Python 内部培训

Python简介

- 快速、高效的开发语言
- 胶水语言
- 生态链完善
- 广泛用于科学计算、数据挖掘等领域

本讲义约定使用Python 2.x版本

3.x版本由于库没有跟上，暂时不推荐使用

语法特色

- 动态语言特性 —— 可在运行时改变对象本身(属性和方法等)
- 基于C/C++和JAVA，但有很大区别
- 缩进方式，建议使用空格，不要用TAB
- 多个语句在一行使用;分隔
- 注释符是#，多行使用docstring(’’’…’’’)
- 变量无需类型定义
- 可进行函数式编程FP
- Python3.x的变迁

编程规范

- PEP8 编码规范
- Google Python 编码规范

Google Python > PEP8编码规范

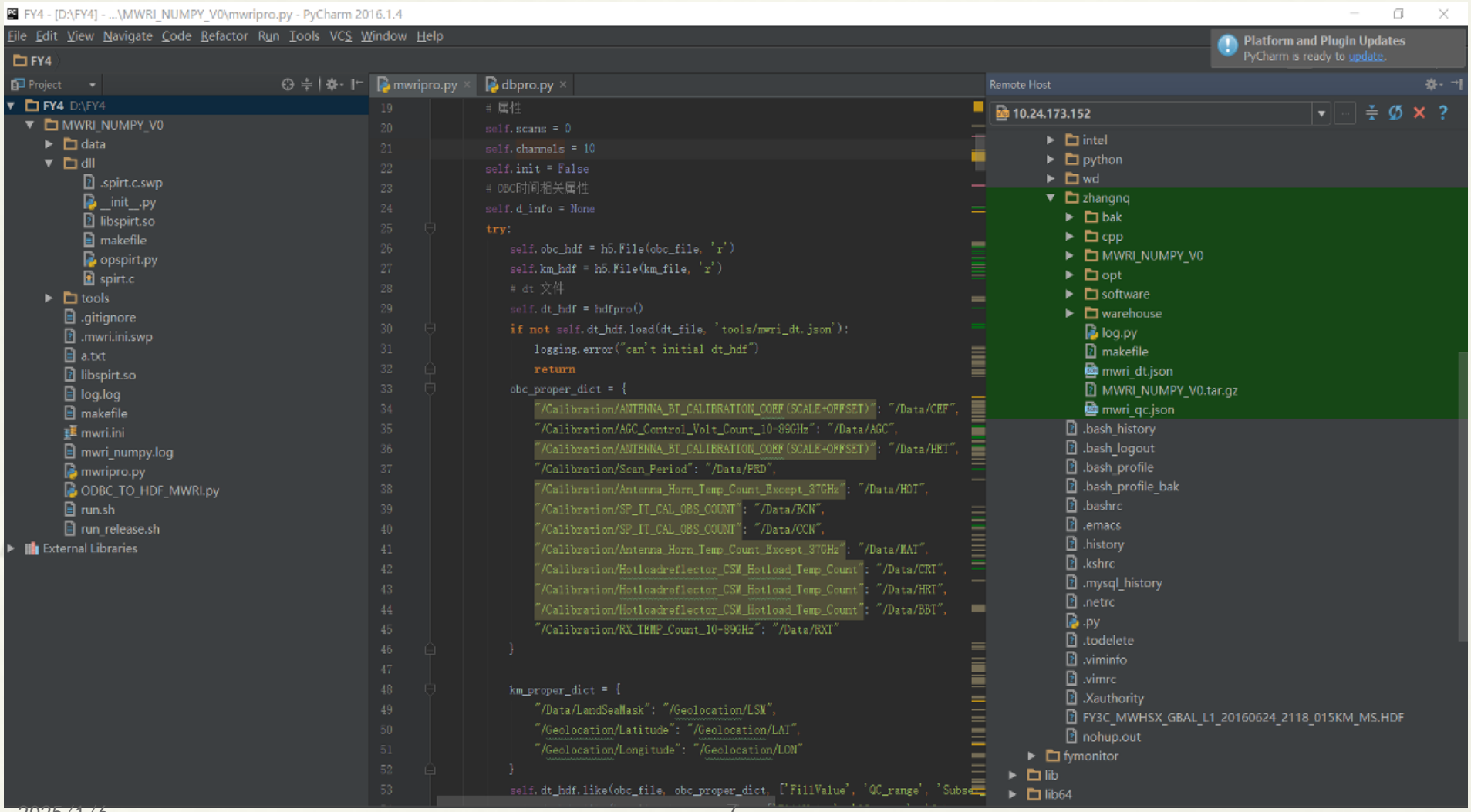
开发环境

- PyCharm（支持PEP8 语法规范、跨平台、远程调试、上传…）
- IPython
- VIM - 主要在linux下使用
- 其它编辑器

  UE，notepad++，editplus…

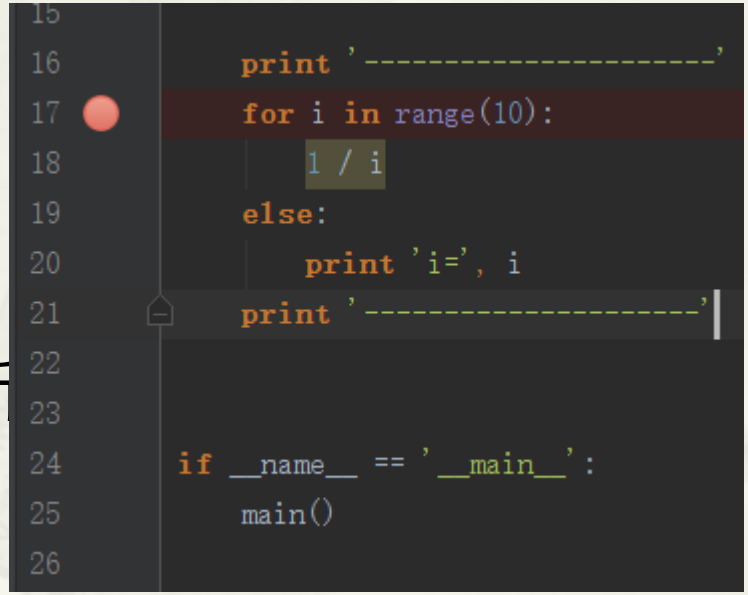
工欲善其，事必先利其器

开发环境


FY4 - [D:\FY4] - ...\MWRI_NUMPY_V0\mwripro.py - PyCharm 2016.1.4
File Edit View Navigate Code Refactor Run Tools VCS Window Help
Platform and Plugin Updates
PyCharm is ready to update.
FY4
Project
FY4 D:\FY4
MWRI_NUMPY_V0
data
dll
.spirt.c.swp
__init__.py
libspirt.so
makefile
opspirt.py
spirt.c
tools
.gitignore
.mwri.ini.swp
a.txt
libspirt.so
log.log
makefile
mwri.ini
mwri_numpy.log
mwripro.py
ODBC_TO_HDF_MWRI.py
run.sh
run_release.sh
External Libraries
mwripro.py
dbpro.py
# 属性
self.scans = 0
self.channels = 10
self.init = False
# OBC时间相关属性
self.d_info = None
try:
self.obc_hdf = h5.File(obc_file, 'r')
self.km_hdf = h5.File(km_file, 'r')
# dt 文件
self.dt_hdf = hdfpro()
if not self.dt_hdf.load(dt_file, 'tools/mwri_dt.json'):
logging.error("can't initial dt_hdf")
return
obc_proper_dict = {
"/Calibration/ANTENNA_BT_CALIBRATION_COEF(SCALE+OFFSET)": "/Data/CEF",
"/Calibration/AGC_Control_Volt_Count_10-89GHz": "/Data/AGC",
"/Calibration/ANTENNA_BT_CALIBRATION_COEF(SCALE+OFFSET)": "/Data/HET",
"/Calibration/Scan_Period": "/Data/PRD",
"/Calibration/Antenna_Horn_Temp_Count_Except_37GHz": "/Data/HOT",
"/Calibration/SP_IT_CAL_OBS_COUNT": "/Data/BCN",
"/Calibration/SP_IT_CAL_OBS_COUNT": "/Data/CCN",
"/Calibration/Antenna_Horn_Temp_Count_Except_37GHz": "/Data/MAI",
"/Calibration/Hotloadreflector_CSM_Hotload_Temp_Count": "/Data/CRT",
"/Calibration/Hotloadreflector_CSM_Hotload_Temp_Count": "/Data/HRT",
"/Calibration/Hotloadreflector_CSM_Hotload_Temp_Count": "/Data/BBT",
"/Calibration/RX_TEMP_Count_10-89GHz": "/Data/RXT"
}
km_proper_dict = {
"/Data/LandSeaMask": "/Geolocation/LSM",
"/Geolocation/Latitude": "/Geolocation/LAT",
"/Geolocation/Longitude": "/Geolocation/LON"
}
self.dt_hdf.like(obc_file, obc_proper_dict, ['FillValue', 'QC_range', 'Subset
Remote Host
10.24.173.152
intel
python
wd
zhangnq
bak
cpp
MWRI_NUMPY_V0
opt
software
warehouse
log.py
makefile
mwri_dt.json
MWRI_NUMPY_V0.tar.gz
mwri_qc.json
.bash_history
.bash_logout
.bash_profile
.bash_profile_bak
.bashrc
.emacs
.history
.kshrc
.mysql_history
.netrc
.py
.todelete
.viminfo
.vimrc
.Xauthority
FY3C_MWHSX_GBAL_L1_20160624_2118_015KM_MS.HDF
nohup.out
fymonitor
lib
lib64

PyCharm 调试

- Ctrl+Alt+S　　--属性配置
- Ctrl+Shift+Alt+L --格式化文件 PPE8
- Ctrl+B　　-- 查看对象
- Ctrl+L　　--查找
- Ctrl+N　　--切换类
- Ctrl+Shift+N　--切换文件
- Shift+F9/F10　--Debug/Run
- 单击行　　--设置断



工欲善其，事必先利其器

PDB调试

- Python -m pdb xxx.py
- b --设置断点 （行，
- s -- step over
- n -- next
- c --continue
- r --run
- L --显示代码
- Exit()

```
[root@sparkmaster basic]# python -m pdb pythonApp.py
> /root/learnpython/basic/pythonApp.py(2)<module>()
-> import sys
(Pdb) l
  1     #!/usr/bin/env python
  2  -> import sys
  3     import time
  4     import math
  5     from app import main
  6     import re
  7     import os
  8     import getopt
  9     import numpy as np
 10
 11
(Pdb) b r_m
Breakpoint 1 at /root/learnpython/basic/pythonApp.py:142
(Pdb) b 131
Breakpoint 2 at /root/learnpython/basic/pythonApp.py:131
(Pdb) r
```

工欲善其，事必先利其器

字符编码

- Python2.x默认的是OS的本地编码
- Python3.x是unicode内部编码
- .py文件第一行:#coding=utf-8，不指定编码时，文件中包含非ASCII字符会报错

```
s1 = "中文1"
s2 = u"中文2"
print unicode(s1,'utf-8').encode('gbk')
print s2.encode('gbk')
print type(unicode(s1,'utf-
  8')),type(s2),type(s2.encode('gbk'))
```

Hello world

- 表达式

```
2 + 3
3 + (7 * 4)
3 ** 5
 'Hello'  +  'World'
```

- 变量赋值

```
a = 4 << 3
b = a * 4.5
c = (a+b)/2.5
a = "Hello World"
x,y = 4+2," python"
```

- pass 语句 - 不做任何事时使用

```
if a < b:
   pass
else:
   c = a
```

None

- Python特有的空值表示
- 与C/C++中的NULL是不同的
- 函数没有明确返回的话，默认返回是None
- 不能与其它类型进行运算

字符串string

- str[::], str[0], str[1:2], str[-1:]
- find/index()　　#没有找到子串，前者不会扔出异常
- replace(),split(), strip()
- “sub” in “str”　　#是否存在子串
- join()

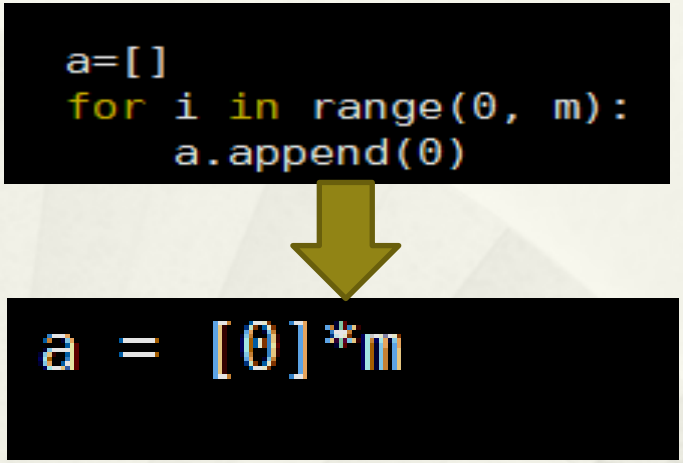
```
>>>lst = [ ‘1’, ’2’, ’abc’, ’4’, ’5’]
>>> ‘,’.join(lst)
 ‘1,2,abc,4,5’
```

列表list

- 赋值

```
a = [2, 3, 4]
b = [2, 7, 3.5, “Hello”]
c = []
d = [2, [a, b]]
e = a + b
```

```
a=[]
for i in range(0, m):
    a.append(0)
```



- 操作

```
x = a[1]
y = b[1:3]  # Retur
z = d[1][0][2]
b[0] = 42
print sum(a)
x = a.pop(0)
```

```
def testlist():
    lst = [(x*x,x) for x in range(1,10) if x > 1 ]
    print 'lst=',lst

lst= [(4, 2), (9, 3), (16, 4), (25, 5), (36, 6), (49, 7), (64, 8), (81, 9)]
```

tuple

- 赋值

```
f = (2,3,4,5)                          # A tuple of integers
g = (,)                                # An empty tuple
h = (2, [3,4], (10,11,12))# A tuple containing mixed objects
```

- 操作

```
x = f[1]                # Element access. x = 3
y = f[1:3]              # Slices. y = (3,4)
z = h[1][1]   # Nesting. z = 4
```

- 特色
  - 与list类似，最大的不同tuple是一种只读且不可变更的数据结构
  - 不可取代tuple中的任意一个元素，因为它是只读不可变更的，也不能进行像list一样的加法操作

字典dict

- 赋值

```
a = { }                                  # An empty dictionary
b = { ’ x’ : 3, ’ y’ : 4 }               #有点类似json格式
c = { ’ uid’ : 105,
      ’ login’ : ’ beazley’ ,
      ’ name’  : ’ David Beazley’
    }
```

- 操作

```
u = c[’ uid’ ]
c[’ shell’ ] = ″/bin/sh″
dict2 = dict2.update(dict1) #使用
if c.has_key(″directory″):
   d = c[’ directory’ ]
else:
   d = None

d = c.get( “directory” ,None) # 常
```

```
def hello():
    print 'hello'

def begin():
    print 'begin'
def hellohash():
    h={'hello':hello,'begin':begin}
    h['hello']()
    h['begin']()
```

集合set

```
>>> set( [“hello”, “world”, “of”, “words”, “of”, “world”] )
    set(['world', 'hello', 'words', 'of'])
```

- 如何删除重复数据

```
Ls1 = [1,3,5,3,7,4,5]
Ls2 = list(set(Ls1))
```

- 可以使用&、|求两个set的交集、并集、补集、全集

```
s1 = set([1,2,3])
s2 = set([2,4])
s1 & s2          #{2}
s1 | s2          #{1,2,3,4}
s1 - s2          #{1,3}
s1 ^ s2          #{1,3,4}
```

Hello world

- if…elif…else语句:
  没有switch，有更高级的变通方式(dict字典方式)

```
if a == ‘+’ :
   b = ‘+’
elif a == ‘-’ :
   b = ‘-’
else:
   b = None
```

- 布尔表达式 - and, or, not

```
if b >= a and b <= c:
   print ‘bool is True’
if not (b < a or c > c):
   print ‘not expr, value is True’
```

循环

- While..else语句

```
while a < b:
   a = a + 1
else:
  print  ‘a=’ ,a
```

- For语句(遍历序列的元素)

```
for item in [3, 4, 10, 25]:
   print item
else:
   print  ‘final’
# Print characters one at a time
for c in "Hello World":
   print c

# Loop over a range of numbers
for i in xrange(0,100,2):
   print i
for i in xrange(len(list1)):
   print list1[i]
```

死循环怎么办？

- 桌面应用可以马上知道，并杀死对应进程
- 服务器应用怎么去监控？
  - 计数器：在循环的最里面计数，超过指定数值就退出，缺点太多了
  - 让函数带有超时功能

```
import threading

def test():
    while 1:
        pass

def funcWithTimeout(timeout):
    t1 = threading.Thread(target=test)
    t1.setDaemon(True)
    t1.start()
    t1.join(timeout)

if __name__ == "__main__":
    print 'start'
    funcWithTimeout(2)
    print 'end'
```