

ABAQUS Python 二次开发攻略

苏景鹤 江丙云

目录

| | |
|---|----|
| 目录..... | 2 |
| 序一..... | 8 |
| 自序..... | 10 |
| Chapter 1 ABAQUS 二次开发简介..... | 11 |
| 1.1 为什么是 Python..... | 11 |
| 1.2 Python, FORTRAN 与 ABAQUS..... | 13 |
| 1.3 基于 Python 二次开发..... | 14 |
| Chapter 2 Python 能力确认..... | 17 |
| 2.1 测试程序..... | 17 |
| 2.2 程序运行结果..... | 23 |
| Chapter 3 脚本的运行与开发环境..... | 25 |
| 3.1 ABAQUS 中脚本的运行..... | 25 |
| 3.1.1. 命令区 KCLI (kernel command line interface) | 26 |
| 3.1.2 CAE-Run Script..... | 26 |
| 3.1.3 ABAQUS Command..... | 27 |
| 3.1.4 ABAQUS PDE..... | 27 |
| 3.2 选择自己的 Python 开发环境..... | 28 |
| 3.2.1 ABAQUS PDE..... | 28 |
| 3.2.2 IDLE..... | 30 |
| 3.2.3 Notepad++..... | 32 |
| 3.2.4 EditPlus..... | 34 |
| 3.2.5 选择合适的编程环境..... | 39 |
| Chapter 4 Python 数据类型与操作符..... | 41 |
| 4.1 基本数据类型..... | 41 |
| 4.2 列表、元组和字符串..... | 43 |
| 4.2.1 列表 (list) | 43 |
| 4.2.2 元组 (tuple) | 46 |
| 4.2.3 字符串 (str) | 48 |
| 4.2.4 列表、元组和字符串的关系..... | 51 |
| 4.3 字典..... | 51 |
| 4.4 集合..... | 53 |
| 4.5 操作符..... | 55 |
| 4.5.1 赋值操作符..... | 55 |
| 4.5.2 数字类型的操作符..... | 55 |
| 4.5.3 序列类型的操作符..... | 57 |
| 4.5.4 字典和集合的操作符..... | 59 |
| Chapter 5 表达式和流程控制..... | 61 |
| 5.1 表达式和程序执行流程..... | 61 |
| 5.2 分支语句 if-else..... | 62 |
| 5.3 循环语句..... | 64 |

| | |
|----------------------------------|-----|
| 5.3.1 while 循环语句..... | 64 |
| 5.3.2 for 循环语句..... | 65 |
| 5.4 中断和退出..... | 68 |
| 5.4.1 break 语句..... | 68 |
| 5.4.2 continue 语句..... | 70 |
| 5.5 特殊语句 pass..... | 71 |
| Chapter 6 函数..... | 72 |
| 6.1 定义函数..... | 72 |
| 6.2 函数中的参数传递与调用方法..... | 74 |
| 6.3 几个特殊的函数关键字..... | 76 |
| 6.3.1 Lambda 关键字与匿名函数..... | 76 |
| 6.3.2 Map 关键字与批量化函数操作..... | 78 |
| 6.3.3 Reduce 关键字和求和..... | 79 |
| 6.3.4 Filter 关键字和条件选择..... | 79 |
| Chapter 7 对象和类..... | 80 |
| 7.1 对象..... | 80 |
| 7.2 类..... | 81 |
| 7.2.1 那么如何定义类..... | 81 |
| 7.2.2 如何使用类..... | 83 |
| 7.2.3 子类、父类和继承..... | 84 |
| 7.2.4 几个特殊的实例属性和类方法..... | 86 |
| 7.3 模块和包..... | 87 |
| 7.3.1 模块..... | 87 |
| 7.3.2 模块的路径搜索..... | 88 |
| 7.3.3 名称空间..... | 89 |
| 7.3.4 包..... | 90 |
| Chapter 8 文件和目录..... | 92 |
| 8.1 文件读写操作..... | 92 |
| 8.2 目录操作..... | 95 |
| 8.3 文件的压缩和备份..... | 99 |
| 8.4 综合实例..... | 100 |
| Chapter 9 异常处理..... | 103 |
| 9.1 Python 中常见的异常..... | 105 |
| 9.2 自定义异常..... | 107 |
| 9.3 使用异常..... | 108 |
| 9.4 再看异常处理的作用..... | 109 |
| Chapter 10 常用 Python 扩展模块介绍..... | 111 |
| 10.1 NumPy 和高效数据处理..... | 112 |
| 10.1.1 创建数组..... | 113 |
| 10.1.2 数组操作..... | 114 |
| 10.1.3 数组运算..... | 115 |
| 10.1.4 线性代数..... | 116 |
| 10.2 SciPy 与数值计算..... | 116 |
| 10.2.1 插值..... | 117 |

| | |
|---|-----|
| 10.2.2 拟合..... | 117 |
| 10.2.3 极值问题..... | 119 |
| 10.3 Matplotlib 和图表绘制..... | 120 |
| 10.3.1 二维点线数据绘制..... | 120 |
| 10.3.2 辅助散点和线图绘制..... | 121 |
| 10.3.3 简单三维数据可视化..... | 124 |
| 10.4 Xlrd/xlwt 与读写 Excel..... | 126 |
| 10.4.1 读取 Excel 文件..... | 126 |
| 10.4.2 写入 Excel 数据..... | 127 |
| 10.5 Reportlab 和 PDF..... | 127 |
| 10.6 联合使用类库..... | 129 |
| Chapter 11 Python 编程中的效率问题..... | 133 |
| 11.1 时间成本优化..... | 133 |
| 11.1.1 使用内建函数 (built-in Function) | 133 |
| 11.1.2 循环内部的变量创建..... | 135 |
| 11.1.3 循环内部避免不必要的函数调用..... | 136 |
| 11.1.4 使用列表解析..... | 137 |
| 11.1.5 尽量减少 IO 读写..... | 138 |
| 11.1.6 使用优秀的第三方库..... | 139 |
| 11.1.7 其他..... | 139 |
| 11.2 空间成本优化..... | 140 |
| 11.2.1 使用 xrange 处理长序列..... | 140 |
| 11.2.2 注意数据类型的使用..... | 141 |
| 11.2.3 使用 iterator..... | 141 |
| Chapter 12 ABAQUS Script 入门..... | 143 |
| 12.1 GUI 操作 Vs rpy 脚本日志..... | 143 |
| 12.2 对脚本进行简单的二次开发..... | 157 |
| Chapter 13 ABAQUS/PYTHON 基础..... | 159 |
| 13.1 ABAQUS/Python 中的数据类型..... | 159 |
| 13.1.1 符号常值 (SymbolicConstants) | 159 |
| 13.1.2 布尔值 (Booleans) | 159 |
| 13.1.3 特有的模型对象..... | 160 |
| 13.1.4 序列 (Sequences) | 160 |
| 13.1.5 仓库 (Repositories) | 161 |
| 13.2 ABAQUS/Python 的对象的访问和创建..... | 163 |
| 13.2.1 对象的访问..... | 163 |
| 13.2.2 对象数据的修改..... | 165 |
| 13.2.3 对象的创建..... | 165 |
| 13.3 ABAQUS/Python 中的主要对象概况..... | 166 |
| 13.3.1 ABAQUS 中的 session 对象..... | 167 |
| 13.3.2 ABAQUS 中的 mdb 对象..... | 169 |
| 13.3.3 ABAQUS 中的 odb 对象..... | 171 |
| Chapter 14 Session 对象的使用..... | 173 |
| 14.1 Viewport 及其相关对象..... | 174 |

| | |
|---|-----|
| 14.2 Path 对象..... | 180 |
| 14.3 XYData 对象..... | 181 |
| 14.4 XYCurve 和 XYPlot 对象..... | 183 |
| 14.5 writeXYReport 和 writeFieldReport 函数..... | 187 |
| Chapter 15 Mdb 对象的使用..... | 191 |
| 15.1 Model 类与有限元模型的建立..... | 192 |
| 15.1.1 Sketch 和 Part 对象..... | 193 |
| 15.1.2 Material 和 Section 对象..... | 198 |
| 15.1.3 Assembly 对象..... | 200 |
| 15.1.4 Step 对象..... | 202 |
| 15.1.6 Region 对象..... | 203 |
| 15.1.7 Constraint 和 Interaction 对象..... | 204 |
| 15.1.8 Mesh 函数..... | 205 |
| 15.1.9 BoundaryCondition 和 Load 对象..... | 207 |
| 15.2 Job 命令..... | 210 |
| Chapter 16 Odb 对象的使用..... | 212 |
| 16.1 Odb 对象中模型数据..... | 213 |
| 16.1.1 Material 对象..... | 213 |
| 16.1.2 孤立网格数据信息..... | 213 |
| 16.1.3 集合对象..... | 217 |
| 16.2 Odb 对象中结果数据的读取..... | 221 |
| 16.2.1 场变量数据的处理..... | 222 |
| 16.2.2 历史变量数据的处理..... | 226 |
| 16.3 Odb 数据文件的写入..... | 227 |
| 16.3.1 已有模型添加特定数据..... | 227 |
| 16.3.2 生成完整的 Odb 对象..... | 230 |
| Chapter 17 几个常见问题..... | 233 |
| 17.1 几何和网格元素的选择..... | 233 |
| 17.1.1 内置的选择函数..... | 233 |
| 17.1.2 基于特征的筛选方法..... | 235 |
| 17.2 几何元素的特征操作..... | 237 |
| 17.3 具有集合性质的对象..... | 240 |
| 17.4 监测任务运行过程和结果..... | 243 |
| 17.5 交互式输入与 GUI 插件..... | 246 |
| 17.5.1 交互输入..... | 246 |
| 17.5.2 GUI 插件制作..... | 248 |
| Chapter 18 悬帘线问题..... | 260 |
| 18.1 悬链线的方程..... | 260 |
| 18.2 利用 Abaqus 分析悬链线曲线特征..... | 264 |
| 18.2.1 建立分析脚本..... | 264 |
| 18.2.2 确定合适的初始拉伸量..... | 266 |
| 18.2.3 拉伸刚度的影响..... | 267 |
| Chapter 19 扭矩弹簧的刚度..... | 271 |
| 19.1 扭力弹簧的理论分析公式..... | 271 |

| | |
|--|-----|
| 19.2 利用 Abaqus 分析扭力弹簧..... | 273 |
| 19.2.1 梁单元模拟扭力弹簧..... | 273 |
| 19.2.2 实体单元模拟扭力弹簧..... | 279 |
| 19.3 结果对比..... | 281 |
| Chapter 20 圆角处网格研究..... | 283 |
| 20.1 带孔薄板..... | 283 |
| 20.1.1 理论分析..... | 283 |
| 20.1.2 模型计算..... | 284 |
| 20.2 台阶板倒角处的应力..... | 290 |
| 20.2.1 理论分析..... | 290 |
| 20.2.2 有限元模拟..... | 291 |
| Chapter 21 优化问题..... | 297 |
| 21.1 水下圆筒的抗屈曲设计..... | 297 |
| 21.1.1 问题的描述..... | 297 |
| 21.1.2 参数化模型..... | 298 |
| 21.1.3 优化策略..... | 300 |
| 21.1.4 求解与结果..... | 303 |
| 21.2 过盈配合设计..... | 307 |
| 21.2.1 问题描述..... | 307 |
| 21.2.2 参数化模型建模..... | 308 |
| 21.2.3 优化策略与结果..... | 314 |
| 21.3 笔盖的插入力的确定..... | 320 |
| 21.3.1 问题描述..... | 320 |
| 21.3.2 参数化模型建模..... | 321 |
| 21.3.3 优化策略与结果..... | 327 |
| Chapter 22 分析之间的数据传递..... | 331 |
| 22.1 数据传递方法之 InitialState..... | 331 |
| 22.1.1 数据传递前的准备..... | 331 |
| 22.1.2 Standard 数据导入 Explicit 的步骤..... | 332 |
| 22.1.3 数据导入实例：冲压成型分析..... | 333 |
| 22.2 数据传递方法之 Map solution..... | 339 |
| 22.2.1 Map solution 使用格式..... | 339 |
| 22.2.2 数据映射实例：拉拔成型..... | 342 |
| Chapter 23 Python 和子程序..... | 353 |
| 23.1 Fortran 基本用法..... | 353 |
| 23.1.1 Fortran 基本语法..... | 353 |
| 23.1.2 Fortran 程序实例..... | 354 |
| 23.2 Python 处理子程序的一般方法..... | 355 |
| 23.3 实例：Dload 动态轴承载荷..... | 357 |
| 23.3.1 滚子间力的分布..... | 357 |
| 23.3.2 Hertz 接触理论..... | 359 |
| 23.3.3 Dload 子程序模板..... | 360 |
| 23.3.4 Python 建模程序..... | 362 |
| 23.4 实例：基于 Dflux 的焊接热分析..... | 364 |

| | |
|-------------------------|-----|
| 23.4.1 焊接分析热源类型..... | 366 |
| 23.4.2 Dflux 子程序模板..... | 367 |
| 23.4.3 焊接自动化分析脚本..... | 368 |

序一

ABAQUS 是大型通用有限元程序包,拥有强大的非线性分析能力。ABAQUS 的雏形始于 1978 年成立的 HKS 公司(三位创造人 Hibbitt、Karlsson 和 Sorensen 的首写字母),当时的版本只有约 15000 行的 Fortran 代码,可以使用 4 种单元进行分析。ABAQUS 最早期的产品只有隐式求解器 ABAQUS/Standard,显式求解器 ABAQUS/Explicit 于 1991 年推出,实现了与 ABAQUS/Standard 的无缝集成和相互传递数据。2002 年底 HKS 公司改名为 ABAQUS 公司,于 2005 年为 Dassault Systèmes 公司所并购,现为公司力推的 Simulia 产品。

ABAQUS 求解器是采用 input file(扩展名为 inp)驱动的,输入文件基于 keywords。早期的 ABAQUS 没有前后处理器,需要手写输入 input file,也可借助于第三方软件进行,这对复杂几何模型具有很大的局限性。ABAQUS 于 1999 年推出了 ABAQUS/CAE 这一集成分析环境,使得这一局面得到了极大改善。ABAQUS/CAE 基于现代 CAD 理想和 feature 建模概念,采用 Part 生成 Assembly,可以高效实现几何模型构建并生成有限元网格。ABAQUS/CAE 可将生成的模型在后台生成 input file,并提交给 ABAQUS/Standard 或 ABAQUS/Explicit 求解器。ABAQUS/CAE 可以实时监测求解器运行情况,并提供了多种灵活方便的方式对分析结果进行后处理。需要说明的一点是 ABAQUS/CAE 并不是支持 ABAQUS 求解器的所有的 keywords,目前仍有少数功能需要手动修改 input file 来实现,这也说明 ABAQUS/CAE 仍在持续发展中,对求解器 keywords 的支持在不断的增强。

Python 是一个独立的程序语言,其语法简洁而清晰。ABAQUS/CAE 采用 Python 作为脚本语言,和微软的 Excel 等 Office 软件采用 VBA 作为后台脚本是很类似的。当用户打开 ABAQUS/CAE 时,会自动实时产生一个 replay 文件(扩展名为 rpy),里面几乎记录每一步的操作。事实上,当用户保存 ABAQUS/CAE 模型时,都会有个 journal file(扩展名为 jnl)产生,里面是生成 CAE 模型所需的 Python 脚本代码。Journal file 清晰明了,可作为蓝本进行 Python 脚本程序开发。ABAQUS/Python 有着巨大的潜力,使用 Python 脚本不但可以减少很多 ABAQUS/CAE 前后处理的重复性工作,大大提高效益,更重要是的还可以程序化实现原本手动不太可能做的事情。

我和作者苏景鹤素未谋面,于 2012 年相识于虚拟的网络空间,当时我正开发大变形有限元分析方法,基于 Map solution 采用网格重划技术遇到了一些问题,景鹤丰富的 ABAQUS/Python 知识给我提供了不少技术支持,更令我难忘的是他的侠骨热肠。承蒙景鹤不嫌我知识浅陋,邀我为本书作序,在读完初稿后我欣然同意。在目前并不多的系统地介绍 ABAQUS/Python 的教程中,本书以笔记的形式娓娓道来,不尚八股教条,可以看出本书是作者大量使用 Python 脚本开发程

序的经验总结。我相信本书将十分有助于初学者进入 ABAQUS/Python 程序开发的殿堂，对有一定基础的用户也很有参考价值。

田英辉

2015 年 6 月 22 日

西澳大学 天鹅河畔

自序

随着计算机性能的飞速发展，有限元作为工程应用领域的重要方法，在许多行业尤其是制造业中得到了广泛的应用。每一个成功的设计都离不开有限元分析的数据支持，它能确保轻便的设计，它能确保稳定的设计，它能确保高效的设计，它能确保安全的设计。

计算机技术和商业有限元软件的迅速发展，直接推动了有限元分析在设计中的使用。当前工业界常用的 ABAQUS, ANSYS 等软件无论在界面易用性还是求解器效率方面都愈趋成熟，工程师可以迅速地借助软件实现分析任务。随着设计活动对有限元分析的需求越来越大，有限元分析也变得越来越常规，分析任务也越来越繁重。对企业来说，与其增加资源（人力财力）来满足日益增长的分析需求，不如想办法定制自动化分析流程，帮助使用者提高分析的效率。

达索的 ABAQUS 软件为使用者提供了这样的可能——使用 Python 脚本语言实现分析的自动化。利用它使用者可以简化某些重复性操作，定制特定的分析流程以提高工作效率，甚至于进一步和其他软件结合使用拓展 ABAQUS 的使用场景。本书中笔者结合自己几年的 ABAQUS 使用经验，采用语言 API 讲解+实例说明的方法来记录 ABAQUS/Python 使用过程中比较有意思的一些知识点，希望能帮助读者进入 ABAQUS/Python 的领域。

由于能力有限，难免有认识不足和错误的地方，希望能和大家一起讨论进步，如有问题请发送邮件至 su.jinghe@outlook.com。

2015.04.10 于南京

Chapter 1 ABAQUS 二次开发简介

对于 CAE 软件提供商，打造大平台，集成 CAD/CAE/CAM 已经成为一种趋势，而对于特定的 CAE 用户或者企业，制定适合自己的流程化 CAE 软件包越来越重要。对 CAE 提供商来说，要想直接定制适合企业使用的程序包，工业背景的缺乏是横在 CAE 开发者面前的一道坎，另外开发针对特定行业的软件无疑会缩小自己的市场，这一点也会减缓 CAE 提供商开发特定行业软件的步伐。对于 CAE 企业用户虽然掌握了工业应用背景 and 知识，但是花费不菲的人力物力财力去开发一款自己使用的 CAE 分析软件包也是得不偿失的。CAE 软件包本身为用户提供全面易用的二次开发的接口，就是一种折中的解决方案。基于上面这些考虑，当前主流的 CAE 软件包都会为用户提供相应的二次开发的接口程序，方便用户定制化适合自己的工业应用程序，比如 ABAQUS 中使用的 Python/FORTRAN 子程序，ANSYS 中使用的 APDL，HYPERWORKS 中提供的 Tcl/Tk 接口，以及 PATRAN/NASTRAN 使用的 PCL 等等。笔者主要使用 ABAQUS/PATRON，因此本书都是基于对 ABAQUS 二次开发认识的基础上展开的。

1.1 为什么是 Python

初识 Python 的时候，一位朋友说过：ABAQUS 软件是一群绝顶高手开发并维护的，他们选择了 Python 作为 ABAQUS 的接口语言，聪明人的选择也是聪明的。我对这个观点深信不疑，虽然当时对 Python 一无所知。现在有了几年的 Python 开发经验以后，回头想想这个的确是聪明的选择。

Python 是简洁的，Python 是免费的，Python 是跨平台的，Python 是大众的：这些特色直接决定，对 CAE 工程师这样的“业余”程序使用者，Python 是完美的选择。

Python 是目前最为火热的脚本语言之一，而且现在的 Python 已经远远超出脚本语言的范畴，向着大语言发展。表 1-1 是来自权威的 TIOBE 编程语言排行榜¹（2015 年 3 月）的数据，可以看出除了主流的编程语言 C / C++ / C# / VB / JAVA 外，脚本语言中 Python 跻身前十，热度相当于 Perl 和 Ruby 的总和，这数据足以表明当前 Python 的火热。

Python 的流行确保了 Python 讨论和学习的环境比较平坦，易于掌握，加上

¹ TIOBE 排行榜基于互联网上有经验的程序员、课程、和第三方厂商的数量，并使用搜索引擎（如 google、bing、yahoo!、百度）以及 wikipedia、amazon、youtube 统计出的排名数据，主要反映某个编程语言的热门程度，并不说明语言的优劣。

Python 本身所提倡的简洁特性，使得 Python 语言成为 ABAQUS 的“官方”接口语言。

Python 是跨平台的，这意味着使用 Python 编写的二次开发程序可以直接从 WINDOWS 平台移植到 Linux 环境下使用。这一点对于 CAE 工程师非常重要：由于 Linux 系统对硬件资源更好的调用，大部分 CAE 分析用的系统都是 Linux，而日常交流又常常在 WINDOWS 平台上完成，这个时候语言的跨平台特性就十分重要。

Python 的流行也得益于许多优秀的第三方 Python 类库，比如 xlrld/xlwt（用于 Python 操作 MS Excel 文件），matplotlib（绘制仿 Matlab 风格的二维数据图片绘制），pyQt（Qt 类库的 Python 封装，可以生成漂亮的软件界面），NumPy 和 SciPy（处理大规模数据和进行科学计算的 Python 类库，实现类似 Matlab 的基本功能）以及 Reportlab（编辑生成 pdf 文件的 Python 类库）等等。这些优秀的类库大部分都是免费的，而 ABAQUS/CAE 界面就是基于一种免费易用的跨平台的 GUI 类库 Fox 类库的 Python 封装实现的。

表 1-1 TIOBE 排行 TOP 20（2015 年-3 月数据）

| Mar 2015 | Mar 2014 | Change | Programming Language | Ratings | Change |
|----------|----------|--------|----------------------|---------|--------|
| 1 | 1 | | C | 16.642% | -0.89% |
| 2 | 2 | | Java | 15.580% | -0.83% |
| 3 | 3 | | Objective-C | 6.688% | -5.45% |
| 4 | 4 | | C++ | 6.636% | +0.32% |
| 5 | 5 | | C# | 4.923% | -0.65% |
| 6 | 6 | | PHP | 3.997% | +0.30% |
| 7 | 9 | ▲ | JavaScript | 3.629% | +1.73% |
| 8 | 8 | | Python | 2.614% | +0.59% |
| 9 | 10 | ▲ | Visual Basic .NET | 2.326% | +0.46% |
| 10 | - | ▲ | Visual Basic | 1.949% | +1.95% |
| 11 | 12 | ▲ | F# | 1.510% | +0.29% |
| 12 | 13 | ▲ | Perl | 1.332% | +0.18% |
| 13 | 15 | ▲ | Delphi/Object Pascal | 1.154% | +0.27% |
| 14 | 11 | ▼ | Transact-SQL | 1.149% | -0.33% |
| 15 | 21 | ▲ | Pascal | 1.092% | +0.41% |
| 16 | 31 | ▲ | ABAP | 1.080% | +0.70% |
| 17 | 19 | ▲ | PL/SQL | 1.032% | +0.32% |
| 18 | 14 | ▼ | Ruby | 1.030% | +0.06% |
| 19 | 20 | ▲ | MATLAB | 0.998% | +0.31% |
| 20 | 45 | ▲ | R | 0.951% | +0.72% |

1.2 Python, FORTRAN 与 ABAQUS

目前 ABAQUS 的二次开发有两种，求解器层次的 FORTRAN 和前后处理层次的 Python。在说明 ABAQUS/Python 能做什么之前，我们必须先弄清楚 ABAQUS 中两种二次开发的区别。我们需要先了解一下 ABAQUS 软件包的架构。

ABAQUS 软件包包括两大部分：用来进行前后处理的 ABAQUS/CAE（包括 ABAQUS/GUI 和 ABAQUS/Kernel）以及用来对有限元模型进行求解计算的求解器（包括 ABAQUS/standard, ABAQUS/Explicit, ABAQUS/CFD 或者 ABAQUS/Aqua），如图 1-1 所示。Abaqus/CAE 运行后会产生三个进程：abq6141.exe, ABQcaeG.exe（Abaqus/CAE GUI）和 ABQcaeK.exe（Abaqus/CAE Kernel）。

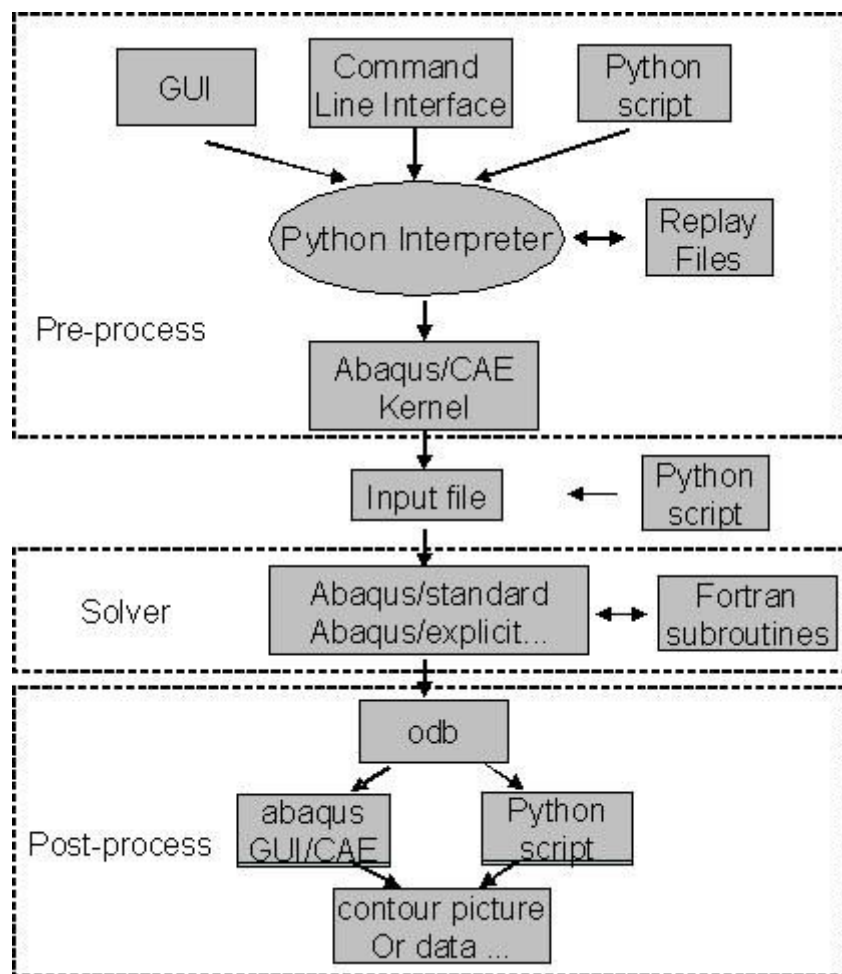


图 1-1 ABAQUS 软件环境结构

GUI 或者负责收集建模参数交给 Kernel 建模并最终形成 INP 文件，或者打开现有的 ODB 文件，提取数据并显示云图，这一过程基本上都是 Python 语言完成的。达索公司为 ABAQUS/CAE 提供了丰富的接口，如对模型操作的 MDB 相关接口，对结果数据 ODB 操作的接口，以及常用的 CAE 相关的 session 操作的

接口。ABAQUS/Python 二次开发主要就是基于这一部分进行的，目的是或者快速自动建模并形成 INP，或者处理现有的 ODB 结果并提取所需数据。

无论通过 CAE 或者手动编辑最终都要形成 INP 文件，它记录建立的网格模型，载荷以及边界条件和分析类型等，它是 Abaqus standard/ Abaqus explicit...等求解模块唯一可识别的输入类型。这些计算模块就可以利用 INP 文件所描述的网格模型和边界条件求解问题的解并记录到结果文件中。像 ABAQUS/Standard 等求解器都是使用 FORTRAN 语言（或者 C/C++）实现的，这些程序可以满足大部分求解分析需求。但对一些比较复杂的问题，ABAQUS 为用户提供了对基本模块功能进行扩展的接口，如我们熟知的用于描述复杂材料本构的材料模型子程序如 UMAT/VUMAT 等；用于描述复杂加载方式的 DFLUX 或者 DLOAD 等；用于描述变化边界条件的 UMOTIONS/UMESHMOTION 等；以及用于描述用户定义特性单元的 UEL。这个层次的二次开发都必须使用 FORTRAN 来完成。

当然因为 INP 文件有自己固有的格式，这个也就方便使用者绕过 Abaqus/CAE 直接利用 Python 程序生成 INP 文件，然后利用对应的 Solver 来求解。这个方法对于一些特定的问题十分有效，也可以认为是二次开发的范畴。

本书所介绍就是基于 Python 实现的前后处理层次上的二次开发。或者是编写程序段完成某一特定的分析优化计算；或者是利用 Python 对大量计算结果进行后处理提取想要的结果；或者是编写更契合用户使用习惯的 GUI 界面，简化使用者的操作流程。

1.3 基于 Python 二次开发

从上面的介绍可以看出 ABAQUS/Python 二次开发就是要代替 ABAQUS/CAE 实现前后处理工作，因此基本上能在 ABAQUS/CAE 中实现的前后处理操作都可以通过 Python 程序来实现。只不过有些问题使用 ABAQUS/CAE 更为高效，比如从第三方 CAD 软件中导入的几何模型常常需要几何清理，这个工作就不适合使用 ABAQUS/Python 来实现。下面从笔者自己的认识列出几个判断问题是否适合进行二次开发的标准。

1) 当前所面对的分析问题在日常工作中是否经常遇到。这个实际上是做应用开发首要考虑的问题，确保问题值得花费额外的时间去做二次开发。一般二次开发都是在了解了分析过程的基础上进行，如果仅仅是偶发问题，使用 ABAQUS/CAE 建立模型分析一次解决问题即可，花时间做二次开发就显得有些多此一举。对那些日常工作中常常要碰到的问题，二次开发就可以帮我们节省大量的时间，避免重复劳动。举一个简单的情况，如果模型中存在非常多的 part 部件，要赋予相同的截面属性，如果在 ABAQUS/CAE 中操作，会非常繁琐，我们可以考虑编写一个对模型中所有 part 赋予相同截面属性的功能插件，后面面对类似的情况，只需要点击一下按钮就可以。

2) 一些涉及参数优化或者参数灵敏性分析的问题，常常需要对同一模型进行

多次分析，这类问题就非常适合做二次开发。

- 3) 有时候借助程序进行二次开发是必须的，例如在建立一些随机模型的时候，利用程序可以得到一些伪随机数来完成建模的过程。
- 4) 面对没有特定规律的问题，通常不适合二次开发。编写程序实际上是对有特定规律问题的统一化解决方案，如果问题本身就是不确定的，比如几何清理²的问题，那么二次开发也就无从谈起。

举个例子，对于焊接温度分布的模拟，ABAQUS 中可以使用 DFLUX 子程序来完成。这一系列模拟流程非常适合二次开发：分析流程比较固定，不同的计算只需要使用对应的几何参数即可完成。因此可以编写一个完成焊接分析流程的小插件。该插件使用如图 1-2 所示的 GUI 界面收集输入参数，点击 OK 后程序自动建立如图 1-3 所示的模型以及模拟的过程中需要的 DFLUX 子程序，等程序计算完成后可以得到如图 1-4 所示的温度分布云图。

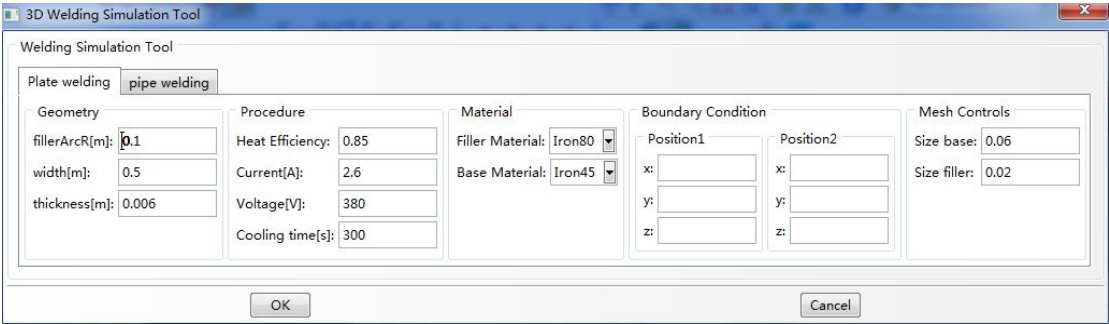


图 1-2 焊接插件界面



图 1-3 两种工况：平板焊和圆管焊

² 实际上几何清理在一定程度上也可以利用二次开发来解决，比如可以通过判断面的大小来删除一些小面，通过判断四边形面中两对边长度的比值大小来判断细长面的存在，但是由于工程问题对应的模型都比较复杂，因此这种二次开发往往不能彻底解决问题。

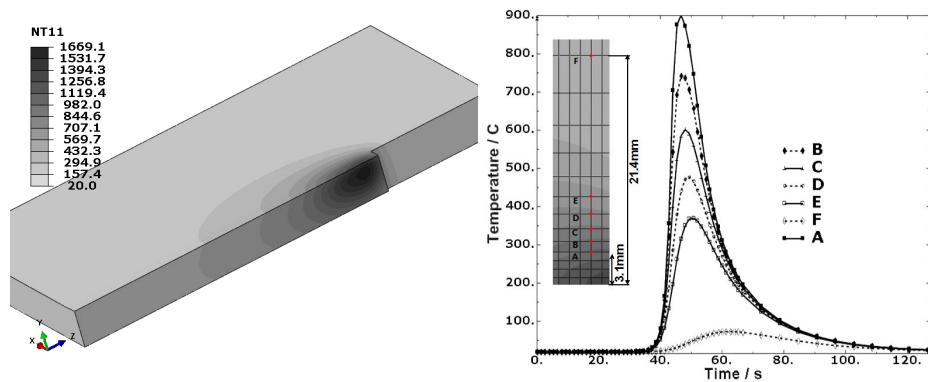


图 1-4 温度分布云图与时间历程

ABAQUS/Python 的二次开发可以在工程应用以及科学研究中发挥重要作用，笔者尝试记录自己在使用 ABAQUS/Python 中的心得，希望能帮助大家更快的学习和使用 ABAQUS/Python 来高效完成自己手头的工作。

Chapter 2 Python 能力确认

为了能让读者有针对性地阅读本书，笔者这里给出三个例子，让大家可以通过读程序来确认自己的 Python 功底，从而确认从那一章节阅读此书比较合适。

pyTest1.py 设计了一个包含基本运算和流程控制的函数，对 C 语言比较熟悉的读者应该可以准确的推断出程序的输出。这个就是本书 4-6 章节所要讲的内容，如果这个程序可以看懂，那么就可以跳过 4-6 章直接从第 7 章开始看起。

pyTest2.py 设计了一个文件读写和类创建的程序段，对 C++ 和 Java 这类面向对象语言比较熟悉的同学应该可以推断出程序的输出。这个就是本书 7-9 章节所要讲的内容，如果这个程序可以看懂，那么就可以跳过 7-9 章直接从第 10 章开始看起。

pyTest3.py 给出的是一个简单的工程实例二次开发代码。这个就是本书 12-16 章节所要讲的内容，如果这个程序可以看懂，那么就可以跳过 12-16 章直接从第 17 章开始看看 Python 语言在二次开发过程的一些应用案例。

2.1 测试程序

测试程序一（pyTest1.py）：

```
#
1  # -*- coding: utf-8 -*-
2
3  def is_prime(num):
4      # Initial to presume it's a prime
5      rt = True
6      # To test the numbers if it can be divided exactly by a smaller number.
7      for i in range(2, num):
8          if num % i == 0:
9              rt = False
10             break
11     return rt
12
13 a = []
14 b = {}
15 for i in range(1,10):
16     if not(is_prime(i)):
17         a.append(i)
18     else:
19         b["Prime Number"+str(i)*2] = i
20 for key,value in b.iteritems():
```

```
21     print "%s = %s" % (key, value)
22     print "the composite numbers is %s" % a
#=====
```

测试程序二（pyTest2.py）:

```
#=====
1  # -*- coding: utf-8 -*-
2  import sys
3  import math
4  import re
5  import os
6  import csv
7
8  class myMaterial:
9      """this is a class to define the optimization material"""
10
11     infor='Author JingheSu, Email:: su.jinghe@outlook.com'
12     cluster='optimization'
13
14     def __init__(self,density,elastic,plastic,expansion,specificheat,
15                 conductivity):
16         self.elastic=elastic
17         self.plastic=plastic
18         self.expansion=expansion
19         self.specificheat=specificheat
20         self.conductivity=conductivity
21         self.density=density
22
23     def setElastic (self,elastic):
24         self.elastic=elastic
25
26     def setPlastic (self,plastic):
27         self.plastic=plastic
28
29     def setExpansion (self,expansion):
30         self.expansion=expansion
31
32     def setSpecificheat (self,specificheat):
33         self.specificheat=specificheat
34
35     def setConductivity (self,conductivity):
36         self.conductivity=conductivity
37
38     def setDensity (self,density):
```

```

39         self.density=density
40
41     def printInfor (self):
42         print self.infor
43         print 'this material belongs to set: '+self.cluster
44
45     #=====
46
47     def updateMaterial():
48         """this function is used to update the material in current
49         directory. with this function you can insert your material
50         just by updating your material data using a csv file"""
51         csvlist=[]
52         cdir=os.getcwd()
53         clist=os.listdir(cdir)
54         matDict={}
55         for item in clist:
56             filedir=os.path.join(cdir, item)
57             if os.path.isfile(filedir) and str(item).endswith('mat.csv'):
58                 csvlist.append(filedir)
59         if len(csvlist)==0:
60             print 'you got no material to use, material collection programm' \
61             'exit!'
62             return 0
63         else:
64             for item in csvlist:
65                 csvreader=csv.reader(file(item))
66                 tempDensity=[]
67                 tempElastic=[]
68                 tempPlastic=[]
69                 tempExpansion=[]
70                 tempSpecifichheat=[]
71                 tempConductivity=[]
72                 for row in csvreader:
73                     num=len(row)
74                     templist=[]
75                     typename=row[0].strip()
76                     for data in row[1:]:
77                         posteddata=data.strip()
78                         if posteddata!="":
79                             templist.append(float(posteddata))
80                     if typename=='density':
81                         tempDensity.append(templist)
82                     elif typename=='elastic':

```

```

83         tempElastic.append(templist)
84     elif typename=='plastic':
85         tempPlastic.append(templist)
86     elif typename=='expansion':
87         tempExpansion.append(templist)
88     elif typename=='specificheat':
89         tempSpecificheat.append(templist)
90     elif typename=='conductivity':
91         tempConductivity.append(templist)
92     tempMat=myMaterial(tempDensity,tempElastic,tempPlastic,
93         tempExpansion,tempSpecificheat,tempConductivity)
94     (filepath, filename) = os.path.split(item)
95     matDict[filename[:-4]]=tempMat
96
97     return matDict
98     #=====
99
100 if __name__ == '__main__':
101     result=updateMaterial()
102     print len(result)
103     print result.values()[0].density
104     print result.values()[0].specificheat

```

测试程序三（pyTest3.py）:

```

1  # -*- coding: utf-8 -*-
2
3  from abaqus import *
4  from abaqusConstants import *
5  from viewerModules import *
6  import regionToolset
7  import mesh
8
9  length = 1000 #mm
10 Cload = 40 #N
11 radius = 3.0 #mm
12 Mdb()
13 #: Create a mdb: model-1.
14 s = mdb.models['Model-1'].ConstrainedSketch(name='beam',
15     sheetSize=200.0)
16 g, v, d, c = s.geometry, s.vertices, s.dimensions, s.constraints
17 s.Line(point1=(0.0, 0.0), point2=(length, 0.0))
18 p = mdb.models['Model-1'].Part(name='beam', dimensionality=THREE_D,
19     type=DEFORMABLE_BODY)
20 p = mdb.models['Model-1'].parts['beam']

```

```

21 p.BaseWire(sketch=s)
22 del mdb.models['Model-1'].sketches['beam']
23
24 mdb.models['Model-1'].Material(name='steel')
25 mdb.models['Model-1'].materials['steel'].Elastic(table=((210000.0, 0.28), ))
26 mdb.models['Model-1'].materials['steel'].Density(table=((7.8e-09, ), ))
27 mdb.models['Model-1'].CircularProfile(name='Profile-1', r=radius)
28 mdb.models['Model-1'].BeamSection(name='Section-beam', profile='Profile-1',
29     integration=DURING_ANALYSIS, poissonRatio=0.28, material='steel',
30     temperatureVar=LINEAR)
31 p = mdb.models['Model-1'].parts['beam']
32 e = p.edges
33 region = regionToolset.Region(edges=e)
34 p.SectionAssignment(region=region, sectionName='Section-beam', offset=0.0,
35     offsetType=MIDDLE_SURFACE, offsetField="",
36     thicknessAssignment=FROM_SECTION)
37 e = p.edges
38 region=regionToolset.Region(edges=e)
39 p.assignBeamSectionOrientation(region=region, method=N1_COSINES, n1=(0.0, 0.0,
40     -1.0))
41
42 a = mdb.models['Model-1'].rootAssembly
43 a.DatumCsysByDefault(CARTESIAN)
44 p = mdb.models['Model-1'].parts['beam']
45 a.Instance(name='beam-1', part=p, dependent=ON)
46
47 mdb.models['Model-1'].StaticStep(name='Step-load', previous='Initial',
48     nlgeom=ON)
49
50 a = mdb.models['Model-1'].rootAssembly
51 v1 = a.instances['beam-1'].vertices
52 verts1 = v1.findAt(((0,0,0),),)
53 a.Set(vertices=verts1, name='Set-fix')
54 verts1 = v1.findAt(((length,0,0),),)
55 a.Set(vertices=verts1, name='Set-force')
56 region = a.sets['Set-fix']
57 mdb.models['Model-1'].DisplacementBC(name='BC-fix', createStepName='Step-load',
58     region=region, u1=0.0, u2=0.0, u3=0.0, ur1=0.0, ur2=0.0, ur3=0.0,
59     amplitude=UNSET, fixed=OFF, distributionType=UNIFORM, fieldName="",
60     localCsys=None)
61 region = a.sets['Set-force']
62 mdb.models['Model-1'].ConcentratedForce(name='Load-load',
63     createStepName='Step-load', region=region, cf2=-1.0*Cload,
64     distributionType=UNIFORM, field="", localCsys=None)

```

```

65
66 p = mdb.models['Model-1'].parts['beam']
67 e = p.edges
68 p.seedEdgeBySize(edges=e, size=length/100.0, deviationFactor=0.1,
69     constraint=FINER)
70 elemType1 = mesh.ElemType(elemCode=B32, elemLibrary=STANDARD)
71 pickedRegions =(e, )
72 p.setElementType(regions=pickedRegions, elemTypes=(elemType1, ))
73 p.generateMesh()
74 a = mdb.models['Model-1'].rootAssembly
75 a.regenerate()
76 mdb.Job(name='beam-load', model='Model-1', description="", type=ANALYSIS,
77     atTime=None, waitMinutes=0, waitHours=0, queue=None, memory=50,
78     memoryUnits=PERCENTAGE, getMemoryFromAnalysis=True,
79     explicitPrecision=SINGLE, nodalOutputPrecision=SINGLE, echoPrint=OFF,
80     modelPrint=OFF, contactPrint=OFF, historyPrint=OFF, userSubroutine="",
81     scratch="", multiprocessingMode=DEFAULT, numCpus=1)
82 mdb.jobs['beam-load'].submit(consistencyChecking=OFF)
83
84 mdb.jobs['beam-load'].waitForCompletion()
85 odbpath = os.path.join(os.getcwd(), "beam-load.odb")
86 pngPath = os.path.join(os.getcwd(), "deformation")
87 oo = session.openOdb(name=odbpath)
88 vp = session.Viewport(name='myView')
89 vp.makeCurrent()
90 vp.maximize()
91 vp.setValues(displayedObject=oo)
92 vp.odbDisplay.setPrimaryVariable(variableLabel='U',
93     outputPosition=NODAL, refinement=(INVARIANT, 'Magnitude'), )
94 vp.odbDisplay.display.setValues(plotState=CONTOURS_ON_DEF)
95 session.graphicsOptions.setValues(backgroundStyle=SOLID,
96     backgroundColor='#FFFFFF')
97 vp.viewportAnnotationOptions.setValues(legendDecimalPlaces=2,
98     legendNumberFormat=SCIENTIFIC, triad=OFF, legendBox=OFF)
99 vp.viewportAnnotationOptions.setValues(
100     legendFont='-*-verdana-medium-r-normal-*-180-*-p-*-*)
101 vp.viewportAnnotationOptions.setValues(
102     legendFont='-*-verdana-bold-r-normal-*-180-*-p-*-*)
103 vp.odbDisplay.contourOptions.setValues(spectrum='Black to white')
104 vp.viewportAnnotationOptions.setValues(
105     titleFont='-*-verdana-medium-r-normal-*-140-*-p-*-*)
106 vp.viewportAnnotationOptions.setValues(
107     stateFont='-*-verdana-medium-r-normal-*-140-*-p-*-*)
108 vp.view.fitView()

```

```

109 session.printOptions.setValues(vpDecorations=OFF, reduceColors=False)
110 session.printToFile(fileName=pngPath, format=TIFF, canvasObjects=(
111     vp, ))
112

```

2.2 程序运行结果

测试程序一（pyTest1.py）

该程序定义了一个函数 `is_prime`，其用来判断一个正整数是否为素数：对 1、2、3……，9 这九个数字根据其是否为素数将其分别存储在一个列表和字典类型的变量中，最后以一定的格式打印显示。程序的执行结果如下：

```

----- Python -----
Primer Number11 = 1
Primer Number22 = 2
Primer Number77 = 7
Primer Number55 = 5
Primer Number33 = 3
the composite numbers is [4, 6, 8, 9]

```

测试程序二（pyTest2.py）

该程序定义了一个存储材料物性参数的类 `myMaterial`，以及一个可以对当前目录下文件进行扫描并将所有以 `mat.csv` 结尾的文件中的信息提取出来的函数 `updateMaterial`。该函数可以生成一个由文件名为键名，`myMaterial` 类的对象为键值的字典。程序的执行结果如下：

```

----- Python -----
3
[[2700.0]]
[[900.0, 20.0], [921.0, 100.0], [1005.0, 200.0], [1047.0, 300.0], [1089.0, 400.0], [1129.0, 2000.0]]

```

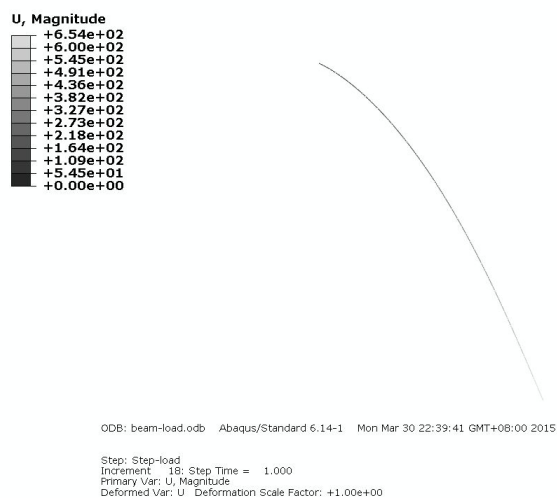


图 2-1 pyTest3 生成的图片

测试程序三（pyTest3.py）

该程序是一个 ABAQUS 二次开发的完整实例。程序可以自动完成对一个悬臂梁端部受集中载荷分析，并将变形情况以图片的形式输出到当前工作目录下（如图 2-1 所示），命名为 deformation.png。

Chapter 3 脚本的运行与开发环境

在开始二次开发学习之前我们需要先了解 ABAQUS 中二次开发的环境。图 3-1 给出的界面就是 ABAQUS/CAE 软件启动后的界面，菜单栏，模型树，主视窗，快捷按钮，信息提示区等等。这些对于常常使用 ABAQUS 软件的读者应该是再熟悉不过了。下面的笔者将逐一介绍与二次开发相关的一些区域和功能。

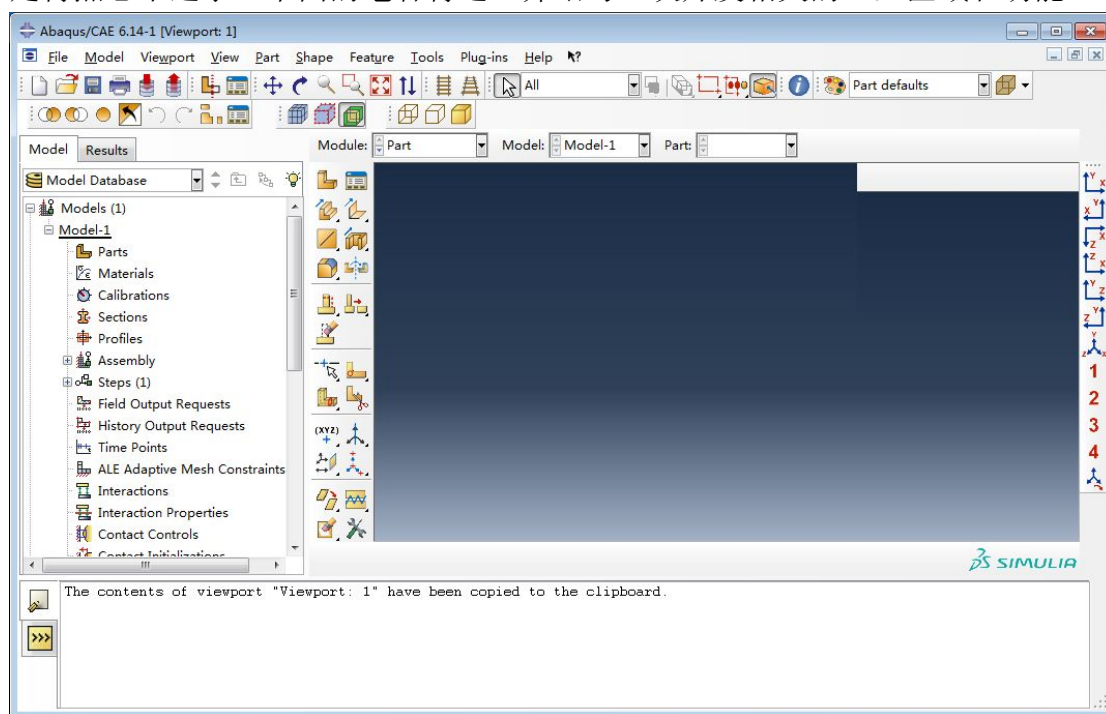


图 3-1 ABAQUS/CAE 的主界面

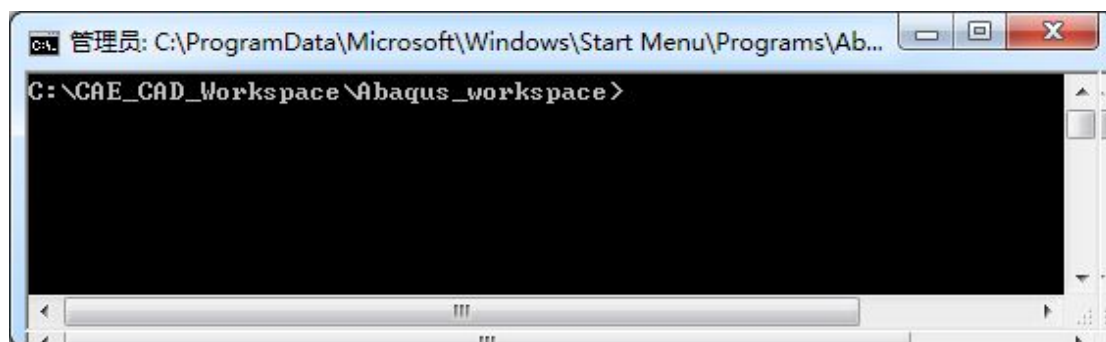


图 3-2 ABAQUS COMMAND 界面

3.1 ABAQUS 中脚本的运行

在 ABAQUS 中脚本运行可以使用下面任一种方法实现：

3.1.1. 命令区 KCLI (kernel command line interface)

脚本可以在 KCLI 中逐行运行 (按钮, 位于主界面左下角)。点击按钮, 进入命令接口区, 可以看到 ABAQUS 中输入提示符为>>>, 表示程序在等待用户的输入, 如图 3-3 所示。



图 3-3 KCLI 命令接口区

输入 2+3, 回车就可以得到结果 5。实际上这一过程就完成了最简单的单行脚本执行过程。



图 3-4 KCLI 中命令的执行

对于编写好的程序脚本也可以使用 execfile 函数运行, 使用时候需要确保给出正确的脚本文件路径, 比如对于如下的脚本

C:\CAE_CAD_Workspace\Abaqus_workspace\PythonBook\Chapter3\print.py

就可以在 KCLI 中使用如下的命令运行:

```
execfile('C:\CAE_CAD_Workspace\Abaqus_workspace\PythonBook\Chapter3\print.py')
```

运行的效果和直接输入 2+3 相同。

3.1.2 CAE-Run Script

从 File->Run Script...选择脚本文件 (.py) 运行, 使用记事本编辑生成文本文件 print.py, 并保存。文本内容如下:

```
# -*- coding: utf-8 -*-  
"""this is a example for chapter 1: print.py"""  
print 2+3
```

从 File->Run Script...选择框中选择 print.py 文件, 可以在信息提示框中看到脚本运行的结果: 5。



图 3-5 print.py 在 CAE 中的执行结果

3.1.3 ABAQUS Command

在 ABAQUS Command 命令行下使用 `abaqus cae noGUI=script.py`（前后处理都可以使用）或者使用 `abaqus Python script.py`（仅仅适用于进行后处理的脚本程序执行）命令来执行脚本。

确保将刚才的 `print.py` 文件复制到当前工作目录下，分别使用 `abaqus Python print.py` 和 `abaqus cae noGUI=print.py` 来执行，结果如图 3-6 所示。

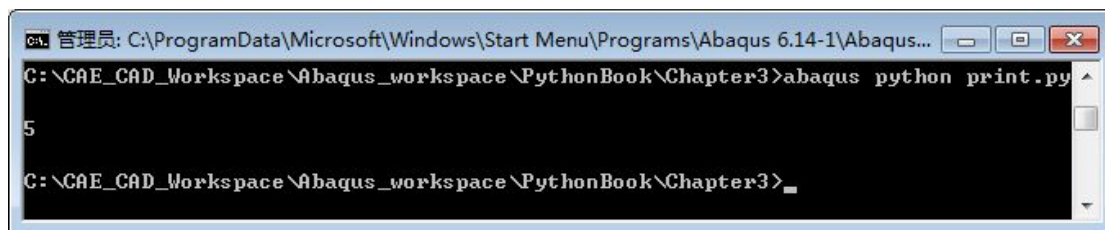


图 3-6 `print.py` 在 Command 中的执行结果

◆ Tips:

1. 如果要运行的文件不在当前的目录下，可以先使用 `dos` 命令 `cd` 或者是 `cd..` 命令切换目录到目标文件所在目录再执行上述命令。

`cd C:\CAE_CAD_Workspace\Abaqus_workspace\PythonBook\Chapter3`

2. 细心的朋友可以发现了 `abaqus Python print.py` 和 `abaqus cae noGUI=print.py` 的执行结果不同，一个成功打印出了结果 5，另一个没有。这是由于两种执行机制不同导致，`abaqus cae noGUI=print.py` 执行相当于使用 `File->Run Script` 方法来执行程序，结果被输出到了信息提示区，不打开 CAE 界面时候就看不到所打印的结果。

3. 另外特别值得注意的是 `abaqus Python print.py` 执行过程是没有检查 licensing 的。这个方便我们在没有 licensing 的情况下对结果文件 ODB 进行数据出，提取自己关心的数据。

3.1.4 ABAQUS PDE

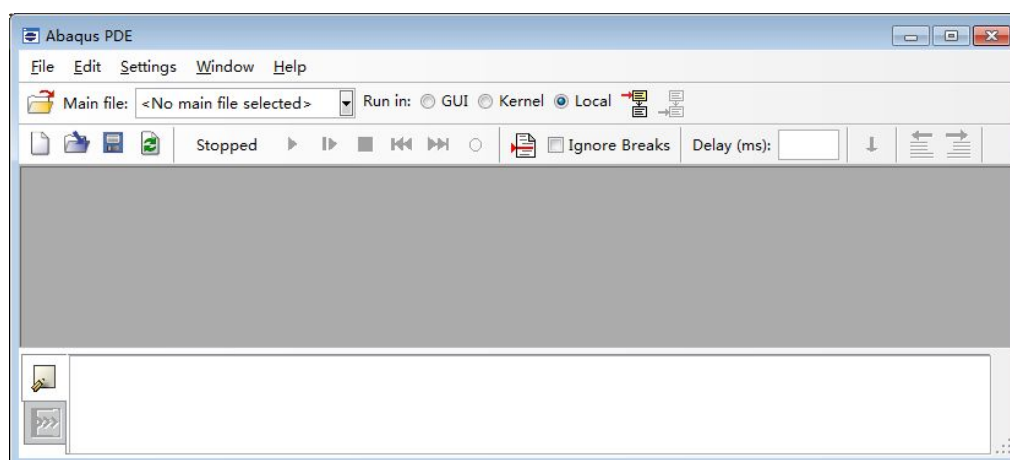


图 3-7 Abaqus\Python 集成开发环境

使用 ABAQUS 提供的 Python development environment (PDE)来运行脚本。

在 Abaqus Command 中键入 `abaqus pde`，进入集成开发环境，如图 3-7。在 PDE 中打开 `print.py` 文件，点击 `play` 按钮（如图 3-8 所示）就可以执行 `print.py` 脚本，程序运行的结果信息会被打印到 ABAQUS Command 空间中。

每一种运行方式都有自己的优势和侧重点，比如 KCLI 下的运行比较方便，尤其是当程序段比较短，或者想测试某段小程序的结果时可以在 KCLI 中直接运行查看效果；ABAQUS PDE 主要用来对程序进行调试，同查看变量在执行过程中的变化情况来调试程序达到特定的目的；一般具有某一特定功能的程序写好后，最终的测试都是 ABAQUS Command 或者 `File->Run Script` 下的运行，在这一过程之前需要使用文本编辑器来编辑程序，下面我们就介绍几款广受好评的文本编辑器以及在其基础上搭建的 Python 开发环境。

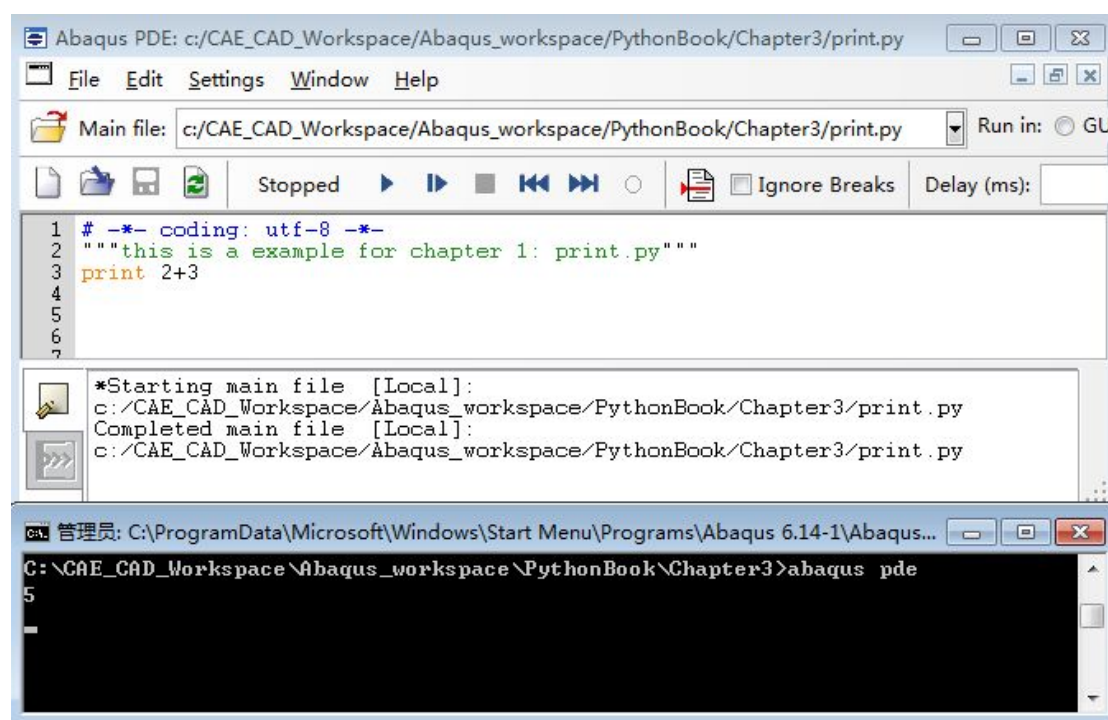


图 3-8 PDE 中脚本的执行

3.2 选择自己的 Python 开发环境

当前用于 Python 开发的文本编辑器非常多，笔者仅仅就自己使用过的几款做简单的介绍（主要包括 ABAQUS PDE，IDLE，Notepad++和 EditPlus），每个人可以根据自己的喜好选用自己的 ABAQUS Python 开发环境。

3.2.1 ABAQUS PDE

由于 ABAQUS 中的 Python 并不是原生的 Python 版本，而是经过达索公司定制的。进行二次开发时常常要引用达索自定义的一些类和模块，因此 ABAQUS PDE 的先天优势就是可以直接调试程序和运行程序，效果和 CAE 中运行相同，

不用担心类库无法引用的问题，这一点是其他第三方开发环境所不具备的。ABAQUS PDE 的另一好处就是不需要额外安装，已经包含于 ABAQUS 的安装程序中。此外，语法高亮和自动缩进也是 ABAQUS PDE 的优势，尤其是自动缩进特性在 Python 语言中非常重要，因为 Python 是靠缩进而不是括号来识别语法的。

ABAQUS PDE 有三种启动方式：

1. 当前有 ABAQUS/CAE 运行时，可以在菜单栏选择 File->Abaqus PDE。
2. 在 Abaqus Command 中键入 `abacus cae -pde`。
3. 在 Abaqus Command 中键入 `abacus pde`，

前两种方法打开的 ABAQUS/PDE 和对应的 ABAQUS/CAE 位于相同的 session，可以方便的将 ABAQUS/CAE 的各步骤操作命令录制到 guiLog 文件中，加快开发的速度；后一种方法单独打开 ABAQUS/PDE，并不与 ABAQUS/CAE 工作区间相关联，但是由于不打开 ABAQUS/CAE，因而不占用许可证，并且启动速度较快。

◆ Tips:

1. guiLog 文件会记录使用者在 CAE 软件中的每一步 GUI 操作，而 rpy 文件中记录对应操作的内核函数命令，对于开发者 rpy 文件更有用。
2. 第三种方法更适合对现有的 Python 程序文件进行编辑，后续也可以进一步以 Kernel 的方式调试。

进入 PDE 集成开发环境，如图 3-7。各部分功能的介绍如图 3-9 所示，

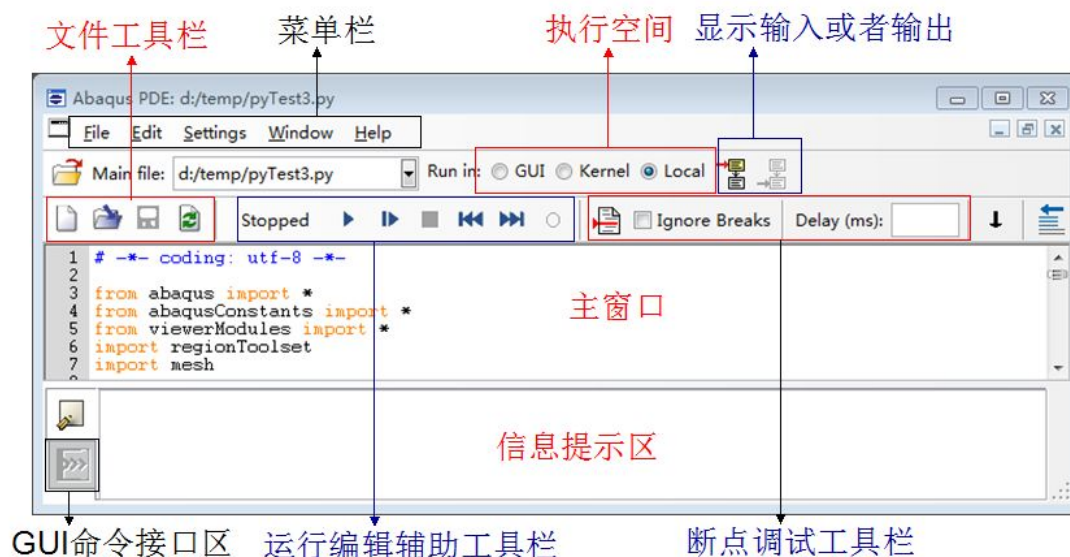




图 3-9 PDE 中各部分的功能

说到 ABAQUS/PDE 和其他开发环境的比较，不能不说其最大的优点：可以直接对 Python 程序实行断点调试或者直接运行。这里我们简单介绍一下 ABAQUS/PDE 中的断点调试方法。断点调试主要是用来确定程序在执行过程中是不是完全按照初始设计的线路运行的，通过检查执行过程中变量值得变化来确

定出问题的程序段。ABAQUS/PDE 中的断点调试需要遵循如下的流程，

1. 在出问题的程序位置前设置几个断点 Break point: 使用调试工具栏的按钮, 设置过断点的语句行会被高亮显示并使用星号标出, 如图 3-10 中 11 和 20 行所示。

2. 根据出问题语句确定几个观察变量 Watch variable: 选择 Window->Debug Windows->Watch List, 在出现的 Watch List 框中右键单击并选择 add watch line, 在 Variable 对应的下面输入需要观察的变量名, 如图 3-10 中的 g, radius 和 p。

3. 使用调试工具栏按钮执行断点调试, 或者选择 Window->Debugger 开始断点调试。

4. 使用调试按钮, Step、Next 等进行调试, 直到程序运行到需要的位置停止: 比如图 10 所示的运行状态, 高亮显示的语句(第 18 行)就是下一步要执行的语句, 在当前状态下变量 g 已经被赋予了对象(ConstrainedSketchGeometry), 变量 radius 也被赋予了值 3.0, 而变量 p 还并不存在。

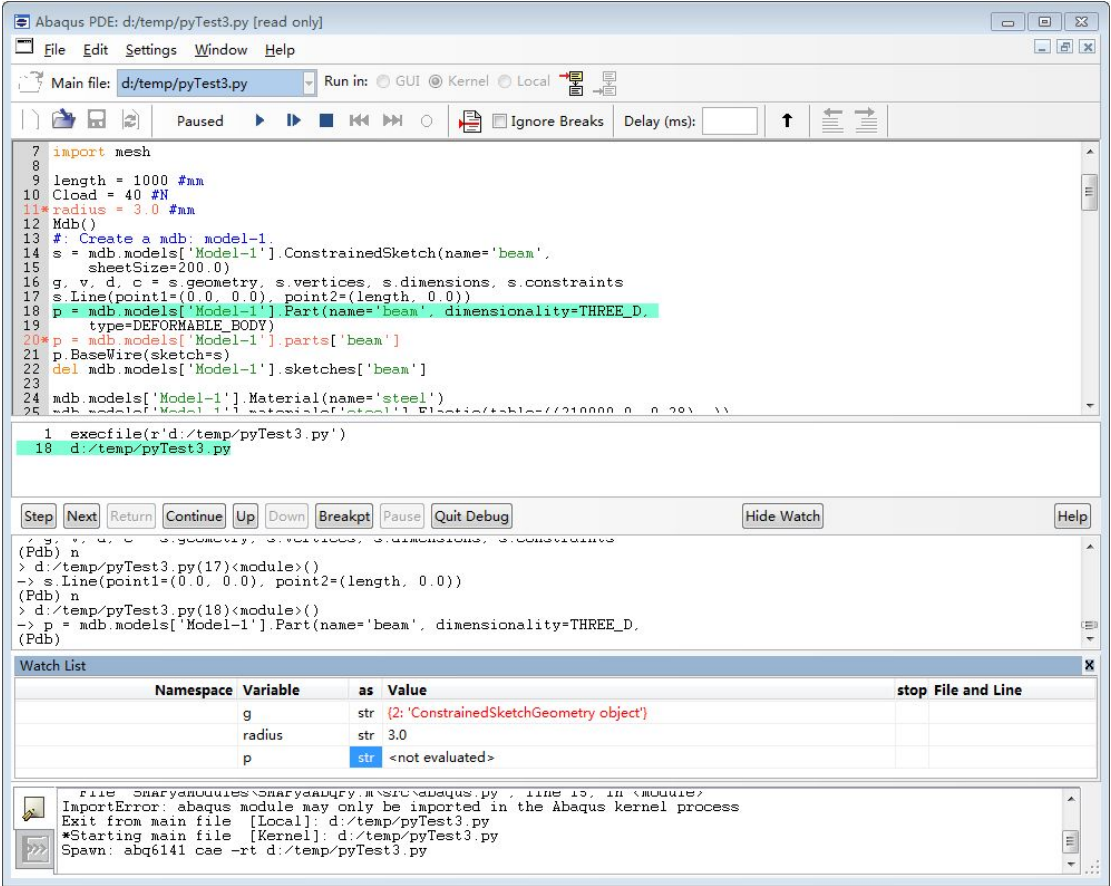


图 3-10 PDE 中程序的断点调试

3.2.2 IDLE

对于以前已经对 Python 有所了解的同学, 对 IDLE 肯定不陌生, 这是标准 Python 安装包自带的编辑器。IDLE 简单而不奢华, 可以满足 Python 程序员的大

部分要求：简单的自带补全功能，简单的智能提示功能。简单也就意味着功能并不完美，各项表现都很平均，没有代码折叠功能，使用古老的 Tkinter GUI 库，运行比较慢，而且从现在的观点看，它的界面风格并不美观。

因为 ABAQUS 6.14 中内置的 Python 版本为 Python2.7.3，因而笔者下面介绍的独立 Python 运行解释器都选择 Python2.7。在 www.Python.org 或者 www.activestate.com 网站下载 Python2.7，双击安装，运行后界面如图 3-11。

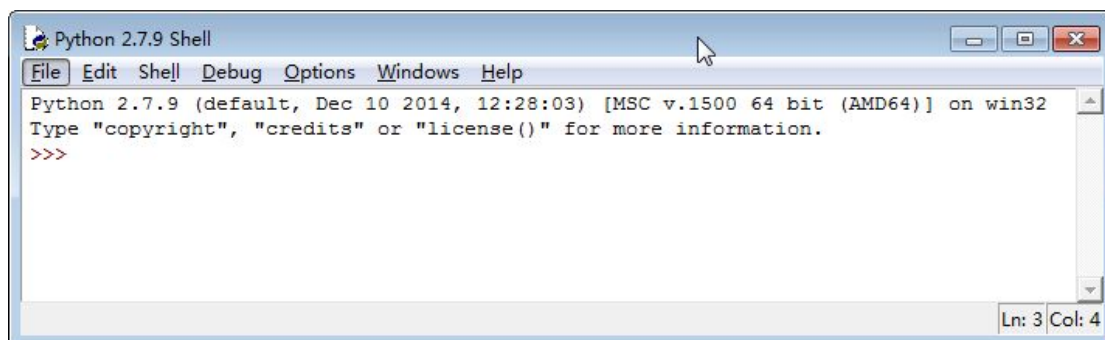


图 3-11 IDLE 的运行界面

Python Shell 是自带的交互式运行界面，用户可以在其中输入简单的程序语句（输入提示符为>>>），回车后直接执行语句。

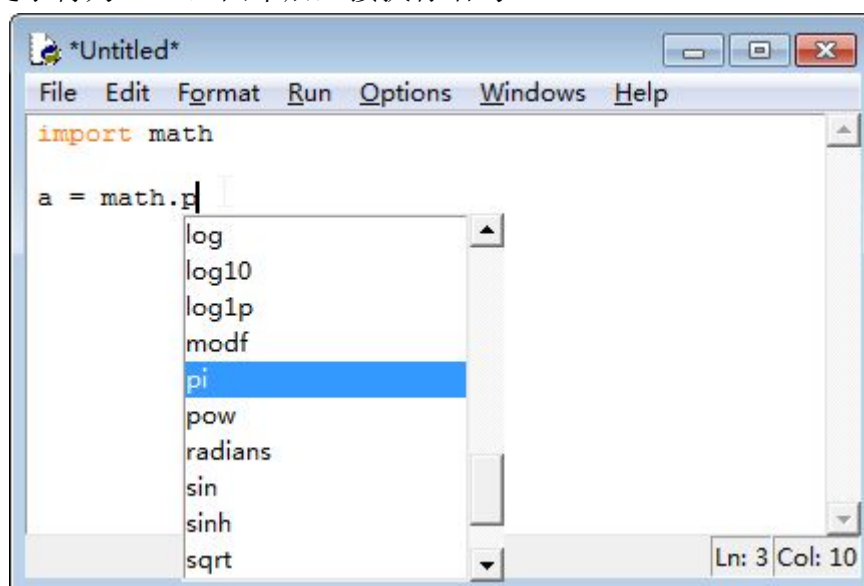


图 3-11 IDLE 的自动补全

实现比较复杂功能的程序段可以单独在 IDLE 编辑器中编辑保存为 py 文件（Python Shell 中菜单栏选择 File->New window 即可打开编辑器）。IDLE 编辑器有自动缩进功能，完善的语法高亮，简单的自动补全和智能提示功能，编辑好的程序可以使用快捷键 F5 解释运行，并且可以进行简单的断点调试，对使用者非常方便。

从功能上说，Python Shell 和 ABAQUS 中的 KCLI 类似；IDLE 中的编辑器和 ABAQUS/PDE 类似。但是由于独立 Python 运行解释器中不包含 ABAQUS 开

发需要的类库，因而任何包含 ABAQUS 类库的程序并不能在 IDLE 中调试，这也是所有第三方编辑环境的共同弱点。

虽然 ABAQUS/PDE 和 IDLE 中都可以打开任意格式的文本文件，但是当文本文件比较大时速度慢而且表现不稳定。ABAQUS 涉及的文本文件常常动辄几十 Mb，这个时候就必须使用专业的第三方文本编辑器来提高效率。下面介绍两种第三方文本编辑器，它们在操作大文本文件时候速度和效率都要远远高于上面提到的两个工具，并且通过简单的配置它们也可以直接运行 ABAQUS Python 程序，基本上可以替代上面的两个原始工具。

3.2.3 Notepad++

Windows 的使用者一定对 Notepad（记事本）不会陌生，对 Notepad++ 可能就不是那么熟悉了。Notepad++ 可以看成是一个加强版的记事本。它是由台湾 IT 工程师侯今吾开发的开源软件，由于其强大的功能广受好评，在许多开发语言的编辑器它都占据了一席之地。Notepad++ 的界面如图 3-13 所示。

然而 Notepad++ 只是一个文本编辑器，需要一定的配置才可以进行程序的编译运行，下面介绍如何配置 Notepad++ 使得其可以实现 ABAQUS Python 程序的编译执行。因为 Notepad++ 和我们后续要介绍的 EditPlus 一样自身不包含 Python 解释器的，为使程序可以顺利解释运行，需要使用 ABAQUS 内置的 Python 解释器。

在 Notepad++ 官方网站(<http://Notepad-plus-plus.org/>)下载最新的 Notepad++，解压安装（也可以选择免安装版本，直接解压使用）。

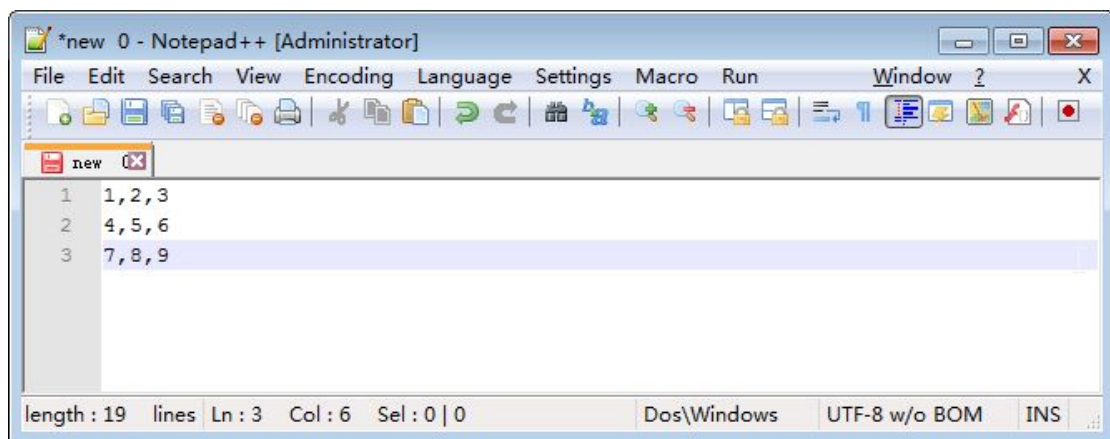


图 3-13 Notepad++ 运行界面

我们选择从菜单栏选择 Language->Python 即可设定当前高亮 Python 语法关键字。

Python 中缩进也是语法的一部分，虽然可以使用 Tab 字符进行各级的缩进，但是 PEP 编码规范中推荐使用 4 个空格符作为各级缩进的标识。为了方便，我们可以在 Notepad++ 中定义 Tab 字符为 4 个空格符，Setting->Preferences->Tab

Setting->Replace by space，具体设置如图 3-14，此时按下 Tab 键就相当于输入了 4 个空格符。

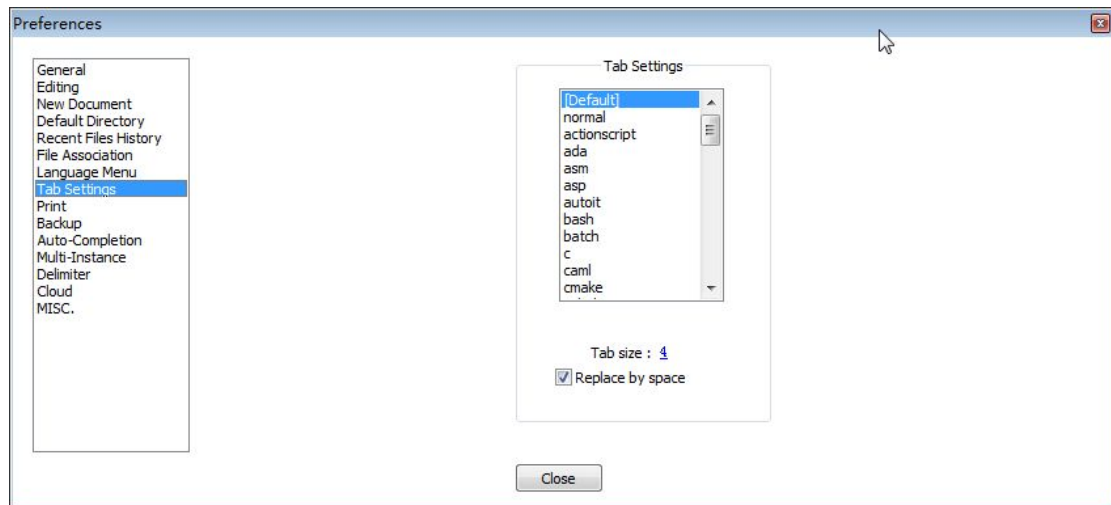


图 3-14 Tab 字符的设置

下面我们设置 Notepad++ 中 Python 程序直接编译执行功能。打开 Notepad++，按 F5，打开“运行”对话框，在文本框输入 `cmd /k cd "$(CURRENT_DIRECTORY)" & C:\Python27\python.exe "$(FULL_CURRENT_PATH)" & ECHO. & PAUSE & EXIT`，然后点击“save”，填写个名字，比如“Run with Python”（如果需要可以配置下面的快捷键），点 OK 后即可在运行菜单上点“Run with Python”来执行当前的 Python 程序。

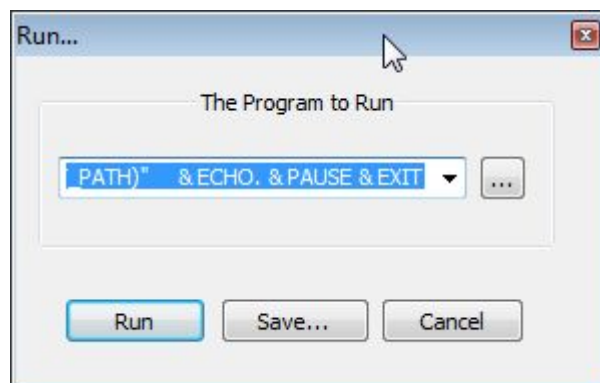


图 3-15 运行对话框

同样的如果我们想要 Notepad++ 直接调用 ABAQUS 自带的 Python 解释器来执行程序，可以按照上面同样的方法操作。

如果需要使用 ABAQUS Python 命令执行后处理脚本，我们需要将“运行”命令换成 `cmd /k cd "$(CURRENT_DIRECTORY)" & ABAQUS Python "$(FULL_CURRENT_PATH)" & ECHO. & PAUSE & EXIT`，并保存为另一个名字，比如“Run with AbaqusPost”。

当然 ABAQUS 中脚本运行还有其他两种运行方式，ABAQUS CAE

noGUI=xxxx.py 以及 ABAQUS CAE script=xxxx.py。同样的方法，将 Notepad++ 中的运行命令改为 `cmd /k cd "$(CURRENT_DIRECTORY)" & ABAQUS CAE "noGUI=$(FULL_CURRENT_PATH)" & ECHO. & PAUSE & EXIT`（保存为 Run with AbaqusnoGUI）或者命令 `cmd /k cd "$(CURRENT_DIRECTORY)" & ABAQUS CAE "script=$(FULL_CURRENT_PATH)" & ECHO. & PAUSE & EXIT`（保存为 Run with AbaqusGUI）。

建好上面几个快捷键以后我们就可以在 Notepad++ 中直接编译执行所写的 Python 程序了。至于上面几种运行方式的区别我们后面会逐一解释。

3.2.4 EditPlus

EditPlus³是一款由韩国 Sangil Kim（ES-Computing）出品的小巧但是功能强大的可处理文本编辑器，经过用户自己的配置可以成为一个简单但是编辑、编译、运行和调试功能俱佳的 IDE（集成开发环境）。EditPlus 界面如图 3-16。

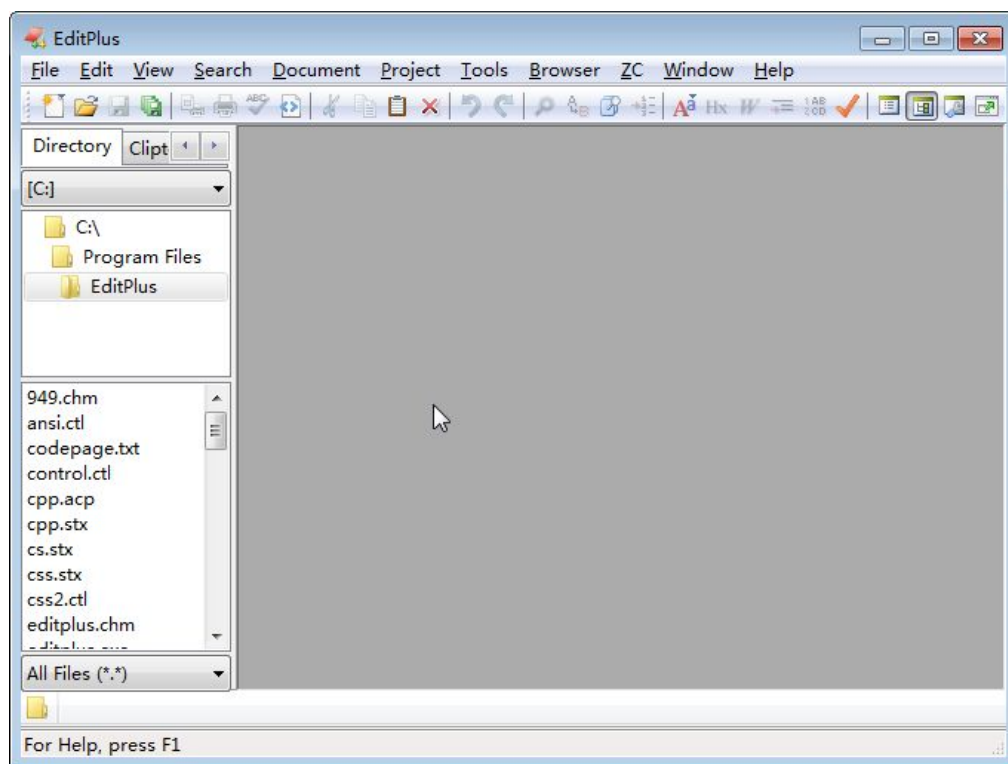


图 3-16 EditPlus 用户界面

EditPlus 可以自动识别并高亮 Python 源程序中的关键字。为了能够更有效的编写 Python 程序，在下载安装了 EditPlus 程序后可以对其进行一些配置，使其能具有部分智能补全功能。从 EditPlus 的官方网站上下载 Python.acp，打开 EditPlus，Tools -> preferences -> File|Settings&syntax，点击 Python 项，并如图 3-16 加载对应的文件，然后点击 Ok 保存。此时打开编写 Python 源代码时候输入 for 以后程序会生成书写格式。

³ EditPlus 官方提供 30 天试用，超过 30 天后需要购买许可才能继续使用，目前最新版本是 3.8

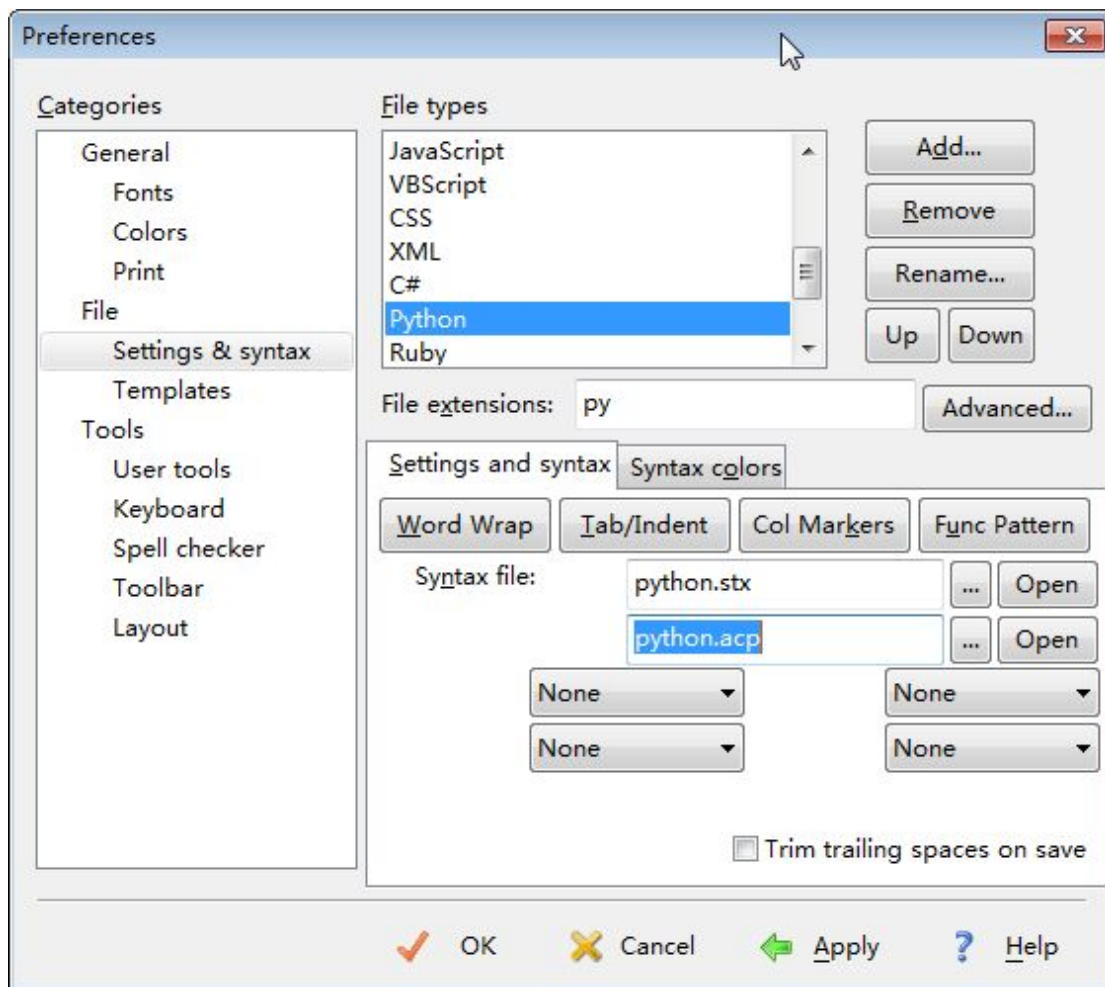


图 3-17 语法高亮和自动补全的配置

EditPlus 中文本文件默认的保存扩展名为.txt，每次要保存自己修改为.py。我们可以选择从特定的模板文件来建立文件，不仅仅省去了修改扩展名的问题，还可以节省一些不必要的工作量。建立如下的文件并保存为 **template.py**，这个模板文件中默认导入几个常用的模块，省去建立文件后重复键入的麻烦。

```
# -*- coding: utf-8 -*-
import sys, math, re

if __name__ == '__main__':
    pass
```

打开 EditPlus，Tools -> preferences -> File| templates，点击 Add 选择刚才保存的 **template.py** 文件，确定后在 Menu Text 中输入语言名：Python，如图 3-18 所示，点 OK 保存。此时就可以从菜单栏的 File->New|Python 新创建一个已经有部分内容的 py 文件。

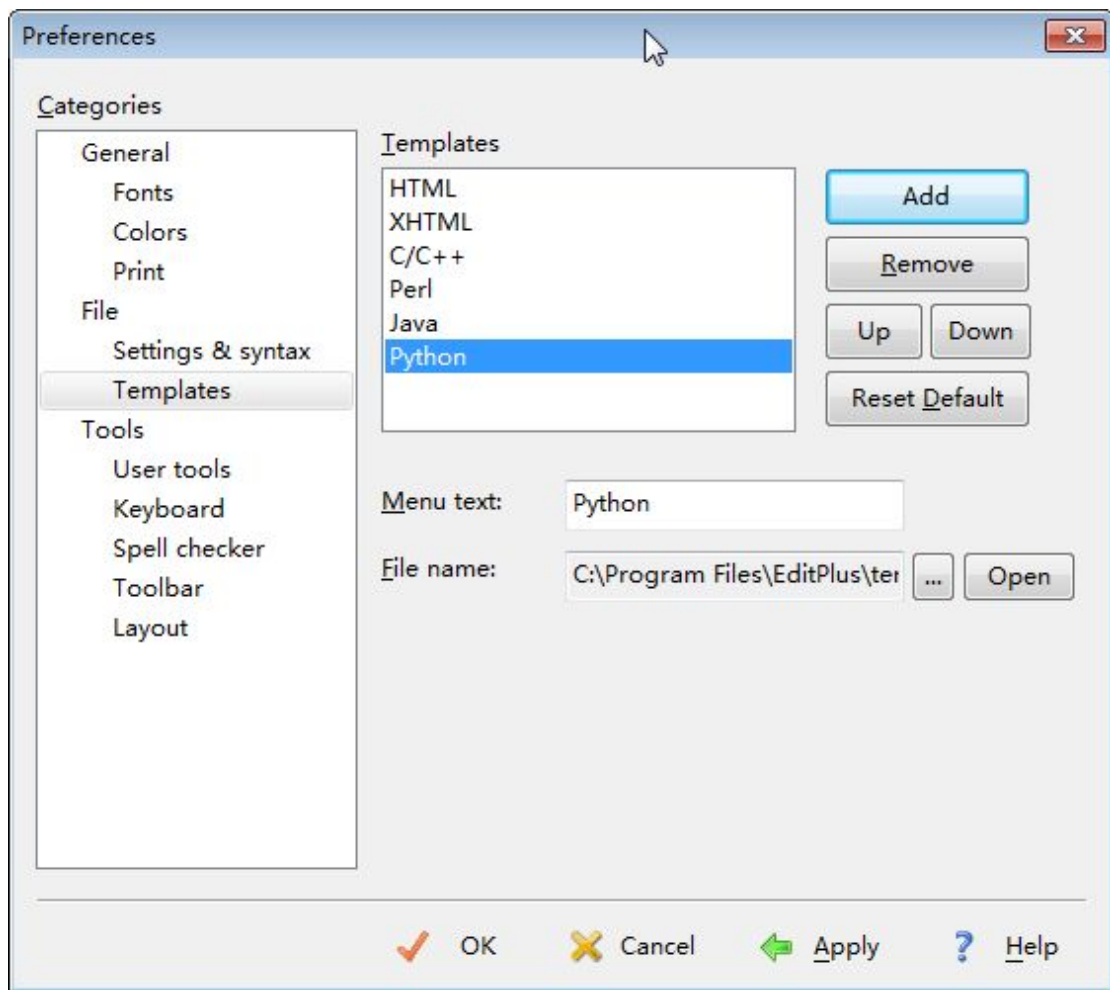


图 3-18 模板文件的使用设置

下面我们需要设置 EditPlus 使得可以从 EditPlus 中直接执行 Python 程序，返回结果。打开 EditPlus，Tools -> preferences -> Tools|User tools，在 group and tool items 下面选择一个还没有使用的组名，点击 Group Name，输入 Python，然后点击 OK。

完成后我们就可以在 Python 工具中添加自己的命令工具了。点击 Add Tool，按照图 3-19 所示设置，完成后点击 OK。

需要注意的是，Command 后面的路径是笔者电脑上 Python 的安装路径，读者需要根据自己的情况适当修改；Action 中选择 Capture output 或者 Output Windows。

完成上述设置以后，我们就可以直接在 EditPlus 中执行 Python 程序，程序执行的结果会打印在输出窗口中如图 3-20 所示的黑色框区（如果没有看到输出窗口，可以在 View->Toolbars/views 中勾选）。

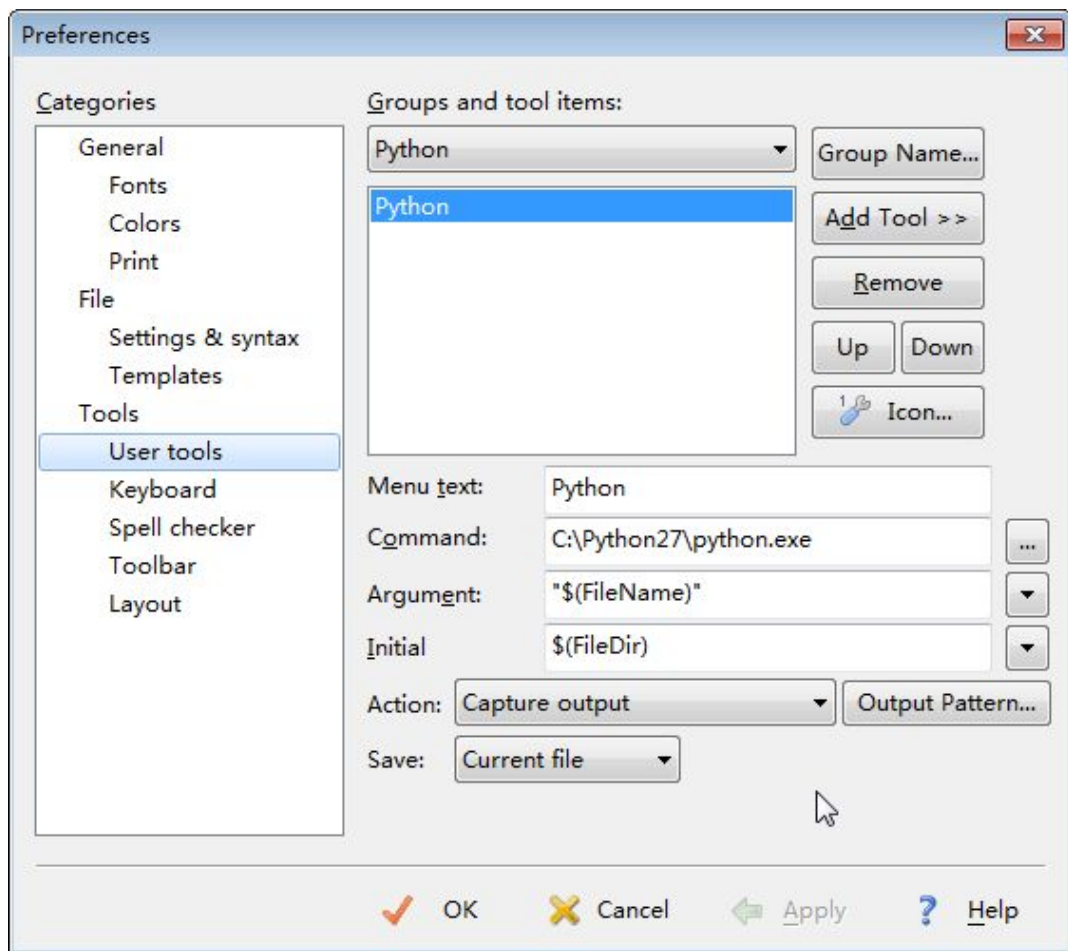


图 3-19 Python 快捷执行设置一

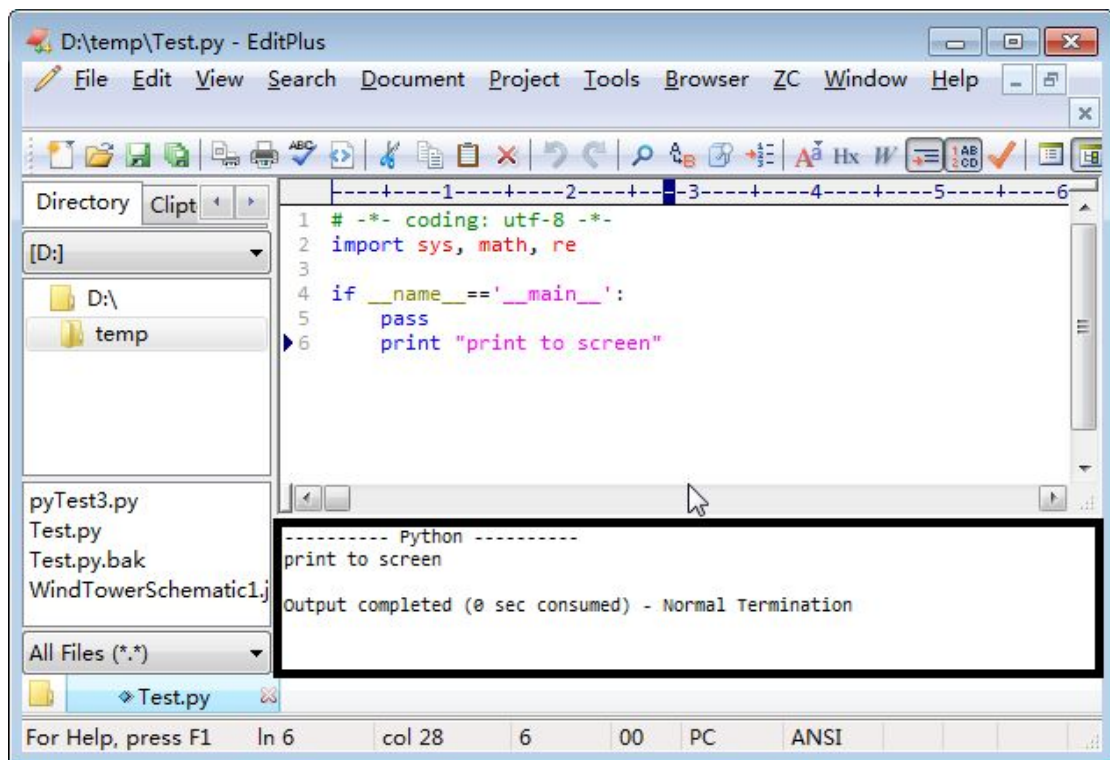


图 3-20 EditPlus 中执行结果示意

同样的如果希望在 EditPlus 中调用 ABAQUS Python 或者 ABAQUS CAE 命令可以按照图 3-21 和图 3-22 所示的进行设置。

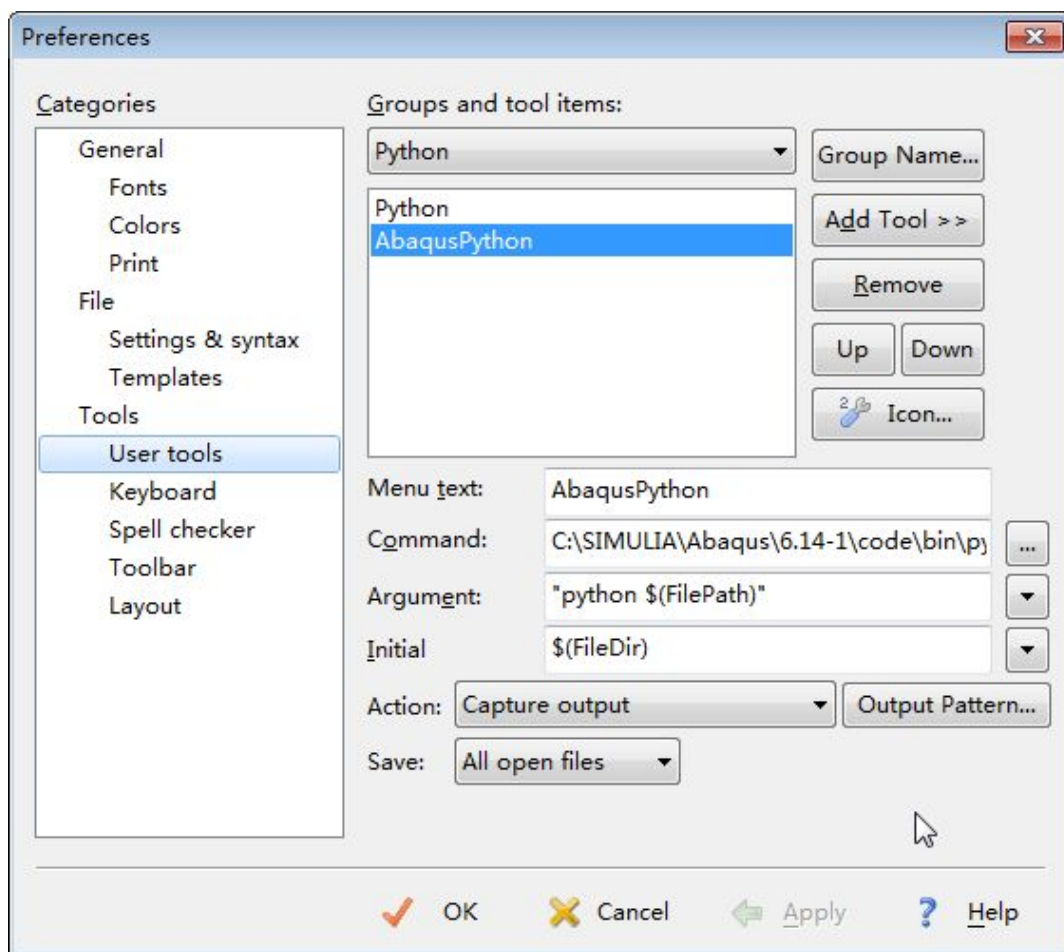


图 3-21 EditPlus 中设置 abaqus Python 命令

注意 AbaqusPython 命令 Command 框中需要填写自己电脑上安装的 ABAQUS 6.14 中的 Python 编译器路径。C:\SIMULIA\Abaqus\6.14-1\code\bin\python.exe 是默认安装的位置，读者需要根据自己的安装版本和安装目录填写合适的路径。

对于 ABAQUS CAE noGUI=xxx.py 命令，我们需要创建一个批处理文件。

建立文本文件，内容如下：

```
@echo off
```

```
"C:\SIMULIA\Abaqus\6.14-1\code\bin\abq6141.exe" cae %*
```

保存为 AbaqusCAEnoGUI.bat，并在 Command 框中选择刚才保存的文件 AbaqusCAE.bat 即可，其他如图 3-22 设置。

如果想直接打开 CAE 界面运行脚本，只需要将图 3-22 中的 Argument 改为："script=\$(FileName)"即可。

◆ Tips:

为了更好的使用 EditPlus 编写 Python 程序，可以设置使用 4 个空格代替 Tab 键，具体操作，打开 EditPlus，Document->Tab Indent，如图 3-23 所示设置。

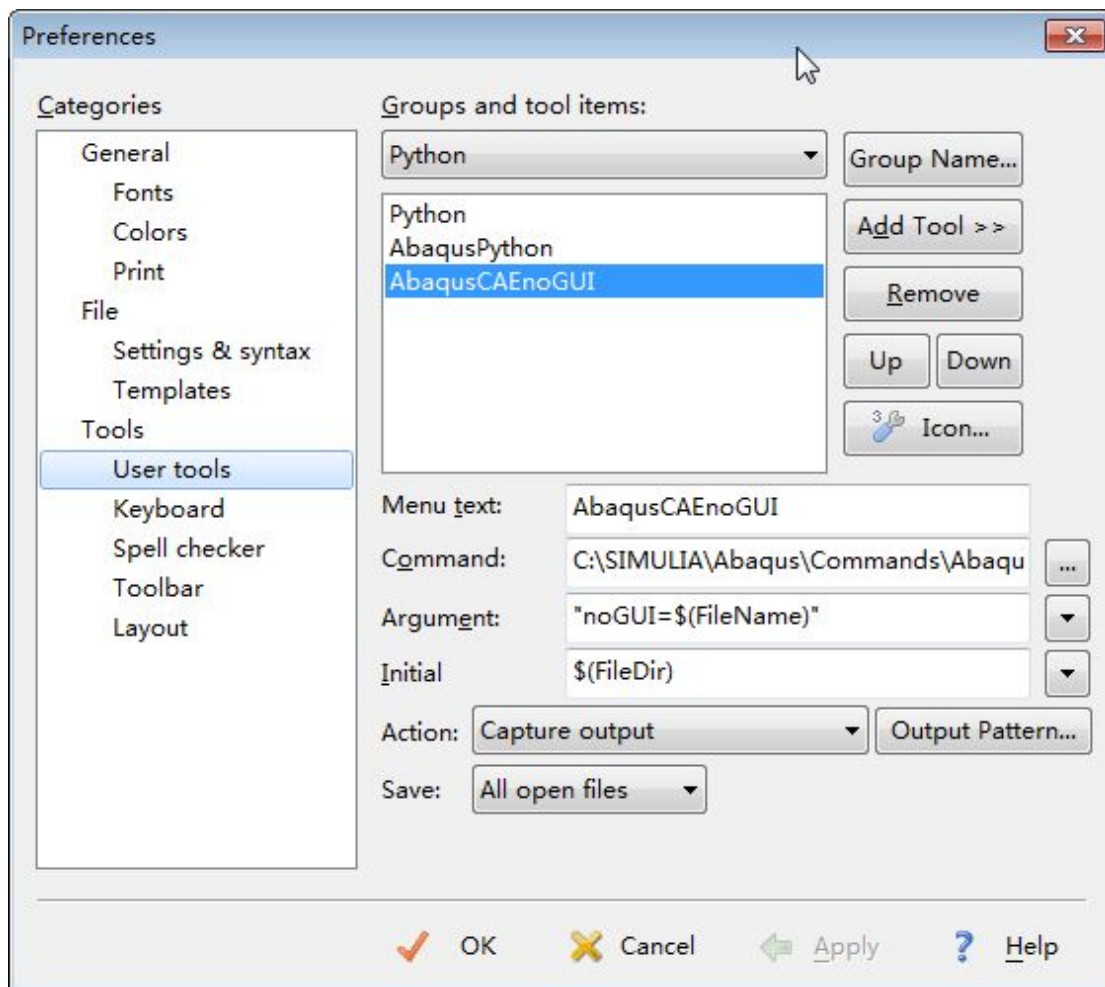


图 3-22 EditPlus 中设置 ABAQUS CAE noGUI 命令

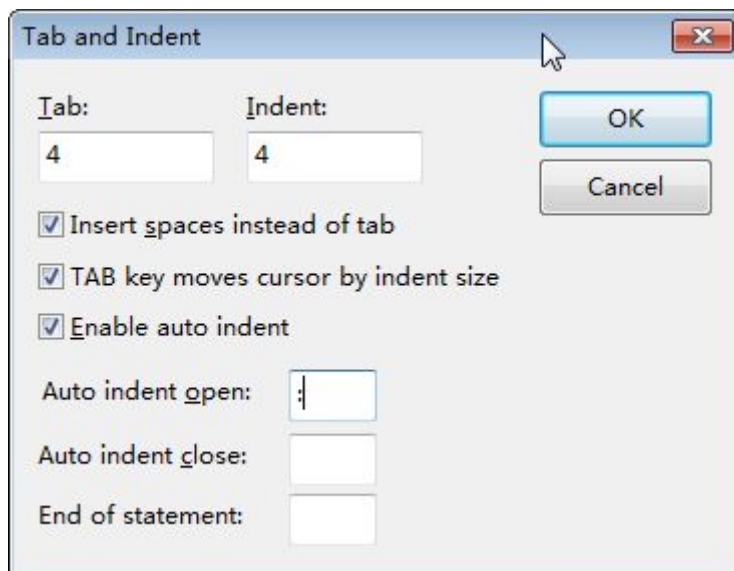


图 3-23 EditPlus 中 Tab|Indent 的设置

3.2.5 选择合适的编程环境

上面分别简单介绍了 4 种 Python 开发工具的使用方法，下面总结一下各自的

优缺点，读者可以根据自己的需求来选择。

表 3-1 几种编辑器的比较

| | 语法高亮 | 自动补全 | 脚本断点 调试 | 大文件打 开速度 | 免费 | 易用程度 |
|------------|------|------|------------|-------------|----|------|
| Abaqus PDE | 有 | 无 | 可以 | 较慢 | 不 | 较难 |
| IDLE | 有 | 有 | 不可以 | 慢 | 免费 | 简单 |
| Notepad++ | 有 | 无 | 不可以 | 快 | 免费 | 简单 |
| EditPlus | 有 | 部分 | 不可以 | 快 | 不 | 简单 |