入职前学习

1. GIT开源版本控制工具学习

GIT 和 SVN的差别：SVN集中式存储，GIT分布式存储

* BitKeeper (BitMover 公司)
* 作者：Linus 使用了1天左右的时间开发

**基础命令**：

1. git init //at your directory, run [git init], them it will set it as the repository
2. Git add
3. Git commit –m “your customized message”
4. Git status // get the current modified status of the file
5. Git log –pretty=oneline //show the log in an elegant way
6. Git reflog //show the HEAD pointer
7. Git reset HEAD/HEAD^/HEAD^^/HEAD

git reset --hard commit\_id (//according to the commit id, reset to the older version)

1. Git checkout – file //cancel the modified that hasn’t been add to the temporary zone; if the change has been added to temporary zone, use【git reset HEAD file】to return the file version into work zone.
2. Git rm file //if you incautiously deleted a file, you can use 【git checkout -- file】to resume it. This will return the latest version of this file, but lose the current change you have done on this file after the latest commit. git remote add origin git@server-name:path/repo-name.git
3. git remote add origin [git@server-name:path/repo-name.git](mailto:git@server-name:path/repo-name.git) //connect the romote repository in github
4. git push -u origin master//the first time:push the content to the remote repository
5. git push origin master
6. git checkout –b dev//add a new branch other than master

equal to the following commands:a) git branch <branch name>

b) git checkout <branch name>

1. git branch//checkout the branches we have
2. git merge <name>//merge the branch into master branch
3. git branch –d <name>//delete the branch
4. git checkout <branch name> //switch into the branch

python lambda function:

lambda parameters: expression

call function:函数调用

py2.x 中，函数传参与C++不同，C++分为值传递和引用传参，py2.x全为引用传参。但是，引用传参，分为可改变对象（整数、字符串、元组）和不可改变对象（字典、列表）。

//摘抄段

在C++中，函数传递方式有值传递和引用传递两种方式。python中参数传递类似于引用传递，因为 python中一切都是对象，变量全部是对象的引用。但是要区分参数是不可变对象，还是可变对象。

当函数参数为不可变对象（整数，字符串，元组）时，函数体内的参数在被改变之前，会一直持有该对象的引用，但当参数发生改变时，由于该对象为不可变对象，必须生成一份新的拷贝作为函数的本地变量，函数对该本地变量的修改不会影响函数调用者的变量值，这一点有点类似 C++ 函数的值传递。

def func\_pass\_value(x):

print "before: x=", x, " id=", id(x)

   x = 2

   print "after: x=", x, " id=", id(x)

>>> x = 1

>>> func\_pass\_value(x)

before: x= 1  id= 5363912

after: x= 2  id= 5363888

>>> x, id(x)

(1, 5363912)

当函数参数为可变对象（列表，字典）时，除非发生赋值操作，函数体类的参数会一直持有该对象的引用，函数对该参数的修改也会影响到函数调用者的变量值，类似于C++函数中的引用传递。但在函数体内发生赋值操作时，也会生成一份新的拷贝作为函数的本地变量，函数对该本地变量的修改不会影响到函数调用者的变量值。

def func\_pass\_ref1(list):

print list

list = [4, 5]

print list

def func\_pass\_ref2(list):

print list

list += [4, 5]

print list

>>> al = [1, 2, 3]

>>> func\_pass\_ref1(al)

[1, 2, 3]

[4, 5]

>>> al

[1, 2, 3]

>>> func\_pass\_ref2(al)

[1, 2, 3]

[1, 2, 3, 4, 5]

>>> al

[1, 2, 3, 4, 5]

也就是说：当参数为可改变对象时，函数内对于参数对象的修改，可以影响到引用参数。但，若发生赋值行为，其只修改该对象在函数内产生的副本。

**Python的特性学习基础篇：**

1. 类对象方法:【**\_\_init\_\_**】方法。在一个对象创建的时候，立刻运行。可以用于对对象进行一些初始化。我们没有专门调用**\_\_init\_\_**方法只是在创建一个类的新实例的时候，把参数包括在圆括号内跟在类名后面，从而传递给**\_\_init\_\_**方法。
2. 类变量的域。类的变量，对象的变量。
   1. 类的变量由该类的所有对象共享。类似于C++的类静态变量，类的静态方法。
   2. 对象的变量由类的每个对象/实例拥有。C++中分公共成员，保护成员，私有成员。Python不区分。所有的类成员都是公共的，所有的方法都是有效的。只有一个例外：如果你使用的数据成员名称以双下划线前缀比如\_\_privatevar，Python 的名称管理体系会有效地把它作为私有变量。
3. Python中存在类的继承。也存在通过基类访问子类的，而表现出的多态现象。

2017/04/09

三种攻击手段：

1. SQL Injection

Defence:

1. SQL进行查询时，使用绑定变量方式传递变量。参数化SQL语句
2. 永远不要使用动态拼装SQL，可以使用参数化的SQL或者直接使用存储过程进行数据查询存取
3. 加密
4. Cross-site scripting(XSS)