

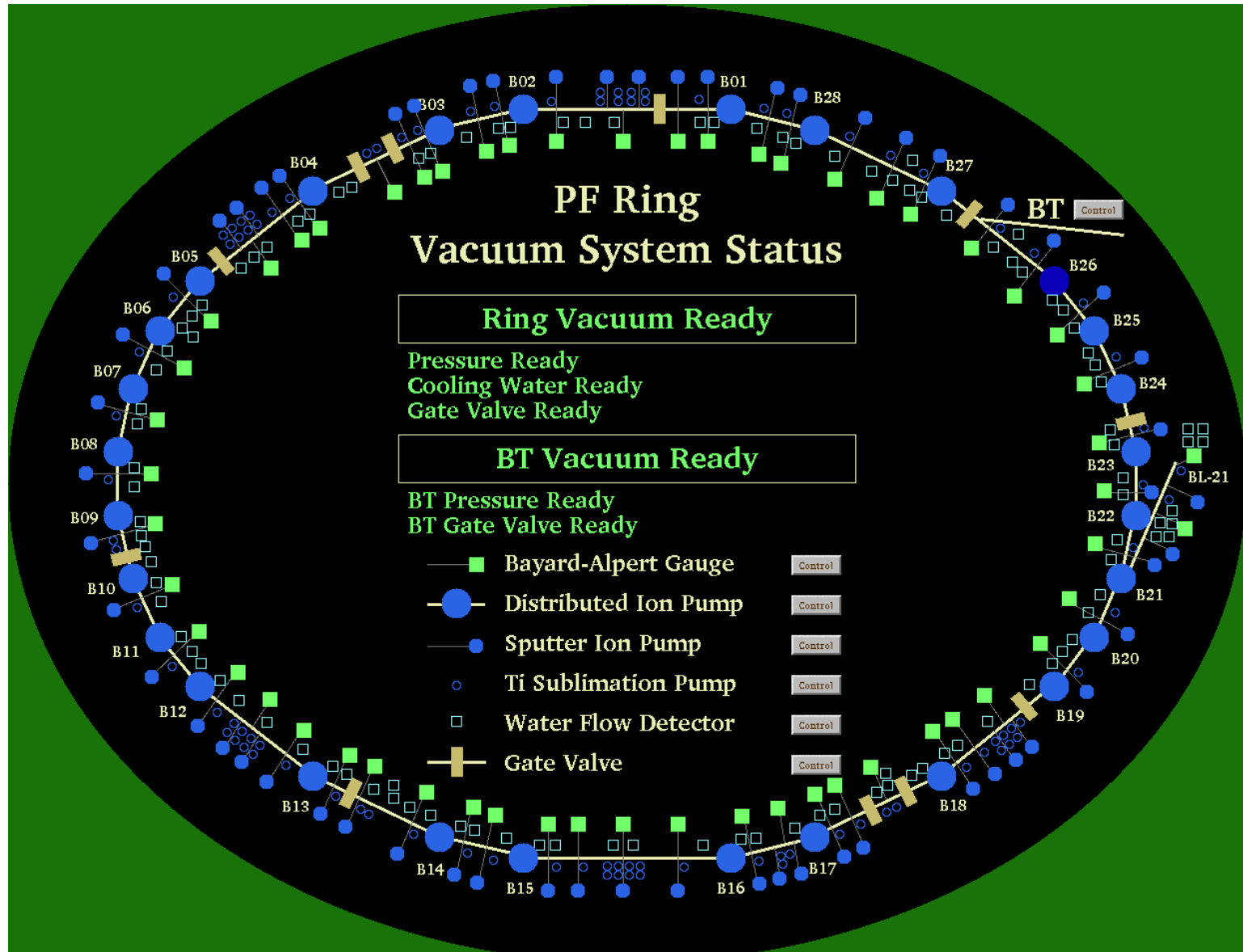
EPICSでのGUI Python/Tkinter

帯名 崇
飛山 真理

EPICSでのGUI

- EPICSでのデータ表示ツール(標準)
 - medm
 - dm2k
 - edm
- その他のツール
 - Tk toolkitを利用するもの
 - tcl/tk, python/Tkinter, perl/Tk, SAD/Tk等
 - 商用のソフトウェア,開発ツールを使うもの
 - Matlab, LabView,
 - Delphi, MS Visual Studio (Visual Basic, C#)
 - データをプロットするだけならばgnuplotも可

例：medm, edm, dm2k



例：SAD/Tkinter

The image displays a complex software interface for RF system control, likely for a particle accelerator. It consists of several interconnected windows and panels.

RF System Control Panel (Top Left): This panel contains input fields for RF Voltage (1.7200 MV), Beam energy (2.5000 GeV), and RF Frequency (500.105893 MHz). It also features a table of RF Status for four stations (#1 to #4) with columns for Operation, Process, Filament, H V, R F, and Beam Ready. Buttons for Start Operation, Pause, Stop RF, and Shutdown are at the bottom.

PF-RF Parameters (Top Middle): This window displays a table of parameters for four stations. The parameters include Vc, Energy, Klystron power, Cavity power, Reflected power, Cavity CCG, Tuner position, Klystron input power, DC high voltage, Klystron beam current, Filament current, Klystron IP current, Cavity water flow, Cavity in-water temp, and Cavity out-water temp.

PF Operation Panel (Top Right): This panel shows a list of operation parameters (MAG, RF, VAC, CHN, CON, MON, WIG, INJ, ID, OP) and their corresponding control programs (Magnet Control, RF Control, Vacuum Control, etc.).

Sequencer Status Monitor for RF Control (Bottom Left): This panel displays a table of status for four stations (#1 to #4) with columns for RF System Control, All RF Parameters, RF Frequency, PF-RF Parameter, Plot PF-RF Parameter, Sequencer Status, and RF power monitor. It also includes a Power Calculation section with a PCAL status field.

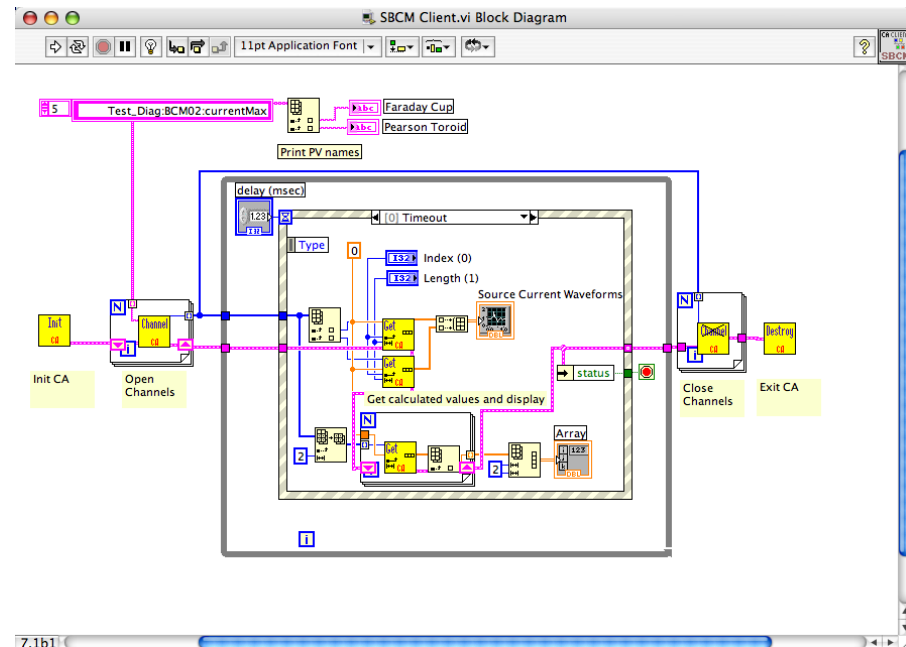
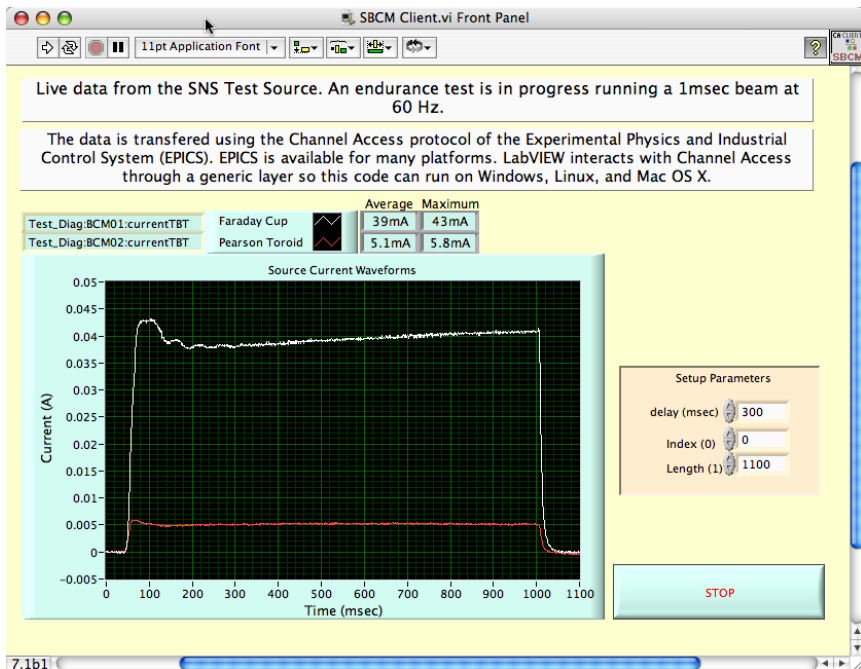
Plot PF-RF Parameters (Bottom Right): This window contains multiple bar charts showing various parameters for four stations. The parameters include Klystron power (kW), Cavity power (kW), Cavity reflection (kW), Cavity CCG (V), Cavity in-water temp. (deg.), Tuner position (cm), DC high voltage (kV), Klystron beam current (A), and Klystron IP current (uA).

Terminal (Bottom Right): A terminal window showing command-line input and output, including commands like `cd tmp`, `ls`, and `import RF_desktop.png`.

例：LabVIEW

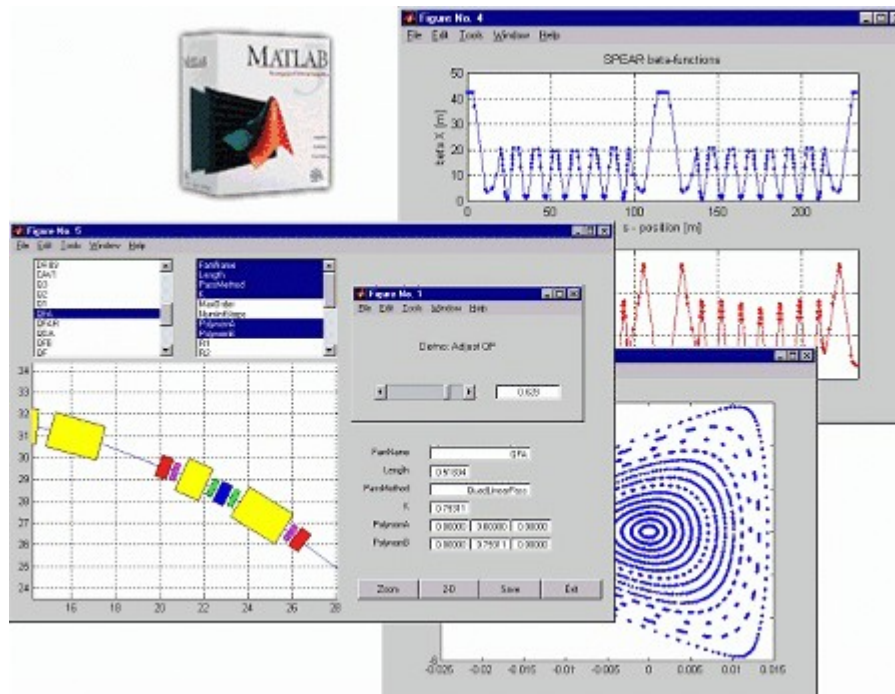
- SNSで開発

- http://neutrons.ornl.gov/diagnostics/documents/epics/LabVIEW/SNS_LabVIEWEPICS.html



例 : Matlab

- Stanford SSRL
 - <http://www-ssrl.slac.stanford.edu/at/>
- その他、最近あちこちのリングで使用されている



GUIに何を選ぶか

- 一人でやっている or テストベンチなど、独立したシステムならば、何を使っても良い
- 制御グループが指針を出している場合、それに従うのが良い。相談できる相手が身近にいることは重要。
 - 気合と実力があれば”こっちの方が良い！”と主張して布教に努めるという方法もあり。
- まずは標準のmedm(dm2k)、edmなどを使ってみる。不満があれば、他のGUIを使う。
- 商用ソフトに手を出す場合は慎重に。バージョンアップなどで使えなくなることも。
- OSにも注意。他のプログラムと連携する必要があるかどうか。

Python入門

- Pythonとは
 - スクリプト言語
 - 軽量言語(Lightweight Language)
 - Linux, Windows, MacOSなど、各種OSで動作
 - オブジェクト指向言語だが、オブジェクト指向的に作らないことも可能.....習得は容易(だと思う)
 - オープンソース
- コメント
 - 今回はpythonを使いますが、「これでなければいけない」ということはありません。スクリプト言語を1つ知っておくと色々便利です。perlでもrubyでも好きならHaskellでも構いません。

実習1: Pythonを使ってみる

- 対話的シェル(インタラクティブシェル)で実行
- 起動と終了
- 電卓として使う

```
$ python
>>> print "Hello!"
Hello!
>>> 1+2
3
>>> print 2+3
5
>>> 2**10
1024
>>> hex(16)
'0xa'
>>> 0x10
16
>>> ^D
```

```
>>> 1/2
>>> 1.0/2

>>> ord('A')
>>> chr(65)

>>> dir()
['__builtins__', '__doc__',
 '__name__']

>>> dir('__builtins__')
```

変数の定義: 宣言不要

```
>>> x = 1.23
>>> y = x + 2.3
>>> print x+y
>>> print x*y
>>> z = 3.4
>>> print x*y
```

文字列、スライス

```
>>> s = "abcdef"
>>> print s
>>> s[0]
>>> s[2:3]
>>> s[-2]
>>> s[:-3]
>>> s[-3:]
>>> s[10]
```

文字列の演算

```
>>> s + "ABC"
>>> s*3
```

```
>>> s = "234.5"
>>> float(s)
>>> int("345")
>>> len(s)
```

リスト

```
>>> a = [1, 5, 2, 4]
>>> a
>>> b = a + [4, 'AB', 3, 'C']
>>> b
>>> a[2]

>>> max(a)
>>> min(a)
>>> range(10)
```

リストに対して関数(メソッド)を適用

```
>>> a.sort()
>>> a.reverse()
>>> a.append('DD')
```

リスト要素の置き換え、削除

```
>>> b[2] = 'Two'
>>> del b[3]
```

多次元配列

```
>>> c = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
>>> c[2][0]
```

辞書(ディクショナリ)、タプル : 実習では省略

```
{key1:item1, key2:item2, .... }
(0.2.3.1)
```

スクリプトファイルでの実行

- 毎回手入力は論外
- テキストファイルに書いて、実行する

実行方法: テキストファイルに書いて
\$ python filename.py

gnomeエディタを使っている場合の設定
「編集」→設定
自動インデント有効
フォントをCourierに設定
タブは8のまま
行番号表示ON

filename.pyファイルを開いた場合や、名前をつけて保存で.py拡張子を付ければ色分けしてくれる。最初に
表示→強調表示モード→スクリプト→Python
を選ぶ方法もある。

Pythonとインデント

- CやC++,Java等、他の言語では論理ブロックは{}で閉じたり、行末は;をつける

```
main()  
{  
    int i;  
    for (i=0; i<10; i++) {  
        printf("i=%d\n", i);  
    }  
}
```

- Pythonでは字下げ(インデント)で表記
 - 他の言語では見栄えをよくするためだがpythonでは必須

条件式等 →

```
print "start..."  
for i in range(10):  
    print i  
    if i==5:  
        print "in the middle"  
        print "....."  
        pass  
    print "inside loop"  
print "finished."
```

インデント

条件分岐、ループ、関数

```
for i in range(10):  
    print i
```

```
j=0  
while j<10:  
    print j  
    i = j+1
```

```
if j==5:  
    print "aaa"  
elif i==7:  
    pass  
else:  
    pass
```

比較演算子: == < > != <= >=

論理演算子: and or

例:

```
if age<=18 or age>=65:  
    ...実行文...
```

switch....case文はありません

関数の定義

```
def func1(arg1,arg2):  
    print "arg1=", arg1  
    print "arg2=", arg2  
    return arg1*arg2
```

```
func1(1,2)  
func1("ABC", 3)
```

クラスの定義(オブジェクト指向)

```
class MyClass:  
    def __init__(self):  
        self.myattr = "initA"  
    def myfunc1(self, arg1):  
        self.myattr = arg1  
    def myfunc2(self):  
        print "MyCA=",self.myattr
```

```
obj1 = MyClass()  
print obj1.myattr  
obj1.myfunc1("modifiedA")  
print obj1.myattr  
obj1.myfunc2()
```

インスタンスの作成
メソッド呼び出し

組み込み関数やモジュールを使う

強力な文字列操作

```
buf = "This is a pen"  
buf.split()  
buf.split("i")
```

```
buf.upper()  
buf.lower()  
buf.find("is")  
buf.rfind("is")
```

その他 : join, index, strip等

文字列フォーマット

```
"i=%d, j=%d, x=%f\n"%(i,j,x)
```

ファイルの読み書き

```
finp = open('tmp.txt')  
fout = open('tmp.out', 'w')  
i=1  
for line in finp.readlines():  
    buf = line.upper()  
    fout.write("%d: %s"%(i,buf))  
    i=i+1  
finp.close()  
fout.close()
```

import文でモジュールをインポート

```
>>> import sys  
>>> dir(sys)  
>>> sys.version  
>>> sys.exit()
```

```
>>> import os  
>>> os.getcwd()
```

```
>>> import time  
>>> time.asctime()  
>>> time.localtime()  
>>> time.localtime.__doc__
```

```
>>> from time import *  
>>> localtime()
```

その他、便利なモジュールがたくさん。
標準のものや、色々な人が作ったもの等。

参考文献

- Web
 - 本家 www.python.org
 - 日本のサイト www.python.jp
 - その他検索
- 入門用書籍
 - Pythonで学ぶプログラム作法 / アラン ゴールド (著)
 - みんなのPython / 柴田 淳 (著)
 - 初めてのPython 第2版 / マーク ルッツ (著)
- おまけ:
 - プログラミングの習得には役に立つかどうか分かりませんが、頭の体操には
<http://www.pythonchallenge.com/>

Python/CA

- Pythonへ実装されたEPICS Channel Access
- KEK 山本昇氏による
- 現バージョンは Thread Safeです。

KEK EPICSグループのページ

http://www-acc.kek.jp/WWW-ACC-exp/EPICS_Gr/

- 分からないことは山本さんに聞きましょう

実習2: Python/CAを使う

- サンプルデータベース
 - 今までの実習で作成したデータベース
 - excasを使って擬似データ作成 ←今回はこちら

必ず、-pオプションで
自分の名前をつけること！！



端末を1つ開き

```
$ excas -pobina:
```

別の端末を開いて

```
$ caget obina:jane
```

```
$ camonitor obina:jane
```

そのほか、fred, freddy, alan, albert等あり

excas はcaServerのexample programですが、クライアントの動作確認をするには便利なので、今回の実習ではこれを使います。jane, fredで値がどのように変化するか確認してください。

対話的シェルで動作確認

```
$ python
>>> import ca
>>> ca.Get("obina:jane")
obina:jane is connected
2.3095548152923584

>>> ca.Monitor("obina:fred")
obina:fred is connected
obina:fred : 0.404595047235 0 0 Sat Jan 10 11:01:50.593 2008
obina:fred : 0.318113744259 0 0 Sat Jan 10 11:02:00.781 2008
obina:fred : 0.336150437593 0 0 Sat Jan 10 11:02:02.777 2008

>>> ca.ClearMonitor("obina:fred")
```

Python/Tkinterの使用方法

- Tkinter
 - Frame(フレーム)とWidget(ウィジェット)で構築
 - 部品を配置し他の後、mainloop()メソッドを呼ぶとユーザーの入出力を受け付けるようになる
- 対話的シェルで試してみる

```
$ python
>>> import Tkinter
>>> f1 = Tkinter.Frame()
>>> t1 = Tkinter.Label("Hello!")
>>> b1 = Tkinter.Button(f1, text="exit",
                        command=f1.master.destroy) #実際は1行で入力
>>> t1.pack()
>>> b1.pack()
>>> f1.pack()
>>> f1.mainloop()
```

CAとTkinterを使ったGUI：その1

- classを使わない場合（実用的とはいえない）
 - 下のコードをファイルに書いて、実行

```
from Tkinter import *
import ca

def cbfunc(ch, args):
    v1.set(args[0])
    print "cbfunc : ", args[0]

root=Tk()
Label(root,text="val=").pack()
v1=DoubleVar(root)
Label(root,textvariable=v1).pack()
Button(root,text="QUIT",command="exit").pack()

ca.Monitor("obina:fred", cbfunc)

mainloop()

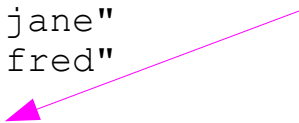
ca.ClearMonitor("obina:fred")
```

CAとTkinterを使ったGUI：その2

```
from Tkinter import *
import ca
import sys
```

```
rec1 = "obina:jane"
rec2 = "obina:fred"
```

Tkinter.Frameクラスを継承してAppクラスを作成



```
class App(Frame):
    def __init__(self, master=None):
        Frame.__init__(self, master)
        self.createpanel()
        ca.Monitor(rec1, self.cbfunc1)
        ca.Monitor(rec2, self.cbfunc2)

    def createpanel(self):
        fr1 = Frame(master, borderwidth=2, relief=RAISED)
        fr1.pack(side=TOP, fill=BOTH, padx=2, pady=2, expand=1)
        fr2 = Frame(master, borderwidth=2, relief=RAISED)
        fr2.pack(side=TOP, fill=BOTH, padx=2, pady=2, expand=1)
        fr3 = Frame(master, borderwidth=2, relief=RAISED)
        fr3.pack(side=TOP, fill=BOTH, padx=2, pady=2, expand=1)

        lblVal1 = Label(fr1, text=rec1).pack()
        self.v1 = DoubleVar(self)
        lbltxt1 = Label(fr1, textvariable=self.v1).pack()

        lblVal2 = Label(fr2, text=rec2).pack()
        self.v2 = DoubleVar(self)
        lbltxt2 = Label(fr2, textvariable=self.v2).pack()

        btnExit = Button(fr3, text="QUIT", command=self.clear_exit)
        btnExit.pack()
        self.pack()
```

class の続き:

```
def cbfunc1(self, ch, args):  
    self.v1.set("%07f"%args[0])
```

```
def cbfunc2(self, ch, args):  
    self.v2.set("%07f"%args[0])
```

```
def clear_exit(self):  
    ca.ClearMonitor(rec1)  
    ca.ClearMonitor(rec2)  
    self.tk.quit()  
    sys.exit()
```

ここからmain

```
if(__name__ == "__main__"):  
    app = App()  
    app.mainloop()
```