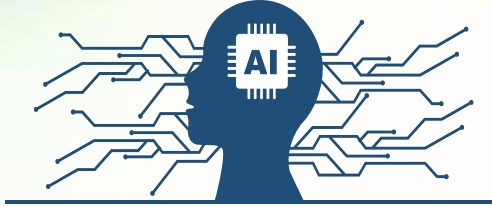


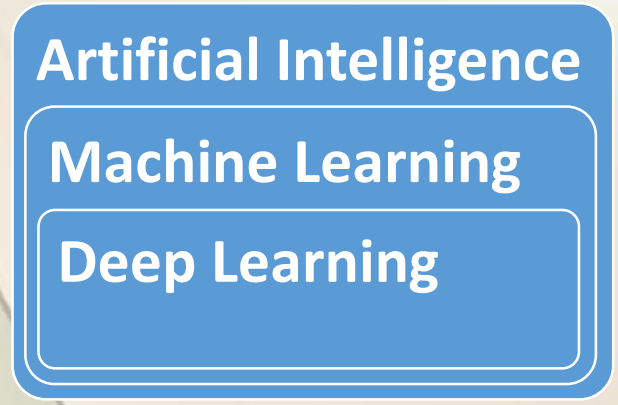
Python Programming

Dr. Sarwan Singh

Agenda

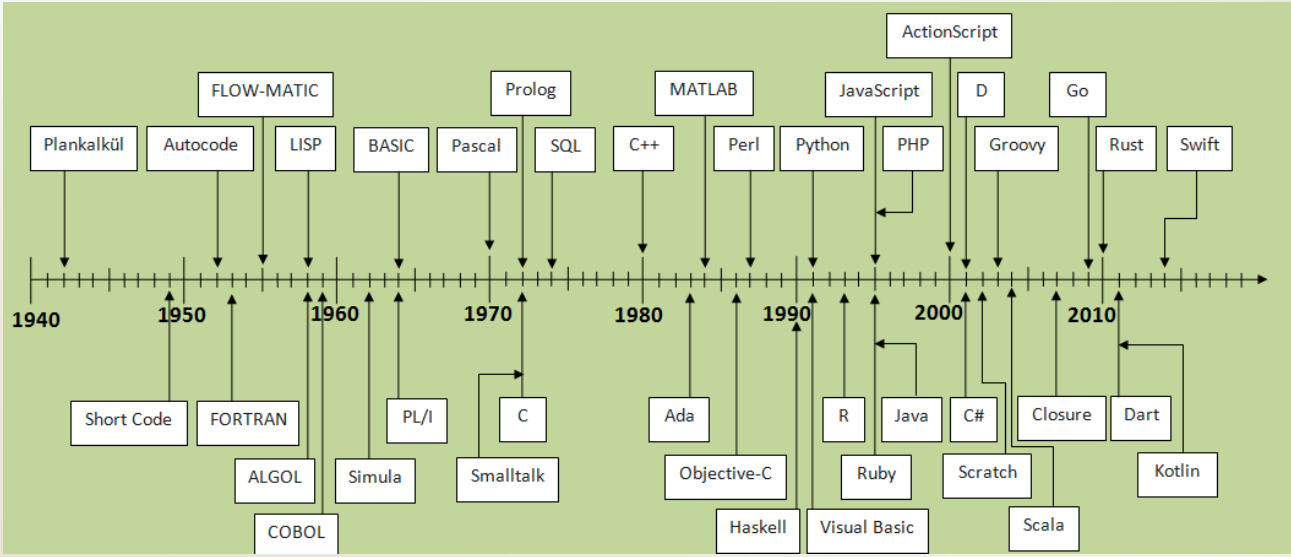
- Python - Introduction, History
- Jupyter notebook
- Python datatypes, operators
- Python control sequences, loops, break continue
- hands on – *programming*

Python was conceived in the late 1980s, & implementation began in December 1989 by **Guido van Rossum**



Python – History

- Python is an object-oriented programming language developed by Guido van Rossum in the early 1990s.
- open-source language that is licensed and freely distributed by the Python Software Foundation, a non-profit organization backed by several prominent companies such as Google and Microsoft



Python … *at first glance*

- Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.
- Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.
- *Python is portable: it runs on many Unix variants, on the Mac, and on PCs under MS-DOS, Windows, Windows NT, and OS/2.*
- *not to mention* around 80 percent of the top computer science programs around the world teach Python as the introduction to the program.

The languages of the 4th millennium

Language	Frequency
Python	26 %
JavaScript	23 %
C#	20 %
Java	15 %
C++	13 %
Ruby	3 %
Perl	0.2%
LISP	< 0.1 %
BASIC	< 0.1 %
Cobol	< 0.1 %
Fortran	< 0.1 %

Source:masterbaboon

History of Python

- A scientist once said

  "I have used a combination of Perl, Fortran, NCL, Matlab, R and others for routine research, but found out this general- purpose language, Python, can handle almost all in an efficient way from requesting data from remote online sites to statistics, and graphics."

- The programming language Python was conceived in the late 1980s, and its implementation was started in December 1989 by Guido van Rossum at CWI in at the National Research Institute for Mathematics and Computer Science in the Netherlands.
- Guido van Rossum was also reading the published scripts from "Monty **Python's** Flying Circus", a BBC comedy series from the 1970s.
- Van Rossum thought he needed a name that was short, unique, and slightly mysterious, so he decided to **call** the language **Python**.

About Python

- Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.
- Python source code is now available under the GNU General Public License (GPL).
- Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language.
- Python is a scripting language like PHP, Perl, Ruby and so much more.
- It can be used for web programming (**django**, Zope, Google App Engine, and much more).
- Can be used for desktop applications (Blender 3D, or even for games pygame).
- Python can also be translated into binary code like java.
- Python can be used for both artificial intelligence and statistical analysis.
- Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

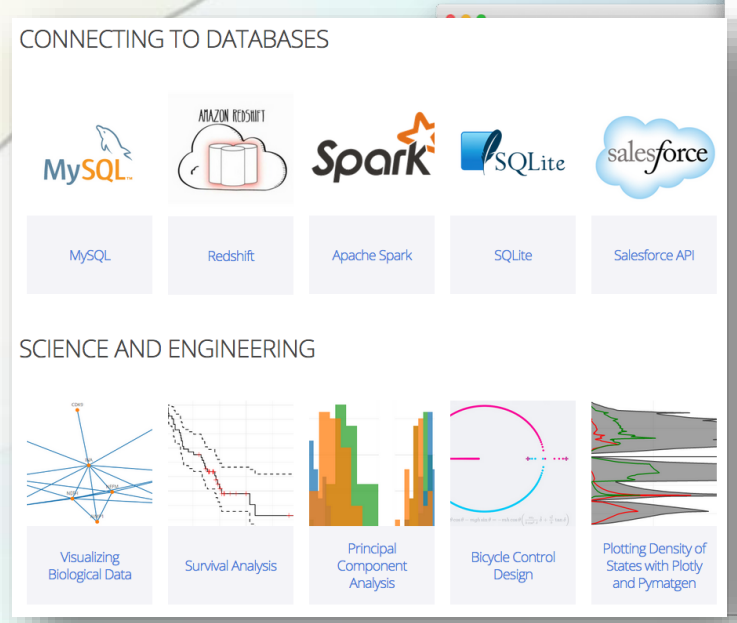
More about Python

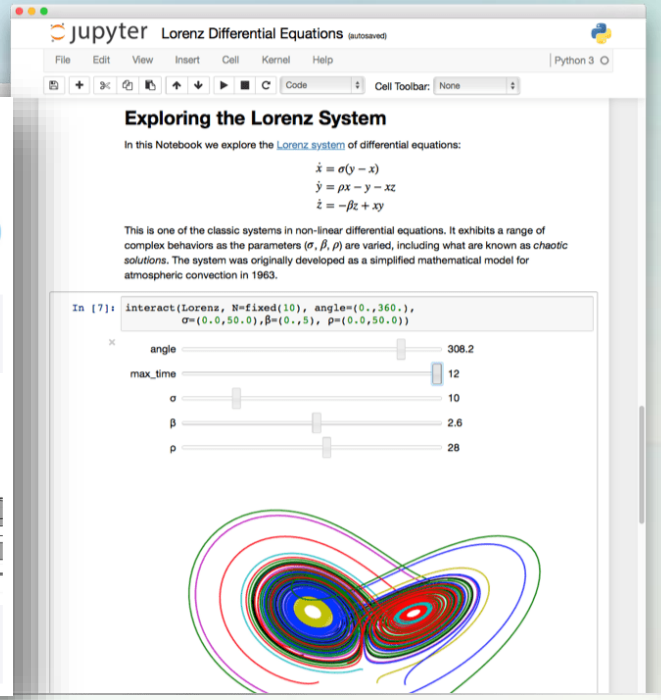
- **Python** is developed under an OSI-approved open source **license**, making it freely usable and distributable, even for **commercial use**. **Python's license** is administered by the **Python** Software Foundation.
- **Python** is not an exception - its most popular implementation is called CPython and is **written in C**
- The biggest difference between Java and Python is that **Java** is statically typed and **Python** is dynamically typed. *This makes **Python** very easy to write and not too bad to read, but difficult to analyze.*
- IronPython is the **Python** running on .NET
- Jython is the **Python** running on the Java Virtual Machine

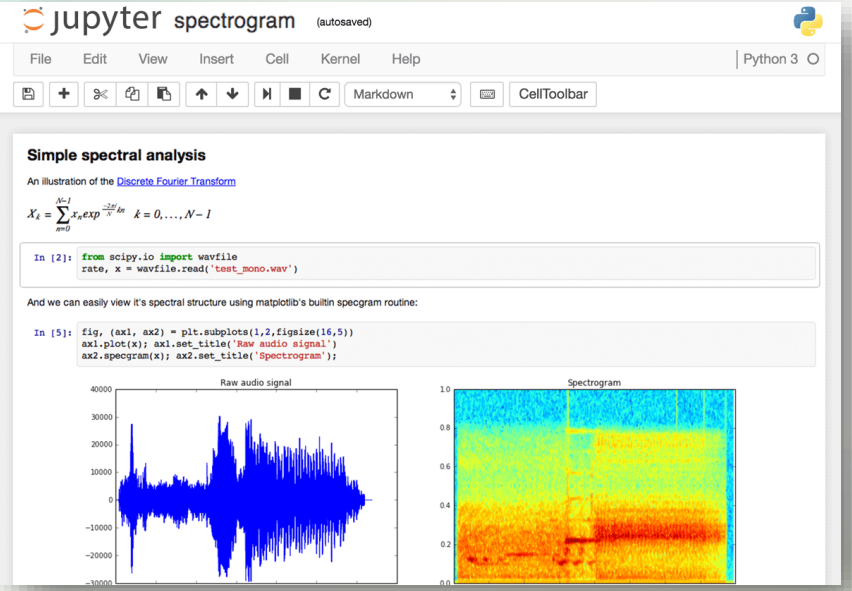
Python-Jupyter Notebook

"Jupyter" is a loose acronym meaning

Julia, Python, and R







Language of choice

The Notebook has support for over 40 programming languages, including Python, R, Julia, and Scala.

Share notebooks

Notebooks can be shared with others using email, Dropbox, GitHub and the Jupyter Notebook Viewer.

Interactive output

Your code can produce rich, interactive output: HTML, images, videos, LaTeX, and custom MIME types.

Big data integration

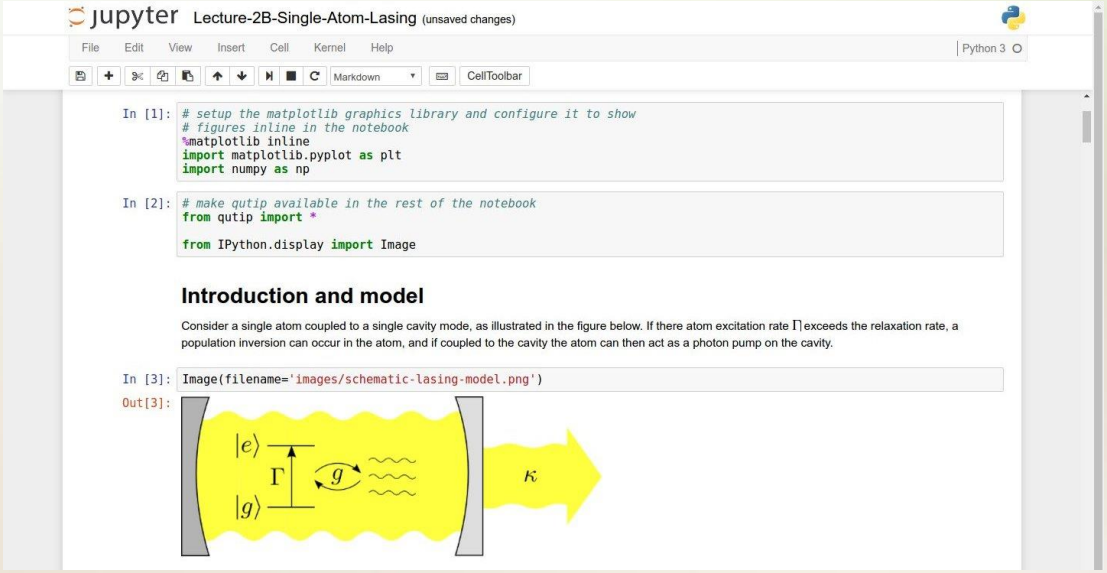
Leverage big data tools, such as Apache Spark, from Python, R and Scala. Explore that same data with pandas, scikit-learn, ggplot2, TensorFlow.

Introduction

- The **Jupyter Notebook** is an open-source web application that allows to create and share documents that contain

  live code, equations, visualizations and narrative text.

- Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.



Source : jupyter.org

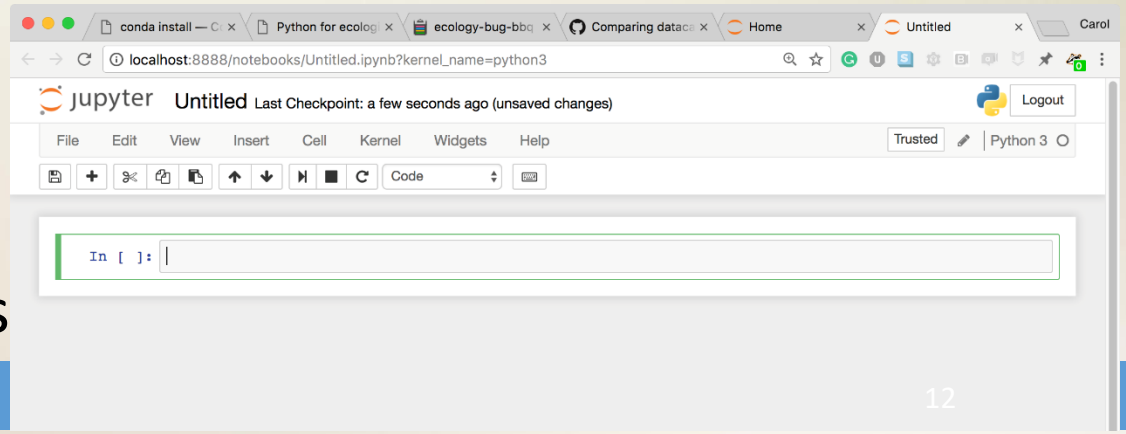
Jupyter Notebook App

- As a server-client application, the Jupyter Notebook App allows you to edit and run your notebooks via a web browser.
- The application can be executed on a PC without Internet access or it can be installed on a remote server, where you can access it through the Internet.
- Two main components are the kernels and a dashboard.
  - A kernel is a program that runs and introspects the user's code. The Jupyter Notebook App has a kernel for Python code, but there are also kernels available for other programming languages.
  - The dashboard of the application not only shows you the notebook documents that you have made and can reopen but can also be used to manage the kernels: you can which ones are running and shut them down if necessary.

History

- Late 1980s, Guido Van Rossum begins to work on Python at the National Research Institute for Mathematics and Computer Science in the Netherlands
- late 2001, twenty years later. Fernando Pérez starts developing IPython.
- In 2005, both Robert Kern and Fernando Pérez attempted building a notebook system. Unfortunately, the prototype had never become fully usable.
- In 2007, they formulated another attempt at implementing a notebook-type system
- In the summer of 2011, the prototype of

  web notebook was incorporated
- In subsequent years, the team got awards

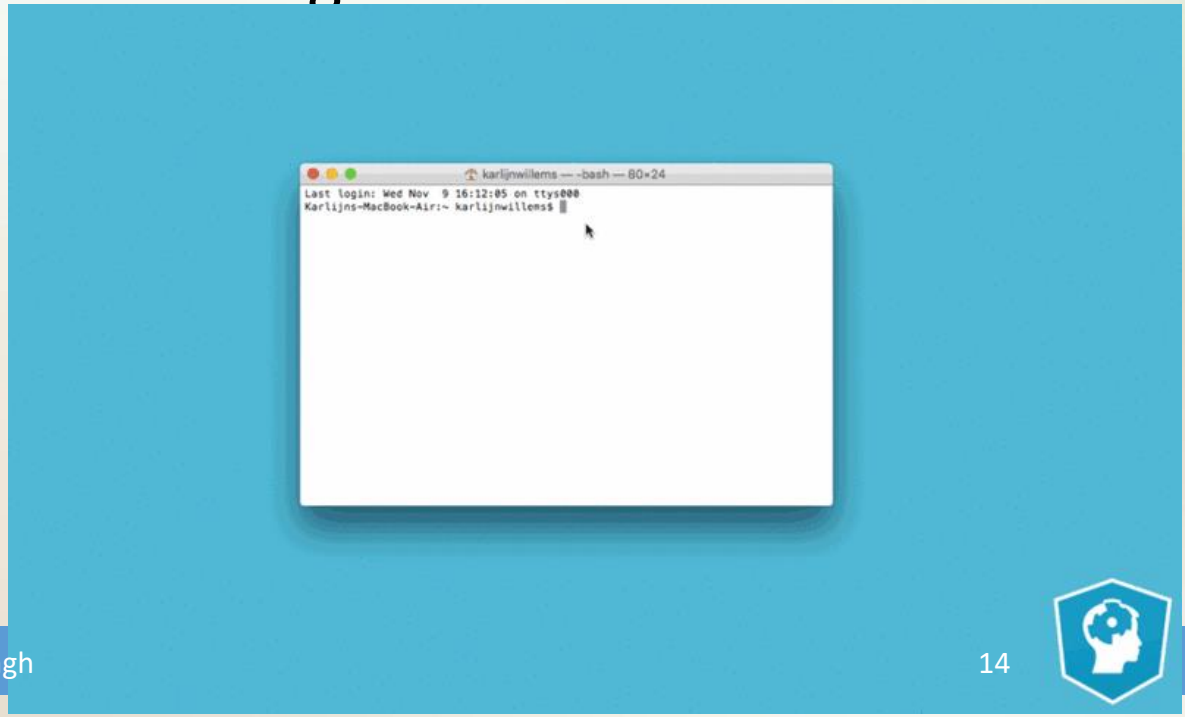


History

- in 2014, Project Jupyter started as a spin-off project from IPython. IPython is now the name of the Python backend, which is also known as the kernel.
- Other products similar to notebook
  - Sage notebook was based on the layout of Google notebooks
  - Mathematica notebooks and Maple worksheets. The Mathematica notebooks were created as a front end or GUI in 1988 by Theodore Gray.

Jupyter Notebooks With The Anaconda Python Distribution

- Use the Anaconda distribution to install both Python and the notebook application.
- The advantage of Anaconda is that you have access to over 720 packages that can easily be installed with Anaconda's conda, a package, dependency, and environment manager.



Source : datacamp.com

IPYNB file

- An **IPYNB file** is a notebook document used by Jupyter Notebook, an interactive computational environment designed to help scientists work with the Python language and their data. ...
- NOTE: Jupyter notebooks were formerly known as IPython notebooks, which is where the "**ipynb**" extension got its name.

Tips To Effectively & Efficiently Use Jupyter Notebooks

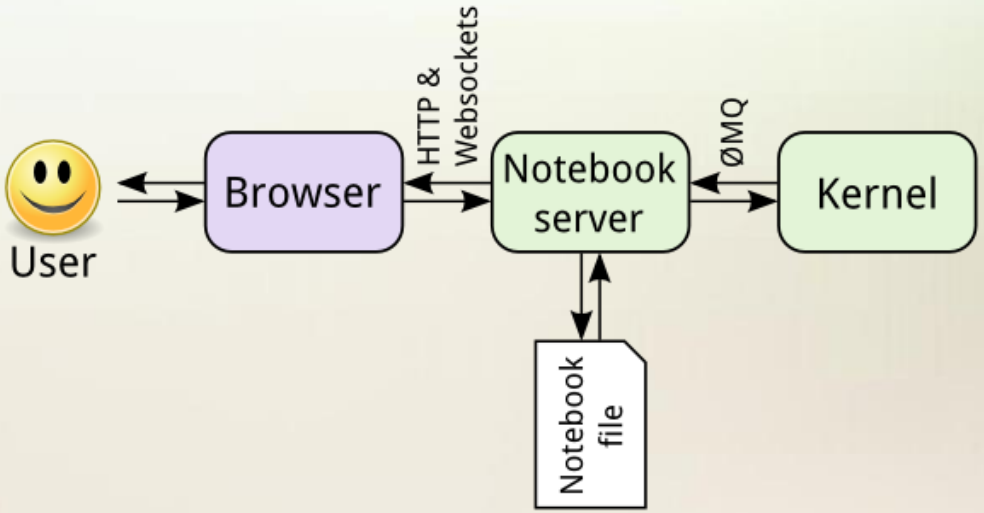
- provide comments and documentation to code.
- consistent naming scheme, code grouping, limit your line length, ...
- Don't forget to name your notebook documents!
- Try to keep the cells of your notebook simple: don't exceed the width of your cell and make sure that you don't put too many related functions in one cell.
- If possible, import your packages in the first code cell of your notebook
- Display the graphics inline. The magic command %matplotlib inline will definitely come in handy to suppress the output of the function on a final line.
- Don't forget to add a semicolon to suppress the output and to just give back the plot itself.
- magic commands such as %run to execute a whole Python script, in a notebook cell.

Source : www.datacamp.com

Notebooks

- The Notebook frontend does something extra. In addition to running your code, it stores code and output, together with markdown notes, in an editable document called a notebook. When you save it, this is sent from your browser to the notebook server, which saves it on disk as a JSON file with a .ipynb extension.

The notebook server, not the kernel, is responsible for saving and loading notebooks, so you can edit notebooks even if you don't have the kernel for that language—you just won't be able to run code. The kernel doesn't know anything about the notebook document: it just gets sent cells of code to execute when the user runs them.



How Jupyter works

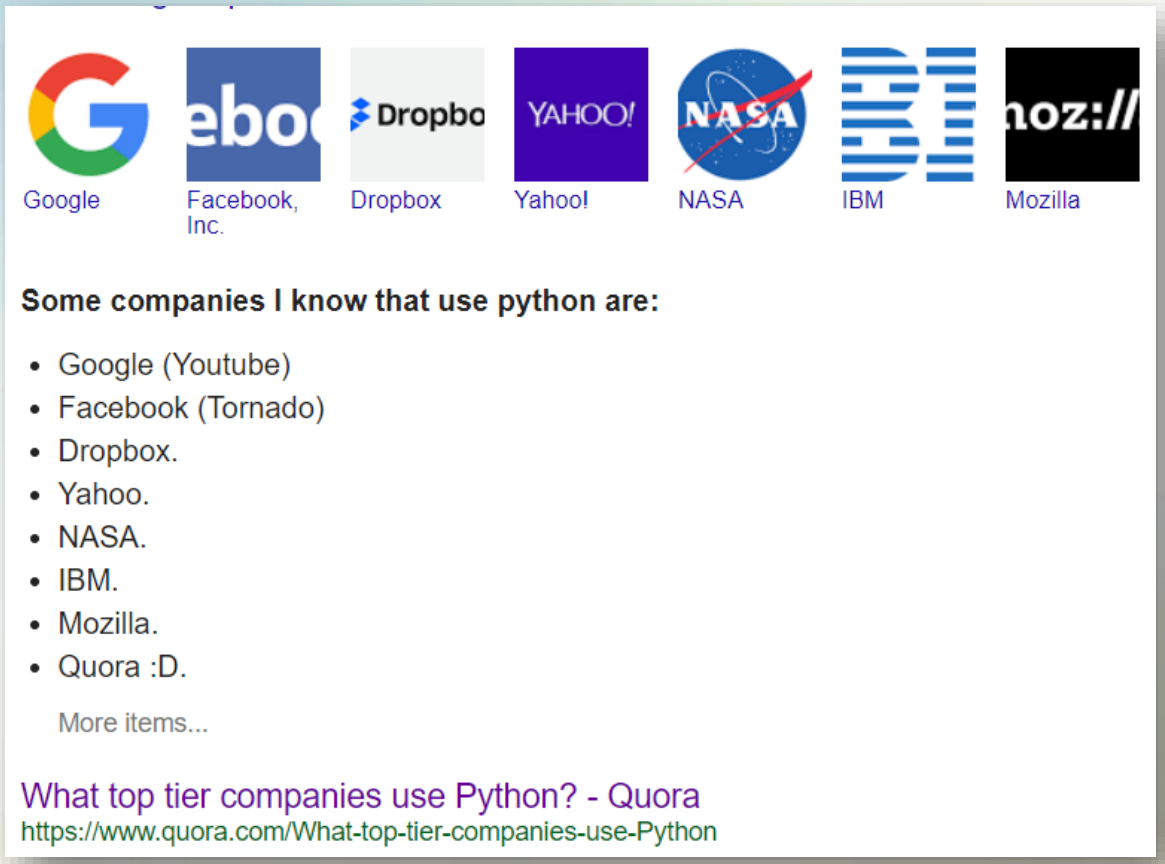
..But first, Terminal IPython

- When you type ipython, you get the original IPython interface, running in the terminal. It does something like this:

```
while True:
    code = input(">>> ")
    exec(code)
```

- Of course, it's much more complex, because it has to deal with multi-line code, tab completion using readline or prompt-toolkit, magic commands, and so on.
- But the model is like that: prompt the user for some code, and when they've entered it, exec it in the same process. This model is often called a REPL, or Read-Eval-Print-Loop.

Programming *with* Python



Python Identifiers

- A Python identifier is a name used to identify a variable, function, class, module or other object.
- An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).
- Python does not allow punctuation characters such as @, $, and % within identifiers.
- Python is a case sensitive programming language.

naming conventions for Python identifiers :

- Class names start with an uppercase letter. All other identifiers start with a lowercase letter.
- Starting an identifier with a single leading underscore indicates that the identifier is private.
- Starting an identifier with two leading underscores indicates a strongly private identifier.
- If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

Building Blocks

- Values and Variables
  - numeric values
  - variables
  - assignment
  - identifiers
  - reserved words
  - comments

Standard Data Types

- Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.
- Python has five standard data types
  - Numbers
  - String
  - List
  - Tuple
  - Dictionary

Python Numbers

- Number data types store numeric values.
- Number objects are created when you assign a value to them. For example –
  - *var1 = 1*
  - *var2 = 10*
- You can also delete the reference to a number object by using the del statement. The syntax of the del statement is –
- *del var1[,var2[,var3[....,varN]]]]*
- You can delete a single object or multiple objects by using the del statement. For example –
  - *del var*
  - *del var_a, var_b*

Checking the type of datatype

```
In [1]: type('hello')
Out[1]: str

In [2]: type (10)
Out[2]: int

In [3]: type(20.5)
Out[3]: float

In [4]: a=10
        a
Out[4]: 10

In [5]: del a

In [6]: a
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-6-60b725f10c9c> in <module>()
----> 1 a

NameError: name 'a' is not defined
```

Assigning Values to Variables

- Python variables do not need explicit declaration to reserve memory space.
- The declaration happens automatically when you assign a value to a variable.
- The equal sign (=) is used to assign values to variables.
- For example –

```
counter = 100          # An integer assignment
miles   = 1000.0       # A floating point
name    = “Nielit"      # A string
print(counter)
Print(miles)
Print(name)
```

- Here, 100, 1000.0 and “Nielit" are the values assigned to *counter*, *miles*, and *name* variables, respectively.

Reserved words

and	exec	not
assert	finally	or
break	for	pass
class	from	print
continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield

Multiline/single line

```
In [7]: a=10;b=20;c=30;
        print(a,b,c)

        10 20 30

In [8]: a\
        =\
        50
        print (a)

        50
```

Data type

Python supports four different numerical types –

- int (signed integers)
- long (long integers, they can also be represented in octal and hexadecimal)
- float (floating point real values)
- complex (complex numbers)

Basic operator

- Arithmetic Operators
- Comparison (Relational) Operators
- Assignment Operators

(+= , -= , *= , //= , %=)

- Logical Operators

(and, or, not)

- Bitwise Operators

Arithmetic Operator
+ Addition
- Subtraction
* Multiplication
/ Division
% Modulus
** Exponent
// Floor division
9//2 -> 4

Comparison Operator
==
!=
<>
>
<
>=
<=

Bitwise Operator
& Binary AND
| Binary OR
^ Binary XOR
~ Binary Ones Complement
<< Binary Left Shift
>> Binary Right Shift

Decision making

```
x = int( input('enter marks'))
if (x>50) : print('pass')
else : print('fail')
```

Or

```
x = int( input('enter marks'))
if (x>50) :
    print('pass')
else :
    print('fail')
```

```
if expression1:
        statement(s)
        if expression2:
                statement(s)
        elif expression3:
                statement(s)
        elif expression4:
                statement(s)
        else:
                statement(s)
  else:
        statement(s)
```