os.popen("pwd")
>>> path=f.read()
>>> print path
/Users/rmarchal/Documents/cours-formation-chimie-theo/2018/Python/cours/examples

>>> quit()
pr075014:examples rmarchal$
```

Outline

Writing a file

How ?

For this, you should first import the « csv » library.
Than, you can define a function like this :
« ecriture=open('filename','wb') » where filename is the name of the file.
Thus you can write in this file using this :
« ecriture.write('what you want to write') »
If you want to go to the new line, this can be done like this : « \n »

Writing a file

An example

```
Python 2.7.14 (default, Sep 22 2017, 00:06:07)
[GCC 4.2.1 Compatible Apple LLVM 8.1.0 (clang-802.0.42)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import csv
>>> ecriture=open('new_file','wb')
>>> ecriture.write('test1 \n')
>>> ecriture.write('test2')
>>> ecriture.write(' test3')
>>> ecriture.write('\n')
>>> ecriture.close()
>>> quit()
pr075014:examples rmarchal$ cat new_file
test1
test2 test3
pr075014:examples rmarchal$
```

Reading a file

Reading a file

How ?

There is several methods for reading a file (through the ≪ csv ≫ library, by the open command, ...).
I will only present mine, based on the ≪ cat ≫ Unix command.

Reading a file

How ?

There is several methods for reading a file (through the « csv » library, by the open command, ...).
I will only present mine, based on the « cat » Unix command.

A remind about the « cat » Unix command

The « cat » Unix command allows you to display the content of a file on the screen.
When redirecting it to a Python variable, it allows you to read a file.

Reading a file

How ?

There is several methods for reading a file (through the « csv » library, by the open command, ...).
I will only present mine, based on the « cat » Unix command.

A remind about the « cat » Unix command

The « cat » Unix command allows you to display the content of a file on the screen.
When redirecting it to a Python variable, it allows you to read a file.

Splitting the lines

To store the file in a list with each element as a line of the file, you should also use the « readlines() » of the « popen » function.

Reading a file

Reading a file

An example

```
pr075014:examples rmarchal$ cat new_file
test1 kghds jgsqdjhgkdsq
test2 test3 klsdlj
ljdlld  djkhd egehdejef efkdskhfds dskhdfskhdfs
pr075014:examples rmarchal$ python
Python 2.7.14 (default, Sep 22 2017, 00:06:07)
[GCC 4.2.1 Compatible Apple LLVM 8.1.0 (clang-802.0.42)] on dar
win
Type "help", "copyright", "credits" or "license" for more infor
mation.
>>> import os
>>> f=os.popen("cat new_file").readlines()
>>> print f
['test1 kghds jgsqdjhgkdsq \n', 'test2 test3 klsdlj\n', 'ljdlld
  djkhd egehdejef efkdskhfds dskhdfskhdfs \n']
```

Reading a file

Reading a file

Splitting the words

You can do it by using the Python function
≪ str.split ≫

Reading a file

Splitting the words

You can do it by using the Python function
≪ str.split ≫

An example

```
pr075014:examples rmarchal$ cat new_file
test1 kghds jgsqdjhgkdsq
test2 test3 klsdlj
ljdlld  djkhd egehdejef efkdskhfds dskhdfskhdfs
pr075014:examples rmarchal$ python
Python 2.7.14 (default, Sep 22 2017, 00:06:07)
[GCC 4.2.1 Compatible Apple LLVM 8.1.0 (clang-802.0.42)] on dar
win
Type "help", "copyright", "credits" or "license" for more infor
mation.
>>> import os
>>> f=os.popen("cat new_file").readlines()
>>> print f
['test1 kghds jgsqdjhgkdsq \n', 'test2 test3 klsdlj\n', 'ljdlld
  djkhd egehdejef efkdskhfds dskhdfskhdfs \n']
>>>
>>> print str.split(f[0])[1]
kghds
```

Outline

conditional expressions

conditional expressions

Aim

Do a part of the program only if a condition is fulfilled.

conditional expressions

Aim

Do a part of the program only if a condition is fulfilled.

How ?

Using the ≪ if ≫ function of python. There is also the ≪ elif ≫ and ≪ else ≫ functions associated.

conditional expressions

An example

```
pr075014:examples rmarchal$ python
Python 2.7.14 (default, Sep 22 2017, 00:06:07)
[GCC 4.2.1 Compatible Apple LLVM 8.1.0 (clang-802.0.42)] on dar
win
Type "help", "copyright", "credits" or "license" for more infor
mation.
>>> a=9.2
>>> if a<10:
...     print "a is lower than 10"
... else:
...     print "a is higer than 10"
...
a is lower than 10
>>> quit()
```

Attention

Be careful with the indentation

Outline

Introduction
History
What is python ?
Why learning and using Python ?
Some basic rules
First hints
Importing libraries
Print a variable
Storing a data sequence
Storing a data sequence
Unix in Python
IOS
Conditional
loops
Conditional loops
Conditional loops

Loops

Loops

Aim

Repeat some operations.

Loops

Aim

Repeat some operations.

How ?

Using the ≪ for ≫ python statement.

Loops

Aim

Repeat some operations.

How ?

Using the « for » python statement.

Syntax

If I want a loop from the value 0 to 10 :
« for i in range(0,10) » If I want a loop from the value 0 to 10 but changing the value by 2 at each time :
« for i in range(0,10,2) »

Loops

An example

```
pr075014:examples rmarchal$ python
Python 2.7.14 (default, Sep 22 2017, 00:06:07)
[GCC 4.2.1 Compatible Apple LLVM 8.1.0 (clang-802.0.42)] on dar
win
Type "help", "copyright", "credits" or "license" for more infor
mation.
>>> for i in range(0,6):
...     print i
...
0
1
2
3
4
5
>>> for i in range(0,6,2):
...     print i
...
0
2
4
```

Outline

Conditional loops

Conditional loops

Aim

Make a loop that stops after a condition is fulfilled

Conditional loops

Aim

Make a loop that stops after a condition is fulfilled

How ?

Using the ≪ while ≫ python statement.

Outline

Conditional loops

Conditional loops

Example

```
pr075014:examples rmarchal$ python
Python 2.7.14 (default, Sep 22 2017, 00:06:07)
[GCC 4.2.1 Compatible Apple LLVM 8.1.0 (clang-802.0.42)] on dar
win
Type "help", "copyright", "credits" or "license" for more infor
mation.
>>> i=1
>>> while i<=10:
...     print i
...     i=i+1
...
1
2
3
4
5
6
7
8
9
10
>>> quit()
```

Advances issues in Python

Outline

Subroutines

What is a subroutine ?

A subroutine is a part of the code that could be considered as outside of the code (STRANGE ! ! !).
A subroutine take ≪ arguments ≫ and return ≪ outputs ≫.

Subroutines

What is a subroutine ?

A subroutine is a part of the code that could be considered as outside of the code (STRANGE ! ! !).
A subroutine take ≪ arguments ≫ and return ≪ outputs ≫.

A subroutine : Why ?

Most of the time, it is used when you aims at doing a particular series of operation several times.

Subroutines

How to create it and call it ?

```
from math import exp
def sub_name(entry):    # Creates a subroutine sub_name taking entry as input

        a = exp(entry)

        return(a)       # return the value of a

b=3.5
f=sub_name(b)           #attribute the output of the subroutine to a variable f
print f
```

Subroutines

Another reason to write subroutines

Make your program more readable and understandable. Indeed, when yoy are creating creating a program, try to do the following :

1. Write the most subroutines as you can (it helps spliting the operations and so help for debugging)
2. Write the smallest $\ll$ main $\gg$ part as possible.

Outline

Numpy

Why to learn Numpy ?

This is one of the most powerful python library for

import Numpy

Most of the time, people are importing Numpy like this :
≪ Import numpy as np ≫

How to create a 0 matrix or array ?

f=np.zeros(3) creates a 3 value long array set to 0.
f=np.zeros((3,3)) creates a 3*3 values set to 0.

Numpy

Some useful matrix operators

np.transpose for matrix transposition
np.dot for dot product
np.linalg.det for the determinant
np.linalg.inv for matrix inversion
np.linalg.solve for solving linear equations
np.linalg.eigvals for eigenvalues

Outline

Matplotlib

Matplotlib

Aim

Is a library that allows you to plot your data

Matplotlib

Aim

Is a library that allows you to plot your data

How to call the library ?

Most of the time, the library is called like this :
import matplotlib.pyplot as plt

Matplotlib example

1. creating the lists to be plotted

We will plot the results of y=2.x equation.

```
pr075014:fin rmarchal$ python
Python 2.7.14 (default, Sep 22 2017, 00:06:07)
[GCC 4.2.1 Compatible Apple LLVM 8.1.0 (clang-802.0.42)]
on darwin
Type "help", "copyright", "credits" or "license" for more
 information.
>>> import matplotlib.pyplot as plt
>>> X=[]
>>> Y=[]
>>> for i in range(0,10):
...     X.append(i)
...     Y.append(2*i)
...
>>>
```

Matplotlib example

2. scatter plot

We will plot the results of y=2.x equation as scatter.

```
>>> plt.scatter(X,Y)
<matplotlib.collections.PathCollection object at 0x1130ce
250>
>>> plt.show()
```

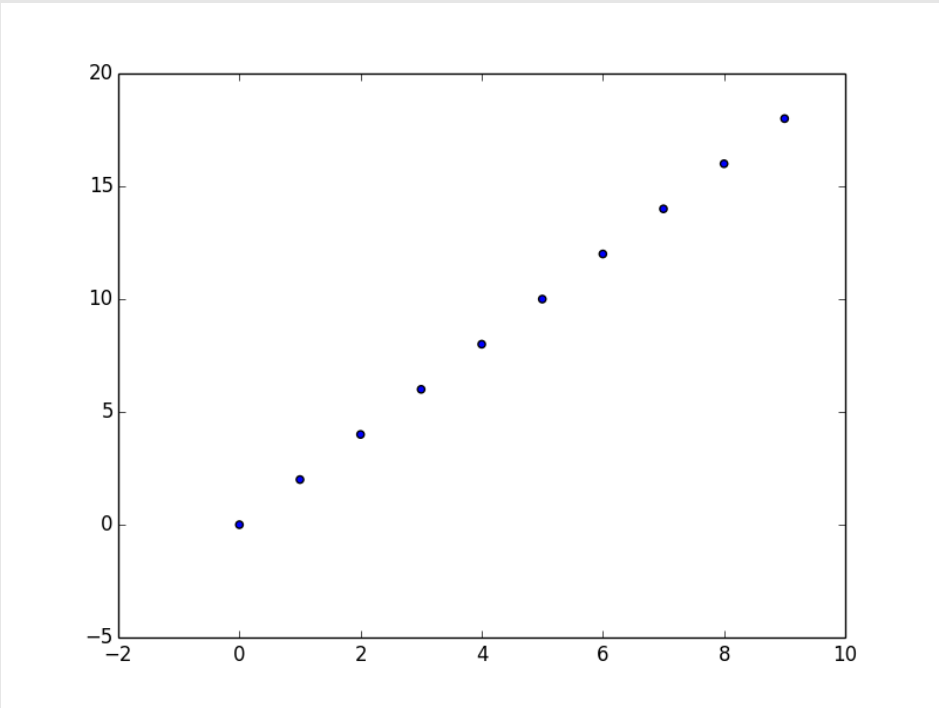
Matplotlib example

2. scatter plot

We will plot the results of y=2.x equation as scatter.

```
>>> plt.scatter(X,Y)
<matplotlib.collections.PathCollection object at 0x1130ce
250>
>>> plt.show()
```

And the following figure will appear



Matplotlib example

3. line plot

We will plot the results of y=2.x equation as curve.

```
>>> plt.plot(X,Y)
[<matplotlib.lines.Line2D object at 0x119ef0f90>]
>>> plt.show()
```

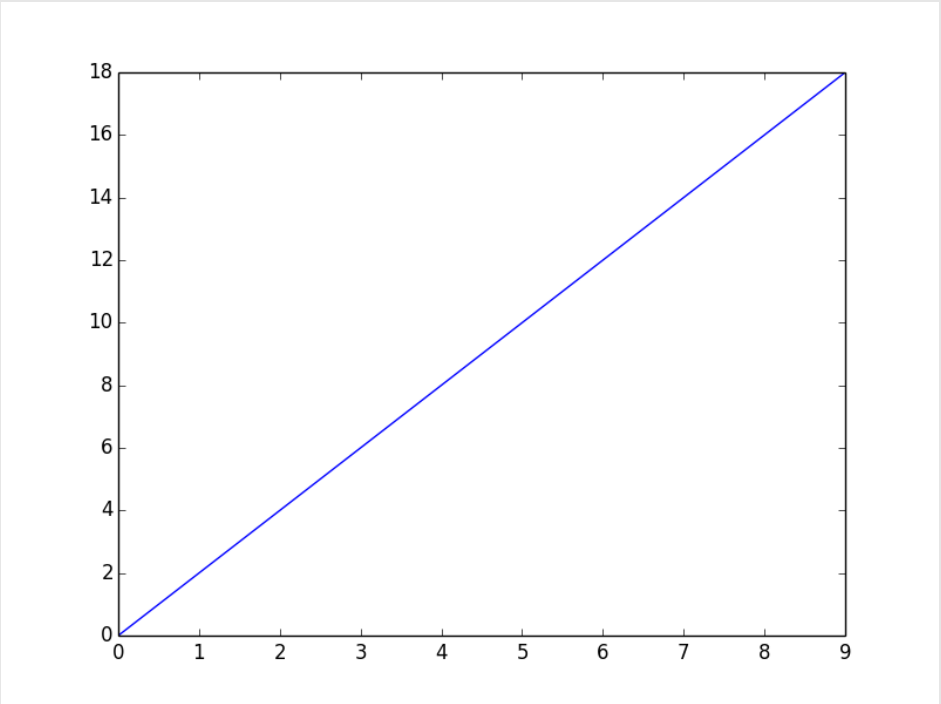
Matplotlib example

3. line plot

We will plot the results of y=2.x equation as curve.

```
>>> plt.plot(X,Y)
[<matplotlib.lines.Line2D object at 0x119ef0f90>]
>>> plt.show()
```

And the following figure will appear



Matplotlib example

4. save the figure in a file

We will plot the results of y=2.x equation as curve.

```
>>> plt.plot(X,Y)
[<matplotlib.lines.Line2D object at 0x11a1e7d50
>]
>>> plt.savefig('figure.pdf')
```

Matplotlib example

4. save the figure in a file

We will plot the results of y=2.x equation as curve.

```
>>> plt.plot(X,Y)
[<matplotlib.lines.Line2D object at 0x11a1e7d50
>]
>>> plt.savefig('figure.pdf')
```

And the file figure.pdf will be created

Matplotlib example

5. let add a legend

We will plot the results of y=2.x equation as curve.

```
>>> plt.plot(X,Y,label="y=2*x")
[<matplotlib.lines.Line2D object at 0x119f0bd50
>]
>>> plt.legend()
<matplotlib.legend.Legend object at 0x1103ef850
>
>>> plt.show()
```

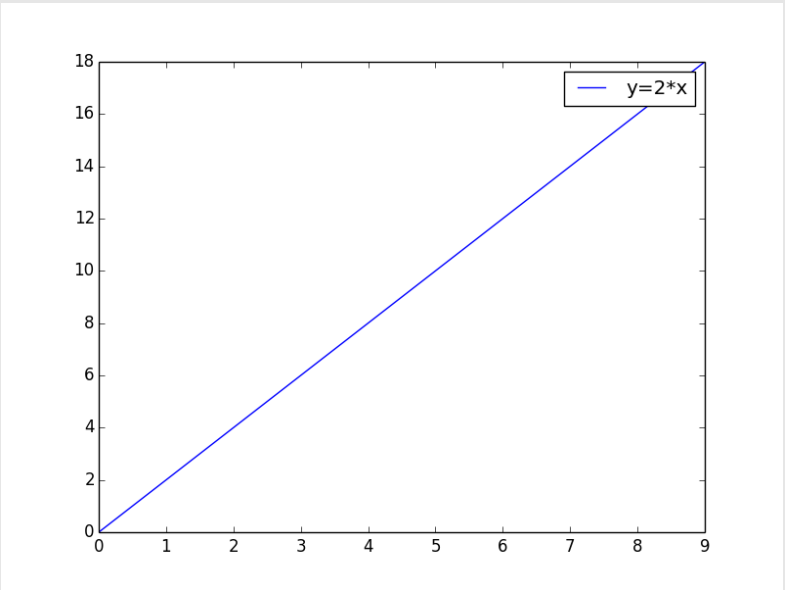
Matplotlib example

5. let add a legend

We will plot the results of y=2.x equation as curve.

```
>>> plt.plot(X,Y,label="y=2*x")
[<matplotlib.lines.Line2D object at 0x119f0bd50
>]
>>> plt.legend()
<matplotlib.legend.Legend object at 0x1103ef850
>
>>> plt.show()
```

And the following figure will appear



Matplotlib example

6. let add axis label and title

We will plot the results of y=2.x equation as curve.

```
>>> plt.plot(X,Y,label="y=2*x")
[<matplotlib.lines.Line2D object at 0x11a2b0210
>]
>>> plt.xlabel("X")
<matplotlib.text.Text object at 0x113753090>
>>> plt.ylabel("Y")
<matplotlib.text.Text object at 0x119f1acd0>
>>> plt.title("curve")
<matplotlib.text.Text object at 0x11a271f10>
>>> plt.show()
```

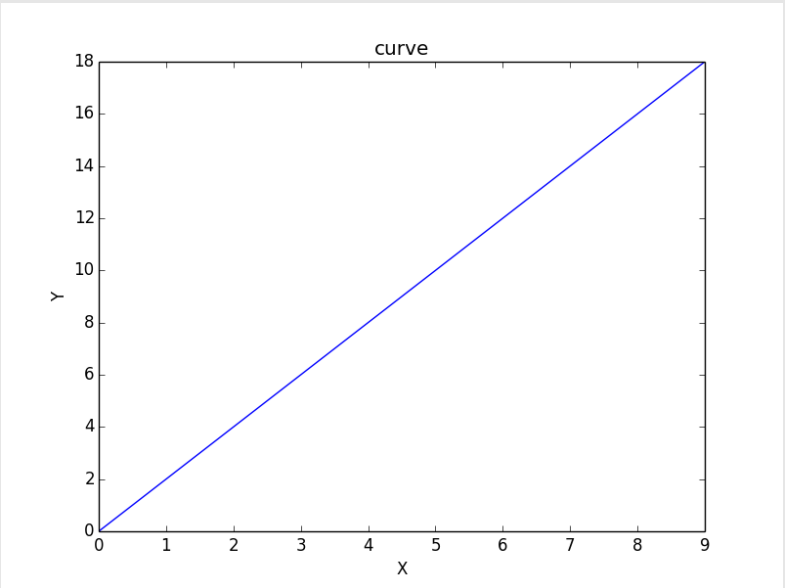
Matplotlib example

6. let add axis label and title

We will plot the results of y=2.x equation as curve.

```
>>> plt.plot(X,Y,label="y=2*x")
[<matplotlib.lines.Line2D object at 0x11a2b0210
>]
>>> plt.xlabel("X")
<matplotlib.text.Text object at 0x113753090>
>>> plt.ylabel("Y")
<matplotlib.text.Text object at 0x119f1acd0>
>>> plt.title("curve")
<matplotlib.text.Text object at 0x11a271f10>
>>> plt.show()
```

And the following figure will appear



webs


IMPORT OS
CUR_PATH = OS.GETCWD()
IGNORE_SET = SET(["__INIT__.PY",
"COUNT_SOURCELINES.PY"])
I DIDN'T KNOW HARRY SPOKE PYTHON.
YEAH, HE'S A PARSER-TONGUE.