基于QT实现的记事本应用程序实验报告

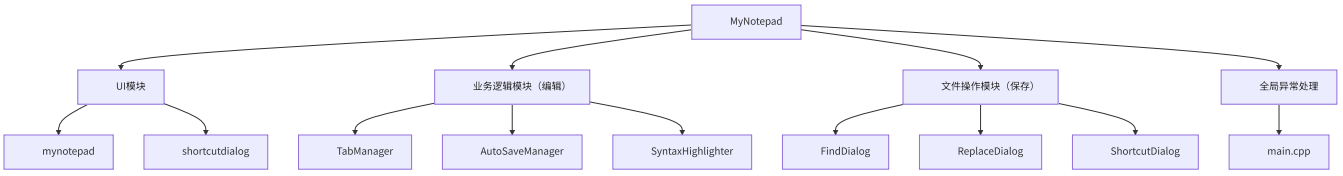
1. 项目概述

本项目是一个基于Qt框架开发的现代化记事本应用程序，具有丰富的文本编辑功能和用户友好的界面。该应用程序不仅支持基本的文本编辑功能，还包含了**语法高亮**、**多标签页管理**、**自动保存**、**快捷键自定义**等高级特性。

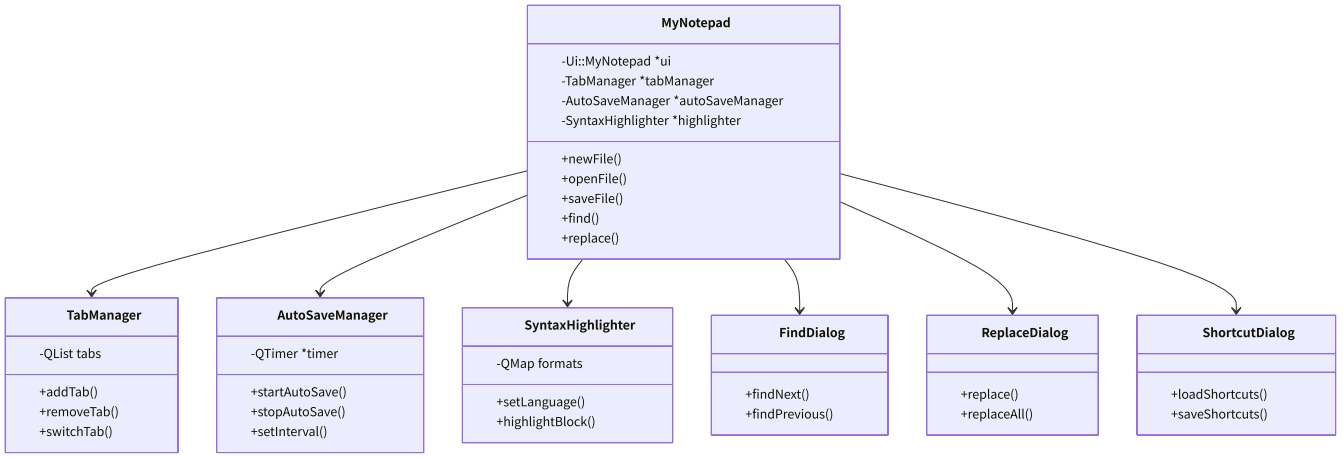
2. 核心功能模块

2.1 项目架构图

项目结构



类关系图



2.2 主窗口模块 (MyNotepad)

主窗口类是整个应用程序的核心，负责协调各个功能模块的工作。主要功能包括：

- 文件操作：新建、打开、保存、另存为等
- 编辑操作：撤销、重做、剪切、复制、粘贴等
- 格式设置：字体、颜色等
- 查找和替换功能
- 多标签页管理
- 自动保存功能

- 快捷键自定义

2.3 标签页管理模块 (TabManager)

负责管理多个文本编辑标签页，主要功能：

- 创建新标签页
- 切换标签页
- 关闭标签页
- 保存标签页状态

2.4 语法高亮模块 (SyntaxHighlighter)

支持多种编程语言的语法高亮（目前支持`Java`、`C++`和`Python`）。

2.5 自动保存模块 (AutoSaveManager)

提供自动保存功能。

- 可配置自动保存间隔
- 自动保存文件到临时目录
- 程序启动时恢复未保存的内容

3. 技术实现要点

3.1 用户界面设计

```
void MyNotepad::createMenus()
{
    // 文件菜单
    QMenu *fileMenu = menuBar()->addMenu(tr("文件(&F)"));
    QAction *newAction = fileMenu->addAction(tr("新建(&N)"), this, &MyNotepad::newFile);
    // ... 其他菜单项
    // 存储所有动作的映射
    actionMap["新建"] = newAction;
    // ... 其他操作项
}
```

- 使用Qt的菜单系统创建标准菜单栏
- 支持快捷键操作，将操作映射到`actionMap`上实现快捷键编辑与菜单动作解耦
- 提供工具栏快速访问常用功能

3.2 文件操作实现

```
void MyNotepad::openFile()
{
    if (maybeSave()) {
        QString fileName = QFileDialog::getOpenFileName(this,
            tr("打开文件"), QString(),
            tr("文本文件 (*.txt);;所有文件 (*)"));
        // ... 文件打开逻辑
    }
}
```

- 使用Qt的文件对话框进行文件选择
- 支持多种文件格式
- 实现文件编码自动识别

3.3 多标签页管理

```
void MyNotepad::setupConnections()
{
    connect(tabManager, &TabManager::currentTabChanged, this, &MyNotepad::onTabChanged);
    connect(tabManager, &TabManager::textChanged, this, &MyNotepad::onTextChanged);
}
```

- 使用信号槽机制实现标签页状态同步
- 支持标签页之间的切换
- 维护每个标签页的独立状态

3.4 自动保存机制

```
void MyNotepad::toggleAutoSave()
{
    autoSaveManager->setAutoSaveEnabled(!autoSaveManager->isAutoSaveEnabled());
}
```

- 使用定时器实现定期保存
- 保存到临时文件
- 程序启动时自动恢复

3.4 全局异常捕获

```
 void customMessageHandler(QtMsgType type, const QMessageLogContext &context, const QString
 &msg)
 {
     Q_UNUSED(context);  // 标记未使用的参数

     QString txt;
     switch (type) {
     case QtDebugMsg:
         txt = QString("Debug: %1").arg(msg);
```

```
        break;
    case QtInfoMsg:
        txt = QString("Info: %1").arg(msg);
        break;
    case QtWarningMsg:
        txt = QString("Warning: %1").arg(msg);
        break;
    case QtCriticalMsg:
        txt = QString("Critical: %1").arg(msg);
        break;
    case QtFatalMsg:
        txt = QString("Fatal: %1").arg(msg);
        break;
    }

    // 创建日志目录
    QDir dir;
    if (!dir.exists("logs")) {
        dir.mkdir("logs");
    }

    // 写入日志文件
    QFile outFile("logs/crash.log");
    outFile.open(QIODevice::WriteOnly | QIODevice::Append);
    QTextStream ts(&outFile);
    ts << QDateTime::currentDateTime().toString("yyyy-MM-dd hh:mm:ss") << " - " << txt << "\n";
    outFile.close();

    // 如果是致命错误，显示错误对话框
    if (type == QtFatalMsg) {
        QMessageBox::critical(nullptr, "程序错误",
            QString("程序发生严重错误：\n%1\n\n"
                    "错误已记录到日志文件：logs/crash.log\n"
                    "请将日志文件发送给开发人员以帮助解决问题。").arg(msg));
    }
}
```

- 注册全局异常处理函数并通过消息处理器将异常处理函数绑定到程序。

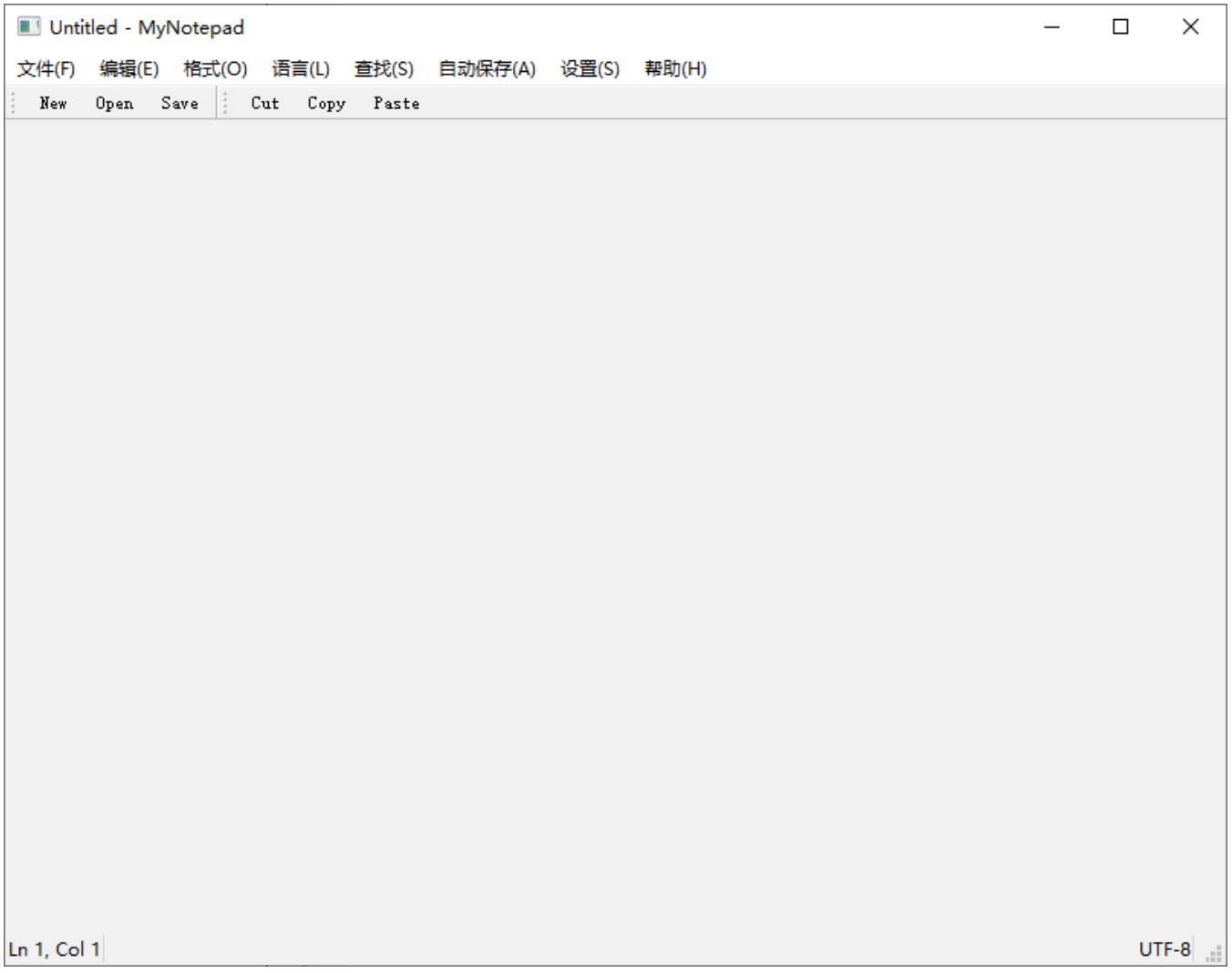
4. 项目特色

1. **模块化设计**：各个功能模块独立封装，便于维护和扩展
2. **用户友好**：提供丰富的快捷键和工具栏
3. **智能保存**：自动保存功能防止意外丢失
4. **多语言支持**：支持多种编程语言的语法高亮
5. **可定制性**：支持自定义快捷键和界面布局

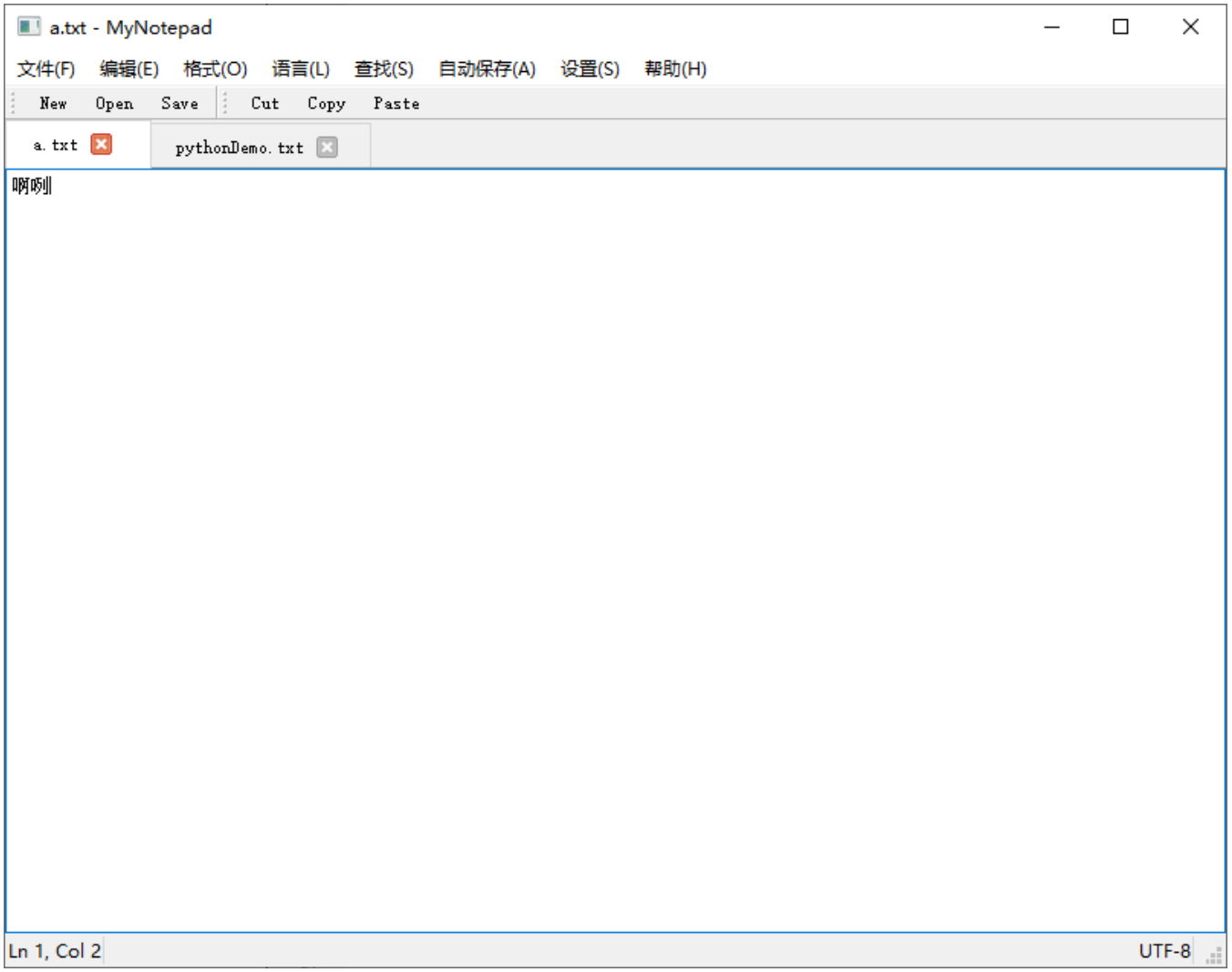
5. 技术难点及解决方案

1. **文件编码处理**
    - 使用QTextCodec自动识别文件编码
    - 支持多种编码格式的读写
2. **多标签页管理**
    - 使用TabManager类统一管理标签页
    - 实现标签页状态的保存和恢复
3. **自动保存机制**
    - 使用定时器实现定期保存
    - 采用临时文件存储机制
4. **语法高亮实现**
    - 使用正则表达式匹配语法规则
    - 支持多种编程语言的语法规则

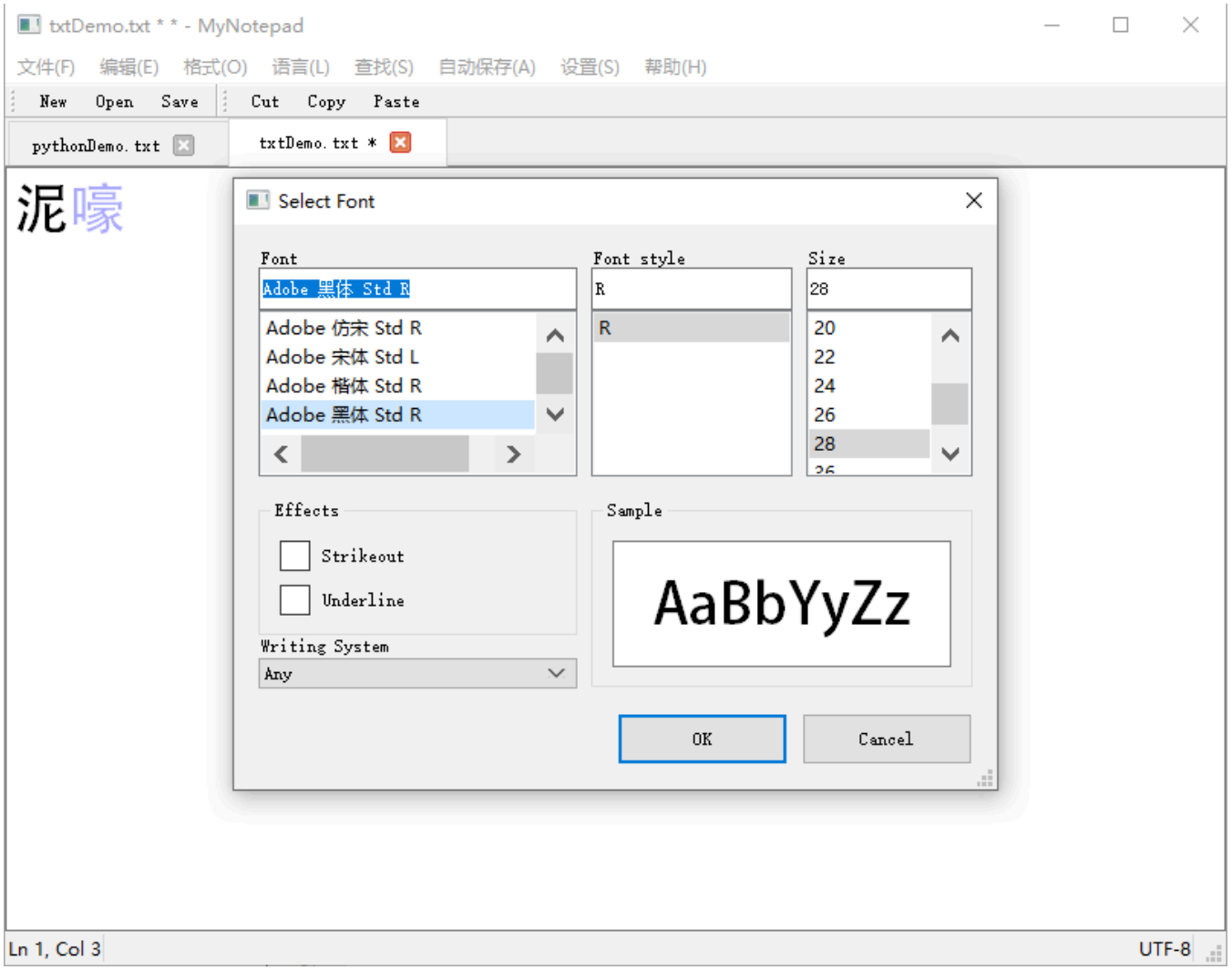
6. 运行效果图

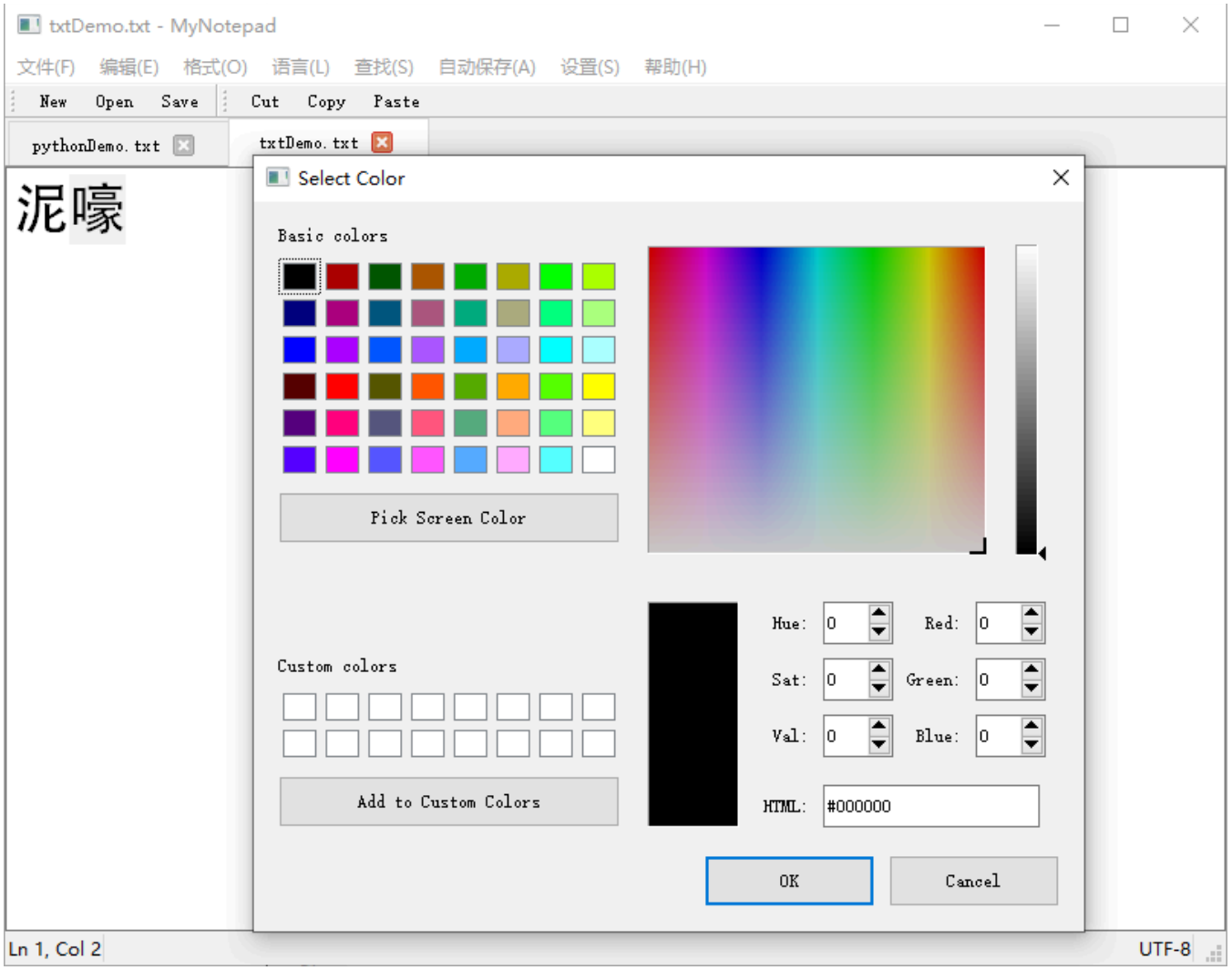


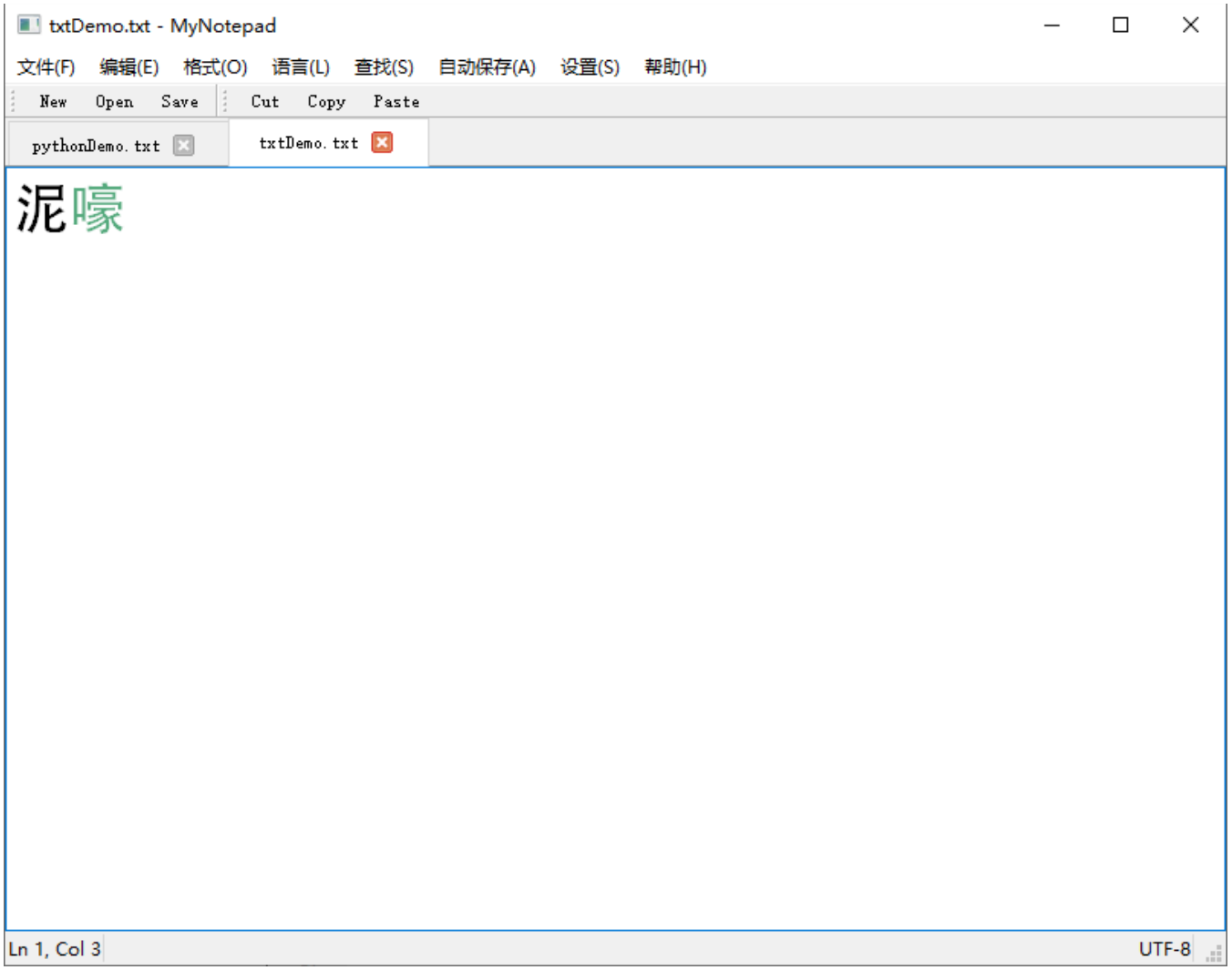
初始化窗口



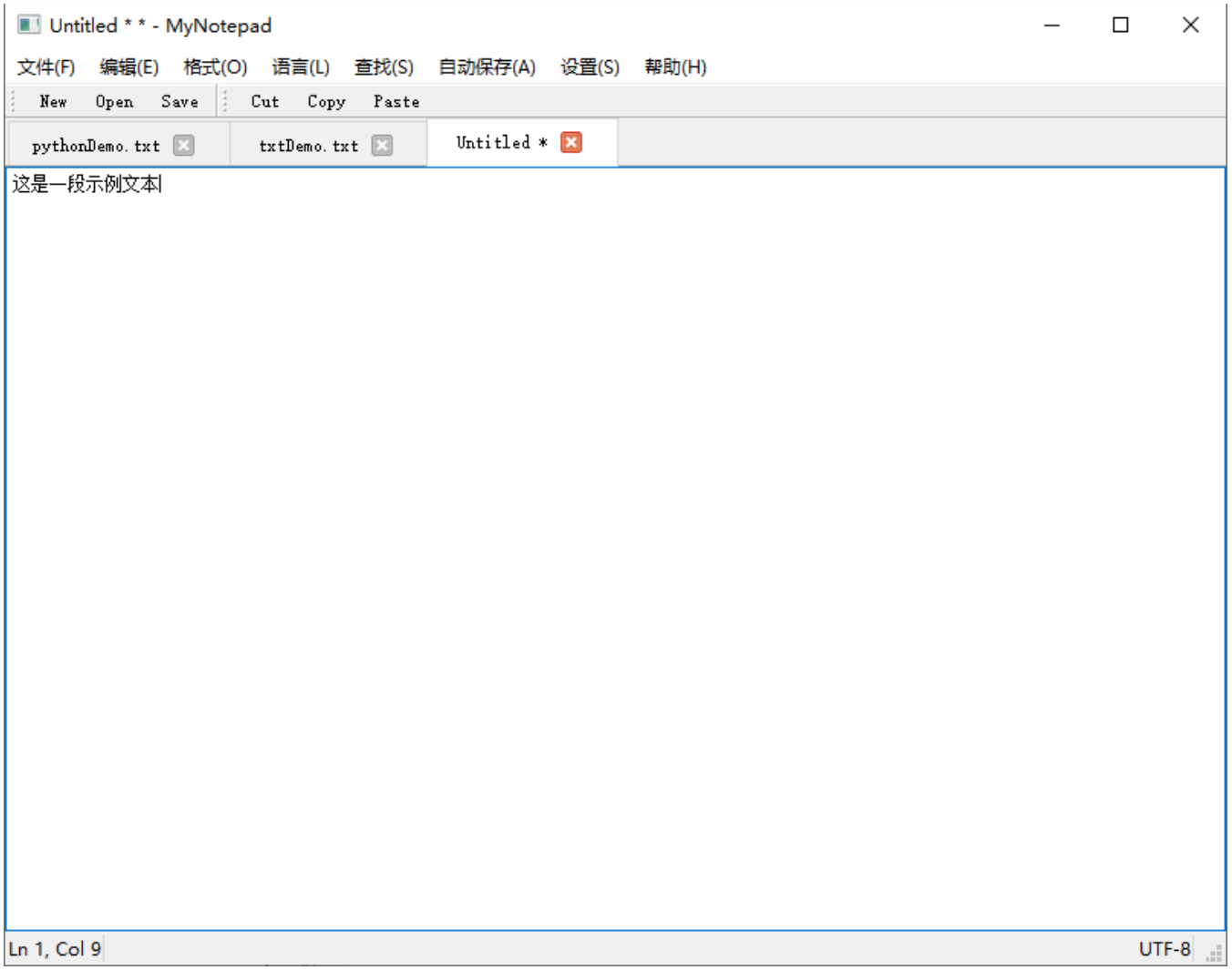
多标签页编辑

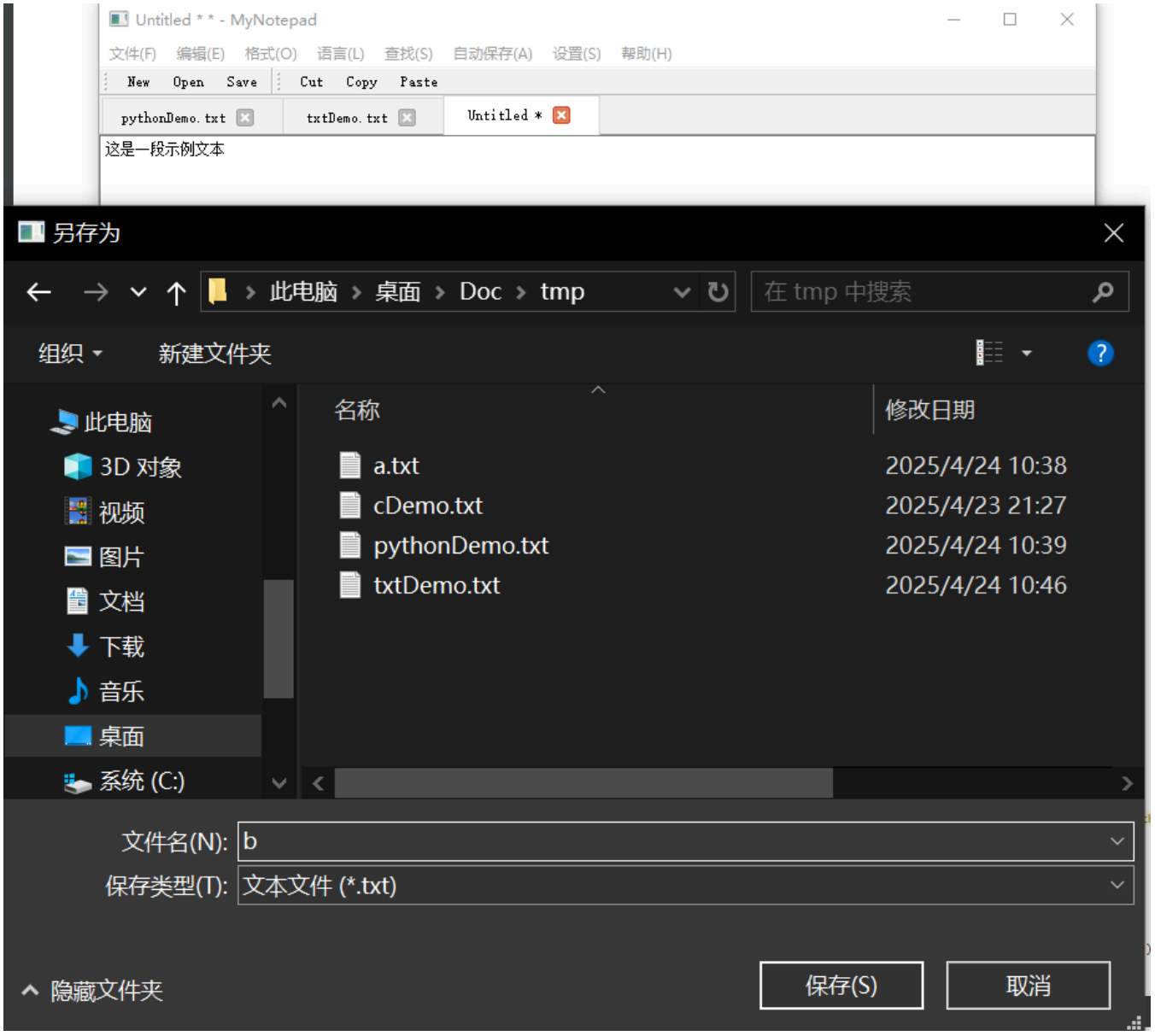

txtDemo.txt * * - MyNotepad
文件(F) 编辑(E) 格式(O) 语言(L) 查找(S) 自动保存(A) 设置(S) 帮助(H)
New Open Save Cut Copy Paste
pythonDemo.txt
txtDemo.txt *
泥嚎
Select Font
Font
Adobe 黑体 Std R
Adobe 仿宋 Std R
Adobe 宋体 Std L
Adobe 楷体 Std R
Adobe 黑体 Std R
Font style
R
R
Size
28
20
22
24
26
28
Effects
Strikeout
Underline
Writing System
Any
Sample
AaBbYyZz
OK
Cancel
Ln 1, Col 3
UTF-8

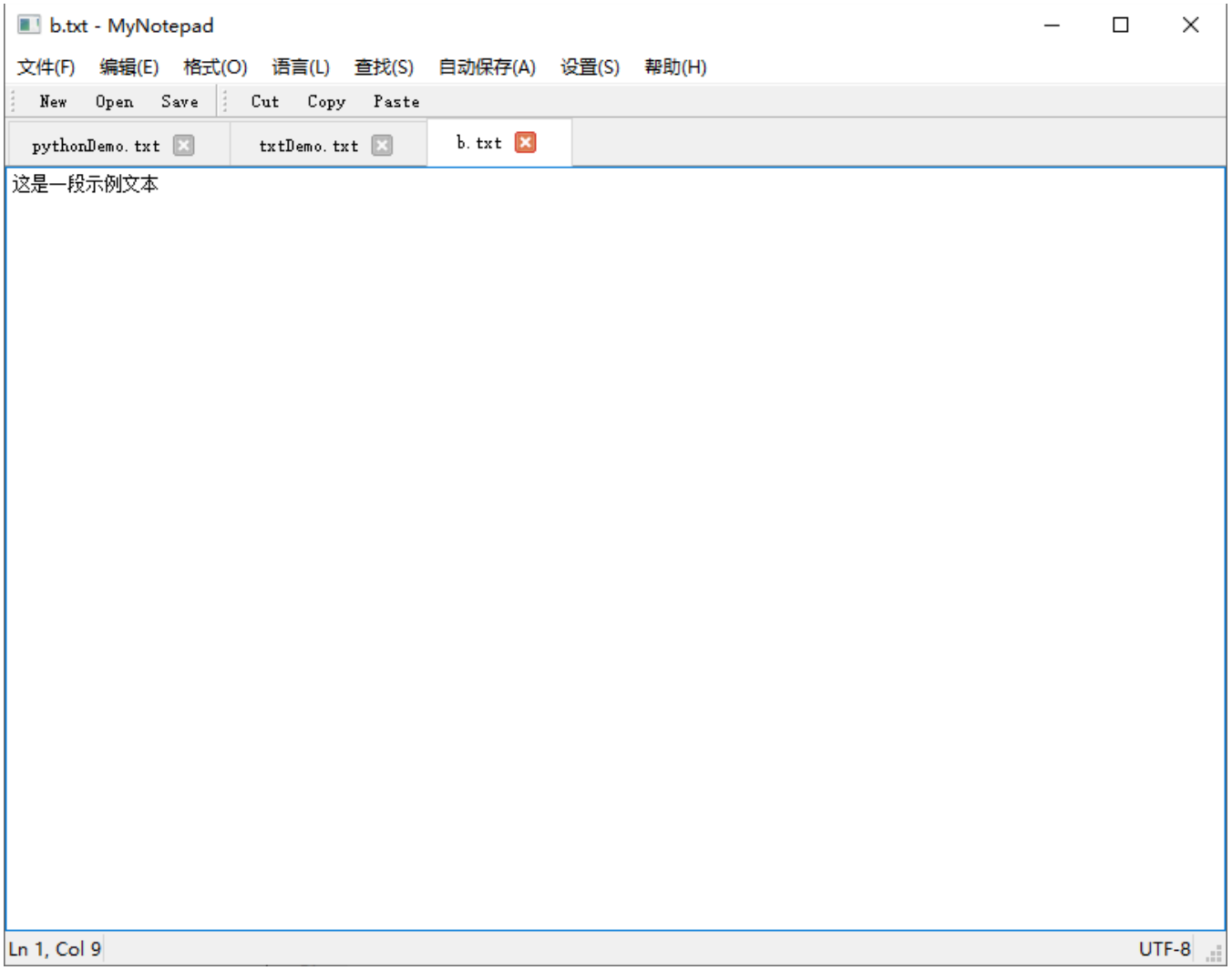

txtDemo.txt - MyNotepad
文件(F) 编辑(E) 格式(O) 语言(L) 查找(S) 自动保存(A) 设置(S) 帮助(H)
New Open Save Cut Copy Paste
pythonDemo.txt
txtDemo.txt
泥嚎
Select Color
Basic colors
Pick Screen Color
Custom colors
Add to Custom Colors
Hue: 0
Sat: 0
Val: 0
Red: 0
Green: 0
Blue: 0
HTML: #000000
OK
Cancel
Ln 1, Col 2
UTF-8


txtDemo.txt - MyNotepad
文件(F) 编辑(E) 格式(O) 语言(L) 查找(S) 自动保存(A) 设置(S) 帮助(H)
New Open Save Cut Copy Paste
pythonDemo.txt
txtDemo.txt
泥曚
Ln 1, Col 3
UTF-8

字体格式设置

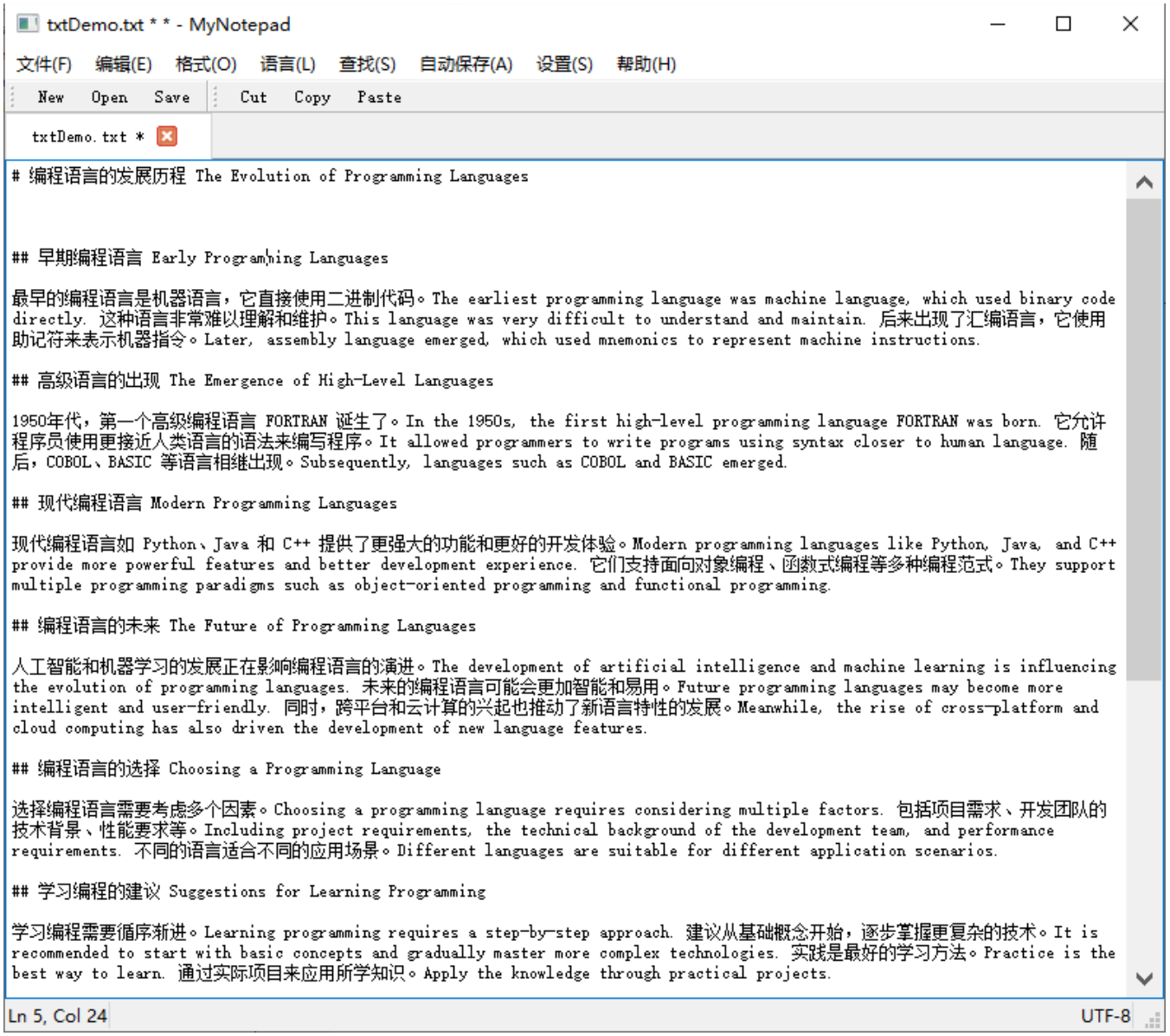

Untitled * * - MyNotepad
文件(F) 编辑(E) 格式(O) 语言(L) 查找(S) 自动保存(A) 设置(S) 帮助(H)
New Open Save Cut Copy Paste
pythonDemo.txt
txtDemo.txt
Untitled *
这是一段示例文本
Ln 1, Col 9
UTF-8

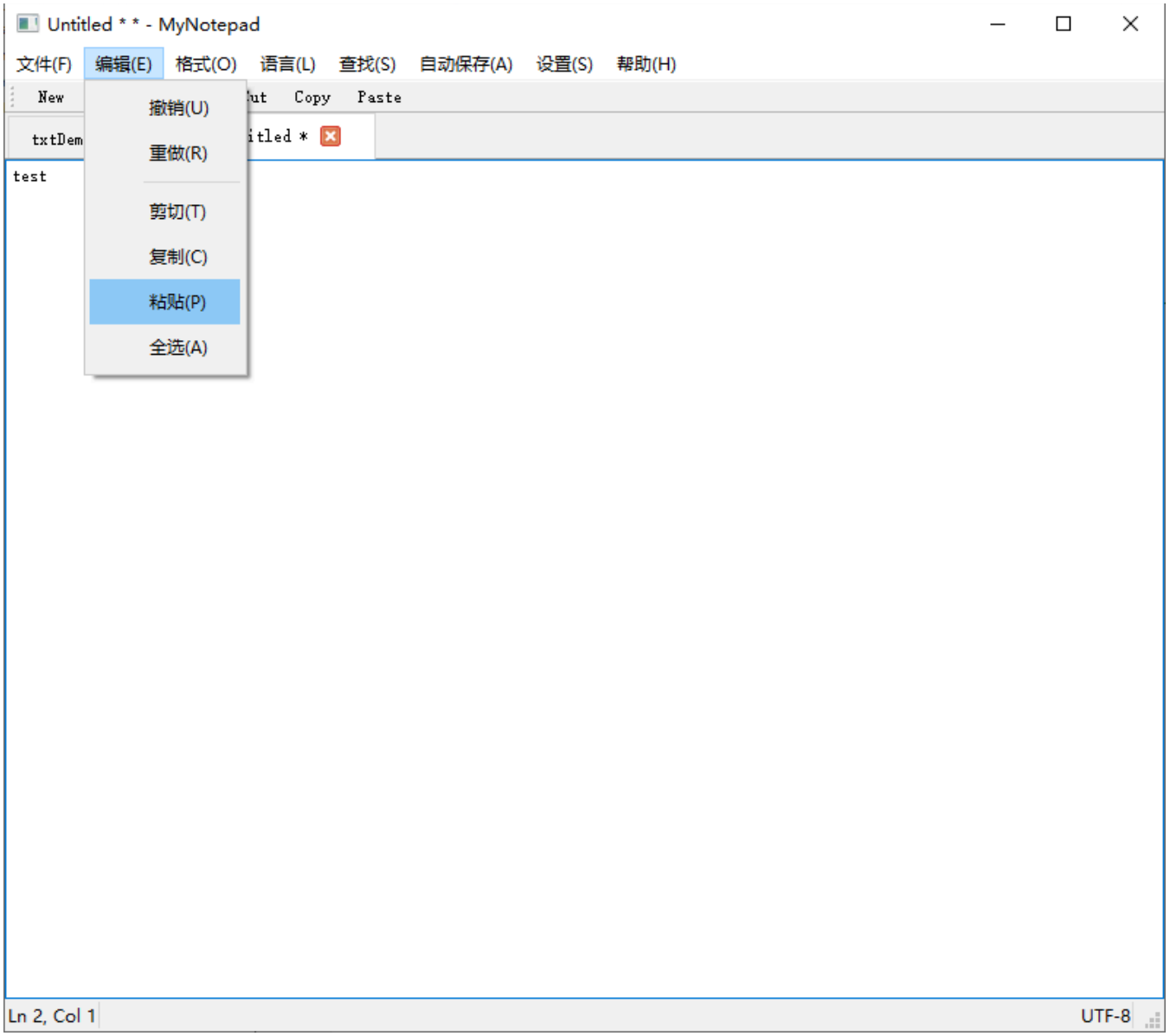

Untitled * - MyNotepad
文件(F) 编辑(E) 格式(O) 语言(L) 查找(S) 自动保存(A) 设置(S) 帮助(H)
New Open Save Cut Copy Paste
pythonDemo.txt
txtDemo.txt
Untitled *
这是一段示例文本
另存为
此电脑 › 桌面 › Doc › tmp
在 tmp 中搜索
组织
新建文件夹
此电脑
3D 对象
视频
图片
文档
下载
音乐
桌面
系统 (C:)
名称
修改日期
a.txt 2025/4/24 10:38
cDemo.txt 2025/4/23 21:27
pythonDemo.txt 2025/4/24 10:39
txtDemo.txt 2025/4/24 10:46
文件名(N): b
保存类型(T): 文本文件 (*.txt)
隐藏文件夹
保存(S)
取消

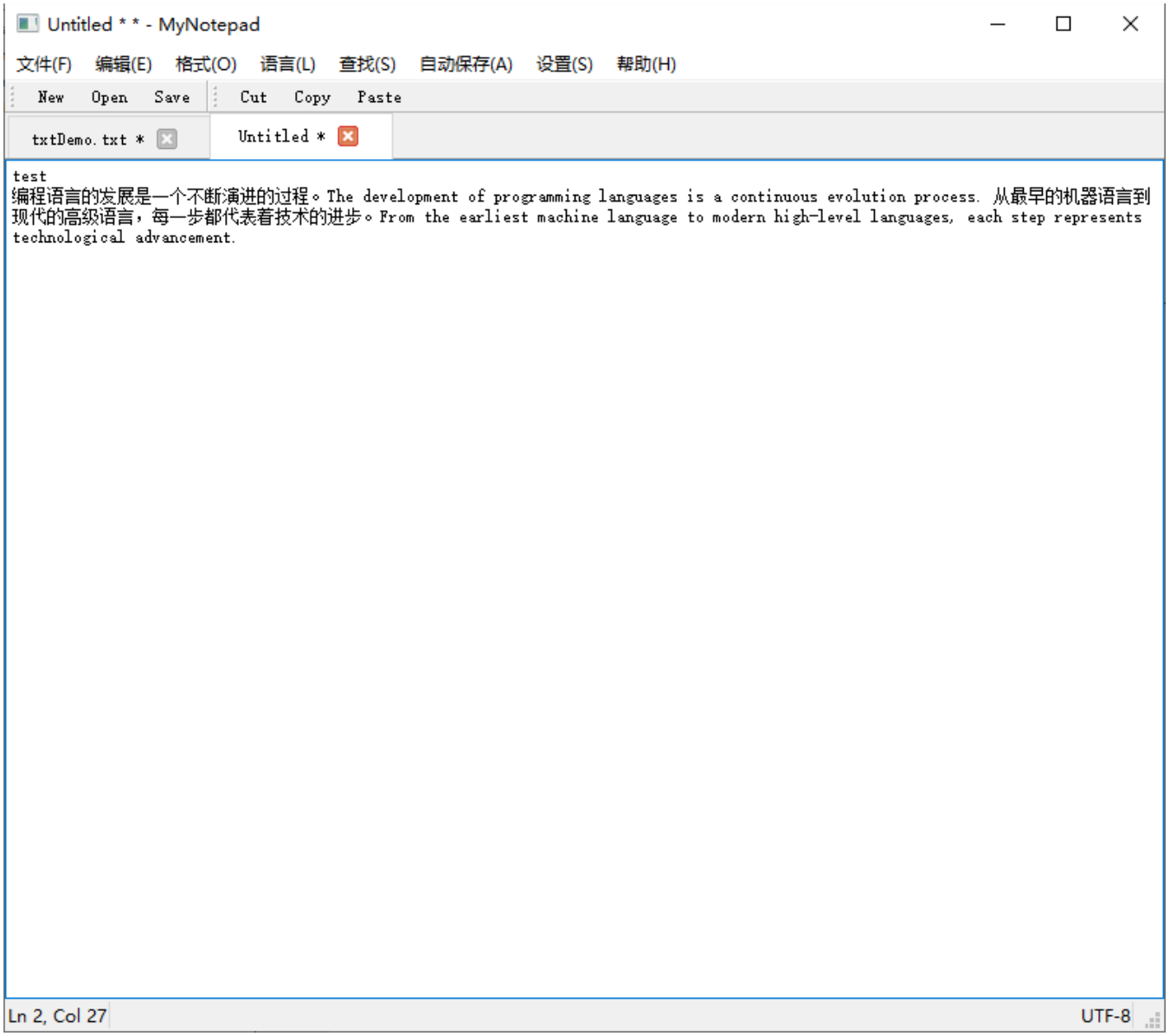

b.txt - MyNotepad
文件(F) 编辑(E) 格式(O) 语言(L) 查找(S) 自动保存(A) 设置(S) 帮助(H)
New Open Save Cut Copy Paste
pythonDemo.txt
txtDemo.txt
b.txt
这是一段示例文本
Ln 1, Col 9
UTF-8

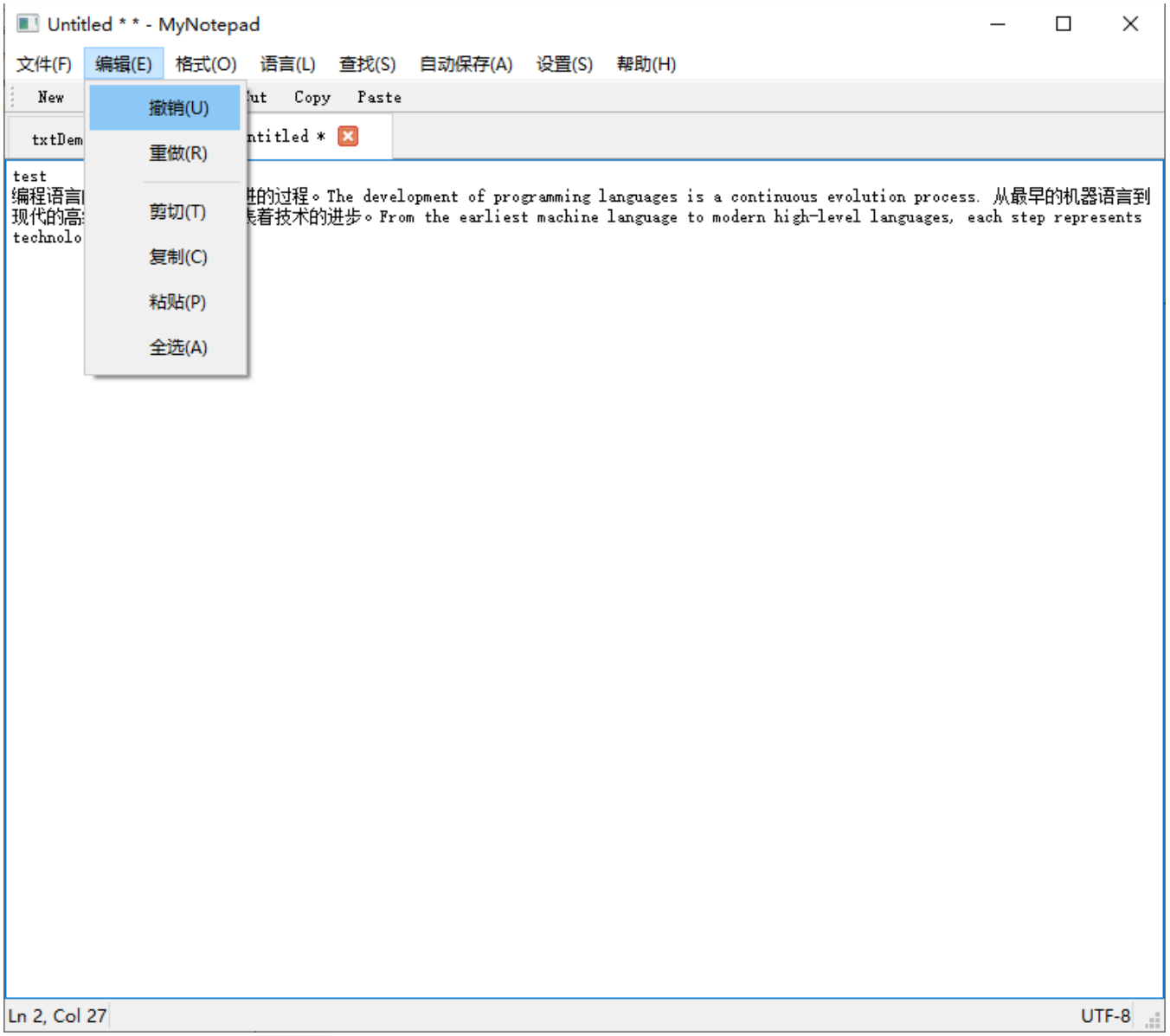
新建文件、保存

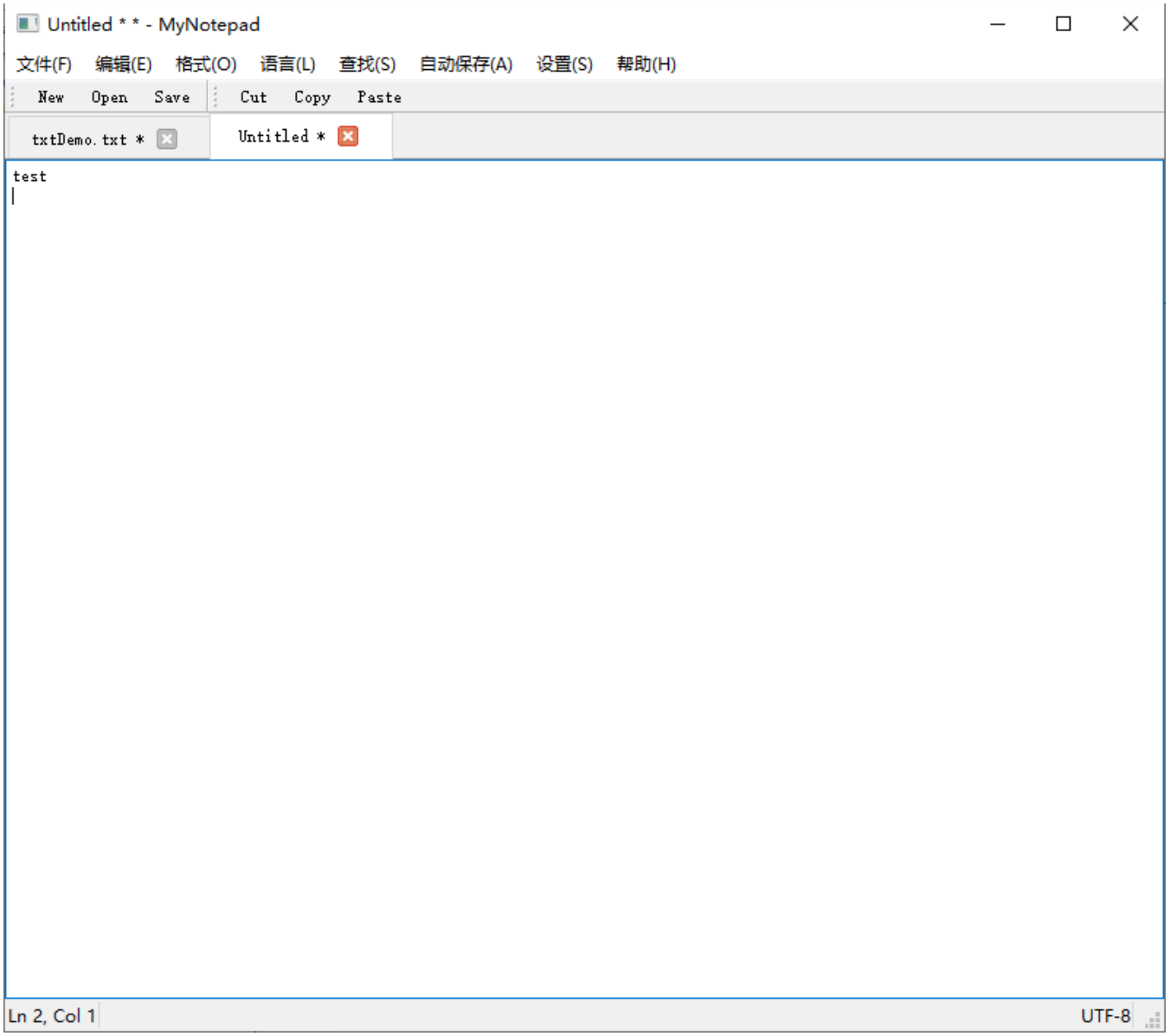

txtDemo.txt - MyNotepad
文件(F)
编辑(E)
格式(O)
语言(L)
查找(S)
自动保存(A)
设置(S)
帮助(H)
New
Copy
Paste
撤销(U)
重做(R)
剪切(T)
复制(C)
粘贴(P)
全选(A)
编程语
olution of Programming Languages
编程语言
进的过程。The development of programming languages is a continuous evolution process. 从最早的机器语言到现代
都代表着技术的进步。From the earliest machine language to modern high-level languages, each step represen
vancement.
早期
amming Languages
最早的编
它直接使用二进制代码。The earliest programming language was machine language, which used binary code directly
理解和维护。This language was very difficult to understand and maintain. 后来出现了汇编语言，它使用助记符来表示机器指令。Later, assembly language emerged, which used mnemonics to represent machine instructions.
高级语言的出现 The Emergence of High-Level Languages
1950年代，第一个高级编程语言 FORTRAN 诞生了。In the 1950s, the first high-level programming language FORTRAN was born. 它允许程序员使用更接近人类语言的语法来编写程序。It allowed programmers to write programs using syntax closer to human language. 随后，COBOL、BASIC 等语言相继出现。Subsequently, languages such as COBOL and BASIC emerged.
现代编程语言 Modern Programming Languages
现代编程语言如 Python、Java 和 C++ 提供了更强大的功能和更好的开发体验。Modern programming languages like Python, Java, and C++ provide more powerful features and better development experience. 它们支持面向对象编程、函数式编程等多种编程范式。They support multiple programming paradigms such as object-oriented programming and functional programming.
编程语言的未来 The Future of Programming Languages
人工智能和机器学习的发展正在影响编程语言的演进。The development of artificial intelligence and machine learning is influencing the evolution of programming languages. 未来的编程语言可能会更加智能和易用。Future programming languages may become more intelligent and user-friendly. 同时，跨平台和云计算的兴起也推动了新语言特性的发展。Meanwhile, the rise of cross-platform and cloud computing has also driven the development of new language features.
编程语言的选择 Choosing a Programming Language
选择编程语言需要考虑多个因素。Choosing a programming language requires considering multiple factors. 包括项目需求、开发团队的技术背景、性能要求等。Including project requirements, the technical background of the development team, and performance requirements. 不同的语言适合不同的应用场景。Different languages are suitable for different application scenarios.
学习编程的建议 Suggestions for Learning Programming
学习编程需要循序渐进。Learning programming requires a step-by-step approach. 建议从基础概念开始，逐步掌握更复杂的技术。It is recommended to start with basic concepts and gradually master more complex technologies. 实践是最好的学习方法。Practice is the
Ln 3, Col 1
UTF-8

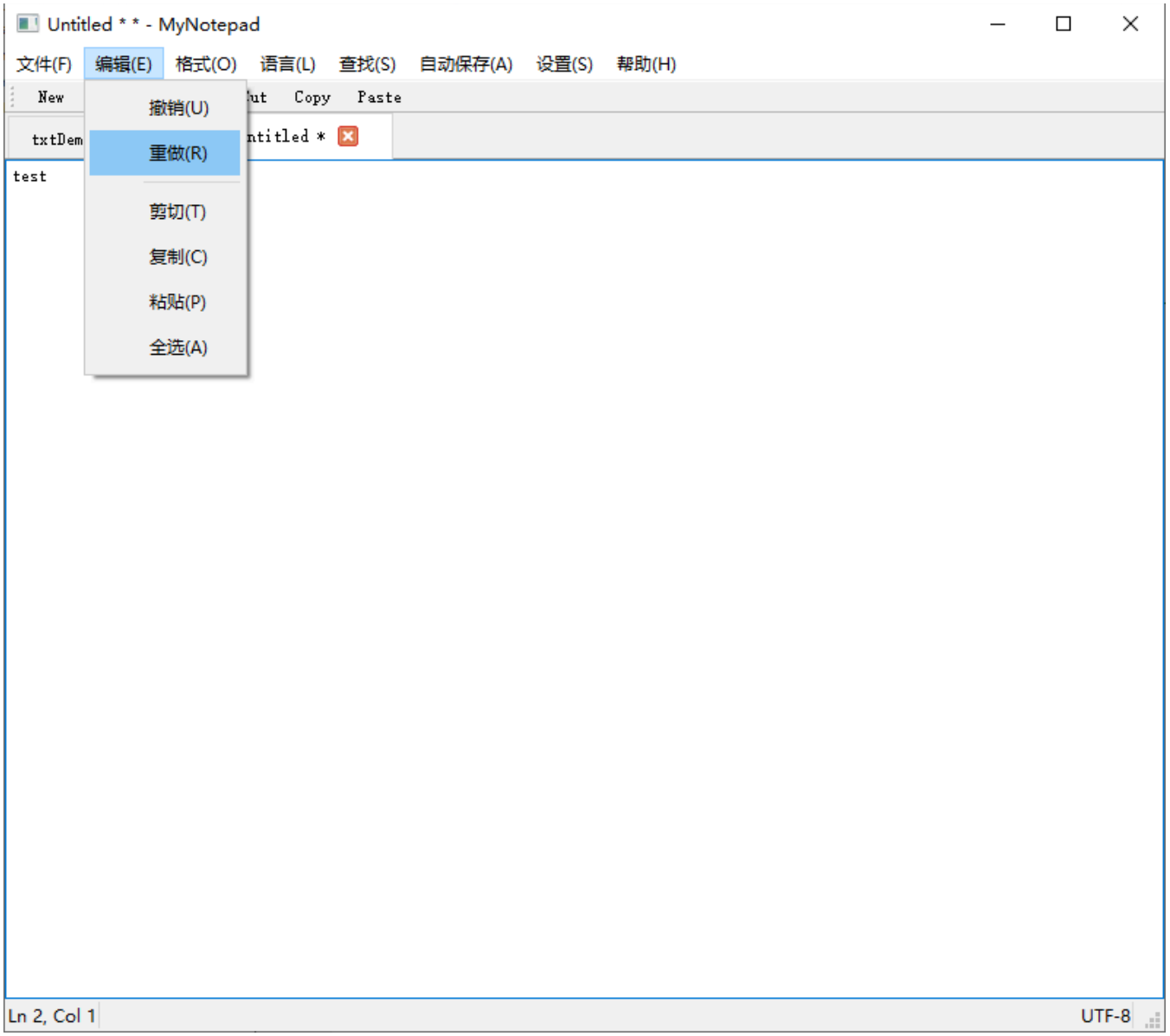

txtDemo.txt * * - MyNotepad
文件(F) 编辑(E) 格式(O) 语言(L) 查找(S) 自动保存(A) 设置(S) 帮助(H)
New Open Save Cut Copy Paste
txtDemo.txt *
编程语言的发展历程 The Evolution of Programming Languages
早期编程语言 Early Programming Languages
最早的编程语言是机器语言，它直接使用二进制代码。The earliest programming language was machine language, which used binary code directly. 这种语言非常难以理解和维护。This language was very difficult to understand and maintain. 后来出现了汇编语言，它使用助记符来表示机器指令。Later, assembly language emerged, which used mnemonics to represent machine instructions.
高级语言的出现 The Emergence of High-Level Languages
1950年代，第一个高级编程语言 FORTRAN 诞生了。In the 1950s, the first high-level programming language FORTRAN was born. 它允许程序员使用更接近人类语言的语法来编写程序。It allowed programmers to write programs using syntax closer to human language. 随后，COBOL、BASIC 等语言相继出现。Subsequently, languages such as COBOL and BASIC emerged.
现代编程语言 Modern Programming Languages
现代编程语言如 Python、Java 和 C++ 提供了更强大的功能和更好的开发体验。Modern programming languages like Python, Java, and C++ provide more powerful features and better development experience. 它们支持面向对象编程、函数式编程等多种编程范式。They support multiple programming paradigms such as object-oriented programming and functional programming.
编程语言的未来 The Future of Programming Languages
人工智能和机器学习的发展正在影响编程语言的演进。The development of artificial intelligence and machine learning is influencing the evolution of programming languages. 未来的编程语言可能会更加智能和易用。Future programming languages may become more intelligent and user-friendly. 同时，跨平台和云计算的兴起也推动了新语言特性的发展。Meanwhile, the rise of cross-platform and cloud computing has also driven the development of new language features.
编程语言的选择 Choosing a Programming Language
选择编程语言需要考虑多个因素。Choosing a programming language requires considering multiple factors. 包括项目需求、开发团队的技术背景、性能要求等。Including project requirements, the technical background of the development team, and performance requirements. 不同的语言适合不同的应用场景。Different languages are suitable for different application scenarios.
学习编程的建议 Suggestions for Learning Programming
学习编程需要循序渐进。Learning programming requires a step-by-step approach. 建议从基础概念开始，逐步掌握更复杂的技术。It is recommended to start with basic concepts and gradually master more complex technologies. 实践是最好的学习方法。Practice is the best way to learn. 通过实际项目来应用所学知识。Apply the knowledge through practical projects.
Ln 5, Col 24
UTF-8

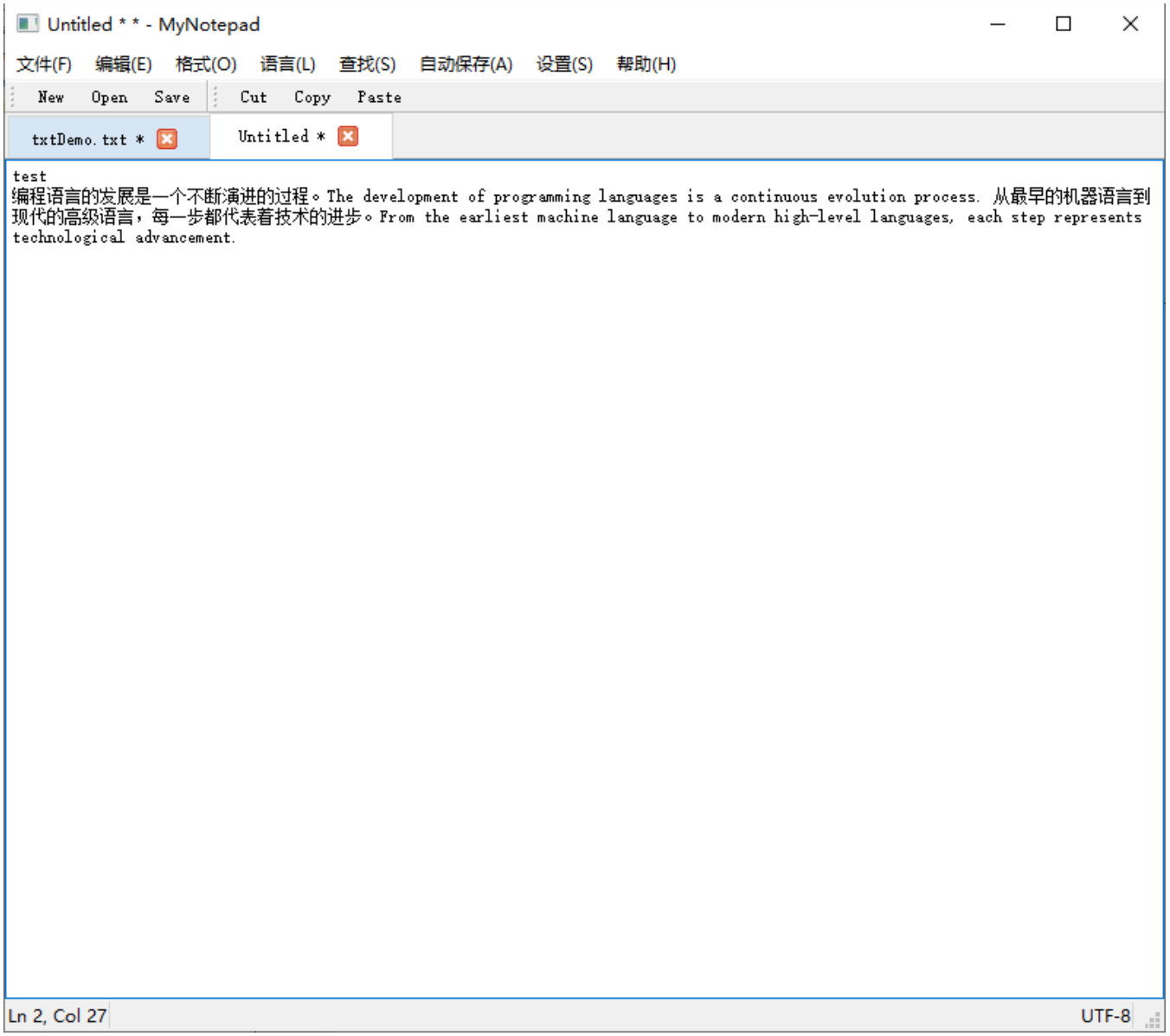

Untitled * * - MyNotepad
文件(F)
编辑(E)
格式(O)
语言(L)
查找(S)
自动保存(A)
设置(S)
帮助(H)
New
Copy
Paste
撤销(U)
重做(R)
剪切(T)
复制(C)
粘贴(P)
全选(A)
test
Ln 2, Col 1
UTF-8


Untitled * * - MyNotepad
文件(F) 编辑(E) 格式(O) 语言(L) 查找(S) 自动保存(A) 设置(S) 帮助(H)
New Open Save Cut Copy Paste
txtDemo.txt *
Untitled *
test
编程语言的发展是一个不断演进的过程。The development of programming languages is a continuous evolution process. 从最早的机器语言到现代的高级语言，每一步都代表着技术的进步。From the earliest machine language to modern high-level languages, each step represents technological advancement.
Ln 2, Col 27
UTF-8

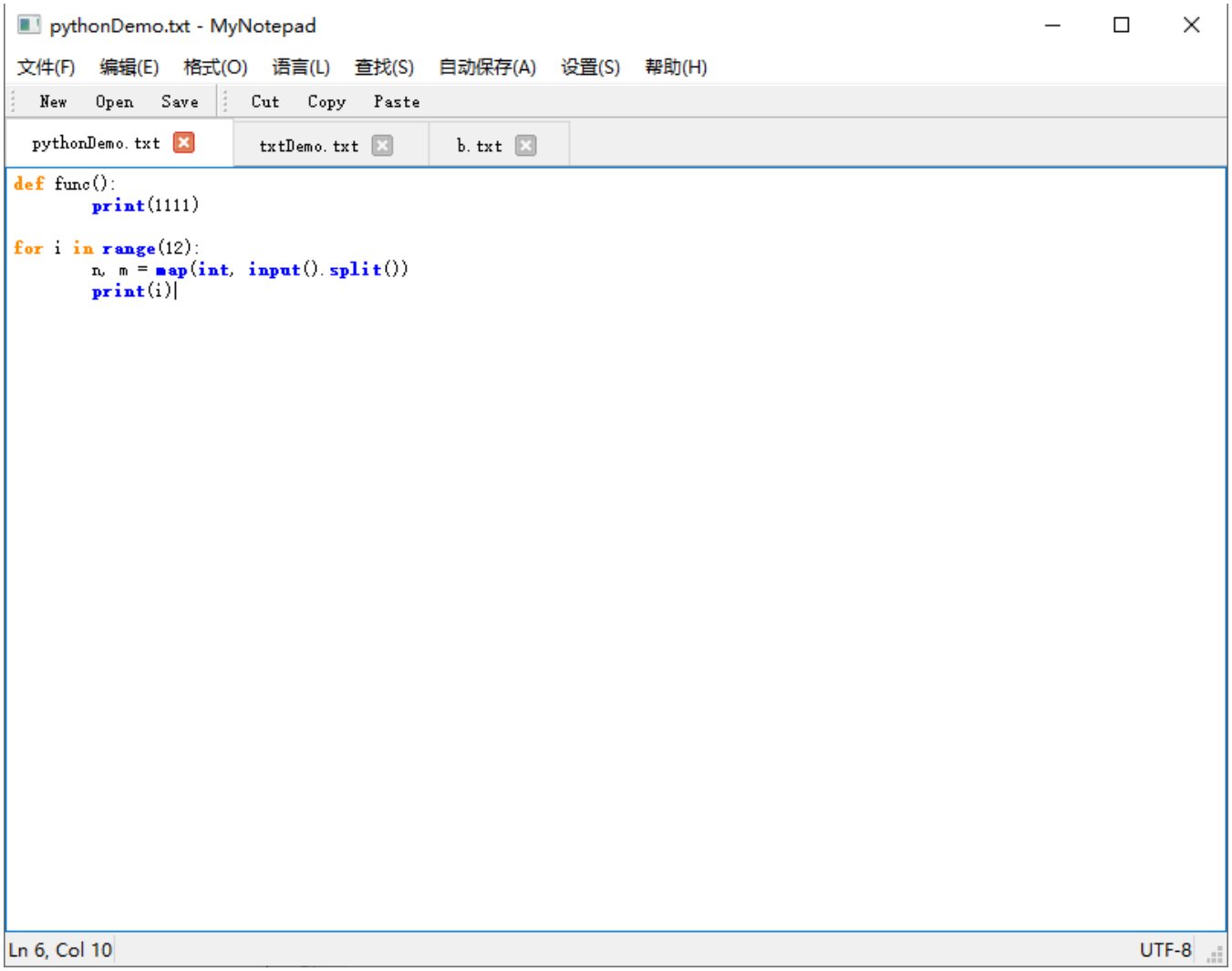

Untitled * * - MyNotepad
文件(F)
编辑(E)
格式(O)
语言(L)
查找(S)
自动保存(A)
设置(S)
帮助(H)
New
ut
Copy
Paste
撤销(U)
重做(R)
剪切(T)
复制(C)
粘贴(P)
全选(A)
ntitled *
test
的过程。The development of programming languages is a continuous evolution process. 从最早的机器语言到
现代的高
着技术的进步。From the earliest machine language to modern high-level languages, each step represents
technolo
Ln 2, Col 27
UTF-8

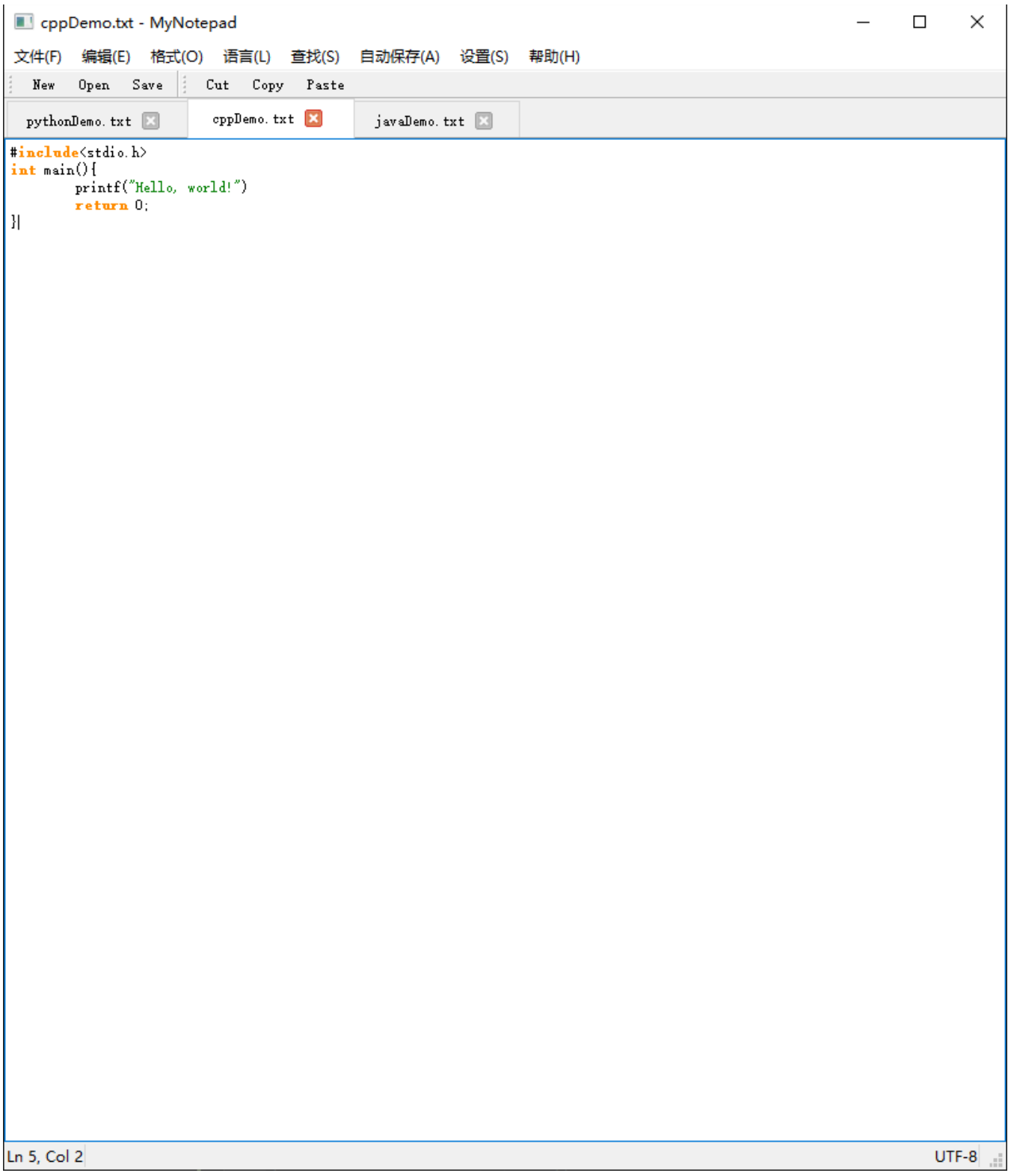

Untitled * * - MyNotepad
文件(F) 编辑(E) 格式(O) 语言(L) 查找(S) 自动保存(A) 设置(S) 帮助(H)
New Open Save Cut Copy Paste
txtDemo.txt *
Untitled *
test
Ln 2, Col 1
UTF-8

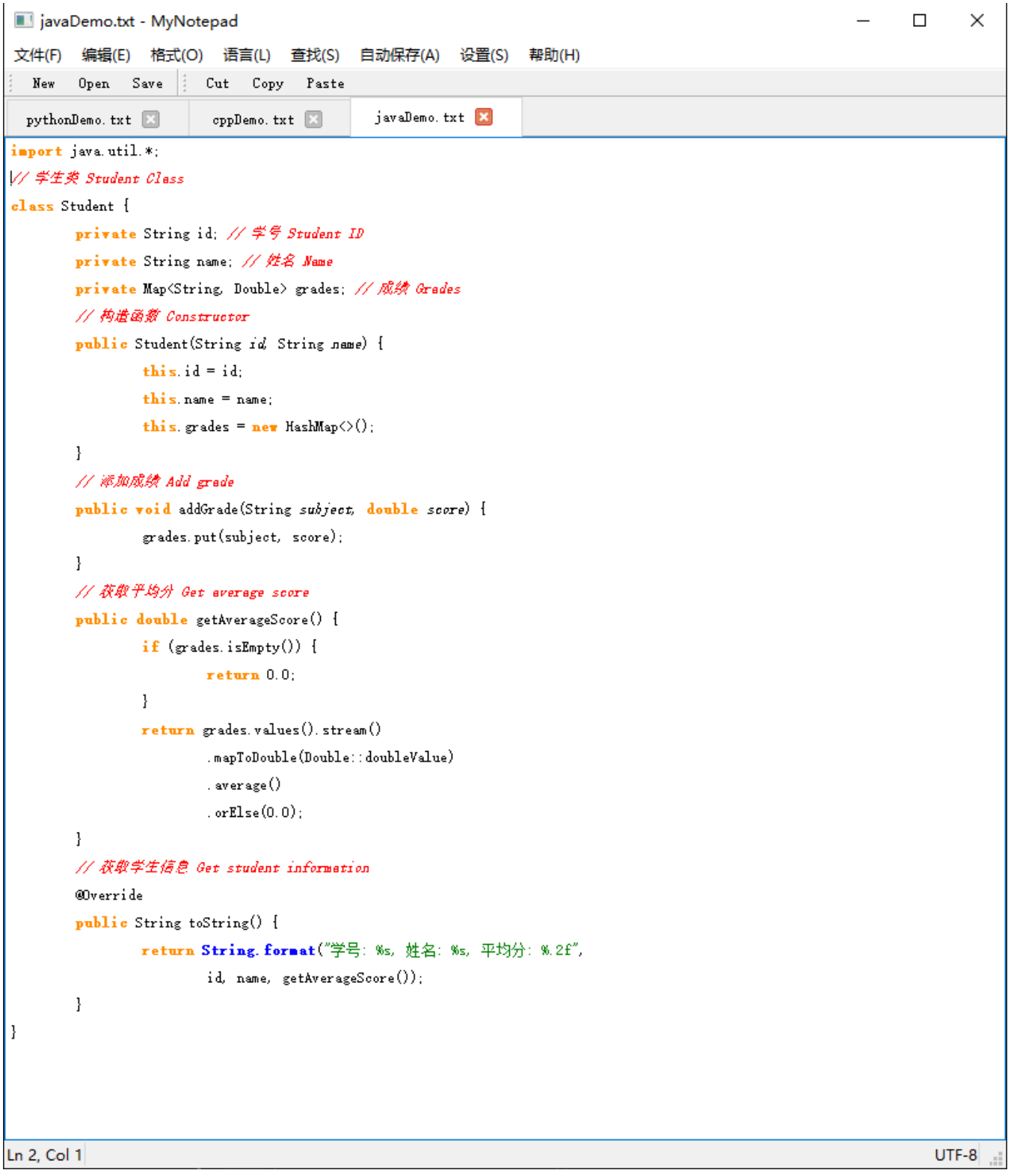

Untitled * * - MyNotepad
文件(F)
编辑(E)
格式(O)
语言(L)
查找(S)
自动保存(A)
设置(S)
帮助(H)
New
Copy
Paste
撤销(U)
重做(R)
剪切(T)
复制(C)
粘贴(P)
全选(A)
test
Ln 2, Col 1
UTF-8


Untitled * * - MyNotepad
文件(F) 编辑(E) 格式(O) 语言(L) 查找(S) 自动保存(A) 设置(S) 帮助(H)
New Open Save Cut Copy Paste
txtDemo.txt *
Untitled *
test
编程语言的发展是一个不断演进的过程。The development of programming languages is a continuous evolution process. 从最早的机器语言到现代的高级语言，每一步都代表着技术的进步。From the earliest machine language to modern high-level languages, each step represents technological advancement.
Ln 2, Col 27
UTF-8

常用编辑操作


pythonDemo.txt - MyNotepad
文件(F) 编辑(E) 格式(O) 语言(L) 查找(S) 自动保存(A) 设置(S) 帮助(H)
New Open Save Cut Copy Paste
pythonDemo.txt
txtDemo.txt
b.txt
def func():
print(1111)
for i in range(12):
n, m = map(int, input().split())
print(i)
Ln 6, Col 10
UTF-8


cppDemo.txt - MyNotepad
文件(F) 编辑(E) 格式(O) 语言(L) 查找(S) 自动保存(A) 设置(S) 帮助(H)
New Open Save Cut Copy Paste
pythonDemo.txt
cppDemo.txt
javaDemo.txt
#include<stdio.h>
int main(){
printf("Hello, world!")
return 0;
}
Ln 5, Col 2
UTF-8


javaDemo.txt - MyNotepad
文件(F) 编辑(E) 格式(O) 语言(L) 查找(S) 自动保存(A) 设置(S) 帮助(H)
New Open Save Cut Copy Paste
pythonDemo.txt
cppDemo.txt
javaDemo.txt
import java.util.*;
// 学生类 Student Class
class Student {
private String id; // 学号 Student ID
private String name; // 姓名 Name
private Map<String, Double> grades; // 成绩 Grades
// 构造函数 Constructor
public Student(String id, String name) {
this.id = id;
this.name = name;
this.grades = new HashMap<>();
}
// 添加成绩 Add grade
public void addGrade(String subject, double score) {
grades.put(subject, score);
}
// 获取平均分 Get average score
public double getAverageScore() {
if (grades.isEmpty()) {
return 0.0;
}
return grades.values().stream()
.mapToDouble(Double::doubleValue)
.average()
.orElse(0.0);
}
// 获取学生信息 Get student information
@Override
public String toString() {
return String.format("学号: %s, 姓名: %s, 平均分: %.2f",
id, name, getAverageScore());
}
}
Ln 2, Col 1
UTF-8

代码高亮

编程语言的发展历程 The Evolution of Programming Languages

编程语言的发展是一个不断演进的过程。The development of programming languages is a continuous evolution process. 从最早的机器语言到现代的高级语言，每一步都代表着技术的进步。From the earliest machine language to modern high-level languages, each step represents technological advancement.

早期编程语言 Early Programming Languages

最早的编程语言是机器语言，它直接使用二进制代码。Th nary code directly. 这种语言非常难以理解和维护。This languag，它使用助记符来表示机器指令。Later, assembly language emerged

高级语言的出现 The Emergence of High-Level Lang

1950年代，第一个高级编程语言 FORTRAN 诞生了。In th 。它允许程序员使用更接近人类语言的语法来编写程序。It allowed p 随后，COBOL、BASIC 等语言相继出现。Subsequently, languag

现代编程语言 Modern Programming Languages

现代编程语言如 Python、Java 和 C++ 提供了更强大的功能和更好的开发体验。Modern programming languages like Python, Java, and C++ provide more powerful features and better development experience. 它们支持面向对象编程、函数式编程等多种编程范式。They support multiple programming paradigms such as object-oriented programming and functional programming.

编程语言的未来 The Future of Programming Languages

人工智能和机器学习的发展正在影响编程语言的演进。The development of artificial intelligence and machine learning is influencing the evolution of programming languages. 未来的编程语言可能会更加智能和易用。Future programming languages may become more intelligent and user-friendly. 同时，跨平台和云计算的兴起也推动了新语言特性的发展。Meanwhile, the rise of cross-platform and cloud computing has also driven the development of new language features.

编程语言的选择 Choosing a Programming Language

选择编程语言需要考虑多个因素。Choosing a programming language requires considering multiple factors. 包括项目需求、开发团队的技术背景、性能要求等。Including project requirements, the technical background of the development team, and performance requirements. 不同的语言适合不同的应用场景。Different languages are suitable for different application scenarios.

学习编程的建议 Suggestions for Learning Programming

学习编程需要循序渐进。Learning programming requires a step-by-step approach. 建议从基础概念开始，逐步掌握更复杂的技术。It is recommended to start with basic concepts and gradually master more complex technologies. 实践是最好的学习方法。Practice is the best way to learn. 通过实际项目来应用所学知识。Apply the knowledge through practical projects.

编程社区的重要性 The Importance of Programming Communities

编程社区是学习和交流的重要平台。Programming communities are important platforms for learning and communication. 在这里，开发者可以分享经验、解决问题。Here, developers can share experiences and solve problems. 开源项目的发展也促进了编程技术的进步。The development of open source projects has also promoted the advancement of programming technology.

编程工具的发展 The Development of Programming Tools

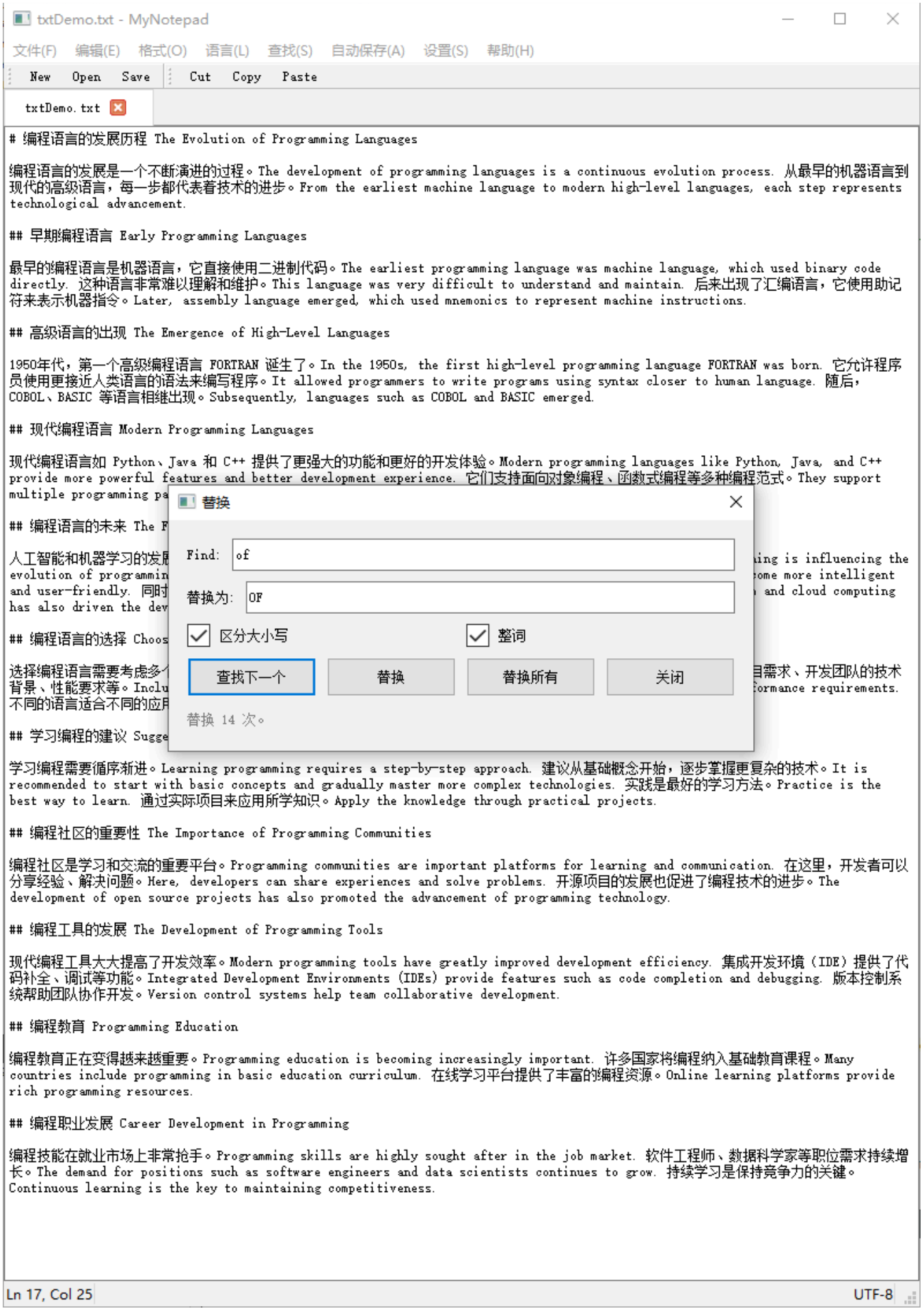
现代编程工具大大提高了开发效率。Modern programming tools have greatly improved development efficiency. 集成开发环境（IDE）提供了代码补全、调试等功能。Integrated Development Environments (IDEs) provide features such as code completion and debugging. 版本控制系统帮助团队协作开发。Version control systems help team collaborative development.

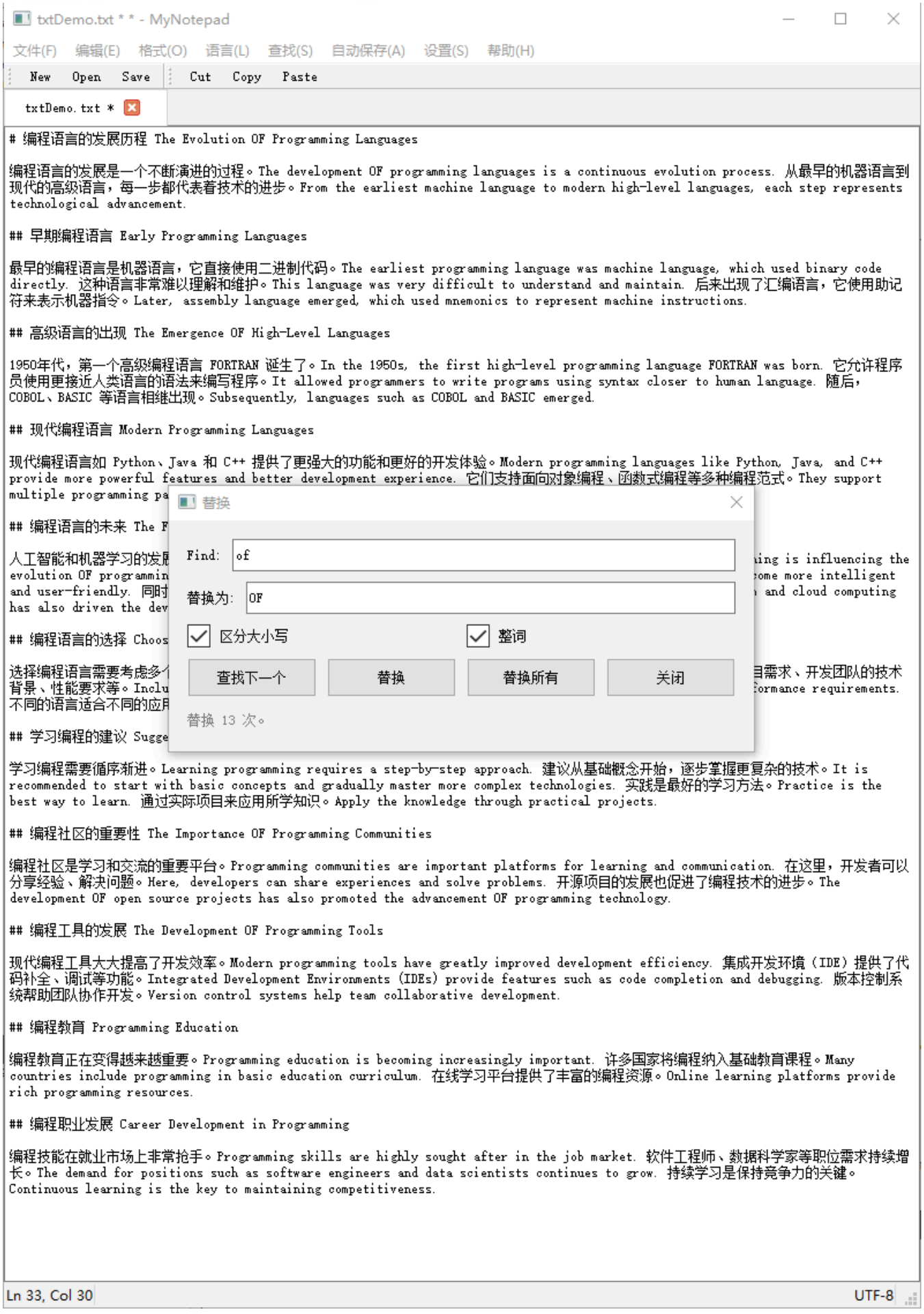
编程教育 Programming Education

编程教育正在变得越来越重要。Programming education is becoming increasingly important. 许多国家将编程纳入基础教育课程。Many countries include programming in basic education curriculum. 在线学习平台提供了丰富的编程资源。Online learning platforms provide rich programming resources.

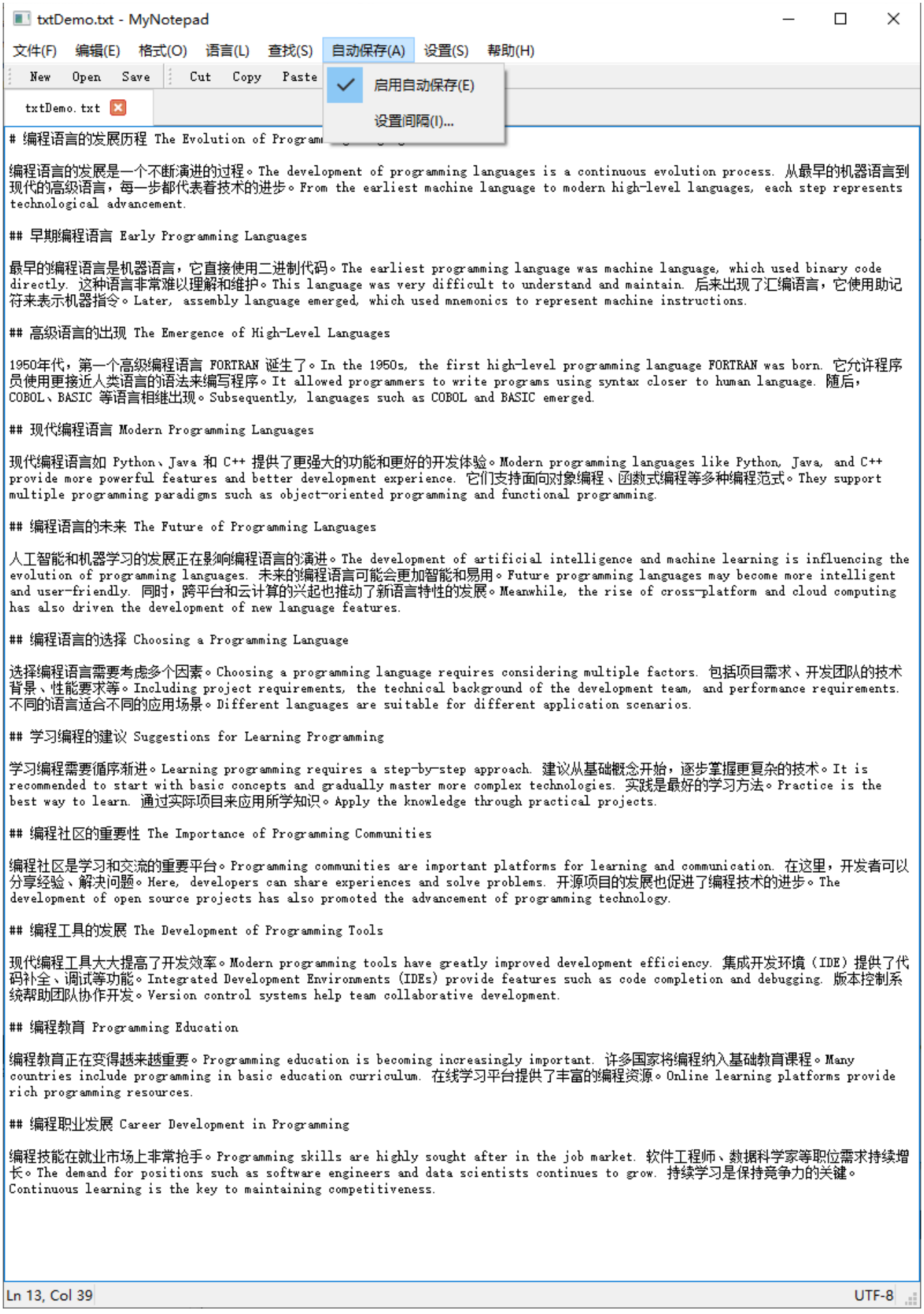
编程职业发展 Career Development in Programming

编程技能在就业市场上非常抢手。Programming skills are highly sought after in the job market. 软件工程师、数据科学家等职位需求持续增长。The demand for positions such as software engineers and data scientists continues to grow. 持续学习是保持竞争力的关键。Continuous learning is the key to maintaining competitiveness.

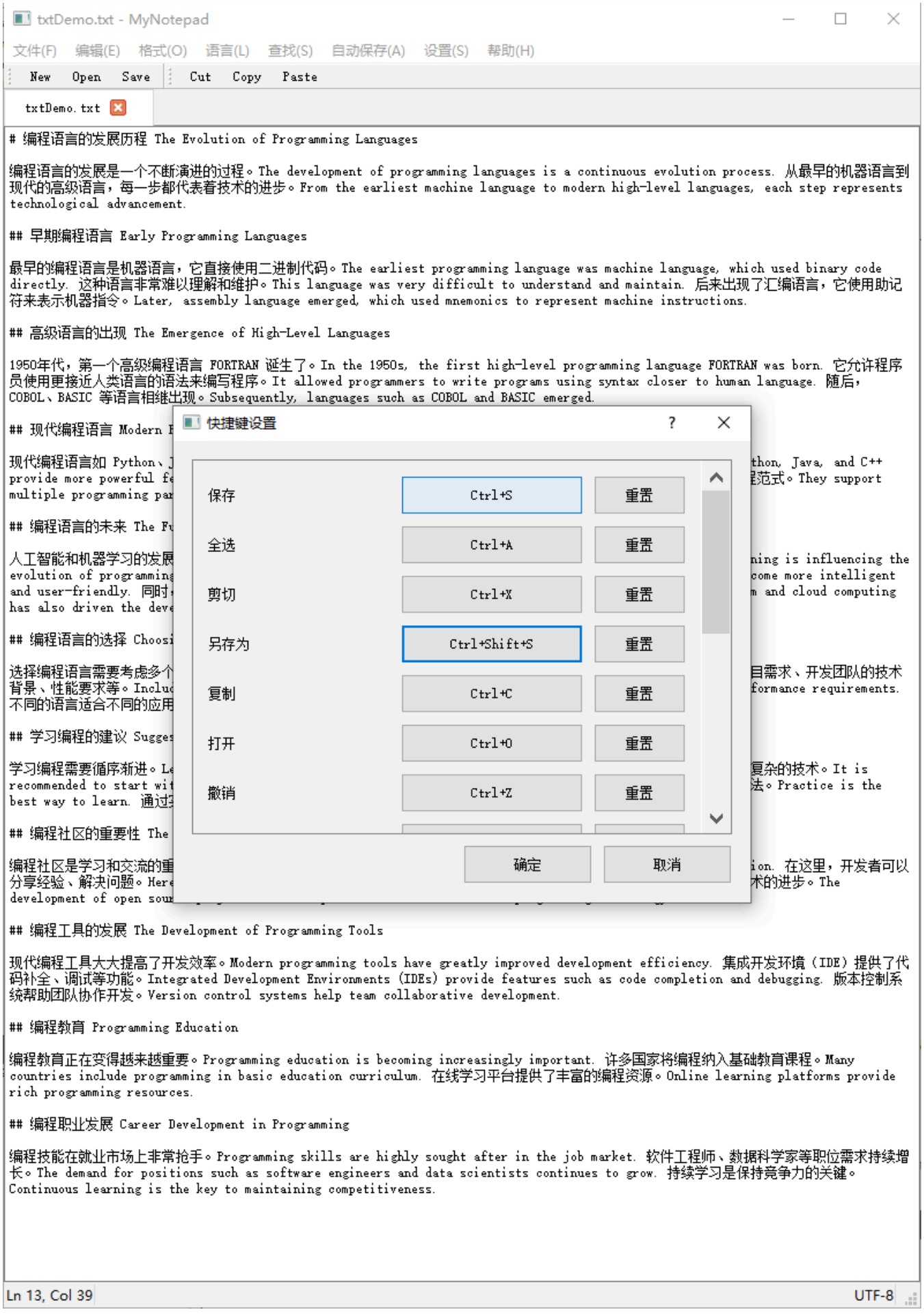

txtDemo.txt - MyNotepad
文件(F) 编辑(E) 格式(O) 语言(L) 查找(S) 自动保存(A) 设置(S) 帮助(H)
New Open Save Cut Copy Paste
txtDemo.txt
编程语言的发展历程 The Evolution of Programming Languages
编程语言的发展是一个不断演进的过程。The development of programming languages is a continuous evolution process. 从最早的机器语言到现代的高级语言，每一步都代表着技术的进步。From the earliest machine language to modern high-level languages, each step represents technological advancement.
早期编程语言 Early Programming Languages
最早的编程语言是机器语言，它直接使用二进制代码。The earliest programming language was machine language, which used binary code directly. 这种语言非常难以理解和维护。This language was very difficult to understand and maintain. 后来出现了汇编语言，它使用助记符来表示机器指令。Later, assembly language emerged, which used mnemonics to represent machine instructions.
高级语言的出现 The Emergence of High-Level Languages
1950年代，第一个高级编程语言 FORTRAN 诞生了。In the 1950s, the first high-level programming language FORTRAN was born. 它允许程序员使用更接近人类语言的语法来编写程序。It allowed programmers to write programs using syntax closer to human language. 随后，COBOL、BASIC 等语言相继出现。Subsequently, languages such as COBOL and BASIC emerged.
现代编程语言 Modern Programming Languages
现代编程语言如 Python、Java 和 C++ 提供了更强大的功能和更好的开发体验。Modern programming languages like Python, Java, and C++ provide more powerful features and better development experience. 它们支持面向对象编程、函数式编程等多种编程范式。They support multiple programming pa
编程语言的未来 The F
人工智能和机器学习的发展
ing is influencing the evolution of programmin
ome more intelligent and user-friendly. 同时
and cloud computing has also driven the dev
编程语言的选择 Choos
选择编程语言需要考虑多个
自需求、开发团队的技术背景、性能要求等。Inclu
formance requirements. 不同的语言适合不同的应用
学习编程的建议 Sugge
学习编程需要循序渐进。Learning programming requires a step-by-step approach. 建议从基础概念开始，逐步掌握更复杂的技术。It is recommended to start with basic concepts and gradually master more complex technologies. 实践是最好的学习方法。Practice is the best way to learn. 通过实际项目来应用所学知识。Apply the knowledge through practical projects.
编程社区的重要性 The Importance of Programming Communities
编程社区是学习和交流的重要平台。Programming communities are important platforms for learning and communication. 在这里，开发者可以分享经验、解决问题。Here, developers can share experiences and solve problems. 开源项目的发展也促进了编程技术的进步。The development of open source projects has also promoted the advancement of programming technology.
编程工具的发展 The Development of Programming Tools
现代编程工具大大提高了开发效率。Modern programming tools have greatly improved development efficiency. 集成开发环境（IDE）提供了代码补全、调试等功能。Integrated Development Environments (IDEs) provide features such as code completion and debugging. 版本控制系统帮助团队协作开发。Version control systems help team collaborative development.
编程教育 Programming Education
编程教育正在变得越来越重要。Programming education is becoming increasingly important. 许多国家将编程纳入基础教育课程。Many countries include programming in basic education curriculum. 在线学习平台提供了丰富的编程资源。Online learning platforms provide rich programming resources.
编程职业发展 Career Development in Programming
编程技能在就业市场上非常抢手。Programming skills are highly sought after in the job market. 软件工程师、数据科学家等职位需求持续增长。The demand for positions such as software engineers and data scientists continues to grow. 持续学习是保持竞争力的关键。Continuous learning is the key to maintaining competitiveness.
替换
Find: of
替换为: OF
区分大小写
整词
查找下一个
替换
替换所有
关闭
替换 14 次。
Ln 17, Col 25
UTF-8


txtDemo.txt * * - MyNotepad
文件(F) 编辑(E) 格式(O) 语言(L) 查找(S) 自动保存(A) 设置(S) 帮助(H)
New Open Save Cut Copy Paste
txtDemo.txt *
编程语言的发展历程 The Evolution OF Programming Languages
编程语言的发展是一个不断演进的过程。The development OF programming languages is a continuous evolution process. 从最早的机器语言到现代的高级语言，每一步都代表着技术的进步。From the earliest machine language to modern high-level languages, each step represents technological advancement.
早期编程语言 Early Programming Languages
最早的编程语言是机器语言，它直接使用二进制代码。The earliest programming language was machine language, which used binary code directly. 这种语言非常难以理解和维护。This language was very difficult to understand and maintain. 后来出现了汇编语言，它使用助记符来表示机器指令。Later, assembly language emerged, which used mnemonics to represent machine instructions.
高级语言的出现 The Emergence OF High-Level Languages
1950年代，第一个高级编程语言 FORTRAN 诞生了。In the 1950s, the first high-level programming language FORTRAN was born. 它允许程序员使用更接近人类语言的语法来编写程序。It allowed programmers to write programs using syntax closer to human language. 随后，COBOL、BASIC 等语言相继出现。Subsequently, languages such as COBOL and BASIC emerged.
现代编程语言 Modern Programming Languages
现代编程语言如 Python、Java 和 C++ 提供了更强大的功能和更好的开发体验。Modern programming languages like Python, Java, and C++ provide more powerful features and better development experience. 它们支持面向对象编程、函数式编程等多种编程范式。They support multiple programming pa
替换
Find: of
替换为: OF
区分大小写
整词
查找下一个
替换
替换所有
关闭
替换 13 次。
编程语言的未来 The F
人工智能和机器学习的发展
evolution OF programmin
and user-friendly. 同时
has also driven the dev
编程语言的选择 Choos
选择编程语言需要考虑多个
背景、性能要求等。Inclu
不同的语言适合不同的应用
学习编程的建议 Sugge
学习编程需要循序渐进。Learning programming requires a step-by-step approach. 建议从基础概念开始，逐步掌握更复杂的技术。It is recommended to start with basic concepts and gradually master more complex technologies. 实践是最好的学习方法。Practice is the best way to learn. 通过实际项目来应用所学知识。Apply the knowledge through practical projects.
编程社区的重要性 The Importance OF Programming Communities
编程社区是学习和交流的重要平台。Programming communities are important platforms for learning and communication. 在这里，开发者可以分享经验、解决问题。Here, developers can share experiences and solve problems. 开源项目的发展也促进了编程技术的进步。The development OF open source projects has also promoted the advancement OF programming technology.
编程工具的发展 The Development OF Programming Tools
现代编程工具大大提高了开发效率。Modern programming tools have greatly improved development efficiency. 集成开发环境（IDE）提供了代码补全、调试等功能。Integrated Development Environments (IDEs) provide features such as code completion and debugging. 版本控制系统帮助团队协作开发。Version control systems help team collaborative development.
编程教育 Programming Education
编程教育正在变得越来越重要。Programming education is becoming increasingly important. 许多国家将编程纳入基础教育课程。Many countries include programming in basic education curriculum. 在线学习平台提供了丰富的编程资源。Online learning platforms provide rich programming resources.
编程职业发展 Career Development in Programming
编程技能在就业市场上非常抢手。Programming skills are highly sought after in the job market. 软件工程师、数据科学家等职位需求持续增长。The demand for positions such as software engineers and data scientists continues to grow. 持续学习是保持竞争力的关键。Continuous learning is the key to maintaining competitiveness.
Ln 33, Col 30
UTF-8

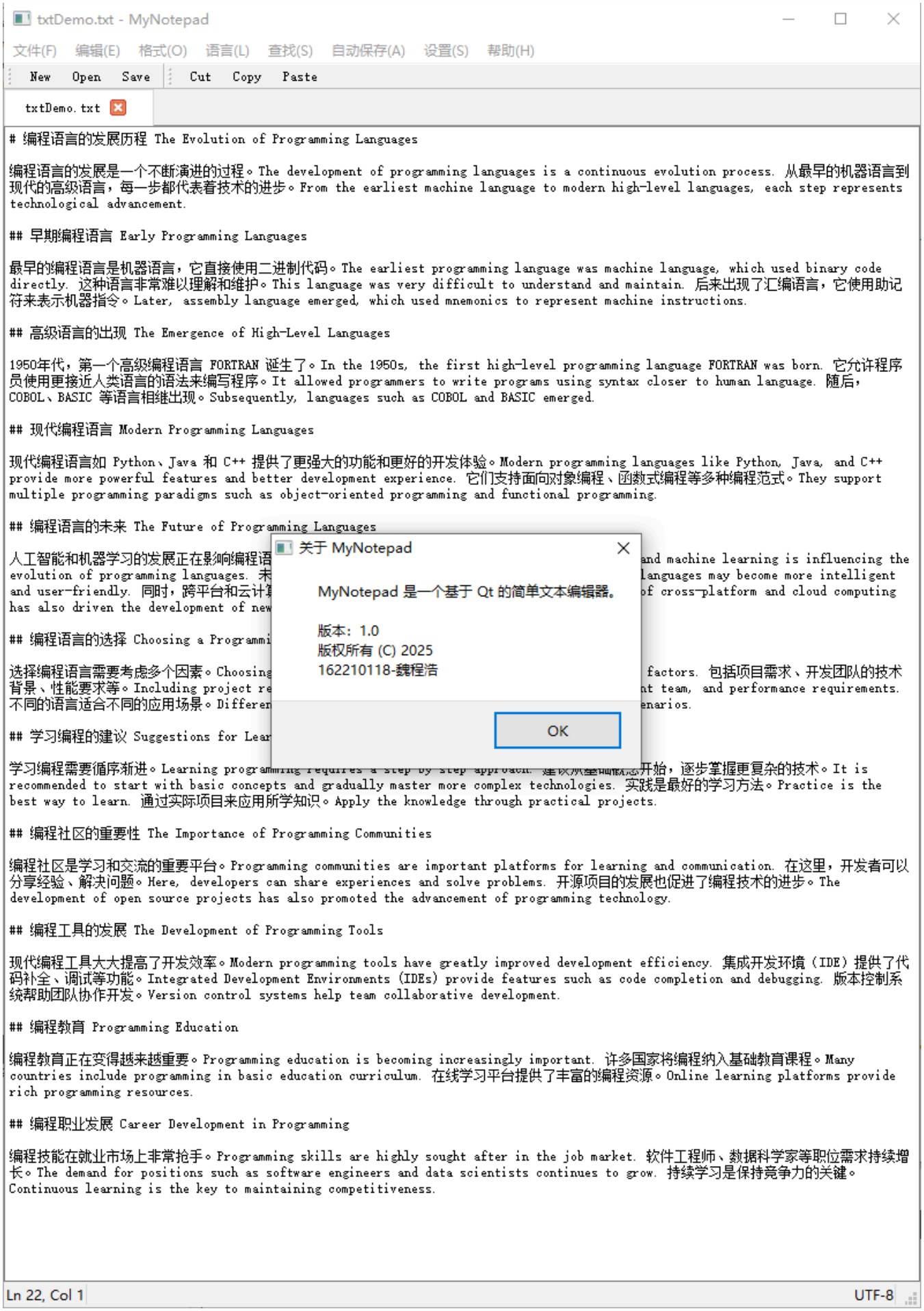
查找&替换


txtDemo.txt - MyNotepad
文件(F) 编辑(E) 格式(O) 语言(L) 查找(S) 自动保存(A) 设置(S) 帮助(H)
New Open Save Cut Copy Paste
启用自动保存(E)
设置间隔(I)...
txtDemo.txt
编程语言的发展历程 The Evolution of Programm
编程语言的发展是一个不断演进的过程。The development of programming languages is a continuous evolution process. 从最早的机器语言到现代的高级语言，每一步都代表着技术的进步。From the earliest machine language to modern high-level languages, each step represents technological advancement.
早期编程语言 Early Programming Languages
最早的编程语言是机器语言，它直接使用二进制代码。The earliest programming language was machine language, which used binary code directly. 这种语言非常难以理解和维护。This language was very difficult to understand and maintain. 后来出现了汇编语言，它使用助记符来表示机器指令。Later, assembly language emerged, which used mnemonics to represent machine instructions.
高级语言的出现 The Emergence of High-Level Languages
1950年代，第一个高级编程语言 FORTRAN 诞生了。In the 1950s, the first high-level programming language FORTRAN was born. 它允许程序员使用更接近人类语言的语法来编写程序。It allowed programmers to write programs using syntax closer to human language. 随后，COBOL、BASIC 等语言相继出现。Subsequently, languages such as COBOL and BASIC emerged.
现代编程语言 Modern Programming Languages
现代编程语言如 Python、Java 和 C++ 提供了更强大的功能和更好的开发体验。Modern programming languages like Python, Java, and C++ provide more powerful features and better development experience. 它们支持面向对象编程、函数式编程等多种编程范式。They support multiple programming paradigms such as object-oriented programming and functional programming.
编程语言的未来 The Future of Programming Languages
人工智能和机器学习的发展正在影响编程语言的演进。The development of artificial intelligence and machine learning is influencing the evolution of programming languages. 未来的编程语言可能会更加智能和易用。Future programming languages may become more intelligent and user-friendly. 同时，跨平台和云计算的兴起也推动了新语言特性的发展。Meanwhile, the rise of cross-platform and cloud computing has also driven the development of new language features.
编程语言的选择 Choosing a Programming Language
选择编程语言需要考虑多个因素。Choosing a programming language requires considering multiple factors. 包括项目需求、开发团队的技术背景、性能要求等。Including project requirements, the technical background of the development team, and performance requirements. 不同的语言适合不同的应用场景。Different languages are suitable for different application scenarios.
学习编程的建议 Suggestions for Learning Programming
学习编程需要循序渐进。Learning programming requires a step-by-step approach. 建议从基础概念开始，逐步掌握更复杂的技术。It is recommended to start with basic concepts and gradually master more complex technologies. 实践是最好的学习方法。Practice is the best way to learn. 通过实际项目来应用所学知识。Apply the knowledge through practical projects.
编程社区的重要性 The Importance of Programming Communities
编程社区是学习和交流的重要平台。Programming communities are important platforms for learning and communication. 在这里，开发者可以分享经验、解决问题。Here, developers can share experiences and solve problems. 开源项目的发展也促进了编程技术的进步。The development of open source projects has also promoted the advancement of programming technology.
编程工具的发展 The Development of Programming Tools
现代编程工具大大提高了开发效率。Modern programming tools have greatly improved development efficiency. 集成开发环境（IDE）提供了代码补全、调试等功能。Integrated Development Environments (IDEs) provide features such as code completion and debugging. 版本控制系统帮助团队协作开发。Version control systems help team collaborative development.
编程教育 Programming Education
编程教育正在变得越来越重要。Programming education is becoming increasingly important. 许多国家将编程纳入基础教育课程。Many countries include programming in basic education curriculum. 在线学习平台提供了丰富的编程资源。Online learning platforms provide rich programming resources.
编程职业发展 Career Development in Programming
编程技能在就业市场上非常抢手。Programming skills are highly sought after in the job market. 软件工程师、数据科学家等职位需求持续增长。The demand for positions such as software engineers and data scientists continues to grow. 持续学习是保持竞争力的关键。Continuous learning is the key to maintaining competitiveness.
Ln 13, Col 39
UTF-8

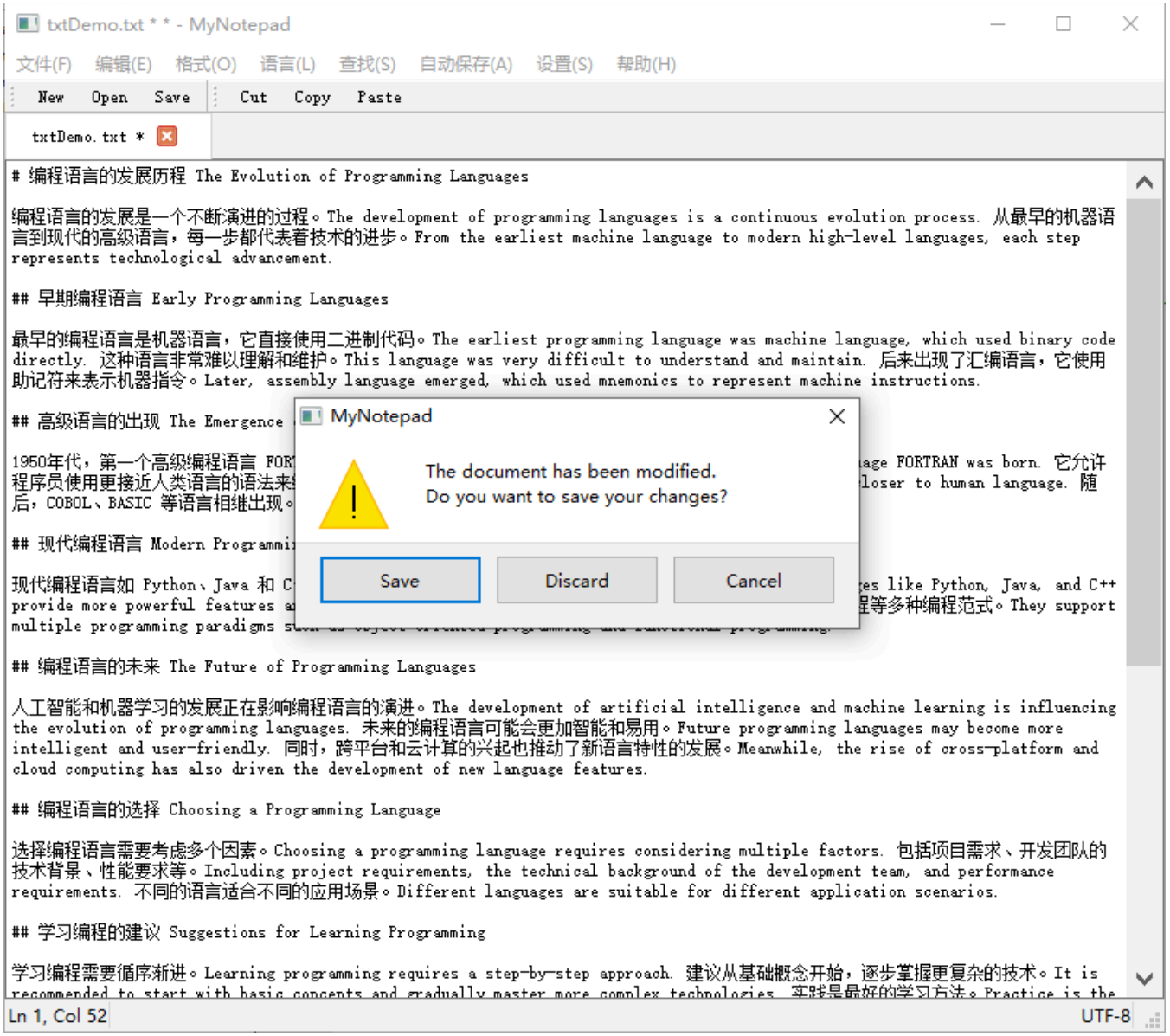
自动保存


txtDemo.txt - MyNotepad
文件(F) 编辑(E) 格式(O) 语言(L) 查找(S) 自动保存(A) 设置(S) 帮助(H)
New Open Save Cut Copy Paste
txtDemo.txt
编程语言的发展历程 The Evolution of Programming Languages
编程语言的发展是一个不断演进的过程。The development of programming languages is a continuous evolution process. 从最早的机器语言到现代的高级语言，每一步都代表着技术的进步。From the earliest machine language to modern high-level languages, each step represents technological advancement.
早期编程语言 Early Programming Languages
最早的编程语言是机器语言，它直接使用二进制代码。The earliest programming language was machine language, which used binary code directly. 这种语言非常难以理解和维护。This language was very difficult to understand and maintain. 后来出现了汇编语言，它使用助记符来表示机器指令。Later, assembly language emerged, which used mnemonics to represent machine instructions.
高级语言的出现 The Emergence of High-Level Languages
1950年代，第一个高级编程语言 FORTRAN 诞生了。In the 1950s, the first high-level programming language FORTRAN was born. 它允许程序员使用更接近人类语言的语法来编写程序。It allowed programmers to write programs using syntax closer to human language. 随后，COBOL、BASIC 等语言相继出现。Subsequently, languages such as COBOL and BASIC emerged.
现代编程语言 Modern
现代编程语言如 Python、J
provide more powerful fe
multiple programming par
编程语言的未来 The Fu
人工智能和机器学习的发展
evolution of programming
and user-friendly. 同时
has also driven the deve
编程语言的选择 Choosi
选择编程语言需要考虑多个
背景、性能要求等。Includ
不同的语言适合不同的应用
学习编程的建议 Sugges
学习编程需要循序渐进。Le
recommended to start wit
best way to learn. 通过
编程社区的重要性 The
编程社区是学习和交流的重
分享经验、解决问题。Here
development of open sour
编程工具的发展 The Development of Programming Tools
现代编程工具大大提高了开发效率。Modern programming tools have greatly improved development efficiency. 集成开发环境（IDE）提供了代码补全、调试等功能。Integrated Development Environments (IDEs) provide features such as code completion and debugging. 版本控制系统帮助团队协作开发。Version control systems help team collaborative development.
编程教育 Programming Education
编程教育正在变得越来越重要。Programming education is becoming increasingly important. 许多国家将编程纳入基础教育课程。Many countries include programming in basic education curriculum. 在线学习平台提供了丰富的编程资源。Online learning platforms provide rich programming resources.
编程职业发展 Career Development in Programming
编程技能在就业市场上非常抢手。Programming skills are highly sought after in the job market. 软件工程师、数据科学家等职位需求持续增长。The demand for positions such as software engineers and data scientists continues to grow. 持续学习是保持竞争力的关键。Continuous learning is the key to maintaining competitiveness.
Ln 13, Col 39
UTF-8
thon, Java, and C++
程范式。They support
ning is influencing the
come more intelligent
m and cloud computing
目需求、开发团队的技术
formance requirements.
复杂的技术。It is
法。Practice is the
ion. 在这里，开发者可以
术的进步。The
快捷键设置
保存 Ctrl+S 重置
全选 Ctrl+A 重置
剪切 Ctrl+X 重置
另存为 Ctrl+Shift+S 重置
复制 Ctrl+C 重置
打开 Ctrl+O 重置
撤销 Ctrl+Z 重置
确定
取消

快捷键自定义


txtDemo.txt - MyNotepad
文件(F) 编辑(E) 格式(O) 语言(L) 查找(S) 自动保存(A) 设置(S) 帮助(H)
New Open Save Cut Copy Paste
txtDemo.txt
编程语言的发展历程 The Evolution of Programming Languages
编程语言的发展是一个不断演进的过程。The development of programming languages is a continuous evolution process. 从最早的机器语言到现代的高级语言，每一步都代表着技术的进步。From the earliest machine language to modern high-level languages, each step represents technological advancement.
早期编程语言 Early Programming Languages
最早的编程语言是机器语言，它直接使用二进制代码。The earliest programming language was machine language, which used binary code directly. 这种语言非常难以理解和维护。This language was very difficult to understand and maintain. 后来出现了汇编语言，它使用助记符来表示机器指令。Later, assembly language emerged, which used mnemonics to represent machine instructions.
高级语言的出现 The Emergence of High-Level Languages
1950年代，第一个高级编程语言 FORTRAN 诞生了。In the 1950s, the first high-level programming language FORTRAN was born. 它允许程序员使用更接近人类语言的语法来编写程序。It allowed programmers to write programs using syntax closer to human language. 随后，COBOL、BASIC 等语言相继出现。Subsequently, languages such as COBOL and BASIC emerged.
现代编程语言 Modern Programming Languages
现代编程语言如 Python、Java 和 C++ 提供了更强大的功能和更好的开发体验。Modern programming languages like Python, Java, and C++ provide more powerful features and better development experience. 它们支持面向对象编程、函数式编程等多种编程范式。They support multiple programming paradigms such as object-oriented programming and functional programming.
编程语言的未来 The Future of Programming Languages
人工智能和机器学习的发展正在影响编程语
and machine learning is influencing the evolution of programming languages. 未
languages may become more intelligent and user-friendly. 同时，跨平台和云计算
of cross-platform and cloud computing has also driven the development of new
编程语言的选择 Choosing a Programmi
选择编程语言需要考虑多个因素。Choosing
factors. 包括项目需求、开发团队的技术背景、性能要求等。Including project re
nt team, and performance requirements. 不同的语言适合不同的应用场景。Differen
enarios.
学习编程的建议 Suggestions for Lear
学习编程需要循序渐进。Learning programming requires a step by step approach. 建议从基础概念开始，逐步掌握更复杂的技术。It is recommended to start with basic concepts and gradually master more complex technologies. 实践是最好的学习方法。Practice is the best way to learn. 通过实际项目来应用所学知识。Apply the knowledge through practical projects.
编程社区的重要性 The Importance of Programming Communities
编程社区是学习和交流的重要平台。Programming communities are important platforms for learning and communication. 在这里，开发者可以分享经验、解决问题。Here, developers can share experiences and solve problems. 开源项目的发展也促进了编程技术的进步。The development of open source projects has also promoted the advancement of programming technology.
编程工具的发展 The Development of Programming Tools
现代编程工具大大提高了开发效率。Modern programming tools have greatly improved development efficiency. 集成开发环境（IDE）提供了代码补全、调试等功能。Integrated Development Environments (IDEs) provide features such as code completion and debugging. 版本控制系统帮助团队协作开发。Version control systems help team collaborative development.
编程教育 Programming Education
编程教育正在变得越来越重要。Programming education is becoming increasingly important. 许多国家将编程纳入基础教育课程。Many countries include programming in basic education curriculum. 在线学习平台提供了丰富的编程资源。Online learning platforms provide rich programming resources.
编程职业发展 Career Development in Programming
编程技能在就业市场上非常抢手。Programming skills are highly sought after in the job market. 软件工程师、数据科学家等职位需求持续增长。The demand for positions such as software engineers and data scientists continues to grow. 持续学习是保持竞争力的关键。Continuous learning is the key to maintaining competitiveness.
Ln 22, Col 1
UTF-8
关于 MyNotepad
MyNotepad 是一个基于 Qt 的简单文本编辑器。
版本：1.0
版权所有 (C) 2025
162210118-魏程浩
OK

About窗口



警告弹窗

7. 未来改进方向

1. 添加更多编程语言的语法高亮支持
2. 实现文件比较功能
3. 添加代码折叠功能
4. 支持插件系统
5. 优化大文件处理性能
6. 版本控制：集成Git基础操作
7. AI 集成：接入开源NLP 模型实现智能补全

8. 总结

本项目实现了一个功能完善的现代化记事本应用程序，通过模块化设计和Qt框架的强大功能，实现了丰富的文本编辑功能。项目代码结构清晰，易于维护和扩展，为用户提供了良好的使用体验。

在项目的编写过程中，我学到了很多：

首先是技术能力的提升。通过项目学习，我掌握了Qt框架的基本使用方法和开发流程，深入理解了面向对象编程在实际项目中的应用，还提高了自身代码组织和架构设计能力。

其次是项目经验的积累。我学会了如何规划和实现一个完整的桌面应用程序，理解了用户界面设计的重要性和基本原则，掌握了文件操作、事件处理等核心功能的实现方法。

同时我的问题解决能力也得到了提高，学会了查阅文档和寻找解决方案的方法。在未来我将进一步深入学习Qt框架的高级特性，探索更多实用的功能模块和优化方案以持续改进项目，提升用户体验。

这次实验不仅让我在技术上有所收获，更重要的是培养了我解决实际问题的能力和项目开发经验。这些经验对我未来的学习和工作都将产生积极的影响。