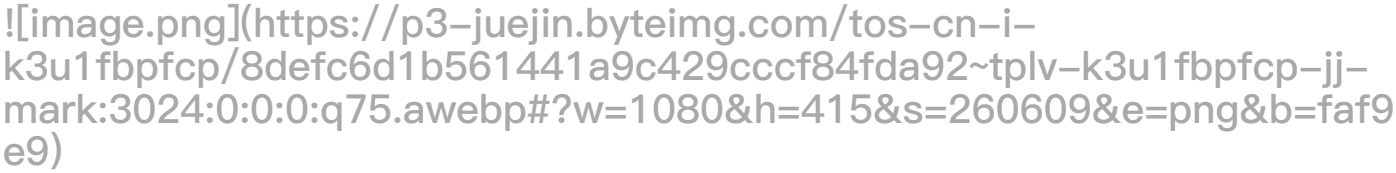
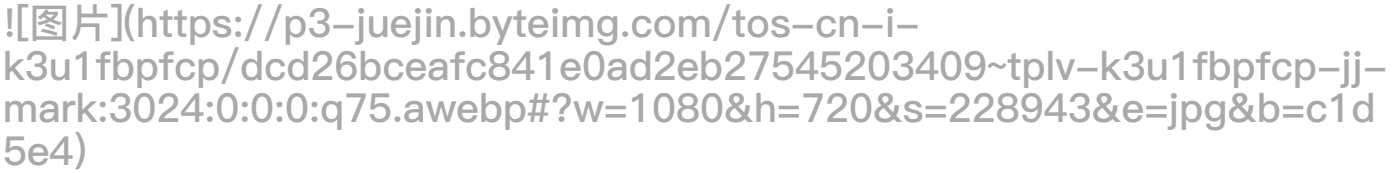
「豆包Marscode体验官」我在Marscode用了3天，转行成为Python程序员

我正在参加「豆包MarsCode初体验」征文活动

豆包MarsCode简介



6 月 26 日，字节跳动在北京发布了基于豆包大模型打造的智能开发工具 – 豆包MarsCode ，面向国内开发者免费开放。本场发布会以“用 AI 激发创造”为主题，在草地露营的轻松氛围中发布了豆包MarsCode 并介绍了其主要功能，同时发布开发者及社区共创计划，吸引了众多业界人士、开发者和科技爱好者的。

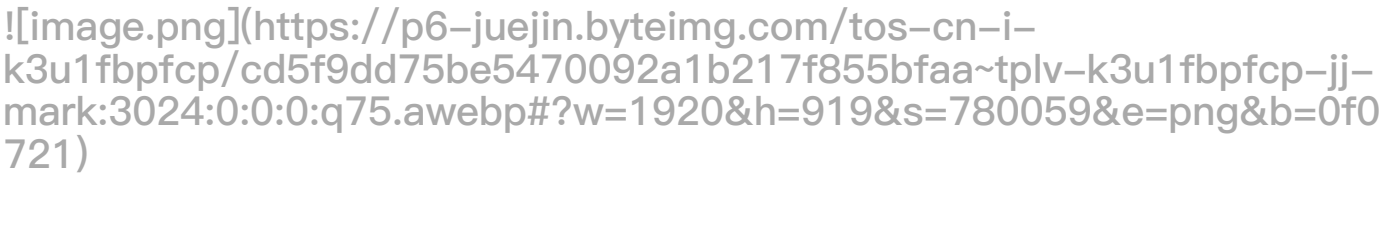


豆包MarsCode——用 AI 激发创造


字节跳动开发者服务团队、豆包MarsCode 负责人 李东江字节跳动开发者服务团队、豆包MarsCode 负责人李东江在正式发布之前，分享了一些对 AI 时代开发工具演进趋势的思考。进入AI 时代，大语言模型在编程语言方面具备强大的优势和潜力，相比起复杂的自然语言，编程语言是更加简洁，更加严谨，更加可预测的。关于“应当如何构建一款 AI 时代的开发者工具”的命题，豆包 MarsCode 团队会更多的从如何赋能开发者的角度入手。李东江认为 **AI 不是替代开发者的“竞争者”，而是开发者的“好帮手”** ，团队更希望打造一款软件，能够助力提升开发者工作效率，让开发者有更多精力和时间用于思考和创造，这也就是为什么发布会的主题是“用 AI 激发创造”。李东江提到，新的模型、新的算力、新的产品、新的技术每天都在出现，无论是产品还是技术，一切都还处在早期，都在快速更新迭代。在 AI 技术驱动下，一定会衍生出下一代的开发工具。而豆包MarsCode 团队，希望与开发者共同探索、建设，一起打造 AI 时代的新的开发者工具。
豆包MarsCode是一款基于豆包大模型的智能开发工具，提供代码补全、生成、解释等功能，旨在提高开发效率和代码质量。

产品定位

豆包MarsCode是一款智能开发工具，旨在通过AI技术提高开发者的创造力和效率。它面向软件开发者、编程爱好者以及技术团队，提供一站式的开发体验，简化开发流程。豆包MarsCode的定位明确，旨在通过AI技术解决开发过程中的痛点，提高开发效率和质量，同时促进技术社区的交流和分享。



用户体验

豆包MarsCode提供AI驱动的云端IDE版和支持VS code和JetBrains等的扩展版，提供一站式开发体验。它具备代码补全、生成、解释功能，支持AI问答和BUG修复功能。豆包MarsCode的用户体验设计考虑到了不同开发者的需求，无论是云端还是本地环境，都能提供高效便捷的编程体验。

豆包MarsCode功能

AI代码生成

豆包MarsCode能够自动生成代码，提高开发效率。它支持多种编程语言，满足不同开发需求。AI代码生成功能是豆包MarsCode的核心功能之一，能够大大缩短开发周期，提高代码质量，同时支持多语言，适应性强。

智能错误检测

豆包MarsCode能够实时检测代码错误，减少调试时间。它提供代码优化建议，提升代码质量。智能错误检测功能能够及时发现并解决代码中的问题，减少开发者的调试时间，提高代码的健壮性。

社区交流平台

豆包MarsCode构建开发者社区，促进技术交流与分享。它提供一键部署功能，提供安全可靠的云开发环境。社区交流平台不仅促进了开发者之间的交流和合作，还提供了一键部署功能，简化了开发流程，提高了开发效率。

豆包MarsCode使用教程

注册与登录

[访问MarsCode官网并注册账号](http://cxyroad.com/”https://www.marscode.cn/home”)，登录后选择相应的开发环境。使用AI代码生成功能，输入开发需求，系统将自动生成代码。豆包MarsCode的使用教程简单明了，即使是编程新手也能快速上手，通过直观的界面和简单的操作，快速实现代码的生成和优化。

功能使用

豆包MarsCode提供代码补全、代码生成、代码解释、代码注释生成、单测生成、缺陷修复、AI问答等功能。通过实际用例演示，展示了豆包MarsCode在日常工作中的优秀落地能力。豆包MarsCode的功能丰富，能够满足开发者在不同阶

段的需求，通过实际用例的演示，进一步证明了其在提高开发效率和代码质量方面的有效性。

豆包MarsCode技术原理

AI技术集成

豆包MarsCode集成了豆包大模型的AI技术，提供智能开发工具。它通过AI技术自动生成代码，实时检测代码错误并提供代码优化方案。豆包MarsCode的技术原理体现了AI技术在软件开发领域的应用，通过集成先进的AI技术，实现了自动化代码生成和错误检测，提高了开发效率和质量。

代码理解与生成

豆包MarsCode能够理解代码上下文并自动生成相应的后续代码片段。它支持通过自然语言描述需求，获取代码推荐。豆包MarsCode的代码理解与生成能力，使得开发者能够更高效地编写代码，减少了手动编写代码的工作量，同时提高了代码的质量和可维护性。

豆包MarsCode安全性

数据安全与隐私保护

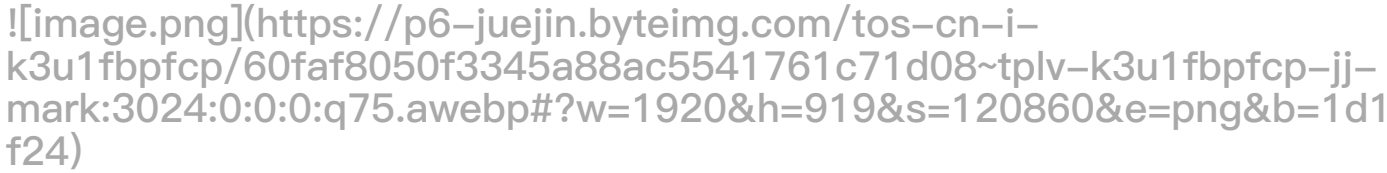
豆包MarsCode在设计和开发过程中充分考虑了数据安全和隐私保护。它提供安全的云开发环境，确保用户数据的安全存储和传输。豆包MarsCode的安全性设计，保护了用户的隐私和数据安全，让用户在使用过程中无后顾之忧，增强了用户对产品的信任度。

豆包MarsCode是一款集成了AI技术的智能开发工具，通过自动生成代码、实时检测错误、提供代码优化建议等功能，显著提高了开发效率和代码质量。同时，豆包MarsCode还提供了社区交流平台和一键部署功能，促进了开发者之间的交流和合作。在技术原理上，豆包MarsCode通过集成豆包大模型的AI技术，实现了对代码的深度理解和生成。此外，豆包MarsCode在数据安全和隐私保护方面也做了充分的考虑，确保用户数据的安全存储和传输。总的来说，豆包MarsCode是一款功能强大、安全可靠、易于使用的智能开发工具，值得广大开发者尝试和使用。

豆包MarsCode项目实战

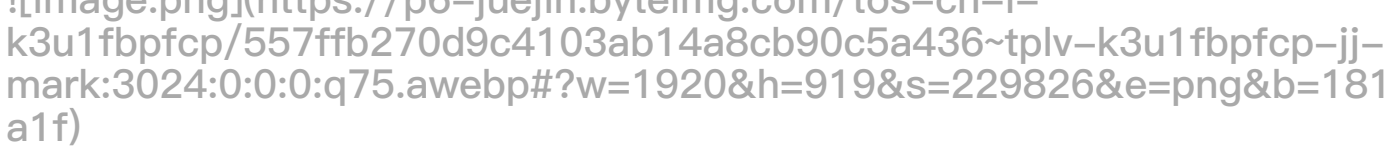
Java程序员转行Python学习之路

俗话说：工欲善其事，必先利其器。在历史的长河中，新手程序员最大的痛点之一就是搭建开发环境。先就是今天，如果你没有 VS Code，甚至也没有其他 IDE，那么也没有关系。豆包 MarsCode 提供免费的云 IDE，无需下载和安装，直接在浏览器使用，并且内置数十款开发模板，还可以拉取 GitHub 仓库，用起来很方便。

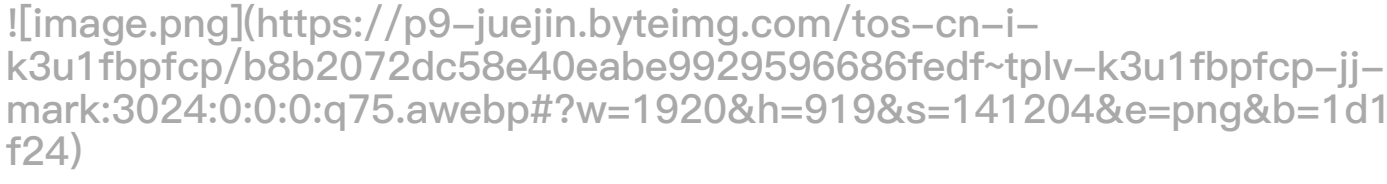


豆包 MarsCode IDE + AI编码助手，助力你3天精通Python语言

新建工程

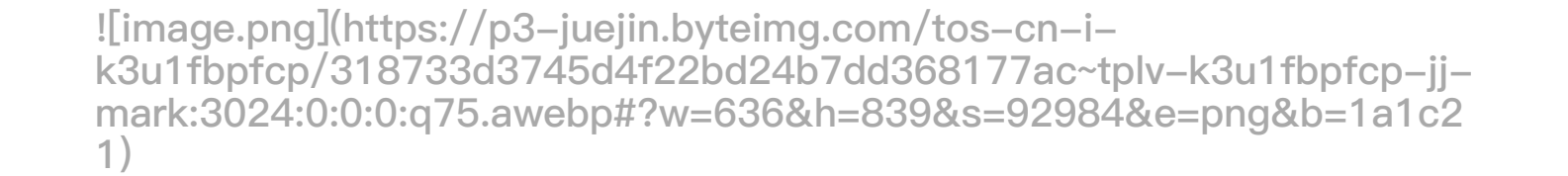


创建Python工程模板



请教AI编码助手

问：我是一名Java程序员，我现在从0基础学习Python，请提供给我一个学习大纲。

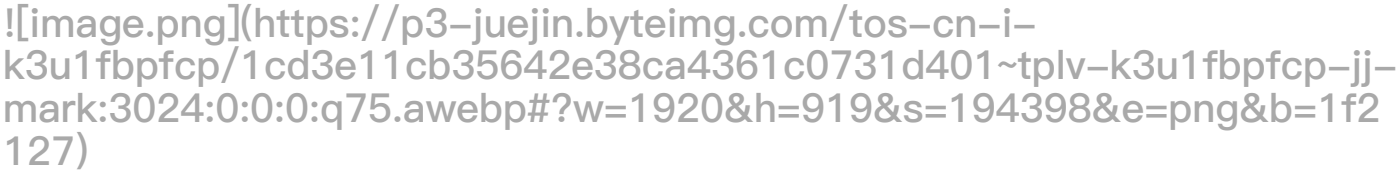


> 全国计算机等级考试二级 Python 语言程序设计考试大纲 2023 版为依据
>
>
> **一、Python 语言基础**
>
>
> * Python 简介
> * Python 开发环境的搭建
> * Python 基本语法
> * Python 中的数据类型（数字、字符串，列表，元组，字典）
> * Python 中的控制流语句（条件语句，循环语句）
> * Python 中的函数
> * Python 模块和包的使用
>
>
> **二、Python 进阶**
>
>
> * Python 中的面向对象编程
> * Python 异常处理
> * Python 文件操作
> * Python 数据库操作（以 SQLite 为例）
> * Python 正则表达式
> * Python 多线程和多进程编程
> * Python 网络编程基础（UDP 和 TCP）
>
>
> **三、数据结构与算法**
>
>
> * 栈与队列
> * 链表
> * 树与二叉树
> * 排序算法（冒泡排序、插入排序、快速排序等）
> * 搜索算法（顺序查找、二分查找）
>
>
> **四、网络爬虫基础**
>
>
>

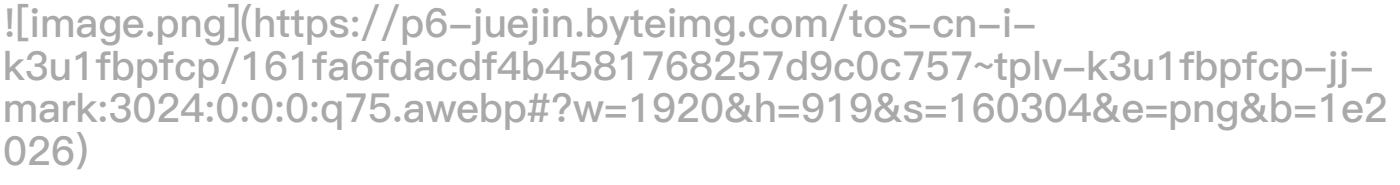
> * 网络爬虫的基本概念
> * requests 库的使用
> * BeautifulSoup 库的使用
> * 数据存储（CSV、JSON、MySQL）
> * 反爬虫策略及应对方法
>
>
> **五、数据预处理与数据可视化**
>
>
> * Numpy 和 Pandas 库的使用
> * 数据清洗和预处理
> * Matplotlib 和 Seaborn 库的使用
> * 数据可视化技巧
>
>
> **六、项目实践**
>
>
> * 选择一个适合的项目，如爬取某个网站的数据并进行分析和可视化。

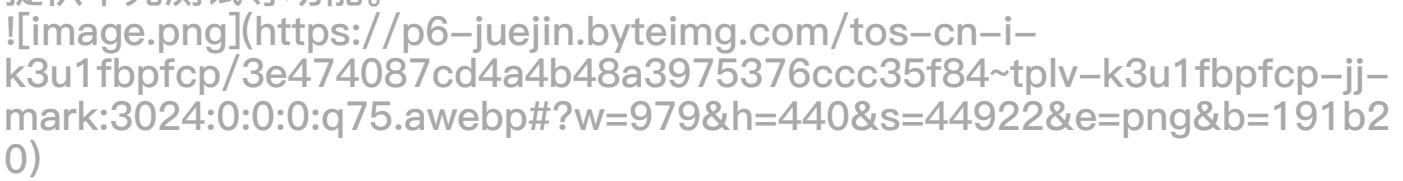
一、Python 语言基础

将代码示例，直接插入工程进行保存。

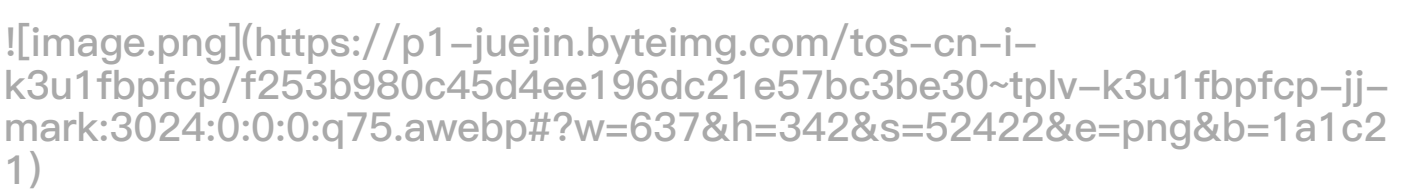


通过鼠标右键，进行代码运行

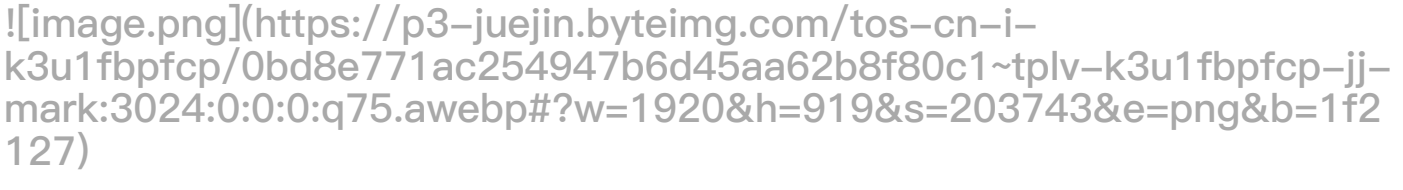


可以选择不懂的地方，与AI编码助手对话，解释其作用，生成代码注释，以及提供单元测试等功能。


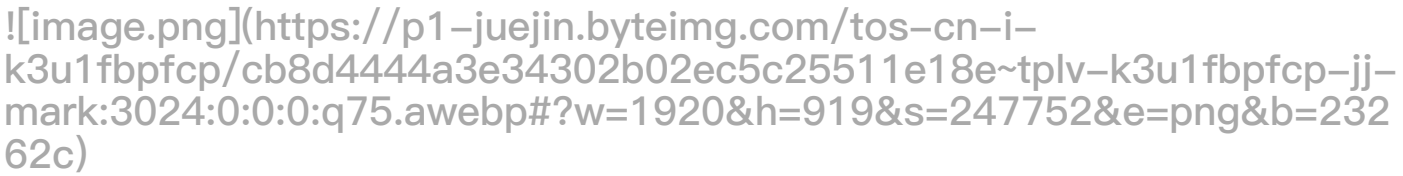
例如：解释



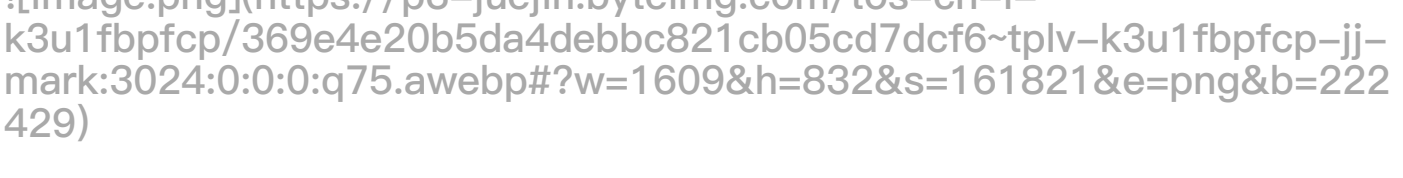
二、Python 进阶



使用AI助手对代码进行注释



注释后，自动比对，可根据需求



三、数据结构与算法

mark:3024:0:0:0:q75.awebp#?w=1920&h=919&s=200195&e=png&b=1e2026)

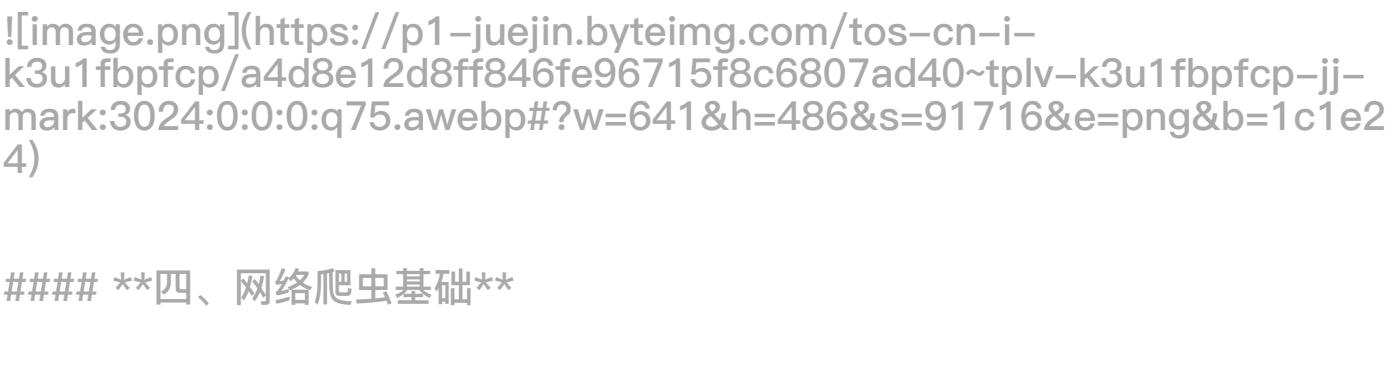
```
# 栈与队列

# 栈的实现
class Stack:
    def __init__(self):
        self.items = []

    def push(self, item):
        self.items.append(item)

    def pop(self):
        if self.is_empty():
            return None
        return self.items.pop()

    def is_empty(self):
        return len(self.items) == 0

    def peek(self):
        if self.is_empty():
            return None
        return self.items[-1]

# 队列的实现
class Queue:
    def __init__(self):
        self.items = []

    def enqueue(self, item):
        self.items.append(item)

    def dequeue(self):
        if self.is_empty():
            return None
        return self.items.pop(0)

    def is_empty(self):
        return len(self.items) == 0

    def peek(self):
        if self.is_empty():
            return None
```

```
        return self.items[0]

# 链表

# 单链表节点类
class ListNode:
    def __init__(self, value=0, next=None):
        self.value = value
        self.next = next

# 树与二叉树

# 二叉树节点类
class TreeNode:
    def __init__(self, value=0, left=None, right=None):
        self.value = value
        self.left = left
        self.right = right

# 排序算法

# 冒泡排序
def bubble_sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]

# 插入排序
def insertion_sort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i-1
        while j >= 0 and key < arr[j]:
            arr[j+1] = arr[j]
            j -= 1
        arr[j+1] = key

# 快速排序
def quick_sort(arr):
    if len(arr) <= 1:
        return arr
    else:
        pivot = arr[len(arr) // 2]
        left = [x for x in arr if x < pivot]
        middle = [x for x in arr if x == pivot]
        right = [x for x in arr if x > pivot]
```

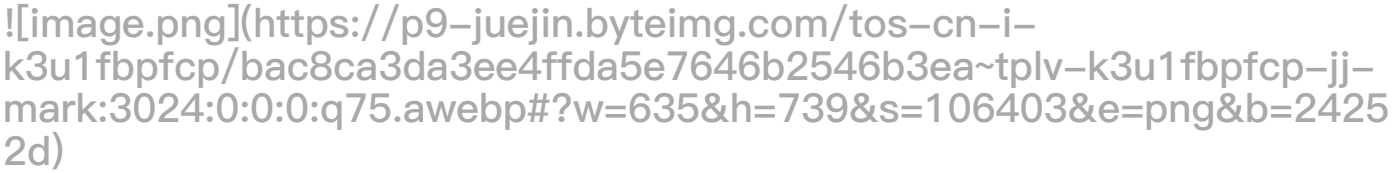
```
        return quick_sort(left) + middle + quick_sort(right)

# 搜索算法

# 顺序查找
def sequential_search(arr, target):
    for i in range(len(arr)):
        if arr[i] == target:
            return i
    return -1

# 二分查找
def binary_search(arr, target):
    low = 0
    high = len(arr) - 1
    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] < target:
            low = mid + 1
        else:
            high = mid - 1
    return -1

# 示例用法

stack = Stack()
stack.push(10)
stack.push(20)
stack.push(30)
print("栈顶元素：", stack.peek())
print("弹出栈顶元素：", stack.pop())
print("栈是否为空：", stack.is_empty())
queue = Queue()
queue.enqueue(10)
queue.enqueue(20)
queue.enqueue(30)
print("队列头部元素：", queue.peek())
print("取出队列头部元素：", queue.dequeue())
print("队列是否为空：", queue.is_empty())

head = ListNode(1)
node2 = ListNode(2)
node3 = ListNode(3)

head.next = node2
node2.next = node3
```

```
current = head
while current:
    print(current.value, end=" ")
    current = current.next

print()

root = TreeNode(1)
root.left = TreeNode(2)
root.right = TreeNode(3)

print("先序遍历")
pre_order_traversal(root)
print("中序遍历")
in_order_traversal(root)
print("后序遍历")
post_order_traversal(root)

numbers = [64, 34, 25, 12, 22, 11, 90]
print("排序前：", numbers)
bubble_sort(numbers)
print("排序后（冒泡排序）：", numbers)
insertion_sort(numbers)
print("排序后（插入排序）：", numbers)
quick_sort(numbers)
print("排序后（快速排序）：", numbers)

target = 22
result = sequential_search(numbers, target)
if result!= -1:
    print(f"元素 {target} 在列表中的索引位置是：{result}")
else:
    print(f"未找到元素 {target}")

numbers.sort()  # 对列表进行排序，以便于后续的二分查找
target = 22
result = binary_search(numbers, target)
if result!= -1:
    print(f"元素 {target} 在列表中的索引位置是：{result}")
else:
    print(f"未找到元素 {target}")
```

AI助手解释代码



四、网络爬虫基础



五、数据预处理与数据可视化

```
import numpy as np
import pandas as pd

data = {'Name': ['Alice', 'Bob', 'Charlie', 'David'],
        'Age': [25, 30, 35, 40],
        'City': ['New York', 'Los Angeles', 'Chicago', 'Houston']}
df = pd.DataFrame(data)

# Using Numpy to perform mathematical operations on DataFrame columns
np meanAge = np.mean(df['Age'])
print("The average age is:", meanAge)

# Using Pandas for data cleaning and preprocessing
df['Age'] = df['Age'].fillna(0)  # Fill Null values with 0
df['City'] = df['City'].str.upper()  # Convert city names to uppercase

print(df)
```

六、项目实践

根据你的需求，使用 Python 的 `requests` 库、`matplotlib` 库以及 `pandas` 库来实现你的项目。以下是项目流程的概述以及每个步骤的代码示例：

* 数据爬取：使用 `requests` 库可以发起 HTTP 请求。你可以编写代码来模拟浏览器的行为，发送请求到目标网站，并接收服务器的响应。

```
import requests

url = "http://example.com/"  # 目标网站的URL
response = requests.get(url)

if response.status_code == 200:
    # 处理响应数据
    pass
else:
    # 处理请求失败的情况
    pass
```

* 数据清洗：清洗数据通常涉及到处理缺失值、重复值、异常值以及数据标准化等任务。`pandas` 库提供了丰富的函数和方法来执行这类数据清洗操作。

```
import pandas as pd

# 假设有一个包含数据的 DataFrame
data = pd.DataFrame({
    'A': [1, 2, None, 4],
    'B': [5, 6, 7, 8],
    'C': ['a', 'b', 'c', 'd']
})

# 处理缺失值
data = data.dropna()  # 移除包含空值的行
data['A'] = data['A'].fillna(0)  # 用0填充空值

# 处理重复值
data = data.drop_duplicates()  # 移除重复的行

#...
```

* 数据存储：清洗和预处理后的数据可以存储为多种格式，如 CSV、JSON、SQL 数据库或其他格式，具体取决于项目的需求和后续使用数据的方式。`pandas` 库再次提供了简单的接口来执行这些数据存储操作。

```
# 存储为 CSV 文件
data.to_csv('processed_data.csv', index=False)

# 存储为 JSON 文件
data.to_json('processed_data.json')

# 存储到 SQL 数据库（使用 SQLAlchemy）
from sqlalchemy import create_engine

engine = create_engine('sqlite:///my_database.db') # 创建数据库引擎
data.to_sql('my_table', engine, index=False) # 将数据存储到数据库中

#...
```

* 数据分析与可视化：使用 `matplotlib` 和 `seaborn` 等库可以创建各种类型的图表和图形，帮助你理解和展示数据中的模式、趋势和关系。

```
import matplotlib.pyplot as plt

# 假设 data 是一个 DataFrame
plt.figure(figsize=(10, 6)) # 创建一个图形区域

# 绘制柱状图
plt.bar(data.index, data['A'])

# 绘制折线图
plt.plot(data.index, data['B'])

# 添加标题和坐标轴标签
plt.title('Data Visualization')
plt.xlabel('Index')
plt.ylabel('Values')

# 显示图形
plt.show()
```

```

这只是一个基本的概述和简单的代码示例。实际的项目可能会涉及到更复杂的任务，如使用正则表达式提取数据、多线程爬虫以提高效率、处理会话和cookies、使用更复杂的可视化技术等。此外，还需要根据目标网站的具体情况来调整爬虫策略。确保在任何数据收集和分析活动中，尊重网站的使用条款和法律法规。

写在最后
---

豆包MarsCode Cloud IDE提供了简洁直观的用户界面，使得用户能够轻松上手。其云端集成开发环境（Cloud IDE）无需复杂的配置，用户只需通过浏览器即可访问，极大地提高了开发效率。云端IDE的易用性是吸引用户的关键因素之一。豆包MarsCode通过简化配置过程，降低了用户的学习成本，使得即使是没有深厚技术背景的用户也能快速开始编程。

### AI功能

豆包MarsCode的AI功能强大，包括代码补全、代码生成、代码解释等，能够显著提升开发效率。AI功能的引入是豆包MarsCode的一大亮点。这些功能不仅能够减少开发者的重复劳动，还能在开发过程中提供智能建议，帮助开发者更快地解决问题。

### 插件市场

豆包MarsCode内置了丰富的插件市场，用户可以根据自己的需求选择合适的插件，进一步提高了开发效率。插件市场的存在使得豆包MarsCode能够适应不同开发者的需求，提供了极大的灵活性和可定制性。这对于那些需要特定功能或工具的开发者来说，是一个巨大的优势。

### 核心功能

豆包MarsCode的核心功能包括AI助手、代码补全、代码生成等，这些功能极大地提高了开发者的生产力。核心功能的强大是豆包MarsCode吸引用户的关键。通过提供这些功能，豆包MarsCode不仅解决了开发过程中的常见问题，还为用户提供了全新的开发体验。
```

扩展功能

除了核心功能外，豆包MarsCode还提供了丰富的扩展功能，如插件市场、云开发环境等，进一步满足了用户的多样化需求。扩展功能的提供显示了豆包MarsCode在满足用户需求方面的灵活性。这些功能使得豆包MarsCode不仅仅是一个代码助手，更是一个完整的开发生态系统。

免费开放

豆包MarsCode目前对国内开发者完全免费，这一策略无疑降低了用户的使用门槛，吸引了大量用户。免费开放的价格策略是豆包MarsCode能够迅速获得市场认可的重要原因。通过提供免费服务，豆包MarsCode不仅能够吸引更多的用户，还能够通过用户反馈不断改进产品。

技术社区

豆包MarsCode建立了活跃的技术社区，用户和开发者可以在社区中分享经验、解决问题，形成了良好的学习氛围。技术社区的建立对于豆包MarsCode的长期发展至关重要。它不仅能够帮助用户解决问题，还能够促进技术的交流和进步，为豆包MarsCode的持续创新提供了支持。

豆包MarsCode Cloud IDE凭借其强大的AI功能、易用的云端开发环境、丰富的插件市场以及免费开放的价格策略，获得了用户和开发者的高度评价。其核心功能和扩展功能能够满足不同开发者的需求，而技术社区的支持则为其长期发展提供了保障。言而总之，豆包MarsCode Cloud IDE是一个值得推荐的智能开发工具。