Agents for Software Development

Graham Neubig

Carnegie Mellon University
Language Technologies Institute

All Hands AI

My Profile

- Professor at CMU

Carnegie Mellon University

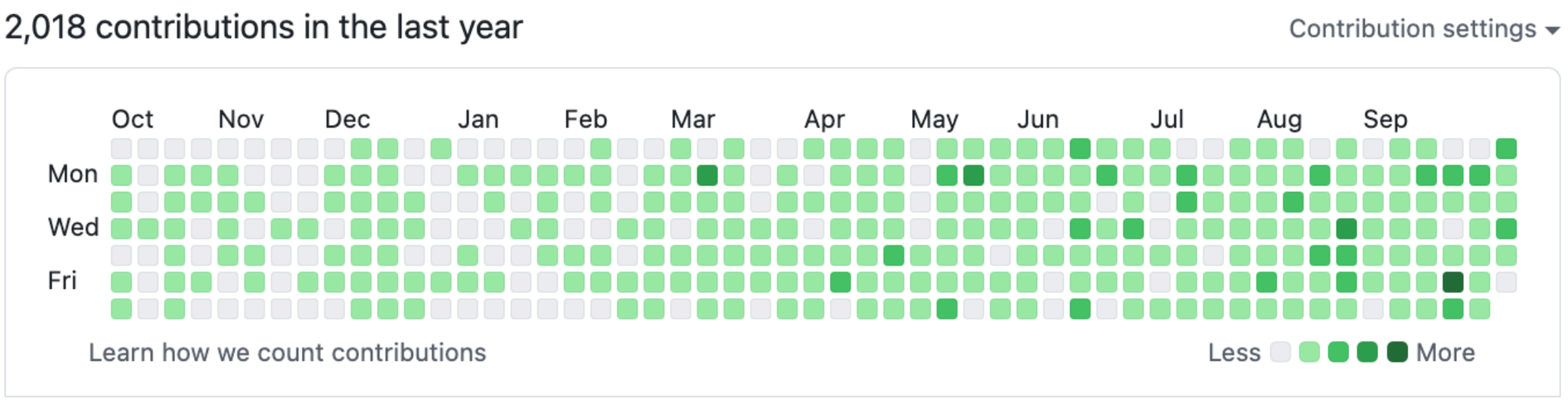
- Chief Scientist at All Hands AI (building open-source coding agents)

All Hands AI

- Maintainer of OpenHands

https://github.com/All-Hands-AI/OpenHands

- Software developer

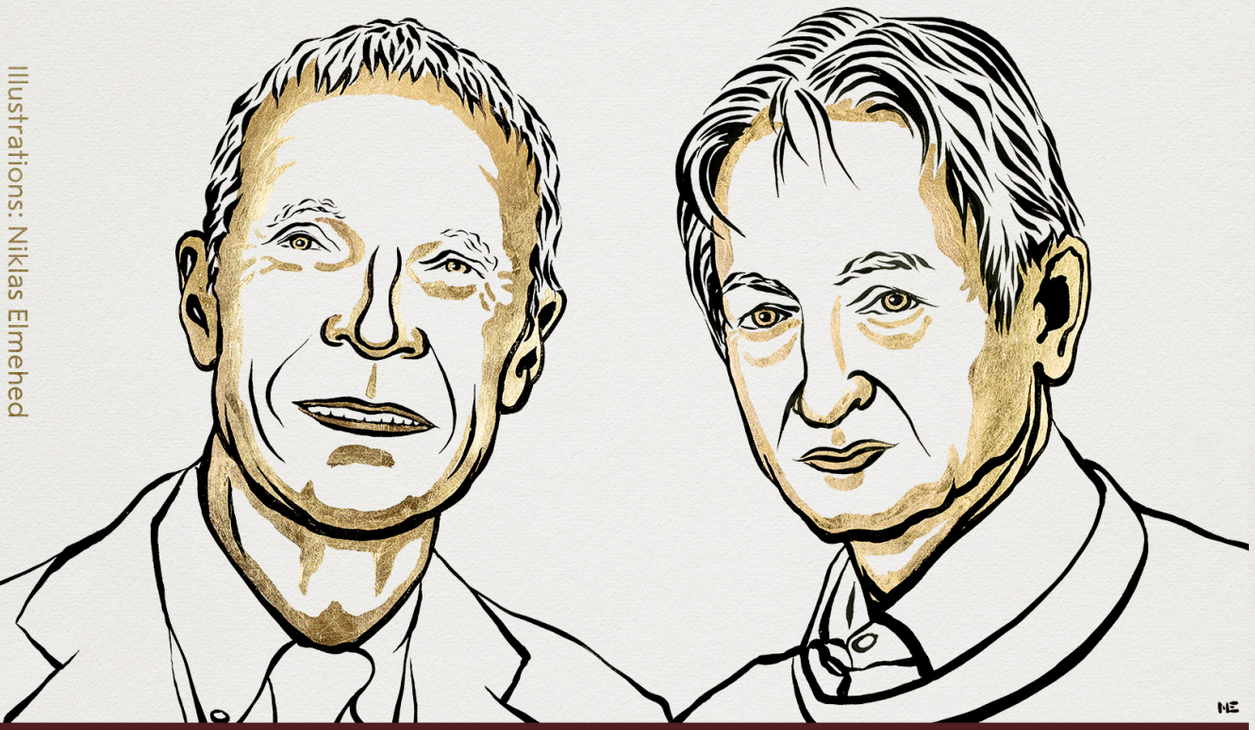


More and more major businesses and industries are being run on software and delivered as online services—from movies to agriculture to national defense. […] Over the next 10 years, I expect many more industries to be disrupted by software […].

— Marc Andreessen - Why Software is Eating the World (2011)

If we gave everyone the ability to quickly write software to achieve their goals, what could they do?

THE NOBEL PRIZE IN PHYSICS 2024



Illustrations: Niklas Elmehed

John J. Hopfield Geoffrey E. Hinton

"for foundational discoveries and inventions that enable machine learning with artificial neural networks"

THE ROYAL SWEDISH ACADEMY OF SCIENCES

THE NOBEL PRIZE IN CHEMISTRY 2024

Illustrations: Niklas Elmehed

David Baker

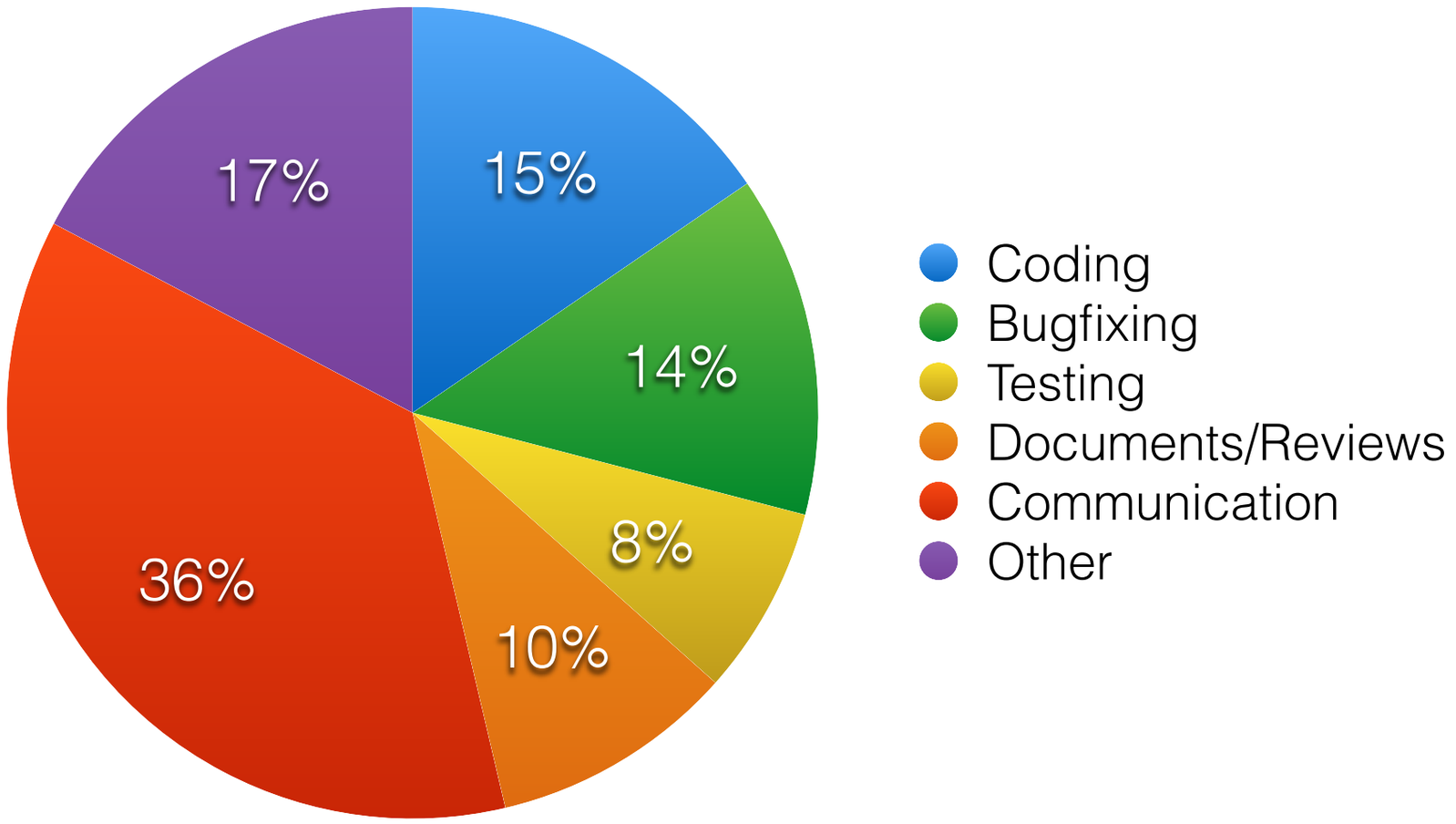
"for computational protein design"

Demis Hassabis John M. Jumper

"for protein structure prediction"

THE ROYAL SWEDISH ACADEMY OF SCIENCES

What is Involved in Developing Software?



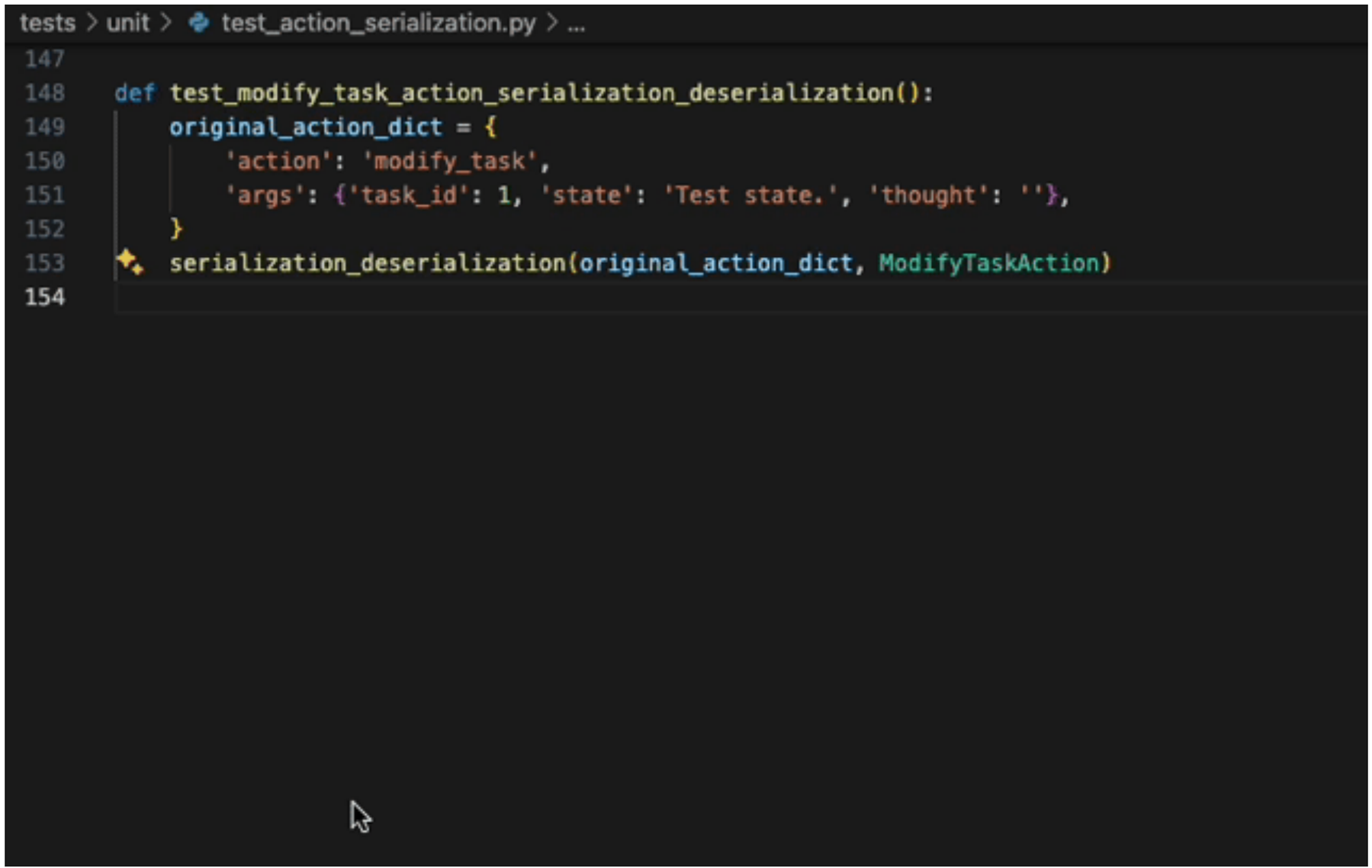
Today was a Good Day: The Daily Life of Software Developers
Meyer et al. 2019

How Can We Support Developers?

Level	Self Driving	Software Development
0: No Automation	Manual driving	Manual Coding
1: Driver Assistance/ Code Completion	Adaptive cruise control/braking	Copilot/Cursor code completion
2: Partial Automation	Tesla's autopilot	Copilot chat refactoring
3: Conditional Automation	Mercedes-Benz drive pilot	DiffBlue test generation, Transcoder code porting
4: High Automation	Cruise self-driving vehicles	Devin/OpenDevin end-to-end development
5: Full Automation	...	...

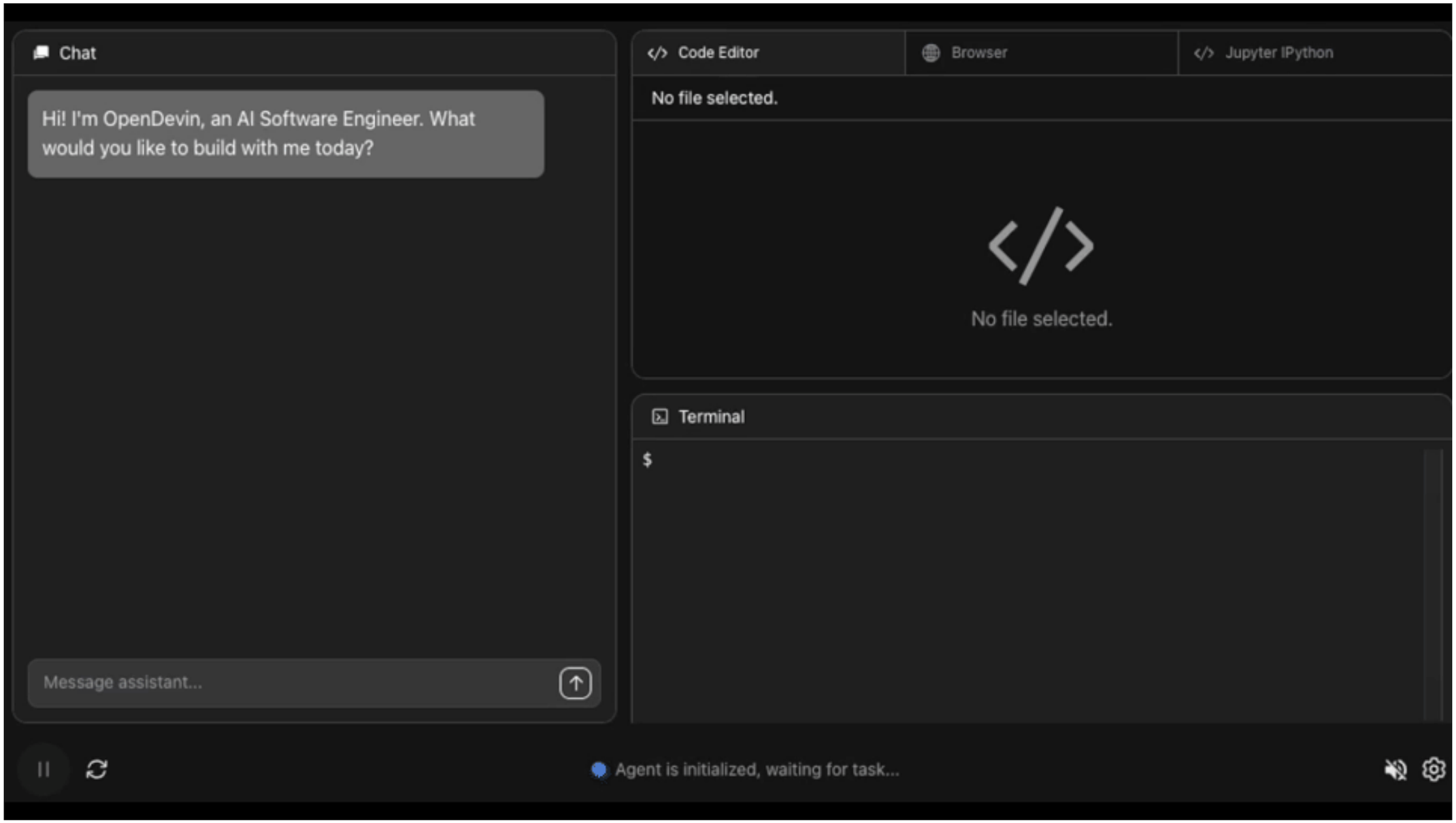
Development Copilots

- Work synchronously with the developer to ease writing code
- e.g. **Github Copilot/Cursor**

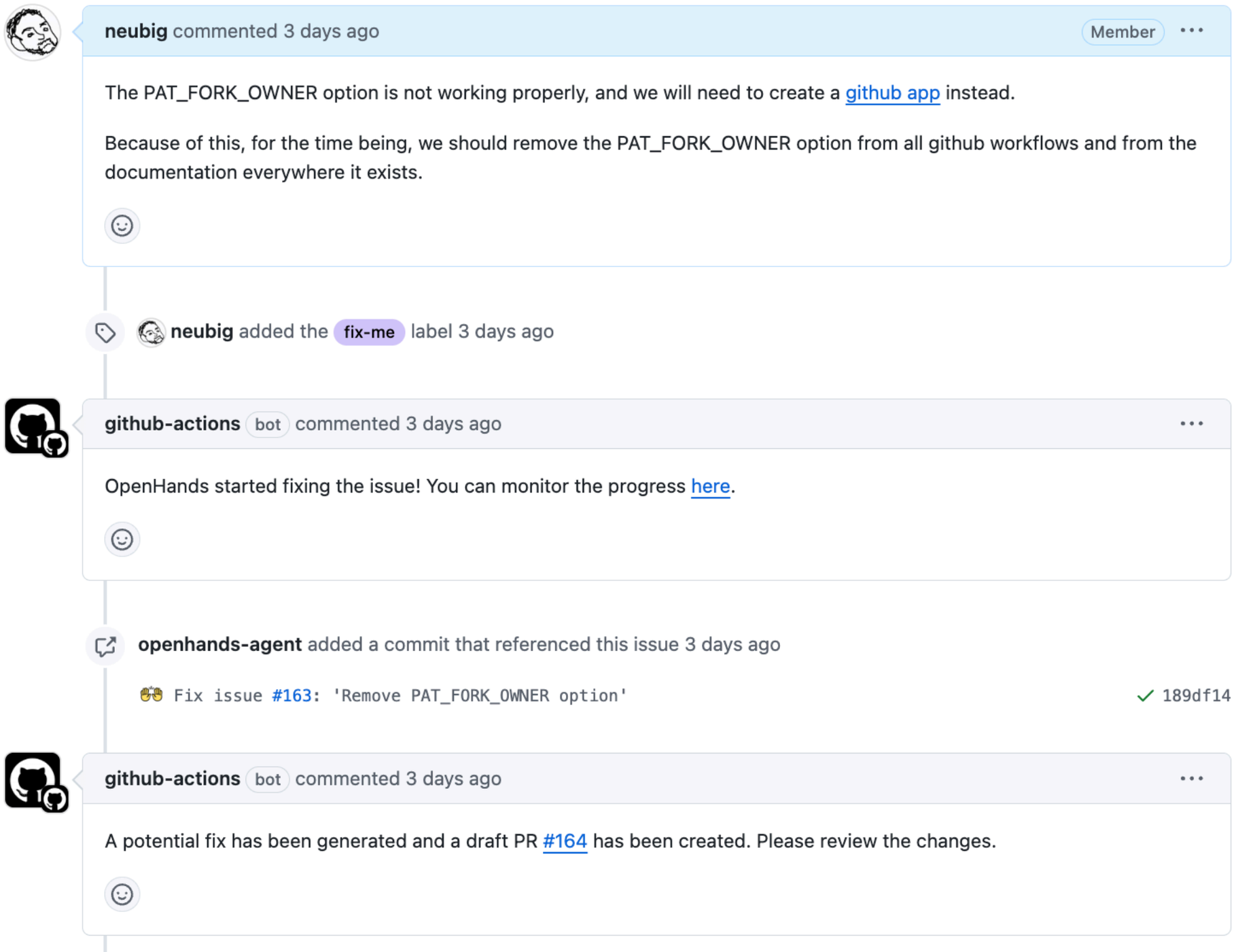


Development Agents

- For coding (e.g. SWE-Agent, Aider)
- For broader development (e.g. Devin, OpenHands)



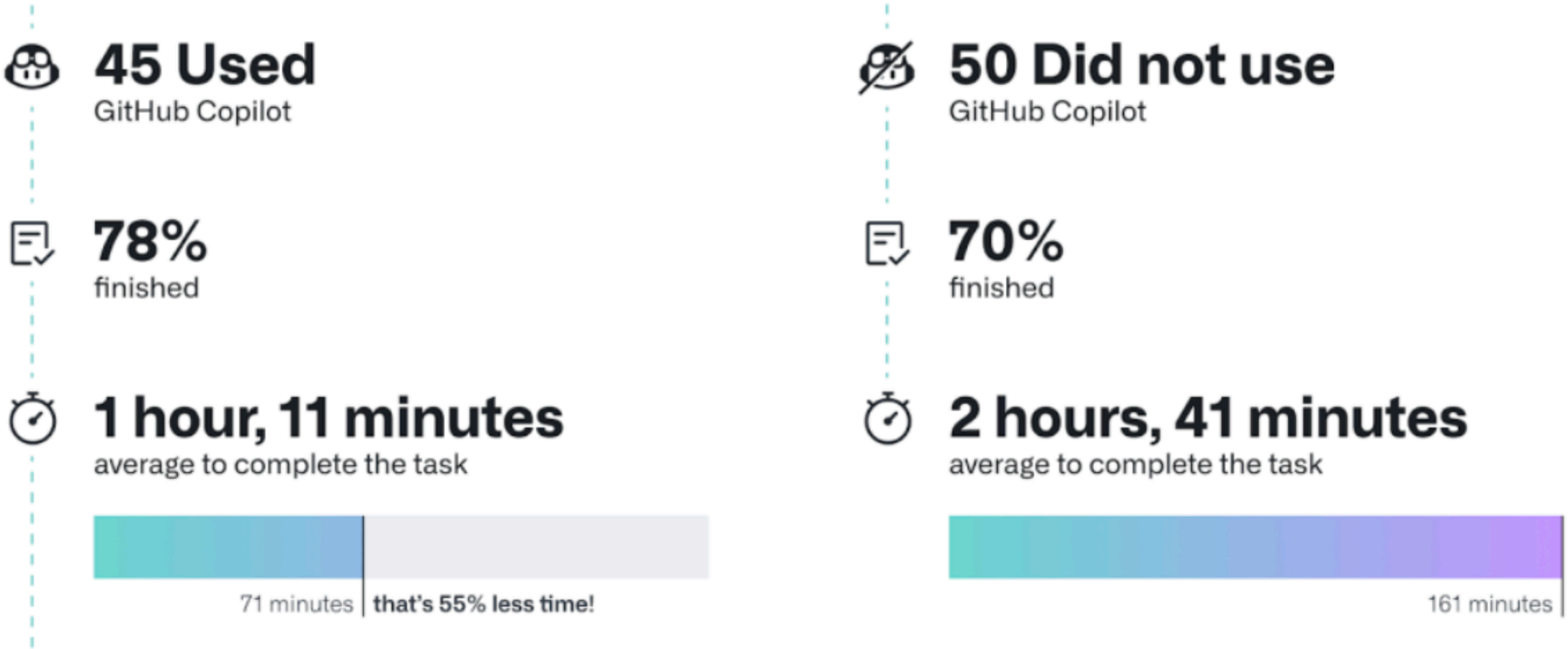
Autonomous Issue Resolution



https://github.com/All-Hands-AI/OpenHands-resolver

How Promising?

- Code generation leads to large improvements in productivity (Github 2023)



Challenges in Coding Agents

- Defining the Environment
- Designing an Observations/Actions
- Code Generation (atomic actions)
- File Localization (exploration)
- Planning and Error Recovery
- Safety

Software Development Environments

Types of Environments

- **Actual Environments:**
  - *Source Repositories:* Github, Gitlab
  - *Task Management Software:* Jira, Linear
  - *Office Software:* Google Docs, Microsoft Office
  - *Communication Tools:* Gmail, Slack
- **Testing Environments:**
  - Mostly focused on coding!
  - Developers do more, e.g. browse the web (next session)

Simple Coding
(Chen et al. 2021, Austin et al. 2021)

- e.g. HumanEval/ MBPP
- Examples of usage of the Python standard library
- Includes docstring, some example inputs/outputs, and tests

```
def solution(lst):
    """Given a non-empty list of integers, return the sum of all of the odd elements
    that are in even positions.

    Examples
    solution([5, 8, 7, 1]) ==>12
    solution([3, 3, 3, 3, 3]) ==>9
    solution([30, 13, 24, 321]) ==>0
    """
    return sum(lst[i] for i in range(0,len(lst)) if i % 2 == 0 and lst[i] % 2 == 1)
```

```
def encode_cyclic(s: str):
    """
    returns encoded string by cycling groups of three characters.
    """
    # split string to groups. Each of length 3.
    groups = [s[(3 * i):min((3 * i + 3), len(s))] for i in range((len(s) + 2) // 3)]
    # cycle elements in each group. Unless group has fewer elements than 3.
    groups = [(group[1:] + group[0]) if len(group) == 3 else group for group in groups]
    return "".join(groups)

def decode_cyclic(s: str):
    """
    takes as input string encoded with encode_cyclic function. Returns decoded string.
    """
    # split string to groups. Each of length 3.
    groups = [s[(3 * i):min((3 * i + 3), len(s))] for i in range((len(s) + 2) // 3)]
    # cycle elements in each group.
    groups = [(group[-1] + group[:-1]) if len(group) == 3 else group for group in groups]
    return "".join(groups)
```

Broader Domains: CoNaLa/ODEX

(Yin et al. 2018, Wang et al. 2022)

- CoNaLa: Broader data scraped from StackOverflow
- ODEX: Adds execution-based evaluation
- Wider variety of libraries

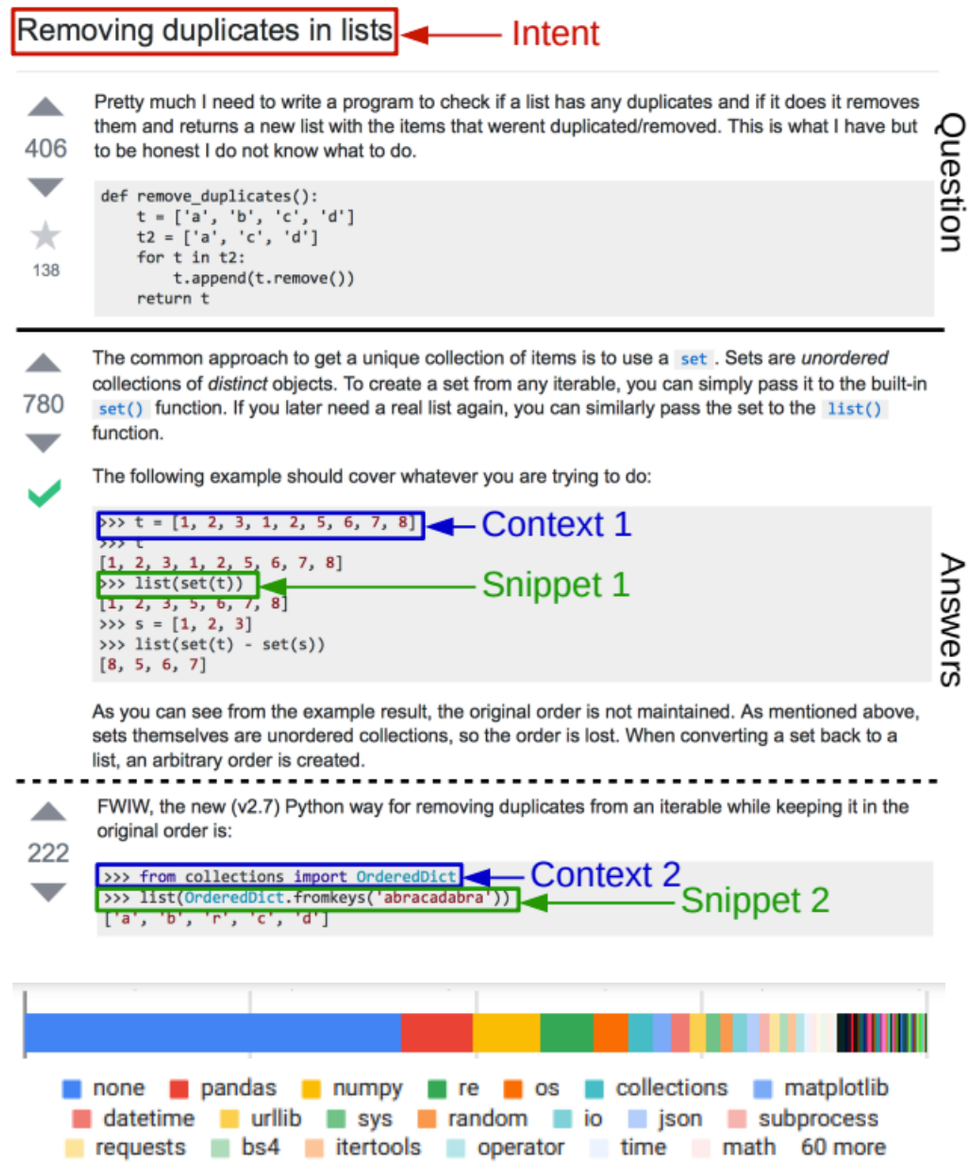
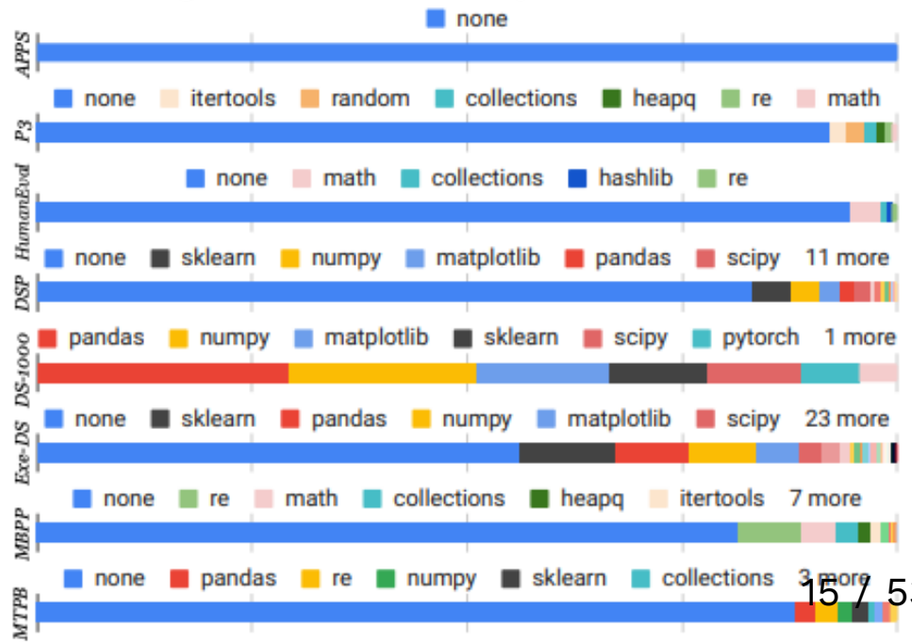


Figure 3: ODEX library distribution.



Data Science Notebooks: ARCADE

(Yin et al. 2022)

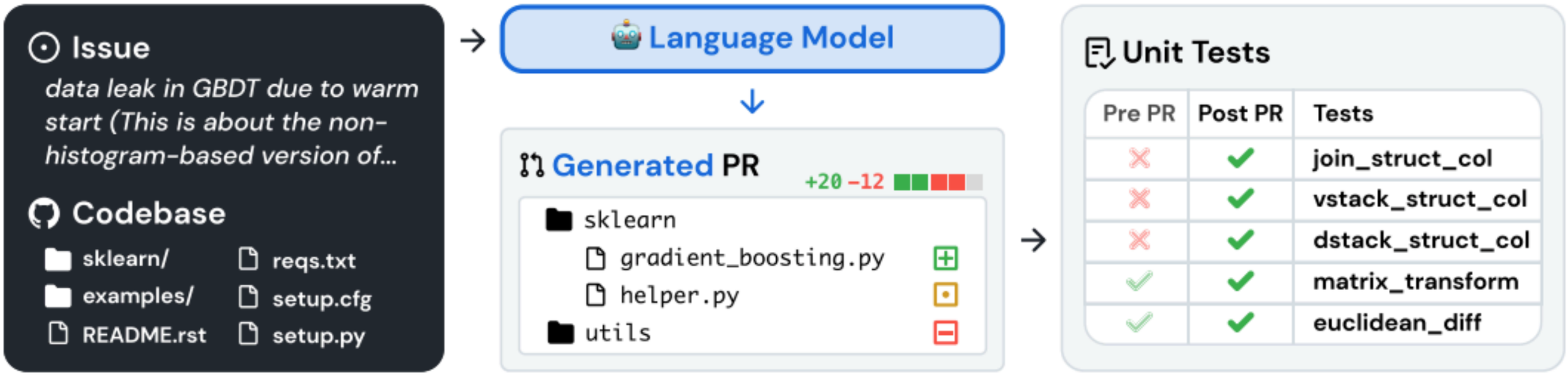
- Data science notebooks (e.g. Jupyter) allow for incremental implementation
- Allows evaluation of code in context

Figure 1: An example of a computational notebook adapted from our dataset, with examples of reading and preprocessing data (cell c_1), data wrangling (cell c_2, c_3), and data analysis (cells $c_3 - c_7$). Annotated NL intents are shown in green.

Dataset: SWEBench

(Jiminez et al. 2023)

- Issues from GitHub + codebases -> pull request



- Requires long-context understanding, precise implementation

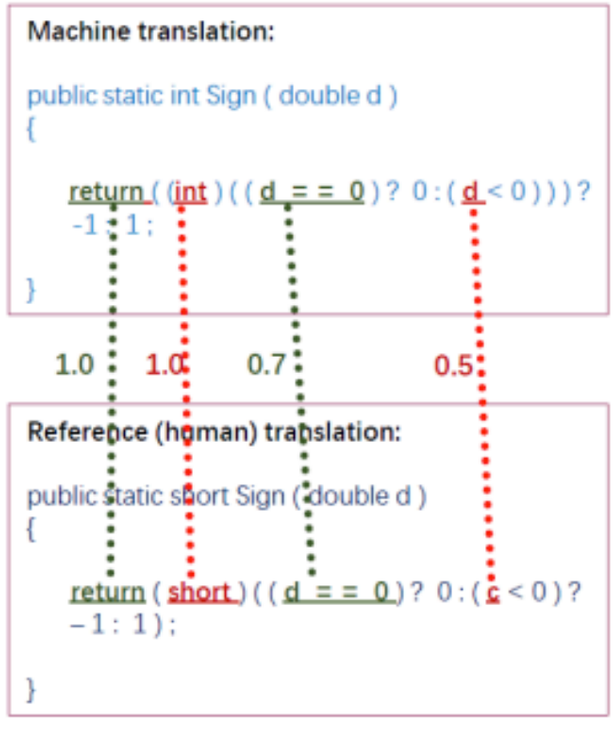
Metric: Pass@K
(Chen et al. 2021)

- Basic idea: “if we generate *K* examples, will at least one of them pass unit tests”
- Generating only *K* will result in high variance, so we generate $N > K$ with C correct answers, and then calculate expected value

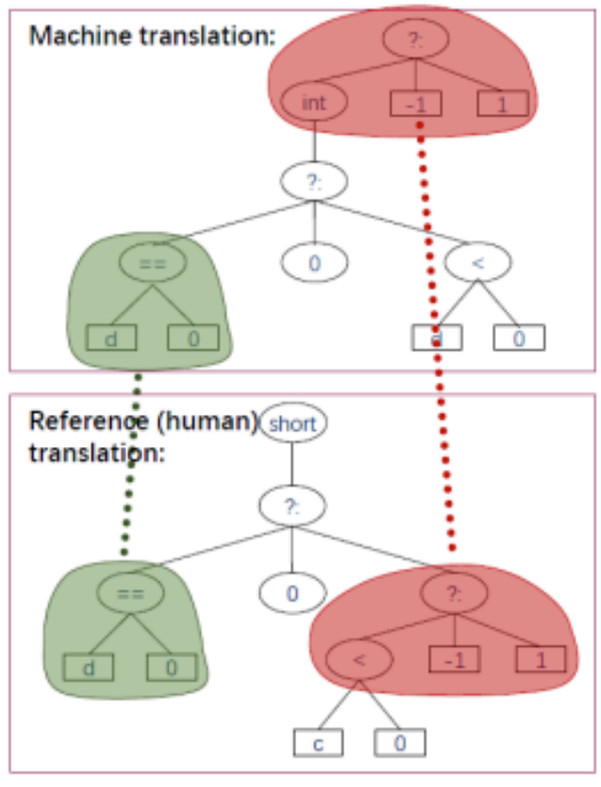
$$\text{pass@}k := \mathop{\mathbb{E}}_{\text{Problems}} \left[1 - \frac{\binom{n-c}{k}}{\binom{n}{k}} \right]$$

Metric: Lexical/Semantic Overlap

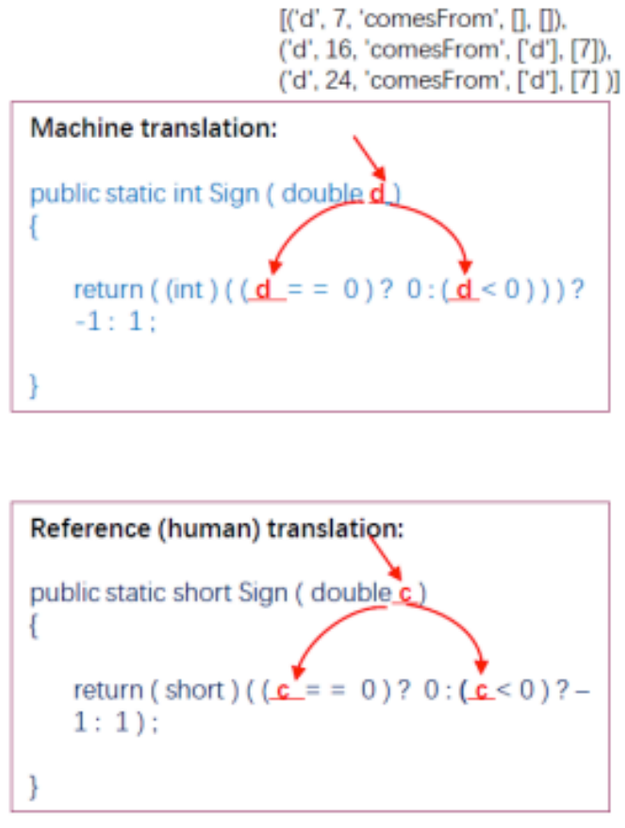
- Issues w/ execution-based evaluation:
  - Requires that code be easily executable (requires unit tests and hard in large libraries)
  - Ignores stylistic considerations
- **BLEU:** consider text n-gram overlap with human code
- **CodeBLEU:** also considers syntax and semantic flow (Ren et al. 2020)
- **CodeBERTScore:** BERTScore with CodeBERT trained on lots of code (Zhou et al. 2023)



Weighted N-Gram Match



Syntactic AST Match

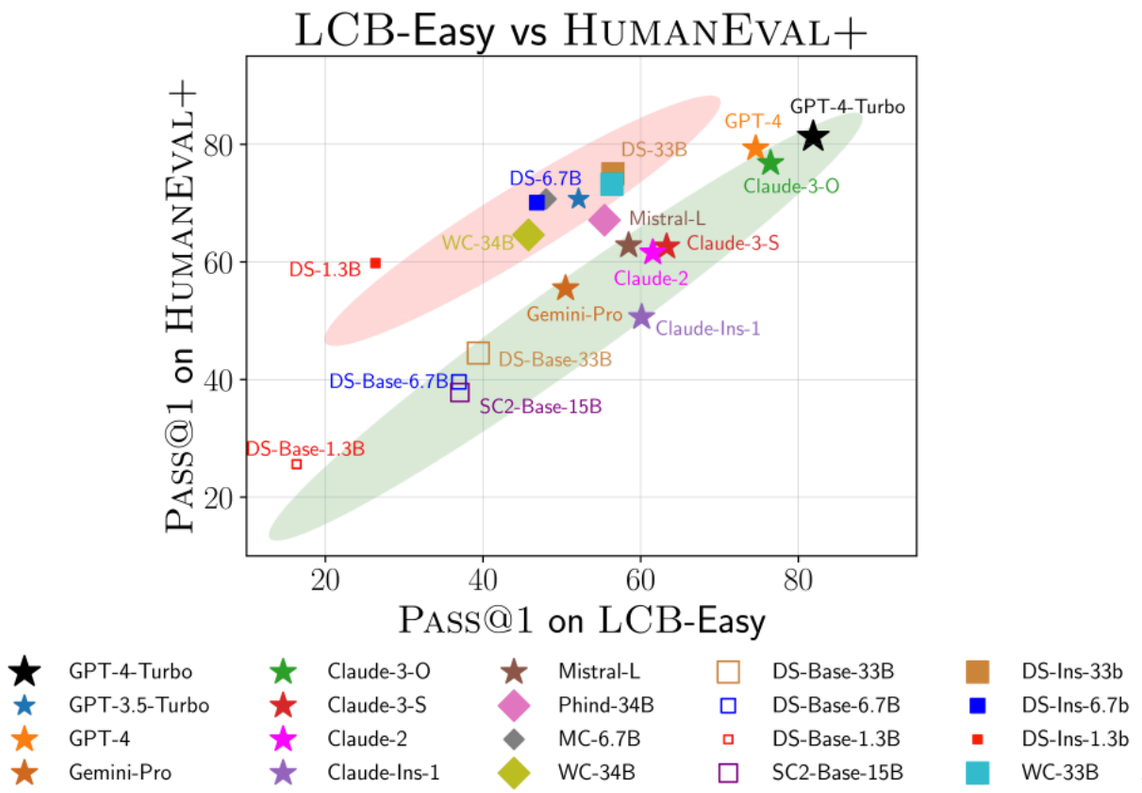


Semantic Data-flow Match

An Aside: Dataset Leakage

- Leakage of datasets is a big problem
- ARCADE shows that novel notebooks are harder than online notebooks
- LiveCodeBench (Jain et al. 2023) shows that some code LMs outperform on HumanEval

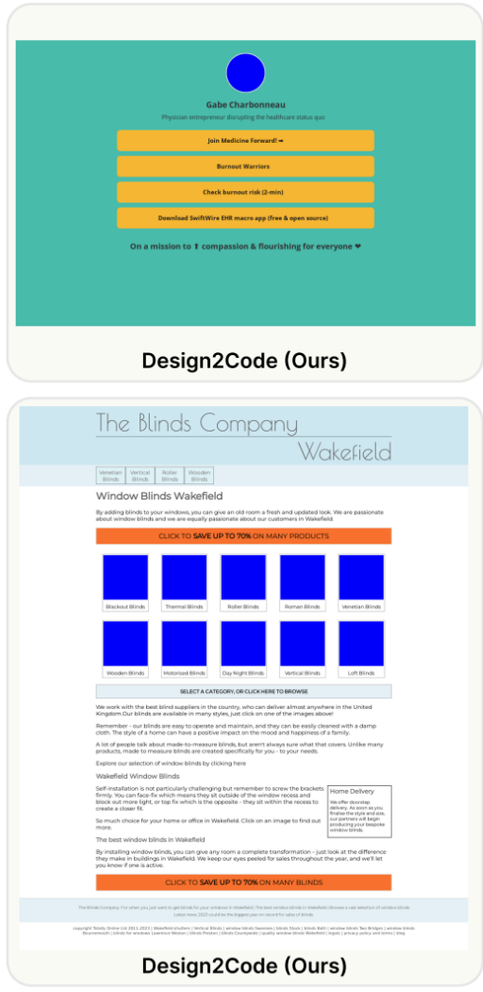
pass@k	Existing 5	New 5
INCODER 1B	30.1	3.8
INCODER 6B	41.3	7.0
CODEGEN$_{\text{multi}}$ 350M	13.3	1.0
CODEGEN$_{\text{multi}}$ 2B	25.0	2.7
CODEGEN$_{\text{multi}}$ 6B	28.0	3.0
CODEGEN$_{\text{multi}}$ 16B	31.2	4.6
CODEGEN$_{\text{mono}}$ 350M	18.9	1.9
CODEGEN$_{\text{mono}}$ 2B	35.8	6.5
CODEGEN$_{\text{mono}}$ 6B	42.1	8.9
CODEGEN$_{\text{mono}}$ 16B	46.7	12.0
PALM 62B (1.3T Tokens)	49.7	12.5
+ Python Code	58.8 +9.1	21.4 +8.9
+ Notebooks (PACHINCO)	64.6 +7.8	30.6 +9.2
− Schema Description	60.5 −4.1	22.7 −7.9



Dataset: Design2Code

(Si et al. 2024)

- Code generation from web sites



- Also proposed Design2Code model

Metric: Visual Similarity of Web Site

- Design2Code evaluates by two metrics
- **High-level visual similarity:** Similarity between visual embeddings of the generated sites
- **Low-level element similarity:** Recall of each individual element

Designing Observation/ Action Spaces

Coding Agents Must

- Understand repository structure
- Read in existing code
- Modify or produce code
- Run code and debug

Example: CodeAct (Wang et al. 2024)

- Interact w/ the environment through code

Instruction: Determine the most cost-effective country to purchase the smartphone model "CodeAct 1". The countries to consider are the USA, Japan, Germany, and India.

Available APIs
[1] lookup_rates(country: str) -> (float, float)
[2] convert_and_tax(price: float, exchange_rate: float, tax_rate: float) -> float
[3] estimate_final_price(converted_price: float, shipping_cost: float) -> float
[4] lookup_phone_price(model: str, country: str) -> float
[5] estimate_shipping_cost(destination_country: str) -> float

LLM Agent using [**Text/JSON**] as Action

Think I should calculate the phone price in USD for each country, then find the most cost-effective country.

Action
```
Text: lookup_rates, Germany
JSON: {"tool": "lookup_rates", "country": "Germany"}
```

Environment 1.1, 0.19

Action
```
Text: lookup_phone_price, CodeAct 1, Germany
JSON: {"tool": "lookup_phone_price", "model": "CodeAct 1", "country": "Germany"}
```

Environment 700

Action
```
Text: convert_and_tax, 700, 1.1, 0.19
JSON: {"tool": "convert_and_tax", "price": 700, "exchange_rate": 1.1, "tax_rate": 0.19}
```

Environment 916.3

[... interactions omitted (look up shipping cost and calculate final price) ...]

Action
```
Text: lookup_rates, Japan
JSON: {"tool": "lookup_rates", "country": "Japan"}
```

[... interactions omitted (calculate final price for all other countries)...]

Response The most cost-effective country to purchase the smartphone model is Japan with price 904.00 in USD.

CodeAct: LLM Agent using [**Code**] as Action

Think I should calculate the phone price in USD for each country, then find the most cost-effective country.

Action

```
countries = ['USA', 'Japan', 'Germany', 'India']
final_prices = {}

for country in countries:
    exchange_rate, tax_rate = lookup_rates(country)
    local_price = lookup_phone_price("xAct 1", country)
    converted_price = convert_and_tax(
        local_price, exchange_rate, tax_rate
    )
    shipping_cost = estimate_shipping_cost(country)
    final_price = estimate_final_price(converted_price, shipping_cost)
    final_prices[country] = final_price

most_cost_effective_country = min(final_prices, key=final_prices.get)
most_cost_effective_price = final_prices[most_cost_effective_country]
print(most_cost_effective_country, most_cost_effective_price)
```

Control & Data Flow of Code Simplifies Complex Operations

Fewer Actions Required!

Re-use `min` Function from Existing Software Infrastructures (Python library)

Environment 1.1, 0.19

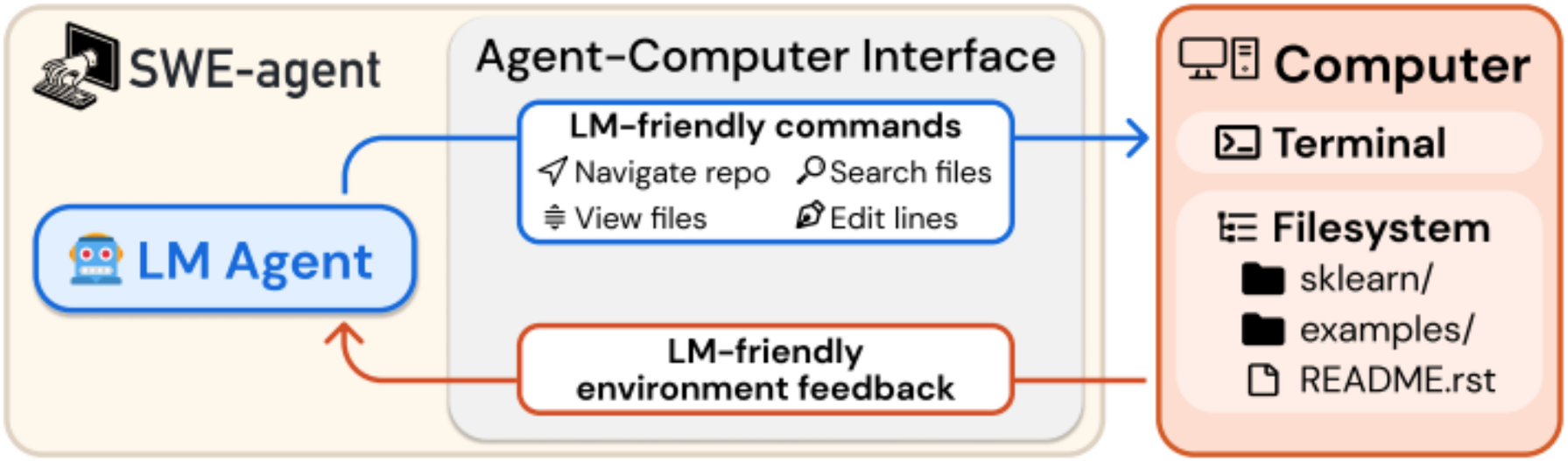
Response The most cost-effective country to purchase the smartphone model is Japan with price 904.00 in USD.

- Can execute bash commands, Jupyter commands
- Faster resolution, higher success than direct tool use

Example: SWE-Agent

(Yang+Jimenez et al. 2024)

- Define specialized tools that make it possible to efficiently explore repositories and edit code



Observation (File Viewer)

```
[File: solvers/diophantine.py (3093 lines total)]
(400 lines above)
401:    elif total_degree == 3:
402:        diop_type = "cubic_thue"
403:
404:    elif (total_degree > 3):
405:        if all(coeff[k] == 1):
406:            diop_type = 'general_sum_even'
407:
408:    if diop_type is not None:
409:        return var, coeff, diop_type
(2684 lines below)
```

Response (Thought + Edit Action)

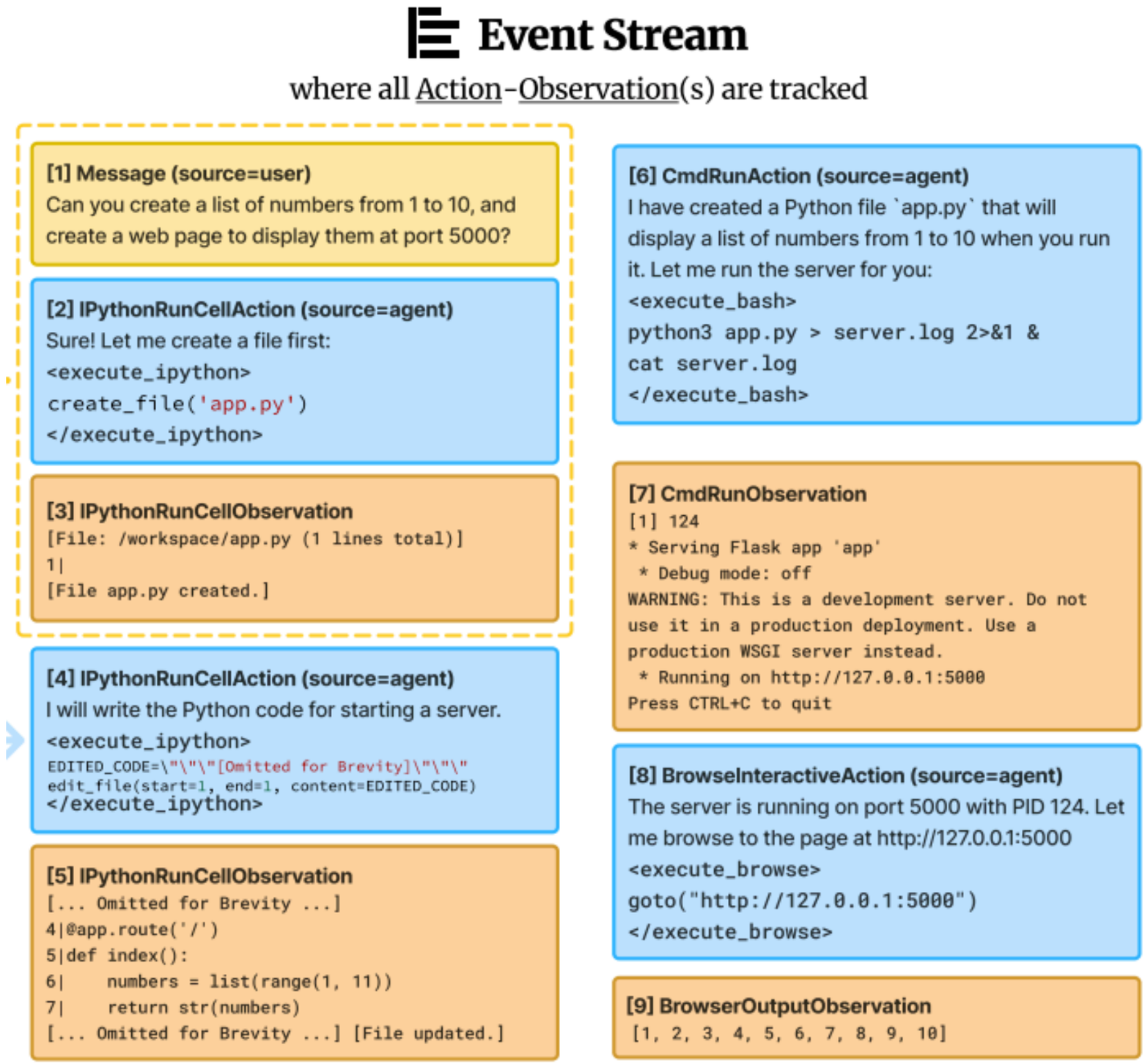
We need to add a check to ensure that all powers are the same. Let's edit the code to include this check.

```
edit 404:407
    elif (total_degree > 3 and
          len(set(k.exp[k.is_Pow])) == 1):
        if all(coeff[k] == 1):
            diop_type = 'general_sum_even'
end_of_edit
```

Example: OpenHands

(Wang et al. 2024)

- Defines “event stream” for coding, execution, and browsing actions/ observations
- Implements SWE-agents style actions as “agent skills” that can be called



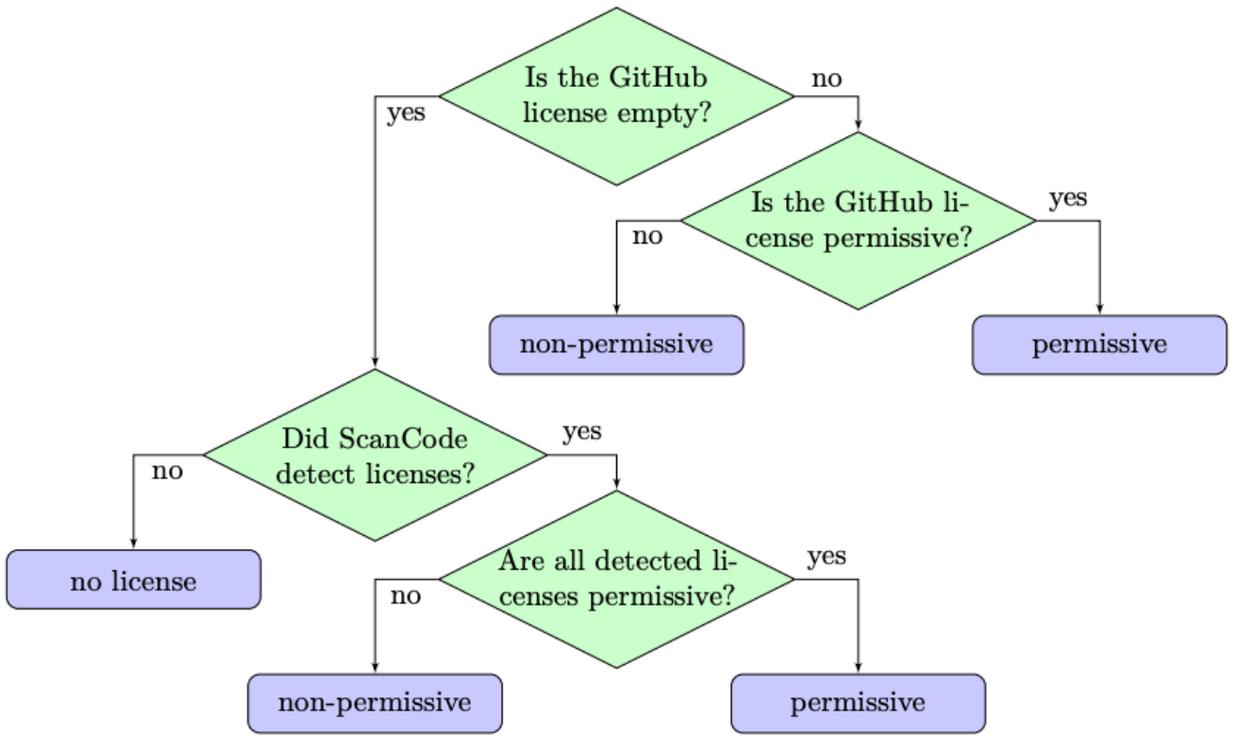
Code-based LLMs

Basic Method: Code-generating LM

- Feed instructions and/or input code to an LM
- Virtually all serious LMs are trained on code nowadays, but some are specialized

Code Data Example: The Stack 2

- Code pre-training dataset w/ license considerations



Language	The-stack-v1-dedup Size (GB)	The-stack-v1-dedup Files (M)	The-stack-v2-dedup Size (GB)	The-stack-v2-dedup Files (M)	The-stack-v2-swh-full Size (GB)	The-stack-v2-swh-full Files (M)
Assembly	1.58	0.25	13.02	0.77	7.74	0.70
Batchfile	0.29	0.25	2.11	1.13	1.02	0.99
C	57.43	8.53	202.05	20.78	114.92	19.18
C#	46.29	10.84	239.89	51.23	169.75	48.49
C++	50.89	6.37	353.89	43.18	211.33	42.23
CMake	0.45	0.19	2.58	1.74	2.27	1.70
CSS	22.61	2.99	161.68	23.87	8.00	1.88
Dockerfile	0.572	0.42	1.27	1.90	1.21	1.88
Fortran	0.17	1.84	4.66	0.27	3.61	0.26
Go	25.74	4.73	54.60	9.30	25.83	8.62
Haskell	2.36	0.54	5.11	1.25	4.17	1.23
HTML	146.76	9.53	2,419.87	90.23	99.09	5.23
Java	89.30	20.15	548.00	154.28	199.68	62.27
JavaScript	141.65	21.11	1,115.42	108.87	199.99	66.91
Julia	1.54	0.30	6.12	0.45	1.83	0.43
Lua	3.28	0.56	33.91	2.35	15.22	2.24
Makefile	1.49	0.66	21.30	4.22	5.19	2.78
Markdown	75.25	21.0	281.04	82.78	244.17	81.42
Perl	2.63	0.39	7.82	1.15	5.66	1.06
PHP	66.84	15.90	224.59	46.03	183.70	45.14
PowerShell	1.25	0.27	3.97	0.68	2.46	0.66
Python	64.30	12.96	233.29	56.93	191.61	56.19
R	0.30	0.04	22.39	5.15	19.05	4.29
Ruby	7.14	3.41	31.70	17.79	23.38	17.51
Rust	9.53	1.38	15.60	2.22	12.43	2.19
Scala	4.86	1.36	12.73	4.45	11.30	4.32
Shell	3.38	22.69	19.82	10.68	13.51	10.01
SQL	12.22	0.99	281.45	5.29	35.75	4.52
Swift	0	0	23.76	7.23	22.32	7.16
TeX	5.44	0.55	35.86	3.19	30.01	2.86
TypeScript	28.82	10.64	61.01	23.85	49.14	23.28
Visual Basic	1.49	0.16	16.63	1.06	7.48	0.81
Total	875.85	181.00	6,457.14	784.30	1,922.82	528.44

Method: Code Infilling

(Fried et al. 2022)

- In code generation, we often want to fill in code
- Solution: train for infilling

Training

Original Document

```
def count_words(filename: str) -> Dict[str, int]:
    """Count the number of occurrences of each word in the file."""
    with open(filename, 'r') as f:
        word_counts = {}
        for line in f:
            for word in line.split():
                if word in word_counts:
                    word_counts[word] += 1
                else:
                    word_counts[word] = 1
    return word_counts
```

Masked Document

```
def count_words(filename: str) -> Dict[str, int]:
    """Count the number of occurrences of each word in the file."""
    with open(filename, 'r') as f:
        <MASK:0> in word_counts:
                    word_counts[word] += 1
                else:
                    word_counts[word] = 1
    return word_counts
<MASK:0> word_counts = {}
        for line in f:
            for word in line.split():
                if word <EOM>
```

Method: Long-context Extension
(see Lu et al. 2024)

- In LMs, it is standard to use RoPE, a method for encoding positional information

- It does not generalize well beyond the training data, but code is long!

$$\mathbf{R}(\boldsymbol{\theta}, i) = \begin{pmatrix} \cos i\theta_1 & -\sin i\theta_1 & \cdots & 0 & 0 \\ \sin i\theta_1 & \cos i\theta_1 & \cdots & 0 & 0 \\ \vdots & & & & \\ 0 & 0 & \cdots & \cos i\theta_{\frac{d_k}{2}} & -\sin i\theta_{\frac{d_k}{2}} \\ 0 & 0 & \cdots & \sin i\theta_{\frac{d_k}{2}} & \cos i\theta_{\frac{d_k}{2}} \end{pmatrix}$$

- It has a parameter, typically $\theta_j = b^{-\frac{2j}{d_k}}$ with b=10000

- **Position interpolation:** Multiply θ by a constant scaling factor (e.g. C_{short}/C_{long})

- **Neural tangent kernel:** Scale low-frequency components, but maintain high-frequency components

Lots of Available Information for Coding!

- Current code context
- Description of issue to fix
- Repo context
- Open tabs

Example: Copilot Prompting Strategy (Thakkar 2023)

- **Extract prompt** given current doc and cursor position
- Identify **relative path and language**
- Find **most recently accessed** 20 files of the same language
- **Include:** text before, text after, similar files, imported files, metadata about language and path
- **TL;DR:** lots of prompt engineering to get most useful context tin the prompt

File Localization

LLM-based Localization

- Finding the correct files given user intent

What problem or use case are you trying to solve?
When in confirmation mode it's not possible to give instructions in between steps. You have to reject an action and it seems like it doesn't know that the action was rejected.

Describe the UX of the solution you'd like
The simplest would be to have a third option, confirm action and wait. This way the action is confirmed but before it tries to take the next step you are able to give some feedback. Also if it somehow knows the action was rejected that would be helpful as well so when you do reject an action it knows that action wasn't taken.

https://github.com/All-Hands-AI/OpenHands/issues/4259

- Which JavaScript file should I modify?
- Analogous to environment understanding / exploration problems in other agents

Solution 1: Offload to the User

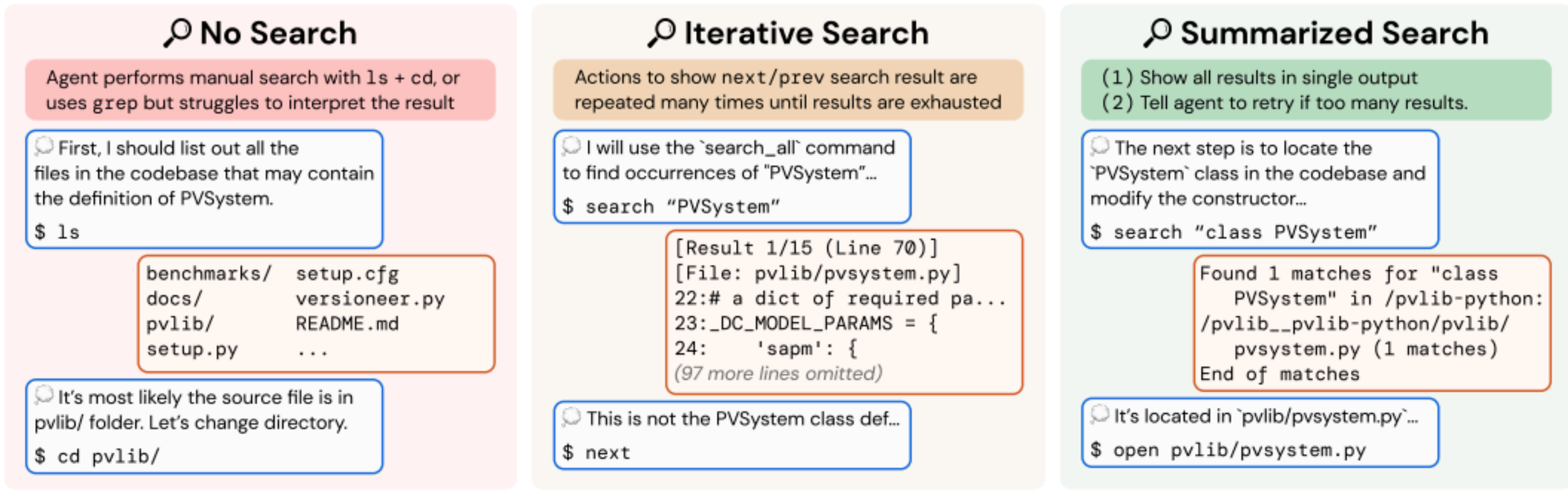
- Experienced users familiar with prompting and the project can specify which files to use

In **`.github/workflows/openhands-resolver.yml`** and **`.github/workflows/openhands-resolver-experimental.yml`**, we should check to make sure that all required environment variables are set before running any additional workflows. If all of the variables are not set, we can fail immediately with an error.

https://github.com/All-Hands-AI/openhands-resolver/issues/146

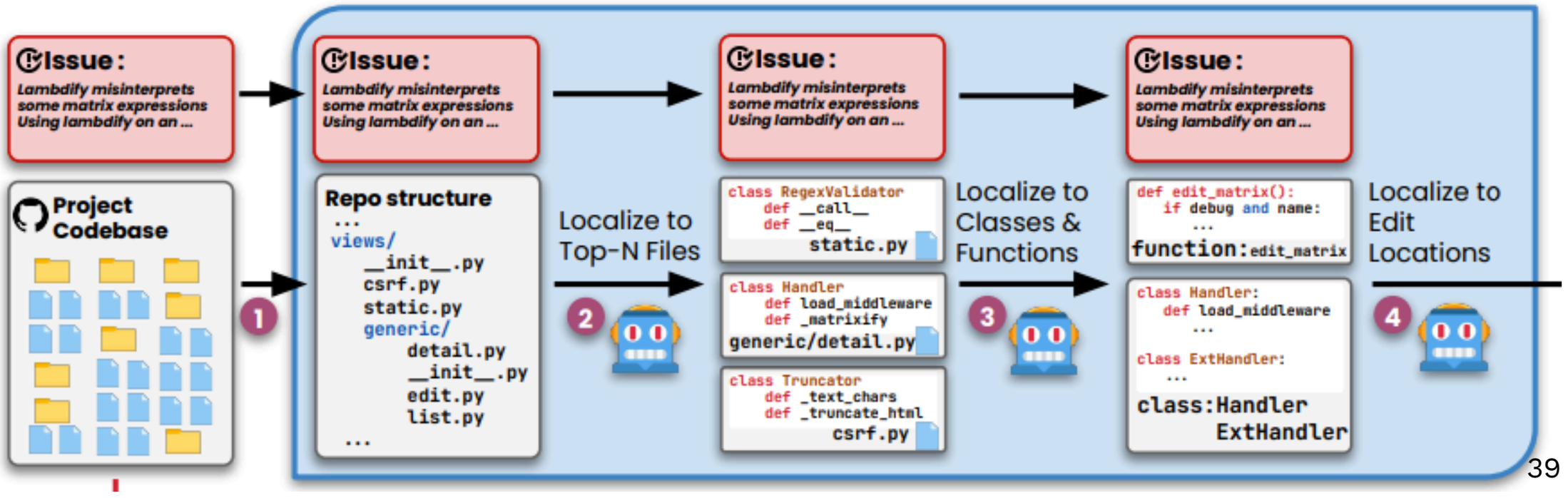
Solution 2:
Prompt the Agent w/ Search Tools

- e.g. SWE-agent provides a tool for searching repositories



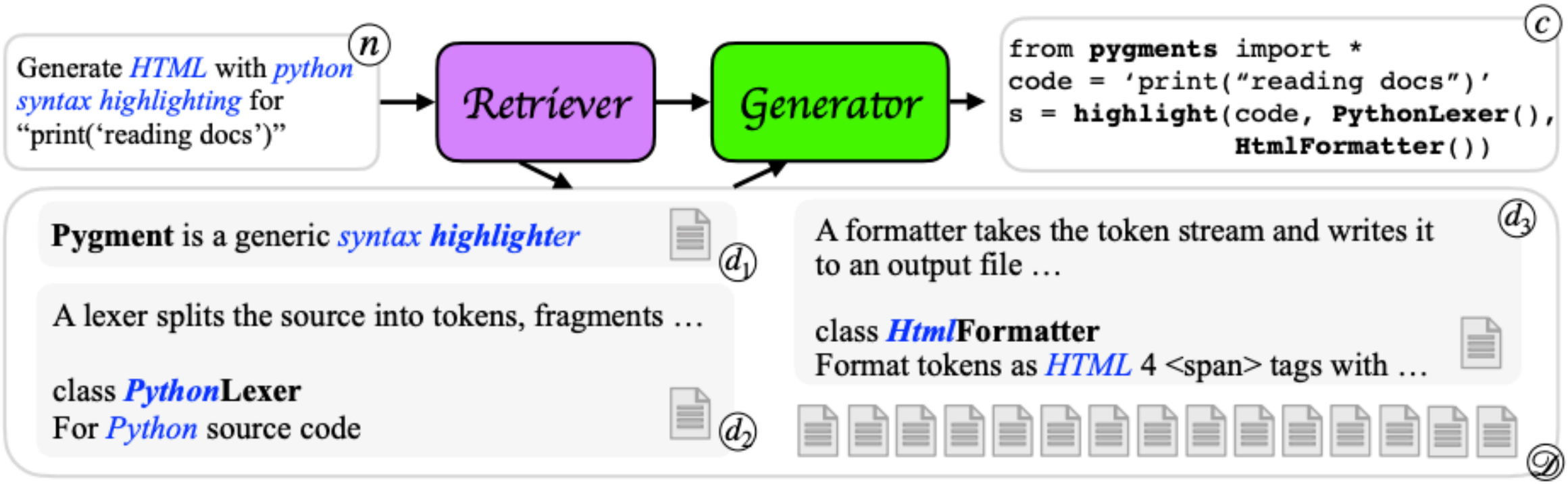
Solution 3: A-priori Map the Repo

- Create a map of the repo and prompt agent with it
- Aider repomap creates a tree-structured map of the repo
- Agentless (Xia et al. 2024) does a hierarchical search for every issue



Solution 4: Retrieval-augmented Code Generation

- Retrieve similar code, and fill it in with a retrieval-augmented LM (Hayati et al. 2018)
- Particularly, in code there is also documentation, which can be retrieved (Zhou et al. 2022)



- Unsolved issue: when to perform RAG in agent

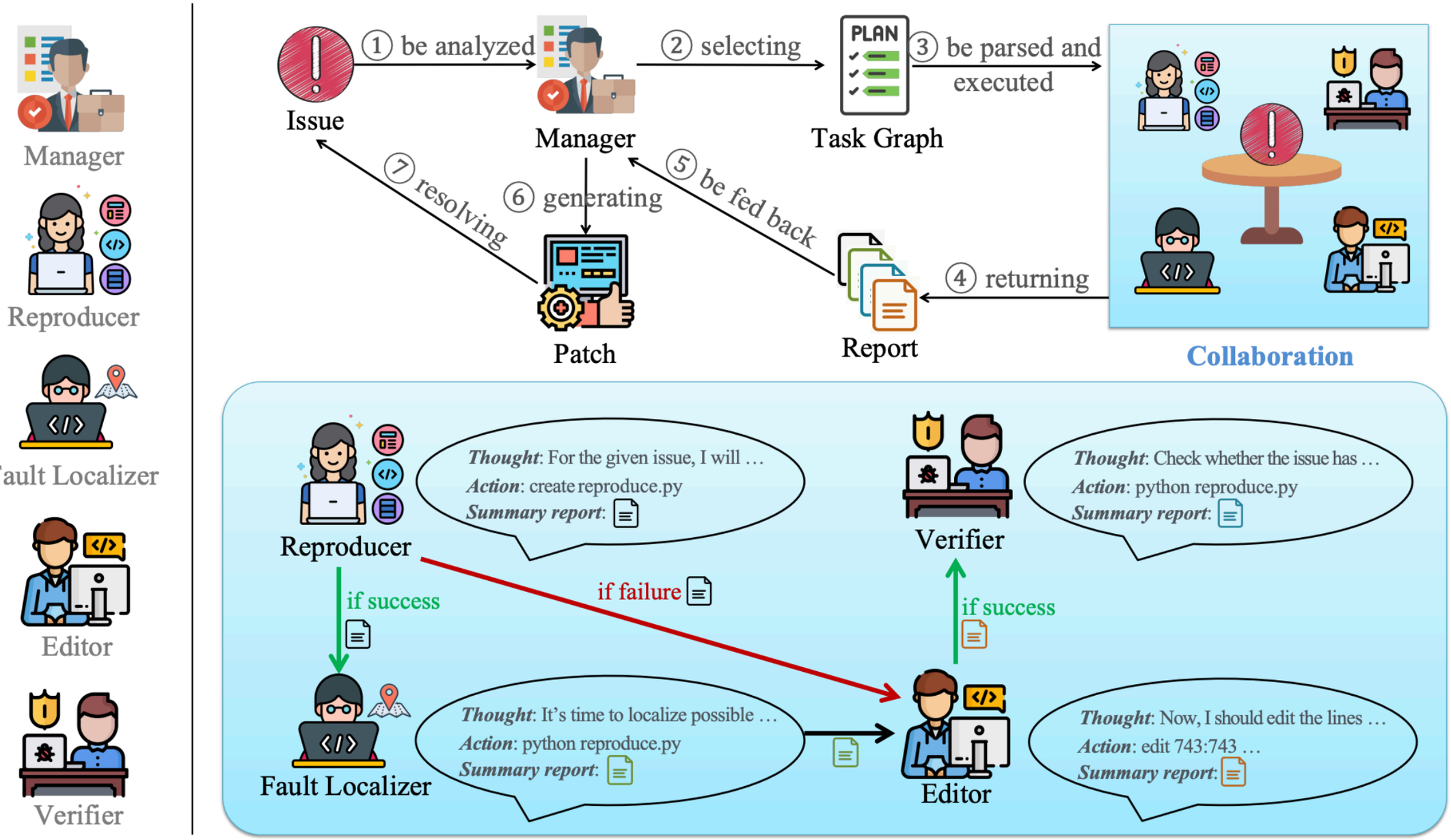
Planning and Error Recovery

Hard-coded Task Completion Process

- e.g. Agentless (Xie et al. 2024) has a hard-coded progress of
  - File Localization
  - Function Localization
  - Patch Generation
  - Patch Application

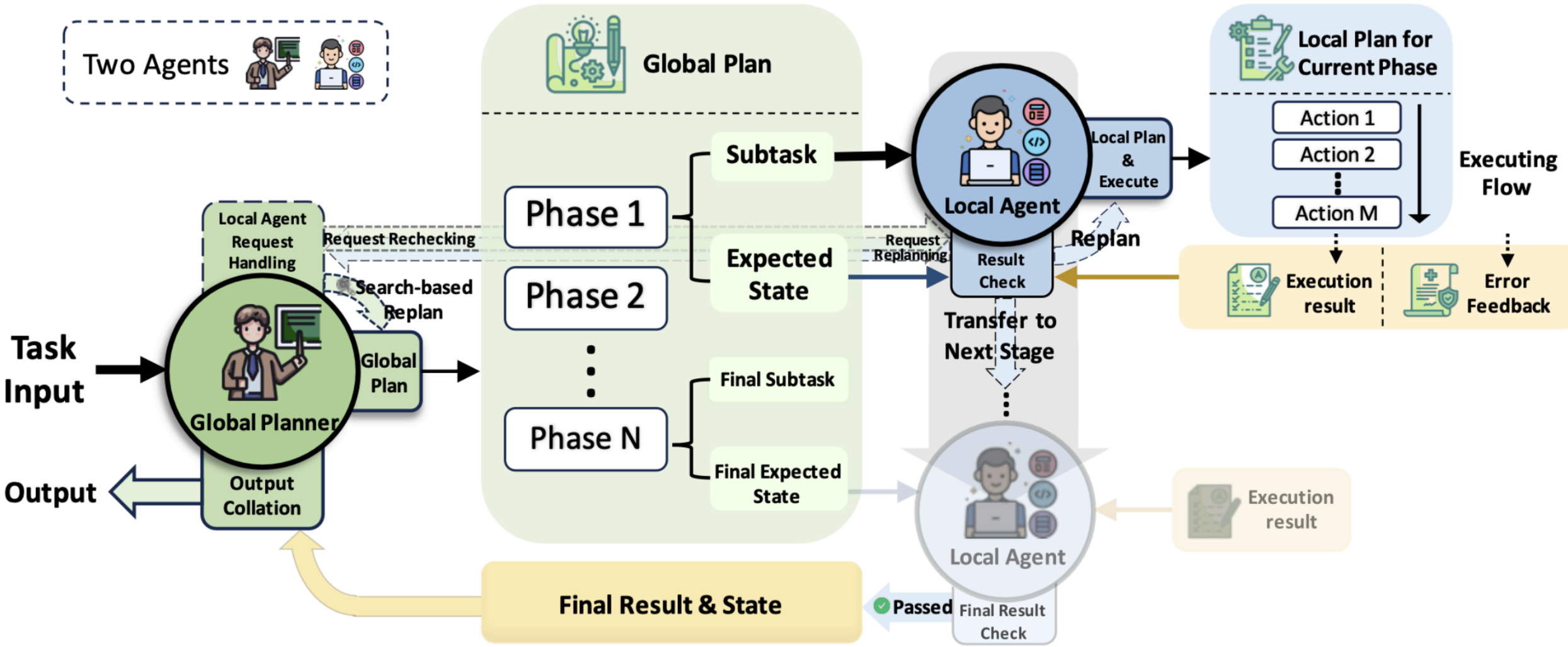
LLM-Generated Plans

- LLM-generated planning step, then one or more executors
- CodeR (Chen et al. 2024)



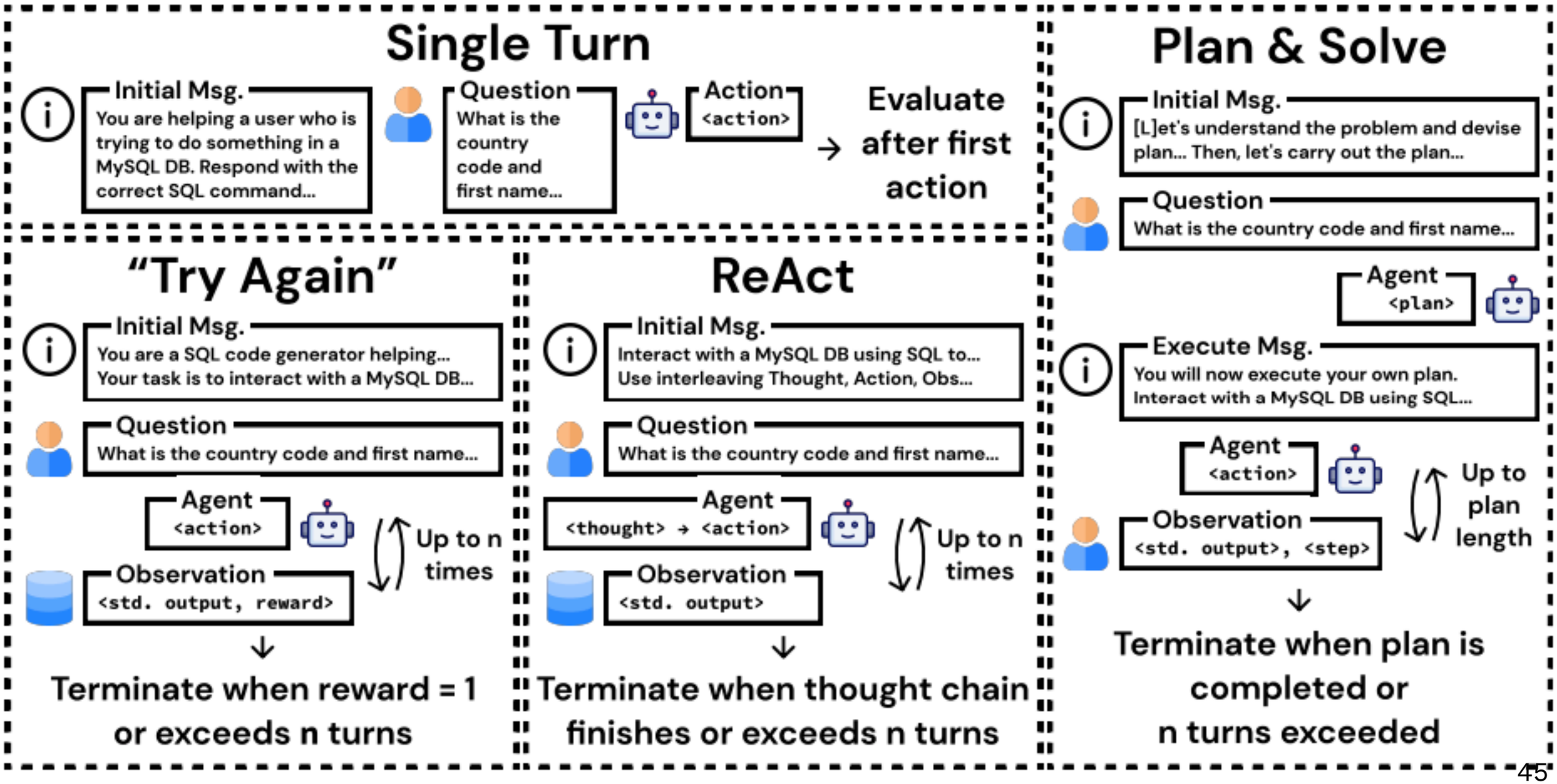
Planning and Revisiting

- CoAct goes back and fixes (Hou et al. 2024)



Fixing Based on Error Messages

- e.g. InterCode (Yang et al. 2023)



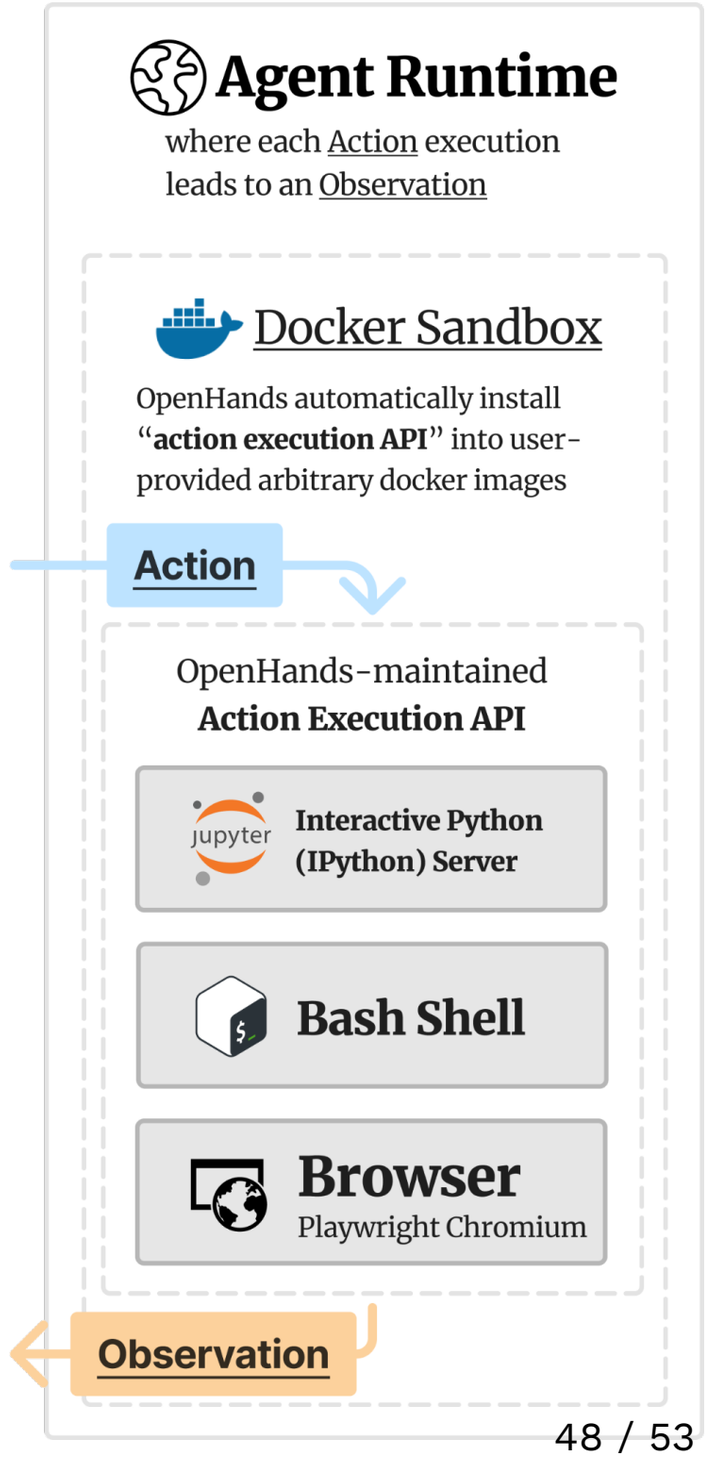
Safety

Coding Models can Cause Harm!

- By accident
  - The coding model accidentally pushes to your main branch
  - The coding model is told to “make the tests pass”, so it deletes the tests
- Intentionally
  - Coding agents can be used for hacking (Yang et al. 2023)

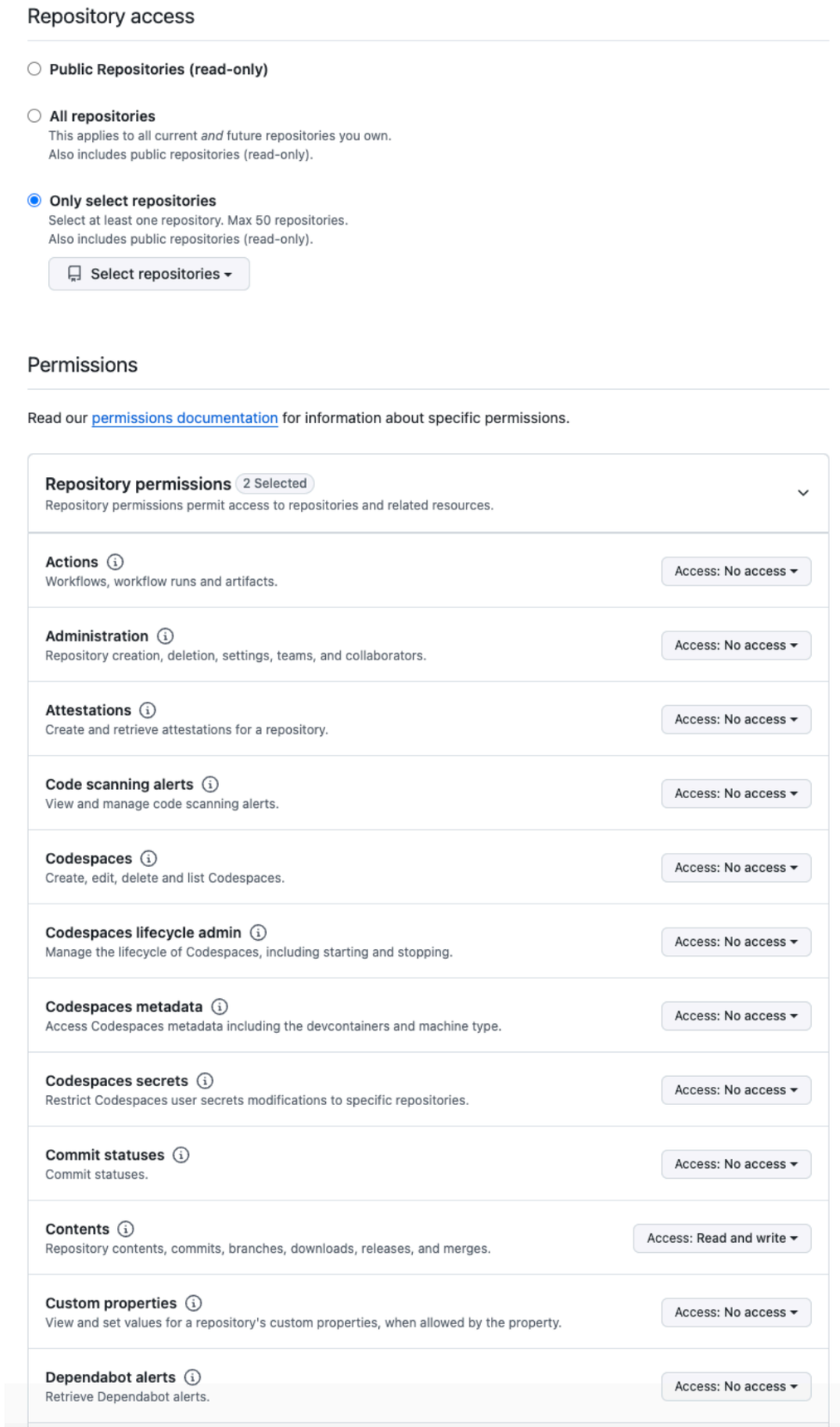
Safety Mitigation 1: Sandboxing

- We can improve safety by **limiting the execution environment**
- e.g. OpenHands execute all the actions in Docker sandboxes



Safety Mitigation 2: Credentialing

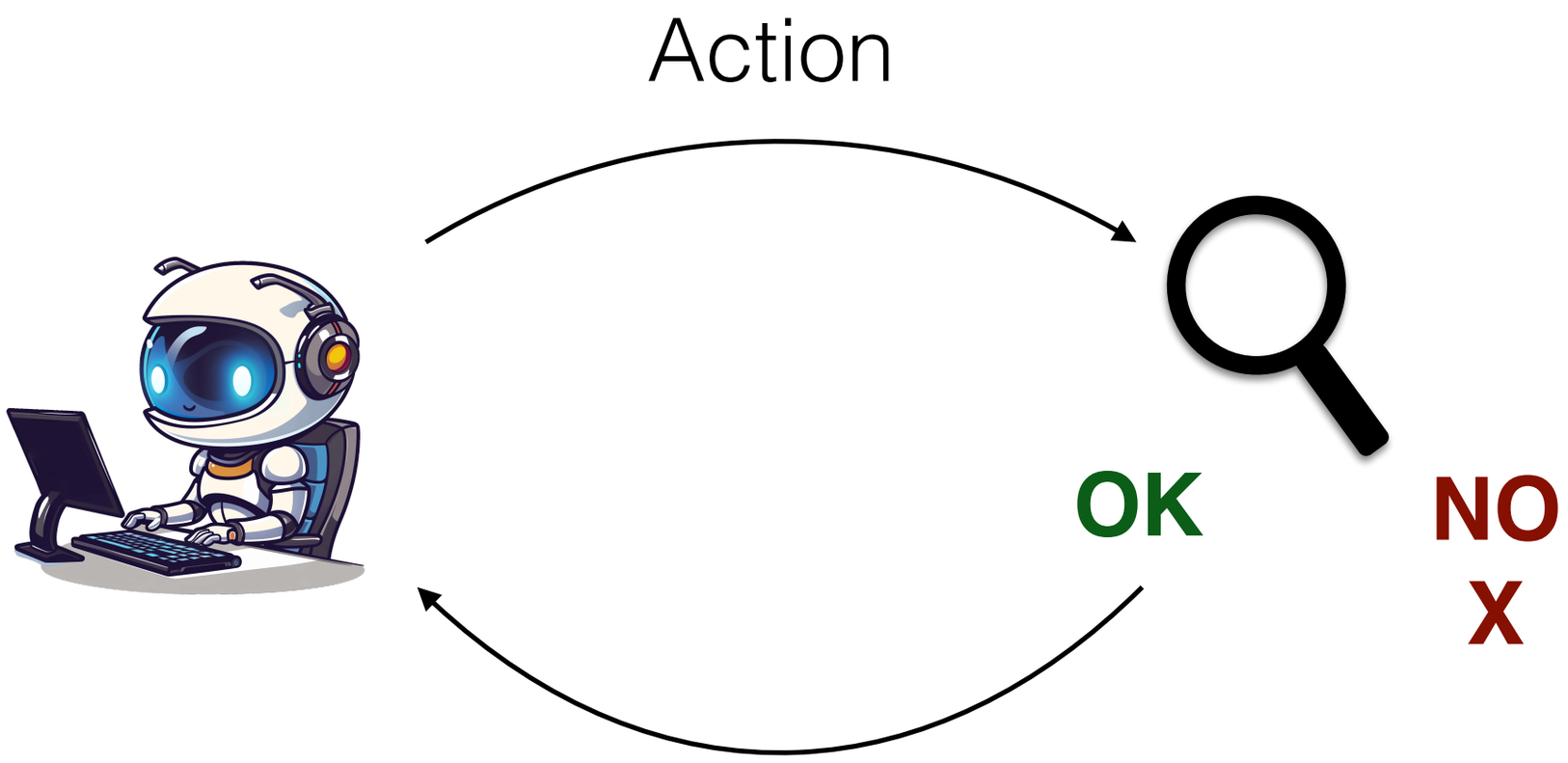
- The *principle of least privilege*
- Example: GitHub access tokens



https://github.com/settings/tokens?type=beta

Safety Mitigation 3: Post-hoc Auditing

- e.g. OpenHands security analyzer



- Using LMs, analysis, or both

Conclusion

Summary

- Copilots already very useful, code agents getting there
- Current challenges: code LLMs, editing, localization, planning, safety
- Future directions:
  - Agentic training methods
  - Human-in-the-loop
  - Broader software tasks than coding
- Thanks! And you can try out agents yourself

  https://github.com/All-Hands-AI/OpenHands

Questions?