AutoAgent

By Ahmed Maghri & Kyu In Choi

Constructor University Hackathon 2025

AutoAgent

- Kyu In Choi
- 2nd year Computer Science and Data Science student.
- +49 1627704107
- kchoi@constructor.university
- Discord: choikyy

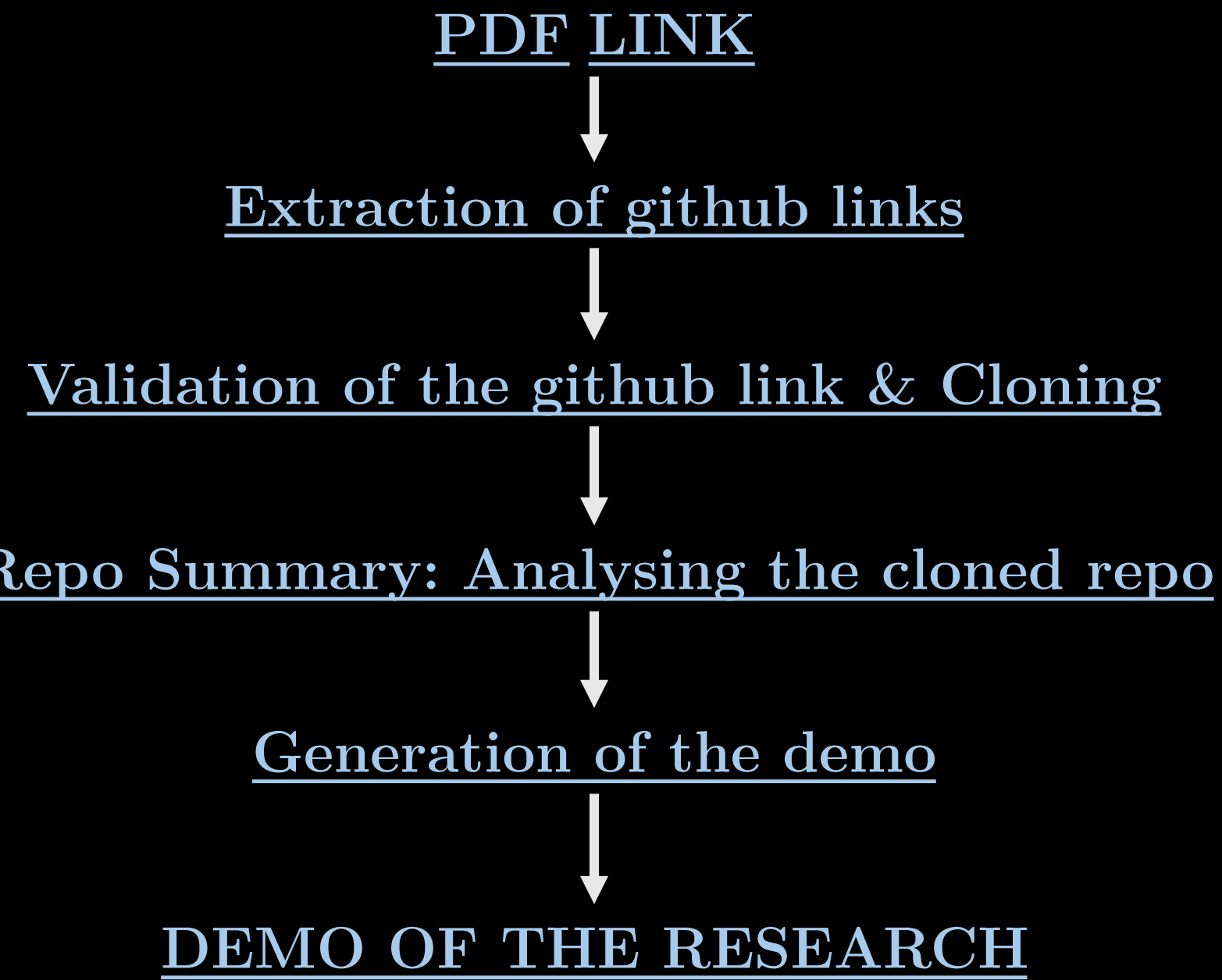
- Ahmed Maghri
- 2nd year Computer Science, Mathematics and Modeling Student.
- +49 15237260733
- amaghri@constructor.university
- Discord: iraamx

AutoAgent

PDF LINK

DEMO OF THE RESEARCH

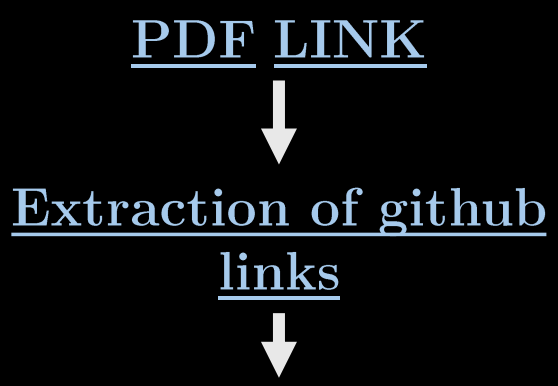
AutoAgent


PDF LINK
Extraction of github links
Validation of the github link & Cloning
Repo Summary: Analysing the cloned repo
Generation of the demo
DEMO OF THE RESEARCH

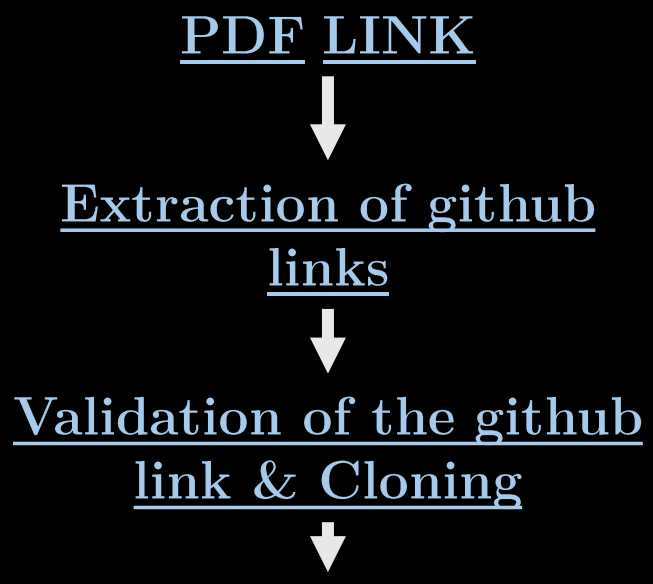
AutoAgent

Using Regular Expressions, we go over the file and search for any possible GitHub link that is present.

No LLM involved.



AutoAgent



1) Input all the GitHub links found.

2) Feed a truncated version of the raw text from the pdf and the GitHub link to an LLM.

3) Prompt engineering with the LLM so that it chooses the correct repo link corresponding to our research paper.

4) Cloning the chosen link locally.

AutoAgent

GOAL: Give a valid summary in order to be able to generate a demo file. (Number of files, the main language used, already present demo, used, entry points...)

PDF LINK

↓

Extraction of github links

↓

Validation of the github link & Cloning

↓

Repo Summary: Analysing the cloned repo

↓

AutoAgent

GOAL: Give a valid summary in order to be able to generate a demo file. (Number of files, the main language used, already present demo, used, entry points...)

PDF LINK

↓

Extraction of github links

↓

Validation of the github link & Cloning

↓

Repo Summary: Analysing the cloned repo

↓

STRATEGTY 1:
Fully Heuristically

- Too many edge cases, and wasn't flexible at all (as expected)

AutoAgent

GOAL: Give a valid summary in order to be able to generate a demo file. (Number of files, the main language used, already present demo, used, entry points...)

PDF LINK

↓

Extraction of github links

↓

Validation of the github link & Cloning

↓

Repo Summary: Analysing the cloned repo

↓

STRATEGTY 2:
Fully AI powered

- Overall worked somewhat well.
- But due to the huge size of the input (the research + the code), it happened to hallucinate a lot.
- Truncating the input didn't work well as it made the accuracy way lower

AutoAgent

PDF LINK

↓

Extraction of github links

↓

Validation of the github link & Cloning

↓

Repo Summary: Analysing the cloned repo

↓

GOAL: Give a valid summary in order to be able to generate a demo file. (Number of files, the main language used, already present demo, used, entry points…)

STRATEGTY 3:

Somewhere in between

1) Using heuristics for some features: language used.

2) Using LLM in case heuristics doesn't work: An LLM/Model file present, searching for entry points, config files…

3) Using completely LLMs + heuristics: Demo file search.

AutoAgent

PDF LINK

↓

Extraction of github links

↓

Validation of the github link & Cloning

↓

Repo Summary: Analysing the cloned repo

↓

Generation of the demo

↓

DEMO OF THE RESEARCH

Input: Summary given by our code scanner.

1) Iterate over all the demo files found.

2) Validates or invalidates each of them to make sure that they are the correct demo.

a) If a valid demo file found, then outputs that demo file.
b) If no demo file found, then generate one with feeding to the LLM the full summary + the readme if present.

3) Saves the demo file returned.

What if more than 1 demo are validated?

AutoAgent

PDF LINK

↓

Extraction of github links

↓

Validation of the github link & Cloning

↓

Repo Summary: Analysing the cloned repo

↓

Generation of the demo

↓

DEMO OF THE RESEARCH

Input: Summary given by our code scanner.

1) Iterate over all the demo files found.

2) Validates or invalidates each of them to make sure that they are the correct demo.

a) If a valid demo file found, then outputs that demo file.
b) If no demo file found, then generate one with feeding to the LLM the full summary + the readme if present .

3) Saves the demo file returned.

What if more than 1 demo are validated?
- Never happened while testing, and that's why we decided to stop the program as soon as it finds one for optimization.

AutoAgent

PDF LINK

↓

Extraction of github links

↓

Validation of the github link & Cloning

↓

Repo Summary: Analysing the cloned repo

↓

Generation of the demo

↓

DEMO OF THE RESEARCH

EVALUATION

AutoAgent

PDF LINK

↓

Extraction of github links

↓

Validation of the github link & Cloning

↓

Repo Summary: Analysing the cloned repo

↓

Generation of the demo

↓

DEMO OF THE RESEARCH

EVALUATION

SCORING SYSTEM

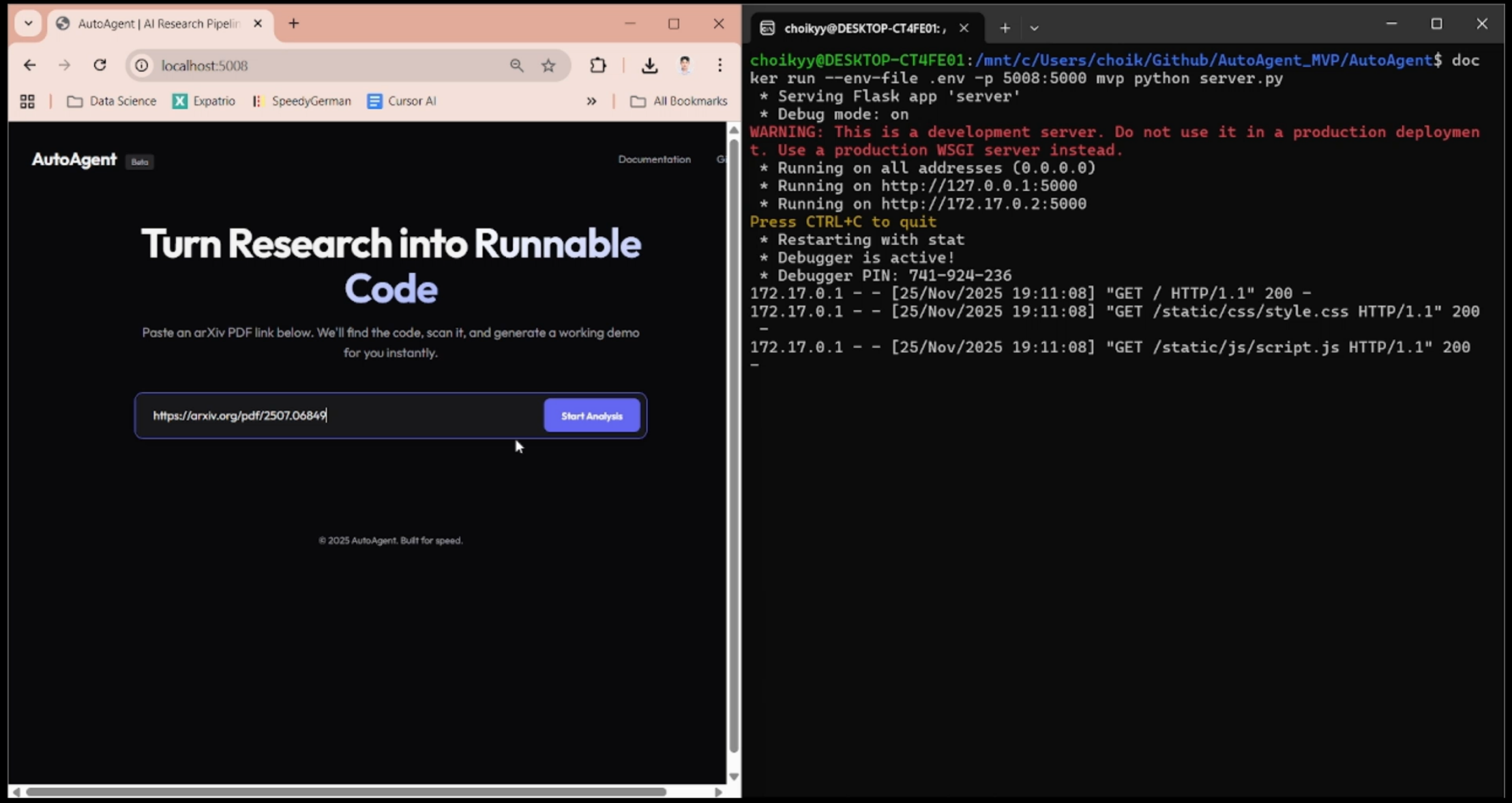
1pt: For no syntax error.
1pt: If exit code is 0.
1pt: If it finishes withing a reasonable amount of time (30 secs in our case).
1pt: If prints something.
1pt: If no error in stderror.

5pts: Judged by LLM, input: generated code + execution output + project summary.

Ask the LLM to score the demo on a scale of 0-5 based on:

- Clarity (is the output human readable?)
- Completeness (Is the demo self contained?)
- Relevance (Did it actually use the github repo)

AutoAgent


AutoAgent | AI Research Pipelin
localhost:5008
Data Science
Expatrio
SpeedyGerman
Cursor AI
All Bookmarks
AutoAgent
Beta
Documentation
Turn Research into Runnable Code
Paste an arXiv PDF link below. We'll find the code, scan it, and generate a working demo for you instantly.
https://arxiv.org/pdf/2507.06849
Start Analysis
© 2025 AutoAgent. Built for speed.
choikyy@DESKTOP-CT4FE01:
choikyy@DESKTOP-CT4FE01:/mnt/c/Users/choik/Github/AutoAgent_MVP/AutoAgent$ docker run --env-file .env -p 5008:5000 mvp python server.py
* Serving Flask app 'server'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on all addresses (0.0.0.0)
* Running on http://127.0.0.1:5000
* Running on http://172.17.0.2:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 741-924-236
172.17.0.1 - - [25/Nov/2025 19:11:08] "GET / HTTP/1.1" 200 -
172.17.0.1 - - [25/Nov/2025 19:11:08] "GET /static/css/style.css HTTP/1.1" 200 -
172.17.0.1 - - [25/Nov/2025 19:11:08] "GET /static/js/script.js HTTP/1.1" 200 -

AutoAgent

Some tweaks done & results after testing

1)
- Sometimes the LLM would give as a valid demo an .md file.
- Sometimes the demo code started with something like “ ```python… “.
- Sometimes the LLM didn't output anything…

Solution: Prompt engineering.

Average score: 7/10
Max: 9/10
Min: 1/10

Average run time: 70 sec
(on the given data set)

2)
Which LLM to choose?
We tried:

a) Groq: Inaccurate, hallucinates regularly, or sometimes outputs nothing.
b) Gemini 2.5: Did an overall good job; however, it hallucinated frequently during demo validation. Even after multiple attempts to tweak the prompt, it would sometimes say that a prompt is valid even though it shouldn’t have.
c) Gpt-4o: Accuracy close to Gemini, but it hallucinates way less after some prompt engineering, so we moved with that one.

AutoAgent

By Ahmed Maghri & Kyu In Choi

Thank you very much

Constructor University Hackathon 2025

AutoAgent

PROMPTS USED

prompt = f"""
You are validating a demo code file for a project.

Project summary:
{scan_summary}

Demo file contents:

{demo_code}

Is this demo runnable and demonstrates the project?

Requirements for YES:
- If the file is a .md file, it is not valid.
- The code must be syntactically correct.
- It must import necessary modules from the project.
- Shows a working example of the main functionality.
- Not just comments or pseudocode.
- It must be runnable.

answer with ONLY: YES or NO.

DO NOT include any punctuation, explanation, or code block markers.
"""

AutoAgent

PROMPTS USED

```
prompt = f"""
You are a code generation expert. Your primary goal is to generate a script that runs successfully.

Given this project structure summary:
{scan_summary}

README excerpt:
{readme}

Example file excerpt (if any):
{example_content}

Generate a SINGLE runnable demo script that:
1. **Dependency Check (CRITICAL):** Includes necessary standard Python imports (os, sys, subprocess) at the top. For any external library required (e.g., 'fire', 'torch', 'scipy'), you MUST write a self-installing try/except block.

Example of required self-installing block:
import os
import sys
import subprocess

def install_missing_dependency(package_name):
    print(f"Installing missing dependency: {{package_name}}...")
    # Use subprocess to ensure it runs correctly inside the container
    subprocess.check_call([sys.executable, "-m", "pip", "install", package_name])

    import <external_library>
except ImportError:
    install_missing_dependency("<external_library>")
    import <external_library>

2. Imports required project modules from the cloned repository.
3. Has no TODOs or placeholders.
4. Uses detected entrypoints if available.
5. Shows a minimal working example that produces clean, informative output to stdout.

Return only the python code for the demo script.
REPEAT: ONLY THE CODE, NO EXTRA TEXT, NO CODE BLOCK MARKERS (e.g., ```python).
```