

Advanced Information and Knowledge Processing

Series Editors

Professor Lakhmi Jain

Lakhmi.jain@unisa.edu.au

Professor Xindong Wu

xwu@cems.uvm.edu

For other titles published in this series, go to
www.springer.com/series/4738

Sergei V. Chekanov

Scientific Data Analysis using Jython Scripting and Java

 Springer

Dr. Sergei V. Chekanov
Argonne National Laboratory (ANL)
9700 S. Cass Ave
Argonne
60439 IL, USA
chakanau@hep.anl.gov

AI&KP ISSN 1610-3947
ISBN 978-1-84996-286-5 e-ISBN 978-1-84996-287-2
DOI 10.1007/978-1-84996-287-2
Springer London Dordrecht Heidelberg New York

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Control Number: 2010930974

© Springer-Verlag London Limited 2010

Apart from any fair dealing for the purposes of research or private study, or criticism or review, as permitted under the Copyright, Designs and Patents Act 1988, this publication may only be reproduced, stored or transmitted, in any form or by any means, with the prior permission in writing of the publishers, or in the case of reprographic reproduction in accordance with the terms of licenses issued by the Copyright Licensing Agency. Enquiries concerning reproduction outside those terms should be sent to the publishers.

The use of registered names, trademarks, etc., in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant laws and regulations and therefore free for general use.

The publisher makes no representation, express or implied, with regard to the accuracy of the information contained in this book and cannot accept any legal responsibility or liability for any errors or omissions that may be made.

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

This book is dedicated to my family

Preface

Over the course of the past twenty years I have learned many things relevant to this book while working in high-energy physics. As everyone in this field in the yearly to mid-90s, I was analyzing experimental data collected by particle colliders using the FORTRAN programming language. Then, gradually, I moved to C++ coding following the general trend at that time. I was not too satisfied with this transition: C++ looked overly complicated and C++ source codes were difficult to understand. With C++, we were significantly constrained by particular aspects of computer hardware and operating system (Linux and Unix) on which the source codes were compiled and linked against existing libraries. Thus, to bring the analysis environment outside the high-energy community to the Windows platform, used by most people, was almost impossible.

I began serious development of ideas that eventually led to the jHepWork Java analysis environment in 2004, when I was struck by the simplicity and by the power of the Java Analysis Studio (JAS) program developed at the SLAC National Accelerator Laboratory (USA). One could run it even on the Windows platform, which was incredible for high-energy physics applications; We never really used Windows at that time, since high-energy physics community had wholly embraced Unix and Linux as the platform of choice, together with its build-in GNU C++ and FORTRAN compilers. More importantly, JAS running on Windows had exactly the same interface and functionality as for Unix and Linux! It was a few months after that I made the decision to focus on a simplified version of this Java framework which, I thought, should benefit from Java scripting, will be simpler and more intuitive. Thus, it should be better suited for general public use. I have called it “jHepWork” (“j” means Java, “HEP” is the abbreviation for high-energy physics, and “work” means a sedentary lifestyle in front of a computer monitor).

Indeed, I was able to simplify the language and semantic of the JAS analysis environment by utilizing more appropriate short names for classes and methods, which are more suited for scripting languages. The entire project had grown tremendously after inclusion of many new GNU-licensed packages and extending the functionality of JAS in many areas, such as 3D graphics, serialized I/O and numerous numerical packages. At present, jHepWork covers an impressive list of Java-written packages

ranged from basic mathematical functions to neural networks and cellular automation. And, eventually, a little of JAS has left inside jHepWork! One important thing, however, has remained: As JAS, jHepWork was still an open-source software that can be downloaded freely from the Web.

For this project, Python was chosen as the main programming language because it is elegant and easy to learn. It is a great language for teaching scientific computation. For developers, this is an ideal language for fast prototyping and debugging. However, since the whole project was written in Java, it is Jython (Python implemented in Java) that was eventually chosen for the jHepWork project.

This book is intended for general audience, for those who use computing power to make sense of surrounding us data. This book is a good source of knowledge on data analysis for students and professionals of all disciplines. Especially, this book is for scientists and engineers, and everyone who devoted themselves to the quest of where we find ourselves in the Universe and what we find ourselves made of.

This book is also for those who study financial market; I hope it will be useful for them because the methods discussed in this book are undoubtedly common to any scientific research. However, I have to admit that this book may have little interest for a commercial use since financial-market analysts, unlike researches in basic scientific fields, could afford costly commercial products.

This book is about how to understand experimental data, how to reduce complexity of data, derive some meaningful conclusions and, finally, how to present results using Java graphical packages. It concentrates on computational aspects of these topics: as you will see, due to the simplicity of Python, one could catch ideas of many examples of this book just by looking at the code snippets without even explaining them in words. This book is also about how to simulate more or less realistic data samples which can mimic real situations. Such simulated data are used in this book in order to give simple and intuitive examples of data analysis techniques using Java scripting.

In this book I did not go deep inside of particular statistical or physics topics, since the aim was to give concrete numeral receipts and examples using Jython scripting language interfaced with Java numerical packages. My aim was also to give an introduction to many data-analysis subjects with sample code snippets based on Jython and jHepWork Java libraries. In cases when I could not cover the subject in detail, a sufficient number of relevant references was given, so the reader can easily find necessary information for each chapter using external sources.

Thus, this book presents practical approaches for data analysis, focusing on programming techniques. Each chapter describes the conceptual and methodological underpinning for statistical tools and their implementation in Java, covering essentially all aspects of data analysis, from simple multidimensional arrays and histograms to clustering analysis, curve fitting, metadata and neural networks. This book includes a comprehensive coverage of various numerical and graphical packages currently implemented in Java that are part of the jHepWork project.

The book was written by the primary developer of the software, and aimed to present a reliable and complete source of reference which lays the foundation for future data-analysis applications using Java scripting. The book includes more than

200 code snippets which are directly runnable and used to produce all graphical plots given in the text. A detailed description and several real-life data-analysis examples which develop a genuine feeling for data analysis techniques and their programming implementation are given in the last chapter of this book.

Finally, I am almost convinced myself that this book is self-contained and does not depend on knowledge of any computing package, Java, Python or Jython (although knowledge of Python and Java is desirable for professionals).

Chicago-Hamburg-Minsk

Sergei V. Chekanov

Acknowledgements

Several acknowledgements are in order. Much of this project grew out of fruitful collaboration with many of my colleagues who devoted themselves to high energy physics.

My scientific career in experimental high-energy physics was most influenced by Prof. Dr. V.I. Kuvshinov and Prof. Dr. E.W. Kittel, who were my supervisors almost fifteen years ago. I've learned experimental computation in the early 90s from Dr. L.F. Babichev and Dr. W.J. Metzger. I've learned experimental physics and its computational aspects from Dr. M. Derrick, Prof. Dr. E. Lohrmann, Dr. J. Repond, Dr. R. Yoshida, Dr. S. Magill, Dr. C. Glasman, Prof. Dr. J. Terron, Dr. J. Proudfoot, Dr. A. Vanyashin and many others.

The author is grateful to many authors writing free scientific software for their dedication to science and open-source analysis tools. I would like to thank many of my colleagues for checking and debugging the jHepWork package, especially J. Dale, E. May, L. Lee, T. Johnson, P. Di Stefano and many others.

I would like to thank my parents and sister for their support, guidance, and love. Not least, I would like to express my eternal gratitude for my dear wife and children for their love and patience to a husband and father who wrote this book at home and thus was only half (mentally) present after coming from his work. Without their patience and understanding, this book would not have been possible.

Contents

Introduction	1
Introduction to Data Analysis and Why This Book Is Special . . .	1
Who Is This Book for	2
1 Jython, Java and jHepWork	3
1.1 Introduction	3
1.1.1 Books You May Read Before	4
1.1.2 Yes, It Is Pure Java	4
1.1.3 Some Warnings	5
1.1.4 Errors	7
1.2 Introduction to Scientific Computing	7
1.2.1 Book Examples and the Power of Jython	7
1.2.2 The History of jHepWork	8
1.2.3 Why Jython?	9
1.2.4 Differences with Other Data-analysis Packages	10
1.2.5 How Fast It Is?	11
1.2.6 Jython and CPython Versions	12
1.3 Installation	13
1.4 Introduction to the jHepWork IDE	14
1.4.1 Source Code Editor	15
1.4.2 jHepWork Java Libraries and Python Packages	15
1.4.3 Jython and Bean Shell Consoles	17
1.4.4 Accessing Methods of Instances	19
1.4.5 Editing Jython Scripts	19
1.4.6 Running Jython Scripts	19
1.4.7 Running a BeanShell Scripts	20
1.4.8 Compiling and Running Java Code	20
1.4.9 Working with Command-line Scripts	21
1.4.10 jHepWork Code Assist	21
1.4.11 Other Features	22

1.5	Third-party Packages and the License	23
1.5.1	Contributions and Third-party Packages	23
1.5.2	Disclaimer of Warranty	25
1.5.3	jHepWork License	25
	References	26
2	Introduction to Jython	27
2.1	Code Structure and Commentary	27
2.2	Quick Introduction to Jython Objects	28
2.2.1	Numbers as Objects	31
2.2.2	Formatted Output	32
2.2.3	Mathematical Functions	33
2.2.4	Complex Numbers	34
2.3	Strings as Objects	34
2.4	Import Statements	35
2.4.1	Executing Native Applications	36
2.5	Comparison Tests and Loops	37
2.5.1	The ‘if-else’ Statement	37
2.5.2	Loops. The “for” Statement	38
2.5.3	The ‘continue’ and ‘break’ Statements	39
2.5.4	Loops. The ‘while’ Statement	39
2.6	Collections	40
2.6.1	Lists	40
2.6.2	List Creation	41
2.6.3	Iteration over Elements	42
2.6.4	Removal of Duplicates	43
2.6.5	Tuples	45
2.6.6	Functional Programming. Operations with Lists	46
2.6.7	Dictionaries	48
2.7	Java Collections in Jython	50
2.7.1	List. An Ordered Collection	50
2.7.2	Set. A Collection Without Duplicate Elements	53
2.7.3	SortedSet. Sorted Unique Elements	54
2.7.4	Map. Mapping Keys to Values	55
2.7.5	Java Map with Sorted Elements	55
2.7.6	Real Life Example: Sorting and Removing Duplicates	56
2.8	Random Numbers	57
2.9	Time Module	58
2.9.1	Benchmarking	59
2.10	Python Functions and Modules	60
2.11	Python Classes	63
2.11.1	Initializing a Class	65
2.11.2	Classes Inherited from Other Classes	66
2.11.3	Java Classes in Jython	66
2.11.4	Topics Not Covered	67

2.12	Used Memory	67
2.13	Parallel Computing and Threads	67
2.14	Arrays in Jython	68
2.14.1	Array Conversion and Transformations	70
2.14.2	Performance Issues	70
2.15	Exceptions in Python	71
2.16	Input and Output	72
2.16.1	User Interaction	72
2.16.2	Reading and Writing Files	72
2.16.3	Input and Output for Arrays	74
2.16.4	Working with CSV Python Module	75
2.16.5	Saving Objects in a Serialized File	77
2.16.6	Storing Multiple Objects	77
2.16.7	Using Java for I/O	78
2.16.8	Reading Data from the Network	79
2.17	Real-life Example. Collecting Data Files	80
2.18	Using Java for GUI Programming	83
2.19	Concluding Remarks	84
	References	84
3	Mathematical Functions	85
3.1	Jython Functions	85
3.2	1D Functions in jHepWork	87
3.2.1	Details of Java Implementation	89
3.2.2	Integration and Differentiation	90
3.3	Plotting 1D Functions	91
3.3.1	Building a Graphical Canvas	92
3.3.2	Drawing 1D Functions	95
3.3.3	Plotting 1D Functions on Different Pads	97
3.3.4	Short Summary of HPlot Methods	98
3.3.5	Examples	98
3.4	2D Functions	100
3.4.1	Functions in Two Dimensions	100
3.4.2	Displaying 2D Functions on a Lego Plot	101
3.4.3	Using a Contour Plot	104
3.5	3D Functions	105
3.5.1	Functions in Three Dimensions	105
3.6	Functions in Many Dimensions	105
3.6.1	FND Functions	105
3.6.2	Drawing FND Functions	106
3.7	Custom Functions Defined by Jython Scripts	107
3.7.1	Custom Functions and Their Methods	107
3.7.2	Using External Libraries	110
3.7.3	Plotting Custom Functions	111
3.8	Parametric Surfaces in 3D	113

3.8.1	FPR Functions	113
3.8.2	3D Mathematical Objects	116
3.9	Symbolic Calculations	116
3.10	File Input and Output	119
	References	120
4	One-dimensional Data	121
4.1	One Dimensional Arrays	121
4.2	P0D Data Container	122
4.2.1	P0D Transformations	125
4.2.2	Analyzing P0D and Summary Statistics	126
4.2.3	Displaying P0D Data	128
4.3	Reading and Writing P0D Files	130
4.3.1	Serialization	131
4.3.2	XML Format	131
4.3.3	Dealing with Object Collections	133
5	Two-dimensional Data	135
5.1	Two Dimensional Data Structures	135
5.2	Two Dimensional Data with Errors	136
5.2.1	Viewing P1D Data	140
5.2.2	Plotting P1D Data	142
5.2.3	Contour Plots	144
5.3	Manipulations with P1D Data	145
5.3.1	Advanced P1D Operations	146
5.3.2	Weighted Average and Systematical Uncertainties	148
5.4	Reading and Writing P1D Data	151
5.4.1	Dealing with a Single P1D Container	151
5.4.2	Reading and Writing Collections	153
5.5	Real-life Example I: Henon Attractor	154
5.6	Real-life Example II. Weighted Average	155
	References	159
6	Multi-dimensional Data	161
6.1	P2D Data Container	161
6.1.1	Drawing P2D and HPlot3D Canvas	161
6.2	P3D Data Container	164
6.3	PND Data Container	166
6.3.1	Operations with PND Data	167
6.4	Input and Output	169
7	Arrays, Matrices and Linear Algebra	171
7.1	Jaida Data Containers	171
7.1.1	Jaida Clouds	172
7.2	jMathTools Arrays and Operations	174
7.2.1	1D Arrays and Operations	174

7.2.2	2D Arrays	176
7.3	Colt Data Containers	177
7.4	Statistical Analysis Using Jython	178
7.5	Matrix Packages	181
7.5.1	Basic Matrix Arithmetic	183
7.5.2	Elements of Linear Algebra	184
7.5.3	Jampack Matrix Computations and Complex Matrices	185
7.5.4	Jython Vector and Matrix Operations	186
7.5.5	Matrix Operations in SymPy	188
7.6	Lorentz Vector and Particle Representations	189
7.6.1	Three-vector and Lorentz Vector	189
7.6.2	Classes Representing Particles	191
	References	192
8	Histograms	193
8.1	One-dimensional Histogram	193
8.1.1	Probability Distribution and Probability Density	198
8.1.2	Histogram Characteristics	198
8.1.3	Histogram Initialization and Filling Methods	199
8.1.4	Accessing Histogram Values	201
8.1.5	Integration	201
8.1.6	Histogram Operations	203
8.1.7	Accessing Low-level Jaida Classes	204
8.1.8	Graphical Attributes	205
8.2	Histogram in 2D	205
8.2.1	Histogram Operations	207
8.2.2	Graphical Representation	209
8.3	Histograms in Jaida	212
8.4	Histogram in 3D	214
8.5	Profile Histograms	214
8.5.1	Profile Histograms in 1D	215
8.5.2	Profile Histograms in 2D	215
8.6	Histogram Input and Output	217
8.6.1	External Programs for Histograms	218
8.7	Real-life Example. Analyzing Histograms from Multiple Files	220
	References	221
9	Random Numbers and Statistical Samples	223
9.1	Random Numbers in Jython	223
9.2	Random Numbers in Java	225
9.3	Random Numbers from the Colt Package	226
9.4	Random Numbers from the jhplot.math Package	227
9.4.1	Apache Common Math Package	229
9.5	Random Sampling	229
9.5.1	Methods for 1D Arrays from jhplot.math	230

9.5.2	Methods for 2D Arrays from <code>jhplot.math</code>	232
9.6	Sampling Using the Colt Package	233
	References	233
10	Graphical Canvases	235
10.1	HPlot Canvas	236
10.2	Working with the HPlot Canvas	238
10.2.1	Find USER or NDC Coordinators	238
10.2.2	Zoom in to a Certain Region	238
10.2.3	How to Change Titles, Legends and Labels	238
10.2.4	Edit Style of Data Presentation	239
10.2.5	How to Modify the Global Margins	239
10.2.6	Saving Plots in XML Files	240
10.2.7	Reading Data	240
10.2.8	Cleaning the HPlot Canvas from Graphics	241
10.2.9	Axes	241
10.2.10	Summary of the HPlot Methods	242
10.2.11	Saving Drawings in an Image File	242
10.3	Labels and Keys	244
10.3.1	Simple Text Labels	244
10.3.2	Interactive Labels	245
10.3.3	Interactive Text Labels with Keys	246
10.4	Geometrical Primitives	248
10.5	Text Strings and Symbols	249
10.6	SHPlot Class. HPlot Canvas as a Singleton	249
10.7	Visualizing Interconnected Objects	251
10.8	Showing Charts	253
10.9	SPlot Class. A Simple Canvas	254
10.9.1	Henon Attractor Again	256
10.10	Canvas for Interactive Drawing	257
10.10.1	Drawing Diagrams	258
10.10.2	SHPlotJa Class	259
10.11	HPlot2D Canvas	260
10.12	3D Canvas	262
10.13	HPlot3D Canvas	263
10.13.1	HPlot3DP Canvas	263
10.13.2	3D Geometry Package	266
10.14	Combining Graphs with Java Swing GUI Components	267
10.15	Showing Streams of Data in Real Time	270
	References	271
11	Input and Output	273
11.1	Non-persistent Data. Memory-based Data	273
11.2	Serialization of Objects	274
11.3	Storing Data Persistently	276

11.3.1	Sequential Input and Output	276
11.3.2	GUI Browser for Serialized Objects	278
11.3.3	Saving Event Records Persistently	279
11.3.4	Buffer Size for I/O Intensive Operations	280
11.3.5	Input and Output to XML Files	281
11.3.6	Non-sequential Input and Output	282
11.4	Compressed PFile Format	283
11.4.1	Browser Dialog for PFile Files	286
11.5	Reading ROOT and AIDA Files	287
11.5.1	Reading ROOT Histograms	287
11.5.2	Reading ROOT Trees	288
11.5.3	Plotting ROOT or AIDA Objects Using jHepWork IDE	290
11.6	Working with Relational SQL Databases	291
11.6.1	Creating a SQL Database	292
11.6.2	Working with a Database	293
11.6.3	Creating a Read-only Compact Database	294
11.7	Reading and Writing CSV Files	295
11.7.1	Reading CSV Files	295
11.7.2	Writing CSV File	296
11.8	Google's Protocol Buffer Format	297
11.8.1	Prototyping Data Records	298
11.8.2	Dealing with Data Using Java	299
11.8.3	Switching to Jython	302
11.8.4	Adding New Data Records	303
11.8.5	Using C++ for I/O in the Protocol Buffers Format	303
11.8.6	Some Remarks	306
11.9	EFile Data Output	306
11.10	Reading DIF Files	309
11.11	Summary	310
11.11.1	Dealing with Single Objects	310
11.11.2	Dealing with Collections of Objects	310
	References	311
12	Miscellaneous Analysis Issues Using jHepWork	313
12.1	Accessing Third-party Libraries	313
12.1.1	Extracting Data Points from a Figure	313
12.1.2	Cellular Automaton	314
12.2	Downloading Files from the Web	315
12.3	Macro Files for jHepWork Editor	315
12.4	Data Output to Tables and Spreadsheets	316
12.4.1	Showing Data in a Sortable Table	316
12.4.2	Spreadsheet Support	318
12.5	Accessing External Java and Jython Libraries	318
12.6	Working with a jHepWork Project	319
12.6.1	Pure Jython Project	319

12.6.2	Pure Java Project	320
12.6.3	Mixing Jython with Java	320
12.7	Working with Images	321
12.7.1	Saving Plots in External Image File	321
12.7.2	View an Image. IView Class	321
12.7.3	Analyzing and Editing Images	321
12.8	Complex Numbers	322
12.9	Building List of Files	322
12.10	Reading Configuration Files	323
12.10.1	Configuration Files Using Jython	323
12.10.2	Reading Configuration Files Using Java	325
12.11	Jython Scripting with jHepWork	326
12.11.1	Jython Operations with Data Holders	328
12.12	Unwrapping Jython Code. Back to Java	329
12.13	Embedding jHepWork in Applets	331
	References	334
13	Data Clustering	335
13.1	Data Clustering. Real-life Example	335
13.1.1	Preparing a Data Sample	336
13.1.2	Clustering Analysis	338
13.1.3	Interactive Clustering with JMinHEP	342
	References	342
14	Linear Regression and Curve Fitting	343
14.1	Linear Regression	343
14.1.1	Data Set	343
14.1.2	Analyzing the Data Set	344
14.2	Curve Fitting of Data	346
14.2.1	Preparing a Fit	348
14.2.2	Creating a Fit Function	349
14.3	Displaying a Fit Function	354
14.3.1	Making a Fit	354
14.4	Real-life Example. Signal Plus Background	357
14.4.1	Preparing a Data Sample	357
14.4.2	Performing Curve Fitting	357
14.4.3	Fitting Multiple Peaks	359
14.4.4	Fitting Histograms in 3D	362
14.5	Interactive Fit	363
	References	365
15	Neural Networks	367
15.1	Introduction	367
15.1.1	Generating a Data Sample	368
15.1.2	Data Preparation	369

15.1.3	Building a Neural Net	371
15.1.4	Training and Verifying	373
15.2	Bayesian Networks	375
15.3	Self-organizing Map	376
15.3.1	Non-interactive BSOM	378
15.4	Neural Network Using Python Libraries	379
	References	382
16	Steps in Data Analysis	383
16.1	Major Analysis Steps	383
16.2	Real Life Example. Analyzing a Gene Catalog	385
16.2.1	Data Transformation	386
16.2.2	Data Skimming	386
16.2.3	Data Slimming	387
16.2.4	Data Sorting	387
16.2.5	Removing Duplicate Records	389
16.2.6	Sorting and Removing Duplicate Records Using Java	390
16.3	Using Metadata for Data Mining	391
16.3.1	Analyzing Data Using Built-in Metadata File	392
16.3.2	Using an External Metadata File	395
16.4	Multiprocessor Programming	396
16.4.1	Reading Data in Parallel	397
16.4.2	Processing a Single Input File in Parallel	399
16.4.3	Numerical Computations Using Multiple Cores	401
16.5	Data Consistency and Security. MD5 Class	403
16.5.1	MD5 Fingerprint at Runtime	403
16.5.2	Fingerprinting Files	404
	References	405
17	Real-life Examples	407
17.1	Measuring Single-particle Densities	407
17.1.1	Preparing a Data Sample	408
17.1.2	Analyzing Data	409
17.2	Many-particle Densities, Fluctuations and Correlations	411
17.2.1	Building a Data Sample for Analysis	411
17.2.2	Analyzing the Data	415
17.2.3	Reading the Data and Plotting Multiplicities	416
17.3	Analyzing Macro Data: Nearby Galaxies	420
17.4	Analyzing Micro Data: Elementary Particles	423
17.5	A Monte Carlo Simulation of Particle Decays	425
17.6	Measuring the Speed of Light Using the Internet	428
17.6.1	Getting Host Names in Each Continent	429
17.6.2	Checking Response from Servers	430
17.6.3	Creating Histograms with the Response Time	430

17.6.4 Final Measurements	431
References	433
Index of Examples	435
Index	437

Conventions and Acronyms

In this book, we will use the following typographical convention: A box with a code inside usually means interactive Jython commands typed in the Jython shell. All such commands start with the symbol `>>>`, which is the usual invitation in Python to type a command. This is shown in the example below:

```
>>> print 'Hello, jHepWork'
```

Working interactively with the Jython prompt has the drawback that it is impossible to save typed commands. In most cases, the code snippets are not so short, although they are still much shorter than in any other programming language. Therefore, it is desirable to save the typed code in a file for further modification and execution. In this case, we use Jython macro files, i.e. we write a code using the jHepWork (or any other) editor, save it in a file with the extension “.py”, and run it using the keyboard shortcut [F8] or the button “run” from the jHepWork tool bar menu. In the book, such code examples are also shown inside the box, but code lines do not start with the Jython invitation symbol `>>>`. In such situations, the example codes will be shown as:

```
print 'Hello, jHepWork'
```

If a code snippet should be used as a Jython module for inclusion into other programs, then we should write our program inside a file. A Python code always imports an external module using its file name. Since the file names are important, we always indicate which exactly file name should be used on the top of the box with a code. For example, if a program code is considered as a module to be imported by another code example, we will show it as:

```
Module 'hello.py'  
print 'Hello, jHepWork'
```

with the box title indicating the file name. For instance, we call the module above as:

```
>>> import hello
Hello, jHepWork
```

when using the Jython prompt (recall the `>>>` symbol!). The code imports the file `'hello.py'` and executes it, printing the string `'Hello, jHepWork'`. In other cases, one can use any file names for the code snippets.

We use `typewriter` font for Jython and Java classes and methods. For file names and directories, we also use the same font style after adding additional parentheses.

We remind also that the directory name separators are backward slashes for Windows, and slashes for Linux and Mac computers. In this book, we use the latter convention. For example, the directory with examples will be shown as:

```
... /macro/examples/
```

while for Windows computers, the same directory should be shown as:

```
C:... \macro\examples\
```

The dots in this example are used to indicate the jHepWork installation directory.

This is all. We will try to avoid using abbreviations. When we use abbreviations, we will explain their meaning directly in the text.