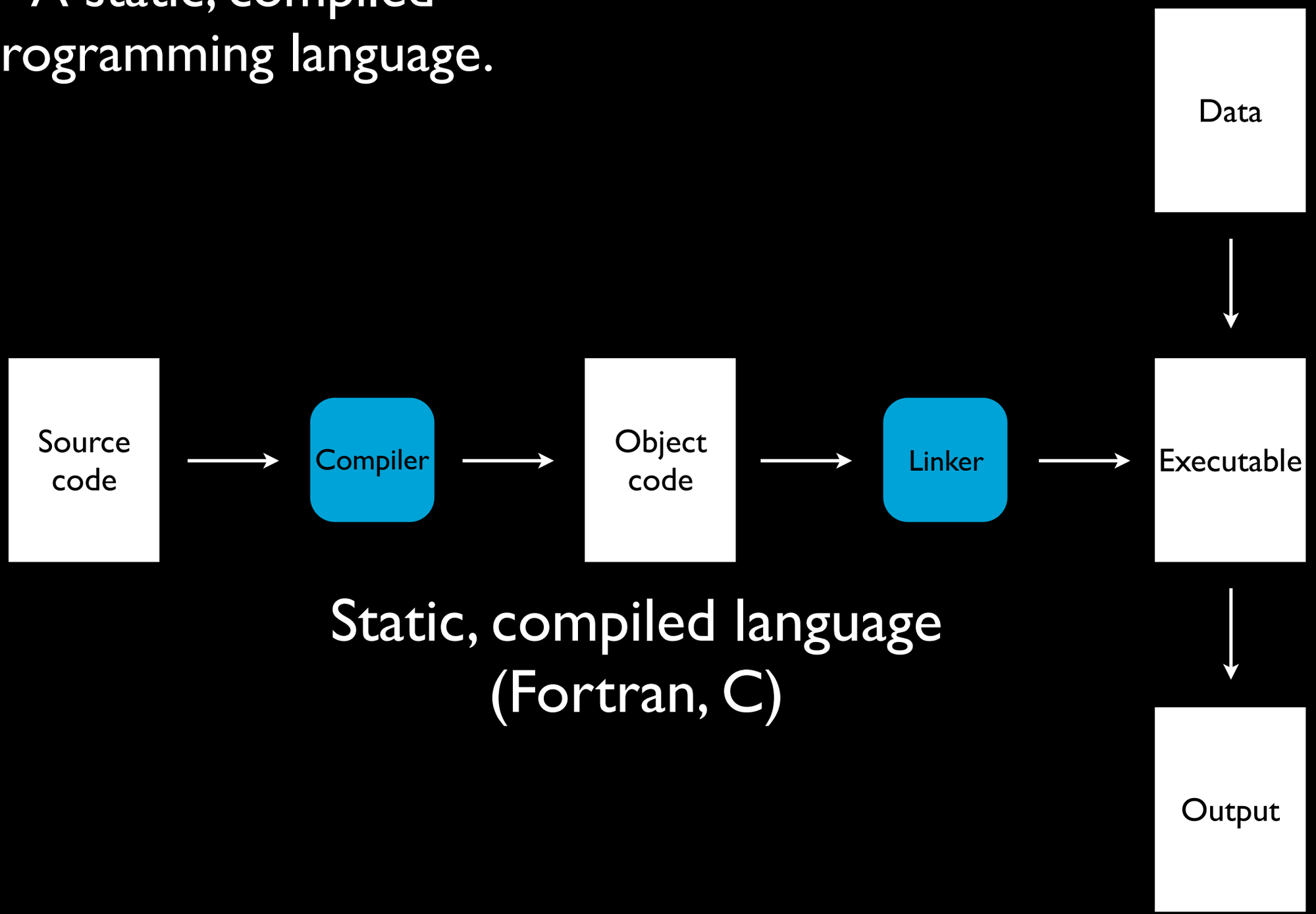
Python Bytes for Earth Scientists

Leo Siqueira
lsiqueira@rsmas.miami.edu

Python is Not:

A static, compiled programming language.

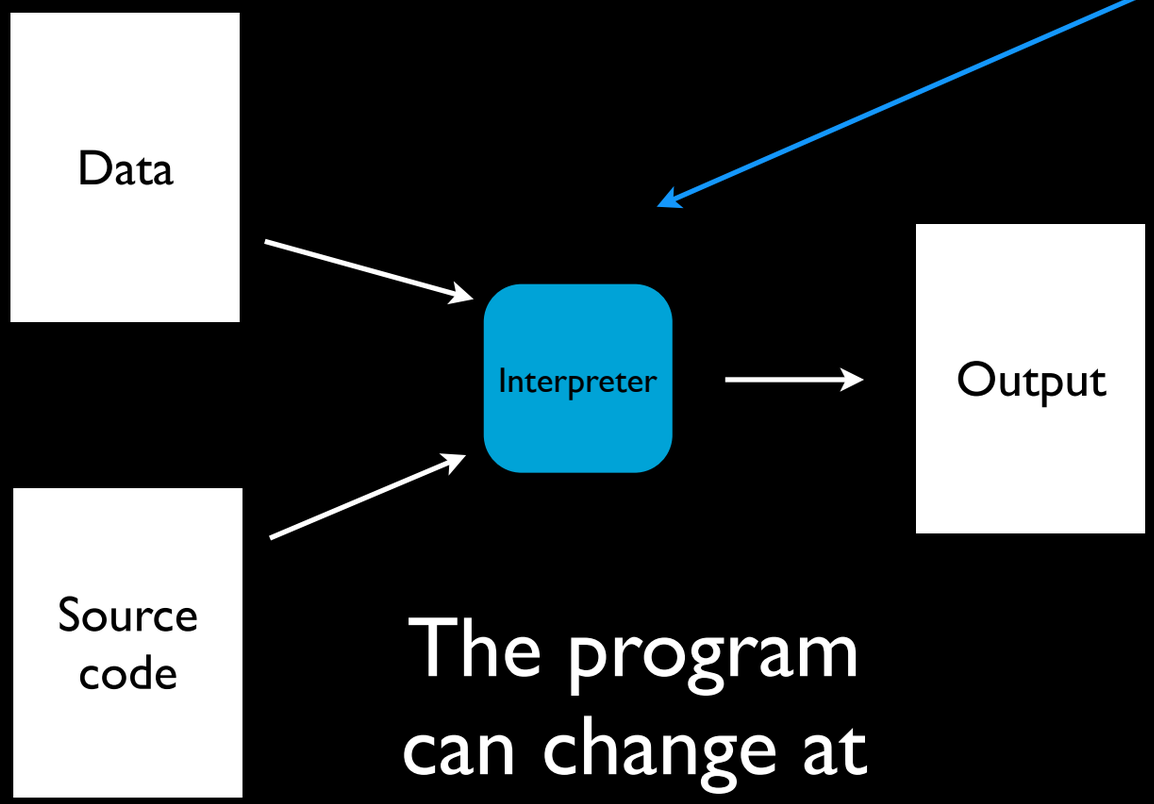


Static, compiled language (Fortran, C)

Python is:

A dynamic, interpreted programming language.

An object-oriented programming language.



The program can change at run time

Object-oriented Programming
OOP

- Anything of the real (or mathematical) world which needs to be manipulated by the computer is modeled by an object.
- Each object is an instance of some class (of objects).

Object-oriented Programming
OOP

- Object: A portion of memory which contains the information needed to model the real world thing.
- Class: Defines the data structure used to store the objects which are instances of the class together with their behavior.
- Unlike Java, Python does not force you to use OOP paradigm exclusively.
- Python also supports procedural programming with modules and functions, so you can select the most suitable programming paradigm for each part of your program (scripting, GUI applications, etc…)

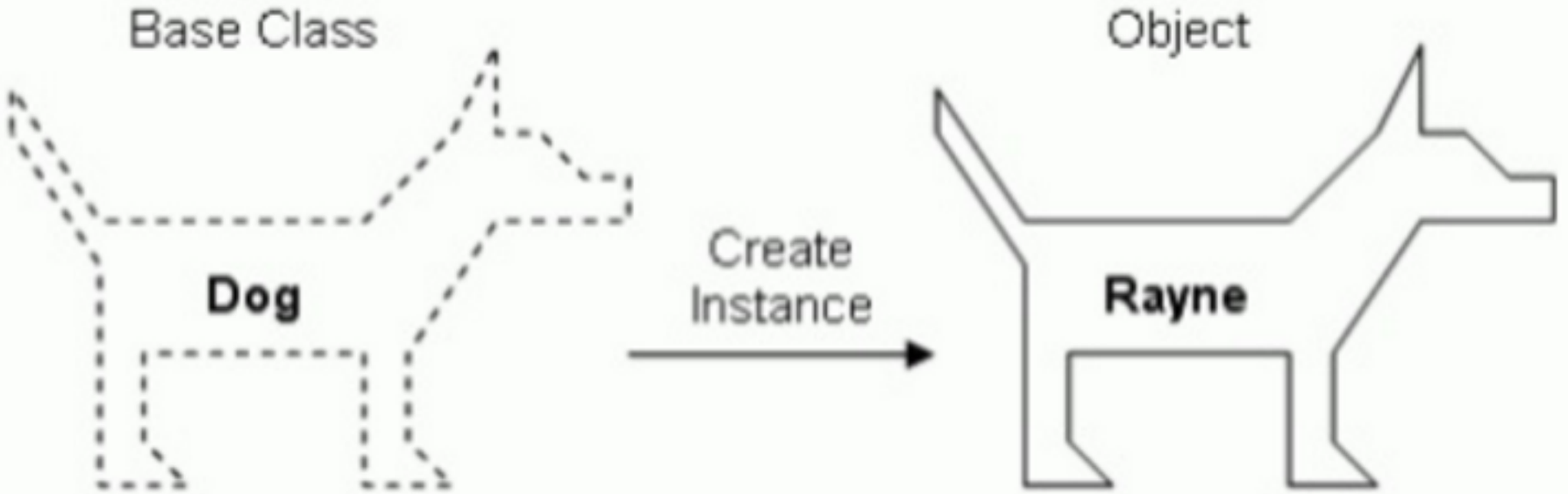
What is an object?

- Objects are the basic tun-time entities in an object-oriented system.
- Objects are Python's abstraction for data. All data in a Python program is represented by objects.
- They may represent any item that the program must handle (e.g., data types, person, bank account).
- Objects have two components - attributes and methods (defined within a given "class" of object).

Defining a Class (of object)

- A class is a special data type which defines how to build a certain kind of object.
- The class stores attributes (data items) that are shared by all the instances of this class (reusable, avoids code repetition).
- Instances are objects that are created which follow the definition given inside of the class.

Defining a Class



Properties	Methods
Color	Sit
Eye Color	Lay Down
Height	Shake
Length	Come
Weight	

Property values	Methods
Color: Gray, White, and Black	Sit
Eye Color: Blue and Brown	Lay Down
Height: 18 Inches	Shake
Length: 36 Inches	Come
Weight: 30 Pounds	

Defining a Class (of objects)

```
class BankAccount:
    def __init__(self):
        self.balance = 0

    def withdraw(self, amount):
        self.balance -= amount
        return self.balance

    def deposit(self, amount):
        self.balance += amount
        return self.balance

class MinimumBalanceAccount(BankAccount):
    def __init__(self, minimum_balance):
        BankAccount.__init__(self)
        self.minimum_balance = 10

    def withdraw(self, amount):
        if self.balance - amount < self.minimum_balance:
            print 'Sorry, minimum balance must be maintained.'
        else:
            BankAccount.withdraw(self, amount)
```

Method to initialize the object (constructor)

First arg of a method is a reference to the current instance of the class

it's very clear you mean the instance attribute self.balance and not “some other object called "balance"

Inheritance: new child class inherits features from parent (base) class adding new features with little mod to existing class

Re-usability of code! Mistakes are copied and to change anything one has to remember the location of all the copies.

Everything is an object, even a Class.

You might want to consider R if:

- R is clearly at the forefront in new statistical algorithm development meaning you are most likely to find that new(ish) procedure in R.
- Performance is of secondary importance.
- Free is important.

You might want to consider MATLAB if:

- Commercial support, and a clean channel to report issues, is important.
- Documentation and organization of modules is more important than raw routine availability.
- Performance is more important than scope of available packages. MATLAB has optimizations, which is not readily available in most other packages.

But why Python?

Open

Portable

Multi-paradigm

Large and stable community

Data set construction

Wide library support

Easily work with Fortran/C/C++

High-quality modules

Publication quality figure ploting

Easily read/write netcdf and grib data

powerful & flexible

Knowledge of Python, is complementary to R/MATLAB

End-to-end solution

Comes with batteries included

Free

Standard library

~300 official modules designed to be shipped with interpreter

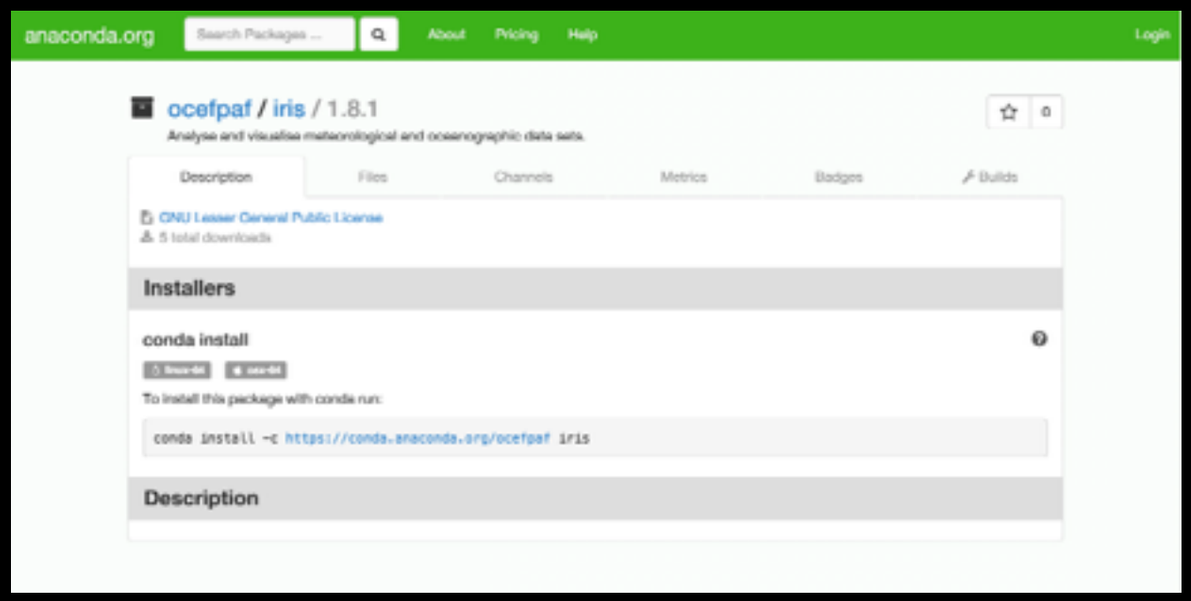
- Numpy - matrix lab + numerics
- Matplotlib - plotting
- Basemap - mapping
- NetCDF4 - read/write
- Pandas - Statistics
- DASK - Threading and multiprocessing

End-to-end solution

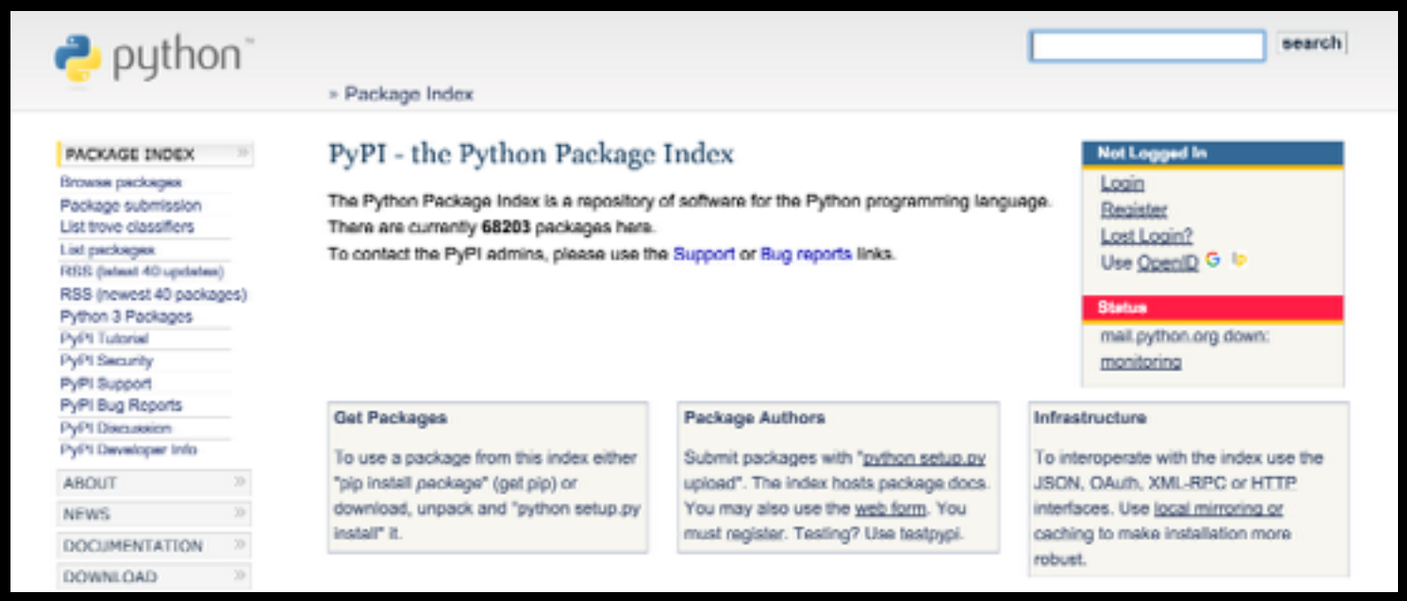
http://docs.python.org/library/

Package index Conda and Pypi

~15.000 free to use packages (collections of modules) in a searchable index



https://anaconda.org/



http://pypi.python.org/

Distribution

The interpreter

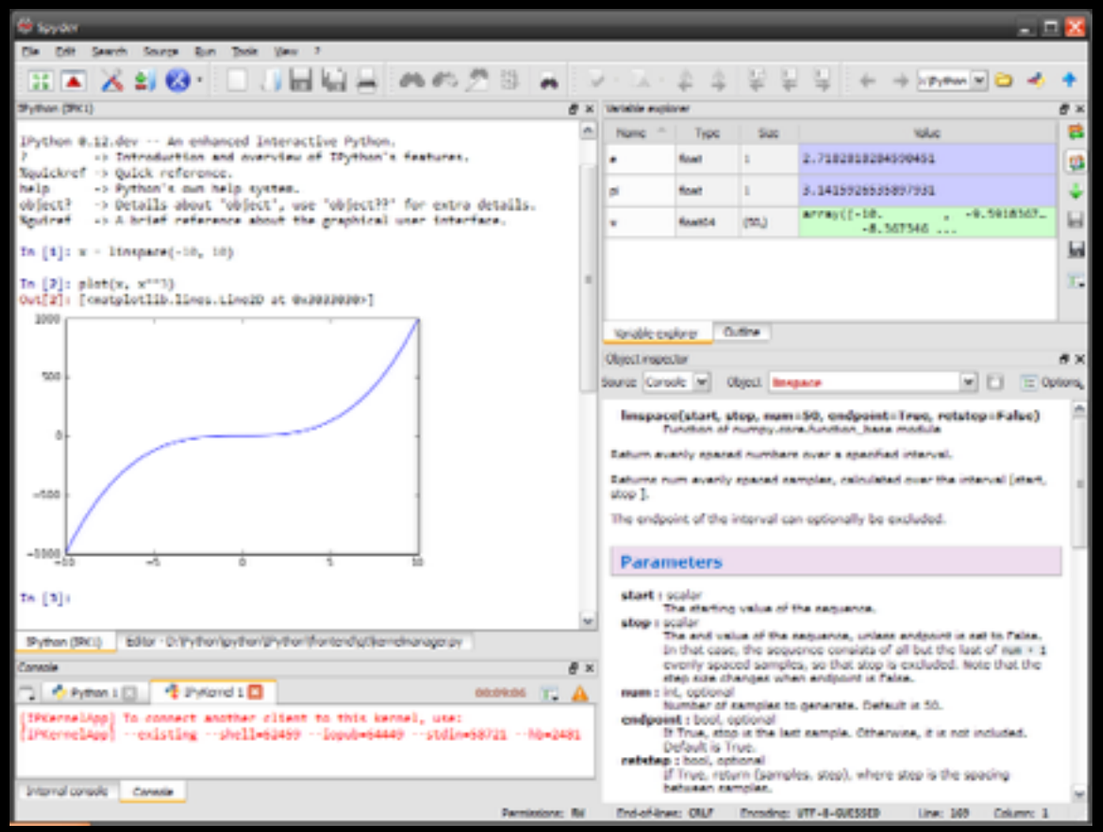
python

Language parser and implementation

Standard library modules

Third party modules

Module management and upgrade tool



Interpreter ± collection of modules ± other stuff = Python distribution

Anaconda Python Distribution

- Anaconda includes Python interpreters, IDEs, easy automatic installation of over 150 packages.

- Over 200 additional open source packages can be individually installed from the Anaconda repository by using the "conda install" command.

- Others can be downloaded using the "pip install" command which is included and installed with Anaconda.

- Anaconda is available for Linux, OS X and Windows, and is free and Open Source.

http://docs.continuum.io/anaconda

Anaconda Python Distribution

- Don't need root privileges to install Anaconda.
- Miniconda is a small "bootstrap" version that includes only conda and conda-build, and installs Python.
- Scientific packages and their dependencies can be installed individually as needed.

Python Bindings

- pip install cdo (use the power of cdo)
- Cython, Numba (use embedded C code)
- f2py - use embedded Fortran code
- Pyngl - use NCAR graphics language

Thank you