Introduction to Python

Parham Alvani
Amirkabir University of Technology

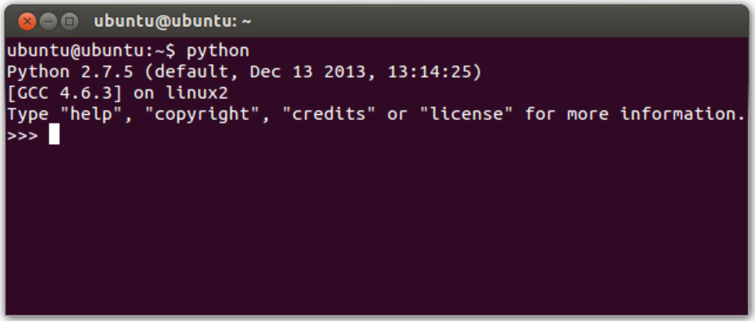
parham.alvani@gmail.com

May 14, 2015

what is python...

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable..

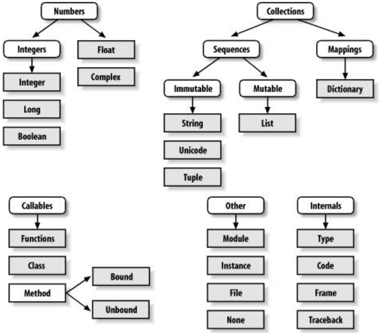
Environment



```
ubuntu@ubuntu:~$ python
Python 2.7.5 (default, Dec 13 2013, 13:14:25)
[GCC 4.6.3] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

Variable Types

- Numbers
- String
- List
- Tuple
- Dictionary



Numbers
Integers
Integer
Long
Boolean
Float
Complex

Collections
Sequences
Immutable
String
Unicode
Tuple
Mutable
List
Mappings
Dictionary

Callables
Functions
Class
Method
Bound
Unbound

Other
Module
Instance
File
None

Internals
Type
Code
Frame
Traceback

Numbers

- Number data types store numeric values.
- They are immutable data types, means that changing the value of a number data type results in a newly allocated object.

Strings

- Strings are amongst the most popular types in Python.
- We can create them simply by enclosing characters in quotes.
- Python treats single quotes the same as double quotes.

Lists

- The most basic data structure in Python is the sequence.
- Each element of a sequence is assigned a number - its position or index.
- The first index is zero, the second index is one, and so forth.

Tuples

- A tuple is a sequence of immutable Python objects.
- Tuples are sequences, just like lists.
- The differences between tuples and lists are :
  - the tuples cannot be changed unlike lists
  - tuples use parentheses, whereas lists use square brackets.

Dictionary

- Each key is separated from its value by a colon (:)
- the items are separated by commas
- and the whole thing is enclosed in curly braces.

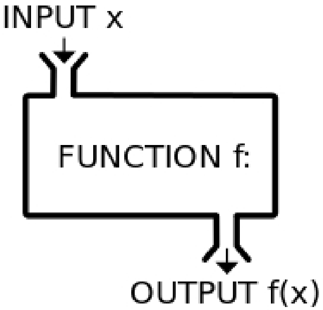
Flow Control

- If-Then-Else
- For
- While
- Exceptions

FLOW CONTROL

Functions

- Function blocks begin with the keyword def, followed by the function name and parentheses.
- Any input parameters or arguments should be placed within these parentheses.
- The first statement of a function can be an optional statement - the documentation string of the function or docstring.



Functions

```
def square(x):
    return x * x
```

```
def hello():
    return "Hello"
```

```
def printme( str ):
    "This prints a passed string into this function"
    print str
    return
```

Classes

- The class statement creates a new class definition.
- The name of the class immediately follows the keyword class followed by a colon as follows

Classes

```
class Employee:
    """
    Common base class for all employees
    """
    empCount = 0

    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
        Employee.empCount += 1

    def displayCount(self):
        print "Total Employee %d" % Employee.empCount

    def displayEmployee(self):
        print "Name : ", self.name,  ", Salary: ", self.salary
```

```
emp1 = Employee("Zara", 2000)
```

Garbage Collection

- Python deletes unneeded objects automatically to free the memory space.

Inheritance

- Instead of starting from scratch, you can create a class by deriving it from a preexisting class by listing the parent class in parentheses after the new class name.

Inheritance

```
class SubClassName (ParentClass1[, ParentClass2, ...]):
    """
    Optional class documentation string
    """
    # class_suite
```

Base Overloading Methods

- __init__(self [,args...]) : Constructor (with any optional arguments)
- __del__(self) : Destructor, deletes an object
- __repr__(self) : Evaluatable string representation
- __str__(self) : Printable string representation
- __lt__(self, other):
- __le__(self, other):
- __eq__(self, other):
- __ne__(self, other):
- __gt__(self, other):
- __ge__(self, other):
  These are the so-called rich comparison methods,
  and are called for comparison operators in preference to __cmp__() below.

Base Overloading Methods

- __cmp__ (self, x) : Called by comparison operations if rich comparison is not defined.
- __add__(self, other):
- __sub__(self, other):
- __mul__(self, other):
- __floordiv__(self, other):
- __mod__(self, other):
- __divmod__(self, other):
- __pow__(self, other[, modulo]):

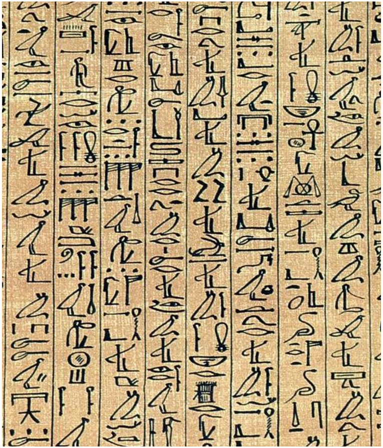
Base Overloading Methods

- __lshift__(self, other):
- __rshift__(self, other):
- __and__(self, other):
- __xor__(self, other):
- __or__(self, other):
  These methods are called to implement the binary arithmetic operations
  +, -, *, //, %, divmod(), pow(), **, <<, >>, &, ^, |

A Brief History of Data Management!

4000 B.C

- Manual recording
- From tablets to papyrus, to parchment, and then to paper

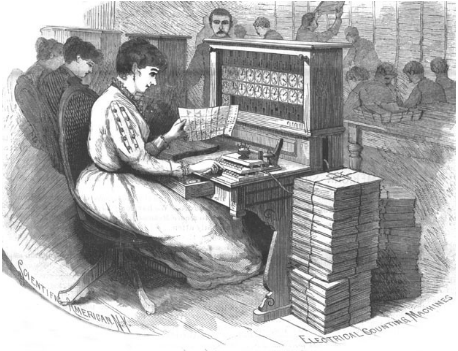


1450

- Gutenberg's printing press

1800's - 1940's

- Punched cards (no fault-tolerance)
- Binary data
- 1890: US census
- 1911: IBM appeared



IBM

1940's - 1970's

- Magnetic tapes
- Batch transaction processing
- File-oriented record processing model (e.g., COBOL)
- Hierarchical DBMS (one-to-many)
- Network DBMS (many-to-many)

1980's

- Relational DBMS (tables) and SQL
- ACID
- Client-server computing
- Parallel processing

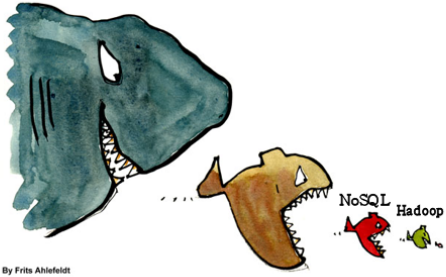
A
C
I
D
ATOMICITY
CONSISTENCY
ISOLATION
DURABILITY

1990's - 2000's

- The Internet...

2010's

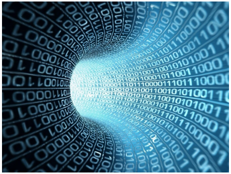
- NoSQL: BASE instead of ACID
- Big Data



Big Data

- In recent years we have witnessed a dramatic increase in available data.

- For example, the number of web pages indexed by Google, which were around one million in 1998, have exceeded one trillion in 2008, and its expansion is accelerated by appearance of the social networks.

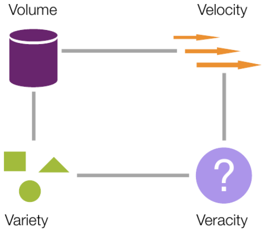


Big Data Definition

- Big Data refers to datasets and flows large enough that has outpaced our capability to store, process, analyze, and understand.

The Four Dimensions of Big Data

- Volume: data size
- Velocity: data generation rate
- Variety: data heterogeneity
- Veracity: uncertainty of accuracy and authenticity of data

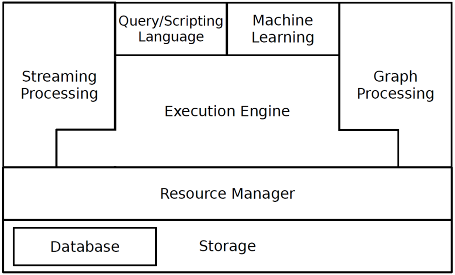


Big Data Market Driving Factors

- Mobile devices
- Internet of Things (IoT)
- Cloud computing
- Open source initiatives

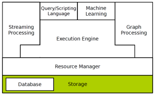
The Big Data Stack!

Big Data Analytics Stack



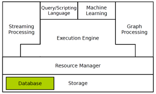
Big Data - Storage (Filesystem)

- Traditional filesystems are not well-designed for large-scale data processing systems.
- Efficiency has a higher priority than other features, e.g., directory service.
- Massive size of data tends to store it across multiple machines in a distributed way.
- HDFS, Amazon S3, ...



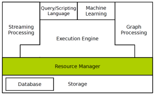
Big Data - Database

- Relational Databases Management Systems (RDMS) were not designed to be distributed.
- NoSQL databases relax one or more of the ACID properties: BASE
- Different data models: key/value, column-family, graph, document.
- Dynamo, Scalaris, BigTable, Hbase, Cassandra, MongoDB, Voldemort, Riak, Neo4J, ...



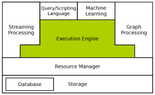
Big Data - Resource Management

- Different frameworks require different computing resources.
- Large organizations need the ability to share data and resources between multiple frameworks.
- Resource management share resources in a cluster between multiple frameworks while providing resource isolation.
- Mesos, YARN, Quincy, ...



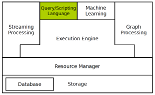
Big Data - Execution Engine

- Scalable and fault tolerance parallel data processing on clusters of unreliable machines.
- Data-parallel programming model for clusters of commodity machines.
- MapReduce, Spark, Stratosphere, Dryad, Hyracks, ...



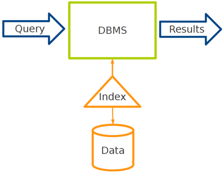
Big Data - Query/Scripting Language

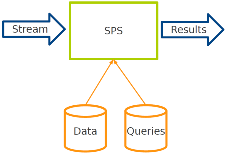
- Low-level programming of execution engines, e.g., MapReduce, is not easy for end users.
- Need high-level language to improve the query capabilities of execution engines.
- It translates user-defined functions to low-level API of the execution engines.
- Pig, Hive, Shark, Meteor, DryadLINQ, SCOPE, ...



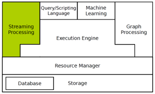
Big Data - Stream Processing

- Providing users with fresh and low latency results.
- Database Management Systems (DBMS) vs. Stream Processing Systems (SPS)



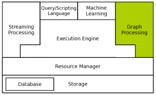


- Storm, S4, SEEP, D-Stream, Naiad, ...



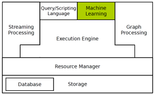
Big Data - Graph Processing

- Many problems are expressed using graphs: sparse computational dependencies, and multiple iterations to converge.
- Data-parallel frameworks, such as MapReduce, are not ideal for these problems: slow
- Graph processing frameworks are optimized for graph-based problems.
- Pregel, Giraph, GraphX, GraphLab, PowerGraph, GraphChi, ...

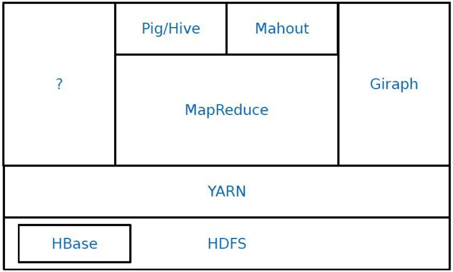


Big Data - Machine Learning

- Implementing and consuming machine learning techniques at scale are difficult tasks for developers and end users.
- There exist platforms that address it by providing scalable machine-learning and data mining libraries.
- Mahout, MLBase, SystemML, Ricardo, Presto, ...



Hadoop Big Data Analytics Stack



?

Pig/Hive

Mahout

MapReduce

Giraph

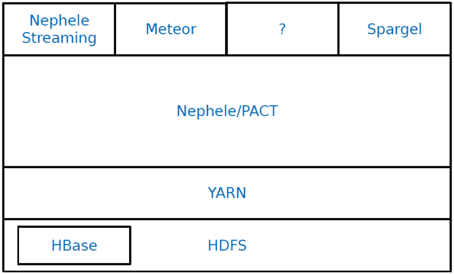
YARN

HBase

HDFS

Stratosphere Big Data Analytics Stack

StratoSphere
Above the Clouds



Nephele Streaming

Meteor

?

Spargel

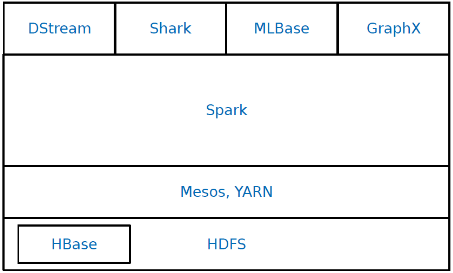
Nephele/PACT

YARN

HBase

HDFS

Spark Big Data Analytics Stack



DStream

Shark

MLBase

GraphX

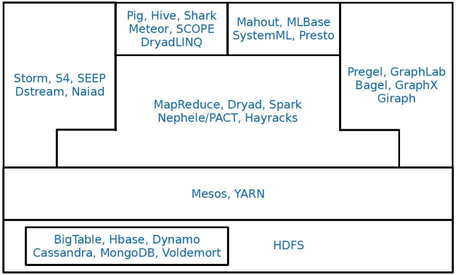
Spark

Mesos, YARN

HBase

HDFS

Summary



Storm, S4, SEEP
Dstream, Naiad

Pig, Hive, Shark
Meteor, SCOPE
DryadLINQ

Mahout, MLBase
SystemML, Presto

Pregel, GraphLab
Bagel, GraphX
Giraph

MapReduce, Dryad, Spark
Nephele/PACT, Hayracks

Mesos, YARN

BigTable, Hbase, Dynamo
Cassandra, MongoDB, Voldemort

HDFS

Questions?