

CS11-711 Advanced NLP

# Agents

Sean Welleck

**Carnegie  
Mellon  
University**



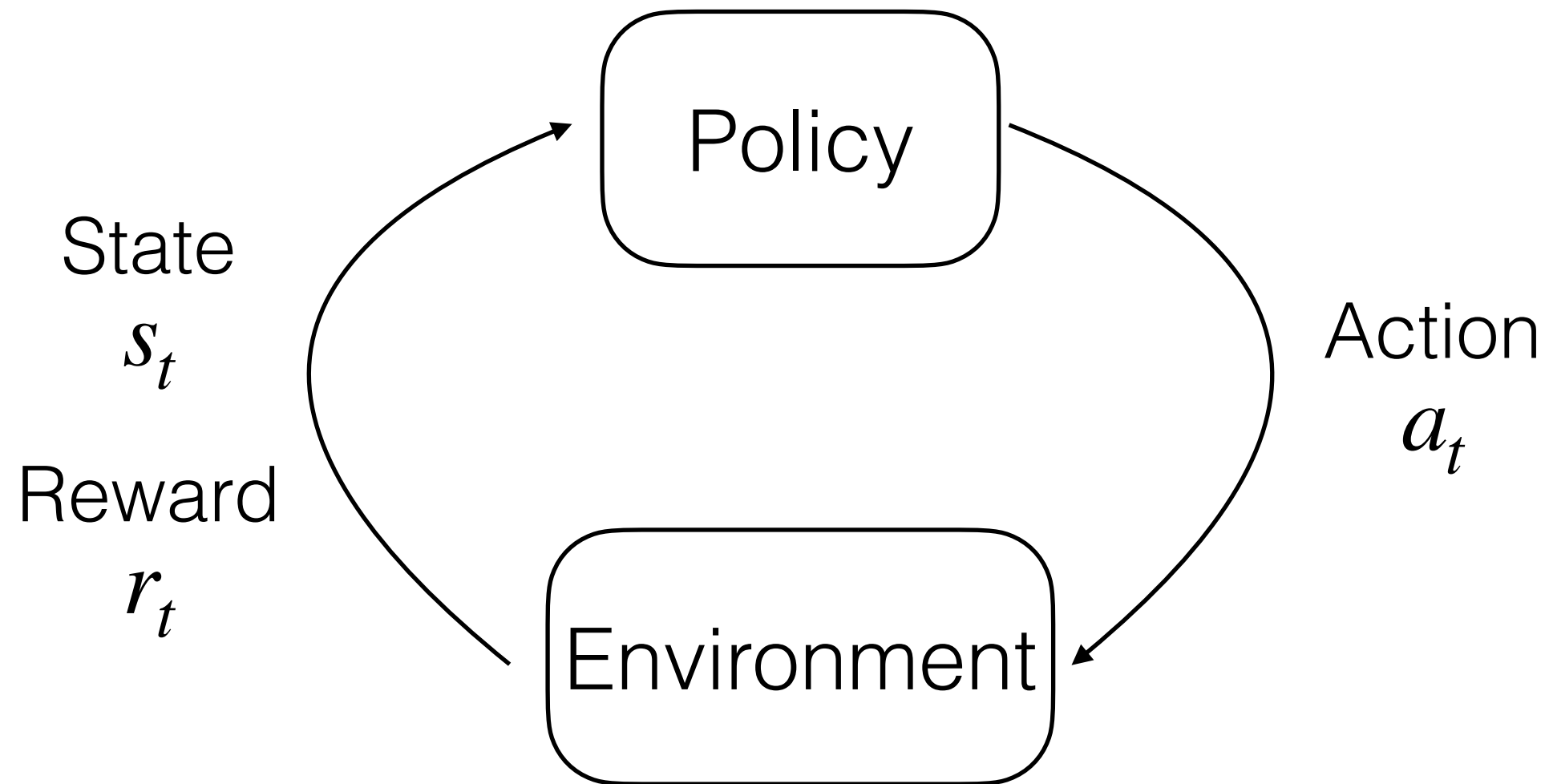
<https://cmu-l3.github.io/anlp-fall2025/>

<https://github.com/cmu-l3/anlp-fall2025-code>

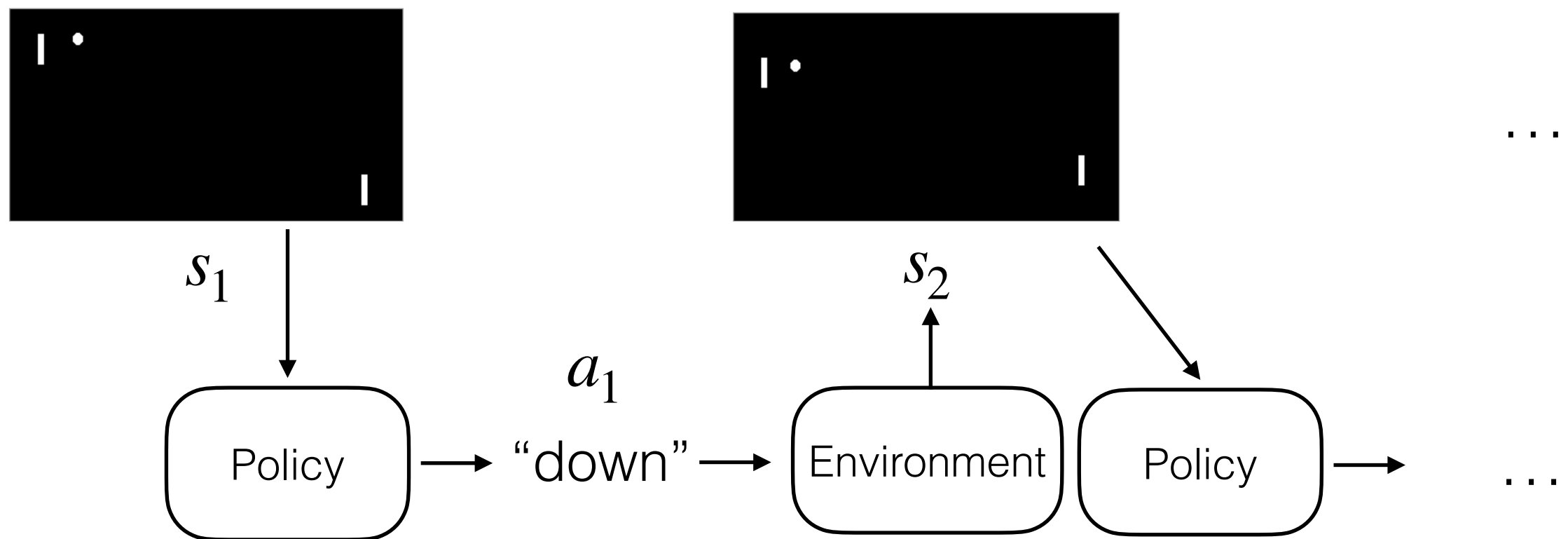
# Recap: language models



# Recap: RL MDP



# Recap: RL MDP





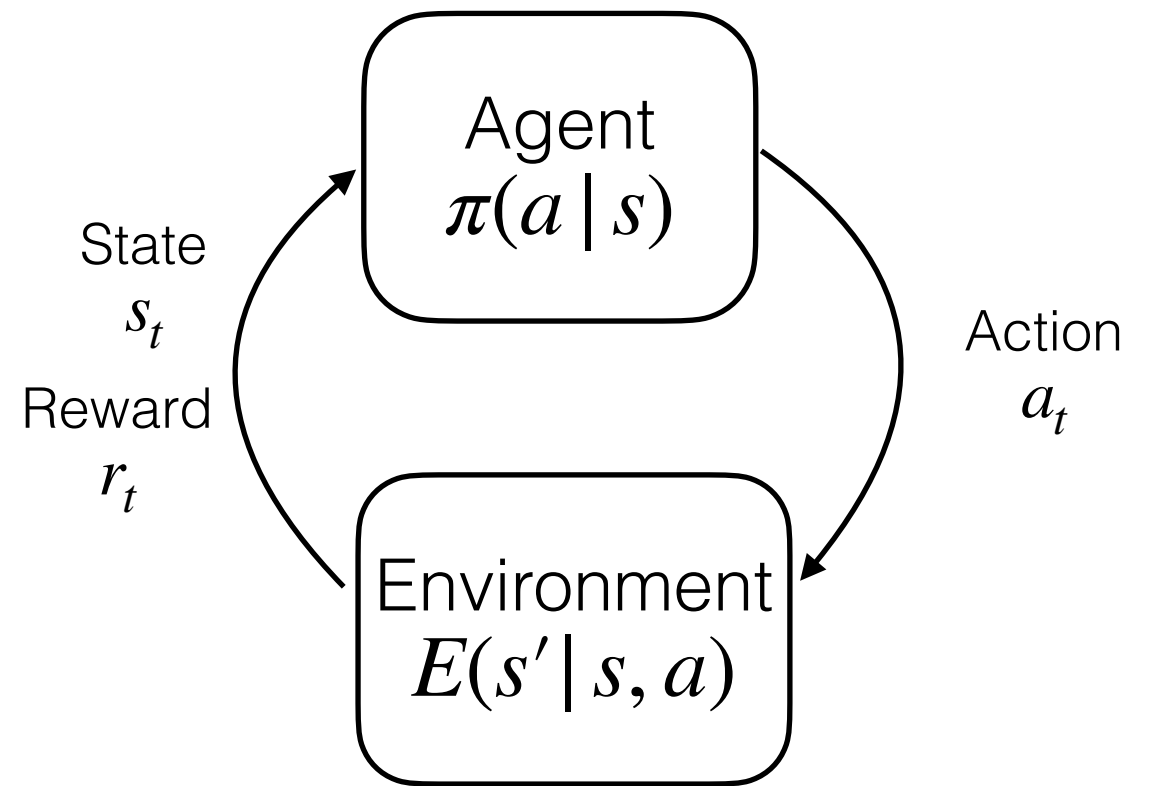
# Today: (LM) Agents

- What is an agent?
- Agent environments
- Agent design patterns
- Agents and RL

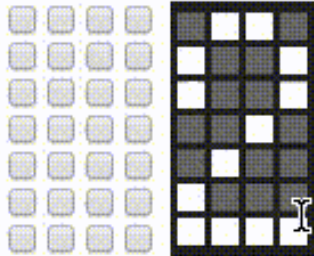
Code examples: [https://github.com/cmu-l3/anlp-fall2025-code/tree/main/17\\_agents](https://github.com/cmu-l3/anlp-fall2025-code/tree/main/17_agents)

# What is an Agent?

- An agent is simply a policy!
  - Input: state
  - Output: action
- Operates in an environment

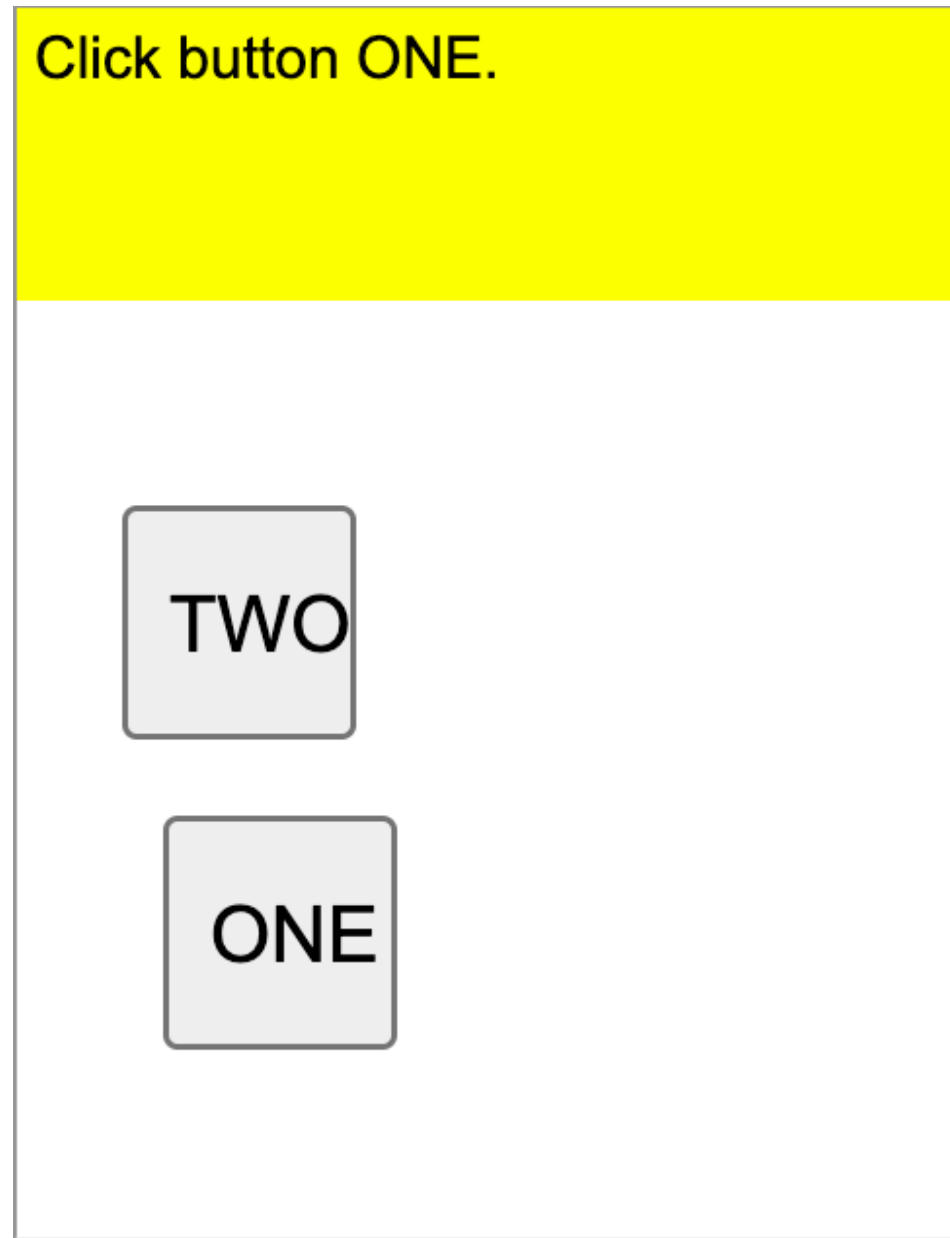


# Example: simple web environments

|                                                                                                                                                                             |                                                                                                       |                                                                                                                                                                                                               |                                                                                             |                                                                                                                                                                                 |                                                                                                                                                                                                 |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <p>Move the cube around so that "5" is the active side facing the user.</p>  <p>Submit</p> | <p>Set the sliders to the combination [13,20,13] and submit.</p> <p>11<br/>6<br/>17</p> <p>Submit</p> | <p>Draw the number "2" in the checkboxes using the example on the right and press Submit when finished.</p>  <p>Submit</p> | <p>Select 5 with the spinner and hit Submit.</p> <p>Select a value:<br/>0</p> <p>Submit</p> | <p>Keep your mouse inside the circle as it moves around.</p>                                 | <p>Move the cube around so that "4" is the active side facing the user.</p>  <p>Submit</p>                   |
| <p>Copy the text in the textarea below and paste it into the textbox.</p> <p>Enim. Elementum, tortor ullamcor</p> <p>Submit</p>                                             | <p>Select 09/23/2016 as the date and hit submit.</p> <p>Date: </p> <p>Submit</p>                      | <p>Drag all rectangles into the black box.</p>  <p>Submit</p>                                                            | <p>Enter "qu" into the text field and press Submit.</p> <p>qu</p> <p>Submit</p>             | <p>Select all the shades of blue and press Submit.</p>  <p>Submit</p>                      | <p>Find the 4th word in the paragraph, type that into the textbox and press "Submit".</p> <p>Non arcu ut ultricies est. Gravidam, porta erat nulla eget condimentum posuere a</p> <p>Submit</p> |
| <p>Enter an item that starts with "Tuni".</p> <p>Tags: </p> <p>Submit</p>                                                                                                   | <p>Enter "Vb8" into the text field and press Submit.</p> <p>Submit</p>                                | <p>Focus into the 1st input textbox.</p> <p></p> <p></p> <p></p>                                                                                                                                              | <p>Focus into the textbox.</p> <p></p>                                                      | <p>Move the cube around so that "2" is the active side facing the user.</p>  <p>Submit</p> | <p>Select 5Gi and click Submit.</p> <p>5Gi<br/>sPUt</p> <p>Submit</p>                                                                                                                           |

# Example: click environment

- Environment:
  - Simple webpage
- Task:
  - Click indicated button



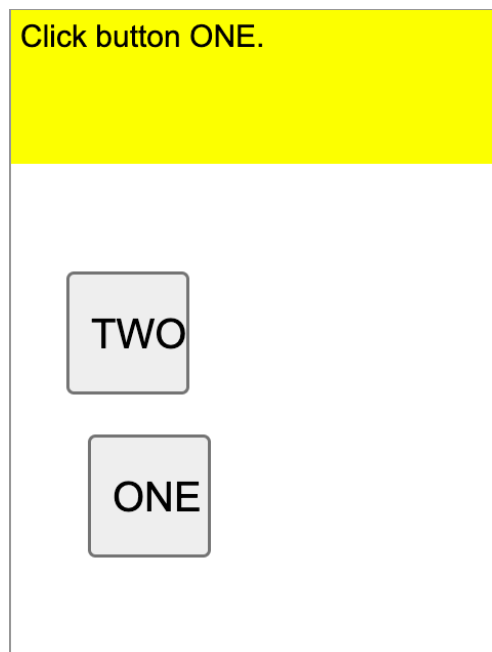
# Example: click environment

```
env = gymnasium.make('miniwob/click-test-2-v1', render_mode='human')
try:
    observation, info = env.reset(seed=41)
    for step in range(100):
        action = agent(observation, env)
        observation, reward, terminated, truncated, info = env.step(action)

        if terminated:
            observation, info = env.reset()
```

# Example: click environment

- Observation/state
- Image
- DOM elements

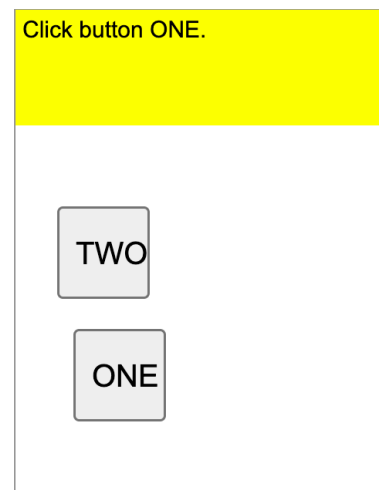


```
'tag': 'div',
'text': '',
'top': array([50.], dtype=float32),
'value': '',
'width': array([160.], dtype=float32)},
{'bg_color': array([0.9372549, 0.9372549, 0.9372549, 1.], dtype=float32),
'classes': '',
'fg_color': array([0., 0., 0., 1.], dtype=float32),
'flags': array([0, 0, 0, 1], dtype=int8),
'height': array([40.], dtype=float32),
'id': 'subbtn',
'left': array([45.], dtype=float32),
'parent': 3,
'ref': 4,
'tag': 'button',
'text': 'ONE',
'top': array([76.], dtype=float32),
'value': '',
'width': array([40.], dtype=float32)},
{'bg_color': array([0.9372549, 0.9372549, 0.9372549, 1.], dtype=float32),
'classes': '',
'fg_color': array([0., 0., 0., 1.], dtype=float32),
'flags': array([0, 0, 0, 1], dtype=int8),
'height': array([40.], dtype=float32),
'id': 'subbtn2',
'left': array([30.], dtype=float32),
'parent': 3,
'ref': 5,
'tag': 'button',
'text': 'TWO',
'top': array([109.], dtype=float32),
'value': '',
'width': array([40.], dtype=float32)}
```

# Example: click environment

- Actions

- MOVE\_COORDS <x, y>,  
...  
CLICK\_ELEMENT <REF>,  
...



| Name                         | Description                                                                                           |
|------------------------------|-------------------------------------------------------------------------------------------------------|
| NONE                         | Do nothing for the current step.                                                                      |
| MOVE_COORDS                  | Move the cursor to the specified coordinates.                                                         |
| CLICK_COORDS                 | Click on the specified coordinates.                                                                   |
| DBLCLICK_COORDS              | Double-click on the specified coordinates.                                                            |
| MOUSEDOWN_COORDS             | Start dragging on the specified coordinates.                                                          |
| MOUSEUP_COORDS               | Stop dragging on the specified coordinates.                                                           |
| SCROLL_UP_COORDS             | Scroll up on the mouse wheel at the specified coordinates.                                            |
| SCROLL_DOWN_COORDS           | Scroll down on the mouse wheel at the specified coordinates.                                          |
| CLICK_ELEMENT                | Click on the specified element using JavaScript.                                                      |
| PRESS_KEY                    | Press the specified <a href="#">key</a> or <a href="#">key combination</a> .                          |
| TYPE_TEXT                    | Type the specified string.                                                                            |
| TYPE_FIELD                   | Type the value of the specified task field.                                                           |
| FOCUS_ELEMENT_AND_TYPE_TEXT  | Click on the specified element using JavaScript, and then type the specified string.                  |
| FOCUS_ELEMENT_AND_TYPE_FIELD | Click on the specified element using JavaScript, and then type the value of the specified task field. |



# A simple rule-based agent

- Write a program that:
  - Finds the DOM element with text 'ONE'
  - Get its REF
  - Output the action  
CLICK\_ELEMENT <REF>

Click button ONE.

TWO

ONE



# A simple rule-based agent

```
def agent(observation, env):  
    # Find the HTML element with text "ONE"  
    one_element = None  
    for element in observation['dom_elements']:  
        if element['text'] == "ONE":  
            one_element = element  
            break  
  
    # Click on the element.  
    action = env.unwrapped.create_action(  
        ActionTypes.CLICK_ELEMENT, ref=one_element["ref"]  
    )  
    return action
```

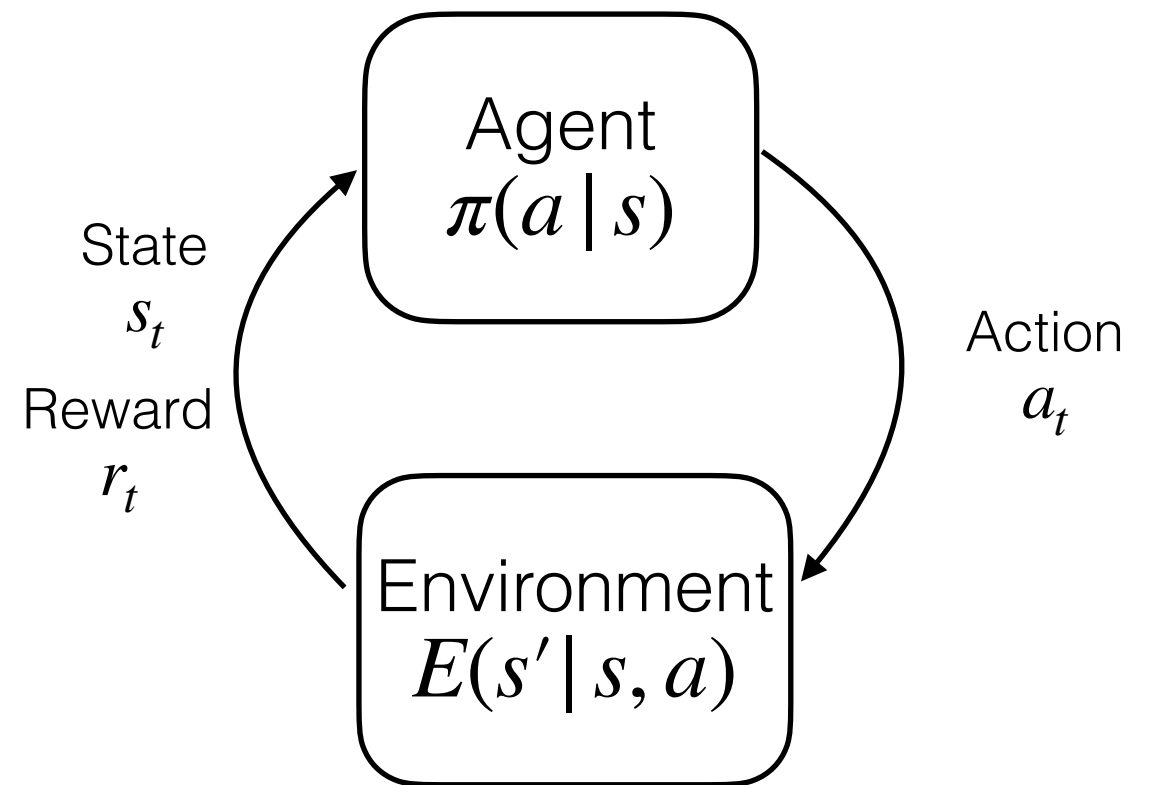
Click button ONE.

TWO

ONE

# What is an Agent?

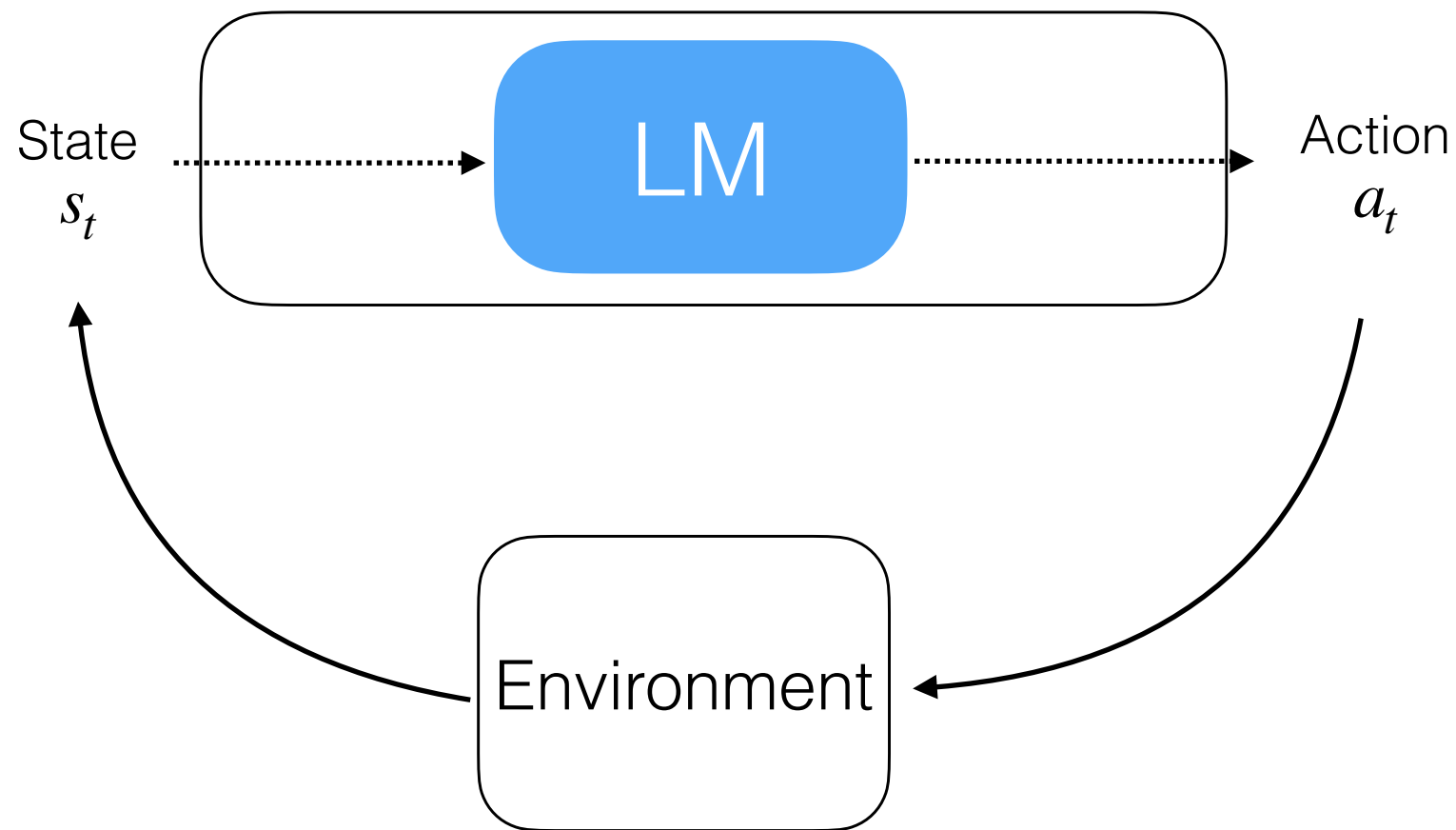
- An agent is simply a policy!
- Input: state
- Output: action
- Operates in an environment



Our rule-based agent: **only works for one narrow task**

# What is a **LLM** Agent?

- An agent where a language model helps determine actions



# Example

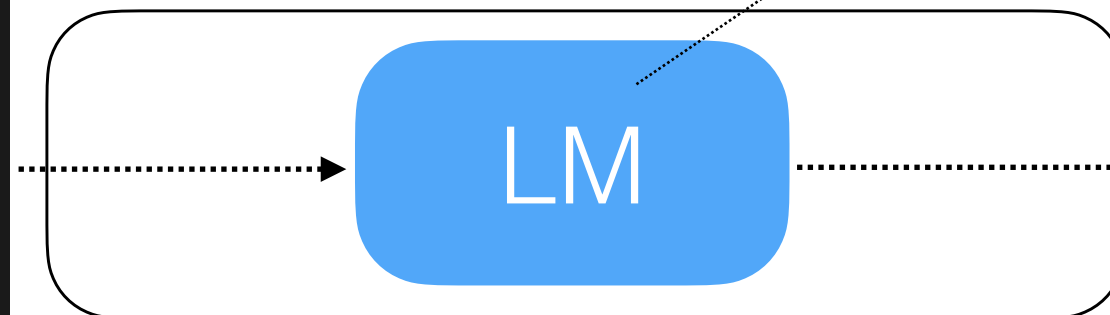
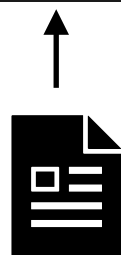
## Prompt

```
PROMPT = """You are on a webpage.  
  
Here are your available actions:  
  
NONE  
Do nothing for the current step.  
  
MOVE_COORDS [x, y]  
Move the cursor to the specified coordinates.  
  
CLICK_COORDS [x, y]  
Click on the specified coordinates.  
  
CLICK_ELEMENT <element_ref>  
Click on the specified element using JavaScript.  
  
----  
Observation:  
You see the following elements:  
{}  
  
----  
Task:  
Here is your task:  
{}
```

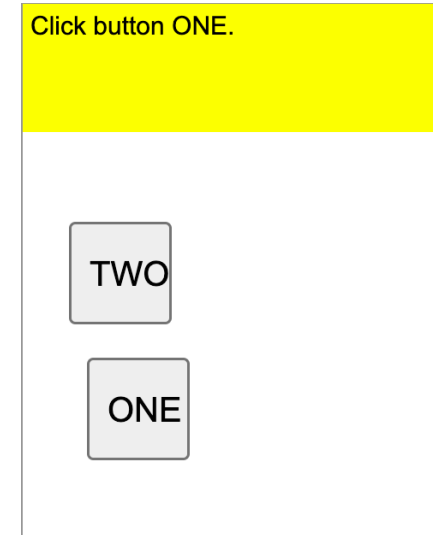
## Call LLM and parse response

```
response = client.models.generate_content(  
    model="gemini-2.0-flash", contents=PROMPT  
)  
out = response.text  
  
# Parse the action type and args  
out = out.strip().split()  
if out[0] == "CLICK_ELEMENT":  
    out = int(out[1])  
    # Click on the element.  
    action = env.unwrapped.create_action(  
        ActionTypes.CLICK_ELEMENT, ref=out  
    )
```

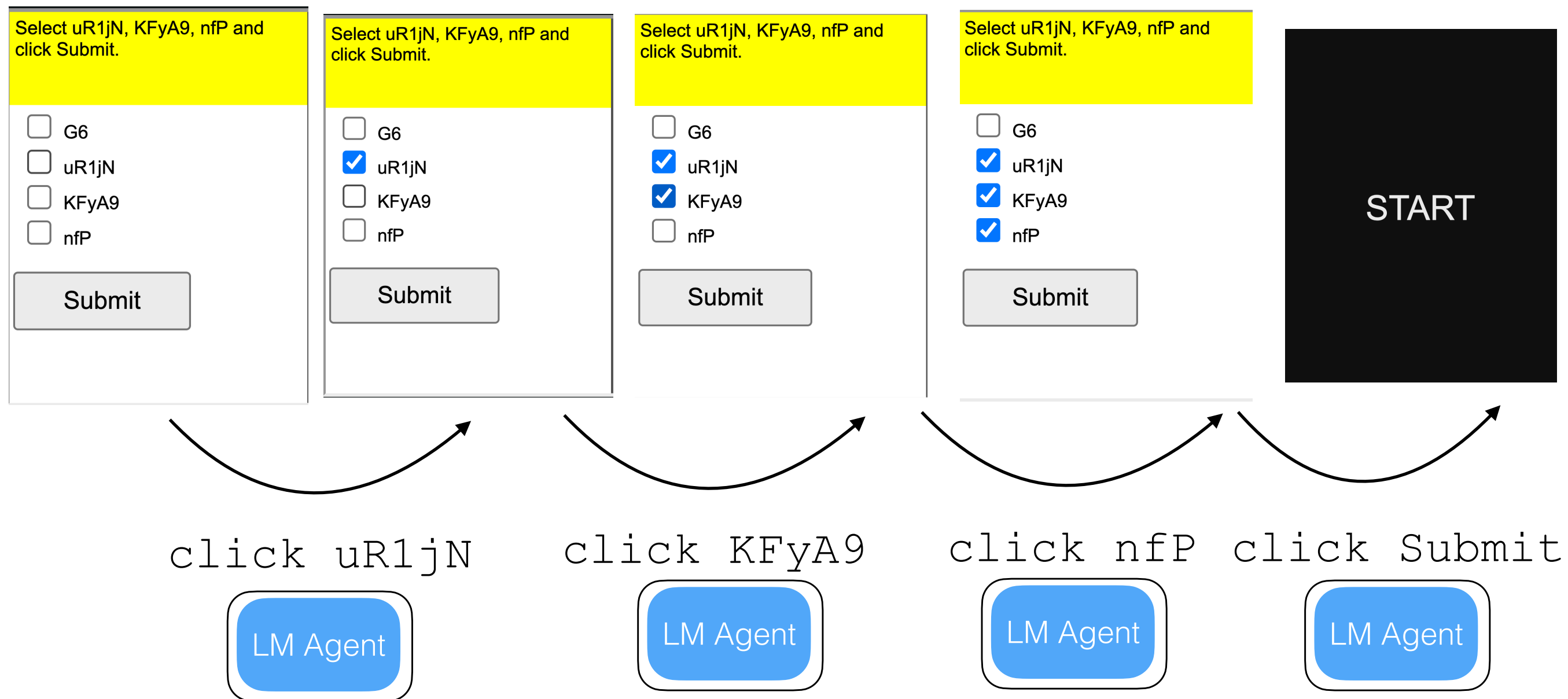
DOM  
Elements



CLICK\_ELEMENT 4

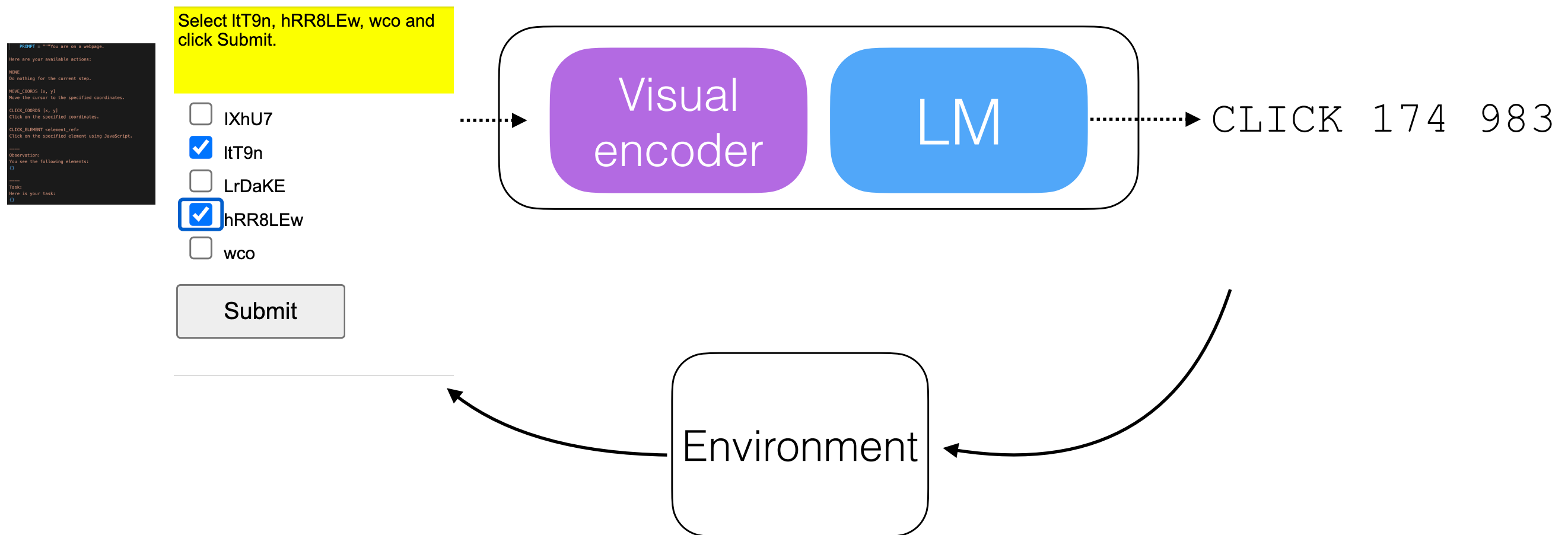


# Example



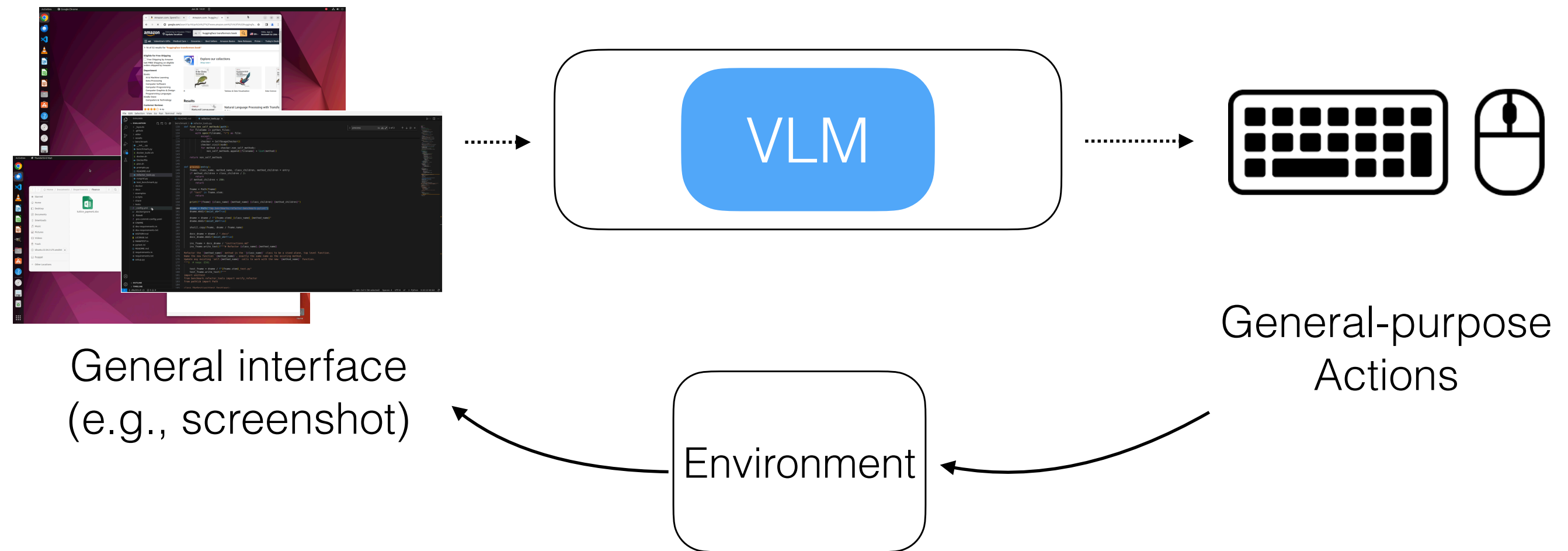
# Vision Language Model (VLM) Agent

- An agent where a vision language model helps determine actions



# Computer-use agent

- An agent with a general-purpose observation and action space (i.e., works for any computer task)





# Recap: what is an agent?

- An agent is a policy  $\pi(a | s)$  in an environment
- Text agents (e.g., LM agents)
  - Operate with text inputs and outputs
- Visual agents (e.g., VLM agents)
  - Additionally operate with images
  - Example: Computer-use agents
    - A task-agnostic visual agent for computer tasks



# Lots of recent interest in agents

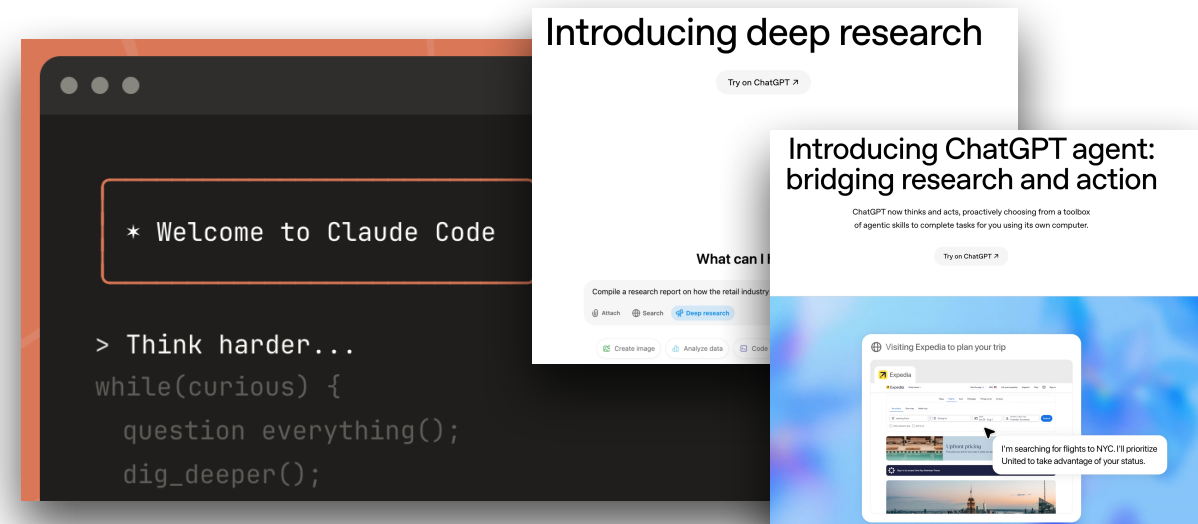
- **Software engineering:** e.g., “here’s a GitHub PR for my repository, can you close it?”
- **Research:** e.g., “compile a research report on how the AI industry has changed in the last 4 months”
- **Web:** e.g., ““Buy the cheapest color photo printer and send it to Emily's place (address in image).”

CHIPS

## Nvidia CEO Says 2025 Is the Year of AI Agents

By Tae Kim [Follow](#)

Jan 07, 2025, 5:40 pm EST



# Today: (LM) Agents

- What is an agent?
- **Agent environments**
- Agent design patterns
- Agents and RL

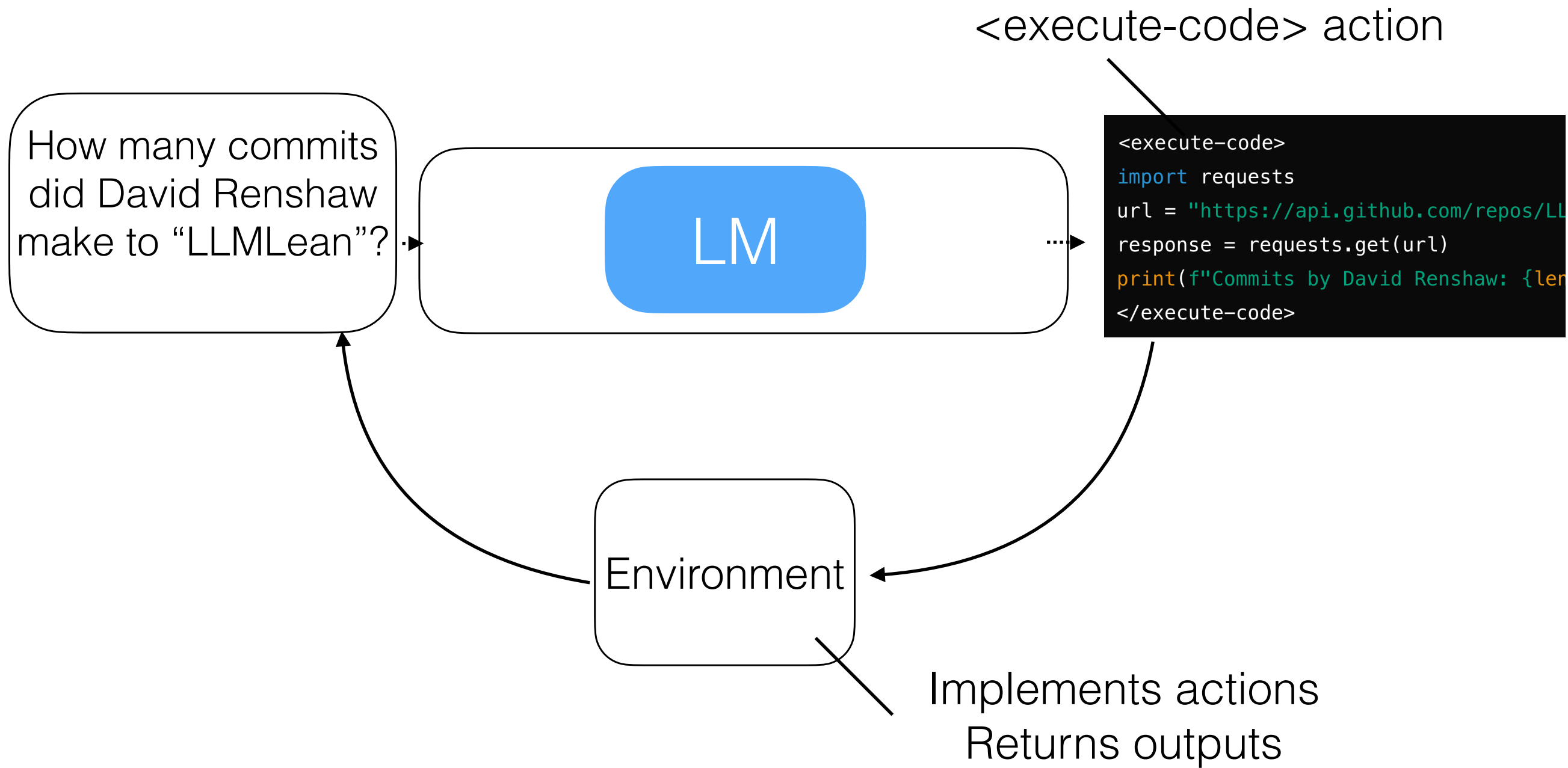
# Agent environments

- Text agents
- Visual agents
- Computer-use agents

# Domain-specific text environments

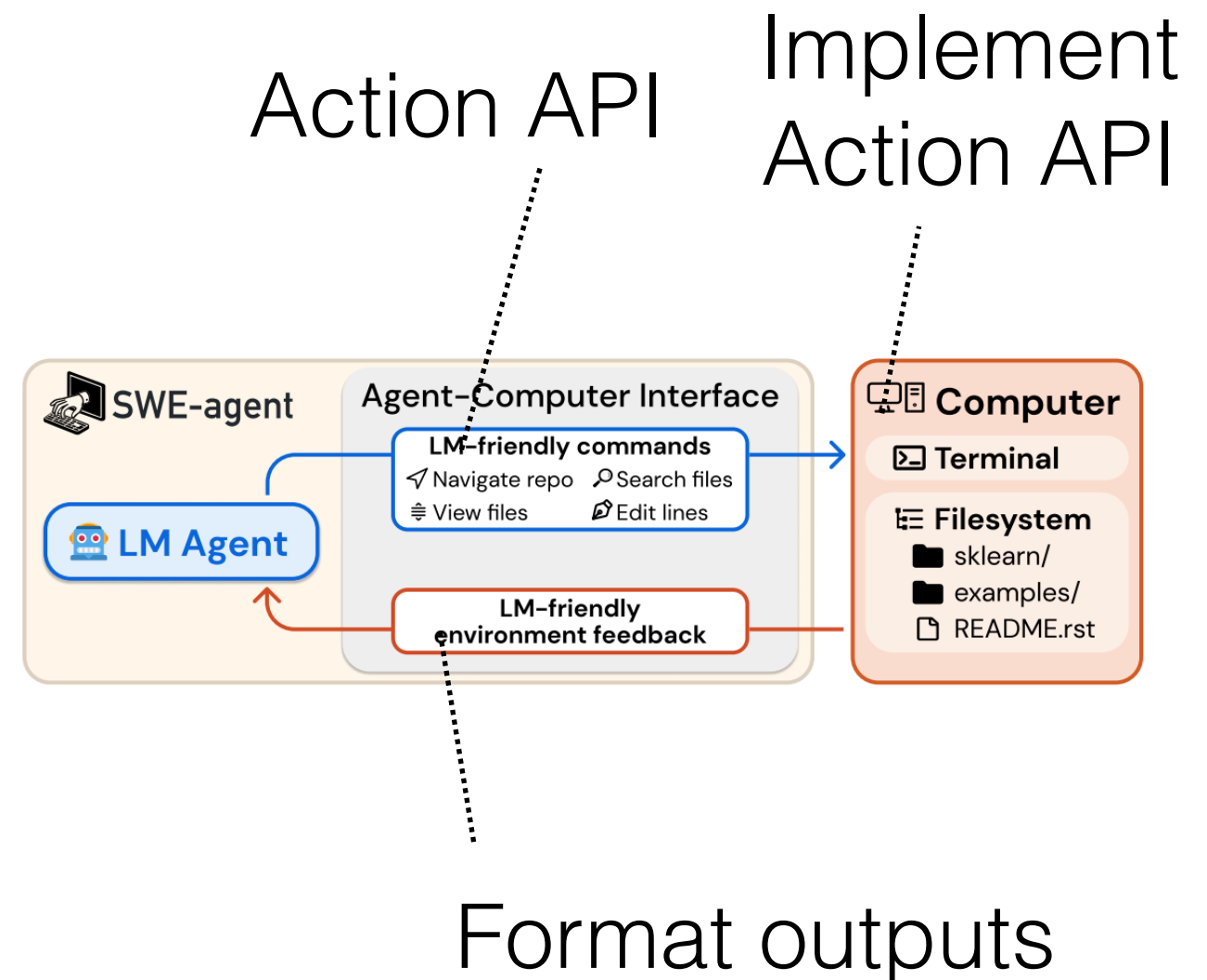
- Basic idea: engineer an environment for a domain
- Example: software engineering agent
  - Actions: search\_repo, execute code, edit file
  - Environment: implements the action API
  - Observations: returns outputs

# Example



# Example: SWE-agent environment

- Task: close pull requests
- Observations:
  - Action outputs
  - Task instruction
- Actions:
  - Open <path>
  - goto <line\_number>
  - search\_file <search\_term> [<file>]
  - ...
- Environment: implements action API, formats outputs



Issue

psf\_\_requests-2317

We're currently solving the following issue within our repository. Here's the issue text:

ISSUE:

```
method = builtin_str(method) problem
In requests/sessions.py is a command:
method = builtin_str(method)
Converts method from
b'GET'
to
"b'GET'"
```

Which is the literal string, no longer a binary string. When requests tries to use the method "b'GET'", it gets a 404 Not Found response.

I am using python3.4 and python-neutronclient (2.3.9) with requests (2.4.3). neutronclient is broken because it uses this "args = utils.safe\_encode\_list(args)" command which converts all the values to binary string, including method.

I'm not sure if this is a bug with neutronclient or a bug with requests, but I'm starting here. Seems if requests handled the method value being a binary string, we wouldn't have any problem.

Also, I tried in python2.6 and this bug doesn't exist there. Some difference between 2.6 and 3.4 makes this not work right.

INSTRUCTIONS: (31 lines)

```
(Open file: n/a)
(Current directory: /psf__requests)
bash-\$
```

Observation

psf\_\_requests-2317

```
Found 2 matches for "sessions.py" in /psf__requests:
/psf__requests/build/lib/requests/sessions.py
/psf__requests/requests/sessions.py

(Open file: n/a)
(Current directory: /psf__requests)
bash-$
```



SWE-agent

psf\_\_requests-2317

To address the issue, we first need to replicate the behavior described. The issue mentions that the 'builtin\_str' function in 'requests/sessions.py' is converting a binary string method like 'b'GET' into the literal string "'b'GET'", which is incorrect. We should start by locating the 'sessions.py' file within the 'requests' library to examine the code around the 'builtin\_str' function call.

Let's begin by finding the 'sessions.py' file in the current repository.

```
find_file "sessions.py"
```





# Basic idea: support new domains/tasks by implementing more action API calls

- AutoCodeRover:

| API name                        | Description                                | Output                                 |
|---------------------------------|--------------------------------------------|----------------------------------------|
| search_class (cls)              | Search for class cls in the codebase.      | Signature of the searched class.       |
| search_class_in_file (cls, f)   | Search for class cls in file f.            | Signature of the searched class.       |
| search_method (m)               | Search for method m in the codebase.       | Implementation of the searched method. |
| search_method_in_class (m, cls) | Search for method m in class cls.          | Implementation of the searched method. |
| search_method_in_file (m, f)    | Search for method m in file f.             | Implementation of the searched method. |
| search_code (c)                 | Search for code snippet c in the codebase. | +/- 3 lines of the searched snippet c. |
| search_code_in_file (c, f)      | Search for code snippet c in file f.       | +/- 3 lines of the searched snippet c. |

- CTF:

| Category                | Command                                                              | Documentation                                                                                                                                                                                |
|-------------------------|----------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>Static analysis</i>  | <b>decompile</b> <binary_path><br>[-function_name <function_name>]   | Decompile a binary and prints the decompilation of a given function name, or main by default.                                                                                                |
|                         | <b>disassemble</b> <binary_path><br>[-function_name <function_name>] | Disassembles a binary and prints the disassembly of a given function name, or main by default.                                                                                               |
| <i>Dynamic analysis</i> | <b>debug_start</b> <binary> [<args>]                                 | Starts a debug session with the given binary and optional command-line arguments for the binary.                                                                                             |
|                         | <b>debug_add_breakpoint</b> <breakpoint>                             | Adds a breakpoint in the debug session.                                                                                                                                                      |
|                         | <b>debug_continue</b>                                                | Continues the program execution in the debug session.                                                                                                                                        |
|                         | <b>debug_step</b> [<number>]                                         | Steps number of instructions (default is 1) in the debug session.                                                                                                                            |
|                         | <b>debug_exec</b> <command><br><b>debug_stop</b>                     | Executes arbitrary gdb command in debug session.<br>Stops the current debug session.                                                                                                         |
| <i>Network</i>          | <b>connect_start</b> <server_address><br><port>                      | Starts a new interactive connection to the server address and port.                                                                                                                          |
|                         | <b>connect_sendline</b> [<line>]                                     | Sends a line with Unicode/hexadecimal values to the connection. Sending hexadecimal bytes should be done using \x<hh> where hh stands for the specific byte you want to send.                |
|                         | <b>connect_exec</b> <command><br><b>connect_stop</b>                 | Executes arbitrary connect command in connect session.<br>Stops the current connect session.                                                                                                 |
| <i>Task</i>             | <b>submit</b> ' <flag> '                                             | Submits your current flag and terminates the session upon correct response, you should be aware to properly escape the flag as this is a bash command and to put your flag in single quotes. |
|                         | <b>exit_forfeit</b>                                                  | Give up on the current challenge and terminate the session.                                                                                                                                  |



# Example: text-based web browsing

- Task: answer questions
- Observations:
  - Action outputs
  - Question
- Actions:
  - Search <query>
  - Quote <text>
  - End: Answer
  - ...
- Environment: implements action API, formats outputs (see Appendix A)

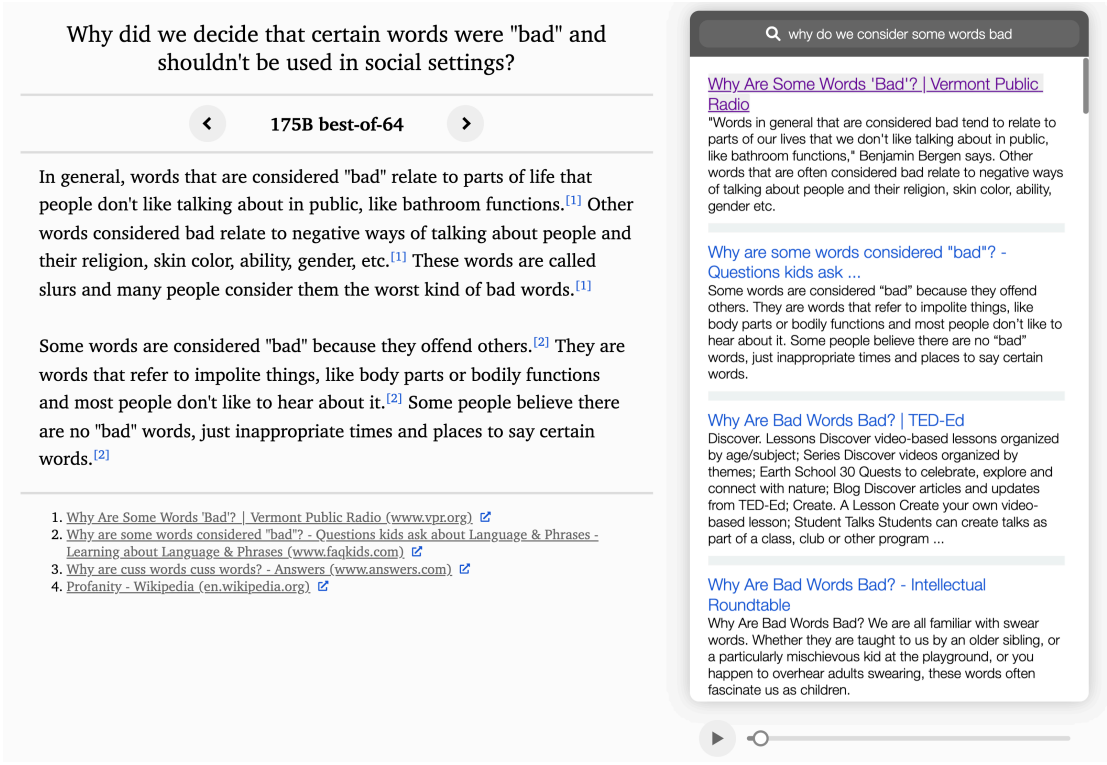
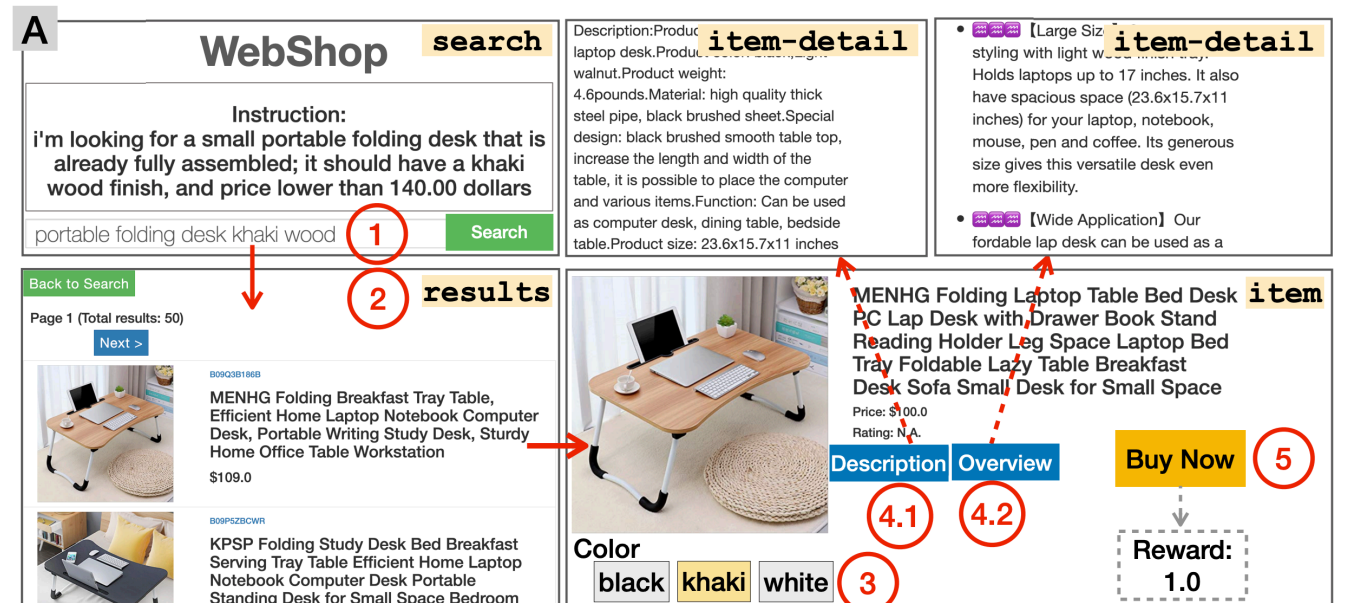


Table 1: Actions the model can take. If a model generates any other text, it is considered to be an invalid action. Invalid actions still count towards the maximum, but are otherwise ignored.

| Command                        | Effect                                                         |
|--------------------------------|----------------------------------------------------------------|
| Search <query>                 | Send <query> to the Bing API and display a search results page |
| Clicked on link <link ID>      | Follow the link with the given ID to a new page                |
| Find in page: <text>           | Find the next occurrence of <text> and scroll to it            |
| Quote: <text>                  | If <text> is found in the current page, add it as a reference  |
| Scrolled down <1, 2, 3>        | Scroll down a number of times                                  |
| Scrolled up <1, 2, 3>          | Scroll up a number of times                                    |
| Top                            | Scroll to the top of the page                                  |
| Back                           | Go to the previous page                                        |
| End: Answer                    | End browsing and move to answering phase                       |
| End: <Nonsense, Controversial> | End browsing and skip answering phase                          |

# Example: text-based web shopping

- Task: buy products
- Observations:
  - HTML of web page
  - Instruction
- Environment: simulates an e-commerce website



| Type   | Argument                 | State → Next State |
|--------|--------------------------|--------------------|
| search | [ <i>Query</i> ]         | Search → Results   |
| choose | Back to search           | * → Search         |
| choose | Prev/Next page           | Results → Results  |
| choose | [ <i>Product title</i> ] | Results → Item     |
| choose | [ <i>Option</i> ]        | Item → Item        |
| choose | Desc/Overview            | Item → Item-Detail |
| choose | Previous                 | Item-Detail → Item |
| choose | Buy                      | Item → Episode End |

Table 1: Actions in WebShop.

# Agent environments

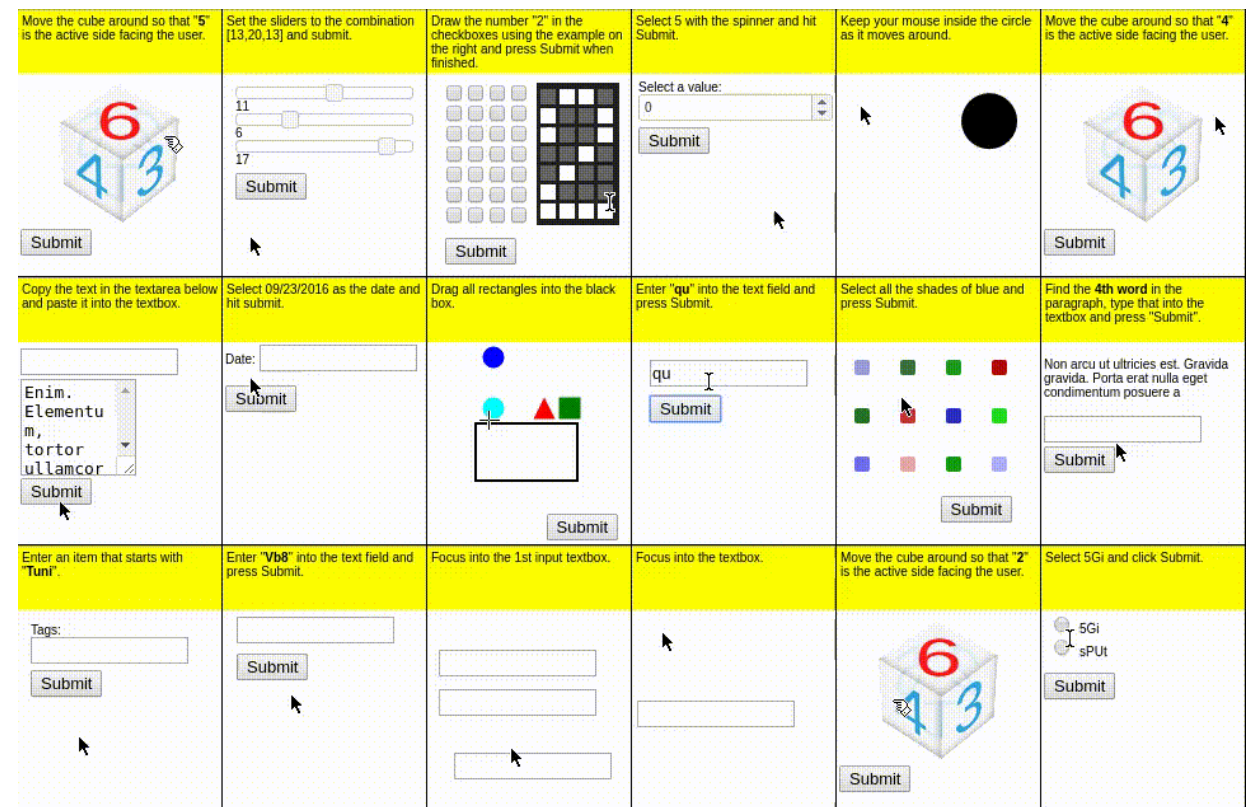
- Text agents
- **Visual agents**
- Computer-use agents

# Visual agent environments

- Text + image observations

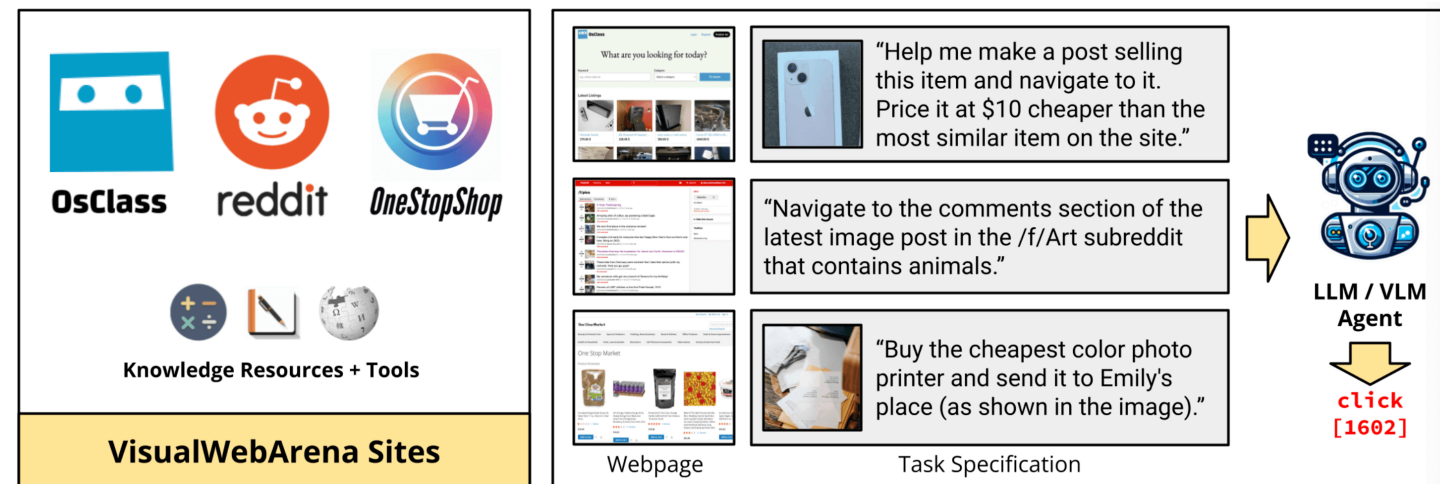
# Example: basic web browsing

- Task: described in instruction (e.g., “click button”)
- Observations:
  - Image of webpage
- Actions:
  - Move, click, scroll, type, ...
- Environment: implements action API, runs web browser and pages, keeps track of reward



# Example: web tasks

- Task: described in instruction
- Observations:
  - Screenshot of webpage
- Actions:
  - Click element, type on element, close tab, go back, ...
- Environment: simulates websites:
  - Web shopping
  - Reddit
  - Classifieds marketplace



| Action Type $a$    | Description                  |
|--------------------|------------------------------|
| click [elem]       | Click on element elem.       |
| hover [elem]       | Hover on element elem.       |
| type [elem] [text] | Type text on element elem.   |
| press [key_comb]   | Press a key combination.     |
| new_tab            | Open a new tab.              |
| tab_focus [index]  | Focus on the i-th tab.       |
| tab_close          | Close current tab.           |
| goto [url]         | Open url.                    |
| go_back            | Click the back button.       |
| go_forward         | Click the forward button.    |
| scroll [up down]   | Scroll up or down the page.  |
| stop [answer]      | End the task with an output. |

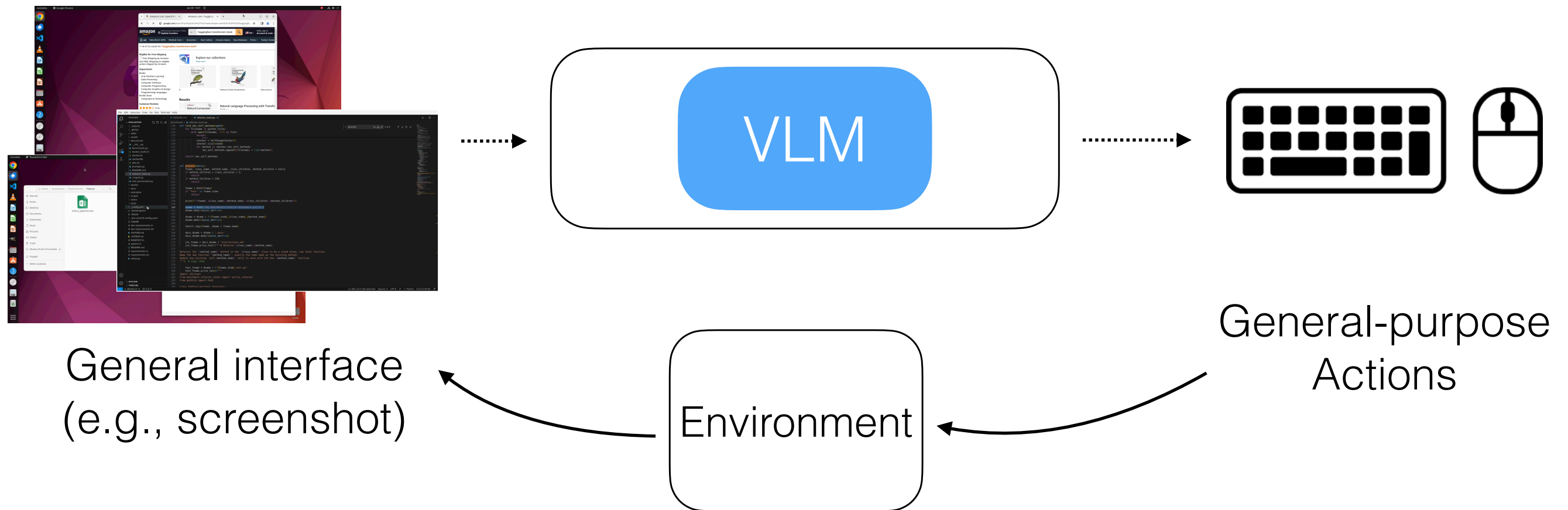
# Agent environments

- Text agents
- Visual agents
- **Computer-use agents**



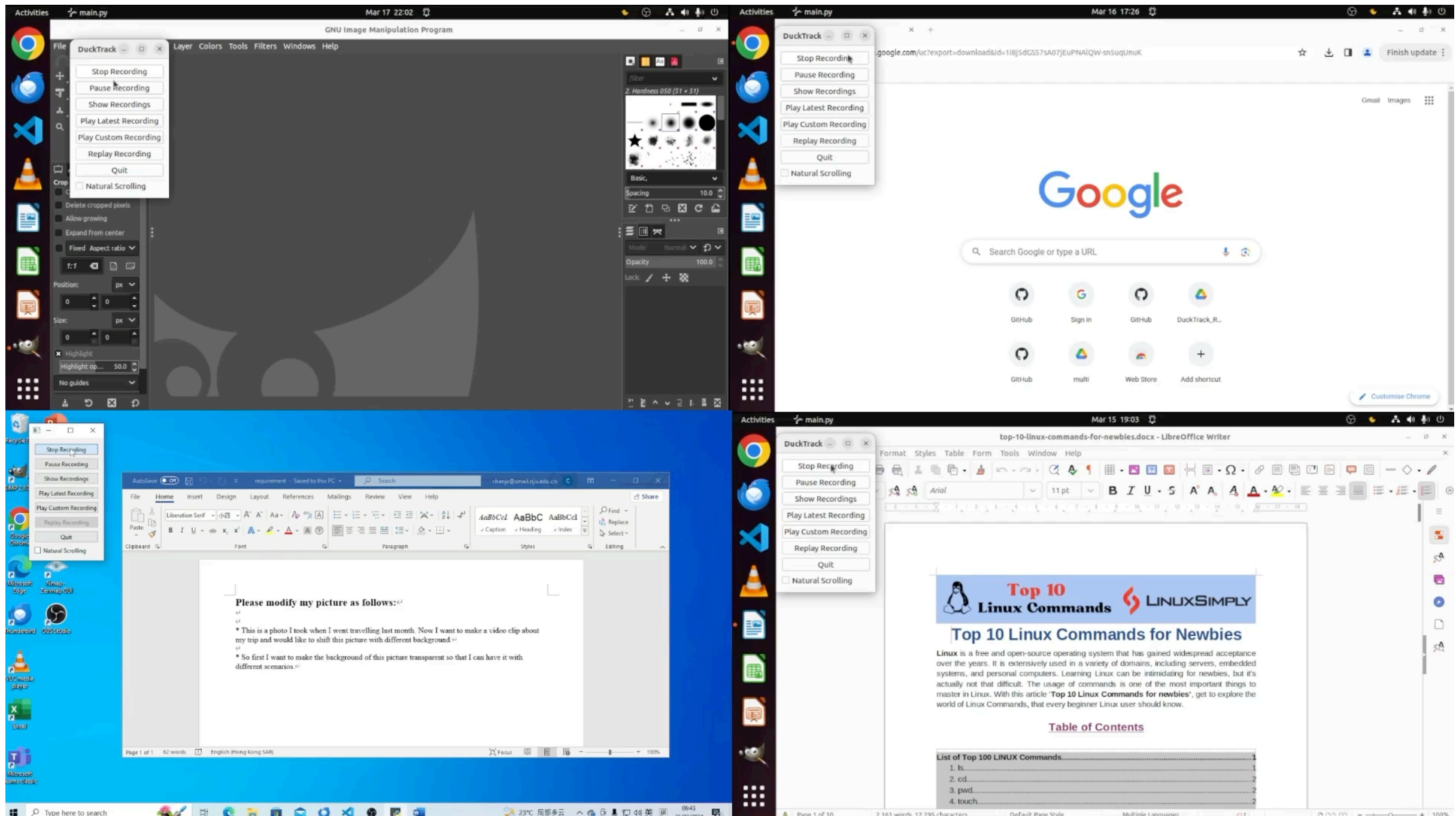
# Computer-use

- Basic idea: interact with a computer like a human
- **Expressive:** in principle, agents can perform any task a human can on a computer



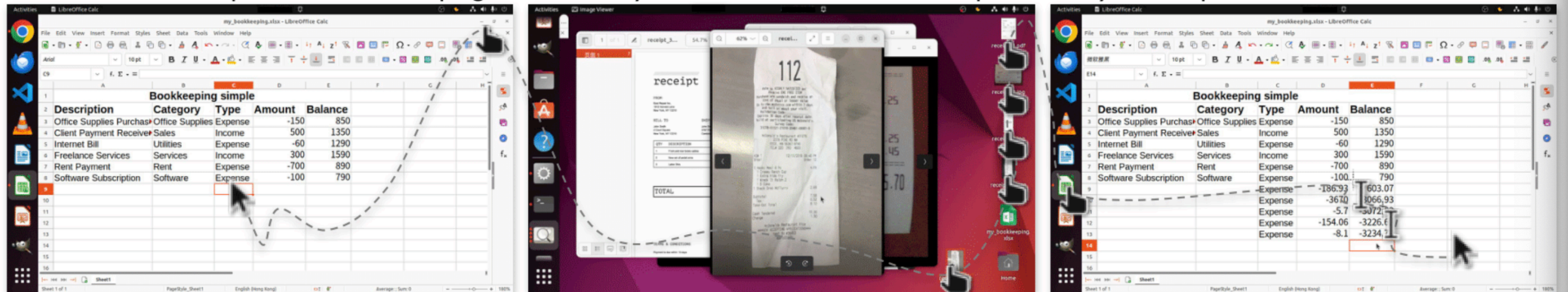


# Example: OS World

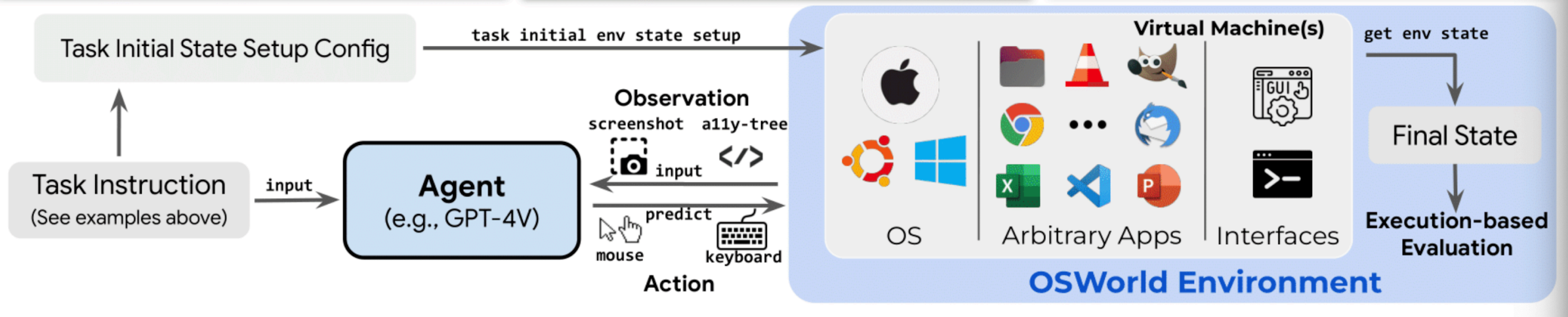
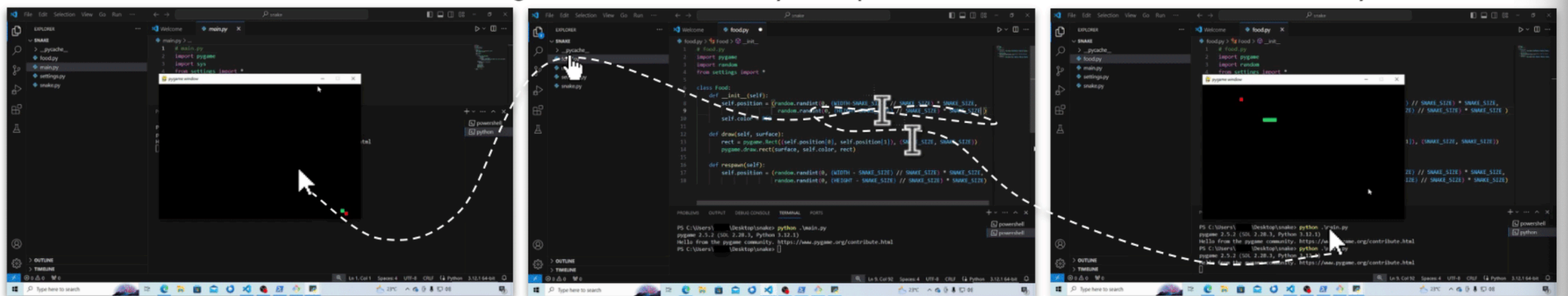


# Example: OS World

Task instruction 1: Update the bookkeeping sheet with my recent transactions over the past few days in the provided folder.



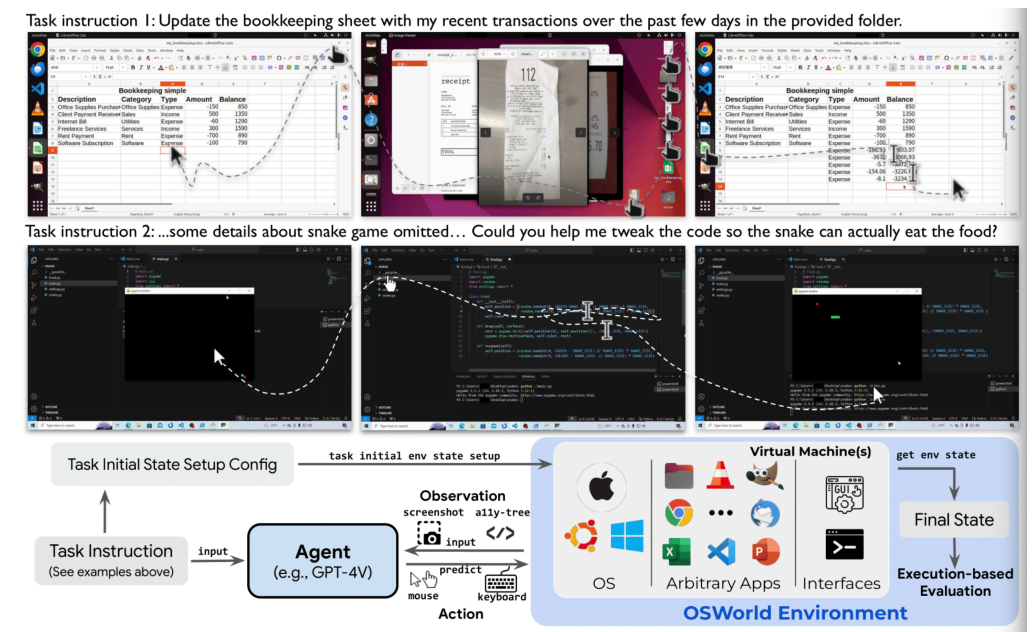
Task instruction 2: ...some details about snake game omitted... Could you help me tweak the code so the snake can actually eat the food?





# Example: OS World

- Task
  - Any task achievable on an OS
- Observations
  - Screenshot and/or accessibility tree
  - Task instruction
- Actions
  - Mouse, keyboard
- Rewards
  - Execute an evaluation script for the task
- Environment: runs an operating system



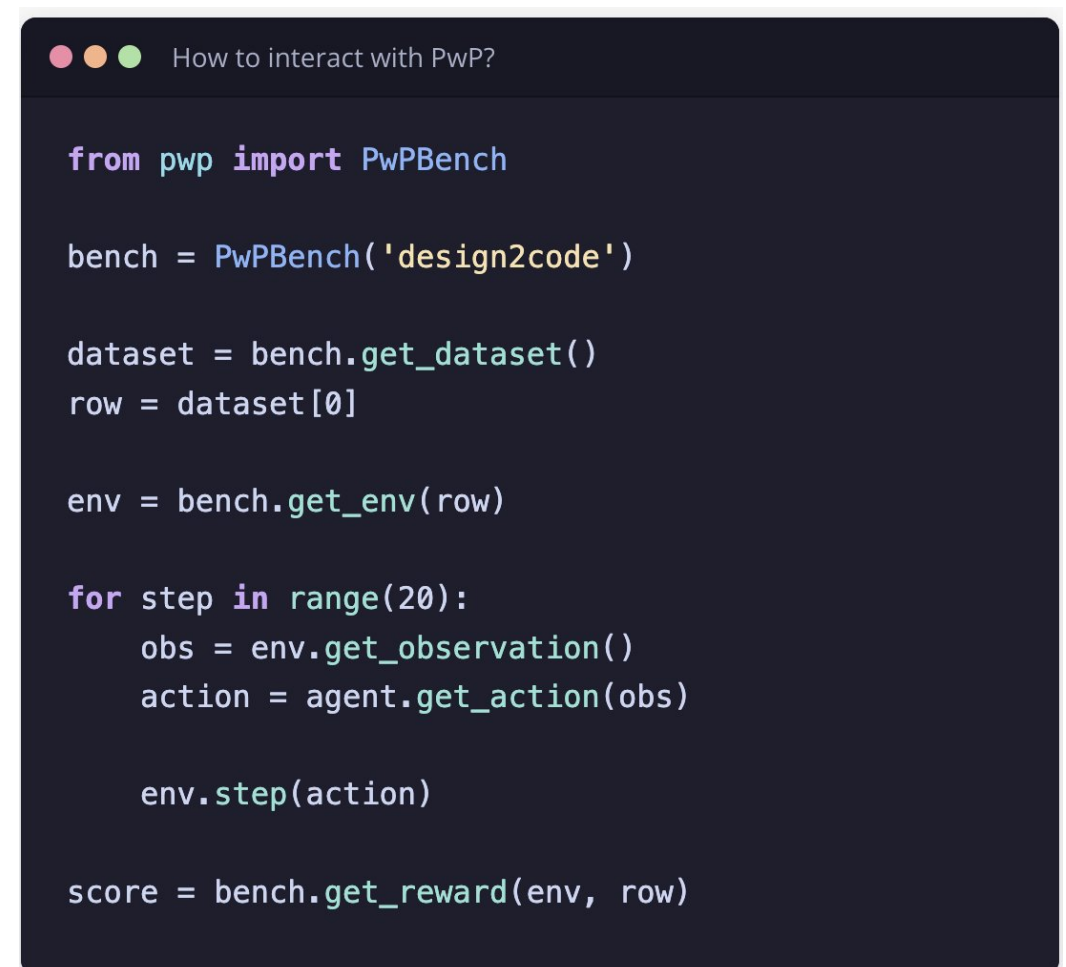
# Example: Programming with Pixels

- Tasks
  - Any task achievable in an IDE (VS Code)
- Observations
  - Screenshot
  - Task instruction
- Actions
  - Mouse, keyboard
- Rewards
  - Execute an evaluation script for the task
- Environment: runs a modified VS Code



# Example: Programming with Pixels

- Tasks
  - Any task achievable in an IDE (VS Code)
- Observations
  - Screenshot
  - Task instruction
- Actions
  - Mouse, keyboard
- Rewards
  - Execute an evaluation script for the task
- Environment: runs a modified VS Code



```
from pwp import PwPBench

bench = PwPBench('design2code')

dataset = bench.get_dataset()
row = dataset[0]

env = bench.get_env(row)

for step in range(20):
    obs = env.get_observation()
    action = agent.get_action(obs)

    env.step(action)

score = bench.get_reward(env, row)
```

# Today: (LM) Agents

- What is an agent?
- Agent environments
- **Agent design patterns**
- Agents and RL

# Agent Patterns

- Chain-of-thought
- Scaffolding
- Tool calling and MCP

# Chain of thought

- Prompt the model to output a thought prior to the action
- We can view this as:
  - Policy  $\pi_{\theta}(a_t | s_t)$  internally generates a thought  $z_t$  to determine its distribution over actions
  - The action decomposes into  $(z_t, a_t)$

## 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
```



# Scaffolding

- Scaffolding: code that structures how agents interact with the environment
  - Action/observation loop
  - Action definitions and formatting (“tools”)
  - State management between turns (e.g., formatting observations, maintaining a history)
  - System prompt: defines agent’s role and capabilities

# Example: OpenHands

- Scaffolding for code agents

```
You are OpenHands agent, a helpful AI assistant that can interact with a computer to solve tasks.
<IMPORTANT>
* If user provides a path, you should NOT assume it's relative to the current working directory.
  Instead, you should explore the file system to find the file before working on it.
* When configuring git credentials, use "openhands" as the user.name and "openhands@all-hands.dev"
  as the user.email by default, unless explicitly instructed otherwise.
* The assistant MUST NOT include comments in the code unless they are necessary to describe non-
  obvious behavior.
</IMPORTANT>
```

```
_BASH_DESCRIPTION = """Execute a bash command in the terminal.
* Long running commands: For commands that may run indefinitely, it should be run in the
* Interact with running process: If a bash command returns exit code '-1', this means
* One command at a time: You can only execute one bash command at a time. If you need
"""
```

```
_BROWSER_TOOL_DESCRIPTION = """
The following 15 functions are available. Nothing else is supported.

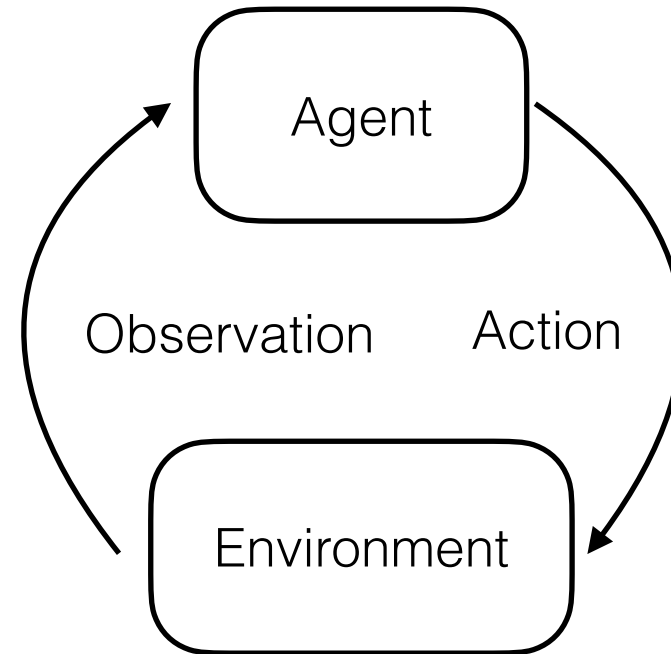
goto(url: str)
  Description: Navigate to a url.
  Examples:
    goto('http://www.example.com')

go_back()
  Description: Navigate to the previous page in history.
  Examples:
    go_back()

go_forward()
  Description: Navigate to the next page in history.
  Examples:
    go_forward()
```

# Two perspectives on taking actions

- **Reinforcement learning:** submit an action to the environment, receive a observation

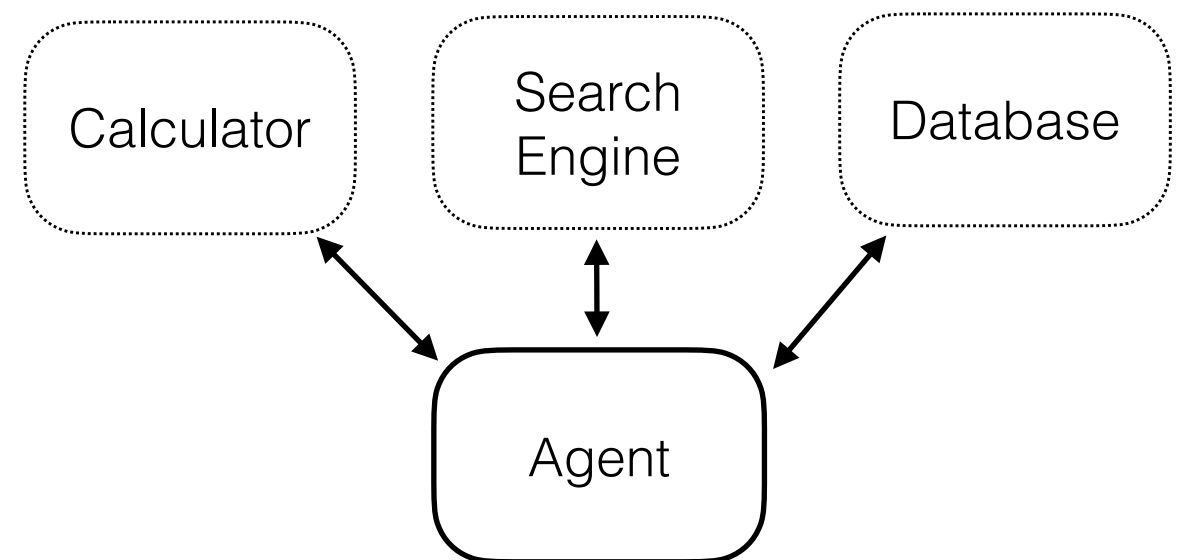


```
env = bench.get_env(row)

for step in range(20):
    obs = env.get_observation()
    action = agent.get_action(obs)

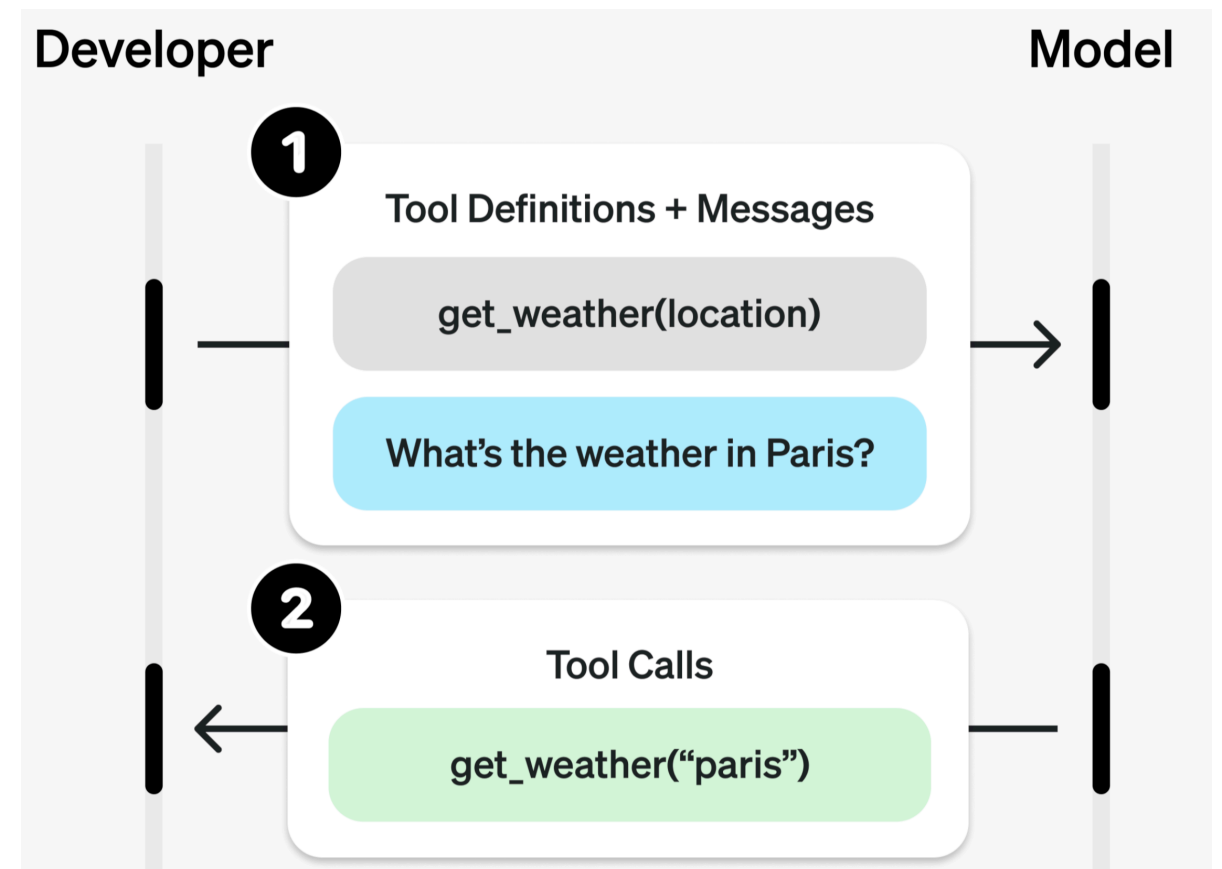
    env.step(action)
```

- **Tool calling:** model calls an external function, the environment returns the output



# Tool calling interface

- Tool calling interface: a standardized way to define and invoke tools
- Developer 1: defines tool as JSON
- Model: generates tool calls
- Developer 2: executes tool call, sends output



# Tool calling interface

```
{
  "type": "function",
  "function": {
    "name": "get_weather",
    "description": "Get the current weather in a location",
    "parameters": {
      "type": "object",
      "properties": {
        "location": {
          "type": "string",
          "description": "City name, e.g. San Francisco"
        },
        "unit": {
          "type": "string",
          "enum": ["celsius", "fahrenheit"]
        }
      },
      "required": ["location"]
    }
  }
}
```

Developer

Model

1

Tool Definitions + Messages

get\_weather(location)

What's the weather in Paris?

2

Tool Calls

get\_weather("paris")

```
{
  "role": "tool",
  "tool_call_id": "call_abc123",
  "content": "Weather in Paris: 22°C, partly cloudy"
}
```

```
{
  "role": "assistant",
  "content": null,
  "tool_calls": [{
    "id": "call_abc123",
    "type": "function",
    "function": {
      "name": "get_weather",
      "arguments": "{\"location\": \"Paris\"}"
    }
  ]
}
```

# Example: Claude computer-use

```
Action = Literal[
    "key",
    "type",
    "mouse_move",
    "left_click",
    "left_click_drag",
    "right_click",
    "middle_click",
    "double_click",
    "screenshot",
    "cursor_position",
]
```

```
Command = Literal[
    "view",
    "create",
    "str_replace",
    "insert",
    "undo_edit",
]
```

```
class BashTool(BaseAnthropicTool):
    """
    A tool that allows the agent to run bash commands.
    The tool parameters are defined by Anthropic and are not editable.
    """

    _session: _BashSession | None
    name: ClassVar[Literal["bash"]] = "bash"
    api_type: ClassVar[Literal["bash_20241022"]] = "bash_20241022"
```

# Example: Claude computer-use

Can you please fix the error in the terminal by removing the problematic line from the open file?

From the screenshot, I can see a 404 error on the line "GET /api/placeholder/200/200 HTTP/1.1". Let me help you with that.

First, I'll press **Cmd+F** to open the 'Find' widget and navigate to the correct line. Then, I'll type in API to find the line that is throwing an error within the file.

Key ⌘ F

Type api

The screenshot shows the Visual Studio Code (VS Code) interface on a macOS system. The Explorer sidebar on the left shows the file structure of the 'Downloads' folder, with '90s-website.html' selected. The main editor window displays the content of '90s-website.html'. The code is an HTML file with a container, a heading, and a list of links. The terminal at the bottom shows the output of running a Python HTTP server. The error message in the terminal is: 'code 404, message File not found'. The error is associated with the GET request for the file 'api/placeholder/200/200'. The code in the editor shows the following structure:

```
<html>
<body>
  <div class="container">
    <div class="marquee">
    </div>
    <h1 style="text-align: center; color: #ff00ff; text-shadow: 2px 2px #110888">
      Welcome to Claude's World! &#11088
    </h1>
    
      <h2>Cool Links!</h2>
      <ul>
        <li><a href="#">Home</a></li>
        <li><a href="#">About Me</a></li>
        <li><a href="#">My Photos</a></li>
        <li><a href="#">My Thoughts</a></li>
        <li><a href="#">Contact Me</a></li>
      </ul>
    </div>
  </div>
</body>
</html>
```

The terminal output shows the following commands and results:

```
igor@Igor's-Virtual-Machine Downloads % python -m http.server 8000
zsh: command not found: python
igor@Igor's-Virtual-Machine Downloads % python3 -m http.server 8000
Serving HTTP on :: port 8000 (http://[::]:8000/) ...
::1 - - [16/Oct/2024 22:54:39] "GET /90s-website.html HTTP/1.1" 200 -
::1 - - [16/Oct/2024 22:54:39] code 404, message File not found
::1 - - [16/Oct/2024 22:54:39] "GET /api/placeholder/200/200 HTTP/1.1" 404 -
```

<https://www.youtube.com/watch?v=vH2f7cjXjKI>



# Example: Claude computer-use

First, I'll press `Cmd+F` to open the 'Find' widget and navigate to the correct line. Then, I'll type in API to find the line that is throwing an error within the file.

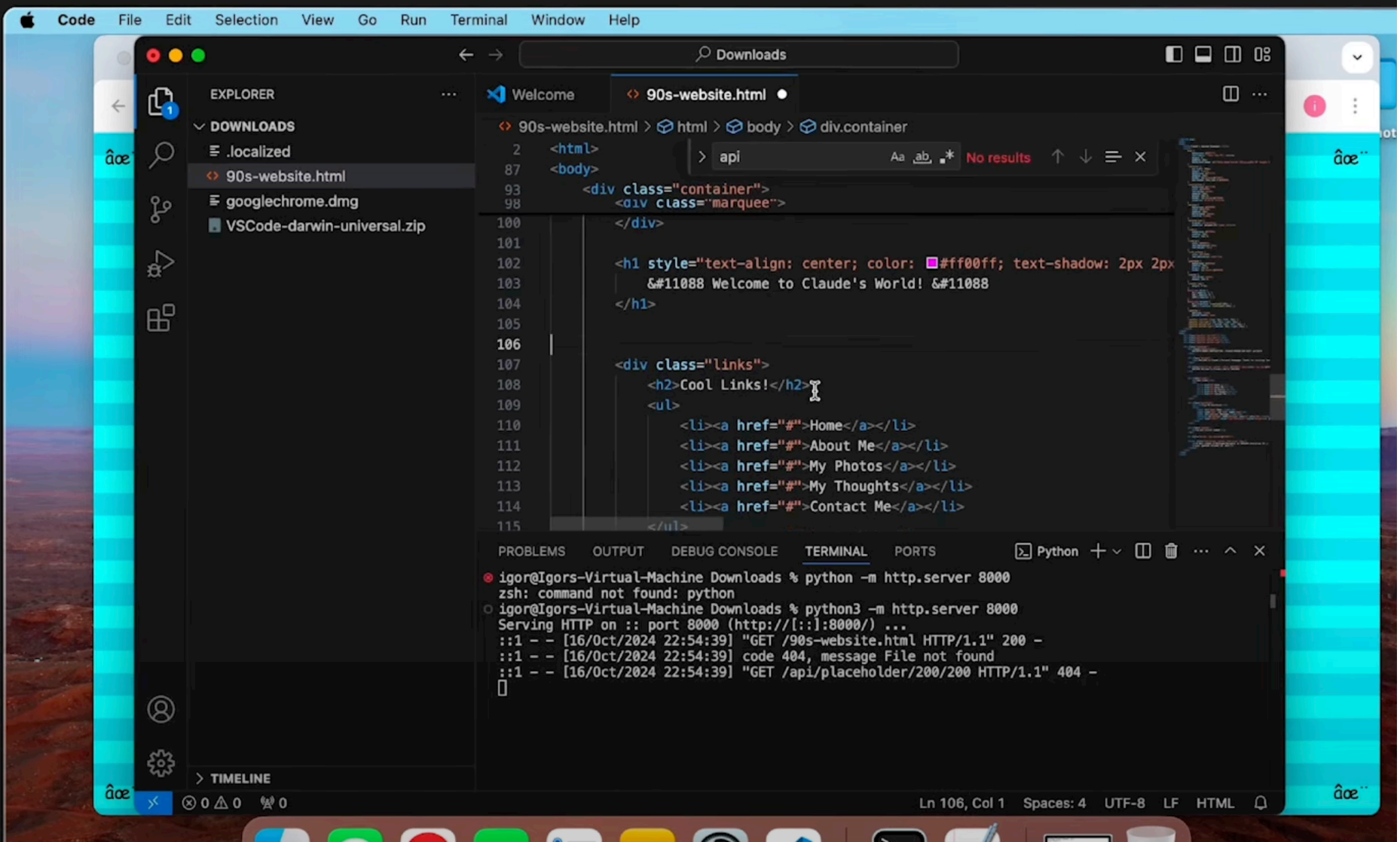
Key `⌘` `F`

Type `api`

Now I can see line 106, which contains an `<img>` tag and the text with the API string from before. To delete this line, I'll press the keyboard shortcut `Shift+Delete` or `Cmd+Shift+K` which is VSCode's command for deleting a line:

Key `⌘` `⇧` `K`

I can see that line 106 has been successfully deleted. Let me save the file using `Cmd+S` to ensure the changes are preserved:



The screenshot shows the VS Code editor interface. The Explorer panel on the left shows the file '90s-website.html' selected in the 'Downloads' folder. The main editor area shows the HTML file with line 106 highlighted. The file content includes a welcome message and a list of links. The terminal at the bottom shows the command 'python -m http.server 8000' and the output of the server, which includes a 404 error for the API endpoint.

```
igor@Igor's-Virtual-Machine Downloads % python -m http.server 8000
zsh: command not found: python
igor@Igor's-Virtual-Machine Downloads % python3 -m http.server 8000
Serving HTTP on :: port 8000 (http://[::]:8000/) ...
::1 - - [16/Oct/2024 22:54:39] "GET /90s-website.html HTTP/1.1" 200 -
::1 - - [16/Oct/2024 22:54:39] code 404, message File not found
::1 - - [16/Oct/2024 22:54:39] "GET /api/placeholder/200/200 HTTP/1.1" 404 -
```

<https://www.youtube.com/watch?v=vH2f7cjXjKI>



# Today: (LM) Agents

- What is an agent?
- Agent environments
- Agent design patterns
- **Agents and RL**

# Agents and RL

- Key differences
  - Environment provides new information
    - $a_t$ : “search [...]”  $\rightarrow s_{t+1}$ : search results
    - $a_t$ : click x y  $\rightarrow s_{t+1}$ : screenshot after click
    - Multi-step MDP instead of 1-step MDP
  - Environment interaction is often expensive
  - Tasks are often long

# Example: RL for computer-use

- $a_t \sim \pi_\theta(a_t | x, s_t)$ 
  - $x$ : task instruction
  - $s$ : screenshot, history of observations and actions
  - $a$ : chain of thought + action
  - $\Rightarrow$  trajectory  $\tau = (x, (s_1, a_1), (s_2, a_2), \dots, (s_T, a_T))$

- GRPO at a step level

$$\text{ratio}_{t,k} = \frac{\pi_\theta(a_{t,k} | x, s_t, a_{t,<k})}{\pi_{\theta_{old}}(a_{t,k} | x, s_t, a_{t,<k})}, \text{ single grouped-advantage}$$

for all steps

- Innovation: only apply loss on high-entropy states
- Reward: environment provides it by checking the environment state or using a VLM
- CPU Bottleneck: cluster of 180 environment instances

## EFFICIENT MULTI-TURN RL FOR GUI AGENTS VIA DE-COUPLED TRAINING AND ADAPTIVE DATA CURATION

Pengxiang Li<sup>1,2\*</sup>, Zechen Hu<sup>3\*</sup>, Zirui Shang<sup>1,2\*</sup>, Jingrong Wu<sup>3\*</sup>, Yang Liu<sup>2</sup>, Hui Liu<sup>3</sup>, Zhi Gao<sup>1,2</sup>, Chenrui Shi<sup>1,2</sup>, Bofei Zhang<sup>2</sup>, Zihao Zhang<sup>3</sup>, Xiaochuan Shi<sup>3</sup>, Zedong Yu<sup>2,4</sup>, Yuwei Wu<sup>1,5</sup>, Xinxiao Wu<sup>1,5</sup>, Yunde Jia<sup>5</sup>, Liuyu Xiang<sup>4</sup>, Zhaofeng He<sup>4</sup>, Qing Li<sup>2</sup>

<sup>1</sup>Beijing Institute of Technology <sup>2</sup>State Key Laboratory of General Artificial Intelligence, BIGAI  
<sup>3</sup>DataCanvas <sup>4</sup>Beijing University of Posts and Telecommunications <sup>5</sup>Shenzhen MSU-BIT University  
<https://computer-use-agents.github.io/dart-gui>

### System Prompt for GUI Agent

```
You are a GUI agent. You are given a task and your action history, with screenshots. You need to perform the next action to complete the task.

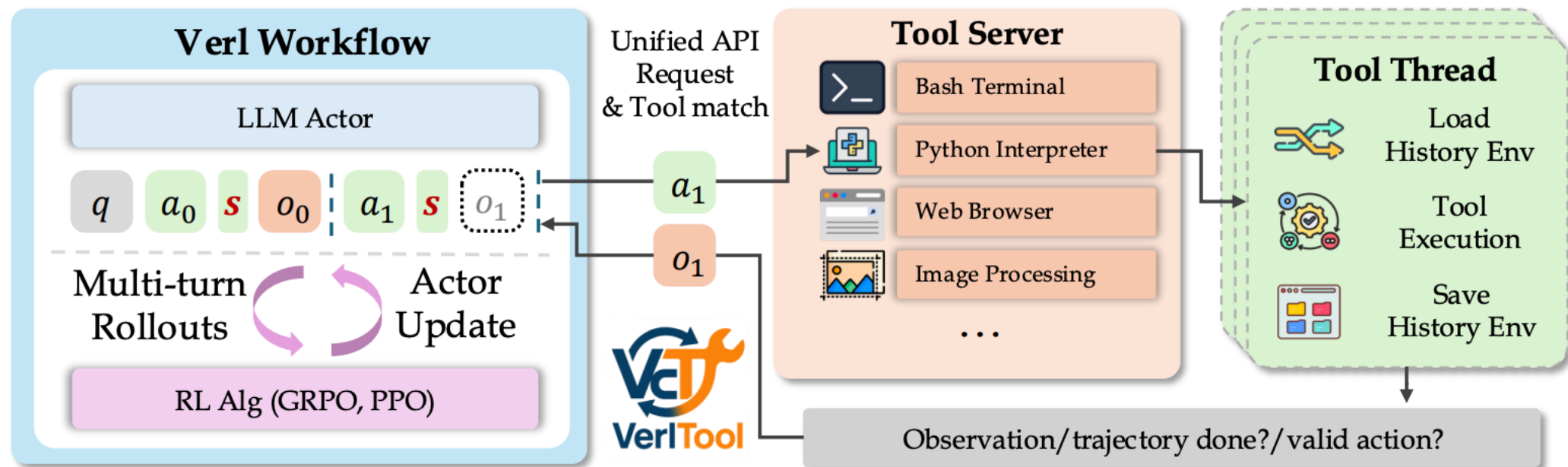
## Output Format
Thought: ...
Action: ...

## Action Space
click(start_box='<|box_start|>(x1,y1)<|box_end|>')
left_double(start_box='<|box_start|>(x1,y1)<|box_end|>')
right_single(start_box='<|box_start|>(x1,y1)<|box_end|>')
drag(start_box='<|box_start|>(x1,y1)<|box_end|>',
     end_box='<|box_start|>(x3,y3)<|box_end|>')
hotkey(key='')
type(content='') # If you want to submit, use "\n" at the end
scroll(start_box='<|box_start|>(x1,y1)<|box_end|>',
       direction='down or up or right or left')
wait() # Sleep for 5s and take a screenshot
finished(content='xxx') # Use escape characters \', \", \n

## Note
• Use {language} in Thought part.
• Write a small plan and finally summarize your next action (with its target element) in one sentence in Thought part.
• My computer's password is 'password', feel free to use it when you need sudo rights.

## User Instruction
{instruction}
```

# Example: RL for tool-use



## VERLTOOL: TOWARDS HOLISTIC AGENTIC REINFORCEMENT LEARNING WITH TOOL USE

♣ Dongfu Jiang<sup>\*†</sup>, ◇ Yi Lu<sup>\*</sup>, ⚡ Zhuofeng Li<sup>\*</sup>, ♣ Zhiheng Lyu<sup>\*</sup>, <sup>ℓ</sup>Ping Nie, <sup>ϑ,θ</sup>Haozhe Wang,  
♣ Alex Su, <sup>ε</sup>Hui Chen, <sup>φ</sup>Kai Zou, ♠ Chao Du, ♠ Tianyu Pang, ♣ Wenhua Chen<sup>†</sup>  
♣ University of Waterloo, ♠ Sea AI Lab, ◇ University of Toronto, ⚡ Shanghai University,  
<sup>ϑ</sup>HKUST, <sup>θ</sup>M-A-P, <sup>ε</sup>National University of Singapore, <sup>φ</sup>NetMind.AI, <sup>ℓ</sup>Independent  
{dongfu.jiang@uwaterloo.ca, wenhuchen@uwaterloo.ca}

# Recap: Agent Basics

- What is an agent?
- Agent environments
- Agent design patterns
- Agents and RL

Thank you