Macintoshモジュールリファレンス

リリース *2.4*

Guido van Rossum
Fred L. Drake, Jr., editor
日本語訳: Python ドキュメント翻訳プロジェクト

平成 18 年 5 月 11 日

Python Software Foundation
Email: docs@python.org

概 要

このライブラリリファレンスマニュアルでは、Macintosh 用の Python 拡張に関して詳しく記述します。*Python Library Reference* も同時に参照しながら利用するようにしてください。標準ライブラリと組み込み型はそちらに詳しく書かれています。

このマニュアルを読むにあたっては、Python 言語の基礎知識を持っていることが必要です。Python の肩のこらない入門編が必要なら、*Python Tutorial* を読んでください。*Python Reference Manual* は、構文や意味論に関する疑問を解決するため、それなりに分かった人が読むべきものです。最後にひとつ。インタプリタの拡張と組み込み *Extending and Embedding the Python Interpreter* という名のマニュアルでは、Python へ新たに拡張機能を追加する方法と、他のアプリケーションに組み込む方法について述べています。

目次

一章

Mac OS 9でPythonを利用する

Macintosh 上での Python の使い方は、特に Mac OS 9 上においては UNIX ライクなシステムや Windows システム上で使うのと全く異なっています (MacPython-OSX には完全な UNIX Python が入っているので、UNIX ライクなシステムとほぼ同じ知識を利用できます)。ほとんどの Python 文書や、「公式の」文書、刊行されている書籍では、UNIX ライクなシステムや Windows システム上でどう Python を使うかについて述べているにすぎず、MacPython-OS9 を初めて利用する人を混乱させる原因になっています。この章では、簡単な入門編として Macintosh 上で Python を具体的にどう使うのかを解説していきます。

IDE についての章 (1.3 節) は MacPython-OSX にも関係しています。

1.1 MacPython-OSX の取得とインストール

Python 2.3a2 の時点では、自分のマシンに MacPython-OSX をインストールするにはソース配布物を取得し、いわゆる "framework Python" をビルドするのが唯一の正しい方法です。ビルド方法の詳細は 'Mac/OSX/README' にあります。

バイナリインストーラが利用可能になれば、その詳細が http://www.cwi.nl/~{}jack/macpython.html に掲示されるはずです。

インストールを行うと、様々なものが入ります:

- 'Applications' フォルダ下の 'MacPython-2.3' フォルダ。このフォルダの中には、 PythonIDE 統合開発環境、ファインダからダブルクリックして Python スクリプトを起動するための PythonLauncher、Package Manager が入っています。
- ほぼ標準の UNIX 版のコマンドライン Python インタプリタ。'/usr/local/bin/python' にインストールされます。ただし、通常作成される '/usr/local/lib/python' はできません。
- フレームワーク '/Library/Frameworks/Python.framework' 。実際の処理にかかわる部分ですが、たいていの場合それを気にする必要はありません。

単に上の 3 つを削除すれば、MacPython をアンインストールできます。

PythonIDE には "MacPython Help" という名前の Apple Help Viewer ブックが入っています。このヘルプはヘルプメニューからアクセスできます。まったくの Python の初心者は、このドキュメントの IDE の説明から読み始めるとよいでしょう。

他の UNIX プラットフォーム上で動作する Python について詳しいなら、UNIX シェルからの Python スクリプトの実行を説明している節を読むのがよいでしょう。

1.1.1 Python スクリプトの実行方法

Mac OS X 上で Python を始めるには PythonIDE 統合開発環境に触れてみるのが最良の方法です。章 1.3 を見るか、IDE が起動しているならヘルプメニューから呼び出せる Apple ヘルプビューア書類の IDE 入門を

読みながら IDE に触れてみてください。

Python を Terminal ウィンドウのコマンドラインや Finder から起動したいなら、まずはスクリプトを書くエディタが必要になります。Mac OS X には、**vi** や **emacs** のような、様々な標準の UNIX コマンドラインエディタがついてきます。もっと Mac らしいエディタを使いたければ、 Bare Bones Software (http://www.barebones.com/products/bbedit/index.shtml) の **BBEdit** か **TextWrangler** を選ぶと良いでしょう。フリーウェアの **BBEdit Lite** のサポートは表向き中断されていますが、まだ入手できます。**AppleWorks** や、ASCII 形式でファイルを保存できるその他のワードプロセッサでもかまいませんが、**TextEdit** はいけません: '.rtf' フォーマットで保存を行うからです。

Terminal のウィンドウから自作のスクリプトを起動するには、シェルの検索パスで '/usr/local/bin' が '/usr/bin' よりも前にくるようにしておかねばなりません。というのも、'/usr/bin' には Apple 提供の Python が入っているからです (Mac OS X 10.2.4 時点では Python 2.2 です)。

Finder から自作スクリプトを実行するには、2 つのやり方があります:

- プログラムを **PythonLauncher** にドラッグします。
- Finder の情報ウィンドウで、作成したスクリプト (又はその他の '.py' スクリプト) を開くためのデフォルトのアプリケーションとして **PythonLauncher** を選択して、スクリプトをダブルクリックします。

PythonLauncher には様々な設定があり、スクリプトの起動方法を制御できるようになっています。オプションキーを押しながらドラッグすると、起動時に設定を変更できます。全体的な設定を変えたければ Perferences メニューを使ってください。

1.1.2 GUI つきのスクリプトの実行

Mac OS X には、一つだけ知っておかねばならないクセがあります: Aqua ウィンドウマネージャとやり取りするような (すなわち、何らかの GUI を持つような) プログラムは、特殊な方法で起動せねばならないのです。GUI を持ったスクリプトを実行するには **python** の代わりに **pythonw** を使ってください。

1.1.3 設定

MacPython では、標準的な UNIX の Python が使う PYTHONPATH のような環境変数全てに従います。しかし、Finder から起動したプログラムでは、こうした変数に対して標準的でない振る舞いを見せます。これは、Finder が起動時に '.profile' や '.cshrc' を読まないためです。Finder から起動するプログラム向けに環境変数を設定したければ、'~/.MacOSX/environment.plist' ファイルを作成してください。詳しくは Apple Technical Document QA1067 を参照してください。

Package Manager を使うと、追加の Python パッケージをとても簡単にインストールできます。詳しくは MacPython ヘルプを参照してください。

1.2 MacPython-OS9 の取得とインストール

最新のリリースバージョンや実験的な最新版は、Jack Jansen が運営している MacPython のページ: http://homepages.cwi.nl/~jack/macpython.html で探すとよいでしょう。

最新版の説明は配布物中の 'README' を参照してください。

MacPython-OS9 は Mac OS X 上で正常に動作し、クラシック環境ではなくネイティブモードで動きます。とはいえ、CFM ベースの Python を使いたいという特別な理由がない限り、Mac OS X 上では MacPython-OSX を使いましょう。

1.2.1 対話型インタプリタを使う

Python のドキュメントでよく見るような対話型インタプリタは、**PythonInterpreter** アイコンをダブルクリックして起動します。落下する 16 トン の分銅ようなアイコンです。起動すると、バージョン情報と '>>> ' プロンプトが表示されます。標準ドキュメントに書かれているインタプリタとまったく同じように使えます。

1.2.2 Python スクリプトの走らせ方

既存の Python スクリプトを起動するにはいくつかやり方があります; よく使われるのは 「ドラッグ＆ドロップ」 や 「ダブルクリック」 です。他にも IDE 中で起動したり (1.3 節参照)、 AppleScript 中で起動するという方法があります。

ドラッグ＆ドロップする

一番簡単な Python スクリプト起動方法の一つは、「ドラッグ＆ドロップ」です。これは、ファインダでテキストファイルをワードプロセッサのアイコンの上に「ドラッグ」し、そこに「ドロップ」して起動させるのと同じです。「ドラッグ＆ドロップ」で Python スクリプトを起動する場合、**IDE** や **Idle** ではなく **PythonInterpreter** を使うように気をつけてください。**IDE** や **Idle** では、後で述べるように違った挙動になってしまいます。

うまく行っていない場合をいくつか示します:

- **PythonInterpreter** の上にスクリプトをドロップした後、ウィンドウがフラッシュしてから消える場合。おそらく設定に問題があります; **PythonInterpreter** は実行完了の後すぐに終了するよう設定されているのに、スクリプトでは何かを出力して、しばらくの間テキストを表示したままにしようとしているからでしょう。修正するには 1.2.5 節を参照してください。
- スクリプトのアイコンを **PythonInterpreter** の上にかざしても **PythonInterpreter** がハイライト表示にならない場合。おそらくクリエータコードと文書のタイプがセットされていない (あるいは誤ってセットされている) ためです。 – Mac 以外のコンピュータからファイルを持ってきた場合に時々起きます。詳しくは 1.2.2 節を参照してください。

クリエータをセットしてダブルクリックする

起動したいスクリプトが適当なクリエータコードとファイルタイプを持っていれば、単にダブルクリックするだけでスクリプトを起動できます。ファイルを「ダブルクリック可能」にするには、ファイルタイプを 'TEXT' に、クリエータコードを 'Pyth' にせねばなりません。

クリエータコードとファイルタイプのセットは、IDE を使う (1.3.2 節および 1.3.4 節参照) か、Python モードを持つエディタ (**BBEdit** など) を使う – 1.2.4 を参照してください – か、その他諸々の Mac 用ユーティリティを使います。とはいえ、最近は、 MacPython の配布物に、Python 用の適切なファイルタイプとクリエータコードを設定できるようにするスクリプト ('fixfiletypes.py') が入るようになりました。

'fixfiletypes.py' スクリプトは、指定したディレクトリに対してファイルタイプとクリエータコードを変えます。'fixfiletypes.py' は以下のように使います:

1. スクリプトを、MacPython 配布物の 'Mac' フォルダ下の 'scripts' フォルダに置きます。
2. 修正したいスクリプトは全てひとつのフォルダに入れます。他のファイルは入れないようにします。
3. 'fixfiletypes.py' アイコンをダブルクリックします。
4. 修正したいファイルの入ったフォルダを選んで "Select current folder" ボタンを押します。

1.2.3 コマンドライン引数をシミュレートする

MacPython-OS9 でコマンドライン引数をシミュレートするには 2 つの方法があります。

1. インタプリタオプションを使う場合

   - スクリプトを起動する時、オプションキーを押しながら行います。Python インタプリタオプションのダイアログボックスが現われます。
   - “Set UNIX-style command line..” ボタンをクリックします。
   - “Argument” フィールドに引数をタイプします。
   - “OK” をクリックします。
   - “Run” をクリックします。

2. ドラッグ&ドロップを使う場合。

   スクリプトをアプレットとして保存した場合 (1.3.4 節参照)、「ドラッグ&ドロップ」でコマンドライン引数もシミュレートできます。この場合、アプレットにドロップしたファイルの名前が `sys.argv` に追加され、ドロップしたファイル名はスクリプト側からはコマンドラインでタイプされたかのように見えます。UNIX システムと同様、`sys.srgv` の最初の要素はアプレット自体へのパスになります。残りの要素はアプレットにドロップしたファイルの名前です。

1.2.4 Python スクリプトを作成する

Python スクリプトは単純なテキストファイルなので、テキストファイルが作れるならどのような方法でも作成可能です。しかし、他にも機能のある特殊な専用ツールもあります。

エディタで作成する

MSWord や **AppleWorks** などのワードプロセッサプログラムでテキストファイルを作成することもできますが、ファイルを “ASCII” か “プレーンテキスト” で保存するよう気を付けてください。**TextEdit** も使用できますが、保存する前に「フォーマット」メニューで「標準テキストにする」コマンドを実行してください。

Python モード付きのエディタ

いくつかのテキストエディタには、Python スクリプトを作成するための機能が付加されています。例えば、コードを読みやすくするための Python キーワードへの色付け表示、モジュールのブラウズ、組み込みデバッガといった機能です。こうしたエディタには、**Alpha**、**Pepper**、 **BBedit**、MacPython IDE(節 1.3) があります。[1]

BBedit

BBEdit でスクリプトを作成しているなら、保存したファイルをダブルクリックして起動できるように、クリエータコードを Python に指定したいと考えるでしょう。

- **BBEdit** を立ち上げます。
- 「Edit」メニューから「Preferences」を選択します。

[1] 訳注：他にも **mi**(`http://www.mimikaki.net/`) などで Python モードをサポートしています。

- スクロールリストから「File Types」を選択します。
- 「Add...」ボタンをクリックし、 MacPython 配布物ディレクトリの下に移動して、**PythonInterpreter** を選び、「open」をクリックします。
- 設定パネルの “Save” ボタンをクリックします。

1.2.5 設定を行なう

MacPython 配布物には、**EditPythonPrefs** というアプレットがついてきます。このアプレットを使うと、MacPython の環境をカスタマイズして、自分の作業方法にあわせる手助けをしてくれます。

EditPythonPrefs

EditPythonPrefs を使うと、 Python の振る舞いを自分の好みに合わせられます。**EditPythonPrefs** には二つの用途があります。ひとつは環境設定を行うという普通の使い方、もうひとつは Python のエンジンをドロップして、使いたいインタプリタのバージョンを制御するというものです。後者は、たとえば普段はプログラムが正常終了した時には出力ウィンドウを閉じるようにしておきたいけれども、出力ウィンドウを閉じないような **PythonInterpreter** のコピーをもうひとつ作っておきたいような場合に便利です。

デフォルトの設定を変更するには、 **EditPythonPrefs** をダブルクリックします。インタプリタのどれかひとつの設定だけを変更したければ、インタプリタを **EditPythonPrefs** の上にドロップします。同じやり方を使えば、**EditPythonPrefs** を使って **Python IDE** や作成したアプレットの設定を変更できます – **??** 節や 1.3.4 節を参照してください。

モジュール検索パスにモジュールを追加する

`import` 文を実行すると、Python はそのモジュールを `sys.path` で定義された場所に探しに行きます。Mac 上で `sys.path` を編集するには、**EditPythonPrefs** を起動して、一番上にある大きめのフィールドにパスを入力します (1 行 に 1 つづつ書きます)。

MacPython ではメインの Python ディレクトリを定義しているので、フォルダをメイン Python ディレクトリに追加するのが最も簡単な方法です。「My Folder」という名前のフォルダを作って自作のスクリプトを入れ、メインの Python ディレクトリに追加したければ、‘`$(PYTHON):My Folder`’ を新たな行に入力します。

OS 9 かそれ以前の OS でデスクトップを追加したければ、‘`StartupDriveName:Desktop Folder`’ を新たな行として入力します。

デフォルトの起動オプション

EditPythonPrefs ダイアログボックスの 「Default startup options...」ボタンを押すと、沢山のオプションが現れます。オプションの中には、スクリプトの終了後に「Output」ウィンドウを開いたままにしておけるようにしたり、スクリプトの実行終了後に対話モードに入れるようにしたりできるものがあります。後者はスクリプト実行中に生成されたオブジェクトを調べたい場合にとても便利です。

1.3 統合開発環境

Python IDE(統合開発環境) は独立したアプリケーションで、Python コードのテキストエディタや、クラスブラウザ、グラフィカルデバッガなどとして動作します。

1.3.1 “Python Interactive” ウィンドウを使う

先に紹介した「ドラッグ＆ドロップ」の手法を使えないことを除き、このウィンドウは **PythonInterpreter** と同じように使えます。プログラムを **Python IDE** の上にドロップすると、プログラムを起動する代わりに、ファイルを別のスクリプトウィンドウ上に開きます (その後で手動で実行できます – 1.3.3 節を参照してください)。

1.3.2 Python スクリプトを書く

Python IDE は、対話的に使うだけでなく、Python プログラムを書き上げたり、順次保存したりでき、全体や一部分の実行もできます。

「File」メニューの適当なメニューアイテムを選択すれば新たにスクリプトを作成したり、前に保存したスクリプトを開いたり、現在開いているスクリプトを保存したりできます。 Python スクリプトを **Python IDE** の上にドロップすると、ファイルを編集用に開きます。

スクリプトを **Python IDE** で開こうにも「Open」ダイアログボックスで場所を特定できない場合や、「Can’t open file of type ...」のようなエラーメッセージが出る場合は、1.2.2 節を参照してください。

Python IDE はスクリプトを保存する際にクリエータコードの設定を使います。この設定は、ドキュメントウィンドウの一番右上の小さな黒い三角形をクリックし、「save options」を選べば操作できます。デフォルトでは、ファイルの **Python IDE** をクリエータコードにして保存します。従って、ファイルのアイコンをダブルクリックするとファイルを編集用に開きます。この動作を変更して、**PythonInterpreter** で開いて実行するようしたいと思う場合もあるでしょう。そうするには、単に「save options」から「Python Interpreter」を選ぶだけです。このオプションはアプリケーションではなく**ファイル** に関連付けられているので注意してください。

1.3.3 統合開発環境の中からスクリプトを実行する

Python IDE の最前面のウィンドウで全部実行 (run all) ボタンを押すと、そのウィンドウのスクリプトを実行できます。しかし、仮に Python の習慣通りに ‘`if __name__ == "__main__":`’ と書いても、スクリプトはデフォルトでは「__main__」に **ならない** ことに注意しておきましょう。そういう風に動作させるには、ドキュメントウィンドウの一番右上の小さな黒い三角形から、“Run as __main__” オプションを選ばねばなりません。このオプションはアプリケーションではなく**ファイル** に関連付けられているので注意してください。とはいえ、このオプションは保存後もそのまま**残ります**。止めたければ、再度このオプションを選んでください。

1.3.4 “Save as” と “Save as Applet” の違い

Python スクリプトを書いたら、ファイルを 「アプレット」としても保存できます (“File” メニューの “Save as applet” を選びます) 。アプレットとして保存すると、ファイルやフォルダをスクリプトにドロップすることで、コマンドライン引数で渡すのと同じようにスクリプトにファイルやフォルダを渡せるという、大きな利点があります。ただし、アプレットを今までのファイルに上書きせず、別のファイルとして保存するように気をつけてください。アプレットとして保存したファイルは二度と編集できないからです。

「ドラッグ＆ドロップ」でアプレットに渡した項目にアクセスするには、標準的な `sys.argv` の動作を使います。詳しくは Python の標準ドキュメントを参照してください。

スクリプトをアプレットとして保存しても、 Python がインストールされていないシステムでは実行できないので注意してください。

二章

MacPython モジュール

このドキュメントで記述されている次のモジュールは、いずれも Macintosh でのみ利用可能です。

mac	os モジュールの実装
macpath	MacOS のパス操作関数
macfs	FSSpec、エイリアスマネージャ、**finder** エイリアス、標準ファイルパッケージのサポート。
ic	インターネット設定へのアクセス。
MacOS	Mac OS 固有のインタープリタ機能へのアクセス。
macostools	ファイル操作を便利にするルーチン集。
findertools	**finder** の Apple Events インターフェースのラッパ。
EasyDialogs	基本的な Macintosh ダイアログ。
FrameWork	対話型アプリケーション・フレームワーク
autoGIL	イベントループ中のグローバルインタープリタの取り扱い

2.1 mac — os モジュールの実装

このモジュールは、標準モジュール os でサポートされるのと同様の機能を、Mac OS 9 オペレーティングシステムに依存した機能として実装しています。このモジュールを利用する一番良い使い方は posix モジュール経由で使うことです。このモジュールは MacPython-OS9 にのみ存在し、MacPython-OSX 上では posix を利用します。

このモジュールでは次の関数が使えます。chdir()、close()、dup()、fdopen()、getcwd()、lseek()、listdir()、mkdir()、open()、read()、rename()、rmdir()、stat()、sync()、unlink()、write()、そして例外 error も定義されています。stat() により返される時間は浮動小数点数で、 MacPython-OS9 で使われる他の時間の値と同様です。

2.2 macpath — MacOS のパス操作関数

このモジュールは os.path モジュールの Macintosh 実装です。このモジュールで、os.path への最も汎用性のあるアクセスができます。os.path のドキュメントに関しては、*Python Library Reference* を参照してください。

次の関数がこのモジュールで利用できます。normcase()、normpath()、isabs()、join()、split()、isdir()、isfile()、walk()、exists()。os.path で利用できる他の関数については、ダミーの関数として相当する物が利用できます。

2.3 macfs — 様々なファイルシステム関連のサービス

リリース 2.3 以降で撤廃された仕様です。 macfs モジュールは旧式のものです。FSSpec、FSRef、Alias の

操作には、Carbon.File または Carbon.Folder モジュールを、ファイルダイアログには EasyDialogs を使ってください。また、このモジュールは UFS パーティションでは正確には動作しないことがわかっています。

このモジュールでは、Macintosh の FSSpec の操作、エイリアスマネージャ、**finder** エイリアス、および標準ファイルパッケージへのアクセスを提供しています。

関数やメソッドが *file* 引数をとるようになっている場合、引数は常に次の 3 つ、(1) Macintosh のフルパス名あるいは部分パス名、(2) FSSpec オブジェクト、(3) *Inside Macintosh: Files* で解説されている 3 要素のタプル (*wdRefNum*, *parID*, *name*) のうちのいずれかになります。

FSSpec は、実在しないファイルでも、実在するフォルダの下に配置されていることになっている限り表現できます。MacPython ではパス名も同じように扱えますが、 unix ベースの Python ではパス名と FSRefs の挙動が異なるため扱えません。詳しくは Apple のドキュメントを参照してください。

エイリアスと標準ファイルパッケージの詳細も Apple のドキュメントに書かれています。

FSSpec (*file*)
指定したファイルに対する FSSpec オブジェクトを作成します。

RawFSSpec (*data*)
FSSpec の C 構造体の生データが入った文字列から FSSpec オブジェクトを作成します。主に FSSpec 構造体をネットワーク経由で得る場合に便利です。

RawAlias (*data*)
Alias の C 構造体の生データが入った文字列から Alias オブジェクトを作成します。主に Alias 構造体をネットワーク経由で得る場合に便利です。

FInfo ()
ゼロで埋めた FInfo オブジェクトを作成します。

ResolveAliasFile (*file*)
エイリアスファイルを解決します (オリジナルのファイルとの対応を取ります)。 (*fsspec*, *isfolder*, *aliased*) からなる 3 要素のタプルを返します。*fsspec* はエイリアス解決によって得られた FSSpec オブジェクトです。*fsspec* がフォルダを指している場合、*isfolder* は true になります。*fsspec* がエイリアスを指している場合、*aliased* は true になります (それ以外の場合には、*fsspec* はファイル自体の FSSpec になります)。

StandardGetFile ([*type, ...*])
標準的イアログ 「入力ファイルを開く」をユーザに提示します。オプションとして、ユーザが選択できるファイルを制限するために 4 文字の文字列で表されたファイルタイプ指定子を 4 つまで渡せます。この関数は、FSSpec オブジェクトとユーザがキャンセルしないでダイアログを閉じたかどうかを示すフラグを返します。

PromptGetFile (*prompt*[, *type, ...*])
StandardGetFile() とほぼ同じですが、ダイアログの一番上に表示されるプロンプトを指定できます。

StandardPutFile (*prompt*[, *default*])
標準ダイアログ 「出力ファイルを開く」をユーザに提示します。*prompt* はプロンプト文字列です。*default* はオプションの引数で、出力ファイル名の初期値を指定します。この関数は、FSSpec オブジェクトとユーザがキャンセルしないでダイアログを閉じたかどうかを示すフラグを返します。

GetDirectory ([*prompt*])
非標準的ダイアログ「ディレクトリを選ぶ」をユーザに提示します。このダイアログでは、選択したいディレクトリを開いておいてから、“select current directory” ボタンをクリックします。*prompt* はプロンプト文字列で、ダイアログの一番上に表示されます。この関数は、FSSpec オブジェクトとユー

ザがキャンセルしないでダイアログを閉じたかどうかを示すフラグを返します。

SetFolder([*fsspec*])

ファイル選択ダイアログを提示する時に最初に表示されるフォルダを設定します。*fsspec* には、フォルダそのものではなく、フォルダ内のファイルを指定します (実在しないファイルでもかまいません)。引数を渡さない場合は、フォルダはカレントディレクトリ、つまり `os.getcwd()` で返されるディレクトリに設定されます。

System 7.5 以降では、ユーザは「一般設定」コントロールパネルで標準ファイルパッケージの振る舞いを変更でき、設定によってはこの関数の呼び出しが無効化されるので注意してください。

FindFolder(*where, which, create*)

ゴミ箱や初期設定フォルダなど、Mac OS が管理している「特別な」フォルダの位置を検索します。*where* は検索したいディスク、*which* は検索したいフォルダを指定する 4 文字の文字列です。*create* を設定すると、該当するフォルダが存在しない場合に新たに生成します。(*vrefnum*, *dirid*) からなるタプルを返します。

where と *which* に指定できる定数は、標準モジュール *Carbon.Folders* にあります。

NewAliasMinimalFromFullPath(*pathname*)

与えられたファイルを指す最小限の `alias` オブジェクトを返します。ファイルはフルパス名で与えなければなりません。これは存在しないファイルを指す `Alias` を作成する唯一の方法です。

FindApplication(*creator*)

4 文字のクリエータコード *creator* を持ったアプリケーションの位置を探し出します。この関数はアプリケーションを指す `FSSpec` オブジェクトを返します。

2.3.1 FSSpec オブジェクト

data

FSSpec オブジェクトの生データです。他のアプリケーションに渡すといった場合に適した形式です。

as_pathname()

`FSSpec` オブジェクトの表すファイルのフルパス名を返します。

as_tuple()

`FSSpec` オブジェクトで記述されたファイルの情報を、(*wdRefNum*, *parID*, *name*) のタプルで返します。

NewAlias([*file*])

この FSSpec で記述されたファイルのエイリアスオブジェクトを作成します。省略可能な *file* パラメータを渡すと、エイリアスはそのファイルに対する相対指定で作成され、その他の場合は絶対指定となります。

NewAliasMinimal()

このファイルを指す最小限のエイリアス情報を作成します。

GetCreatorType()

このファイルの 4 文字のクリエータとタイプを返します。

SetCreatorType(*creator, type*)

このファイルに 4 文字のクリエータとタイプを設定します。

GetFInfo()

ファイルのファインダ情報を示す `FInfo` オブジェクトを返します。

SetFInfo(*finfo*)

ファイルのファインダ情報を *finfo*(`FInfo` オブジェクト) で与えた値に設定します。

GetDates()
作成日、修正日、バックアップ日を意味する 3 つの浮動小数点数からなるタプルを返します。

SetDates(*crdate, moddate, backupdate*)
ファイルの作成日、修正日、バックアップ日を設定します。各々の値は Python が時刻の表現に使っている標準の浮動小数点型です。

2.3.2 エイリアスオブジェクト

data
エイリアス (Alias) レコードの生データです。リソースへの書き込みや他のプログラムへの転送に適した形式です。

Resolve([*file*])
エイリアスを解決します。エイリアスが相対エイリアスとして作成されている場合は、どこからの相対かを示すファイルを渡さねばなりません。エイリアスが指し示すファイルの FSSpec と、`Alias` オブジェクト自体が検索処理中に変更されたかどうかを示すフラグを返します。ファイルは実在しないが、ファイルまでのパスは実在する場合、有効な FSSpec を返します。

GetInfo(*num*)
C のルーチン `GetAliasInfo()` へのインタフェースです。

Update(*file*[, *file2*])
エイリアスを、*file* に指定したファイルを指すように更新します。*file2* を指定していれば、相対エイリアスを作成します。

今のところ、リソースは `Alias` オブジェクトとして直接操作できません。そのため、`Update()` を呼んだ後か、`Resolve()` でエイリアスに変更があったと分かった後は、Python プログラム側で `Alias` オブジェクトから `data` の値を取りだし、リソースを修正しておく責任があります。

2.3.3 FInfo オブジェクト

各フィールドの詳しい説明は *Inside Macintosh: Files* を参照してください。

Creator
ファイルの 4 文字のクリエータコードです。

Type
ファイルの 4 文字のタイプコードです。

Flags
ファイルのファインダフラグで、 16 ビットの整数で表現されています。*Flags* のビット値は、標準モジュール `MACFS` で定義されています。

Location
フォルダ内でファイルのアイコンが配置されている場所を示す Point 値です。

Fldr
ファイルが入っているフォルダ (を整数で表したもの) です。

2.4 ic — インターネット設定へのアクセス

このモジュールでは、Macintosh のインターネット設定 (Internet Config) パッケージへのアクセスを提供しています。このパッケージにはインターネットプログラムの設定、例えばメールアドレス、デフォルトのホームページなどが収録されています。それ以外にも、インターネット設定には Macintosh のクリエータ

／タイプとファイル名の拡張子との対応付けや、ファイルの転送方法 (バイナリ、アスキーなど) に関する情報が収められています。MacOS 9 以降では、このモジュールは「インターネット」という名前のコントロールパネルになりました。

このモジュールには、icglue という低水準の関連モジュールがあり、インターネット設定への基本的なアクセス機能を提供しています。この低水準のモジュールはドキュメント化されていませんが、各ルーチンの docstring にはパラメタの説明があり、ルーチン名には Internet Config に対する Pascal や C のインタフェースと同じ名前を使っているので、このモジュールが必要な場合には標準の IC プログラマドキュメントを利用できます。

ic モジュールでは、例外 error と、インターネット設定から生じる全てのエラーコードに対するシンボル名を定義しています。詳しくはソースコードを参照してください。

exception error

ic モジュール内部でエラーが生じたときに送出される例外です。

ic モジュールは以下のクラスと関数を定義しています：

class IC([*signature*[, *ic*]])

インターネット設定オブジェクトを作成します。*signature* は、IC の設定に影響を及ぼす可能性のある現在のアプリケーションを表す 4 文字のクリエータコード (デフォルトは'Pyth') です。オプションの引数 *ic* は低水準モジュールであらかじめ作成しておいた icglue.icinstance で、別の設定ファイルなどから設定を得る場合に便利です。

launchurl(*url*[, *hint*])
parseurl(*data*[, *start*[, *end*[, *hint*]]])
mapfile(*file*)
maptypecreator(*type, creator*[, *filename*])
settypecreator(*file*)

これらの関数は、後述する同名のメソッドへの「ショートカット」です。

2.4.1 IC オブジェクト

IC オブジェクトはマップ型のインターフェースを持っているので、メールアドレスの取得は単に *ic*['MailAddress'] でできます。値の代入もでき、設定ファイルのオプションを変更できます。

このモジュールは各種のデータ型を知っていて、IC 内部の表現を「論理的な」Python データ構造に変換します。ic モジュールを単体で実行すると、テストプログラムが実行されて IC データベースにある全てのキーと値のペアをリスト表示するので、文書代わりになります。

モジュールがデータの表現方法を推測できなかった場合、data 属性に生のデータが入った ICOpaqueData 型のインスタンスを返します。この型のオブジェクトも代入に利用できます。

IC には辞書型のインターフェースの他にも以下のようなメソッドがあります。

launchurl(*url*[, *hint*])

与えられた URL を解析し、適切なアプリケーションを起動して URL を渡します。省略可能な *hint* は、'mailto:' などのスキーム名で、不完全な URL はこのスキームにあわせて補完します。*hint* を指定していない場合、不完全な URL は無効になります。

parseurl(*data*[, *start*[, *end*[, *hint*]]])

data の中から URL を検索し、URL の開始位置、終了位置、URL そのものを返します。オプションの引数 *start* と *end* を使うと検索範囲を制限できます。例えば、ユーザーが長いテキストフィールドをクリックした場合に、このルーチンにテキストフィールド全体とクリック位置 *start* を渡すことで、ユーザーがクリックした場所にある URL 全体を返させられます。先に述べたように、*hint* はオプションで、不完全な URL を補完するためのスキームです。

mapfile(*file*)

file に対するマッピングエントリを返します。*file* にはファイル名か `macfs.FSSpec()` の戻り値を渡せます。実在しないファイルであってもかまいません。

マッピングエントリは (*version*, *type*, *creator*, *postcreator*, *flags*, *extension*, *appname*, *postappname*, *mimetype*, *entryname*) からなるタプルで返されます。*version* はエントリーのバージョン番号、*type* は 4 文字のファイルタイプ、*creator* は 4 文字のクリエータタイプ、*postcreator* はファイルのダウンロード後にオプションとして起動され、後処理を行うアプリケーションの 4 文字のクリエータコードです。*flags* は、転送をバイナリで行うかアスキーで行うか、などの様々なフラグビットからなる値です。*extension* はこのファイルタイプに対するファイル名の拡張子、*appname* はファイルが属するアプリケーションの印字可能な名前、*postappname* は後処理用アプリケーション、*mimetype* はこのファイルの MIME タイプ、最後の *entryname* はこのエントリの名前です。

maptypecreator(*type, creator*[, *filename*])

4 文字の *type* と *creator* コードを持つファイルに対するマッピングエントリを返します。(クリエータが `'????'` であるような場合に) 正しいエントリが見つかりやすいようにオプションの *filename* を指定できます。

マッピングエントリーは *mapfile* と同じフォーマットで返されます。

settypecreator(*file*)

実在のファイル *file* に対して、拡張子に基づいて適切なクリエータとタイプを設定します。*file* の指定は、ファイル名でも `macfs.FSSpec()` の戻り値でもかまいません。変更は Finder に通知されるので、Finder 上のアイコンは即座に更新されます。

2.5 MacOS — Mac OS インタプリタ機能へのアクセス

このモジュールは、Python インタプリタ内の MacOS 固有の機能に対するアクセスを提供します。例えば、インタプリタのイベントループ関数などです。十分注意して利用してください。

モジュール名が大文字で始まることに注意してください。これは昔からの約束です。

runtimemodel

`'carbon'` か `'macho'` のいずれかです。現在利用している Python が Mac OS X および Mac OS 9 互換性をもつ CarbonLib 形式、あるいは Mac OS X のみの Mach-O 形式をのどちらであるか判断できます。Python の初期のバージョンでは、値がさらに古い Mac OS 8 ランタイムモデル用の`'ppc'` である事があります。

linkmodel

インタープリタがどのような方法でリンクされているかを返します。拡張モジュールがリンクモデル間で非互換性かもしれない場合、パッケージはより多くの適切なエラーメッセージを伝えるためにこの情報を使用することができます。値は静的リンクした Python は `'static'`、Mac OS X framework で構築した Python は `'framework'`、標準の unix 共有ライブラリ (shared library) で構築された Python は `'shared'`、Mac OS 9 互換 Python では `'cfm'` となります。

exception Error

MacOS でエラーがあると、このモジュールの関数か、Mac 固有なツールボックスインターフェースモジュールから、この例外が生成されます。引数は、整数エラーコード (`OSErr` 値) とテキストで記述されたエラーコードです。分かっている全てのエラーコードのシンボル名は、標準モジュール `macerrors` で定義されています。

SetEventHandler(*handler*)

内部のインタプリタループでは、`ScheduleParams()` で止めないかぎり、Python は時々イベント

をチェックします。イベントがある場合は、この関数を使うと、Python イベントハンドラ関数を渡せます。イベントはパラメータとして渡され、イベントが完全に処理された場合は、ハンドラ関数は非ゼロを返さなくてはなりません。それ以外はイベント処理は継続されます (例えば、イベントをコンソールウィンドウパッケージ渡すなど)。

イベントハンドラをクリアするには、パラメータなしで `SetEventHandler()` を呼び出します。既にイベントハンドラがセットされているのに、さらにセットしようとするとエラーになります。

有効性：MacPython-OS9

SchedParams([*doint*[, *evtmask*[, *besocial*[, *interval*[, *bgyield*]]]]])

これはインタプリタの内部ループイベントハンドラに影響を与えます。*Interval* は、インタプリタがどれだけの頻度 (浮動小数点数の秒で表わされる) でイベント処理コードに入るかを指定します。真なら *doint* は割り込み (コマンドドット) チェックが行われます。*evtmask* はインタプリタに、イベントをマスクして (再描画、他のアプリケーションに切り替わるマウスクリックなど) イベント処理するよう指示します。*besocial* フラグは、他のプロセスが動作するチャンスを与えます。Python が最前面で動いている時は、最小限の実行時間が割り当てられ、Python が背景にある場合は、*interval* 当りに *bgyield* 秒が与えられます。

全てのパラメータはオプションで、現在の値がデフォルト値となります。この関数で返される値は、これらのオプションの既存の値からなるタプルです。デフォルトの初期値は、全ての処理がオンで、チェックは 1/4 秒毎、バックグラウンドで動作している場合はプロセッサも 1/4 秒毎に割り当てられます。

最も一般的な使用ケースは完全にインタープリタメインループ中の処理をできなくするために `SchedParams(0, 0)` を呼ぶことです。

有効性：MacPython-OS9

HandleEvent(*ev*)

イベントレコード *ev* を Python のイベントループに渡す、というよりは、`sys.stdout` ウィンドウ (Python をビルドしたコンパイラにもとづいて) のハンドラに渡されることになります。こうすると、Python プログラムが独自のイベント処理を行え、コマンドピリオドやウィンドウの切り替えが行えます。

`SetEventHandler()` でセットしたイベントハンドラからこの関数を呼びだそうとすると、例外が生じます。

有効性：MacPython-OS9

GetErrorString(*errno*)

MacOS のエラーコード *errno* のテキスト表現を返します。

splash(*resid*)

この関数は、*resid* で与えた DLOG リソースの内容で、スプラッシュウィンドウを画面に表示します。引数なしで呼びだすと、スプラッシュ画面を取り除きます。拡張モジュールをたくさんロードさせる前に、初期化のタイミングでアプレットにスプラッシュ画面を表示させたいときに、この関数が便利でしょう。

有効性：MacPython-OS9

DebugStr(*message* [, *object*])

Mac OS 9 上では、メッセージ *message* を出してローレベルデバッガに入ります。オプションの *object* 引数は使われませんが、デバッガから内容を容易に検査することができます。Mac OS X 上では 文字列が単に stderr に印字されます。

この関数を使うときは十分気を付けてください。MacsBug などのローレベルデバッガがインストールされていない場合は、システムがクラッシュしてしまいます。この関数は主に Python 拡張モジュー

ルの開発者のために用意されています。

SysBeep()
ベルを鳴らします。

GetTicks()
システム起動時からのチック数 (clock ticks、1/60 秒) を得ます。

GetCreatorAndType(*file*)
2 つの 4 文字の文字列としてファイルクリエータおよびファイルタイプを返します。*file* 引数はパスもしくは、FSSpec、FSRef オブジェクトを与える事ができます。

SetCreatorAndType(*file, creator, type*)
ファイルクリエータおよびファイルタイプを設定します。*file* 引数はパスもしくは、FSSpec、FSRef オブジェクトを与える事ができます。*creator* と *type* は 4 文字の文字列が必要です。

openrf(*name* [, *mode*])
ファイルのリソースフォークを開きます。引数は組み込み関数 open() と同じです。返されたオブジェクトはファイルのように見えるかもしれませんが、これは Python のファイルオブジェクトではありませんので扱いに微妙な違いがあります。

WMAvailable()
現在のプロセスが動作しているウィンドウマネージャにアクセスします。例えば、Mac OS X サーバー上、あるいは SSH でログインしている、もしくは現在のインタープリタがフルブローンアプリケーションバンドル (fullblown application bundle) から起動されていない場合などのような、ウィンドウマネージャが存在しない場合は False を返します。

Mac OS 9 上ではこの関数はつねに True を返します。

2.6 macostools — ファイル操作を便利にするルーチン集

このモジュールには、Macintosh 上でのファイル操作を便利にするためのルーチンがいくつか入っています。全てファイルパラメタは，パス名か FSRef または FSSpec オブジェクトで指定できます。このモジュールは、リソースフォークつきファイル (forked file) をサポートするファイルシステムを想定しているので、UFS パーティションに使ってはなりません.

macostools モジュールでは以下の関数を定義しています。

copy(*src, dst*[, *createpath*[, *copytimes*]])
ファイル *src* を *dst* へコピーします。*createpath* が真なら、必要に応じて *dst* に至るまでのフォルダを作成します。このメソッドはデータとリソースフォーク，そしていくつかのファインダ情報 (クリエータ、タイプ、フラグ) をコピーします。オプションの *copytypes* を指定すると，作成日、修正日、バックアップ日の情報のコピー (デフォルトではコピーします) を制御できます。カスタムアイコン、コメント、アイコン位置はコピーされません。

copytree(*src, dst*)
src から *dst* へ再帰的にファイルのツリーをコピーします。必要に応じてフォルダを作成してゆきます．*src* と *dst* はパス名で指定しなければなりません。

mkalias(*src, dst*)
src を示すファインダエイリアス *dst* を作成します。

touched(*dst*)
ファイル *dst* のクリエータやタイプなどのファインダ情報が変わったことをファインダに知らせます。ファイルはパス名か FSSpec で指定できます。この呼び出しは，ファインダにアイコンを再描画させるよう命令します。

BUFSIZ
copy に用いるバッファサイズで、デフォルトは 1 メガバイトです。

Apple のドキュメントでは，ファインダエイリアスの作成プロセスを規定していません．そのため，mkalias() で作成したエイリアスが互換性のない振る舞いをする可能性があるので注意してください。

2.7 findertools — finder の Apple Events インターフェース

このモジュールのルーチンを使うと、Python プログラムからファインダが持ついくつかの機能へアクセスできます。これらの機能はファインダへの AppleEvent インターフェースのラッパとして実装されています。全てのファイルとフォルダのパラメータは、フルパス名、あるいは FSRef か FSSpec オブジェクトで指定できます。

findertools モジュールは以下の関数を定義しています。

launch (*file*)
ファインダに *file* を起動するように命令します。起動が意味するものは *file* に依存します。アプリケーションなら起動しますし、フォルダなら開かれ、文書なら適切なアプリケーションで開かれます。

Print (*file*)
ファインダにファイルを印刷するよう命令します。実際の動作はファイルを選択し、ファインダのファイルメニューから印刷コマンドを使うのと同じです。

copy (*file, destdir*)
ファインダにファイルかフォルダである *file* をフォルダ *destdir* にコピーするよう命令します。この関数は新しいファイルを示す Alias オブジェクトを返します。

move (*file, destdir*)
ファインダにファイルかフォルダである *file* をフォルダ *destdir* に移動するように命令します。この関数は新しいファイルを示す Alias オブジェクトを返します。

sleep()
マシンがサポートしていれば、ファインダに Macintosh をスリープさせるよう命令します。

restart()
ファインダに、マシンを適切に再起動するよう命令します。

shutdown()
ファインダに、マシンを適切にシャットダウンするよう命令します。

2.8 EasyDialogs — 基本的な Macintosh ダイアログ

EasyDialogs モジュールには、 Macintosh で単純なダイアログ操作を行うためのルーチンが入っています。全てのルーチンは、オプションとしてリソース ID パラメタ *id* をとります。デフォルトの DLOG のリソース (タイプとアイテムナンバの両方) が一致するようなダイアログがあれば、*id* を使ってダイアログ操作に使われるダイアログオブジェクト情報を上書きできます。詳細はソースコードを参照してください。

EasyDialogs モジュールでは以下の関数を定義しています。

Message (*str*[, *id*[, *ok*]])
メッセージテキスト *str* 付きのモーダルダイアログを表示します。テキストの長さは最大 255 文字です。ボタンのテキストはデフォルトでは “OK” ですが、文字列の引数 *ok* を指定して変更できます。ユーザが “OK” ボタンをクリックすると処理を戻します。

AskString (*prompt*[, *default*[, *id*[, *ok*[, *cancel*]]]])

ユーザに文字列値の入力を促すモーダルダイアログを表示します。*prompt* はプロンプトメッセージで、オプションの *default* 引数は入力文字列の初期値です（指定しなければ`""` を使います)。“OK” と “Cancel” ボタンの文字列は *ok* と *cancel* の引数で変更できます。文字列の長さは全て最大 255 文字です。入力された文字列か、ユーザがキャンセルした場合には `None` を返します。

`AskPassword`(*prompt*[, *default*[, *id*[, *ok*[, *cancel*]]]])

ユーザに文字列値の入力を促すモーダルダイアログを表示します。`AskString()` に似ていますが、ユーザの入力したテキストは点で表示されます。引数は `AskString()` のものと同じ意味です。

`AskYesNoCancel`(*question*[, *default*[, *yes*[, *no*[, *cancel*[, *id*]]]]])

プロンプト *question* と “Yes”、“No”、“Cancel” というラベルの 3 つボタンが付いたダイアログを表示します。ユーザが “Yes” を押した場合には `1` を、“No” ならば `0` を、“Cancel” ならば`-1` を返します。`RETURN` キーを押した場合は *default* の値（*default* を指定しない場合は `0`) を返します。ボタンのテキストはそれぞれ引数 *yes*、*no*、*cancel* で変更できます。ボタンを表示したくなければ引数に`""`を指定します。

`ProgressBar`([*title*[, *maxval*[, *label*[, *id*]]]])

プログレスバー付きのモードレスダイアログを表示します。これは後で述べる `ProgressBar` クラスのコンストラクタです。*title* はダイアログに表示するテキスト文字列 (デフォルトの値は “Working...”) で、 *maxval* は処理が完了するときの値です (デフォルトは `0` で、残りの作業量が不確定であることを示します) 。*label* はプログレスバー自体の上に表示するテキストです。

`GetArgv`([*optionlist*[*commandlist*[, *addoldfile*[, *addnewfile*[, *addfolder*[, *id*]]]]]])

コマンドライン引数リストの作成を補助するためのダイアログを表示します。得られた引数リストを `sys.argv` の形式にします。これは `getopt.getopt()` の引数として渡すのに適した形式です。*addoldfile*、*addnewfile*、*addfolder* はブール型の引数です。これらの引数が真の場合、それぞれ実在のファイル、まだ (おそらく) 存在しないファイル、フォルダへのパスをコマンドラインのパスとして設定できます。(注意: `getopt.getopt()` がファイルやフォルダ引数を認識できるようにするためには、オプションの引数がそれらより前に現れるようにしなければなりません。) 空白を含む引数は、空白をシングルクォートあるいはダブルクォートで囲んで指定できます。

ユーザが “Cancel” ボタンを押した場合、`SystemExit` 例外を送出します。

optionlist には、ポップアップメニューで選べる選択肢を定義したリストを指定します。ポップアップメニューの要素には、次の 2 つの形式、*optstr* または（*optstr*， *descr*）があります。*descr* に短い説明文字列を指定すると、該当の選択肢をポップアップメニューで選択しいる間その文字列をダイアログに表示します。*optstr* とコマンドライン引数の対応を以下に示します:

***optstr* format**	**Command-line format**
`x`	**-x** (短いオプション)
`x:` あるいは `x=`	**-x** (値を持つ短いオプション)
`xyz`	**--xyz** (長いオプション)
`xyz:` あるいは `xyz=`	**--xyz** (値を持つ長いオプション)

commandlist は *cmdstr* あるいは（*cmdstr*， *descr*）の形のアイテムからなるリストです。*descr* は上と同じです。*cmdstr* はポップアップメニューに表示されます。メニューを選択すると *cmdstr* はコマンドラインに追加されますが、それに続く ‘:’ や ‘=’ は (存在していれば) 取り除かれます。

2.0 で追加された仕様です。

`AskFileForOpen`([*message*] [, *typeList*] [, *defaultLocation*] [, *defaultOptionFlags*] [, *location*] [, *clientName*] [, *windowTitle*] [, *actionButtonLabel*] [, *cancelButtonLabel*] [, *preferenceKey*] [, *popupExtension*] [, *eventProc*] [, *previewProc*] [, *filterProc*] [, *wanted*])

どのファイルを開くかをユーザに尋ねるダイアログを表示し、ユーザが選択したファイルを返します。ユーザがダイアログをキャンセルした場合には `None` を返します。*message* はダイアログに表

示するテキストメッセージです。*typeList* は選択できるファイルタイプを表す 4 文字の文字列からなるリスト、*defaultLocation* は最初に表示するルフォルダで、パス名、FSSpec あるいは FSRef で指定します。*location* はダイアログを表示するスクリーン上の位置 (x, y) です。*actionButtonLabel* は OK ボタンの位置に “Open” の代わりに表示する文字列、*cancelButtonLabel* は “Cancel” ボタンの位置に “Cancel” の代わりに表示する文字列です。*wanted* は返したい値のタイプで、str、unicode、AFSSpec、FSRef およびそれらのサブタイプを指定できます。

その他の引数の説明については Apple Navigation Services のドキュメントと EasyDialogs のソースコードを参照してください。

AskFileForSave([*message*] [, *savedFileName*] [, *defaultLocation*] [, *defaultOptionFlags*] [, *location*] [, *clientName*] [, *windowTitle*] [, *actionButtonLabel*] [, *cancelButtonLabel*] [, *preferenceKey*] [, *popupExtension*] [, *fileType*] [, *fileCreator*] [, *eventProc*] [, *wanted*])

保存先のファイルをユーザに尋ねるダイアログを表示して、ユーザが選択したファイルを返します。ユーザがダイアログをキャンセルした場合には None を返します。*savedFileName* は保存先のファイル名 (戻り値) のデフォルト値です。その他の引数の説明については AskFileForOpen() を参照してください。

AskFolder([*message*] [, *defaultLocation*] [, *defaultOptionFlags*] [, *location*] [, *clientName*] [, *windowTitle*] [, *actionButtonLabel*] [, *cancelButtonLabel*] [, *preferenceKey*] [, *popupExtension*] [, *eventProc*] [, *filterProc*] [, *wanted*])

フォルダの選択をユーザに促すダイアログを表示して、ユーザが選択したフォルダを返します。ユーザがダイアログをキャンセルした場合には None を返します。引数についての説明は AskFileForOpen() を参照してください。

参考資料:

Navigation Services Reference
(http://developer.apple.com/documentation/Carbon/Reference/Navigation_Services_Ref/)

Programmer’s reference documentation の Carbon framework の Navigation Services の項。

2.8.1 プログレスバーオブジェクト

ProgressBar オブジェクトでは、モードレスなプログレスバーダイアログのサポートを提供しています。定量プログレスバー (温度計スタイル) と不定量プログレスバー (床屋の螺旋看板スタイル) がサポートされています。プログレスバーの最大値がゼロ以上の場合には定量インジケータに、そうでない場合は不定量インジケータになります。 2.2 で変更された仕様: 不定量プログレスバーのサポートを追加しました。

ダイアログは作られるとすぐに表示されます。ダイアログの “Cancel” ボタンを押すか、Cmd-. (コマンドキーを押しながらピリオド (‘.’) を押す) か、あるいは ESC をタイプすると、ダイアログウィンドウを非表示にして KeyboardInterrupt を送出します (ただし、この応答は次にプログレスバーを更新するときまで、すなわち次に inc() または set() を呼び出してダイアログを更新するまで発生しません) 。それ以外の場合、プログレスバーは ProgressBar オブジェクトを廃棄するまで表示されたままになります。

ProgressBar オブジェクトには以下の属性とメソッドがあります。

curval

プログレスバーの現在の値 (整数型あるいは長整数型) です。プログレスバーの通常のアクセスのメソッドによって curval を 0 と maxval の間にします。この属性を直接変更してはなりません。

maxval

プログレスバーの最大値 (整数型あるいは長整数型) です; プログレスバー (温度計, thermometer) では、curval が maxval に等しい時に全量に到達します。maxval が 0 の場合、不定量プログレスバー (床屋の螺旋看板, barbar pole) になります。この属性を直接変更してはなりません。

title(*[newstr]*)
プログレスダイアログのタイトルバーのテキストを *newstr* に設定します。

label(*[newstr]*)
プログレスダイアログ中のプログレスボックスのテキストを *newstr* に設定します。

set(*value[, max]*)
プログレスバーの現在値 curval を *value* に設定します。*max* も指定した場合、maxval を *max* にします。*value* は前もって 0 と maxval の間になるよう強制的に設定されます。温度計バーの場合、変更内容を反映するよう表示を更新します。変更によって定量プログレスバーから不定量プログレスバーへ、あるいはその逆への推移が起こります。

inc(*[n]*)
プログレスバーの curval を *n* だけ増やします。*n* を指定しなければ 1 だけ増やします。(*n* は負にもでき、その場合は curval を減少させます。) 変更内容を反映するようプログレスバーの表示を更新します。プログレスバーが不定量プログレスバーの場合、床屋の螺旋看板 (barbar pole) 模様を 1 度「回転」させます。増減によって curval が 0 から maxval までの範囲を越えた場合、0 と maxval の範囲に収まるよう強制的に値を設定します。

2.9 FrameWork — 対話型アプリケーション・フレームワーク

FrameWork モジュールは、対話型 Macintosh アプリケーションのクラスで、同時にフレームワークを提供します。プログラマは、サブクラスを作って基底クラスの様々なメソッドをオーバーライドし、必要な機能を実装することでアプリケーションを組み立てられます。機能のオーバーライドは、時によって様々な異なるレベルで行われます。つまり、ある一つのダイアログウィンドウでクリックの処理を普段と違う方法で行うには、完全なイベント処理をオーバーライドする必要はありません。

FrameWork はまだ作成中のところがずいぶん残っています。このドキュメントでは最も重要な機能だけしか記述していませんし、それさえも論理的な形で書かれてもいません。ソースか例題を詳しく見てください。次にあげるのは、MacPython ニュースグループにポストされたコメントで、FrameWork の強力さと限界について述べています。

> FrameWork の最大の強みは、制御の流れをたくさんの異なる部分に分割できることです。例えば W を使って、いろいろな方法でメニューをオン/オフしたり、残りをいじらずにうまくプラグインさせることができます。FrameWork の弱点は、コマンドインタフェースが抽象化されていないこと (といっても難しいわけではないですが)、ダイアログサポートが最低限しかないこと、それからコントロール/ツールバーサポートが全くないことです。

FrameWork モジュールは次の関数を定義しています。

Application()
アプリケーション全体を表現しているオブジェクト。メソッドについての詳細は以下の記述を参照してください。デフォルト __init__() ルーチンは、空のウィンドウ辞書とアップルメニューつきのメニューバーをを作成します。

MenuBar()
メニューバーを表現するオブジェクト。このオブジェクトは普通はユーザは作成しません。

Menu(*bar, title[, after]*)
メニューを表現するオブジェクト。生成時には、メニューが現われる MenuBar と、*title* 文字列、メニューが表示されるべき (1 から始まる) 位置 *after* (デフォルトは末尾) を渡します。

MenuItem(*menu, title[, shortcut, callback]*)
メニューアイテムオブジェクトを作成します。引数は作成するメニューと、アイテムのタイトル文字

列、オプションのキーボードショートカット、コールバックルーチンです。コールバックは、メニューID、メニュー内のアイテム番号 (1 から数える)、現在のフロントウィンドウ、イベントレコードを引数に呼ばれます。

呼び出し可能なオブジェクトのかわりに、コールバックは文字列でも良いです。この場合、メニューの選択は、最前面のウィンドウとアプリケーションの中でメソッド探索を引き起こします。メソッド名は、コールバック文字列の前に'domenu_' を付けたものです。

MenuBar の fixmenudimstate() メソッドを呼びだすと、現在のフロントウィンドウにもとづいて、適切なディム化を全てのメニューアイテムに対してほどこします。

Separator(*menu*)
メニューの最後にセパレータを追加します。

SubMenu(*menu, label*)
label の名前のサブメニューを、メニュー *menu* の下に作成します。メニューオブジェクトが返されます。

Window(*parent*)
(モードレス) ウィンドウを作成します。*Parent* は、ウィンドウが属するアプリケーションオブジェクトです。作成されたウィンドウはまだ表示されません。

DialogWindow(*parent*)
モードレスダイアログウィンドウを作成します。

windowbounds(*width, height*)
与えた幅と高さのウィンドウを作成するのに必要な、(*left*, *top*, *right*, *bottom*) からなるタプルを返します。ウィンドウは以前のウィンドウに対して位置をずらして作成され、全体のウィンドウが画面からなるべく外れないようにします。しかし、ウィンドウはいつでも全く同じサイズで、そのため一部は画面から隠れる場合もあります。

setwatchcursor()
マウスカーソルを時計型に設定します。

setarrowcursor()
マウスカーソルを矢印型に設定します。

2.9.1 アプリケーションオブジェクト

アプリケーションオブジェクトのメソッドは各種ありますが、次のメソッドをあげておきます。

makeusermenus()
アプリケーションでメニューを使う必要がある場合、このメソッドをオーバーライドします。属性 menubar にメニューを追加します。

getabouttext()
このメソッドをオーバーライドすることで、アプリケーションの説明を記述するテキスト文字列を返します。代わりに、do_about() メソッドをオーバーライドすれば、もっと凝った“アバウト”メッセージを出す事ができます。

mainloop([*mask*[, *wait*]])
このルーチンがメインイベントループで、作成したアプリケーションが動き出すためにはこれを呼ぶことになります。*Mask* は操作したいイベントを選択するマスクです。 *wait* は並行に動作しているアプリケーションに割り当てたいチック数 (1/60 秒) です (デフォルトで 0 ですが、あまり良い値ではありません)。*self* フラグを立ててメインループを抜ける方法はまだサポートされていますが、これはお勧めできません。代わりに self._quit() を呼んでください。

イベントループは小さなパーツに分割されていて、各々をオーバーライドできるようになっています。これらのメソッドは、デフォルトでウィンドウとダイアログや、ドラッグとリサイズの操作、AppleEvent、非 FrameWork のウィンドウに関するウィンドウの操作などに関するイベントを分岐することなどまで面倒をみてくれます。

原則として、全てのイベントハンドラは、イベントが完全に取り扱われた場合は `1` を返さなくてはいけませんし、それ以外では `0` を返さなくてはいけません (例えば、前面のウィンドウは FrameWork ウィンドウではない場合を考えてください)。こうしなくてはいけない理由は、アップデートイベントなどが Sioux コンソールウィンドウなどの他のウィンドウにきちんと渡されるようにするためです。*our_dispatch* やその呼び出し元の内部から `MacOS.HandleEvent()` を呼んではいけません。そうしたコードが Python の内部ループのイベントハンドラを経由して呼ばれると、無限ループになりかねないからです。

asyncevents(*onoff*)

非同期でイベント操作をしたい場合は、非ゼロの引数でこのメソッドを呼んでください。こうすることで、イベントが生じた時に、内部のインタプリタのループで、アプリケーションイベントハンドラ *async_dispatch* が呼ばれることになります。すると、長時間の計算を行っている場合でも、FrameWork ウィンドウがアップデートされ、ユーザーインターフェースが動き続けるようになります。ただし、インタプリタの動作が減速し、非リエントラントのコード (例えば FrameWork 自身など) に奇妙な動作が見られるかもしれません。デフォルトでは *async_dispatch* はすぐに *our_dispatch* を呼びますが、このメソッドをオーバーライドすると、特定のイベントを非同期で操作しても良くなります。処理しないイベントは Sioux などに渡されることになります。

on あるいは off 値が返されます。

_quit()

実行中の `mainloop()` 呼び出しを、次の適当なタイミングで終了させます。

do_char(*c, event*)

ユーザーが文字 *c* をタイプした時に呼ばれます。イベントの全詳細は event 構造体の中にあります。このメソッドはウィンドウオブジェクト内で使うためにも提供されています。このオブジェクトのウィンドウが最前面にある場合は、アプリケーション全般について本ハンドラをオーバーライドします。

do_dialogevent(*event*)

イベントループ内部で最初に呼ばれて、モードレスダイアログイベントを処理します。デフォルトではメソッドは単にイベントを適切なダイアログに分岐するだけです (関連したダイアログウィンドウオブジェクトを経由してではありません)。特別にダイアログイベント (キーボードショートカットなど) を処理する必要がある場合にオーバーライドしてください。

idle(*event*)

イベントが無い場合にメインイベントループから呼ばれます。 null イベントも渡されます (つまりマウス位置などを監視することができます)。

2.9.2 ウィンドウオブジェクト

ウィンドウオブジェクトは特に次のメソッドを持ちます。

open()

ウィンドウを開く時はこのメソッドをオーバーライドします。MacOS ウィンドウ ID を `self.wid` に入れて `do_postopen()` メソッドを呼ぶと、親アプリケーションにウィンドウを登録します。

close()

ウィンドウを閉じるときに特別な処理をする場合はこのメソッドをオーバーライドします。親アプリケーションからウィンドウの登録を削除するには、`do_postclose()` を呼びます。

do_postresize (*width, height, macoswindowid*)
ウィンドウがリサイズされた後に呼ばれます。InvalRect を呼び出す以外にもすることがある場合はこれをオーバーライドします。

do_contentclick (*local, modifiers, event*)
ウィンドウのコンテント部分をユーザーがクリックすると呼ばれます。引数は位置座標 (ウィンドウを基準)、キーモディファイア、生のイベントです。

do_update (*macoswindowid, event*)
ウィンドウのアップデートイベントが受信された時に呼ばれます。ウィンドウを再描画します。

do_activate (*activate, event*)
ウィンドウがアクティブ化 (*activate* == 1)、非アクティブ化 (*activate* == 0) する際に呼ばれます。フォーカスのハイライトなどを処理します。

2.9.3 コントロールウィンドウオブジェクト

コントロールウィンドウオブジェクトには Window オブジェクトのメソッドの他に次のメソッドがあります。

do_controlhit (*window, control, pcode, event*)
コントロール *control* のパートコード *pcode* がユーザーにヒットされた場合に呼ばれます。トラッキングなどは任せておいてかまいません。

2.9.4 スクロールウィンドウオブジェクト

スクロールウィンドウオブジェクトは、次のメソッドを追加したコントロールウィンドウオブジェクトです。

scrollbars ([*wantx*[, *wanty*]])
水平スクロールバーと垂直スクロールバーを作成します (あるいは破棄します)。引数はどちらが欲しいか指定します (デフォルトは両方)。スクロールバーは常に最小値 0 、最大値 32767 です。

getscrollbarvalues ()
このメソッドは必ず作っておかなくてはいけません。現在のスクロールバーの位置を与えるタプル (*x*, *y*) を (0 の 32767 間で) 返してください。バーの方向について全文書が可視状態であること知らせるため None を返す事もできます。

updatescrollbars ()
文書に変更があった場合はこのメソッドを呼びます。このメソッドは getscrollbarvalues() を呼んでスクロールバーを更新します。

scrollbar_callback (*which, what, value*)
あらかじめ与えておくメソッドで、ユーザーとの対話により呼ばれます。*which* は 'x' か 'y' 、*what* は '-','--','set','++','+' のどれかです。'set' の場合は、*value* に新しいスクロールバー位置を入れておきます。

scalebarvalues (*absmin, absmax, curmin, curmax*)
getscrollbarvalues() の結果から値を計算するのを助ける補助的なメソッドです。文書の最小値と最大値、可視部分に関する最先頭値 (最左値) と最底値 (最右値) を渡すと、正しい数か None を返します。

do_activate (*onoff, event*)
ウィンドウが最前面になった時、スクロールバーのディム (dimming)/ハイライトの面倒をみます。このメソッドをオーバーライドするなら、オーバーライドしたメソッドの最後でオリジナルのメソッドを呼んでください。

do_postresize(*width, height, window*)

スクロールバーを正しい位置に移動させます。オーバーライドする時は、オーバーライドしたメソッドの一番最初でオリジナルのメソッドを呼んでください。

do_controlhit(*window, control, pcode, event*)

スクロールバーのインタラクションを処理します。これをオーバーライドする時は、オリジナルのメソッドを最初に呼び出してください。非ゼロの返り値はスクロールバー内がヒットされたことを意味し、実際に処理が進むことになります。

2.9.5 ダイアログウィンドウオブジェクト

ダイアログウィンドウオブジェクトには、`Window` オブジェクトのメソッドの他に次のメソッドがあります。

open(*resid*)

ID *resid* の DLOG リソースからダイアログウィンドウを作成します。ダイアログオブジェクトは `self.wid` に保存されます。

do_itemhit(*item, event*)

アイテム番号 *item* がヒットされた時に呼ばれます。トグルボタンなどの再描画は自分で処理してください。

2.10 autoGIL — イベントループ中のグローバルインタープリタの取り扱い

`autoGIL` モジュールは、自動的にイベントループを実行する場合 、Python のグローバルインタープリタをロックしたり、ロックの解除をしたりするための関数 `installAutoGIL` を提供します。

exception AutoGILError

例えば現在のスレッドがループしていないなど、オブザーバにコールバックができない場合に発生します。

installAutoGIL()

現在のスレッドのイベントループ (CFRunLoop) 中のオブザーバにコールバックを行ない、適切な時にグローバルインタープリタロック (GIL) を、イベントループが使用されていない間、他の Python スレッドの起動ができるようにロックしたり、ロックの解除をしたりします。

有効性：OSX 10.1 以降

三章

MacPython OSA モジュール

Python は オープンスクリプティングアーキテクチャ(Open Scripting Architecure、OSA、一般的には AppleScript と呼ばれる) のかなり完全な実装を行っていて、Python プログラムからスクリプト可能なアプリケーションを操作したり、Python へのインターフェースを備えたものにすることができます。

AppleScript と OSA の様々なコンポーネントの記述のために、また、アーキテクチャおよび用語についての理解を得るために、アップルの文書を読む必要があります。"Applescript Language Guide" は概念のモデルおよび用語、Standard Suite について説明した文書です。"Open Scripting Architecture" 文書は、アプリケーションプログラマの視点から OSA を使用する方法について説明しています。これらの文書は Apple ヘルプビューワの Developer Documentation 中の Core Technologies セクションにあります。

アプリケーションをスクリプトで操作する例として、次の AppleScript は、一番前の **Finder** ウィンドウの名前を取得し、それを印字します。

```
tell application "Finder"
    get name of window 1
end tell
```

Python では以下のコードで同じ事ができます。

```
import Finder

f = Finder.Finder()
print f.get(f.window(1).name)
```

配布されている Python ライブラリは、Standard Suite を実装したパッケージに加えて、いくつかの一般的なアプリケーションへのインターフェースを実装したパッケージが含まれています。

アプリケーションに AppleEvent を送るためには、アプリケーションの用語 (**Script Editor** が「辞書」と呼ぶもの) に接続する Python パッケージを最初に作成しなければなりません。これは、**PythonIDE** の内部から、あるいは、コマンドラインからのスタンドアロンのプログラムとして 'gensuitemodule.py' モジュールを実行することにより行うことができます。

'gensuitemodule.py' モジュールで生成される出力は多くのモジュールを備えたパッケージのため、全ての Suite をプログラムの中で 1 つにまとめて利用できるようにするために `__init__` モジュールが追加されています。Python 継承グラフは AppleScript 継承グラフを理解するので、Standard Suite をサポートしていて、余分な引数を備えた 1 つあるいは 2 つの変数を拡張する事ができるようにプログラム辞書が書かれていた場合、出力された Suite は、`StdSuites.Standard_Suite` からすべてをインポートして再エクスポートし、さらに拡張機能をもったメソッドをオーバーライドするモジュール `Standard_Suite` を含みます。`gensuitemodule` の出力は人間に判読可能で、Python docstrings 中にはオリジナルの AppleScript 辞書にあった文書を含んでいます。したがって、それを読むことは有用な情報源となります。

出力されたパッケージは、メソッドとして AppleScript 変数をすべて含み、第 1 の引数としての直接オブ

ジェクトを含み、キーワード引数としてのすべてのオプションの引数を含む、パッケージと同じ名前を備えた主要なクラスを実装しています。また AppleScript クラスは Python クラス、そして類事物その他のもろもろの物として実装されています。

変数を実装する主要な Python クラスは、さらに AppleScript クラス"application" で宣言されたプロパティおよび要素へのアクセスを許可します。現在のリリースでオブジェクト指向的にやろうとするならば、例えば、より Python 的な `f.window(1).name.get()` の代りに `f.get(f.window(1).name)` を利用する必要があります。

AppleScript 識別子が Python 識別子と同じでない場合、名前は少数の規則によって判別します。

- スペースは下線に置換されます。
- `_xx_` が 16 進法の文字値である場合、他の英数字でない文字は `xx` と置換されます。
- あらゆる Python 予約語には下線を追加します。

Python は、さらに Python でスクリプト対応アプリケーションを作成する事をサポートしています。次のモジュールは MacPython の AppleScript サポートに適切です。

gensuitemodule	OSA 辞書からスタブパッケージを作成します。
aetools	Apple Event を送るための基本的なサポート
aepack	Python 変数と AppleEvent データコンテナ間の変換
aetypes	Apple Event オブジェクトモデルの Python 表現
MiniAEFrame	オープンスクリプティングアーキテクチャ(OSA) サーバ (“Apple Events”) のサポート。

さらに、`Finder`, `Terminal`, `Explorer`, `Netscape`, `CodeWarrior`, `SystemEvents` そして `StdSuites` のサポートモジュールは、あらかじめ生成されています。

3.1 `gensuitemodule` — OSA スタブ作成パッケージ

`gensuitemodule` モジュールは AppleScript 辞書によって特定のアプリケーションに実装されている AppleScript 群のためのスタブコードを実装した Python パッケージを作成します。

このモジュールは、通常は **PythonIDE** からユーザによって起動されますが、コマンドラインからスクリプトとして実行する (オプションとしてヘルプに **--help** を与えてみてください) こともできますし、Python コードでインポートして利用する事もできます。使用例として、どのようにして標準ライブラリに含まれているスタブパッケージを生成するか、‘Mac/scripts/genallsuites.py’ にあるソースを見てください。

このモジュールは次の関数を定義しています。

is_scriptable(*application*)

`application` としてパス名を与えたアプリケーションがスクリプト可能でありそうな場合、真を返します。返り値はやや不確実な場合があります。**Internet Explorer** はスクリプト不可能なように見えてしまいますが、実際はスクリプト可能です。

processfile(*application*[, *output, basepkgname, edit_modnames, creatorsignature, dump, verbose*])

パス名として渡された `application` のためのスタブパッケージを作成します。‘.app’ として一つのパッケージにまとめてあるプログラム群のために内部の実行プログラムそのものではなくパッケージへのパス名を渡すだけでよくなっています。パッケージ化されていない CFM アプリケーションではアプリケーションバイナリのファイル名を渡す事もできます。

この関数は、アプリケーションの OSA 用語リソースを捜し、これらのリソースを読み取り、その結果データをクライアントスタブを実装した Python コードパッケージを作成するために使用します。

output は作成結果のパッケージを保存するパス名で、指定しない場合は標準の「別名で保存 (save file as)」ダイアログが表示されます。basepkgname はこのパッケージの基盤となるパッケージを指定します。デフォルトは StdSuites になります。StdSuites 自体を生成する場合だけ、このオプションを指定する必要があります。edit_modnames は自動生成によって作成されてあまり綺麗ではないモジュール名を変更するために使用することができる辞書です。creator_signature はパッケージ中の 'PkgInfo' ファイル、あるいは CFM ファイルクリエータ署名から通常得られる 4 文字クリエータコードを無視するために使用することができます。dump にはファイルオブジェクトを与えます、これを指定するとリソースを読取った後に停止して processfile がコード化した用語リソースの Python 表現をダンプします。verbose にもまたファイルオブジェクトを与え、これを指定すると processfile の行なっている処理の詳細を出力します。

processfile_fromresource (*application*[, *output, basepkgname, edit_modnames, creatorsignature, dump, verbose*])

この関数は、用語リソースを得るのに異なる方法を使用する以外は、processfile と同じです。この関数では、リソースファイルとして application を開き、このファイルから "aete" および "aeut" リソースをすべて読み込む事で、AppleScript 用語リソース読み込みを行ないます。

3.2 aetools — OSA クライアントのサポート

aetools モジュールは Python で AppleScript クライアントとしての機能をサポートするアプリケーションを構築するための基本的な機能を含んでいます。さらに、このモジュールは、 aetypes および aepack モジュールの中核機能をインポートし再 エクスポートします。gensuitemodule によって生成されたスタブパッケージは aetools のかなり適切な部分をインポートするので、通常はそれを明示的にインポートする必要はありません。生成されたパッケージ群を使用することができない場合と、スクリプト対応のためにより低いレベルのアクセスを必要としている場合、例外が発生します。

aetools モジュールはそれ自身、Carbon.AE モジュールによって提供される AppleEvent サポートを利用します。このモジュールにはウィンドウマネージャへのアクセスを必要とするという 1 つの欠点があります。詳細は第 1.1.2 章を見てください。この制限は将来のリリースで撤廃されるかもしれません。

aetools モジュールは下記の関数を定義しています。

packevent (*ae, parameters, attributes*)

あらかじめ作成された Carbon.AE.AEDesc オブジェクト中のパラメーターおよび属性を保存します。parameters と attributes は Python オブジェクトの 4 文字の OSA パラメータのキーを写像した辞書です。このオブジェクトをパックするには aepack.pack() を使います。

unpackevent (*ae*[, *formodulename*])

再帰的に、Carbon.AE.AEDesc イベントを Python オブジェクトへアンパックします。関数は引数の辞書および属性の辞書を返します。formodulename 引数は AppleScript クラスをどこに捜しに行くか制御するために、生成されたスタブパッケージにより使用されます。

keysubst (*arguments, keydict*)

Python キーワード引数辞書 arguments を、写像による 4 文字の OSA キーとして keydict の中で指定された Python 識別子であるキーの交換により packevent によって要求されるフォーマットへ変換します。生成されたパッケージ群によって使用されます。

enumsubst (*arguments, key, edict*)

arguments 辞書が key へのエントリーを含んでいる場合、辞書 edict のエントリーに見合う値に変換します。これは人間に判読可能なように Python 列挙名を OSA 4 文字のコードに変換します。生成されたパッケージ群によって使用されます。

aetools モジュールは次のクラスを定義しています。

class TalkTo (*[signature=None, start=0, timeout=0]*)

アプリケーションとの対話に利用する代理の基底クラスです。signature はクラス属性 _signature (サブクラスによって通常設定される) を上書きした、対話するアプリケーションを定義する 4 文字クリエートコードです。start にはクラスインスタンス上でアプリケーションを実行することを可能にするために、真を設定する事ができます。timeout を明示的に設定する事で、 AppleEvent の返答を待つデフォルトのタイムアウト時間を変更する事ができます。

_start ()

アプリケーションが起動していてるか確認し、起動していなければ起動しようとします。

send (*code, subcode*[*, parameters, attributes*])

OSA 指示子 code, subcode (いずれも通常 4 文字の文字列です) を持った変数のために、parameters をパックし、attributes に戻し、目標アプリケーションにそれを送って、返答を待ち、unpackevent を含んだ返答をアンパックし、AppleEvent の返答を返し、辞書としてアンパックした値と属性を返して、AppleEvent Carbon.AE.AEDesc を作成します。

3.3 aepack — Python 変数と AppleEvent データコンテナ間の変換

aepack モジュールは、Python 変数から AppleEvent ディスクリプタへの変換 (パック) と、その逆に変換 (アンパック) する関数を定義しています。Python 内では AppleEvent ディスクリプタは、組み込み型である AEDesc の Python オブジェクトとして扱われます。AEDesc は Carbon.AE モジュールで定義されています。

aepack モジュールは次の関数を定義しています。

pack (*x*[*, forcetype*])

Python 値 x を変換した値を保持する AEDesc オブジェクトを返します。*forcetype* が与えることで、結果のディスクリプタ型を指定できます。それ以外では、Python 型から Apple Event ディスクリプタ型へのデフォルトのマッピングが使われます。マッピングは次の通りとなります。

Python type	descriptor type
FSSpec	typeFSS
FSRef	typeFSRef
Alias	typeAlias
integer	typeLong (32 bit integer)
float	typeFloat (64 bit floating point)
string	typeText
unicode	typeUnicodeText
list	typeAEList
dictionary	typeAERecord
instance	*see below*

x が Python インスタンスなら、この関数は __aepack__() メソッドを呼びだそうとします。このメソッドは AEDesc オブジェクトを返します。

x の変換が上で定義されていない場合は、この関数は、テキストディスクリプタとしてエンコードされた、値の (repr() 関数による)Python 文字列表現が返されます。

unpack (*x*[*, formodulename*])

x は AEDesc タイプのオブジェクトでなければいけません。この関数は、Apple Event ディスクリプタ *x* のデータの Python オブジェクト表現を返します。単純な AppleEvent データ型 (整数、テキスト、浮動少数点数) の、対応する Python 型が返されます。Apple Event リストは Python リストとして返され、リストの要素は再帰的にアンパックされます。formodulename の指定がない場合、オブジェ

クト参照 (例：line 3 of document 1) が、aetypes.ObjectSpecifier のインスタンスとして返されます。ディスクリプタ型が typeFSS である AppleEvent ディスクリプタが、FSSpec オブジェクトとして返されます。AppleEvent レコードディスクリプタが、再帰的にアンパックされた、型の 4 文字キーと要素を持つ Python 辞書として返されます。

オプションの formodulename 引数は gensuitemodule より作成されるスタブパッケージにより利用され、オブジェクト指定子のための OSA クラスをモジュールの中で見つけられることを保証します。これは、例えば、ファインダがウィンドウに対してオブジェクト指定子を返す場合、Finder.Window のインスタンスが得られ、aetypes.Window が得られないことを保証します。前者は、ファインダ上のウィンドウが持っている、すべての特性および要素のことを知っています。一方、後者のものはそれらのことを知りません。

参考資料:

Carbon.AE モジュール (4.1 節):
Apple Event マネージャルーチンへの組み込みアクセス

aetypes モジュール (3.4 節):
Apple Event ディスクリプタ型としてコードされた Python 定義

Inside Macintosh: Interapplication Communication
(http://developer.apple.com/techpubs/mac/IAC/IAC-2.html)
Macintosh 上でのプロセス間通信に関する情報

3.4 aetypes — AppleEvent オブジェクト

aetypes では、Apple Event データデスクリプタ (data descriptor) や Apple Event オブジェクト指定子 (object specifier) を表現するクラスを定義しています。

Apple Event データはデスクリプタに含まれていて、これらのデスクリプタは片付けされています。多くのデスクリプタは、単に対応する Python の型で表現されています。例えば、OSA 中の typeText は Python 文字列型で、typeFloat は 浮動小数点型になる、といった具合です。このモジュールでは、OSA の型のうち、直接的に対応する Python の型がないもののためにクラスを定義しています。そのようなクラスのインスタンスに対するパックやアンパック操作は、 aepack モジュール自動的に処理します。

オブジェクト指定子は、本質的には Apple Event サーバ中に実装されているオブジェクトへのアドレスです。Apple Event 指定子は、Apple Event のオブジェクトそのものとして、あるいはオプションパラメタの引数として使われます。aetypes モジュールには OSA クラスやプロパティを表現するための基底クラスが入っています。これらのクラスは、gensuitemodule が生成するパッケージ内で、目的に応じてクラスやプロパティを増やす際に使われます。

以前のバージョンとの互換性や、スタブパッケージを生成していないようなアプリケーションをスクリプトで書く必要がある場合のために、このモジュールには Document 、Window、Character、といったよく使われる OSA クラスのいくつかを指定できるオブジェクト指定子も入っています。

AEObjects モジュールでは、以下のようなクラスを定義して、Apple Event デスクリプタデータを表現しています:

class Unknown (*type, data*)
aepack や aetypes がサポートしていない OSA のデスクリプタデータ、すなわち、このモジュールで扱っている他のクラスや、Python の組み込み型の値で表現されていないようなデータを表現するクラスです。

class Enum (*enum*)
列挙値 (enumeration value) を表すクラスです。値は 4 文字の文字列型になります。

class `InsertionLoc`(*of, pos*)
オブジェクト `of` の中の `pos` の位置を表すクラスです。

class `Boolean`(*bool*)
ブール値 (真偽値) を表すクラスです。

class `StyledText`(*style, text*)
スタイル情報 (フォント、タイプフェイスなど) つきのテキストを表すクラスです。

class `AEText`(*script, style, text*)
スクリプトシステム (script system) およびスタイル情報の入ったテキストを表すクラスです。

class `IntlText`(*script, language, text*)
スクリプトシステムと言語情報 (language information) の入ったテキストを表すクラスです。

class `IntlWritingCode`(*script, language*)
スクリプトシステムと言語情報を表すクラスです。

class `QDPoint`(*v, h*)
QuickDraw の点を表すクラスです。

class `QDRectangle`(*v0, h0, v1, h1*)
QuickDraw の矩形を表すクラスです。

class `RGBColor`(*r, g, b*)
色を表すクラスです。

class `Type`(*type*)
OSA の型 (type value) を表すクラスです。4 文字からなる名前を値に持ちます。

class `Keyword`(*name*)
OSA のキーワードです。4 文字からなる名前を値に持ちます。

class `Range`(*start, stop*)
範囲を表すクラスです。

class `Ordinal`(*abso*)
先頭を表す `"firs"` や中央を表す`"midd"` のように、数値でない絶対位置指定子を表すクラスです。

class `Logical`(*logc, term*)
演算子 `logc` を `term` に適用したときの論理式を表すクラスです。

class `Comparison`(*obj1, relo, obj2*)
`obj1` と `obj2` の `relo` による比較を表すクラスです。

以下のクラスは、生成されたスタブパッケージが、AppleScript のクラスやプロパティを Python で表現する上で基底クラスとして利用します。

class `ComponentItem`(*which*[, *fr*])
OSA クラス用の抽象基底クラスです。サブクラスでは、クラス属性 `want` を 4 文字の OSA クラスコードに設定せねばなりません。このクラスのサブクラスのインスタンスは AppleScript オブジェクト指定子と同じになります。インスタンス化を行う最には、`which` にセレクタを渡さねばなりません。また、任意で親オブジェクトを `fr` に渡せます。

class `NProperty`(*fr*)
OSA プロパティ用の抽象基底クラスです。サブクラスでは、クラス属性 `want` と `which` を設定して、どのプロパティを表しているかを指定せねばなりません。このクラスのサブクラスのインスタンスはオブジェクト指定子と同じになります。

class `ObjectSpecifier`(*want, form, seld*[, *fr*])
`ComponentItem` と `NProperty` の基底クラスで、汎用の OSA オブジェクト指定子を表します。パ

ラメタの説明は Apple Open Scripting Architecture のドキュメントを参照してください。このクラスは抽象クラスではないので注意してください。

3.5 MiniAEFrame — オープンスクリプティングアーキテクチャサーバのサポート

MiniAEFrame モジュールは、アプリケーションにオープンスクリプティングアーキテクチャ(OSA) サーバ機能を持たせるためのフレームワークを提供します。つまり、AppleEvents の受信と処理を行わせます。FrameWork と連携させても良いし、単独でも使えます。実例として、このモジュールは **PythonCGISlave** の中で使われています。

MiniAEFrame には以下のクラスが定義されています。

class AEServer()

AppleEvent の分岐を処理するクラス。作成するアプリケーションはこのクラスと、MiniApplication あるいは FrameWork.Application のサブクラスでなければなりません。サブクラス化したクラスでは__init__() メソッドで、継承した両方のクラスの__init__() メソッドを呼びださなければなりません。

class MiniApplication()

FrameWork.Application とある程度互換なクラスですが、機能は少ないです。このクラスのイベントループはアップルメニュー、Cmd-.(コマンドキーを押しながらピリオド. を押す)、AppleEvent をサポートします。他のイベントは Python インタープリタか Sioux（CodeWarrior のコンソールシステム）に渡されます。作成するアプリケーションで AEServer を使いたいが、独自のウィンドウなどを持たない場合に便利です。

3.5.1 AEServer オブジェクト

installaehandler(*classe, type, callback*)

AppleEvent ハンドラをインストールします。*classe* と *type* は 4 文字の OSA クラスとタイプの指定子で、ワイルドカード'****' も使えます。対応する AppleEvent を受けるとパラメータがデコードされ、与えたコールバックが呼び出されます。

callback(*_object, **kwargs*)

与えたコールバックは、OSA ダイレクトオブジェクトを 1 番目のパラメータとして呼び出されます。他のパラメータは 4 文字の指定子を名前にしたキーワード引数として渡されます。他に 3 つのキーワード・パラメータが渡されます。つまり、_class と_type はクラスとタイプ指定子で、_attributes は AppleEvent 属性を持つ辞書です。

与えたメソッドの返り値は aetools.packevent() でパックされ、リプライとして送られます。

現在のクラス設計にはいくつか重大な問題があることに注意してください。引数に名前ではない 4 文字の指定子を持つ AppleEvent はまだ実装されていないし、イベントの送信側にエラーを返すこともできません。この問題は将来のリリースまで先送りにされています。

四章

MacOS ツールボックスモジュール

各種の MacOS ツールボックスへのインターフェースを与えるモジュール群があります。対応するモジュールがあるなら、そのモジュールではツールボックスで宣言された各種の構造体の Python オブジェクトが定義され、操作は定義されたオブジェクトのメソッドとして実装されています。その他の操作はモジュールの関数として実装されています。 C で可能な操作がすべて Python で可能なわけではありませんし (コールバックはよく問題になります)、パラメータが Python だと違ってしまうことはよくあります (特に入力バッファや出力バッファ)。全てのメソッドと関数は `__doc__` 文字列があるので、引数と返り値の説明を得る事ができます。他の情報源としては、*Inside Macintosh* などを参照してください。

これらのモジュールは全て `Carbon` パッケージに含まれています。この名前にもかかわらずそれら全てが Carbon フレームワークの一部なわけではありません。CF は、CoreFoundation フレームワークの中に実際はありますし、Qt は QuickTime フレームワークにあります。ツールボックスモジュールは普通以下のようにして利用します。

```
from Carbon import AE
```

注意！これらのモジュールはまだ文書化されていません。これらのモジュールのどれでもよいですが文書化に協力したいという方は、docs@python.org まで連絡をください。

Carbon.AE	Apple Event ツールボックスへのインタフェース
Carbon.AH	Apple ヘルプマネージャへのインタフェース
Carbon.App	アピアランスマネージャへのインタフェース
Carbon.CF	Core Foundation へのインタフェース
Carbon.CG	Component Manager へのインタフェース
Carbon.CaronEvt	Carbon Event Manager へのインタフェース
Carbon.Cm	Component Manager へのインタフェース
Carbon.Ctl	Control Manager へのインタフェース
Carbon.Dlg	Dialog Manager へのインタフェース
Carbon.Evt	Event Manager へのインタフェース
Carbon.Fm	Font Manager へのインタフェース
Carbon.Folder	Folder Manager へのインタフェース
Carbon.Help	Carbon Help Manager へのインタフェース
Carbon.List	List Manager へのインタフェース
Carbon.Menu	Menu Manager へのインタフェース
Carbon.Mlte	MultiLingual Text Editor へのインタフェース
Carbon.Qd	QuickDraw ツールボックスへのインタフェース
Carbon.Qdoffs	QuickDraw オフスクリーン API へのインタフェース
Carbon.Qt	QuickTime ツールボックスへのインタフェース
Carbon.Res	Resource Manager とハンドルへのインタフェース
Carbon.Scrap	Carbon Scrap Manager へのインタフェース
Carbon.Snd	Sound Manager へのインタフェース
Carbon.TE	TextEdit へのインタフェース
Carbon.Win	Window Manager へのインタフェース
ColorPicker	標準色選択ダイアログへのインターフェース

4.1 Carbon.AE — Apple Events

4.2 Carbon.AH — Apple ヘルプ

4.3 Carbon.App — アピアランスマネージャ

4.4 Carbon.CF — Core Foundation

CFBase, CFArray, CFData, CFDictionary, CFString と CFURL オブジェクトがいくらか部分的にサポートされています。

4.5 `Carbon.CG` — Core Graphics

4.6 `Carbon.CarbonEvt` — Carbon Event Manager

4.7 `Carbon.Cm` — Component Manager

4.8 `Carbon.Ctl` — Control Manager

4.9 `Carbon.Dlg` — Dialog Manager

4.10 `Carbon.Evt` — Event Manager

4.11 `Carbon.Fm` — Font Manager

4.12 `Carbon.Folder` — Folder Manager

4.13 `Carbon.Help` — Help Manager

4.14 `Carbon.List` — List Manager

4.15 `Carbon.Menu` — Menu Manager

4.16 `Carbon.Mlte` — MultiLingual Text Editor

4.17 `Carbon.Qd` — QuickDraw

4.18 `Carbon.Qdoffs` — QuickDraw Offscreen

4.19 `Carbon.Qt` — QuickTime

4.20 `Carbon.Res` — Resource Manager and Handles

4.21 `Carbon.Scrap` — Scrap Manager

4.22 `Carbon.Snd` — Sound Manager

4.23 `Carbon.TE` — TextEdit

4.24 `Carbon.Win` — Window Manager

4.25 ColorPicker — 色選択ダイアログ

ColorPicker モジュールは標準色選択ダイアログへのアクセスを提供します。

GetColor(*prompt, rgb*)

標準色選択ダイアログを表示し、ユーザが色を選択することを可能にします。*prompt* の文字列によりユーザに指示を与えられ、デフォルトの選択色を *rgb* で設定する事ができます。*rgb* は赤、緑、青の色要素のタプルで与えてください。GetColor() はユーザが選択した色のタプルと色が選択されたか、取り消されたかを示すフラグを返します。

五章

文書化されていないモジュール

この章のモジュールは、ほとんど (あるいはまったく) ドキュメント化されていません。これらのモジュールのいずれかについてドキュメントを寄与したいと考えているなら、docs@python.org までご連絡ください。

applesingle	AppleSingle フォーマットファイル用の基本的なデコーダ
buildtools	BuildApplet とその仲間のヘルパーモジュール
py_resource	コンパイル済みアプリケーションに 'PYC ' リソースを作成にするヘルパーモジュール
cfmfile	コードフラグメントリソースを扱うモジュール
icopen	open() と Internet Config の置き換え
macerrors	多くの MacOS エラーコード定数定義
macresource	スクリプトのリソースを見つける
Nav	Navigation Services へのインターフェース
mkcwproject	CodeWarrior プロジェクトの作成
nsremote	Netscape OSA モジュールのラッパー
PixMapWrapper	PixMap オブジェクトのラッパー
preferences	デフォルト設定へのサポートを持つアプリケーション初期設定管理プログラム
pythonprefs	Python インタプリタに特化した初期設定管理プログラム
quietconsole	バッファを用いての不可視の標準出力
videoreader	フレームの継続処理のための QuickTime ムービーのフレーム読み込み
W	FrameWork 上に作られた Mac 用ウイジェット
waste	"WorldScript-Aware Styled Text Engine" へのインターフェース

5.1 applesingle — AppleSingle デコーダー

5.2 buildtools — BuildApplet とその仲間のヘルパーモジュール

5.3 py_resource — Python コードからのリソース生成

このモジュールは **BuildApplet** と **BuildApplication** のヘルパーモジュールとして主に利用されています。コンパイル済みの Python コードに 'PYC ' リソースを付加できます。

5.4 cfmfile — コードフラグメントリソースを扱うモジュール

cfmfile は、コードフラグメントと関連する "cfrg" リソースを処理するモジュールです。このモジュールでコードフラグメントを分解やマージできて、全てのプラグインモジュールをまとめて、一つの実行可能ファイルにするため、BuildApplication によって利用されます。

5.5 icopen — open() と Internet Config の置き換え

icopen をインポートすると、組込み open() を新しいファイル用にファイルタイプおよびクリエーターを設定するために Internet Config を使用するバージョンに置き換えます。

5.6 macerrors — MacOS のエラー

macerrors は、MacOS エラーコードを意味する定数定義を含みます。

5.7 macresource — スクリプトのリソースを見つける

macresource はスクリプトが MacPython 上や MacPython アプレットおよび OSX Python 上で起動されている時、特別な処理をせずにダイアログやメニューなどのようなリソースを見つけるためのヘルパースクリプトです。

5.8 Nav — NavServices の呼出し

Navigation Services の低レベルインターフェース。

5.9 mkcwproject — CodeWarrior プロジェクトの作成

mkcwproject は Metrowerks CodeWarrior 開発環境用のプロジェクトファイルを作成します。これは distutils のためのヘルパーモジュールですが、より多くの制御のために独立して利用できます。

5.10 nsremote — Netscape OSA モジュールのラッパー

nsremote は Netscape の OSA モジュールのラッパーであり、好きなブラウザに対して URL を簡単に送ることができます。*Python Library Reference* に記述のある webbrowser モジュールと緊密な関係があります。

5.11 PixMapWrapper — PixMap オブジェクトのラッパー

PixMapWrapper は PixMap オブジェクトを Python オブジェクトでラップしたもので、各フィールドに対し名前でアクセスできるようになります。PIL 画像との相互の変換をするメソッドも用意されています。

5.12 preferences — アプリケーション初期設定管理プログラム

preferences モジュールを使うと、ユーザの環境設定をシステム全体の環境設定フォルダ中に保存できます。このとき、ユーザの環境設定にアプリケーション自体のデフォルト設定を持たせたり、特定の状況で環境設定をオーバライドさせたりできます。

5.13 pythonprefs — Python の初期設定管理プログラム

このモジュールは Python インタプリタの初期設定の読み込みや書き込みを行えるように preferences モジュールを特化したものです。

5.14 quietconsole — 不可視の標準出力

quietconsole を使うと、バッファの stdio 出力を表示せずに (あるいは、**EditPythonPrefs** で設定されていれば、stdout ウィンドウを全く表示しないで) 保存できます。保存されるのは、 stdin から読み込みを始めるか、バッファリングを止めるかするまでの間で、その時点で全ての出力は今度はウィンドウに送られることになります。 グラフィカルユーザインターフェース (GUI) プログラムで、クラッシュ時の出力を表示したい場合に便利です。

5.15 videoreader — QuickTime ムービーの読み込み

videoreader は QuickTime ムービーを読み込み、デコードし、プログラムへ渡せます。このモジュールはさらにオーディオトラックをサポートしています。

5.16 W — FrameWork 上に作られたウイジェット

W ウィジェットは、**IDE** で頻繁に使われています。

5.17 waste — Apple 製ではない **TextEdit** の置き換え

参考資料:

About WASTE

(http://www.merzwaren.com/waste/)

WASTE ウィジェットとライブラリに関する情報サイト。ドキュメントとダウンロードもここから行なえます。

付録 A

歴史とライセンス

A.1 Python の歴史

Python は 1990 年代の始め、オランダにある Stichting Mathematisch Centrum (CWI, http://www.cwi.nl/ 参照) で Guido van Rossum によって ABC と呼ばれる言語の後継言語として生み出されました。その後多くの人々が Python に貢献していますが、Guido は今日でも Python 製作者の先頭に立っています。

1995 年、 Guido は米国ヴァージニア州レストンにある Corporation for National Reserch Initiatives (CNRI, http://www.cnri.reston.va.us/ 参照) で Python の開発に携わり、いくつかのバージョンをリリースしました。

2000 年 3 月、 Guido と Python のコア開発チームは BeOpen.com に移り、 BeOpen PythonLabs チームを結成しました。同年 10 月、PythonLabs チームは Digital Creations (現在の Zope Corporation, http://www.zope.com/ 参照) に移りました。 そして 2001 年、Python に関する知的財産を保有するための非営利組織 Python Software Foundation (PSF、 http://www.python.org/psf/ 参照) を立ち上げました。このとき Zope Corporation は PSF の賛助会員になりました。

Python のリリースは全てオープンソース (オープンソースの定義は http://www.opensource.org/ を参照してください) です。歴史的にみて、ごく一部を除くほとんどの Python リリースは GPL 互換になっています; 各リリースについては下表にまとめてあります。

リリース	ベース	年	権利	GPL 互換
0.9.0 thru 1.2	n/a	1991-1995	CWI	yes
1.3 thru 1.5.2	1.2	1995-1999	CNRI	yes
1.6	1.5.2	2000	CNRI	no
2.0	1.6	2000	BeOpen.com	no
1.6.1	1.6	2001	CNRI	no
2.1	2.0+1.6.1	2001	PSF	no
2.0.1	2.0+1.6.1	2001	PSF	yes
2.1.1	2.1+2.0.1	2001	PSF	yes
2.2	2.1.1	2001	PSF	yes
2.1.2	2.1.1	2002	PSF	yes
2.1.3	2.1.2	2002	PSF	yes
2.2.1	2.2	2002	PSF	yes
2.2.2	2.2.1	2002	PSF	yes
2.2.3	2.2.2	2002-2003	PSF	yes
2.3	2.2.2	2002-2003	PSF	yes
2.3.1	2.3	2002-2003	PSF	yes
2.3.2	2.3.1	2003	PSF	yes
2.3.3	2.3.2	2003	PSF	yes
2.3.4	2.3.3	2004	PSF	yes

注意:「GPL 互換」という表現は、Python が GPL で配布されているという意味ではありません。Python のライセンスは全て、 GPL と違い、変更したバージョンを配布する際に変更をオープンソースにしなくてもかまいません。GPL 互換のライセンスの下では、GPL でリリースされている他のソフトウェアと Python を組み合わせられますが、それ以外のライセンスではそうではありません。

Guido の指示の下、これらのリリースを可能にしてくださった多くのボランティアのみなさんに感謝します。

A.2 Terms and conditions for accessing or otherwise using Python

PSF LICENSE AGREEMENT FOR PYTHON 2.4

1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"), and the Individual or Organization ("Licensee") accessing and otherwise using Python 2.4 software in source or binary form and its associated documentation.

2. Subject to the terms and conditions of this License Agreement, PSF hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Python 2.4 alone or in any derivative version, provided, however, that PSF's License Agreement and PSF's notice of copyright, i.e., "Copyright © 2001-2004 Python Software Foundation; All Rights Reserved" are retained in Python 2.4 alone or in any derivative version prepared by Licensee.

3. In the event Licensee prepares a derivative work that is based on or incorporates Python 2.4 or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Python 2.4.

4. PSF is making Python 2.4 available to Licensee on an "AS IS" basis. PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, PSF MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON 2.4 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 2.4 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 2.4, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.

7. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between PSF and Licensee. This License Agreement does not grant permission to use PSF trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.

8. By copying, installing or otherwise using Python 2.4, Licensee agrees to be bound by the terms and conditions of this License Agreement.

BEOPEN.COM LICENSE AGREEMENT FOR PYTHON 2.0
BEOPEN PYTHON OPEN SOURCE LICENSE AGREEMENT VERSION 1

1. This LICENSE AGREEMENT is between BeOpen.com (“BeOpen”), having an office at 160 Saratoga Avenue, Santa Clara, CA 95051, and the Individual or Organization (“Licensee”) accessing and otherwise using this software in source or binary form and its associated documentation (“the Software”).

2. Subject to the terms and conditions of this BeOpen Python License Agreement, BeOpen hereby grants Licensee a non-exclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use the Software alone or in any derivative version, provided, however, that the BeOpen Python License is retained in the Software, alone or in any derivative version prepared by Licensee.

3. BeOpen is making the Software available to Licensee on an “AS IS” basis. BEOPEN MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, BEOPEN MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

4. BEOPEN SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF THE SOFTWARE FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

5. This License Agreement will automatically terminate upon a material breach of its terms and conditions.

6. This License Agreement shall be governed by and interpreted in all respects by the law of the State of California, excluding conflict of law provisions. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between BeOpen and Licensee. This License Agreement does not grant permission to use BeOpen trademarks or trade names in a trademark sense to endorse or promote products or services of Licensee, or any third party. As an exception, the “BeOpen Python” logos available at http://www.pythonlabs.com/logos.html may be used according to the permissions granted on that web page.

7. By copying, installing or otherwise using the software, Licensee agrees to be bound by the terms and conditions of this License Agreement.

CNRI LICENSE AGREEMENT FOR PYTHON 1.6.1

1. This LICENSE AGREEMENT is between the Corporation for National Research Initiatives, having an office at 1895 Preston White Drive, Reston, VA 20191 (“CNRI”), and the Individual or Organization (“Licensee”) accessing and otherwise using Python 1.6.1 software in source or binary form and its associated documentation.

2. Subject to the terms and conditions of this License Agreement, CNRI hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Python 1.6.1 alone or in any derivative version, provided, however, that CNRI’s License Agreement and CNRI’s notice of copyright, i.e., “Copyright © 1995-2001 Corporation for National Research Initiatives; All Rights Reserved” are retained in Python 1.6.1 alone or in any derivative version prepared by Licensee. Alternately, in lieu of CNRI’s License Agreement, Licensee may substitute the following text (omitting the quotes): “Python 1.6.1 is made available subject to the terms and conditions in CNRI’s License Agreement. This Agreement together with Python 1.6.1 may be located on the Internet using the following unique, persistent identifier (known as a handle): 1895.22/1013. This Agreement may also be obtained from a proxy server on the Internet using the following URL: http://hdl.handle.net/1895.22/1013.”

3. In the event Licensee prepares a derivative work that is based on or incorporates Python 1.6.1 or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Python 1.6.1.

4. CNRI is making Python 1.6.1 available to Licensee on an "AS IS" basis. CNRI MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, CNRI MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON 1.6.1 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.

5. CNRI SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 1.6.1 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 1.6.1, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.

7. This License Agreement shall be governed by the federal intellectual property law of the United States, including without limitation the federal copyright law, and, to the extent such U.S. federal law does not apply, by the law of the Commonwealth of Virginia, excluding Virginia's conflict of law provisions. Notwithstanding the foregoing, with regard to derivative works based on Python 1.6.1 that incorporate non-separable material that was previously distributed under the GNU General Public License (GPL), the law of the Commonwealth of Virginia shall govern this License Agreement only as to issues arising under or with respect to Paragraphs 4, 5, and 7 of this License Agreement. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between CNRI and Licensee. This License Agreement does not grant permission to use CNRI trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.

8. By clicking on the "ACCEPT" button where indicated, or by copying, installing or otherwise using Python 1.6.1, Licensee agrees to be bound by the terms and conditions of this License Agreement.

ACCEPT

CWI LICENSE AGREEMENT FOR PYTHON 0.9.0 THROUGH 1.2

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Stichting Mathematisch Centrum or CWI not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

STICHTING MATHEMATISCH CENTRUM DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL STICHTING MATHEMATISCH CENTRUM BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

A.3 Licenses and Acknowledgements for Incorporated Software

This section is an incomplete, but growing list of licenses and acknowledgements for third-party software incorporated in the Python distribution.

A.3.1 Mersenne Twister

The `_random` module includes code based on a download from http://www.math.keio.ac.jp/~matumoto/MT2002/emt19937ar.html. The following are the verbatim comments from the original code:

```
A C-program for MT19937, with initialization improved 2002/1/26.
Coded by Takuji Nishimura and Makoto Matsumoto.

Before using, initialize the state by using init_genrand(seed)
or init_by_array(init_key, key_length).

Copyright (C) 1997 - 2002, Makoto Matsumoto and Takuji Nishimura,
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:

 1. Redistributions of source code must retain the above copyright
    notice, this list of conditions and the following disclaimer.

 2. Redistributions in binary form must reproduce the above copyright
    notice, this list of conditions and the following disclaimer in the
    documentation and/or other materials provided with the distribution.

 3. The names of its contributors may not be used to endorse or promote
    products derived from this software without specific prior written
    permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE COPYRIGHT OWNER OR
CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL,
EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO,
PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR
PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF
LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING
NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.


Any feedback is very welcome.
http://www.math.keio.ac.jp/matumoto/emt.html
email: matumoto@math.keio.ac.jp
```

A.3.2 Sockets

The `socket` module uses the functions, `getaddrinfo`, and `getnameinfo`, which are coded in separate source files from the WIDE Project, http://www.wide.ad.jp/about/index.html.

```
Copyright (C) 1995, 1996, 1997, and 1998 WIDE Project.
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
1. Redistributions of source code must retain the above copyright
   notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright
   notice, this list of conditions and the following disclaimer in the
   documentation and/or other materials provided with the distribution.
3. Neither the name of the project nor the names of its contributors
   may be used to endorse or promote products derived from this software
   without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE PROJECT AND CONTRIBUTORS ``AS IS'' AND
GAI_ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED.  IN NO EVENT SHALL THE PROJECT OR CONTRIBUTORS BE LIABLE
FOR GAI_ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON GAI_ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN GAI_ANY WAY
OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.
```

A.3.3 Floating point exception control

The source for the `fpectl` module includes the following notice:

A.3.4 MD5 message digest algorithm

The source code for the `md5` module contains the following notice:

```
Copyright (C) 1991-2, RSA Data Security, Inc. Created 1991. All
rights reserved.

License to copy and use this software is granted provided that it
is identified as the "RSA Data Security, Inc. MD5 Message-Digest
Algorithm" in all material mentioning or referencing this software
or this function.

License is also granted to make and use derivative works provided
that such works are identified as "derived from the RSA Data
Security, Inc. MD5 Message-Digest Algorithm" in all material
mentioning or referencing the derived work.

RSA Data Security, Inc. makes no representations concerning either
the merchantability of this software or the suitability of this
software for any particular purpose. It is provided "as is"
without express or implied warranty of any kind.

These notices must be retained in any copies of any part of this
documentation and/or software.
```

A.3.5 Asynchronous socket services

The `asynchat` and `asyncore` modules contain the following notice:

```
Copyright 1996 by Sam Rushing

                        All Rights Reserved

Permission to use, copy, modify, and distribute this software and
its documentation for any purpose and without fee is hereby
granted, provided that the above copyright notice appear in all
copies and that both that copyright notice and this permission
notice appear in supporting documentation, and that the name of Sam
Rushing not be used in advertising or publicity pertaining to
distribution of the software without specific, written prior
permission.

SAM RUSHING DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE,
INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN
NO EVENT SHALL SAM RUSHING BE LIABLE FOR ANY SPECIAL, INDIRECT OR
CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS
OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT,
NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN
CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
```

A.3.6 Cookie management

The `Cookie` module contains the following notice:

```
Copyright 2000 by Timothy O'Malley <timo@alum.mit.edu>

                All Rights Reserved

Permission to use, copy, modify, and distribute this software
and its documentation for any purpose and without fee is hereby
granted, provided that the above copyright notice appear in all
copies and that both that copyright notice and this permission
notice appear in supporting documentation, and that the name of
Timothy O'Malley  not be used in advertising or publicity
pertaining to distribution of the software without specific, written
prior permission.

Timothy O'Malley DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS
SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS, IN NO EVENT SHALL Timothy O'Malley BE LIABLE FOR
ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS
ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR
PERFORMANCE OF THIS SOFTWARE.
```

A.3.7 Profiling

The `profile` and `pstats` modules contain the following notice:

```
Copyright 1994, by InfoSeek Corporation, all rights reserved.
Written by James Roskind

Permission to use, copy, modify, and distribute this Python software
and its associated documentation for any purpose (subject to the
restriction in the following sentence) without fee is hereby granted,
provided that the above copyright notice appears in all copies, and
that both that copyright notice and this permission notice appear in
supporting documentation, and that the name of InfoSeek not be used in
advertising or publicity pertaining to distribution of the software
without specific, written prior permission.  This permission is
explicitly restricted to the copying and modification of the software
to remain in Python, compiled Python, or other languages (such as C)
wherein the modified or derived code is exclusively imported into a
Python module.

INFOSEEK CORPORATION DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS
SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS. IN NO EVENT SHALL INFOSEEK CORPORATION BE LIABLE FOR ANY
SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER
RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF
CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN
CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
```

A.3.8 Execution tracing

The `trace` module contains the following notice:

```
portions copyright 2001, Autonomous Zones Industries, Inc., all rights...
err...  reserved and offered to the public under the terms of the
Python 2.2 license.
Author: Zooko O'Whielacronx
http://zooko.com/
mailto:zooko@zooko.com

Copyright 2000, Mojam Media, Inc., all rights reserved.
Author: Skip Montanaro

Copyright 1999, Bioreason, Inc., all rights reserved.
Author: Andrew Dalke

Copyright 1995-1997, Automatrix, Inc., all rights reserved.
Author: Skip Montanaro

Copyright 1991-1995, Stichting Mathematisch Centrum, all rights reserved.


Permission to use, copy, modify, and distribute this Python software and
its associated documentation for any purpose without fee is hereby
granted, provided that the above copyright notice appears in all copies,
and that both that copyright notice and this permission notice appear in
supporting documentation, and that the name of neither Automatrix,
Bioreason or Mojam Media be used in advertising or publicity pertaining to
distribution of the software without specific, written prior permission.
```

A.3.9 UUencode and UUdecode functions

The `uu` module contains the following notice:

```
Copyright 1994 by Lance Ellinghouse
Cathedral City, California Republic, United States of America.
                        All Rights Reserved
Permission to use, copy, modify, and distribute this software and its
documentation for any purpose and without fee is hereby granted,
provided that the above copyright notice appear in all copies and that
both that copyright notice and this permission notice appear in
supporting documentation, and that the name of Lance Ellinghouse
not be used in advertising or publicity pertaining to distribution
of the software without specific, written prior permission.
LANCE ELLINGHOUSE DISCLAIMS ALL WARRANTIES WITH REGARD TO
THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS, IN NO EVENT SHALL LANCE ELLINGHOUSE CENTRUM BE LIABLE
FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT
OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Modified by Jack Jansen, CWI, July 1995:
- Use binascii module to do the actual line-by-line conversion
  between ascii and binary. This results in a 1000-fold speedup. The C
  version is still 5 times faster, though.
- Arguments more compliant with python standard
```

A.3.10 XML Remote Procedure Calls

The `xmlrpclib` module contains the following notice:

```
    The XML-RPC client interface is

Copyright (c) 1999-2002 by Secret Labs AB
Copyright (c) 1999-2002 by Fredrik Lundh

By obtaining, using, and/or copying this software and/or its
associated documentation, you agree that you have read, understood,
and will comply with the following terms and conditions:

Permission to use, copy, modify, and distribute this software and
its associated documentation for any purpose and without fee is
hereby granted, provided that the above copyright notice appears in
all copies, and that both that copyright notice and this permission
notice appear in supporting documentation, and that the name of
Secret Labs AB or the author not be used in advertising or publicity
pertaining to distribution of the software without specific, written
prior permission.

SECRET LABS AB AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD
TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANT-
ABILITY AND FITNESS.  IN NO EVENT SHALL SECRET LABS AB OR THE AUTHOR
BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY
DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS
ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE
OF THIS SOFTWARE.
```

付録 B

日本語訳について

B.1 このドキュメントについて

この文書は、Python ドキュメント翻訳プロジェクトによる Macintosh Library Modules Release の日本語訳版です。日本語訳に対する質問や提案などがありましたら、Python ドキュメント翻訳プロジェクトのメーリングリスト

http://www.python.jp/mailman/listinfo/python-doc-jp

または、プロジェクトのバグ管理ページ

http://sourceforge.jp/tracker/?atid=116\&group_id=11\&func=browse

までご報告ください。

B.2 翻訳者一覧 (敬称略)

osawa <osawa at sm.rim.or.jp> (2.0)
sakito <sakito at s2.xrea.com> (2.3)
Hiroyuki Yoshimura <DQB00103 at nifty.ne.jp> (2.3, 2.4)
Yasushi Masuda <y.masuda at acm.org> (2.3.3)

MODULE INDEX

V

W

INDEX

Symbols

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W