1. **Веб - краулер, который скачивает изображения с imageboard:**

***код для запуска с комментариями:***

import os  
import requests  
from bs4 import BeautifulSoup  
import abc

# Определяем интерфейс для стратегии скачивания страницы  
class PageFetcherStrategy(metaclass=abc.ABCMeta):  
    @abc.abstractmethod  
    def fetch(self, url: str) -> str:  
        pass

# Реализация стратегии скачивания страницы напрямую  
class DirectPageFetcher(PageFetcherStrategy):  
    def fetch(self, url: str) -> str:  
        response = requests.get(url)  
        return response.text

# Реализация стратегии скачивания страницы через прокси  
class ProxyPageFetcher(PageFetcherStrategy):  
    def fetch(self, url: str) -> str:  
        proxies = {"http": "http://your\_proxy\_here", "https": "https://your\_proxy\_here"}  
        response = requests.get(url, proxies=proxies)  
        return response.text

# Класс краулера  
class ImageCrawler:  
    def \_\_init\_\_(self, fetcher\_strategy: PageFetcherStrategy):  
        self.fetcher\_strategy = fetcher\_strategy

    def crawl(self, url: str, save\_path: str) -> None:  
        page\_content = self.fetcher\_strategy.fetch(url)  
        self.parse\_page(page\_content, save\_path)

    def parse\_page(self, page\_content: str, save\_path: str) -> None:  
        soup = BeautifulSoup(page\_content, 'html.parser')  
        images = soup.find\_all('img')  # Находим все теги img  
        for idx, img in enumerate(images):  
            img\_url = img['src']  
            self.download\_image(img\_url, save\_path, f"image\_{idx+1}.jpg")

    def download\_image(self, url: str, save\_path: str, filename: str) -> None:  
        response = requests.get(url, stream=True)  
        if response.status\_code == 200:  
            file\_path = os.path.join(save\_path, filename)  
            with open(file\_path, 'wb') as f:  
                for chunk in response.iter\_content(1024):  
                    f.write(chunk)  
            print(f"Downloaded: {file\_path}")  
        else:  
            print(f"Failed to download image: {url}")

# Алгоритм запуска краулера  
if \_\_name\_\_ == "\_\_main\_\_":  
    url = "https://example\_imageboard.com/"  # URL для сбора данных  
    save\_path = "./images"  # Папка для сохранения изображений

    # Проверяем, существует ли папка для сохранения изображений  
    if not os.path.exists(save\_path):  
        os.makedirs(save\_path)

    # Выбор стратегии скачивания: DirectPageFetcher или ProxyPageFetcher  
    fetcher\_strategy = DirectPageFetcher()

    # Создаем экземпляр краулера и запускаем его  
    crawler = ImageCrawler(fetcher\_strategy)  
    crawler.crawl(url, save\_path)

**2. Алгоритм запуска краулера:**

* Определить интерфейс `PageFetcherStrategy` для стратегии скачивания страницы.
* Реализовать различные стратегии скачивания страницы, например, `DirectPageFetcher` и `ProxyPageFetcher`.
* Определить класс `ImageCrawler`, который будет использовать одну из стратегий скачивания для получения и последующего анализа страницы.
* В методе `crawl` класса `ImageCrawler`, вызвать метод `fetch` выбранной стратегии для получения содержимого страницы, затем вызвать `parse\_page` для анализа и извлечения изображений.
* Метод `download\_image` класса `ImageCrawler` будет скачивать изображения по полученным URL.
* Создать экземпляр `ImageCrawler`, передав в конструктор выбранную стратегию.
* Запустить краулер, вызвав метод `crawl` с URL-адресом, который нужно обработать, и папкой для сохранения изображений.

**Аналогично предыдущему примеру, убедитесь, что у вас установлены библиотеки `requests` и `beautifulsoup4` перед запуском кода.**

**Вы можете установить их с помощью команды*:***

***pip install requests beautifulsoup4***