

Python Software Foundation Grant Proposal for:

XAYA – A Language for Thinking With Data

By Mishtu Banerjee, Harmeny Systems Ltd.

This proposal is supported by a website at:

<http://www.harmeny.com/twiki/bin/view/Main/XAYAThinkingWithData>

The website has more detail on XAYA's concepts, proposed methodology, and some preliminary coding examples.

a. **What the objective of the project is:**

The objective is to create XAYA, a Pythonic logic-based mini- language for thinking with data, and to document the creation of that language as an extended tutorial of “Thinking with Data in Python”.

Thinking with Data is a trial-and-error process of exploration of relationships amongst data (in this case “data” can mean everything from numerical observations, to semantic ontologies). One asks questions (queries), one gets answers (query result sets). Is the answer what you expected? No? Revise, and repeat. A great deal of planning, “business intelligence” and analysis work in various industries requires just such “thinking with data”, where questions become iteratively refined based on previous results. One of XAYA's key goals is to provide a framework that can introduce Python to that business environment, and in particular to planners and analysts who are not primarily programmers, but who are required to reason with complex data and make valid inferences.

Python is well known for its support for Imperative, Functional and OO programming styles. The XAYA project focuses on highlighting Python support for a Declarative programming style. Python's strong support for the dictionary data structure and recent inclusion of a “sets” module into the standard library are the basis for developing a declarative programming style in Python. By “declarative programming style” we mean one where the user focuses on the “what is being asked” of a data-set, rather than the specifics of “how is the result being calculated”. This style of programming is familiar to those who have used logic based languages or rules engines (such as Prolog or Jess), or have used query languages (such as SQL or XPATH). Our goal is to facilitate this programming style within Python, as a way of thinking with data, emphasizing the process of asking and refining questions about logical relationships inherent in sets of data.

XAYA will focus on modeling information that integrates diverse data sources¹, that allows for a rapid transition from concepts associated with data representation and storage (e.g. Schemas, Types) to those associated with data analysis (data summarization, statistics, process models). The mini-language is being developed to serve several audiences: (i) a common language for software developers to work with domain experts (who are usually not programmers), (ii) planners, managers and analysts (who are rarely programmers) to work with data and sketch their out information systems without needing to know a great detail about underlying system, or query processes, via a common metaphor – navigating a graph, (iii) “power users” (often a subset of the same planners/managers/analysts) who are not (primarily) programmers to be introduced to a subset of Python that

¹ Diverse data sources: relational databases, XML, RDF, text files, OWL format ontologies, etc.



allows them to build rich, interactive analysis systems. The core data structure in this endeavor is the representation of information as a network (implemented as a dictionary), and the ability to “chain” low-level data to high-level conceptual and analytical models via the dictionary-as-network representation².

This “chaining” of information is achieved via a set of **Analysis Meta Patterns** that have identical representation as a graph, but differing interpretations³. An Analysis Pattern encapsulates some domain knowledge (and as such is the data analogue of Design Patterns), as a set of relationships amongst objects. The domain could be accounting, forest inventory, medical observations, etc.. The relationships could be that between a withdrawal and a balance (accounting), a stand and a plot (forest inventory), a set of symptoms and a named disease (medical observations). An Analysis Meta Pattern, takes the Analysis Pattern concept one step further, and identifies a generic set of templates (the Meta Pattern) which can support a wide range of Analysis Pattern (i.e. filling in the template with specifics of a domain results in a particular Analysis Pattern) XAYA defines a simple Analysis Meta Pattern template format based on graphs, that allows support for a wide range of Analysis Patterns. For example, imagine a set of related YAML style text files, representing graphs as dictionaries. One dictionary/graph could be of data (keys are “rows”, values are list of tuples). Another dictionary/graph could be of a particular database schema (keys are tables, values are list of child tables i.e. those with a one-to-many relation to the key table). Another dictionary/graph could be of Objects (keys are Object Names, values are list of attributes), another could be a list of rules (keys are Objects, values are the list of rules applied to the object, each “rule” being a function name), and so on. The “syntax” of the data is uniform, while its “semantics” varies. Each dictionary/graph has some information, related to another dictionary/graph. Functions that allow one to Define or Find paths through these dictionaries (think of it as linking graphs and filtering graphs) allow one to make inferences using information at multiple levels, from low level observations, various data sources, particular rule-sets. A more extended example -- based on the forest inventory domain -- is provided on the support website.

Via the common representation, it is possible to provide a uniform user interface for querying and reasoning with data, that is abstracted above the data sources themselves, and their source-specific query and data access mechanisms. The language provides the building blocks for a data-centric development style that can be used to construct applications as diverse as OLAP, semantic inference systems, neural and Bayesian nets. However, a key goal is to keep the “core language” minimal, so a non-programmer can use it as an extended tutorial on Python development, and where additional functionality take advantage of key Python libraries (build as little new code as possible), and provide an introduction to Python’s support for distributed information systems.

² see <http://www.python.org/doc/essays/graphs.html> for the simplest implementation of a graph data structure as a dictionary. There are several good implementations of Graph data structures in Python, and a goal is to incorporate the best of the existing implementations, while focusing on developing the information modeling system.

³ This idea is a generalization of Martin Fowler’s work on Analysis Patterns, which encapsulate domain knowledge into classes. XAYA begins with a set of generalized Analysis Patterns that can be used to model a wide range of systems. See “Analysis Patterns: Reusable Object Models” by Martin Fowler.



b. What precisely the funds are needed for:

The funds are needed to support software development time for a six month period, with myself committed half time to XAYA development. Funds cover development time only, and may include some consulting with local Python experts.

c. A delivery plan, indicating what deliverables will be provided at what time:

The project will be delivered in 6 modules, each of which introduces new functionality to XAYA. Each module will have three components, all available from an interactive web-site (wiki): (i) code and unit-tests and usage examples via doc-tests, (ii) an extended illustrative tutorial on usage developing a declarative (“what not how”) programming style within the XAYA language; (iii) a set of development notes detailing an experimental development style⁴. The project is equally weighted towards code creation and “thinking with data” as programming concept extension, emphasizing an exploratory programming approach. Just as physicists modeled via calculus. We are modeling via simulating information relationships and querying. Consider working in the Python interpreter as a series of mini-experiments, and the final code as the documentation of the “successful” experiments, and tests as the lessons learned from the process of experimentation itself .

The key module is XAYAcCore – which defines the language and is only dependant on the Python Standard Library. Other modules would extend the language by leveraging other Python libraries to increase access to data, analytical capabilities, and inference capabilities, GUI support. The goal is to leverage the large body of existing code that supports various forms of information querying, and provide a common wrapper on all this functionality, via the XAYAcCore mini-language.

1. Module 1: XAYAcCore – develops a set of Information Analysis Meta Pattern Templates as graphs, based on a relational model of information, that has been expanded to process node-sets (i.e. querying graphs, rather than tables). Core functionality requires only the Python standard library and is implemented as a set of functions, a base class GraphDict from which each Analysis Meta Pattern template inherits, and with XML-RPC support for the functional version. [Reference will be made to the various existing Graph implementations in Python such as the graphlib module; also leverages YAML format text files]
2. Module 2: XAYAcCoreDB – adds in database support to XAYAcCore for the following common databases: Postgres, MySQL. Other databases accessed via support for ODBC. [Will leverage several DB modules such as psycopg, MySQLdb, odbc modules]
3. Module 3: XAYAcCoreNumeric – adds in numeric processing support to XAYAcCore. This focuses on support for basic statistics, and with an emphasis on robust and resistant exploratory data analysis methods (i.e. methods that work well with “complex”, “messy” data). [Will leverage the Numeric module]
4. Module 4: XAYAcCoreXML – adds in support for querying XML data (via XPath), using DTD and XML-Schema. [Will leverage the XML support modules built into Python Standard Library and PyXML module].

⁴ A similar approach was used in “A Diary of Information Theory” by Alfred Renyi which under the guise of a student learning information theory for the first time, provided a gentle, extended, but ultimately rigorous example of the development of information theory.

5. Module 5: XAYAcorerdf – adds in support for semantic web concepts via RDF and Ontology query support. [Will leverage GraphPath module, and its support for RDF and queries of ontologies]
6. Module 6: XAYAcoregui – a minimal GUI framework for XAYA that is based on tying the mini-language to a GUI. The goal is to provide a very basic API via which information can be visualized as Graphs, and the XAYA Analysis Patterns and their inter-relations can be accessed and manipulated visually. The minimal GUI provides a basic framework other developers can extend to create specific applications for a given domain (e.g. a Forest Estate Model, A Derivatives Pricing System, etc, An Epidemiological model, etc). The GUI will support a visual query interface, and a wizard based approach to filling in “Archetypes” (i.e. allowing a domain expert to specify a model by filling in template Analysis Patterns). [Leverages wxPython for GUI, and Pydot support for GraphViz graph visualization]

. The proposed delivery schedule is:

Series 1: December 15th 2004, January 15th 2005, February 15th 2005. Defines the core language functionality, support a range of databases, and provide the basic numeric processing capability

Series 2: June 15th 2005, July 15th 2005, August 30th 2005. Extend the support to Internet oriented data sources, extend query/inference capabilities and provide a minimal GUI framework for rapid application development The months of March and April are usually ones of contract deadlines, which is why there is the gap in delivery dates from February to June.

I have been using the SCRUM⁵ agile methodology to manage development teams for the last two years, and will apply it to this project. Each module will be defined by a Sprint Backlog and Sprint Signature of ongoing work. The Sprint Backlog and Sprint Signature will be on the project website so that progress is transparent.

d. **A payment plan, indicating what payments should be made at what time.**

The requested funding level is \$18,000 US, broken into 6 installments of \$3000 US, tied to module delivery. Payment will be after the delivery of each module. Modules are considered delivered, when they have been posted to the project website

e. **An extension and community development plan**

Extension will be via the website. The structure of the 6 modules is such, that they can outline an online book documenting the building of XAYA. This “wiki-book” will illustrate how one can rapidly build support for a declarative programming style in Python that can provide a key metaphor for leveraging the rich, distributed information networks that the internet is making possible. Building the documentation as a wiki-book allows for all initial content posted to benefit from the clarification and critique of readers. The project will also be put on Source Forge, with a link pointing back at the project website.

The project website will be the point of community development. The domain name XAYA.org has been registered, and as the project proceeds the website will be transferred from its current location within the

⁵ See “Agile Software Development with Scrum” by Ken Schwaber and Mike Beedle.



Harmeny domain, to its own location. At that point we could incorporate Plone and MoinMoin into the XAYA site to make it fully "Python Powered". Extension could also include talks on the project to the local Python user's group (VanPyZ – Vancouver Python and Zope User's Group). and working with any interested local folks on developing the system. The VanPyZ group has a number of folks skilled in DB and XML technologies who will be good sources of advice and critique.

The wiki-book format will hopefully foster on the website an attitude and conventions promoting community development via: (i) give people a framework to participate, (ii) make it easy to participate and (iii) make it rewarding to participate. Elements to promote such community development are: (i) rapidly iterate the code(weekly releases), (ii) have clear unit-tests, usage examples via doc tests, (iii) tutorials (iv) a clear interface and API for all functions/methods, so folks can re-implement any function they wish, the most parsimonious implementation becomes the "reference version" (v) minimal side-effects, which comes naturally to both functional and declarative programming styles (vi) the ability for people to go off on "tangents" on the XAYA site (alternative implementations, proto-applications for a given domain), allowing for greater diversity (vii) a natural framework for people to build extended functionality and new applications. By focusing on rapidly iterating new sets of minimal functionality, hopefully people will be inspired to contribute what they think is missing, and know their contribution is likely to be accepted as long as it matches the conventions, testing standards and API. An idea might begin as a sketch on the XAYA site, then gather its own group and split off to a linked site of its own. Speciate, speciate, speciate! (I was once upon a time a botanist).

f. About the proponent, Mishtu Banerjee

I am a principal in Harmeny Systems Ltd (www.harmeny.com)⁶ which focuses on developing information systems in the resource industry. Much of the impetus for this project originates in my experience in information system design, and the recognition of commonalities in Analysis Patterns that repeatedly occur across knowledge domains(i.e. the meta-patterns). XAYA is designed to provide a common methodology for capturing such Analysis Patterns via the Analysis Meta Pattern Template. The basic concepts of treating information systems as Graphs has been tested in the context of some of our company's previously developed software, LifeLine⁷. Many of the ideas also originate in extensive data modeling done in the resource industry working with cross-disciplinary teams, and the realization that it's much easier to teach people how to "visualize graphs and paths through information" than it is to teach them how to do things like normalize a database or develop a multi-layer query in some specific query language. As an information system designer and data analyst who commonly works with planners and managers and other business analysts I recognize the need for a framework to bridge the conceptual and data centric details of "What" with the technological details of "How" in terms of integrating the wide range of information available to any organization into useful models.

XAYA also stems from my long term interest in logic and information theory and study of the common basis for developing query systems whether they be in databases, in AI expert shells, in XML, in parsers – in particular

⁶ Brief bio at: <http://www.harmeny.com/about.html>

⁷ Manual for Access Version of LifeLine at: <http://www.harmeny.com/support.html> Manual for Java Version of LifeLine at: <http://www.harmeny.com/twiki/bin/view/Main/LifeLine>



influenced by discussions with Dr. Laks Lakshmanan's Database Lab at University of BC and their papers on querying and semantics, for XML and heterogeneous data sources⁸.

I am a new user of Python, first playing with it briefly several years ago in the context of testing out some ideas on fuzzy logic inference systems. I returned to it recently (this spring), as the ideas that led to XAYA became clearer to me, and I needed a programming system to "experiment in". The VanPyZ organization monthly meetings (<http://www.vanpyz.org/>), and their conference this year were instrumental to supporting my interest in Python and desire to try and develop a fairly challenging piece of software in it.

So, I would come to this project experienced on the mathematical and logic concepts underlying the idea of a general declarative language, with a strong background in managing software projects in an industrial context, but new to Python itself, and hopefully able to convey the excitement of "first discovery" in Python.

Essentially the request for funding is to allow me the freedom to focus on XAYA development for an extended period of time, and use that time to create a body of code and a community website/wiki-book that provides an extended example of how to incorporate a declarative programming style in Python while illustrating the use of Analysis Patterns in a format that allows them to be used by Non-Programmers. While not quite "Computer Programming for Everybody"⁹, I am hoping for "Programming for anyone who can learn to Think With Data".

⁸ List of publications at: <http://www.cs.ubc.ca/~laks/papersByTopic.html> ; key ideas on extending Relational concepts to Graphs at: <http://www.cs.ubc.ca/~laks/tax-dbpl01-cr.pdf> ; "semantic markup" to integrate heterogeneous data sources at <ftp://ftp.cs.ubc.ca/pub/local/laks/papers/webdb03.CR.pdf>

⁹ from <http://www.python.org/doc/essays/cp4e.html>

