Python库参考手册(Python Library Reference)

Release 2.3.3

Guido van Rossum
Fred L. Drake, Jr., editor
翻译团队: 见文末名单

译文最后修改: 2004**年**3**月**22**日**

PythonLabs
Email: docs@python.org

Python is an extensible, interpreted, object-oriented programming language. It supports a wide range of applications, from simple text processing scripts to interactive Web browsers.

Python **是一种可扩展的**, **即译式**, **面向对象规格的编程语言**. **它能应用在极广泛的地方**, **从简单的文字处理工作到交互式的网页浏览器**.

While the *Python Reference Manual* describes the exact syntax and semantics of the language, it does not describe the standard library that is distributed with the language, and which greatly enhances its immediate usability. This library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs.

Python ***语言参考手册*** **中精确地描述了**Python **语言的句法及语义**. **然而语言参考手册中没有提到**Python **所附带功能强大的标准库**. **这个函式库大大地增强了**Python **的实用性**. **其中包括**C **写的内建模组**, **提供介面让程式进行操作系统层次的工作**, **例如档案的输出输入**; **同时也有以**Python **语言本身编写的模组**, **为实际编程时常遇的问题提供标准解决方案**. **这类模组有的经过特别设计以便**Python **程式在跨平台的情况下运行无误**.

This library reference manual documents Python's standard library, as well as many optional library modules (which may or may not be available, depending on whether the underlying platform supports them and on the configuration choices made at compile time). It also documents the standard types of the language and its built-in functions and exceptions, many of which are not or incompletely documented in the Reference Manual.

本参考手册罗列并说明了Python **标准库的各种功能**, **以及许多非核心的模组**(**按不同的操作系统和编译时的设置而定**, **不是每台机上的**Python **都能用这些模组**.) **本手册同时记载了**Python **语言所有的标准数据类型**, **内建函数**, **异常类**, **这些在参考手册中被忽略了或只是扼要的提过一下**.

This manual assumes basic knowledge about the Python language. For an informal introduction to Python, see the *Python Tutorial*; the *Python Reference Manual* remains the highest authority on syntactic and semantic questions. Finally, the manual entitled *Extending and Embedding the Python Interpreter* describes how to add new extensions to Python and how to embed it in other applications.

本手册的读者要对Python **有基本的认识**. **初学者应该从***Python* ***指南*** **开始**. **至于***Python* ***语言参考手册*** **则是该语言的语法和语义问题上的权威阐释**. **最后*****扩展或嵌入****Python* ***解释器*** **一文解说了如何在**Python **中加入新的扩展模组**; **以及怎样把**Python **解释器嵌入到其他的应用程式中**.

目录

第一章

内建对象Built-In Objects

Names for built-in exceptions and functions and a number of constants are found in a separate symbol table. This table is searched last when the interpreter looks up the meaning of a name, so local and global user-defined names can override built-in names. Built-in types are described together here for easy reference.[1]

内建的异常、函数和一组常量的名字都能在一个独立的符号表中找到。当解释器在查找一个名字的意义的时候，这个表格是被最后查找的，所以局部的和全局的用户定义的名字可以覆盖内建的名字。所有的内建类型都统一在本章描述，作为一个简单的参考。[2]

The tables in this chapter document the priorities of operators by listing them in order of ascending priority (within a table) and grouping operators that have the same priority in the same box. Binary operators of the same priority group from left to right. (Unary operators group from right to left, but there you have no real choice.) See chapter 5 of the *Python Reference Manual* for the complete picture on operator priorities.

在本章的表格中，对于操作符的优先级，以它们在表格中列出的顺序，升序排列，(在同一个表格中) 并且把相同优先级的操作符组织在同一格当中。同一优先级的二元操作符已从左到右的顺序结合。(一元操作符从右到左结合，但事实上，你并没有选择的余地。) 对于操作符优先级的一个整体印象参见*Python参考手册*第5章。

[1]Most descriptions sorely lack explanations of the exceptions that may be raised — this will be fixed in a future version of this manual.
[2]**这儿的绝大多数描述都非常缺乏对它们所能抛出的异常的解释— 这将会在本手册的未来版本得到修正。**

第二章

Python运行时服务Python Runtime Services

The modules described in this chapter provide a wide range of services related to the Python interpreter and its interaction with its environment. Here’s an overview:

本章所讲到的这些模块，对于与 Python**解释器相关的，以及解释器和它运行环境之间的交互，提供了丰富的服务。以下是一个综述：**

`__main__` **顶层脚本运行所在的环境**The environment where the top-level script is run.

2.1 `__main__` — 顶层脚本的运行环境Top-level script environment

This module represents the (otherwise anonymous) scope in which the interpreter’s main program executes — commands read either from standard input, from a script file, or from an interactive prompt. It is this environment in which the idiomatic “conditional script” stanza causes a script to run:

这个模块描述了(**否则就是匿名的**)**解释器主程序运行命令的作用域**– **这些命令包括读取自标准输入的**, **脚本文件的**, **或者交互模式提示符的**. **一般我们会用以下的**“**条件脚本**”**块来使一个脚本运行**, **这个环境就是**__main__ .

```
if __name__ == "__main__":
    main()
```

第三章

其他各类服务Miscellaneous Services

The modules described in this chapter provide miscellaneous services that are available in all Python versions. Here's an overview:

本章所包含的为所有版本上的Python **皆有的各杂类模组**. **以下为概览**:

第四章

普遍的操作系统服务Generic Operating System Services

The modules described in this chapter provide interfaces to operating system features that are available on (almost) all operating systems, such as files and a clock. The interfaces are generally modeled after the UNIX or C interfaces, but they are available on most other systems as well. Here's an overview:

本章描述的模组及为存取操作系统提供服务介面。此类服务多为（基本上）所有操作平台所共有的，如档案的读写及时钟值。介面的格式是仿照UNIX **或**C **的风格来建立的，但同时能使用在多数其他的系统上。**

第五章

可选配的操作系统服务Optional Operating System Services

The modules described in this chapter provide interfaces to operating system features that are available on selected operating systems only. The interfaces are generally modeled after the UNIX or C interfaces but they are available on some other systems as well (e.g. Windows or NT). Here's an overview:

本章描述的模组提供介面以存取只有某些特定操作系统才有的功能。介面设计以UNIX **或**C **风格为蓝本**, **但它们也能在其他如**Windows, NT **等平台上使用。以下是概述**:

第六章

互联网通讯协议及其支援Internet Protocols and Support

The modules described in this chapter implement Internet protocols and support for related technology. They are all implemented in Python. Most of these modules require the presence of the system-dependent module `socket`, which is currently supported on most popular platforms. Here is an overview:

本课描述的模组是对互联网通讯协议的实现, 以及和其相关技术的支援。所有这些模组都是以Python 来实现的。大部份此类模组要用到一个依赖於操作系统的`socket`模组，此模组在基本存在於大多数流行的平台中。以下是本章的概述：

第七章

互联网资料处理Internet Data Handling

This chapter describes modules which support handling data formats commonly used on the Internet.

本章描述的模组帮助我们处理在互联网上通常会遇到的资料格式。

`base64`	**使用**MIME Base64**来编码和解码文件**Encode and decode files using the MIME base64 data.
`xml.dom`	Document Object Model API for Python **文档对象模式**.

7.1 `base64` — 编码和解码MIME Base64数据Encode and decode MIME base64 data

This module performs base64 encoding and decoding of arbitrary binary strings into text strings that can be safely sent by email or included as part of an HTTP POST request. The encoding scheme is defined in RFC 1521 (*MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of Internet Message Bodies*, section 5.2, “Base64 Content-Transfer-Encoding”) and is used for MIME email and various other Internet-related applications; it is not the same as the output produced by the **uuencode** program. For example, the string `'www.python.org'` is encoded as the string `'d3d3LnB5dGhvbi5vcmc=\n'`.

这个模块执行任意二进制串到文本串的Base64**编码和解码，从而该数据能够被安全的由**email**传输或者包含并且作为**HTTP POST**请求的一部分。编码的模式在**RFC 1521 (*MIME(Multipurpose Internet Mail Extensions)* ***第一部分：****Mechanisms for Specifying and Describing the Format of Internet Message Bodies*，5.2**节，**”Base64 Base64 Content-Transfer-Encoding”) **中定义，并且在**MIME**邮件以及各种其它**Internet**相关的应用程序中使用；它不同于****uuencode** **程序的输出。例如，字符串**`'www.python.org'` **编码为字符串**`'d3d3LnB5dGhvbi5vcmc=\n'`**。**

`decode`(*input, output*)

Decode the contents of the *input* file and write the resulting binary data to the *output* file. *input* and *output* must either be file objects or objects that mimic the file object interface. *input* will be read until *input*`.read()` returns an empty string.

`decode`(*input, output*)

解码*input* **文件的内容，输出结果二进制数据到***output* **文件。***input* **和***output* **必须都为文件对象或模拟文件对象接口的对象。***input* **将一直读取，直至***input*`.read()` **返回一个空串。**

`decodestring`(*s*)

Decode the string *s*, which must contain one or more lines of base64 encoded data, and return a string containing the resulting binary data.

`decodestring`(*s*)

解码字符串*s***，必须包含一行或多行**Base64**编码数据，返回一个包含结果二进制数据的字符串。**

`encode`(*input, output*)

Encode the contents of the *input* file and write the resulting base64 encoded data to the *output* file. *input* and *output* must either be file objects or objects that mimic the file object interface. *input* will be read until *input*`.read()` returns an empty string. `encode()` returns the encoded data plus a trailing newline character ('\n').

encode(*input, output*)
编码*input* 文件的内容，输出结果Base64编码数据到*output* 文件。*input* 和*output* 必须都为文件对象或模拟文件对象接口的对象。*input* 将一直读取，直至*input*.read() 返回一个空串。encode() 返回编码后的数据外加一个结尾换行符('\n')。

encodestring(*s*)
Encode the string *s*, which can contain arbitrary binary data, and return a string containing one or more lines of base64-encoded data. encodestring() returns a string containing one or more lines of base64-encoded data always including an extra trailing newline ('\n').

encodestring(*s*)
编码字符串*s*，能够包含任意的二进制数据，返回一个包含一行或多行Base64编码数据。encodestring() 返回一个包含一行或多行Base64编码数据的字符串，而且总是包含一个额外的结尾换行符('\n').

See Also:

Module binascii (section **??**):
Support module containing ASCII-to-binary and binary-to-ASCII conversions.

RFC 1521, "*MIME (Multipurpose Internet Mail Extensions) Part One: Mechanisms for Specifying and Describing the Format of In*
Section 5.2, "Base64 Content-Transfer-Encoding," provides the definition of the base64 encoding.

7.2 xml.dom — The Document Object Model 文档对象模式API

New in version 2.0.

The Document Object Model, or "DOM," is a cross-language API from the World Wide Web Consortium (W3C) for accessing and modifying XML documents. A DOM implementation presents an XML document as a tree structure, or allows client code to build such a structure from scratch. It then gives access to the structure through a set of objects which provided well-known interfaces.

文档对象模式,或称"DOM" 是来自互联网协会(W3C) 的一种交叉语言,用以访问和修改XML文档. DOM 可以将XML文档处理成为一种树状的数据结构，或是从客户代码生成这种数据结构. 并且提供了一套易读的对象来访问这种数据结构.

1

The DOM is extremely useful for random-access applications. SAX only allows you a view of one bit of the document at a time. If you are looking at one SAX element, you have no access to another. If you are looking at a text node, you have no access to a containing element. When you write a SAX application, you need to keep track of your program's position in the document somewhere in your own code. SAX does not do it for you. Also, if you need to look ahead in the XML document, you are just out of luck.

DOM 对于随机访问的应用非常有用. 而对于SAX,则只允许你同一时刻只能访问一小片XML文档, 这意味着: 如果你正在关注一个SAX元素,则你无法访问另一个. 如果你正在访问一个文本结点,则你无法访问其容器元素. 当你写SAX应用时,你得在代码中跟踪你程序在XML文档中的作用位置. SAX不能替你进行位置的记录. 所以,如果你需要随时回到前面处理,SAX将无能为力!

Some applications are simply impossible in an event driven model with no access to a tree. Of course you could build some sort of tree yourself in SAX events, but the DOM allows you to avoid writing that code. The DOM is a standard tree representation for XML data.

事务驱动模式的应用使你几乎不可能访问树形结构. 当然你可以自个儿架构某种结构在SAX的事务过程中来存放树结构, 但是DOM 省却了你的这种代码. DOM 是XML数据的标准树表达方式!

The Document Object Model is being defined by the W3C in stages, or "levels" in their terminology. The Python mapping of the API is substantially based on the DOM Level 2 recommendation. The mapping of the Level 3 specification, currently only available in draft form, is being developed by the Python XML Special Interest Group as part of the PyXML package. Refer to the documentation bundled with that package for information on

[1]译按：（cross-language?交叉语言;）

the current state of DOM Level 3 support.

文档对象模式在W3C **的术语体系中被划分为"阶段"**(stages)**或是"阶层"**(levels). Python **充分的映射了**DOM **阶层**2 **推荐的所有特性**. **阶层**3 **的规范当前**Python**的支持仅仅在草案阶段**Python XML **特殊兴趣组也成为**PyXML **包的一部分**.

2

DOM applications typically start by parsing some XML into a DOM. How this is accomplished is not covered at all by DOM Level 1, and Level 2 provides only limited improvements: There is a `DOMImplementation` object class which provides access to `Document` creation methods, but no way to access an XML reader/parser/Document builder in an implementation-independent way. There is also no well-defined way to access these methods without an existing `Document` object. In Python, each DOM implementation will provide a function `getDOMImplementation()`. DOM Level 3 adds a Load/Store specification, which defines an interface to the reader, but this is not yet available in the Python standard library.

典型的DOM**应用由解析一些**XML**成为**DOM **开始**. **这在**Python **已经完全封装了**DOM **阶层**1 **至阶层**2**的规范**,**仅仅除了一点限制**: `DOMImplementation` **对象类提供**`Document` **创建方法**, **但是没有独立执行的**XML **读取**/**解析**/**文档结构创建的方法**. Python**中**,**每次**DOM **的执行都需使用函式**`getDOMImplementation()`. DOM **阶层**3 **增加了**Load/Store **规范**, **可以定义一个实例来进行**XML**的读取**, **但是**,**这还没有在**Python **标准库中实现**.

Once you have a DOM document object, you can access the parts of your XML document through its properties and methods. These properties are defined in the DOM specification; this portion of the reference manual describes the interpretation of the specification in Python.

一但你获得了DOM**文档对象**,**你便可以通过其提供的属性和方法来访问**XML**的各个部分**. **这些属性在**DOM**规范中进行了定义**; **手册的这一部分描述**Python **的**DOM **规范**.

The specification provided by the W3C defines the DOM API for Java, ECMAScript, and OMG IDL. The Python mapping defined here is based in large part on the IDL version of the specification, but strict compliance is not required (though implementations are free to support the strict mapping from IDL). See section 7.2.3, "Conformance," for a detailed discussion of mapping requirements.

W3C**定义的**DOM API **规范有**Java,ECMAScript, **和**OMG IDL **版本的实现**. Python**的映射主要基于**IDL **版本**, **但是并没有严密的匹配**(**尽管允许从**IDL**精确映射执行**). **参考**7.2.3**一节**, **一致性**"Conformance," **讨论详细的映射必要条件**.

See Also:

Document Object Model (DOM) Level 2 Specification DOM ***阶层**2* ***规范***
(http://www.w3.org/TR/DOM-Level-2-Core/)

The W3C recommendation upon which the Python DOM API is based. W3C **推荐**!Python DOM API **完全符合**.

Document Object Model (DOM) Level 1 Specification DOM ***阶层**1* ***规范***
(http://www.w3.org/TR/REC-DOM-Level-1/)

The W3C recommendation for the DOM supported by W3C **建议至少要支持的**,Python **由**`xml.dom.minidom`**提供映射**.

PyXML
(http://pyxml.sourceforge.net)

Users that require a full-featured implementation of DOM should use the PyXML package. **需要全部可执行特性的用户**,**可以使用**PyXML **包**.

CORBA Scripting with Python CORBA ***分布式对象系统脚本与****Python*
(http://cgi.omg.org/cgi-bin/doc?orbos/99-08-02.pdf)

This specifies the mapping from OMG IDL to Python. **论述**Python **对于**OMG IDL **的映射**

7.2.1 Module Contents **模块**s

The `xml.dom` contains the following functions:

[2] **译按:**(levels?**级别**,**阶层**;)

`xml.dom` **包含以下函式**:

`registerDOMImplementation`(*name, factory*)

Register the *factory* function with the name *name*. The factory function should return an object which implements the `DOMImplementation` interface. The factory function can return the same object every time, or a new one for each call, as appropriate for the specific implementation (e.g. if that implementation supports some customization).

[3] **以***name***注册**"**工厂**(*factory*)**函式**". "**工厂函式**"**应该返回一个**`DOMImplementation`**对象实例**. **所谓**"**工厂函式**",**就是每次调用总能返回适当的对象实例**,**依据特定的实现**. (**例如**:**支持某些定制的实现**)

(**译按**:**行文有待斟酌**!)

`getDOMImplementation`([*name*[, *features*]])

Return a suitable DOM implementation. The *name* is either well-known, the module name of a DOM implementation, or `None`. If it is not `None`, imports the corresponding module and returns a `DOMImplementation` object if the import succeeds. If no name is given, and if the environment variable PYTHON_DOM is set, this variable is used to find the implementation.

If name is not given, this examines the available implementations to find one with the required feature set. If no implementation can be found, raise an `ImportError`. The features list must be a sequence of (*feature*, *version*) pairs which are passed to the `hasFeature()` method on available `DOMImplementation` objects.

Some convenience constants are also provided:

`EMPTY_NAMESPACE`

The value used to indicate that no namespace is associated with a node in the DOM. This is typically found as the `namespaceURI` of a node, or used as the *namespaceURI* parameter to a namespaces-specific method. New in version 2.2.

`XML_NAMESPACE`

The namespace URI associated with the reserved prefix `xml`, as defined by *Namespaces in XML* (section 4). New in version 2.2.

`XMLNS_NAMESPACE`

The namespace URI for namespace declarations, as defined by *Document Object Model (DOM) Level 2 Core Specification* (section 1.1.8). New in version 2.2.

`XHTML_NAMESPACE`

The URI of the XHTML namespace as defined by *XHTML 1.0: The Extensible HyperText Markup Language* (section 3.1.1). New in version 2.2.

In addition, `xml.dom` contains a base `Node` class and the DOM exception classes. The `Node` class provided by this module does not implement any of the methods or attributes defined by the DOM specification; concrete DOM implementations must provide those. The `Node` class provided as part of this module does provide the constants used for the `nodeType` attribute on concrete `Node` objects; they are located within the class rather than at the module level to conform with the DOM specifications.

7.2.2 Objects in the DOM

The definitive documentation for the DOM is the DOM specification from the W3C.

Note that DOM attributes may also be manipulated as nodes instead of as simple strings. It is fairly rare that you must do this, however, so this usage is not yet documented.

[3]**译按：**（factory function?**工厂函式**,**范式的一种模式**;）

Interface	Section	Purpose
`DOMImplementation`	7.2.2	Interface to the underlying implementation.
`Node`	7.2.2	Base interface for most objects in a document.
`NodeList`	7.2.2	Interface for a sequence of nodes.
`DocumentType`	7.2.2	Information about the declarations needed to process a document.
`Document`	7.2.2	Object which represents an entire document.
`Element`	7.2.2	Element nodes in the document hierarchy.
`Attr`	7.2.2	Attribute value nodes on element nodes.
`Comment`	7.2.2	Representation of comments in the source document.
`Text`	7.2.2	Nodes containing textual content from the document.
`ProcessingInstruction`	7.2.2	Processing instruction representation.

An additional section describes the exceptions defined for working with the DOM in Python.

DOMImplementation Objects

The `DOMImplementation` interface provides a way for applications to determine the availability of particular features in the DOM they are using. DOM Level 2 added the ability to create new `Document` and `DocumentType` objects using the `DOMImplementation` as well.

`hasFeature`(*feature, version*)

Node Objects

All of the components of an XML document are subclasses of `Node`.

`nodeType`
An integer representing the node type. Symbolic constants for the types are on the `Node` object: `ELEMENT_NODE`, `ATTRIBUTE_NODE`, `TEXT_NODE`, `CDATA_SECTION_NODE`, `ENTITY_NODE`, `PROCESSING_INSTRUCTION_NODE`, `COMMENT_NODE`, `DOCUMENT_NODE`, `DOCUMENT_TYPE_NODE`, `NOTATION_NODE`. This is a read-only attribute.

`parentNode`
The parent of the current node, or `None` for the document node. The value is always a `Node` object or `None`. For `Element` nodes, this will be the parent element, except for the root element, in which case it will be the `Document` object. For `Attr` nodes, this is always `None`. This is a read-only attribute.

`attributes`
A `NamedNodeMap` of attribute objects. Only elements have actual values for this; others provide `None` for this attribute. This is a read-only attribute.

`previousSibling`
The node that immediately precedes this one with the same parent. For instance the element with an end-tag that comes just before the *self* element's start-tag. Of course, XML documents are made up of more than just elements so the previous sibling could be text, a comment, or something else. If this node is the first child of the parent, this attribute will be `None`. This is a read-only attribute.

`nextSibling`
The node that immediately follows this one with the same parent. See also `previousSibling`. If this is the last child of the parent, this attribute will be `None`. This is a read-only attribute.

`childNodes`
A list of nodes contained within this node. This is a read-only attribute.

`firstChild`
The first child of the node, if there are any, or `None`. This is a read-only attribute.

`lastChild`
The last child of the node, if there are any, or `None`. This is a read-only attribute.

`localName`
The part of the `tagName` following the colon if there is one, else the entire `tagName`. The value is a string.

`prefix`
The part of the `tagName` preceding the colon if there is one, else the empty string. The value is a string, or `None`

`namespaceURI`
The namespace associated with the element name. This will be a string or `None`. This is a read-only attribute.

`nodeName`
This has a different meaning for each node type; see the DOM specification for details. You can always get the information you would get here from another property such as the `tagName` property for elements or the `name` property for attributes. For all node types, the value of this attribute will be either a string or `None`. This is a read-only attribute.

`nodeValue`
This has a different meaning for each node type; see the DOM specification for details. The situation is similar to that with `nodeName`. The value is a string or `None`.

`hasAttributes``()`
Returns true if the node has any attributes.

`hasChildNodes``()`
Returns true if the node has any child nodes.

`isSameNode``(`*other*`)`
Returns true if *other* refers to the same node as this node. This is especially useful for DOM implementations which use any sort of proxy architecture (because more than one object can refer to the same node).

Note: This is based on a proposed DOM Level 3 API which is still in the "working draft" stage, but this particular interface appears uncontroversial. Changes from the W3C will not necessarily affect this method in the Python DOM interface (though any new W3C API for this would also be supported).

`appendChild``(`*newChild*`)`
Add a new child node to this node at the end of the list of children, returning *newChild*.

`insertBefore``(`*newChild, refChild*`)`
Insert a new child node before an existing child. It must be the case that *refChild* is a child of this node; if not, `ValueError` is raised. *newChild* is returned.

`removeChild``(`*oldChild*`)`
Remove a child node. *oldChild* must be a child of this node; if not, `ValueError` is raised. *oldChild* is returned on success. If *oldChild* will not be used further, its `unlink()` method should be called.

`replaceChild``(`*newChild, oldChild*`)`
Replace an existing node with a new node. It must be the case that *oldChild* is a child of this node; if not, `ValueError` is raised.

`normalize``()`
Join adjacent text nodes so that all stretches of text are stored as single `Text` instances. This simplifies processing text from a DOM tree for many applications. New in version 2.1.

`cloneNode``(`*deep*`)`
Clone this node. Setting *deep* means to clone all child nodes as well. This returns the clone.

NodeList Objects

A `NodeList` represents a sequence of nodes. These objects are used in two ways in the DOM Core recommendation: the `Element` objects provides one as its list of child nodes, and the `getElementsByTagName()` and `getElementsByTagNameNS()` methods of `Node` return objects with this interface to represent query results.

The DOM Level 2 recommendation defines one method and one attribute for these objects:

`item``(`*i*`)`
Return the *i*'th item from the sequence, if there is one, or `None`. The index *i* is not allowed to be less then zero or greater than or equal to the length of the sequence.

`length`
The number of nodes in the sequence.

In addition, the Python DOM interface requires that some additional support is provided to allow `NodeList` objects to be used as Python sequences. All `NodeList` implementations must include support for `__len__()` and `__getitem__()`; this allows iteration over the `NodeList` in `for` statements and proper support for the `len()` built-in function.

If a DOM implementation supports modification of the document, the `NodeList` implementation must also support the `__setitem__()` and `__delitem__()` methods.

DocumentType Objects

Information about the notations and entities declared by a document (including the external subset if the parser uses it and can provide the information) is available from a `DocumentType` object. The `DocumentType` for a document is available from the `Document` object's `doctype` attribute; if there is no `DOCTYPE` declaration for the document, the document's `doctype` attribute will be set to `None` instead of an instance of this interface.

`DocumentType` is a specialization of `Node`, and adds the following attributes:

`publicId`
The public identifier for the external subset of the document type definition. This will be a string or `None`.

`systemId`
The system identifier for the external subset of the document type definition. This will be a URI as a string, or `None`.

`internalSubset`
A string giving the complete internal subset from the document. This does not include the brackets which enclose the subset. If the document has no internal subset, this should be `None`.

`name`
The name of the root element as given in the `DOCTYPE` declaration, if present.

`entities`
This is a `NamedNodeMap` giving the definitions of external entities. For entity names defined more than once, only the first definition is provided (others are ignored as required by the XML recommendation). This may be `None` if the information is not provided by the parser, or if no entities are defined.

`notations`
This is a `NamedNodeMap` giving the definitions of notations. For notation names defined more than once, only the first definition is provided (others are ignored as required by the XML recommendation). This may be `None` if the information is not provided by the parser, or if no notations are defined.

Document Objects

A `Document` represents an entire XML document, including its constituent elements, attributes, processing instructions, comments etc. Remeber that it inherits properties from `Node`.

`documentElement`
The one and only root element of the document.

`createElement`(*tagName*)
Create and return a new element node. The element is not inserted into the document when it is created. You need to explicitly insert it with one of the other methods such as `insertBefore()` or `appendChild()`.

`createElementNS`(*namespaceURI, tagName*)
Create and return a new element with a namespace. The *tagName* may have a prefix. The element is not inserted into the document when it is created. You need to explicitly insert it with one of the other methods such as `insertBefore()` or `appendChild()`.

`createTextNode`(*data*)
Create and return a text node containing the data passed as a parameter. As with the other creation methods, this one does not insert the node into the tree.

`createComment`(*data*)
: Create and return a comment node containing the data passed as a parameter. As with the other creation methods, this one does not insert the node into the tree.

`createProcessingInstruction`(*target, data*)
: Create and return a processing instruction node containing the *target* and *data* passed as parameters. As with the other creation methods, this one does not insert the node into the tree.

`createAttribute`(*name*)
: Create and return an attribute node. This method does not associate the attribute node with any particular element. You must use `setAttributeNode()` on the appropriate `Element` object to use the newly created attribute instance.

`createAttributeNS`(*namespaceURI, qualifiedName*)
: Create and return an attribute node with a namespace. The *tagName* may have a prefix. This method does not associate the attribute node with any particular element. You must use `setAttributeNode()` on the appropriate `Element` object to use the newly created attribute instance.

`getElementsByTagName`(*tagName*)
: Search for all descendants (direct children, children's children, etc.) with a particular element type name.

`getElementsByTagNameNS`(*namespaceURI, localName*)
: Search for all descendants (direct children, children's children, etc.) with a particular namespace URI and localname. The localname is the part of the namespace after the prefix.

Element Objects

`Element` is a subclass of `Node`, so inherits all the attributes of that class.

`tagName`
: The element type name. In a namespace-using document it may have colons in it. The value is a string.

`getElementsByTagName`(*tagName*)
: Same as equivalent method in the `Document` class.

`getElementsByTagNameNS`(*tagName*)
: Same as equivalent method in the `Document` class.

`getAttribute`(*attname*)
: Return an attribute value as a string.

`getAttributeNode`(*attrname*)
: Return the `Attr` node for the attribute named by *attrname*.

`getAttributeNS`(*namespaceURI, localName*)
: Return an attribute value as a string, given a *namespaceURI* and *localName*.

`getAttributeNodeNS`(*namespaceURI, localName*)
: Return an attribute value as a node, given a *namespaceURI* and *localName*.

`removeAttribute`(*attname*)
: Remove an attribute by name. No exception is raised if there is no matching attribute.

`removeAttributeNode`(*oldAttr*)
: Remove and return *oldAttr* from the attribute list, if present. If *oldAttr* is not present, `NotFoundErr` is raised.

`removeAttributeNS`(*namespaceURI, localName*)
: Remove an attribute by name. Note that it uses a localName, not a qname. No exception is raised if there is no matching attribute.

`setAttribute`(*attname, value*)
: Set an attribute value from a string.

`setAttributeNode`(*newAttr*)
: Add a new attibute node to the element, replacing an existing attribute if necessary if the `name` attribute

matches. If a replacement occurs, the old attribute node will be returned. If *newAttr* is already in use, `InuseAttributeErr` will be raised.

`setAttributeNodeNS``(`*newAttr*`)`
Add a new attribute node to the element, replacing an existing attribute if necessary if the `namespaceURI` and `localName` attributes match. If a replacement occurs, the old attribute node will be returned. If *newAttr* is already in use, `InuseAttributeErr` will be raised.

`setAttributeNS``(`*namespaceURI, qname, value*`)`
Set an attribute value from a string, given a *namespaceURI* and a *qname*. Note that a qname is the whole attribute name. This is different than above.

Attr Objects

`Attr` inherits from `Node`, so inherits all its attributes.

`name`
The attribute name. In a namespace-using document it may have colons in it.

`localName`
The part of the name following the colon if there is one, else the entire name. This is a read-only attribute.

`prefix`
The part of the name preceding the colon if there is one, else the empty string.

NamedNodeMap Objects

`NamedNodeMap` does *not* inherit from `Node`.

`length`
The length of the attribute list.

`item``(`*index*`)`
Return an attribute with a particular index. The order you get the attributes in is arbitrary but will be consistent for the life of a DOM. Each item is an attribute node. Get its value with the `value` attribbute.

There are also experimental methods that give this class more mapping behavior. You can use them or you can use the standardized `getAttribute*()` family of methods on the `Element` objects.

Comment Objects

`Comment` represents a comment in the XML document. It is a subclass of `Node`, but cannot have child nodes.

`data`
The content of the comment as a string. The attribute contains all characters between the leading `<!--` and trailing `-->`, but does not include them.

Text and CDATASection Objects

The `Text` interface represents text in the XML document. If the parser and DOM implementation support the DOM's XML extension, portions of the text enclosed in CDATA marked sections are stored in `CDATASection` objects. These two interfaces are identical, but provide different values for the `nodeType` attribute.

These interfaces extend the `Node` interface. They cannot have child nodes.

`data`
The content of the text node as a string.

Note: The use of a `CDATASection` node does not indicate that the node represents a complete CDATA marked section, only that the content of the node was part of a CDATA section. A single CDATA section may be represented by more than one node in the document tree. There is no way to determine whether two adjacent `CDATASection` nodes represent different CDATA marked sections.

ProcessingInstruction Objects

Represents a processing instruction in the XML document; this inherits from the `Node` interface and cannot have child nodes.

`target`
The content of the processing instruction up to the first whitespace character. This is a read-only attribute.

`data`
The content of the processing instruction following the first whitespace character.

Exceptions

New in version 2.1.

The DOM Level 2 recommendation defines a single exception, `DOMException`, and a number of constants that allow applications to determine what sort of error occurred. `DOMException` instances carry a `code` attribute that provides the appropriate value for the specific exception.

The Python DOM interface provides the constants, but also expands the set of exceptions so that a specific exception exists for each of the exception codes defined by the DOM. The implementations must raise the appropriate specific exception, each of which carries the appropriate value for the `code` attribute.

exception `DOMException`
Base exception class used for all specific DOM exceptions. This exception class cannot be directly instantiated.

exception `DomstringSizeErr`
Raised when a specified range of text does not fit into a string. This is not known to be used in the Python DOM implementations, but may be received from DOM implementations not written in Python.

exception `HierarchyRequestErr`
Raised when an attempt is made to insert a node where the node type is not allowed.

exception `IndexSizeErr`
Raised when an index or size parameter to a method is negative or exceeds the allowed values.

exception `InuseAttributeErr`
Raised when an attempt is made to insert an `Attr` node that is already present elsewhere in the document.

exception `InvalidAccessErr`
Raised if a parameter or an operation is not supported on the underlying object.

exception `InvalidCharacterErr`
This exception is raised when a string parameter contains a character that is not permitted in the context it's being used in by the XML 1.0 recommendation. For example, attempting to create an `Element` node with a space in the element type name will cause this error to be raised.

exception `InvalidModificationErr`
Raised when an attempt is made to modify the type of a node.

exception `InvalidStateErr`
Raised when an attempt is made to use an object that is not or is no longer usable.

exception `NamespaceErr`
If an attempt is made to change any object in a way that is not permitted with regard to the *Namespaces in XML* recommendation, this exception is raised.

exception `NotFoundErr`
Exception when a node does not exist in the referenced context. For example, `NamedNodeMap.removeNamedItem()` will raise this if the node passed in does not exist in the map.

exception `NotSupportedErr`
Raised when the implementation does not support the requested type of object or operation.

exception `NoDataAllowedErr`
This is raised if data is specified for a node which does not support data.

exception `NoModificationAllowedErr`
Raised on attempts to modify an object where modifications are not allowed (such as for read-only nodes).

exception `SyntaxErr`
Raised when an invalid or illegal string is specified.

exception `WrongDocumentErr`
Raised when a node is inserted in a different document than it currently belongs to, and the implementation does not support migrating the node from one document to the other.

The exception codes defined in the DOM recommendation map to the exceptions described above according to this table:

Constant	Exception
`DOMSTRING_SIZE_ERR`	`DomstringSizeErr`
`HIERARCHY_REQUEST_ERR`	`HierarchyRequestErr`
`INDEX_SIZE_ERR`	`IndexSizeErr`
`INUSE_ATTRIBUTE_ERR`	`InuseAttributeErr`
`INVALID_ACCESS_ERR`	`InvalidAccessErr`
`INVALID_CHARACTER_ERR`	`InvalidCharacterErr`
`INVALID_MODIFICATION_ERR`	`InvalidModificationErr`
`INVALID_STATE_ERR`	`InvalidStateErr`
`NAMESPACE_ERR`	`NamespaceErr`
`NOT_FOUND_ERR`	`NotFoundErr`
`NOT_SUPPORTED_ERR`	`NotSupportedErr`
`NO_DATA_ALLOWED_ERR`	`NoDataAllowedErr`
`NO_MODIFICATION_ALLOWED_ERR`	`NoModificationAllowedErr`
`SYNTAX_ERR`	`SyntaxErr`
`WRONG_DOCUMENT_ERR`	`WrongDocumentErr`

7.2.3 Conformance

This section describes the conformance requirements and relationships between the Python DOM API, the W3C DOM recommendations, and the OMG IDL mapping for Python.

Type Mapping

The primitive IDL types used in the DOM specification are mapped to Python types according to the following table.

IDL Type	Python Type
`boolean`	`IntegerType` (with a value of `0` or `1`)
`int`	`IntegerType`
`long int`	`IntegerType`
`unsigned int`	`IntegerType`

Additionally, the `DOMString` defined in the recommendation is mapped to a Python string or Unicode string. Applications should be able to handle Unicode whenever a string is returned from the DOM.

The IDL `null` value is mapped to `None`, which may be accepted or provided by the implementation whenever `null` is allowed by the API.

Accessor Methods

The mapping from OMG IDL to Python defines accessor functions for IDL `attribute` declarations in much the way the Java mapping does. Mapping the IDL declarations

```
readonly attribute string someValue;
         attribute string anotherValue;
```

yields three accessor functions: a "get" method for `someValue` (`_get_someValue()`), and "get" and "set" methods for `anotherValue` (`_get_anotherValue()` and `_set_anotherValue()`). The mapping, in particular, does not require that the IDL attributes are accessible as normal Python attributes: *object*`.someValue` is *not* required to work, and may raise an `AttributeError`.

The Python DOM API, however, *does* require that normal attribute access work. This means that the typical surrogates generated by Python IDL compilers are not likely to work, and wrapper objects may be needed on the client if the DOM objects are accessed via CORBA. While this does require some additional consideration for CORBA DOM clients, the implementers with experience using DOM over CORBA from Python do not consider this a problem. Attributes that are declared `readonly` may not restrict write access in all DOM implementations.

Additionally, the accessor functions are not required. If provided, they should take the form defined by the Python IDL mapping, but these methods are considered unnecessary since the attributes are accessible directly from Python. "Set" accessors should never be provided for `readonly` attributes.

第八章

多媒体服务Multimedia Services

The modules described in this chapter implement various algorithms or interfaces that are mainly useful for multimedia applications. They are available at the discretion of the installation. Here’s an overview:

本章所描述的模组实现了只要用在多媒体应用中遇到的介面及算法。能否调用它们视乎安装 Python **时的选项而定。以下是概述**:

第九章

加密服务Cryptographic Services

The modules described in this chapter implement various algorithms of a cryptographic nature. They are available at the discretion of the installation. Here's an overview:

本章描述的模组实现了具加密性质的不同算法。它们能不能用视安装时的选项而定。以下是概述。

Hardcore cypherpunks will probably find the cryptographic modules written by A.M. Kuchling of further interest; the package adds built-in modules for DES and IDEA encryption, provides a Python module for reading and decrypting PGP files, and then some. These modules are not distributed with Python but available separately. See the URL http://www.amk.ca/python/code/crypto.html for more information.

对加密十分狂热者可以找一找A.M. Kuchling **写的加密模组**; **它提供了**DES **及**IDEA **加密法的内建模组**, **提供了一个**Python **模组来读取及解读**PGP **档案**, **以及别的东西**. **这些模组并不包括在**Python **的发布中**, **但可以从别处取得到。详情请看**http://www.amk.ca/python/code/crypto.html **。**

第十章

Python 语言服务Python Language Services

Python provides a number of modules to assist in working with the Python language. These module support tokenizing, parsing, syntax analysis, bytecode disassembly, and various other facilities.

Python **有一系列模组，对分析**Python **语言本身的工作提供了帮助。这些模组包括标记符的辨识，分词，字元码的的反编译以及其他的一些辅助工具。**

These modules include:

此类模组包括：

第十一章

SGI IRIX 特定服务SGI IRIX Specific Services

The modules described in this chapter provide interfaces to features that are unique to SGI's IRIX operating system (versions 4 and 5).

本章描述的模组为SGI **的**IRIX **操作系统(版本**4 **及**5) **提供介面**, **存取该系统所独有的一些功能。**

`al`	SGI **的音讯功能**Audio functions on the SGI.
`AL`	`al` **模组中用到的常数。**Constants used with the `al` module.

11.1 `al` — SGI 的音讯功能Audio functions on the SGI

This module provides access to the audio facilities of the SGI Indy and Indigo workstations. See section 3A of the IRIX man pages for details. You'll need to read those man pages to understand what these functions do! Some of the functions are not available in IRIX releases before 4.0.5. Again, see the manual to check whether a specific function is available on your platform.

本模组可用来存取SGI Indy **及**Indigo **工作站的音讯设备。详细情况可参考**IRIX **帮助页的第**3A **节。你要看过了那些帮助页才能明白这几个函数是干什么用的。以下有的函数不能在**IRIX 4.0.5 **版以前使用。再说一遍，看相关的手册来检查某个指定的函数是否能用在你的平台上。**

All functions and methods defined in this module are equivalent to the C functions with '`AL`' prefixed to their name.

本模组所定义的所有函数及方法和C **中以**'`AL`' **前缀的同名函数是等价的。**

Symbolic constants from the C header file `<audio.h>` are defined in the standard module `AL`, see below.

在C **标头档**`<audio.h>` **中定义的符号常数定义则被放置到了**`AL`, **看下面。**

Warning: The current version of the audio library may dump core when bad argument values are passed rather than returning an error status. Unfortunately, since the precise circumstances under which this may happen are undocumented and hard to check, the Python interface can provide no protection against this kind of problems. (One example is specifying an excessive queue size — there is no documented upper limit.)

Warning: 音讯函数库目的的版本当调用用时收到不合法的引数时会产生core **档而不会传回错误态。很可惜的是，因为到底哪种情况才会产生这些错无法在现有的文献中找到而且也很难试出来，所以**Python **无法对这些问题提供保护。**(**例如：提交一个额外的队列的大小时，没有任何文献提到它的最大上限是多少。**)

The module defines the following functions:

本模组提供以下的函数：

`openport`(*name, direction*[, *config*])

The name and direction arguments are strings. The optional *config* argument is a configuration object as returned by `newconfig()`. The return value is an *audio port object*; methods of audio port objects are described below.

name **及**direction **引数皆是字串。可选引数***config* **则是一个透过调用**`newconfig()` **而得到的配置对象。函数传回*音讯端口对象*** *audio port object*: **这个对象包含下面描述的方法。**

`newconfig``()`
The return value is a new *audio configuration object*; methods of audio configuration objects are described below.

传回一个新的*音讯配置对象* *audio configuration object*: **该对象包含下面描述的方法。**

`queryparams``(`*device*`)`
The device argument is an integer. The return value is a list of integers containing the data returned by `ALqueryparams()`.

device **引数是一个整数。传回一个整数列表，其中包含由**`ALqueryparams()` **传回的数据。**

`getparams``(`*device, list*`)`
The *device* argument is an integer. The list argument is a list such as returned by `queryparams()`; it is modified in place (!).

device **引数是一个整数。**list **引数则是由**`queryparam()` **传回的列表；这个列表会被即场修改的**(**！**)

`setparams``(`*device, list*`)`
The *device* argument is an integer. The *list* argument is a list such as returned by `queryparams()`.

device **引数是一个整数。**list **引数则是由**`queryparam()` **传回的列表**.

11.1.1 配置对象Configuration Objects

Configuration objects returned by `newconfig()` have the following methods:

配置对象由`newconfig()` **传回**, **它有以下方对**:

`getqueuesize``()`
Return the queue size. **传回队列的大小。**

`setqueuesize``(`*size*`)`
Set the queue size. **设定队列大小。**

`getwidth``()`
Get the sample width. **获得取样本的宽度。**

`setwidth``(`*width*`)`
Set the sample width. **设定取样的宽度。**

`getchannels``()`
Get the channel count. **取得频道的数目。**

`setchannels``(`*nchannels*`)`
Set the channel count. **设定频道数目。**

`getsampfmt``()`
Get the sample format. **取得样本的格式。**

`setsampfmt``(`*sampfmt*`)`
Set the sample format. **设定样本的格式。**

`getfloatmax``()`
Get the maximum value for floating sample formats. **取得浮点样本格式的最大值。**

`setfloatmax``(`*floatmax*`)`
Set the maximum value for floating sample formats. **设定浮点样本格式的最大值。**

11.1.2 端口对象Port Objects

Port objects, as returned by `openport()`, have the following methods:

端口对象由调用`openport()` **得到，它有以下方法：**

`closeport``()`
: Close the port. **关闭端口。**

`getfd``()`
: Return the file descriptor as an int. **传回描述档案的整数值。**

`getfilled``()`
: Return the number of filled samples. **传回已取得值的样本的数目。**

`getfillable``()`
: Return the number of fillable samples. **传回可以取值的样本的数目。**

`readsamps``(`*nsamples*`)`
: Read a number of samples from the queue, blocking if necessary. Return the data as a string containing the raw data, (e.g., 2 bytes per sample in big-endian byte order (high byte, low byte) if you have set the sample width to 2 bytes).

从队列中读取一定数目的样本，以等候态直至读取完毕。传回代表原始数据的字符串，(**例如**, **如你设定取样宽度为两个位元的话**, **则每个样本以双字元大顺位次序存放**(**高字元**, **低字元**)).

`writesamps``(`*samples*`)`
: Write samples into the queue, blocking if necessary. The samples are encoded as described for the `readsamps()` return value.

把样本写到队列中去, **以等候态直至写完。样本的编码方式以**`readsamps()` **函数的传回值所描述的一样。**

`getfillpoint``()`
: Return the 'fill point'. **传回**"**填入点**fill point".

`setfillpoint``(`*fillpoint*`)`
: Set the 'fill point'. **设置 " 填入点**fill point".

`getconfig``()`
: Return a configuration object containing the current configuration of the port.

传回一个配置对象，其中包含了目前端口的配置。

`setconfig``(`*config*`)`
: Set the configuration from the argument, a configuration object.

用引数，一个配置对象来设置端口。

`getstatus``(`*list*`)`
: Get status information on last error.

取得最近一次错误的状态资料。

11.2 AL — `al` 模组中用到的常数。Constants used with the `al` module

This module defines symbolic constants needed to use the built-in module `al` (see above); they are equivalent to those defined in the C header file `<audio.h>` except that the name prefix 'AL_' is omitted. Read the module source for a complete list of the defined names. Suggested use:

此模组定义了使用内建模组`al` (**看上面**)**时要用到的常量，它们和**C **标头档**`<audio.h>` **中所定义的是一样的，只不过忽略掉了所有的**'AL_' **前缀。全部常量的定义可以参看源码。建议用法如下：**

```
import al
from AL import *
```

第十二章

SunOS 特定服务SunOS Specific Services

The modules described in this chapter provide interfaces to features that are unique to the SunOS operating system (versions 4 and 5; the latter is also known as Solaris version 2).

本章所描述的模组提供介面存取SunOX **操作系统所独有的功能。**(**版本**4, **及**5; **后者也被称为**Solaris Version 2)

第十三章

MS Windows 有关的服务MS Windows Services

This chapter describes modules that are only available on MS Windows platforms.

本章描述了只存在于MS Windows **平台上的模组**.

附录 A

翻译团队

- **杜文山**: . . .
- **刘鑫**: . . .
- **余颖**: . . .
- glace : **后勤**
- Leira : . . .
- Hackgou : . . .
- Zoom.Quiet : . . .