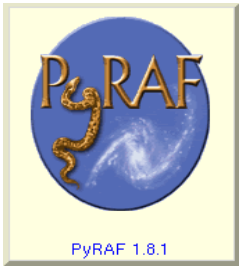
PYTHON SCRIPTING WITH PYRAF/PYFITS

古澤　久徳
国立天文台天文データセンター
2010年2月24日・IRAF講習

Pythonとは

- スクリプト言語のひとつ (Ruby, Perl, AWK など)
  - http://www.python.org/ (Python Software Foundation)
- インタプリタ言語なので、コンパイルは不要
  - 書き換えたら即実行。デバッグや再解析も簡単
- 欧米で、スクリプト言語の主流になってきているため、たくさんの有用な機能拡張用ソフトウェアが使える
- 実際、多くの天文学関係者に浸透しつつある
  - われわれ天文研究者にとってはかなり重要なこと
  - HST解析ソフトウェアの一部 (ACS MultiDrizzle)
  - LSST解析ソフトウェア
  - すばる望遠鏡MOIRCS MOSデータの解析ソフトウェア
  - すばる望遠鏡の次世代観測システム　など

PyRAFとは



- IRAFの演算エンジンを、スクリプト言語Pythonから関数として利用できるようにするためのソフトウェア
- STScIが開発をしている、stsci_pythonという天文学研究用ソフトウェアの中の一部
- http://www.stsci.edu/resources/software_hardware/pyraf
- PyRAFだけを単体でインストールして利用することができます（IRAFがインストールされていることは必須）。講習会マシンにインストールされているのもこちらのバージョン。

（この実習では、参加者の理解の促進のために、用語の正確さは省き、平素な言葉を使うように努めます）

実習の作業環境について

- サンプルファイルは以下のディレクトリ構成で置かれ、サンプルデータはすべて data/に入っています。すでに各ディレクトリに必要最低限のデータが data/へのシンボリックリンクで置かれています。誤削除などで困ったら data/を覗いてください
- sample/disp
  mkflat
  ffield
  multiprocess
  plot
  data　＃全データが入っています
- まず、上記を各自の作業ディレクトリにコピーしましょう
  - % cd /adc/data/[ユーザID (guest??)] ＃各自作業用
  - % cp –rpf /data/furusawa/sample/* .

実習の作業環境について２

- 以下の解説では、演習に用いるファイルのパスをdisp/、 mkflat/などで示します
- サンプルスクリプトは、各ディレクトリの中で実行すると正しく動くように書かれています。必要なデータはカレントディレクトリに置きます
- 各ディレクトリで作業を始める際には、それぞれ一度ずつmkirafを行ってください
- 演習などでコードを編集などされる際は、disp、mkflatなどの各ディレクトリを

  %cp –rpf disp my_disp などのようにコピーして、my_disp内で作業を行うとよいかもしれません
- 必ずしもこのように開発するのが便利なわけではありませんが、実習中の混乱を避けるためです

PyRAFを起動・終了してみる

お膳立てとして、

- まずは disp/ というディレクで（あるいはそれを、各自の専用ディレクトリにコピーして）作業しましょう
- % cd disp
- % mkiraf → xgterm　をえらびます
- login(user).cl　を編集します
  - set stdimage=imt4096　# Suprime-Camのデータを使うため

起動

- % pyraf　＃PyRAFを起動します
  - 引数に –n をつけると起動時ポップアップがでない
  - 引数に前回セッションの途中ファイルを指定できる

終了

- - -> .exit　とします
  - bye　や logout　は使わない

PyRAFをeclの代わりに使ってみる

- eclと同じ感覚で、IRAFのシェルとして使える
- cl互換のコマンド呼びだし（!ds9、disp　など）ができ、ほとんどの clスクリプトを呼び出すこともできる（ようだ）
- メリット
  - コマンド履歴、コマンド補完の利用（tcshと同様の使い勝手。セッションを保存、再開できる）
  - Pythonのインタラクティブプロンプトとして使うことができる
    - たとえば 1+1　とか、数学関数、for loop、if 文などが使える
  - Help, GUIの強化など。
- できないこと
  - packageのアンロードはできない
  - goto文はない
  - IRAFタスク/clスクリプトのバックグラウンドジョブは使えない（!ds9&などはこれまで通りできます）　　など

強化されたGUI

- 外部ウインドウ表示の機能が強化されている

その一例として。。

- epar imstat　で imstat の解析パラメータを設定し、FITS画像の統計値を測定してみよましょう
  - - -> epar imstat
  - eparのGUIからヘルプを参照してみましょう
- ds9と displayタスクを使って、Suprime-Camのデータを表示してみましょう
  - - -> ！ds9 &　（#ds9は先に起動しておきます）
  - - -> display SUPA00207540.fits 1
- imexamで天体の形状を調べてみましょう
  - - -> imexam SUPA00207540.fits
  - - -> ds9の画像上で、a, e, j, kなどしてみます。qで終了します
  - Windowタブからグラフィックウインドウを複数開くこともできます

Scripting言語としてのPyRAF

- PyRAFは、IRAF用インタラクティブシェルとして使うだけでもメリットがあると言えます。しかし、
- PyRAFを使うことの一番の強みは、Pythonスクリプトの中で、関数としてIRAFタスクを利用できることにあります
- clやUNIX shellでもIRAFタスクをスクリプトとしてつなぎあわせることは可能ですが、Pythonは、文字列処理も強力で、すでに他にも多数の有用な（科学数学演算、可視化など）関数が利用できるので、それらとPyRAFタスクを組み合わせて解析アプリケーションを作ることで、大きな恩恵を得ることができます

IRAFタスクをPythonの関数として呼んでみる

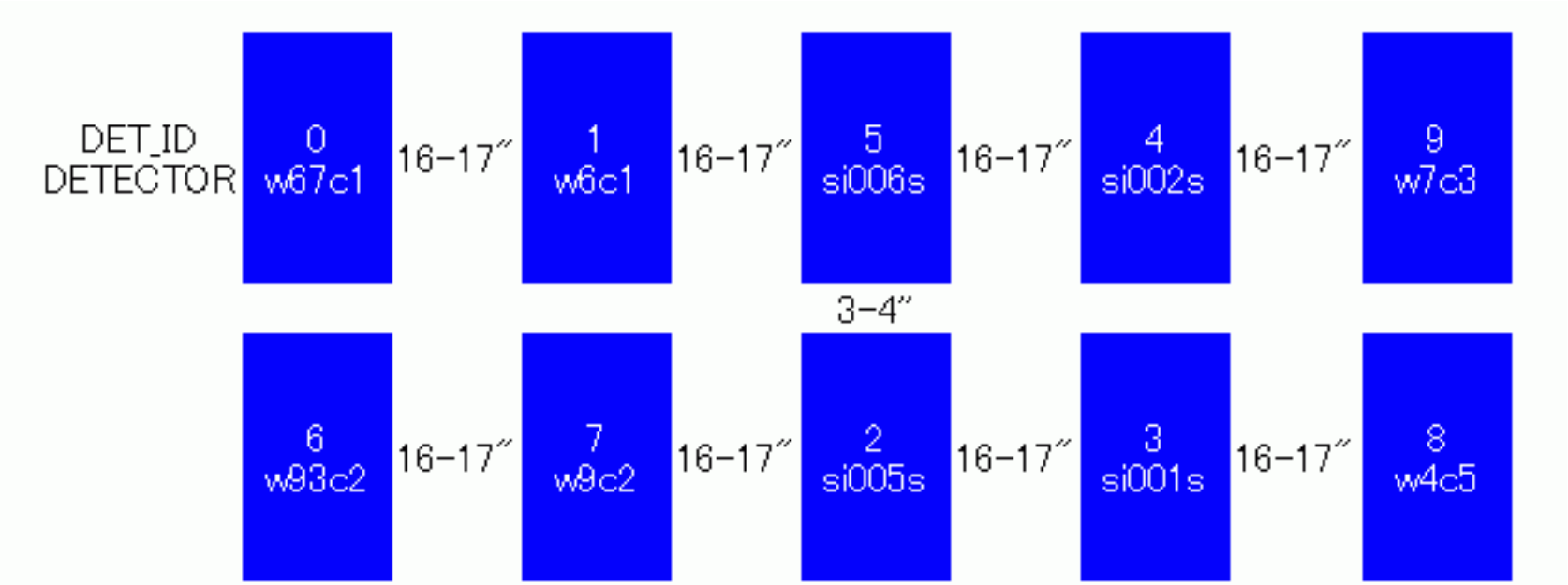
- Python関数としてIRAFタスクを呼ぶときは
  - - -> iraf.display('dev$pix', 1)
  - - -> display dev$pix 1　と同じ意味になります
- いくつかパラメータを追加したり変更したりして、動作を試してみましょう。作業用FITSデータは data/にあります

  たとえば
  - iraf.display('SUPA00207540.fits', 2)
  - iraf.display('SUPA00207540.fits', 2, fill=1, zscale=0)
  - iraf.display('SUPA00207540.fits', 2, fill=True, zscale=True, zrange=False, z1=None, z2=None)
  - iraf.display(image='SUPA00207540.fits', frame=2, zscale=0, zrange=0, z1=10000, z2=20000)

  など。（SUPA*.fitsはSubaru/Suprime-Camのraw dataです）
- 違うタスクも呼んでみましょう
  - iraf.imstat('SUPA00207540.fits[1000:1100,2000:2100]')
  - 領域指定も文字列の一部として渡します

Suprime-Camのデータ

- 10個のCCDが5x2のアレンジでに並んでいる
- 1CCD=1FITSファイル、1ショット＝10 FITS
- ヘッダのDET-IDキーワードまたはファイル名＝SUPA番号の下１桁の数字で、どのCCDかを区別できる



http://www.naoj.org/Observing/Instruments/SCam/ccd_mit.html より

Pythonスクリプトを使ってみる – Suprime-Camの１ショット10CCDを１コマンドで表示してみる

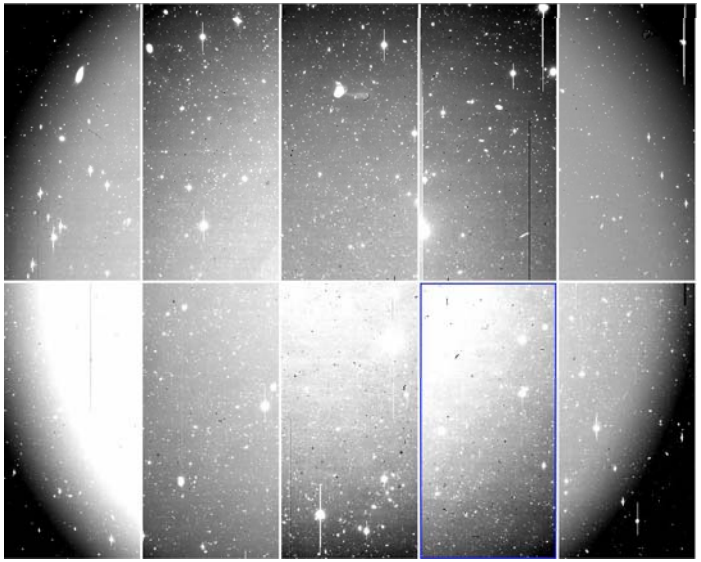
- お膳立て
  - ds9を開きます
    - - -> !ds9 &
  - ds9のメニューから、Frame→Frame Parameters→Tile→ Tile Parameters→Manual 5x2 [Apply]。Frame→tile　としておきます
  - 作業ディレクトリにある uparm/tvdisply.par （最初にiraf.display を実行したときに作られます）の中身を見て、”frame,i,a,1,1,16,…”という行の最後の数字が16 になっていることを確認します。この数字がds9で同時に開けるフレームの個数です。
- 一番単純なPyRAFスクリプト ～コマンドの羅列
  - disp/dispSUPE0.py

Pythonスクリプトを使ってみる – Suprime-Camの１ショット10CCDを１コマンドで表示してみる

- 繰り返し処理 (for loop) を使ってみましょう
  - disp/dispSUPE1.py （ファイル名はhard-coded）
  - Pythonスクリプトでは、インデント（字下げ：通常４文字）でループや関数のスコープ（作業単位）をあらわします。C言語の{ … } に相当します
  - 変数は宣言不要で、気にせず使います
- 演習1－1：サンプルスクリプトを書き換えて他のショットを読み込んでみてください
- FITS名をファイルから読み込んでみましょう
  - disp/dispSUPE2.py input.lis
  - ファイルの閉じ忘れなど気にしなくて大丈夫です

フレーム番号とCCDの対応付け

- FITS名をファイルから読み込む一続き
  - このままでは、正しい配置で表示されていません



Surime-Camの正しいCCD配置
視野周辺では光学系のけられがあるので、
視野周辺では円形に暗くなり、左のような光量分布になります。

  - CCDの並びを正しくするには input2.lis を使います
  - 演習1-2：別の入力リストを作って他のショットを表示してみましょう

余裕のある人用の演習

- プログラムを発展させてみましょう
  - disp/dispSUPE2.py では、CCDの並び順にFITS名をリストしたinput2.lis をショットごとに毎回準備しなければなりません。CCDのリストがフレーム番号順に並んでいなくても、正しい配置で表示できるようにしてみましょう
  - たとえば、CCD番号（DET-ID)とds9 のFrame番号の組を別のテキストファイル (ccd_arrange.lis) としておいて利用できないでしょうか
- 参考：
  - disp/dispSUPE3.py — SUPA0020754というコマンド引数を与えると、SUPA00207540.fits – SUPA00207549.fits を表示する
  - disp/dispSUPE4.py --- 10FITSの名前を書いたリストを与えると、正しい配置で表示する
  - きっとよりよい手法があると思いますで、考えついたらぜひ教えてください。

（さらに物足りない人へ）ds9との通信を行うXPAというツールを使うことで、もう少し自動化を進めることもできると思います。Pyds9という、PythonからXPAを利用できる外部ソフトウェアがあります

解析アプリケーションの開発：フラットを作ってみましょう

- お膳立て：Supirme-Camのデータを練習に使いましょう。一般にフラット作成はCCD番号（DET-ID)　ごとに解析を行います。DET-ID=0 のデータ (バイアス引き算済み: data/oss_SUPA*0.fits) で作業しましょう
- やるべき作業を考えてみましょう

1. 各フレームのスカイの高さを測り 1 に規格化する
2. それらをmedianで合成する

作業１ー カウントの規格化

- 作業をブレークダウンします
  1. フレーム全面でmedianを測る
  2. その値を変数 に保存
  3. 全面のピクセル値を そのmedianで割る（規格化）
- これをIRAFのタスクで考えると。。？
  - （PyRAFコマンドプロンプトで確認してみます）
  1. iraf.imstat(fits, fields='midpt')
  2. ret=iraf.imstat(fits, fields='midpt', Stdout=1) ＃返り値はリストになっていますので、もう一手必要です
  3. iraf.imarith(fits, '/', med, outfits)
- 演習

2-1. imstatからmedianの値を受け取り、float値として変数に入れてみましょう。まずはPyRAFコマンドプロンプトで動作を確かめてください

カウント規格化ー演習続き

2-2. 規格化プログラムとしてまとめたPythonスクリプトが (mkflat/norm1.py)です。

- 内容を理解して動作確認してみてください
- nrm_*.fitsファイルが存在すると、上書きできない旨のエラーになります。事前に削除しておいてください
- Median測定でシグマクリップ法を使ってみるなど、独自に変更をしてみましょう

2-3. dispでの作業を思い出して、ファイル名をリストファイルから取り入れるように変更しましょう（参考：mkflat/norm2.py,　oss.listが入力リスト)

（発展）余裕のある人は、アルゴリズムをコマンドラインで指定したり、medianを測る領域を指定できるようにしてみましょう　（参考：mkflat/norm2ext.py）

作業２－規格化済み画像の合成

- 作業をブレークダウンします
  1. 規格化済み画像のリストを作成する
  2. それらの画像を合成（スタック）して1枚の画像にする
- IRAFタスクで考えると。。
  1. Pythonの機能でリストを作成する → fitslist
  2. iraf.imcombine(input=fitslist, combine='median', output=flatname)　＃形式
     fitslist='fits1,fits2,fits3,…,fitsn'

規格化済み画像の合成ー続き

- 演習

2-4. imcombタスクの入力として渡すための、ファイルリスト(fitslist) を作る箇所をコーディングしてみましょう

  - 思いつく手段でよいです。ファイルを１行読むごとに文字列をつなぎ合わせていくのでもよいと思います

2-5. 作ったリストを入力値として、画像を1枚のフラットへとスタックしましょう。スタックアルゴリズムも「外れ値のclippingあり」等に変えてみましょう

  - 注）あらかじめ、先のnorm2.pyなどを走らせてnrm_*.fitsをカレントディレクトリに作成しておきます

参考：mkflat/comb1.py

作業単位を関数にしてつなげてみる

- 作業１と２でできた解析ソフトウェアをひとつのアプリケーションとして組み上げてみましょう
- それぞれの作業を関数としておくほうが、ソースの可読性もよく、今後の開発効率の面でも有利です
- 関数は def で定義します
  - 関数の名前と引数を示した定義を最初に書き、関数の中身は一段インデントして書きます
- mkflat/norm2.py を関数norm()にしてみましょう
  - 参考：mkflat/norm3.py
  - 関数を使うときの都合を考えて、mkflat/norm2.pyでコマンドラインの引数になっていた入力リストは関数の引数としてあります

関数の組み上げ―続き

- 演習

  2-6. mkflat/comb1.pyを同じように関数comb()にしてみましょう。コマンド引数は関数引数に。(参考：mkflat/comb2.py）

  2-7. norm() と comb() をつなげてひとつのPythonアプリケーション mkflat.py にしてみましょう

  参考：mkflat/mkflat1.py（第一版）, mkflat/mkflat2.py（完成版）, mkflat/mkflat3.py （おまけ）

  - 出力ファイルはスクリプト実行前に消しておいてください

- 参考までに

  - % python　＃pyrafではなく、生のpythonです
  - >>> import norm3　＃機能拡張ソフトとして読み込みます
  - >>> norm3.norm(‘oss.list‘)　#とすると。。？
  - こういう使い方もできます、という例です
  - >>> ^D　Pythonは（コントロール＋D）で終了します

続き

- ぜんぜん物足りない人のための演習課題
  - できれば10 CCDをDET-IDを意識することなく入力し、１コマンドで10 CCDのフラットが完成するようにしたいですよね。どういう追加をすればよいでしょうか？
  - その解析結果を、自動表示してみるというのはいかがでしょうか？（さきほどの 10 CCD表示プログラムを参考に）
  - 参考：mkflat/mkflat4.py
    - 実行前にds9であらかじめFrame1から10を作ってタイル表示しておきます。oss_*.fits 50ファイルを作業ディレクトリにおきます
    - mkflat4.py oss.list flat とすると flat[0-9].fits が作られます

フラットフィールディングしてみましょう

- せっかくなので、作ったフラットで各OBJECTファイルを割ってみましょう（フラットフィールディングと呼び、ピクセル間の感度を補正する操作です）
- 演習

  3-1. ffield.py input.list flat0.fits というようなコマンドで、flt_oss_SUPA*.fitsというフラットフィールド済み画像を作るプログラムを作りましょう

  - 出力ファイルが存在するとエラーになるので、flt_oss_*.fitsは削除しておいてください。（慣れてきた方は access, imdeleteタスクなどを使って自動削除させるとよいかもしれません）

  3-2. 入力とフラットのFITSヘッダを参照し、FILTER01 キーワードを比較して、異なったらエラーメッセージを出力するようにしてみましょう

  - 参考：ffield/ffield1.py, ffield/ffield2.py

Pythonの威力を体験する

- おめでとうございます。ここまで完了された皆さんは、Web上の情報や参考書などをもとに、十分自力で習得を進めていただけることと思います。（実際、ググってみると、Pythonによる実験科学プログラミングのtipsはインターネット上に多数ころがっています）
- ここまでの内容にも、Pythonスクリプト内だから簡単に実現できた作業が多々あるのですが、このくらいならclで組んだほうが早いや、という方もいらっしゃるかもしれません
- ここからは、Pythonならではの機能も使ってみましょう

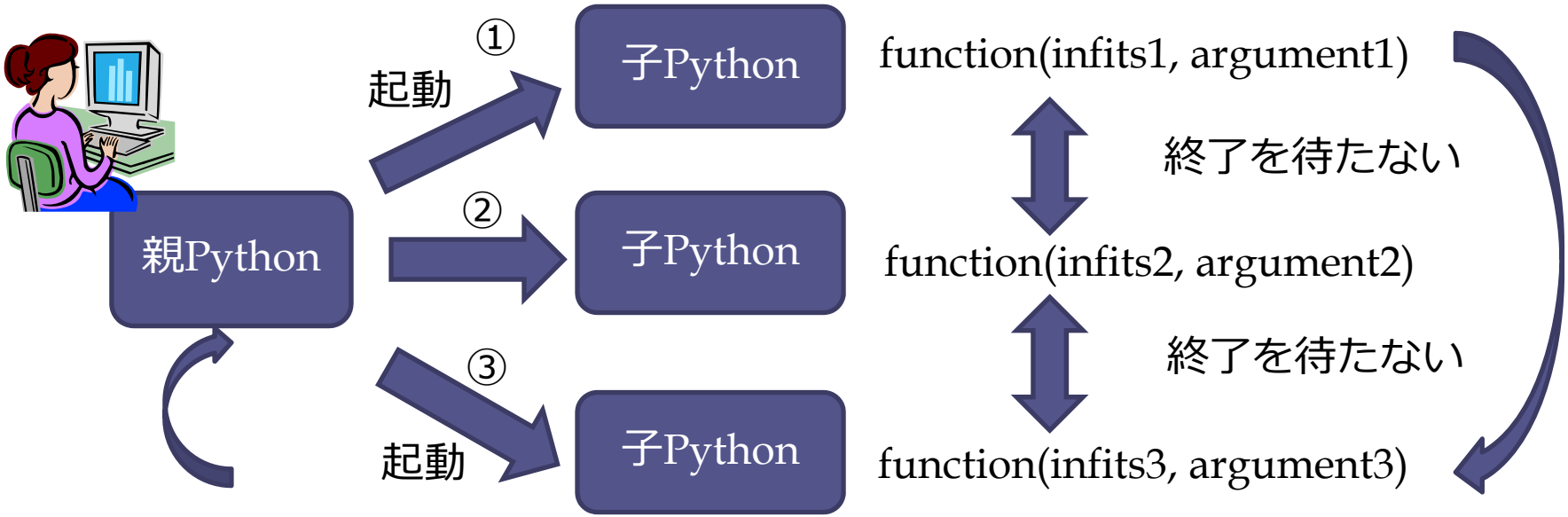
マルチCPUコアを生かす

- 近年のPCには、複数のコアを持つCPUが搭載されているものが珍しくなくなりました。大量にデータ処理する際に、この特性を生かしたいものです
- Python2.6 の multiprocessingという機能拡張ソフトウェア（モジュール）を使うと、Pythonの子プロセスを簡単に自動で複数個起動し、独立な繰り返し作業などを効率的に行えます
- 参考：ffield/ffield3.py
  - 多重起動するべきプロセスの個数を設定する
  - 複数のPythonの中で並列に実行したい関数と、それぞれのプロセスで使われる関数の引数を入力リストとしてあたえる
- 演習（注！Solarisで動作確認済。LinuxではPyRAFタスクとの相性で動作しないことがあります）

  4-1. Pythonの子プロセス起動個数を変えてみて、topコマンドなどで、実際に指定の個数だけPythonが起動することを確かめてみましょう。

  余裕がある方は、さきほどのnorm()などをマルチプロセス化するなど、応用を試されるとよいでしょう。

どういうことかというと。。。



Pool (子の数) → map (関数と引数)

```
function( [
            (infits1, argument1),
            (infits2, argument2),
            (infits3, argument3),
              …]   )
```

引数の組み合わせのリストを与えると、それぞれの組が別々のPythonで関数functionの引数として使われるわけです

プロット用の関数を利用する

- Pythonには、スクリプト中の変数値などを用いて簡単にプロットを作成できる拡張ソフトウェアがあります。

  - Matplotlib　http://matplotlib.sourceforge.net/
  - Gnuplot.py　http://gnuplot-py.sourceforge.net/　など
- Matplotlibを用いて、フラット済み画像の結果簡易チェックツールを作ってみましょう
- 作業項目をブレークダウンします
  - 画像の上下２領域の高さを測り、両者が何%異なるかを求める
  - その値をグラフに表示する
- 参考 plot/check_ff.py

プロット用の関数ー続き

- 演習

  5-1. たとえば統計値を取るメッシュを増やして最大・最小値の差（peak to peak difference）がスカイレベルの何%にあたるか表示してみましょう

  5-2. 横軸をデータのEXPTIMEなど、サンプルコードのフレーム番号とは別の値を軸にとりなおしてプロットできるでしょうか

  余裕がある方：　peak-to-peak差に加えて、スカイ高さのばらつきの標準偏差がスカイレベルの何%にあたるかなども表示してみましょう

  - Pythonだけでコーディングするのは少し面倒かもしれませんが、import math 、math.sqrt()　などを使えます
  - 参考：plot/check_ff2.py、plot/check_ff3.py (NumPyを使用)

- ここではMatplotlibの関数を細かくは述べませんので、用途に応じて機能を調べるようにされるとよいでしょう

PyFITSとNumPy

- 今回は、PythonとPyRAFタスクによるスクリプティングとはどういうものか、という説明に注力しましたので、PyFITSについてはご紹介のレベルにとどめます
- PyFITS

  - http://www.stsci.edu/resources/software_hardware/pyfits
  - Pythonにおける、FITSIO/CFITSIOに相当するもので、FITSデータを読み書き、またピクセルデータとヘッダ（HDU）の加工などができる関数群が使える追加ソフトウェアです。
  - PyRAFと同様にSTScIのstsci_pythonの一部であり、PyRAFとは独立の独立です
  - PyFITSを用いたコーディングでは、ピクセルデータの操作にNumPyを利用します
- NumPy

  - http://numpy.scipy.org/
  - 配列の操作をメインとした数学・科学用関数をPythonから利用できるようにする機能拡張ソフトウェアです（バックエンドはCなどで実装されていて高速）

PyFITSを使ったプログラム開発

- メリット
  - PyFITSはIRAFに依存しないので、軽量にプログラムを組むことができ、IRAFタスクに存在しないアルゴリズムや操作を比較的簡単に開発できます
  - プログラム配布時など、IRAFが入っているシステムは限られるが、PyFITS＋NumPyなら壁が低いかも
  - NumPyの強力な配列操作関数を使うことで、ピクセルデータの操作を比較的容易に行えます。また、NumPyの強力な数学関数もあわせて利用できます
- デメリット
  - ある程度、解析タスクをPythonの枠組みで独力で開発するための知識と経験が必要となります（日常の研究生活で便利に使うというレベルなら、ちょっとした努力でOK）

PyFITSを使ったプログラム例：mkflat

- さきほど作ったmkflat/mkflat2.pyをPyFITSを使って書きかえてみます
- 各IRAFタスクはおおむね次のように書き換えられます。もちろん、PyRAFとの混合もできます
  - FITSデータのHDUからピクセルデータとヘッダ情報を取得
  - ピクセルデータ配列を演算・操作し、結果のピクセルデータ配列を生成
  - 結果をFITSデータとして書き出し
  - この方法がベストというわけではないですが。

- 参考：mkflat/mkflat2pyfits.py
  - どうしても気になる人は、 mkflat2.pyで作ったフラットとmkflat2pyfits.pyで作ったフラットを割り算してみましょう
- 比較的簡単に同等の解析プログラムを組めることがわかっていただければよいです

ほかの有用な機能

- mathモジュール（モジュール＝機能拡張ソフトウェア）
  - 基本的な数学関数群です
- datetimeモジュール
  - 日時の足し引きや、文字列フォーマットを扱えます。観測は時間との戦いですから、結構使う機会があるかもしれません
- ftplibモジュール
  - 文字通りFTPプロトコルを扱えます。データ転送は観測の一環でもあります。存在を知っておいて損はないです
- re モジュール
  - 正規表現です。文字列操作関数だけで行き詰まったら考慮するとよいでしょう
- 上記はPythonのデフォルトでインストールされます
- ほか
  - いいモジュールを見つけたら（作ったら）ぜひ教えて下さい

まとめ

- PythonとPyRAFの使い方を浅く取り上げましたので、不明な点も多いと思います。なんとかなりそうだな、便利かもしれない、という感触を少しでも持ってもらえたらば、幸いです。
  - 今回はClassなど、オブジェクト指向言語としての使い方には触れていません。興味をもたれたら挑戦されるとよいと思います。
- Pythonによるデータ処理には得手不得手の面があるので、上手に他のソフトウェアと組み合わせてプログラムを作るとよいでしょう
  - たとえば入出力ファイルのリスト生成や結果の評価などをPythonの機能を使って、
  - ピクセルごとに演算を繰り返すような作業はPyRAFのタスクや、NumPyの数学関数など。。といったアプローチがありえます

Appendix - Pythonの手法

- ファイルから一行ずつ読み出す
  - for line in open(filename):　＃ファイルの最後まで繰り返し
    line=line.strip()　＃行頭行末の空白・改行文字を削除
    line=line.rstrip(',') ＃文字列右端の','を削除
    line=line.split()　＃空白文字で分割
    val_col1=line[0]
    val_col2=line[1] …
- 文字列の操作
  - 数値に変換、数値を文字列に変換
  - val_int=int(string)
  - val_float=float(string)
  - val_string=str(val_numeric)
- コマンドラインから引数の値を取得する
  - hikisuu1=sys.argv[1]
  - argc = len(sys.argv)　＃関数len()はリストの項目数を返す

続き

- リストをつくり、値を追加していく
  - a=[]　＃空のリストを作っておいて
  - a.append(new_value) ＃値を追加していきます
  - a.sort()　　＃ リストをソートします
  - print a[0]　＃できたリストの第１要素を表示します
  - print a[-1]　＃リストの最終要素を表示します
  - print a[0:5]　＃　１から５要素目（index=0,…,4) を表示します
  - ‘ ,‘.join(a)　　＃ ’,’を境界文字として要素を連結し文字列にします
  - リストの他に、タプル（内容の変更付加）、辞書（キーとその値のペア、追加は可、順序は不定）などのコンテナ組み込み型があります
- “困ったときの”system関数もあります
  - import os
  - os.system(command_line_string)

続き

- 便利なループ処理

```
>>>  a=[‘nausicaa’, ‘kiki’, ‘fio’,’sophie’, ‘sheeta’]
>>>  for  indx, ccdname  in  enumerate(a):
            print  indx, ccdname
.. 0  ‘nausicaa’
.. 1  ‘kiki’   …
>>>  b={‘kaze’:’nausicaa’,  ‘majo’:’kiki’ , ‘buta’:’fio’]
>>>  for keyword, heroine  in  b. iteritems():
            print  keyword, heroine
… ‘kaze’  ‘nausicaa’
… ‘majo’  ‘kiki’
… ‘buta’   ‘fio’
```

- 使える関数が分からなかったら？
  - % python
  - >>> a=[]
  - >>> dir(a)
  - >>> help(a)

  ＃リストに付随する関数の情報

```
>>>  import math
>>>  dir(math)
>>>   help(math)
```

＃mathモジュールに付随する関数の情報