python 编辑器推荐使用 python tools for visual studio ，个人感觉是最好用的 Python 编辑工具，比 Wing 之类的好用多了。

VPTS 打开使用 Wing 编辑过的 py 文件，如果编辑任何代码都提示语法错误，是因为 python 不允许同时使用 tab 和空格进行格式对齐。（老子鄙视任何一切使用 table 进行对齐的程序员，你他妈没做过跨平台，就不知道有多坑爹）

VS 默认的是使用 4 个空格代替 tab 建的，要想保留 tab 制表符，需要到选项 >> 文本编辑器中修改为保留制表符。

如果出现空格和 table 混用，使用快捷键 CTRL+R,CTRL+W 来切换空格的显示状态，删除不需要的符号

1.读取 python 文件位置

2.C++调用 python 函数

PyArg_ParseTuple 和 PyArg_Parse 的区别

PyArg_ParseTuple 的返回值为元组，返回多个元素

PyArg_Parse 返回值只有一个元素

3. C++调用 python

```
Py_Initialize();                //使用 python 之前，要调用 Py_Initialize();这个函数进行初始化
PyRun_SimpleString( "import sys" );
PyRun_SimpleString( "sys.path.append( '../pythonparser/' )" );
PyRun_SimpleString( "print sys.path" );
PyRun_SimpleString( "print '-----------------------------------'" );




PyObject *pName, *pModule, *pDict, *pFunc, *pArg, *pRetVal;
pModule = PyImport_ImportModule("fishtest");
pFunc = PyObject_GetAttrString(pModule,"getstudent");
pDict = PyModule_GetDict(pModule);
PyCallable_Check(pFunc);
char* pStuName;
int iBufferSize = 0;
int iAge = 0;
char *pSex = NULL;
char* pClass;
PyObject* pResult = PyEval_CallObject(pFunc, NULL);
int iLen = 0;


//"(s#is)s" #返回字符串长度，不带结束符 ()表示元组
//注意字符串不需要上级开辟空间
int iRes = PyArg_ParseTuple(pResult, "(s#is)s", &pStuName, &iLen, &iAge, &pSex, &pClass);
if (pResult && iRes)
{
cout << "Class:" << pClass << "   " << pStuName << "     " << pSex << endl;
}

Py_DECREF(pResult);
```

```
Py_DECREF(pFunc);
pArg = Py_BuildValue("(i, i, i)", 5, 6, 2);


pFunc = PyObject_GetAttrString(pModule,"FishSum");
PyObject* pResult1 = PyEval_CallObject(pFunc, pArg);
int c = 0;
PyArg_Parse(pResult1, "i", &c);
int iValue = PyInt_AS_LONG(pResult1);
Py_DECREF(pResult1);
Py_DECREF(pResult1);
        Py_Finalize();                    //调用 Py_Finalize,这个和 Py_Initialize 相对应的.
return 0;
```

获取 python 异常的代码行和所在文件

```
def tableparser(server, strkey, htmltext):
 restext = ''
 try:
  restext = tableparserEx(server, strkey, htmltext)
 except Exception , e:
  exc_type, exc_value, exc_traceback = sys.exc_info()
  errinfo   = exc_traceback.tb_next
  while(errinfo):   #进入到最内层出错的函数模块
   strerr = 'errorinfo: '
   strerr = strerr + "   filename: " + errinfo.tb_frame.f_code.co_filename
   strerr = strerr + "   functionname: " + errinfo.tb_frame.f_code.co_name
   strerr = strerr + "   lineno: " + str(errinfo.tb_lineno)
   errinfo = errinfo.tb_next
   logging.debug(strerr)
 logtext = strkey + ":      "   + restext
 logging.debug(logtext)
 return restext
```