



blackhat[®]
EUROPE 2024
DECEMBER 11-12, 2024
BRIEFINGS

CodeCloak: A DRL-Based Method for Mitigating Code Leakage by LLM Code Assistants

Speaker: Amit Finkman

CODECLOAK: A METHOD FOR MITIGATING CODE LEAKAGE BY LLM CODE ASSISTANTS

Amit Finkman Noah^{*}, Avishag Shapira^{*}, Eden Bar Kochva^{*}, Inbar Maimon, Dudu Mimran, Yuval Elovici, Asaf Shabtai

Department of Software and Information Systems Engineering
Ben-Gurion University of The Negev

ABSTRACT

LLM-based code assistants are becoming increasingly popular among developers. These tools help developers improve their coding efficiency and reduce errors by providing real-time suggestions based on the developer's codebase. While beneficial, the use of these tools can inadvertently expose the developer's proprietary code to the code assistant service provider during the development process. In this work, we propose a method to mitigate the risk of code leakage when using LLM-based code assistants. CodeCloak is a novel deep reinforcement learning agent that manipulates the prompts before sending them to the code assistant service. CodeCloak aims to achieve the following two contradictory goals: (i) minimizing code leakage, while (ii) preserving relevant and useful suggestions for the developer. Our evaluation, employing StarCoder and Code Llama, LLM-based code assistants models, demonstrates CodeCloak's effectiveness on a diverse set of code repositories of varying sizes, as well as its transferability across different models. We also designed a method for reconstructing the developer's original codebase from code segments sent to the code assistant service (i.e., prompts) during the development process, to thoroughly analyze code leakage risks and evaluate the effectiveness of CodeCloak under practical development scenarios.



<https://arxiv.org/pdf/2404.09066>

About Myself and the Team



Amit Finkman



Eden Bar-Kochva



Avishag Shapira



Dudu Mimran



Inbar Maimon



Prof. Asaf Shabtai



Prof. Yuval Elovici

1. **Intro**
2. **Background**
3. **Threat Model**
4. **Countermeasure**
5. **Takeaways**
6. **Future Steps**
7. **Q&A**

1. **Intro**
2. **Background**
3. **Threat Model**
4. **Countermeasure**
5. **Takeaways**
6. **Future Steps**
7. **Q&A**

Today's AI code Assistants

Today's AI code Assistants



```
1 def bubble_sort(arr):
2     for i in range(len(arr)):
3         for j in range(len(arr) - i - 1):
4             if arr[j] > arr[j+1]:
5                 arr[j], arr[j+1] = arr[j+1], arr[j]
6     return arr
7
8 def merge_sort(arr):
9     if len(arr) < 2:
10        return arr
11    mid = len(arr) // 2
12    left_arr = merge_sort(arr[:mid])
13    right_arr = merge_sort(arr[mid:])
14    return merge(left_arr, right_arr)
15
16 def quick_sort(arr):
17     if len(arr) < 2:
18         return arr
19     pivot = arr[0]
20     left_arr = [x for x in arr[1:] if x < pivot]
21     right_arr = [x for x in arr[1:] if x >= pivot]
22     return quick_sort(left_arr) + [pivot] + quick_sort(right_arr)
23
24 def merge(left_arr, right_arr):
25     result = []
26     while len(left_arr) > 0 and len(right_arr) > 0:
27         if left_arr[0] < right_arr[0]:
28             result.append(left_arr.pop(0))
29         else:
30             result.append(right_arr.pop(0))
31     result += left_arr
32     result += right_arr
33     return result
```



```

1 def bubble_sort(arr):
2     for i in range(len(arr)):
3         for j in range(len(arr) - i - 1):
4             if arr[j] > arr[j+1]:
5                 arr[j], arr[j+1] = arr[j+1], arr[j]
6     return arr
7
8 def merge_sort(arr):
9     if len(arr) < 2:
10        return arr
11    mid = len(arr) // 2
12    left_arr = merge_sort(arr[:mid])
13    right_arr = merge_sort(arr[mid:])
14    return merge(left_arr, right_arr)
15
16 def quick_sort(arr):
17     if len(arr) < 2:
18         return arr
19     pivot = arr[0]
20     left_arr = [x for x in arr[1:] if x < pivot]
21     right_arr = [x for x in arr[1:] if x >= pivot]
22     return quick_sort(left_arr) + [pivot] + quick_sort(right_arr)
23
24 def merge(left_arr, right_arr):
25     result = []
26     while len(left_arr) > 0 and len(right_arr) > 0:
27         if left_arr[0] < right_arr[0]:

```

```
1 def bubble_sort(arr):
2     for i in range(len(arr)):
3         for j in range(len(arr) - i - 1):
4             if arr[j] > arr[j+1]:
5                 arr[j], arr[j+1] = arr[j+1], arr[j]
6     return arr
```

Already typed

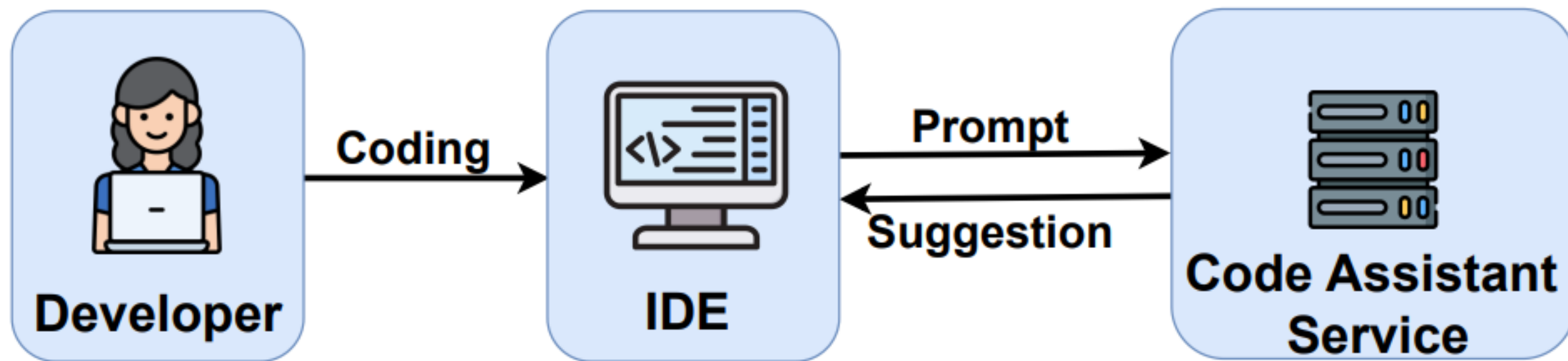
```
7
8 def merge_sort(arr):
9     if len(arr) < 2:
10         return arr
11     mid = len(arr) // 2
12     left_arr = merge_sort(arr[:mid])
13     right_arr = merge_sort(arr[mid:])
14     return merge(left_arr, right_arr)
```

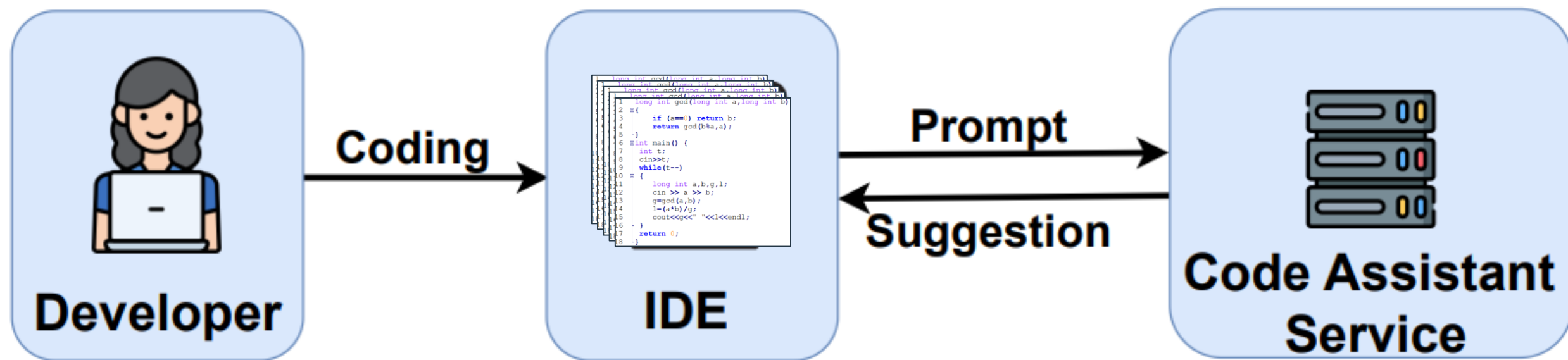
```
15
16 def quick_sort(arr):
17     if len(arr) < 2:
18         return arr
19     pivot = arr[0]
20     left_arr = [x for x in arr[1:] if x < pivot]
21     right_arr = [x for x in arr[1:] if x >= pivot]
22     return quick_sort(left_arr) + [pivot] + quick_sort(right_arr)
```

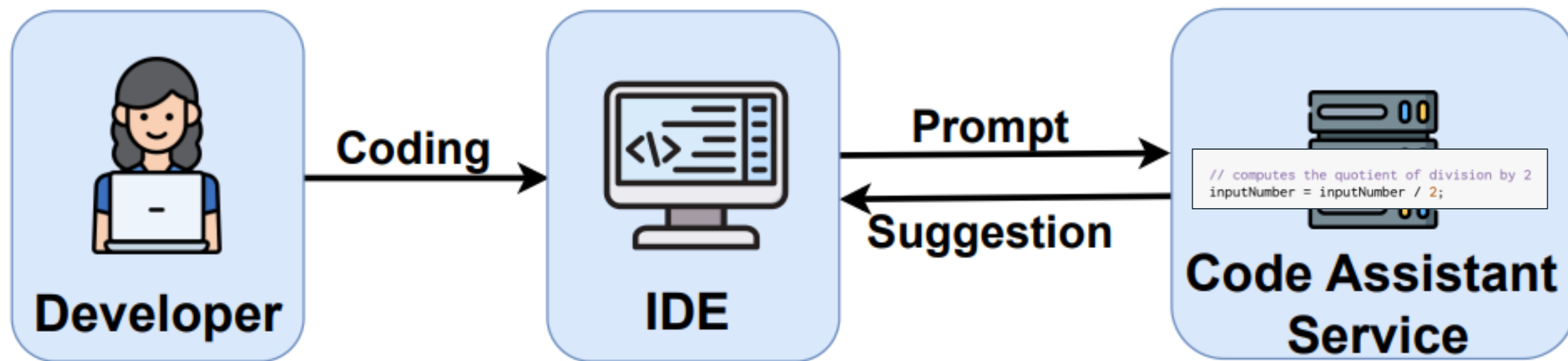
Developer is coding

The suggestion of how
to complete the code
snippet

```
23 def merge(left_arr, right_arr):
24     result = []
25     while len(left_arr) > 0 and len(right_arr) > 0:
26         if left_arr[0] < right_arr[0]:
```







1. **Intro**
2. **Background**
3. **Threat Model**
4. **Countermeasure**
5. **Takeaways**
6. **Future Steps**
7. **Q&A**

1. **Intro**
2. **Background**
3. **Threat Model**
4. **Countermeasure**
5. **Takeaways**
6. **Future Steps**
7. **Q&A**

The Problem

Leakage To the Service Providers



Information Classification: General

#BHEU @BlackHatEvents

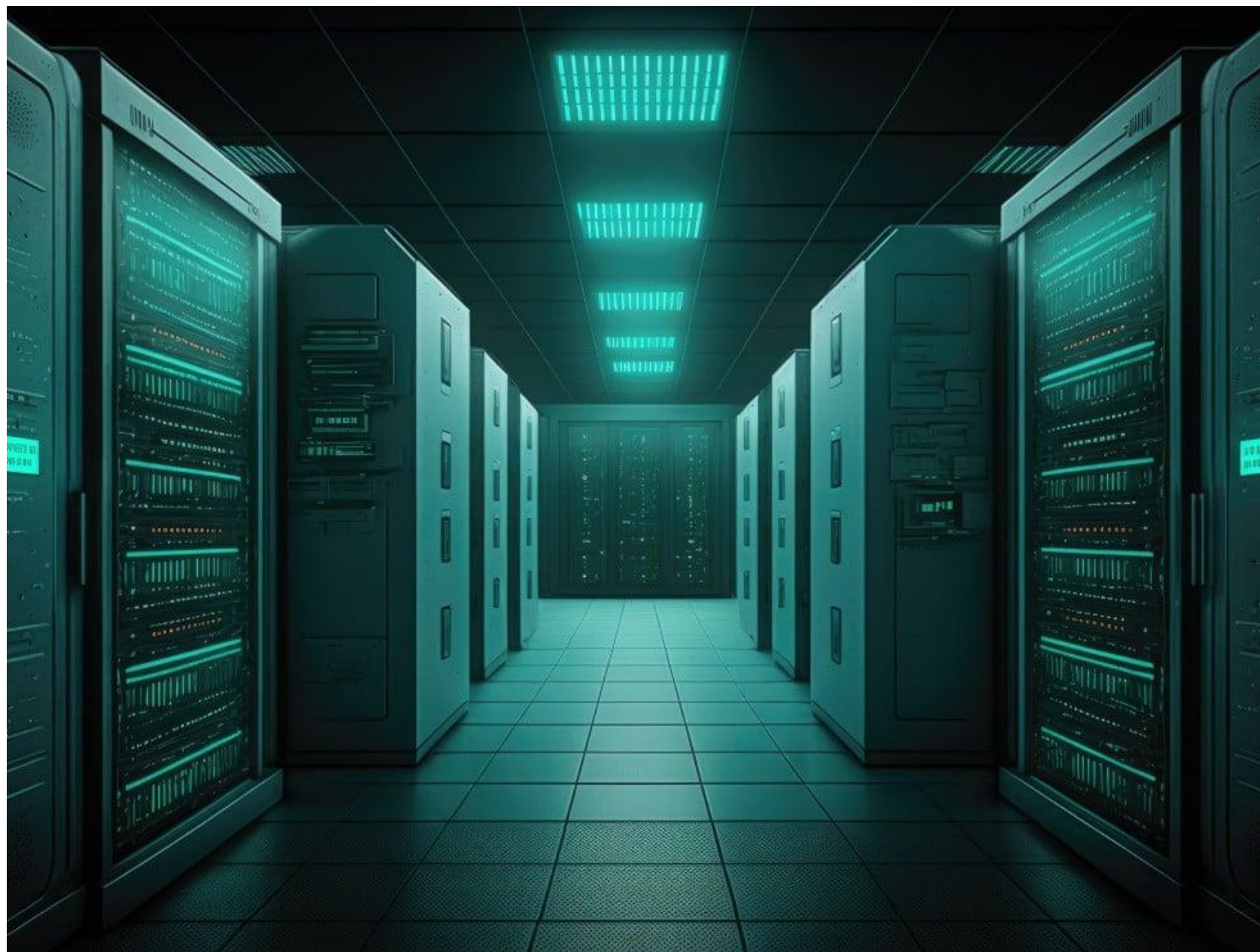
Attackers



Information Classification: General

#BHEU @BlackHatEvents

Leakage To the Service Providers





ARTIFICIAL INTELLIGENCE

Leakage To the Service Provider

FORTUNE

Subscribe Now

SIGN IN

Home

News

Tech

Finance

Leadership

Well

Recommends

Fortune 500

TECH · APPLE

Apple clamps down on employees using ChatGPT as more companies fear sensitive data sharing with A.I. models

BY NICHOLAS GORDON

May 19, 2023 at 11:56 AM GMT+3



Illustration by Alex Castro

reported Thursday, joining a growing list of companies concerned about sensitive internal information being leaked through AI.

May 2, 2023, 07:17am EDT



ARTIFICIAL INTELLIGENCE

Leakage To the Service Provider

FORTUNE

Subscribe Now

SIGN IN

Home

News

Tech

Finance

Leadership

Well

Recommends

Fortune 500

Potential Intellectual Property Violation!

more companies fear sensitive data sharing with A.I. models

BY NICHOLAS GORDON

May 19, 2023 at 11:56 AM GMT+3



Illustration by Alex Castro

reported Thursday, joining a growing list of companies concerned about sensitive internal information being leaked through AI.

May 2, 2023, 07:17am EDT

The Problem

Leakage To the Service Providers



Information Classification: General

#BHEU @BlackHatEvents

Attackers



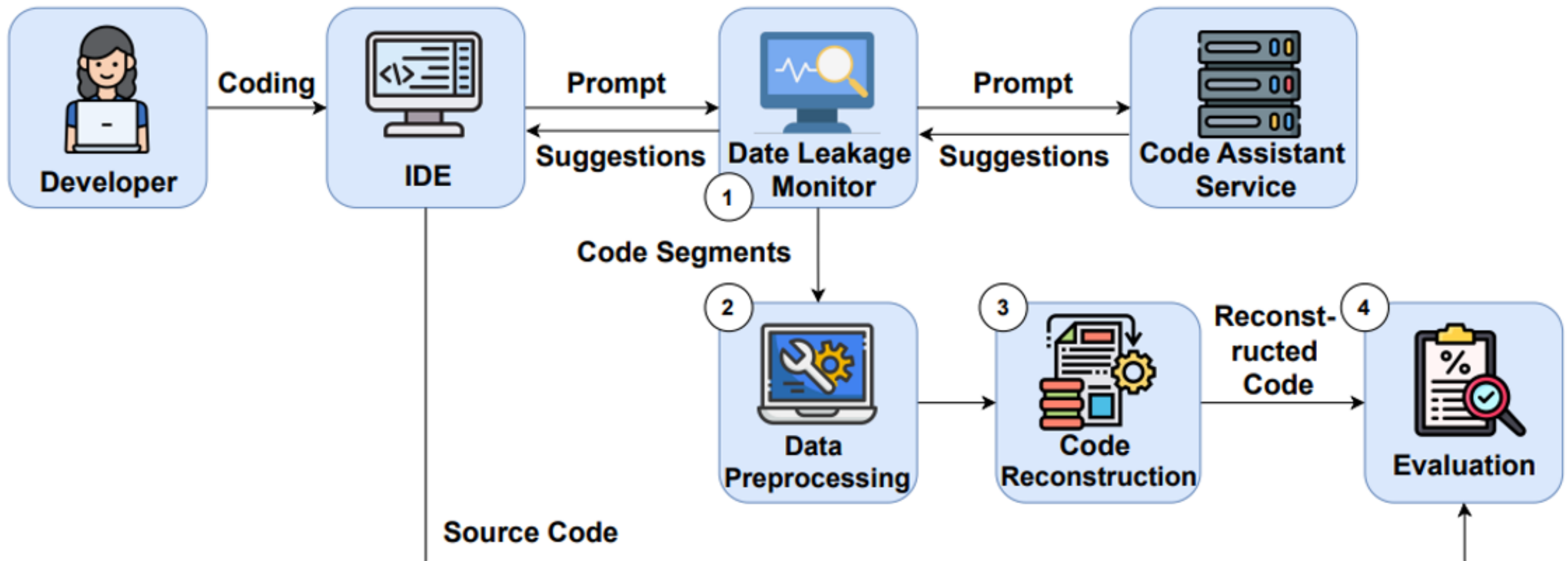
Information Classification: General

#BHEU @BlackHatEvents

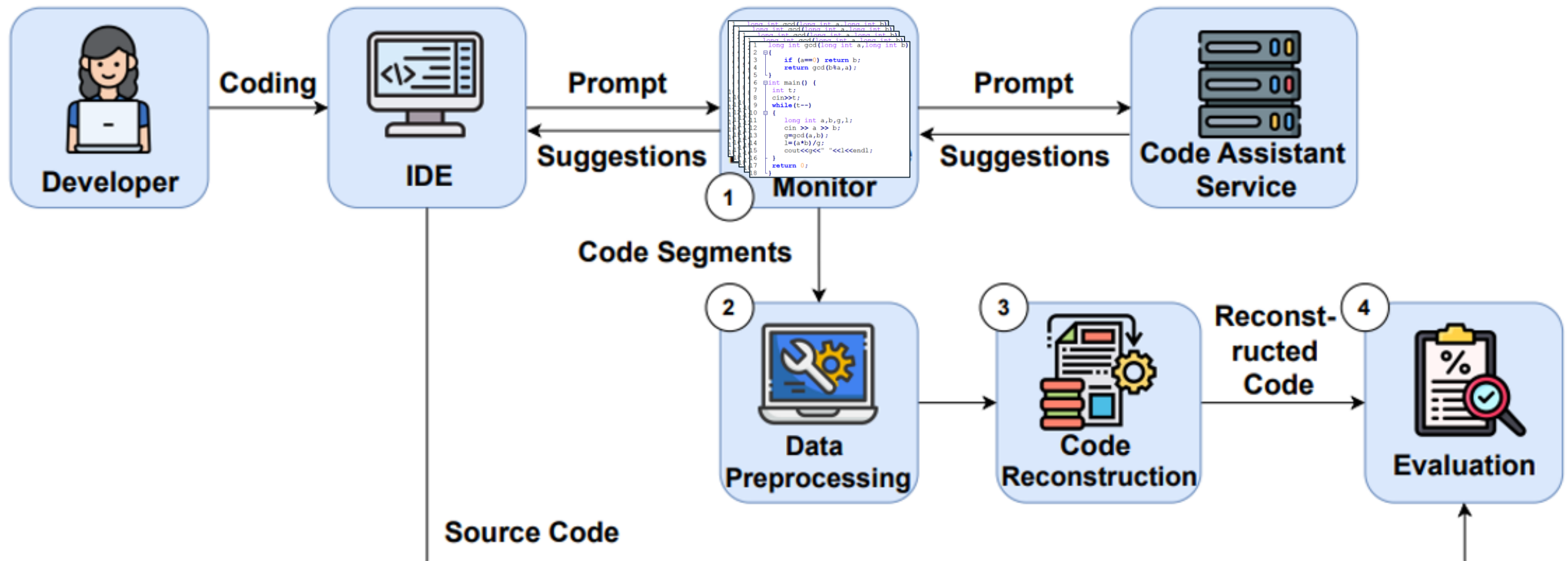
Attackers



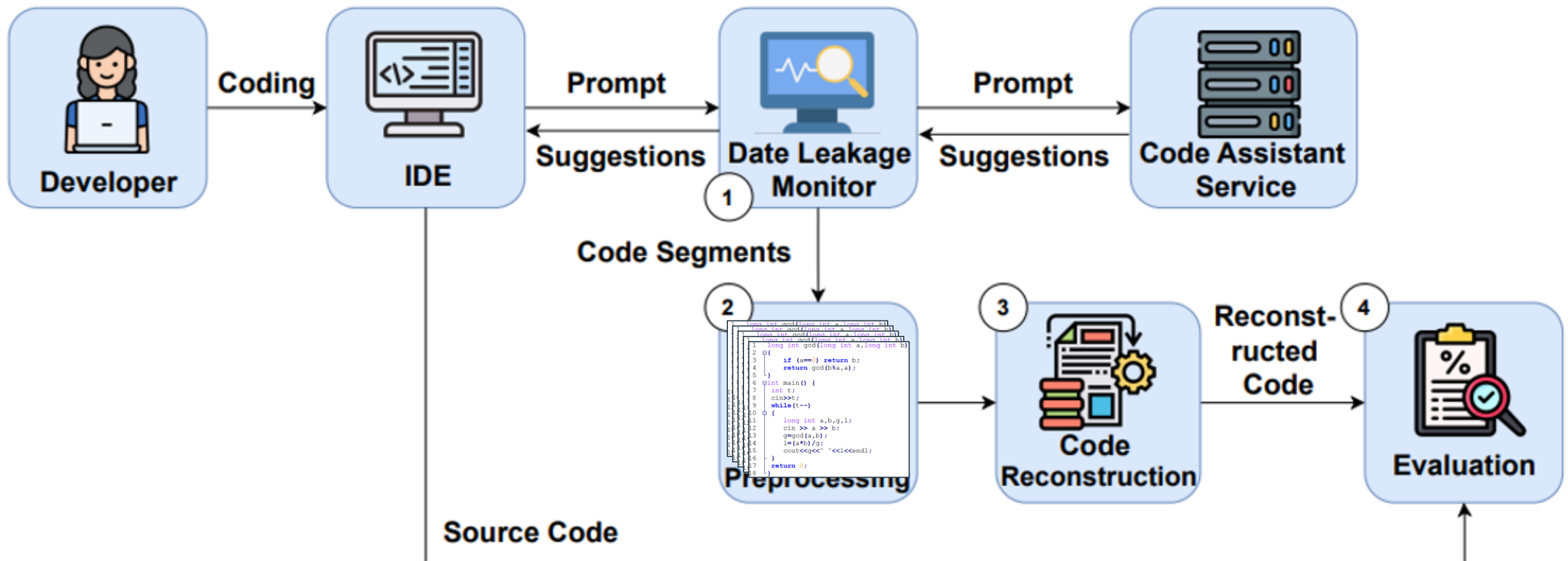
Threat Model



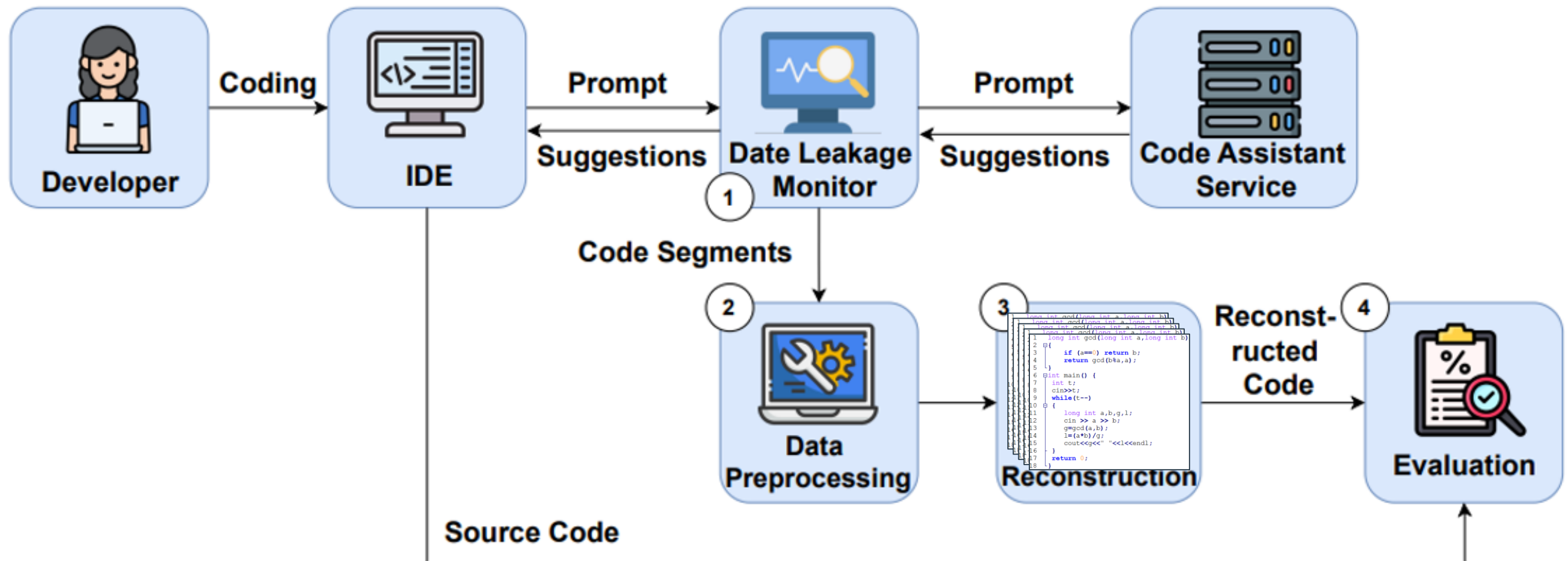
Threat Model



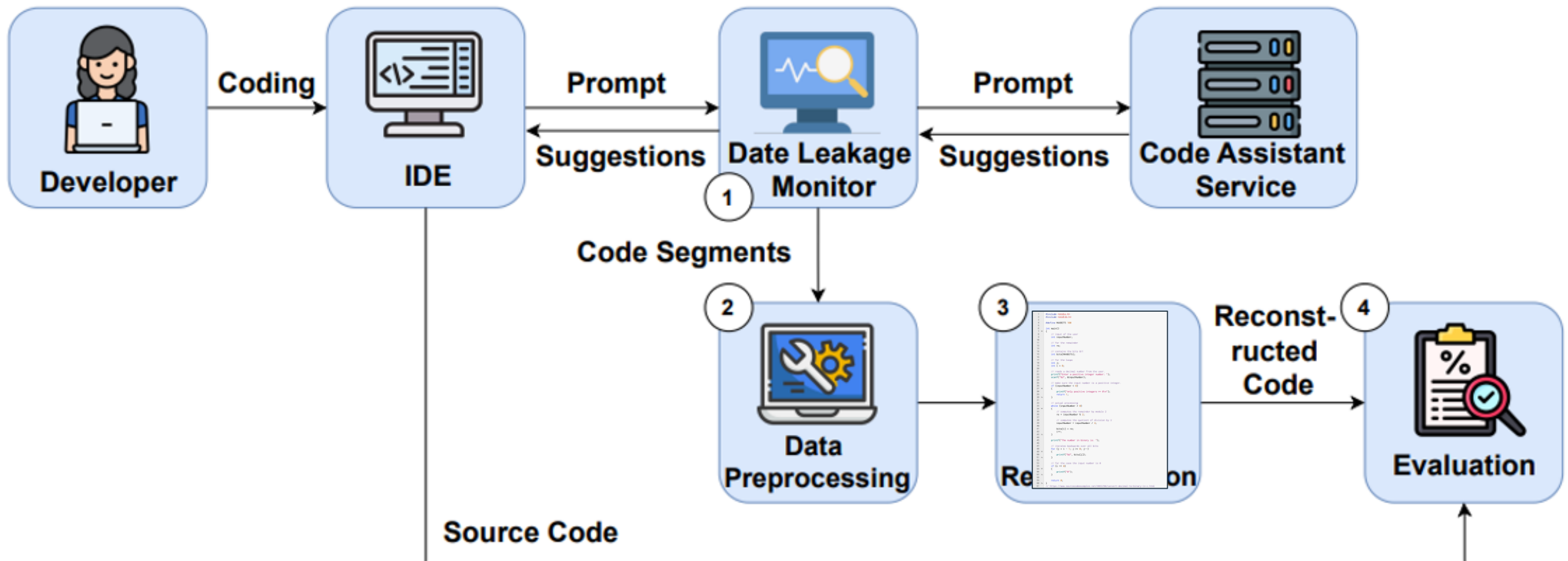
Threat Model



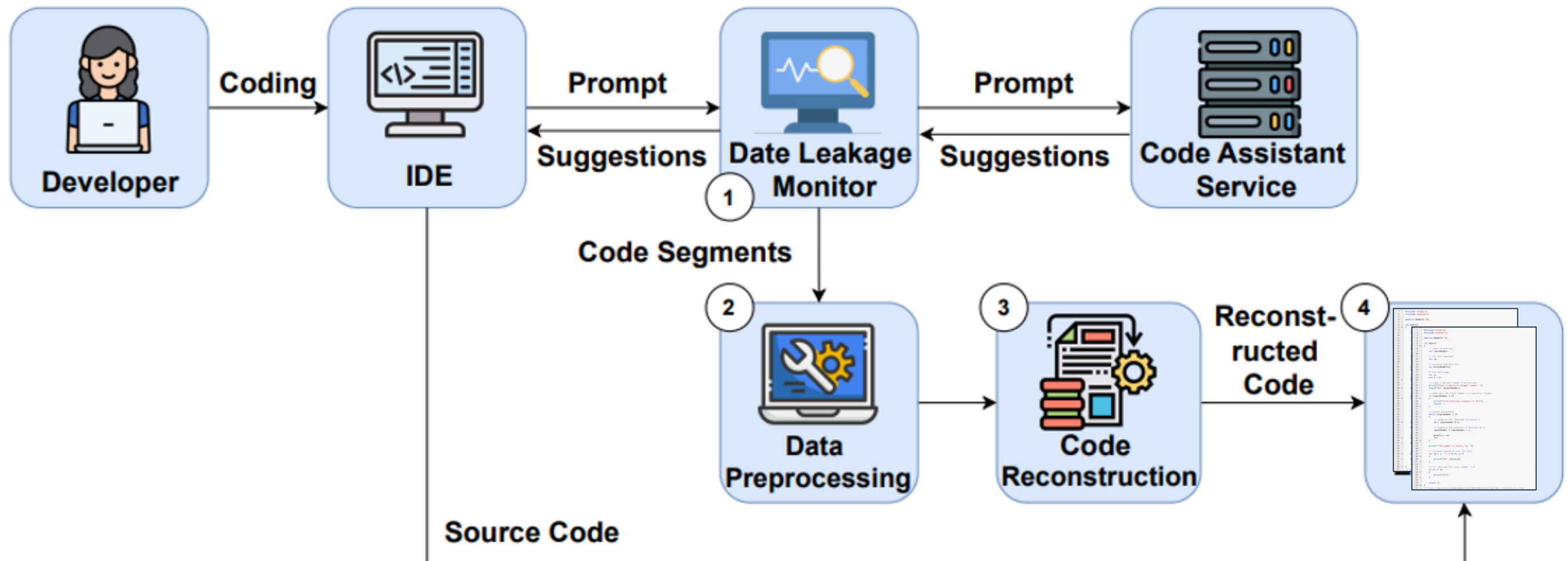
Threat Model



Threat Model



Threat Model



Threat Model

System Prompt

"Your task is to reconstruct python code of a developer that have been sent in segments at the time the developer was coding..."

User Prompt

```
Path: addition.py
'''Create the MPF files for the addition example and run using the scheduler'''
import os
from model.mpf.interfaces.actr import actr_to_mpf
```

```
Path: addition.py
'''Create the MPF files for the addition example and run using the scheduler'''
import os
from model.mpf.interfaces.actr import actr_to_mpf
from model.mpf.execution_engine import EvaluableGraph
from model.mpf.villa import load_mpf
def main():
```

```
Path: addition.py
'''Create the MPF files for the addition example and run using the scheduler'''
import os
from model.mpf.interfaces.actr import actr_to_mpf
from model.mpf.execution_engine import EvaluableGraph
from model.mpf.villa import load_mpf
def main():
    '''Take addition.lisp, converts to MPF and runs using the scheduler.'''
    file_name = os.path.dirname(os.path.realpath(__file__)) + "/addition.lisp"
    print(file_name)
```

```
Path: addition.py
'''Create the MPF files for the addition example and run using the scheduler'''
import os
from model.mpf.interfaces.actr import actr_to_mpf
from model.mpf.execution_engine import EvaluableGraph
from model.mpf.villa import load_mpf
def main():
    '''Take addition.lisp, converts to MPF and runs using the scheduler.'''
    file_name = os.path.dirname(os.path.realpath(__file__)) + "/addition.lisp"
    print(file_name)
    mpf = actr_to_mpf(file_name)
    mpf_graph = load_mpf(file_name[:5] + ".json").graphs[0]
    eg = EvaluableGraph(graph=mpf_graph, vertices=False)
    term = False
    goal = {}
```

...



Reconstructed Code

```
'''Create the MPF files for the addition example and run using the scheduler'''
import os
from model.mpf.interfaces.actr import actr_to_mpf
from model.mpf.execution_engine import EvaluableGraph
from model.mpf.villa import load_mpf
def main():
    '''Take addition.lisp, converts to MPF and runs using the scheduler.'''
    file_name = os.path.dirname(os.path.realpath(__file__)) + "/addition.lisp"
    print(file_name)
    mpf = actr_to_mpf(file_name)
    mpf_graph = load_mpf(file_name[:5] + ".json").graphs[0]
    eg = EvaluableGraph(graph=mpf_graph, vertices=False)
    term = False
    goal = {}
    retrieval = {}
    while not term:
        eg.evaluate(initializer={"goal_input": goal, "ex_input": retrieval})
        term = {
            eg.nodes["check_termination"].evaluable_outputs["check_output"].curr_value
        }
        goal = {
            eg.nodes["fire_production"]
            .evaluable_outputs["fire_pred_output_to_goal"]
            .curr_value
        }
        retrieval = {
            eg.nodes["fire_production"]
            .evaluable_outputs["fire_pred_output_to_retrieval"]
            .curr_value
        }
    print("Final Goal:")
    print(eg.nodes["goal_buffer"].evaluable_outputs["goal_output"].curr_value)
    if __name__ == "__main__":
        main()
```

Threat Model

System Prompt

"Your task is to reconstruct python code of a developer that have been sent in segments at the time the developer was coding..."

Reconstructed Code

```
"""Create the RDP files for the addition example and run using the scheduler."""  
import os  
from addnet_net.interfaces.actr import actr_to_rdp  
from addnet_net.scheduler.scheduler import Scheduler
```

From this Prompts that was Sent to the Service Provider We Succeed to Recover ~80% of the Source Codes!

```
from addnet_net.interfaces.actr import actr_to_rdp  
from addnet_net.scheduler.scheduler import Scheduler  
from addnet_net.utils import load_net  
def main():  
    """Take addition task, converts to RDP, and runs using the scheduler."""  
    file_name = os.path.dirname(os.path.realpath(__file__)) + "/addition.rdp"  
    print(file_name)  
    net = actr_to_rdp(file_name)  
    net_graph = load_net(file_name)[-1] + ".json".graphviz  
    sg = EvaluableGraph(graph=net_graph, vertices=net_graph.vertices)  
    task = False  
    goal = 0
```

```
.evaluable_outputs["fire_pred_output_to_retrieval"]  
.curr_value  
}  
print("Final Goal:")  
print(sg.nodes["goal_buffer"].evaluable_outputs["goal_output"].curr_value)  
if __name__ == "__main__":  
    main()
```


Threat Model

System Prompt

"Your task is to reconstruct python code of a developer that have been sent in segments at the time the developer was coding..."

User Prompt

```
Path: addition.py
"""Create the MDP files for the addition example and run using the scheduler."""
import os
from model.mdp.interfaces.actr import actr_to_mdp
from model.mdp.execution_engine import EvaluableGraph
from model.mdp.villa import load_mdp
def main():
    """Takes addition.tlog, converts to MDP, and runs using the scheduler."""
    file_name = os.path.dirname(os.path.realpath(__file__)) + "/addition.tlog"
    print(file_name)
    mdp = actr_to_mdp(file_name)
    mdp_graph = load_mdp(file_name)[-1] + ".json".graph[0]
    eg = EvaluableGraph(graph=mdp_graph, vertices=False)
    term = False
    goal = {}
```

Reconstructed Code

```
"""Create the MDP files for the addition example and run using the scheduler."""
import os
from model.mdp.interfaces.actr import actr_to_mdp
from model.mdp.execution_engine import EvaluableGraph
from model.mdp.villa import load_mdp
def main():
    """Takes addition.tlog, converts to MDP, and runs using the scheduler."""
    file_name = os.path.dirname(os.path.realpath(__file__)) + "/addition.tlog"
    print(file_name)
    mdp = actr_to_mdp(file_name)
    mdp_graph = load_mdp(file_name)[-1] + ".json".graph[0]
    eg = EvaluableGraph(graph=mdp_graph, vertices=False)
    term = False
    goal = {}
    retrieval = {}
    print("Final Goal:")
    print(eg.nodes["goal_buffer"].evaluable_outputs["goal_output"].curr_value)
    if __name__ == "__main__":
        main()
```

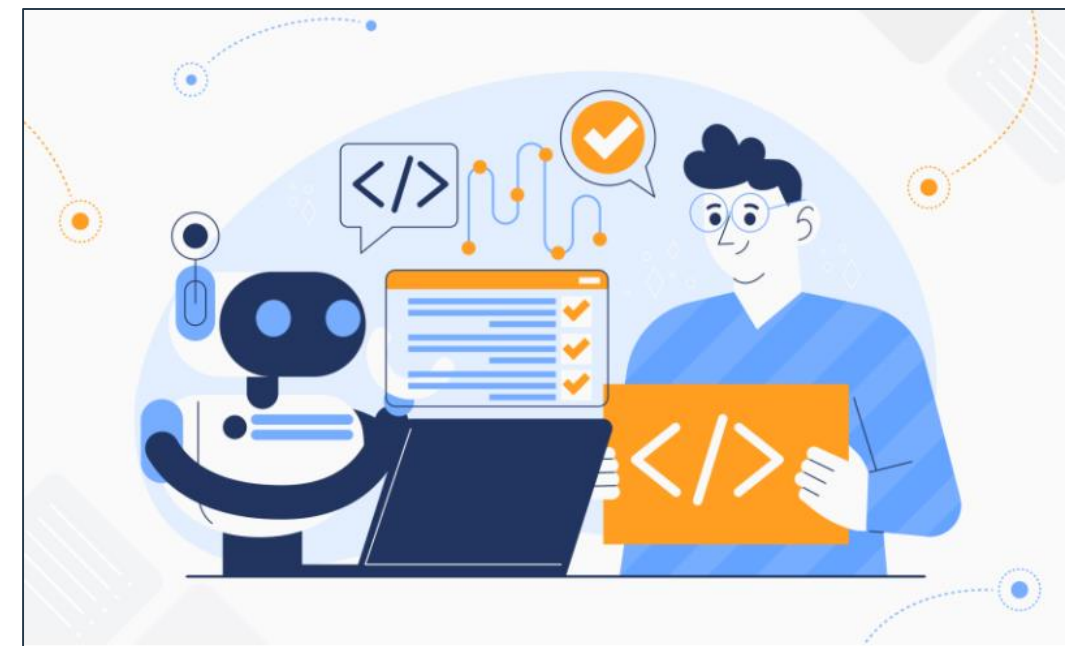
The Solution: CodeCloak



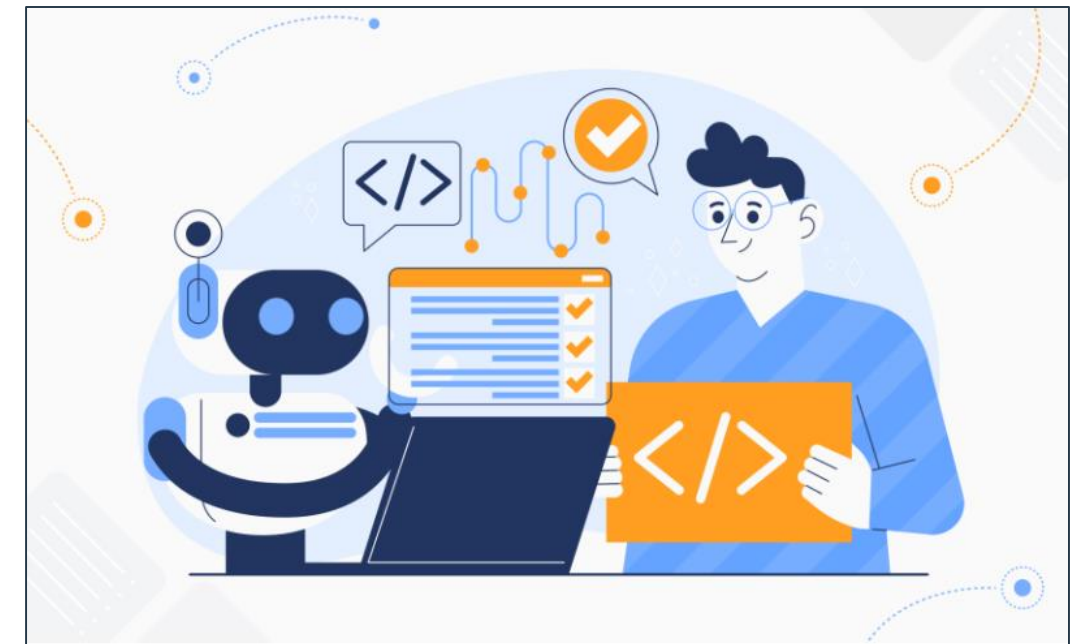
1. **Intro**
2. **Background**
3. **Threat Model**
4. **Countermeasure**
5. **Takeaways**
6. **Future Steps**
7. **Q&A**

1. **Intro**
2. **Background**
3. **Threat Model**
4. **Countermeasure**
5. **Takeaways**
6. **Future Steps**
7. **Q&A**

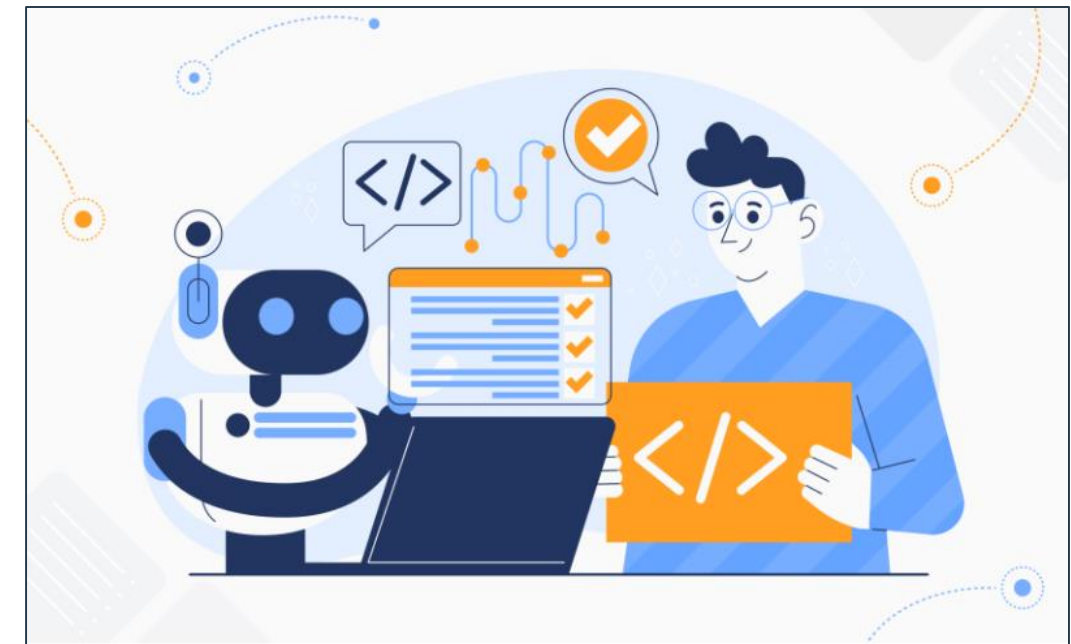
Mitigating Risks in AI-code assistants models



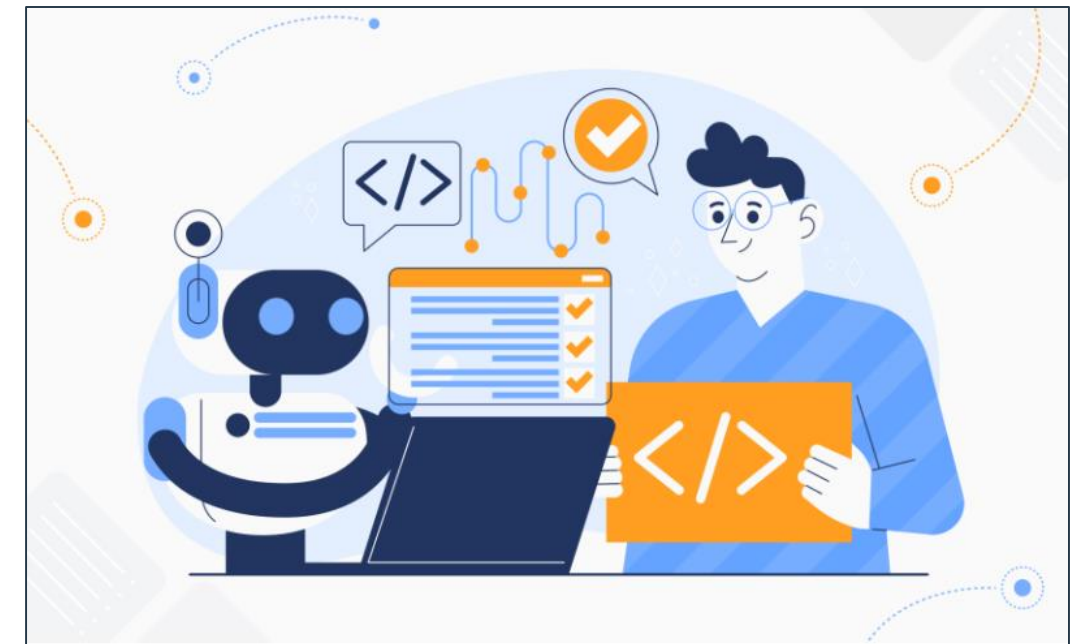
1. Protecting Intellectual Property.



1. **Protecting Intellectual Property.**
2. **Protect Organizations.**



1. **Protecting Intellectual Property.**
2. **Protect Organizations.**
3. **Strengthen AI code assistants models.**



CodeCloak: Goal

- **Reduce code leakage.**
- **Preserve the AI code assistant productivity**



CodeCloak: Example

Original prompt

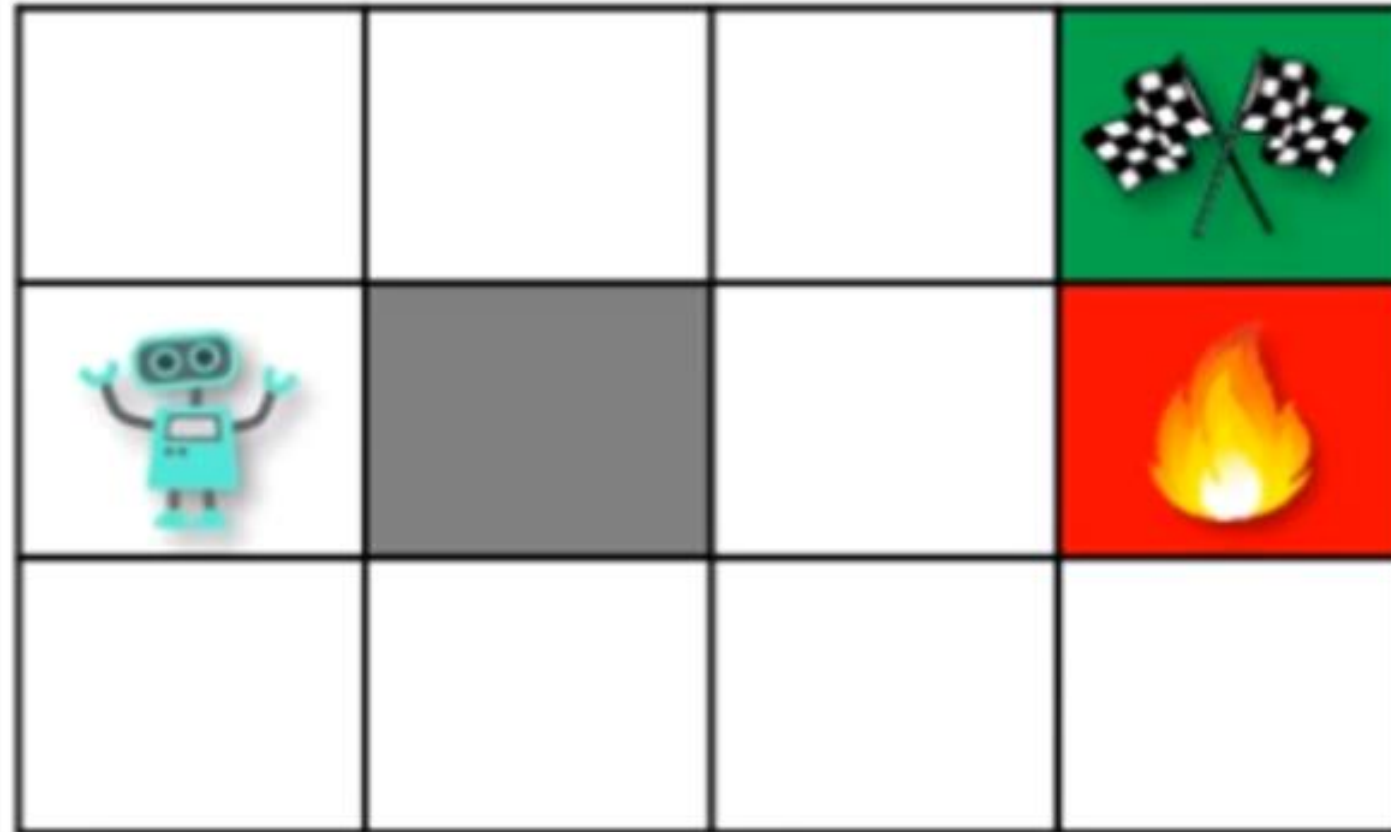
```
42 while must_retry and may_retry:
43     must_retry = not predicate()
44     if must_retry:
45         net_time = now() - start_time
46         if timeout and net_time + sleep_duration > timeout:
47             may_retry = False
48         else:
49             left = "[-%d sec] " % (timeout - net_time) if timeout else ""
50             log_info("-- waiting %s--" % left)
51             time.sleep(sleep_duration)
52     return not must_retry
53
54 4 usages
55 def now():
56     return time.time()
57
58 # OS Functions
59 def which(program):
60     fpath, fname = os.path.split(program)
61     if fpath:
62         if is_exe(program):
63             return program
64     else:
65         for path in os.environ["PATH"].split(os.pathsep):
66             exe_file = os.path.join(path, program)
67             if is_exe(exe_file):
68                 return exe_file
```

same suggestion

CodeCloak Prompt

```
16
17 def ibda7(thing):
18     # Return a string representation of a None or None in Pyt
19
20 def m8edb(document):
21     # pretty_print_token in python
22
23 def w8939(object):
24     # Listify a Python object
25
26 def m519a(value):
27     # Check if value is a URL in a URL scheme
28
29 def w62a1():
30     return time.time()
31
32 # OS Functions
33 def h80f8(program):
34     fpath, fname = os.path.split(program)
35     if fpath:
36         if is_exe(program):
37             return program
38     else:
39         for path in os.environ["PATH"].split(os.pathsep):
40             exe_file = os.path.join(path, program)
41             if is_exe(exe_file):
42                 return exe_file
```

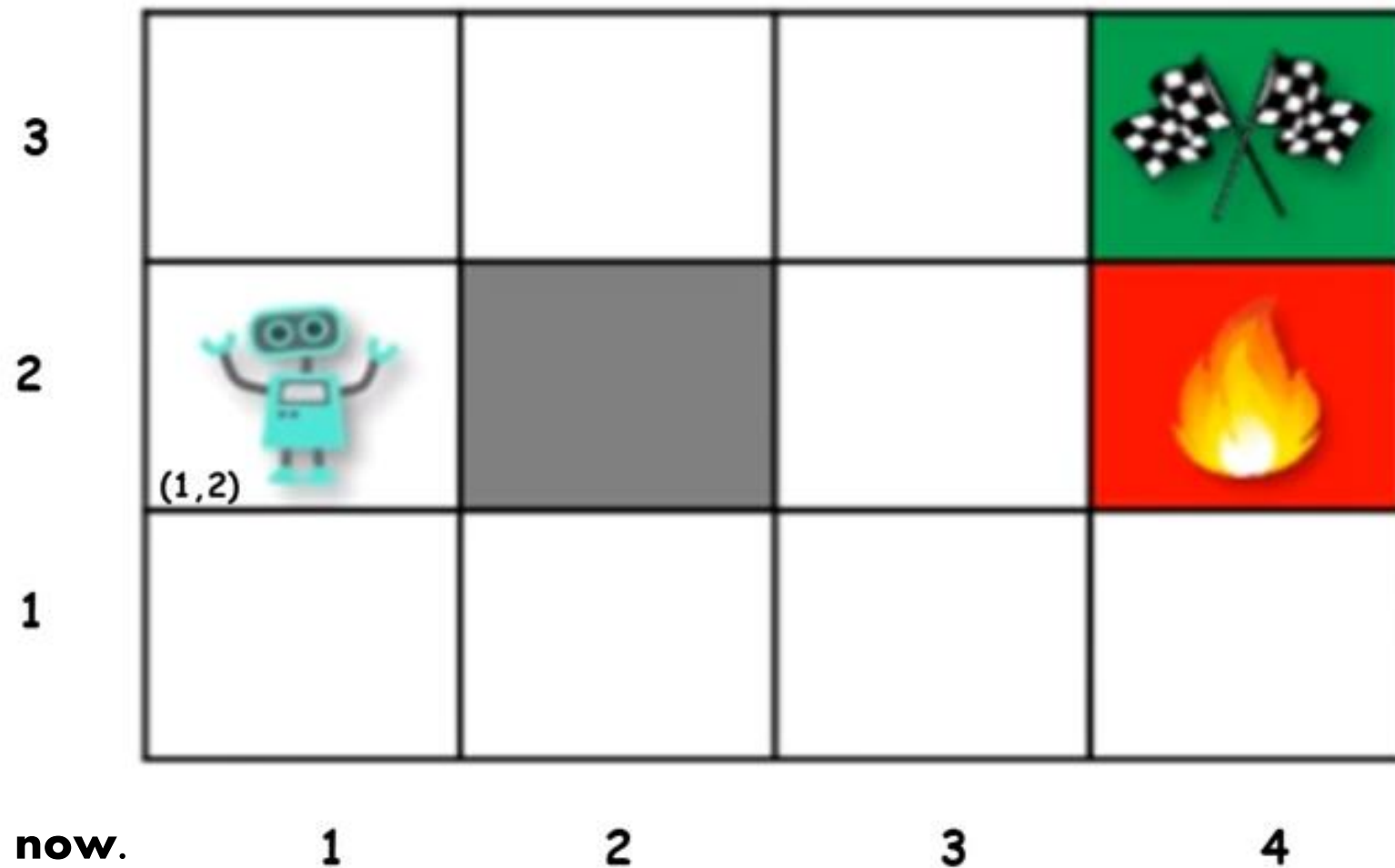
Reinforcement Learning



Reinforcement Learning

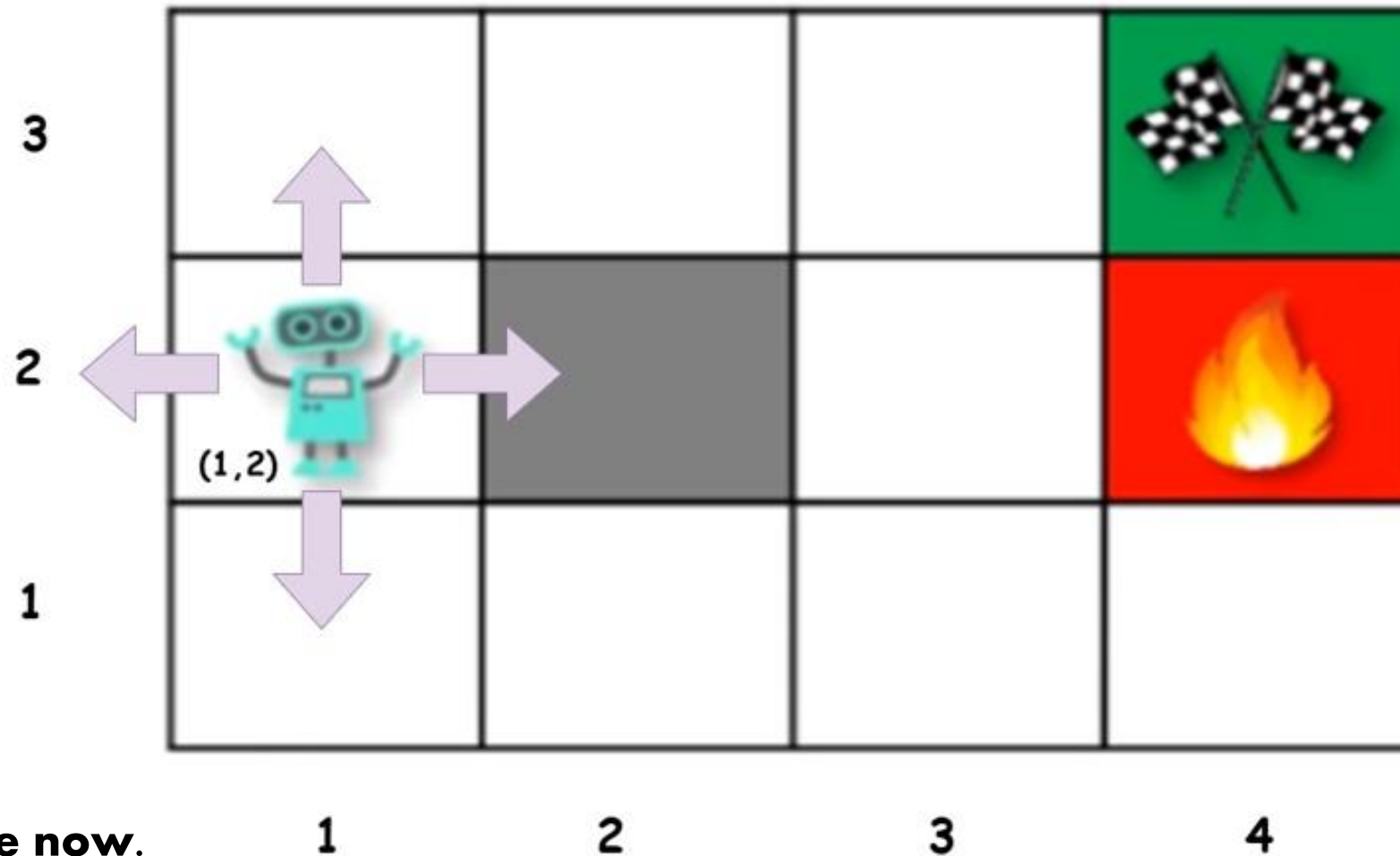
- **States: Where we are now.**
- **Actions: What we can do.**
- **Rewards: Feedback for doing it right.**

Reinforcement Learning (states)



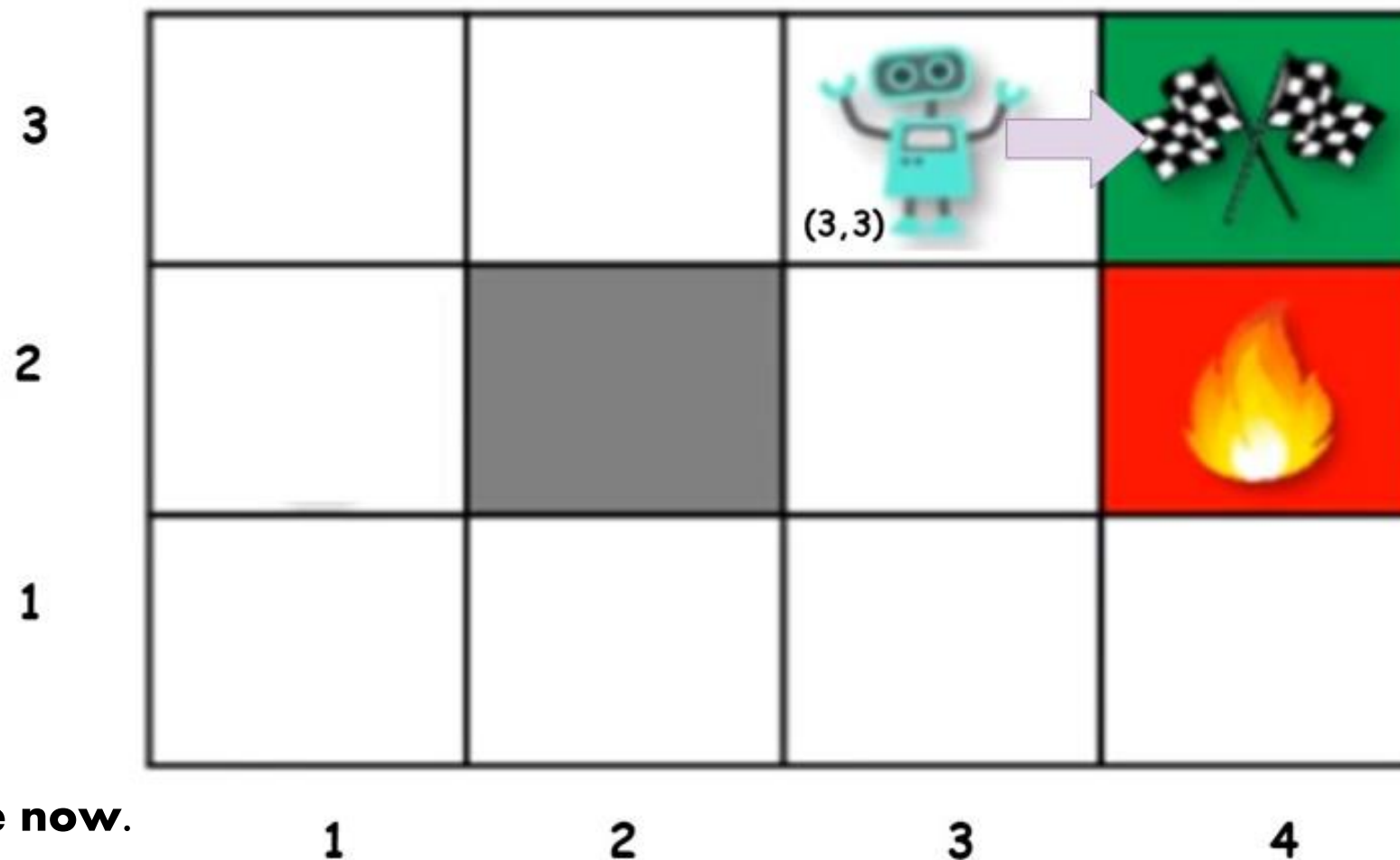
- **States:** Where we are now.
- **Actions:** What we can do.
- **Rewards:** Feedback for doing it right.

Reinforcement Learning (actions)



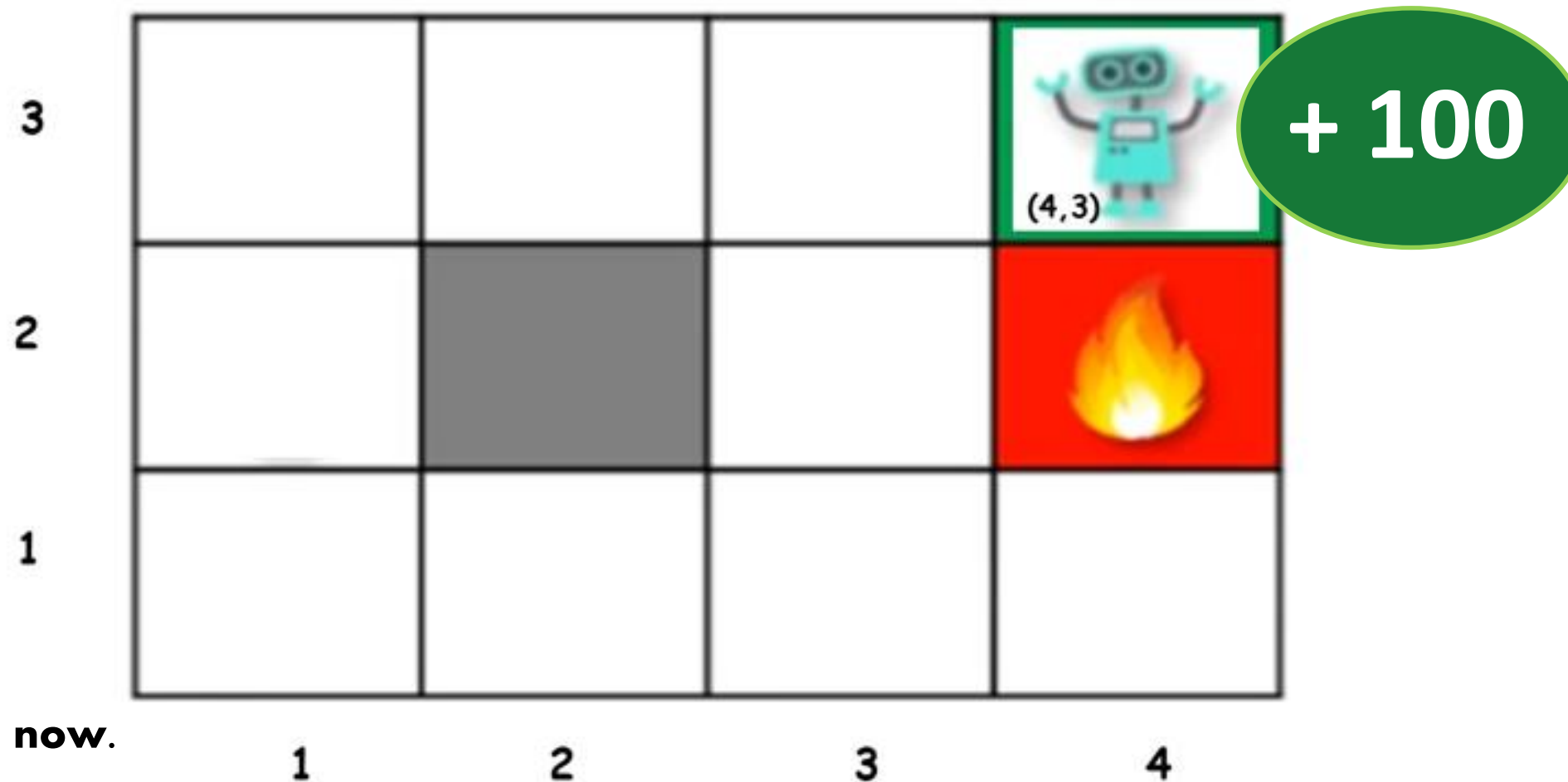
- **States:** Where we are now.
- **Actions:** What we can do.
- **Rewards:** Feedback for doing it right.

Reinforcement Learning (rewards)



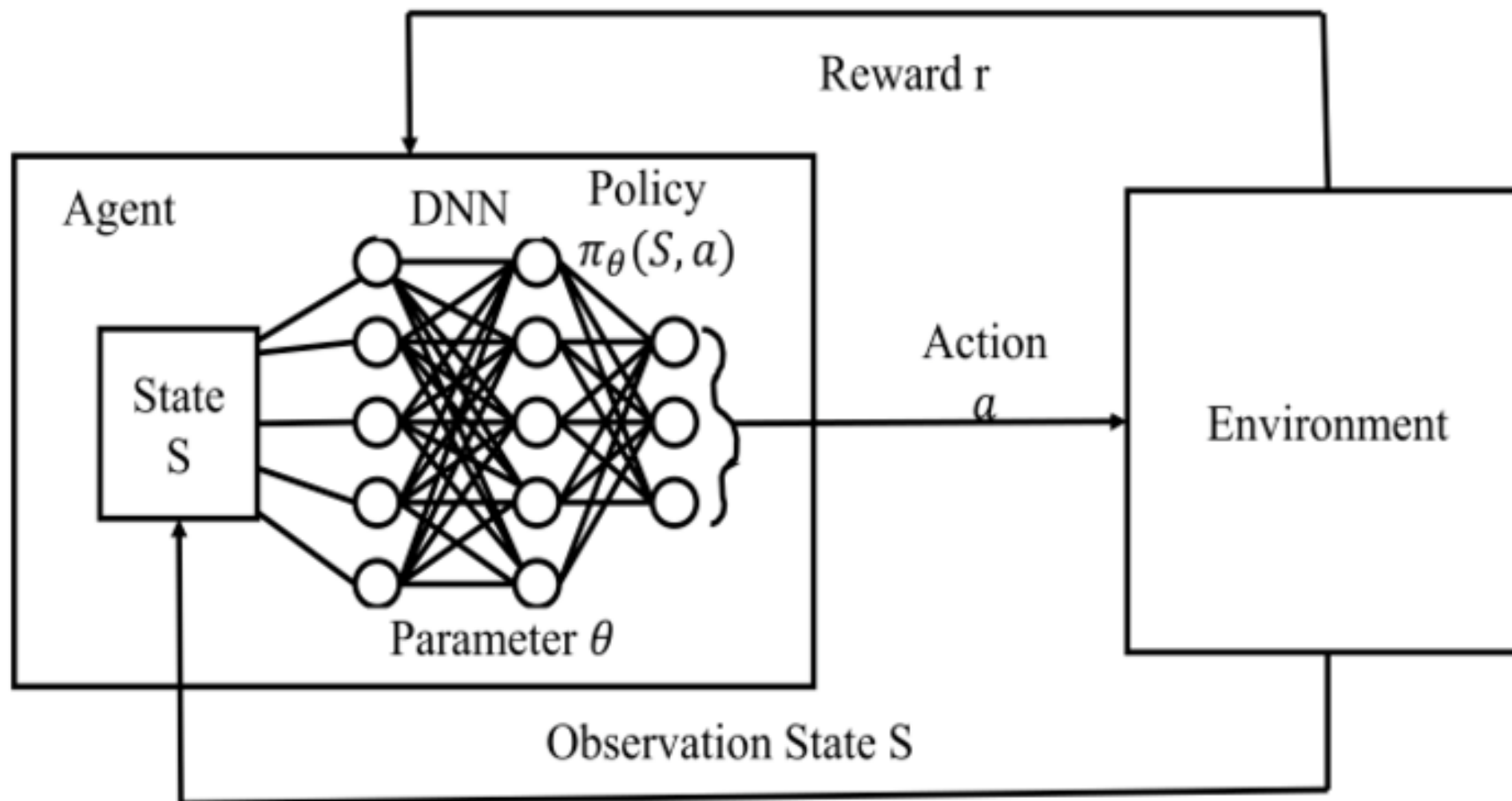
- **States:** Where we are now.
- **Actions:** What we can do.
- **Rewards:** Feedback for doing it right.

Reinforcement Learning (rewards)





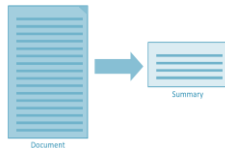

- **States:** Where we are now.
- **Actions:** What we can do.
- **Rewards:** Feedback for doing it right.


From RL to DRL



CodeCloak modeling

States = {  ,  ,  }

Actions = {  ,  ,  , ... ,  }

Rewards = 

CodeCloak modeling - states

```

pip install opencv-python pyautogui numpy keyboard

import cv2
import numpy as np
import pyautogui
import keyboard

screen_size = pyautogui.size()
fps = 20
fourcc = cv2.VideoWriter_fourcc(*"XVID")
output_file = "screen_recording_clcoding.mp4"
out = cv2.VideoWriter(output_file, fourcc, fps,
                      (screen_size.width, screen_size.height))

print("Recording... Press 'q' to stop.")
while True:

    screen = pyautogui.screenshot()
    frame = np.array(screen)
    frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
    out.write(frame)

    if keyboard.is_pressed('q'):
        print("Recording stopped.")
        break

out.release()
print(f"Video saved to {output_file}")

#source code --> clcoding.com

```



```

pip install opencv-python pyautogui numpy keyboard

import cv2
import numpy as np
import pyautogui
import keyboard

screen_size = pyautogui.size()
fps = 20
fourcc = cv2.VideoWriter_fourcc(*"XVID")

fourcc = cv2.VideoWriter_fourcc(*"XVID")
output_file = "screen_recording_clcoding.mp4"
out = cv2.VideoWriter(output_file, fourcc, fps,
                      (screen_size.width, screen_size.height))

print("Recording... Press 'q' to stop.")
while True:

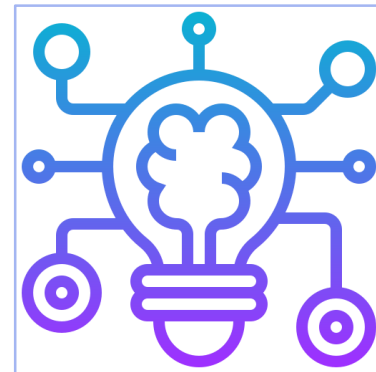
    screen = pyautogui.screenshot()
    frame = np.array(screen)
    frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
    out.write(frame)

    if keyboard.is_pressed('q'):
        print("Recording stopped.")
        break

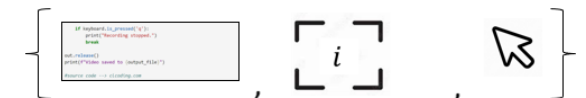
out.release()
print(f"Video saved to {output_file}")

#source code --> clcoding.com

```



State



Examples:



Examples:

- **Detect and replace personally identifiable information (PII)**



Examples:

- **Detect and replace personally identifiable information (PII)**
- **Change/delete/insert random lines of code**



Examples:

- . **Detect and replace personally identifiable information (PII)**
- . **Change/delete/insert random lines of code**
- . **Delete function bodies and replace with summaries**



Examples:

- **Detect and replace personally identifiable information (PII)**
- **Change/delete/insert random lines of code**
- **Delete function bodies and replace with summaries**
- **Rename variables, functions, arguments**



Examples:

- **Detect and replace personally identifiable information (PII)**
- **Change/delete/insert random lines of code**
- **Delete function bodies and replace with summaries**
- **Rename variables, functions, arguments**
- **Stop manipulations and send the manipulated prompt**



CodeCloak modeling - actions

```
fourcc = cv2.VideoWriter_fourcc(*"XVID")
output_file = "screen_recording_clcoding.mp4"
out = cv2.VideoWriter(output_file, fourcc, fps,
                      (screen_size.width, screen_size.height))

print("Recording... Press 'q' to stop.")
while True:

    screen = pyautogui.screenshot()
    frame = np.array(screen)
    frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
    out.write(frame)
```



CodeCloak modeling - actions

```
fourcc = cv2.VideoWriter_fourcc(*"XVID")
output_file = "screen_recording_clcoding.mp4"
out = cv2.VideoWriter(output_file, fourcc, fps,
                      (screen_size.width, screen_size.height))

print("Recording... Press 'q' to stop.")
while True:

    screen = pyautogui.screenshot()
    frame = np.array(screen)
    frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
    out.write(frame)
```

Prompt Manipulation:
delete lines



CodeCloak modeling - actions

```
fourcc = cv2.VideoWriter_fourcc(*"XVID")  
output_file = "screen_recording_clcoding.mp4"  
out = cv2.VideoWriter(output_file, fourcc, fps,  
                        (screen_size.width, screen_size.height))  
  
print("Recording... Press 'q' to stop.")  
while True:  
  
    screen = pyautogui.screenshot()  
    frame = np.array(screen)  
    frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)  
    out.write(frame)
```

Prompt Manipulation:
delete lines



CodeCloak modeling - actions

```
output_file = "screen_recording_clcoding.mp4"

print("Recording... Press 'q' to stop.")
while True:
    screen = pyautogui.screenshot()
    frame = np.array(screen)
    out.write(frame)
```

Prompt Manipulation:
delete lines



CodeCloak modeling – rewards

Machine translation:

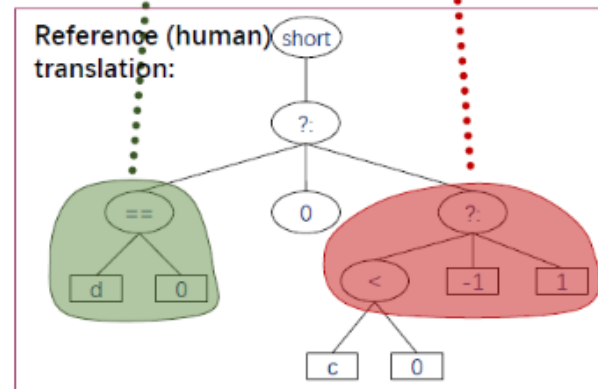
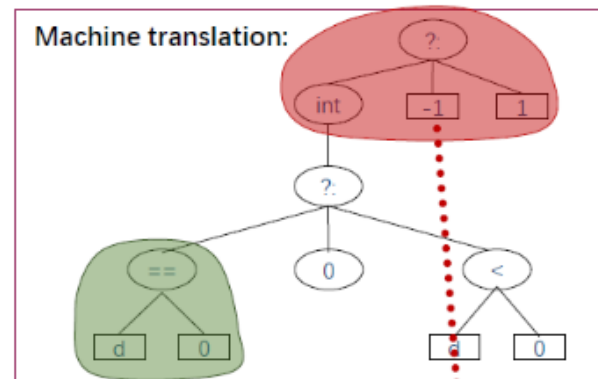
```
public static int Sign ( double d )
{
    return ( (int) ) ( ( d == 0 ) ? 0 : ( d < 0 ) ) ?
    -1 : 1;
}
```

Reference (human) translation:

```
public static short Sign ( double d )
{
    return ( short ) ( ( d == 0 ) ? 0 : ( d < 0 ) ?
    -1 : 1 );
}
```

Weighted N-Gram Match

1.0 1.0 0.7 0.5



Syntactic AST Match

[('d', 7, 'comesFrom', [], []),
('d', 16, 'comesFrom', ['d'], [7]),
('d', 24, 'comesFrom', ['d'], [7])]

Machine translation:

```
public static int Sign ( double d )
{
    return ( (int) ) ( ( d == 0 ) ? 0 : ( d < 0 ) ) ?
    -1 : 1;
}
```

Reference (human) translation:

```
public static short Sign ( double c )
{
    return ( short ) ( ( c == 0 ) ? 0 : ( c < 0 ) ?
    -1 : 1 );
}
```

Semantic Data-flow Match

$$\text{CodeBLEU} = \alpha \cdot \text{N-Gram Match (BLEU)} + \beta \cdot \text{Weighted N-Gram Match} + \gamma \cdot \text{Syntactic AST Match} + \delta \cdot \text{Semantic Data-flow Match}$$



CodeCloak modeling – rewards

```
fourcc = cv2.VideoWriter_fourcc(*"XVID")
output_file = "screen_recording_clcoding.mp4"
out = cv2.VideoWriter(output_file, fourcc, fps,
                      (screen_size.width, screen_size.height))

print("Recording... Press 'q' to stop.")
while True:

    screen = pyautogui.screenshot()
    frame = np.array(screen)
    frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
    out.write(frame)
```

```
fourcc = cv2.VideoWriter_fourcc(*"XVID")
output_file = "screen_recording_clcoding.mp4"
out = cv2.VideoWriter(output_file, fourcc, fps,
                      (screen_size.width, screen_size.height))

print("Recording... Press 'q' to stop.")
while True:

    screen = pyautogui.screenshot()
    frame = np.array(screen)
frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
    out.write(frame)
```

Prompts Similarity

0.83



CodeCloak modeling – rewards

```
fourcc = cv2.VideoWriter_fourcc(*"XVID")
output_file = "screen_recording_clcoding.mp4"
out = cv2.VideoWriter(output_file, fourcc, fps,
                      (screen_size.width, screen_size.height))

print("Recording... Press 'q' to stop.")
while True:

    screen = pyautogui.screenshot()
    frame = np.array(screen)
    frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
    out.write(frame)
```

```
if keyboard.is_pressed('q'): print("Recording stopped."); break
```

```
fourcc = cv2.VideoWriter_fourcc(*"XVID")
output_file = "screen_recording_clcoding.mp4"
out = cv2.VideoWriter(output_file, fourcc, fps,
                      (screen_size.width, screen_size.height))

print("Recording... Press 'q' to stop.")
while True:

    screen = pyautogui.screenshot()
    frame = np.array(screen)
frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
    out.write(frame)
```

```
if keyboard.is_pressed('q'): print("Recording stopped."); break
```

Suggestions Similarity

1.0



CodeCloak modeling – rewards

```
fourcc = cv2.VideoWriter_fourcc(*"XVID")
output_file = "screen_recording_clcoding.mp4"
out = cv2.VideoWriter(output_file, fourcc, fps,
                      (screen_size.width, screen_size.height))

print("Recording... Press 'q' to stop.")
while True:

    screen = pyautogui.screenshot()
    frame = np.array(screen)
    frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
    out.write(frame)
```

```
if keyboard.is_pressed('q'): print("Recording stopped."); break
```

```
fourcc = cv2.VideoWriter_fourcc(*"XVID")
output_file = "screen_recording_clcoding.mp4"
out = cv2.VideoWriter(output_file, fourcc, fps,
                      (screen_size.width, screen_size.height))

print("Recording... Press 'q' to stop.")
while True:

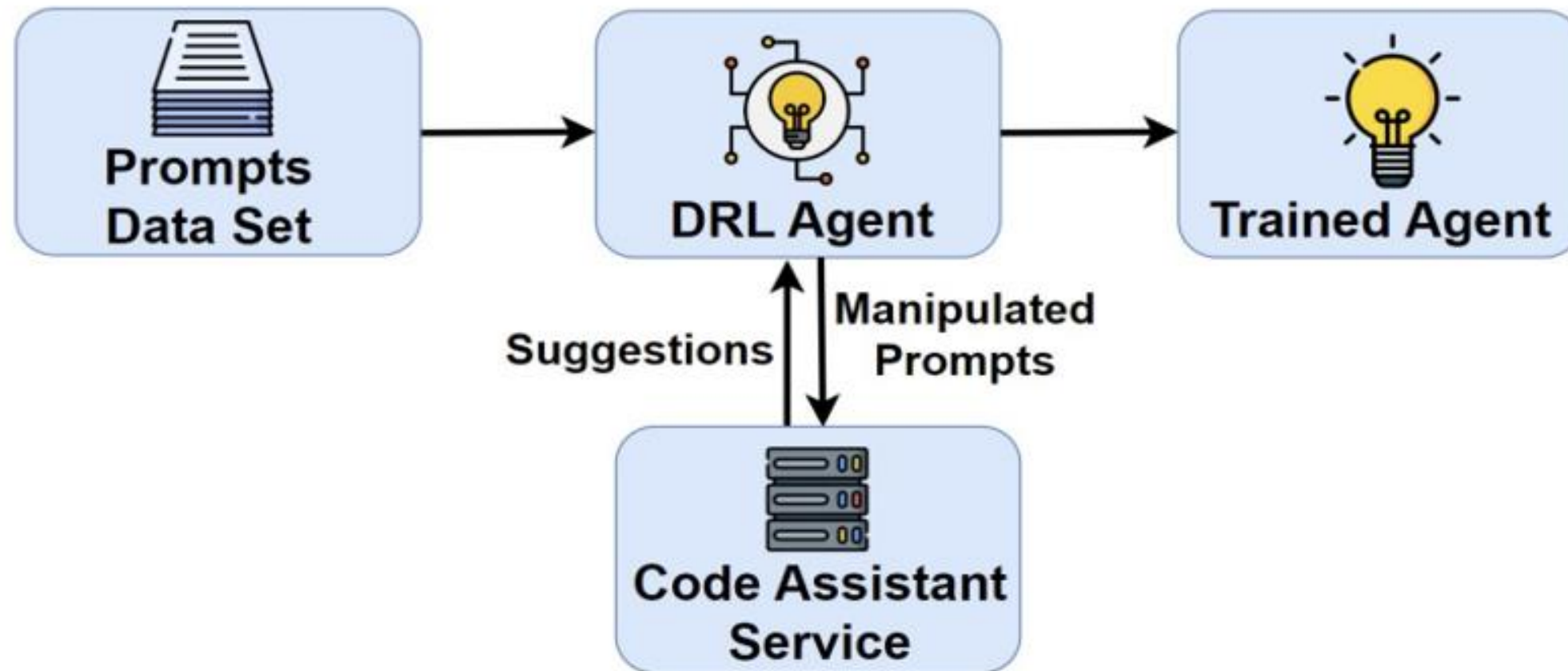
    screen = pyautogui.screenshot()
    frame = np.array(screen)
frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
    out.write(frame)
```

```
if keyboard.is_pressed('q'): print("Recording stopped."); break
```

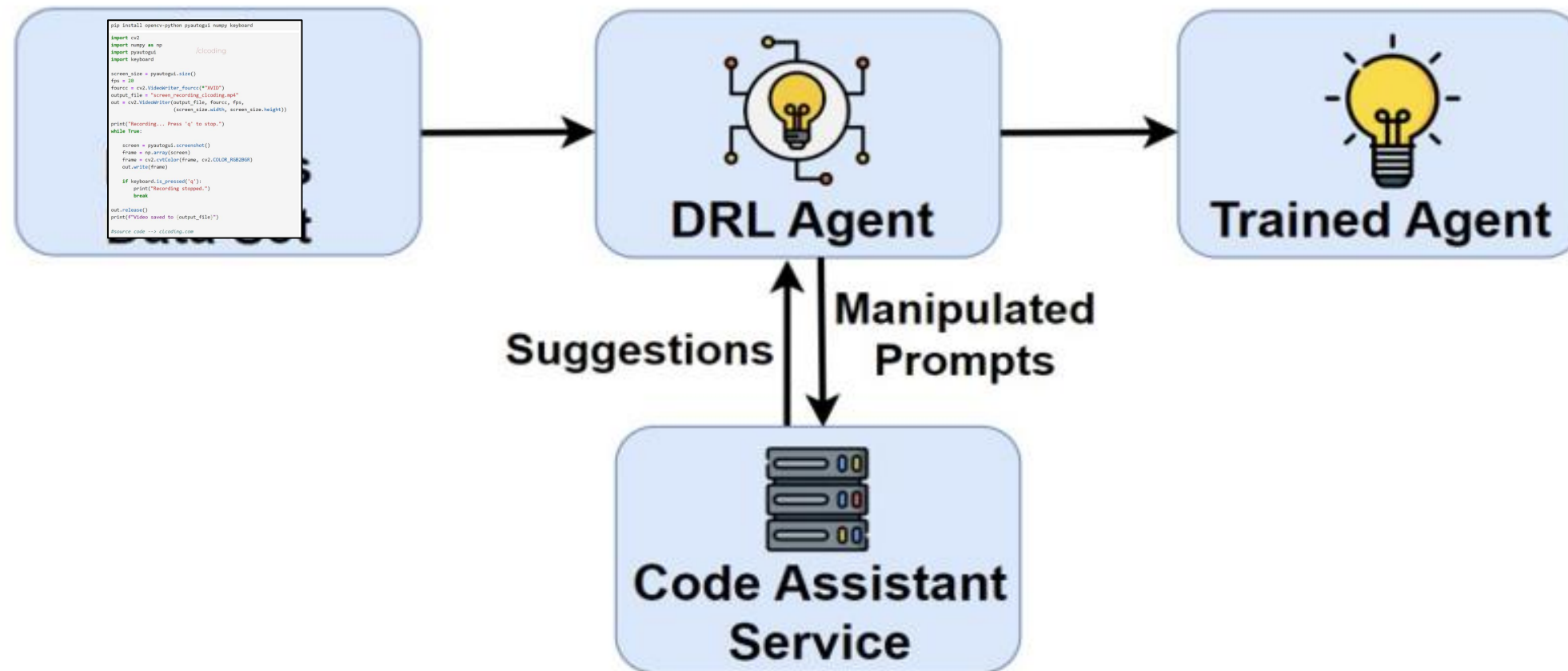
$$\text{Reward} = 1.0 - 0.83 = 0.17$$



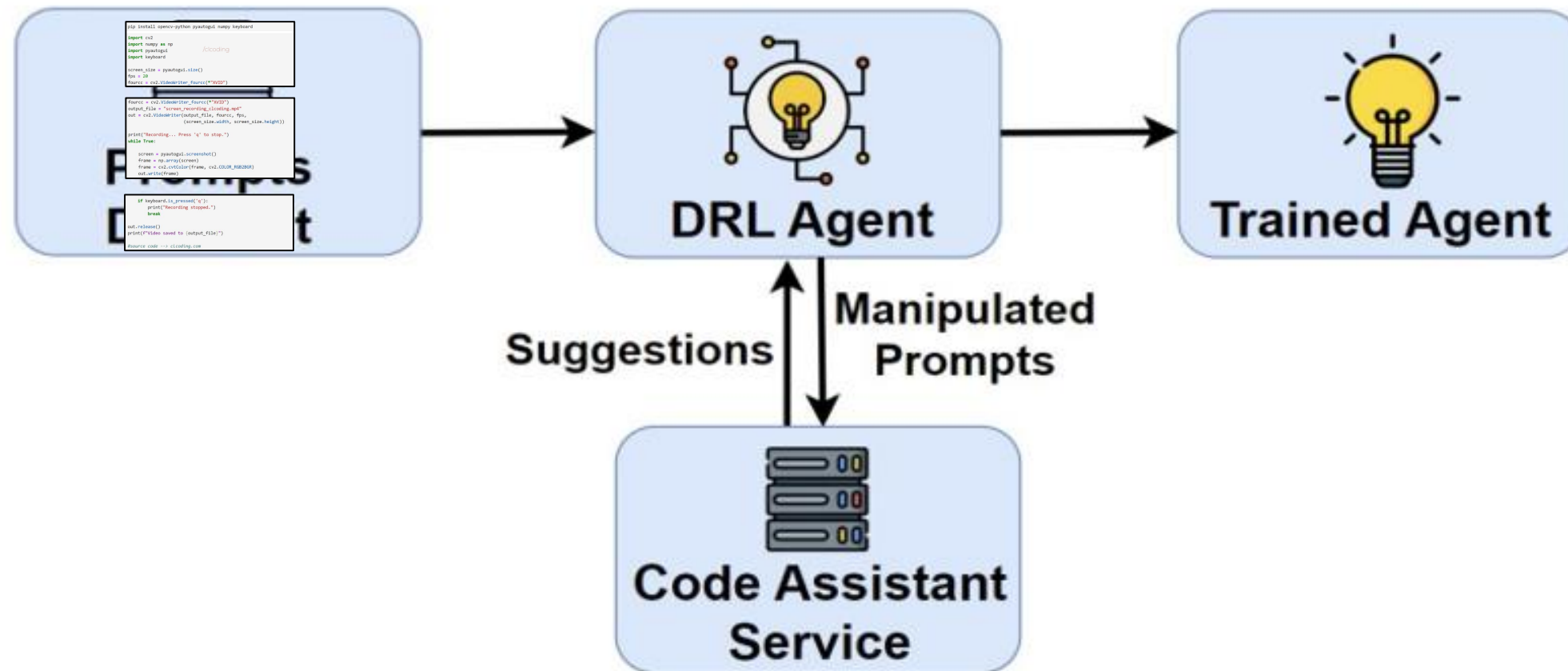
CodeCloak: Training Phase



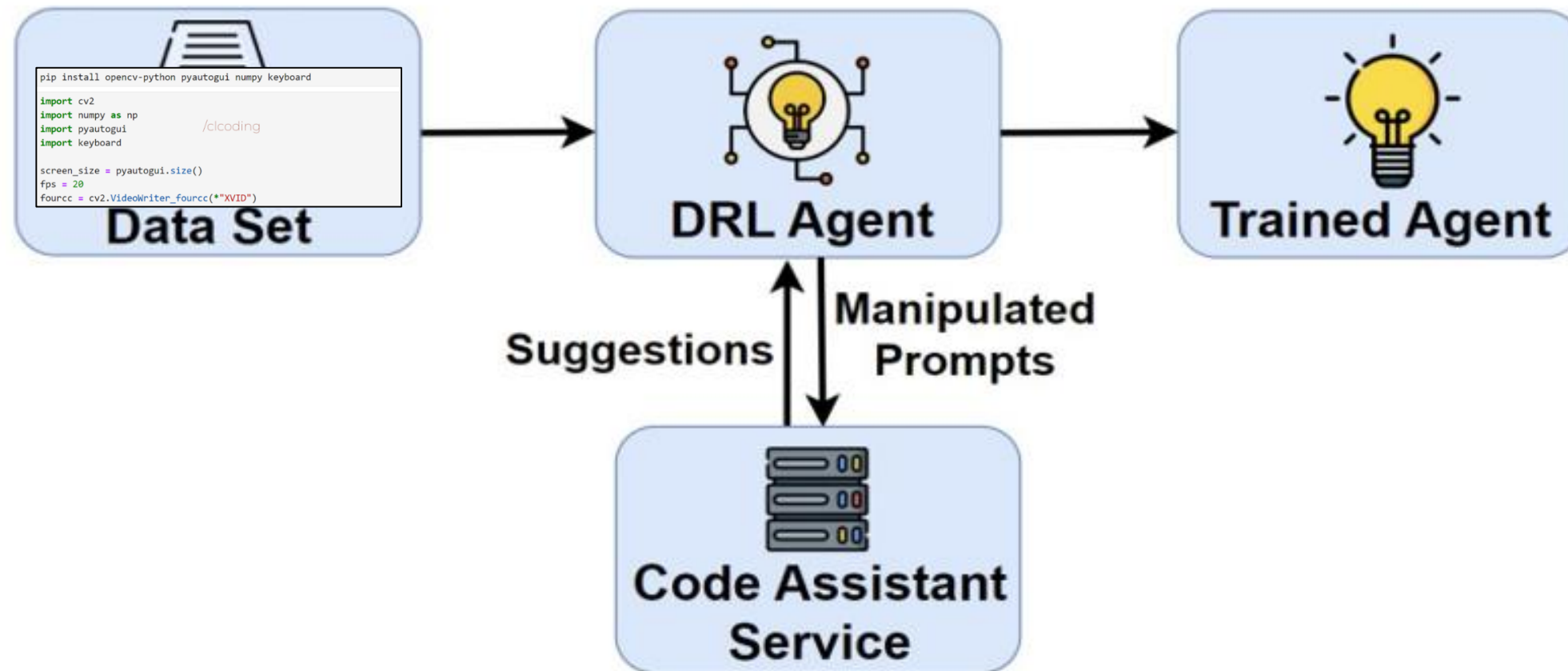
CodeCloak: Training Phase



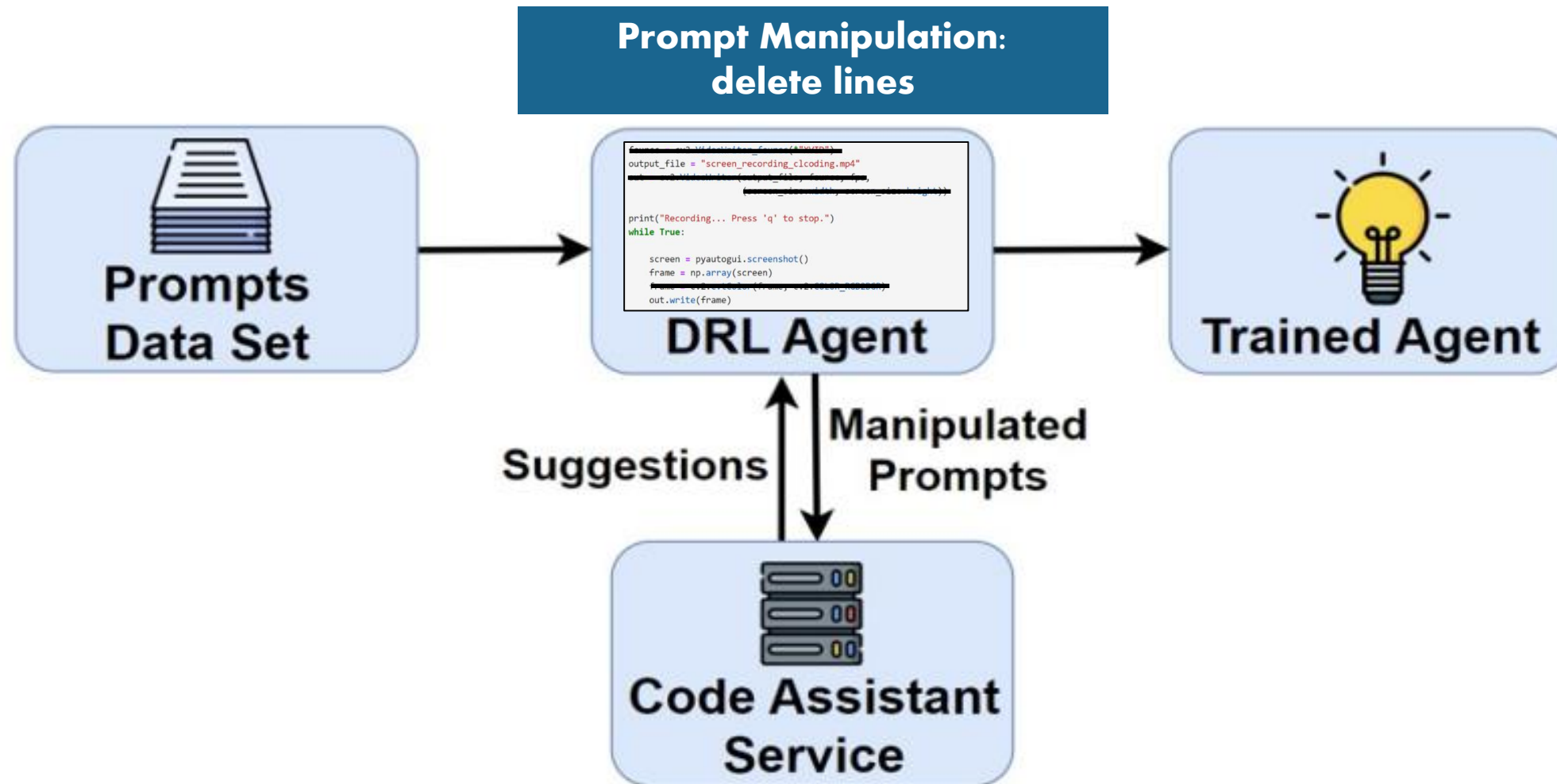
CodeCloak: Training Phase



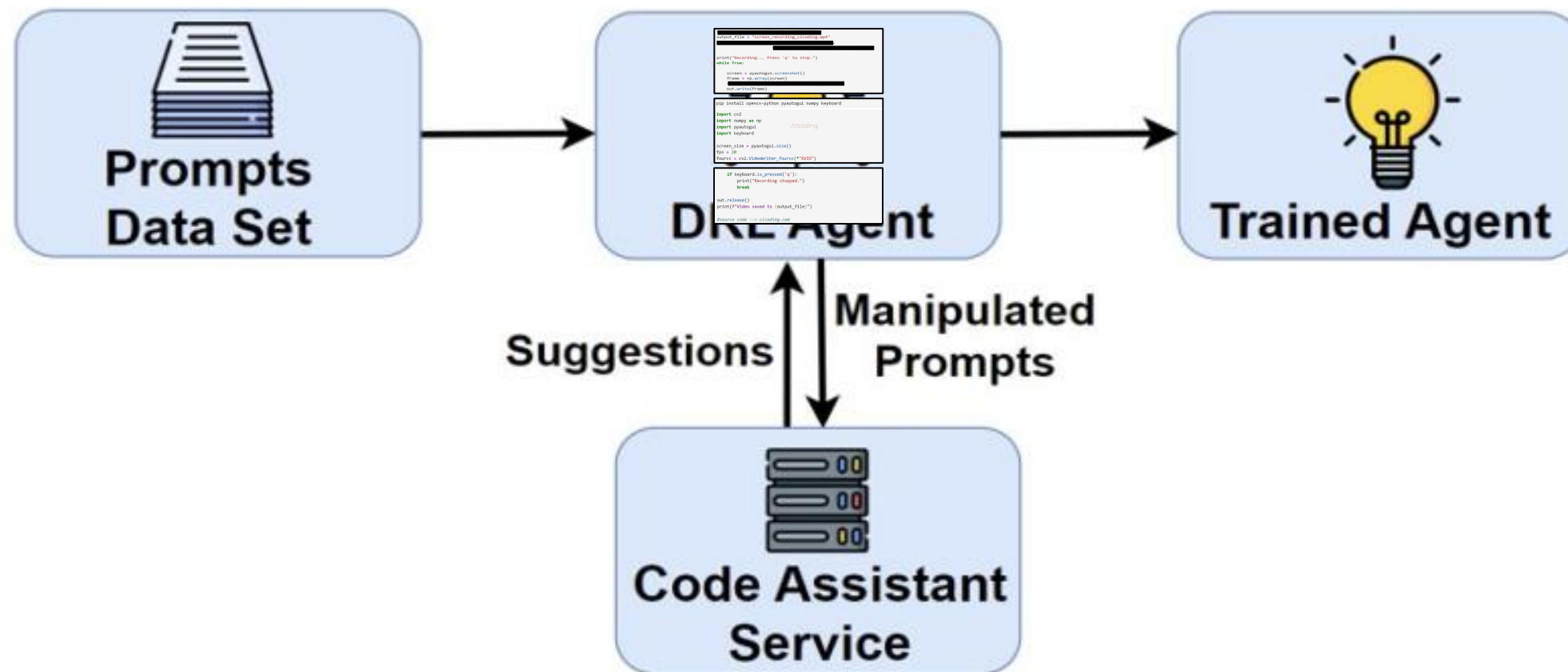
CodeCloak: Training Phase



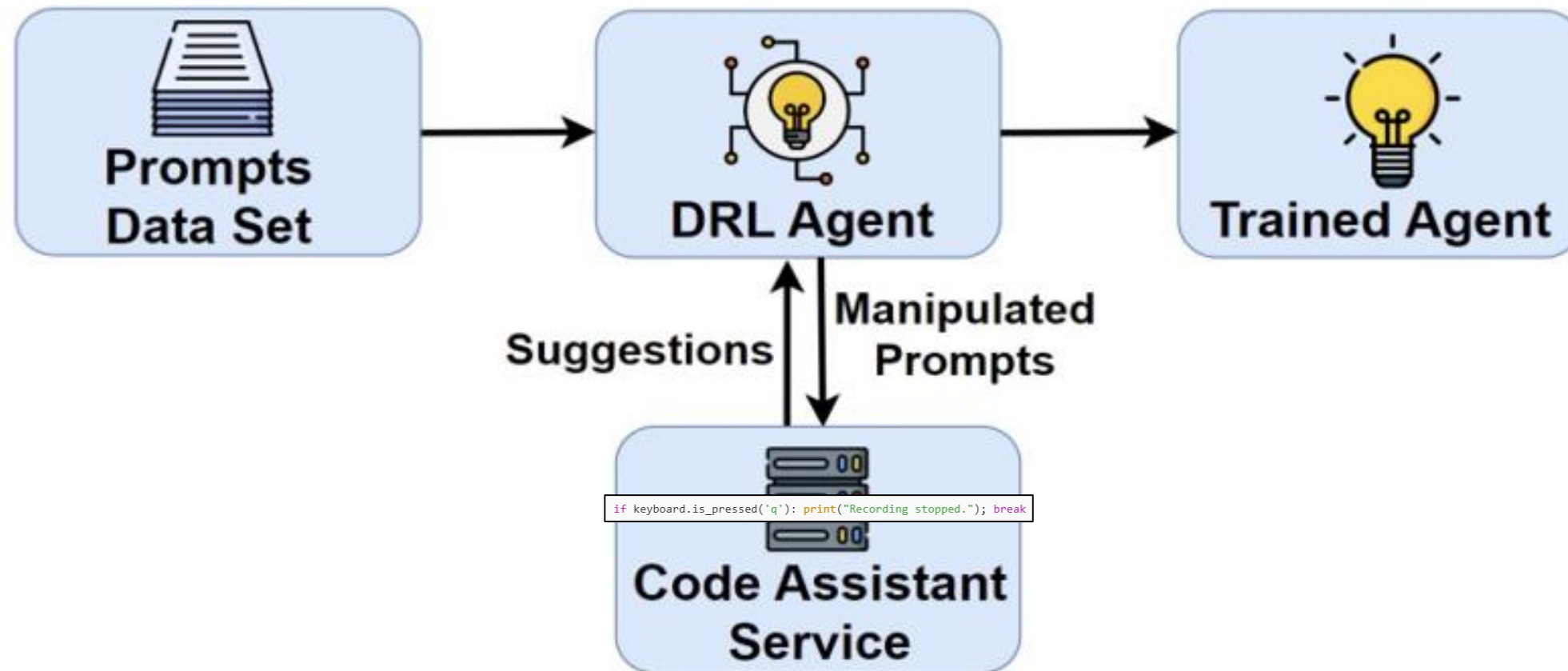
CodeCloak: Training Phase



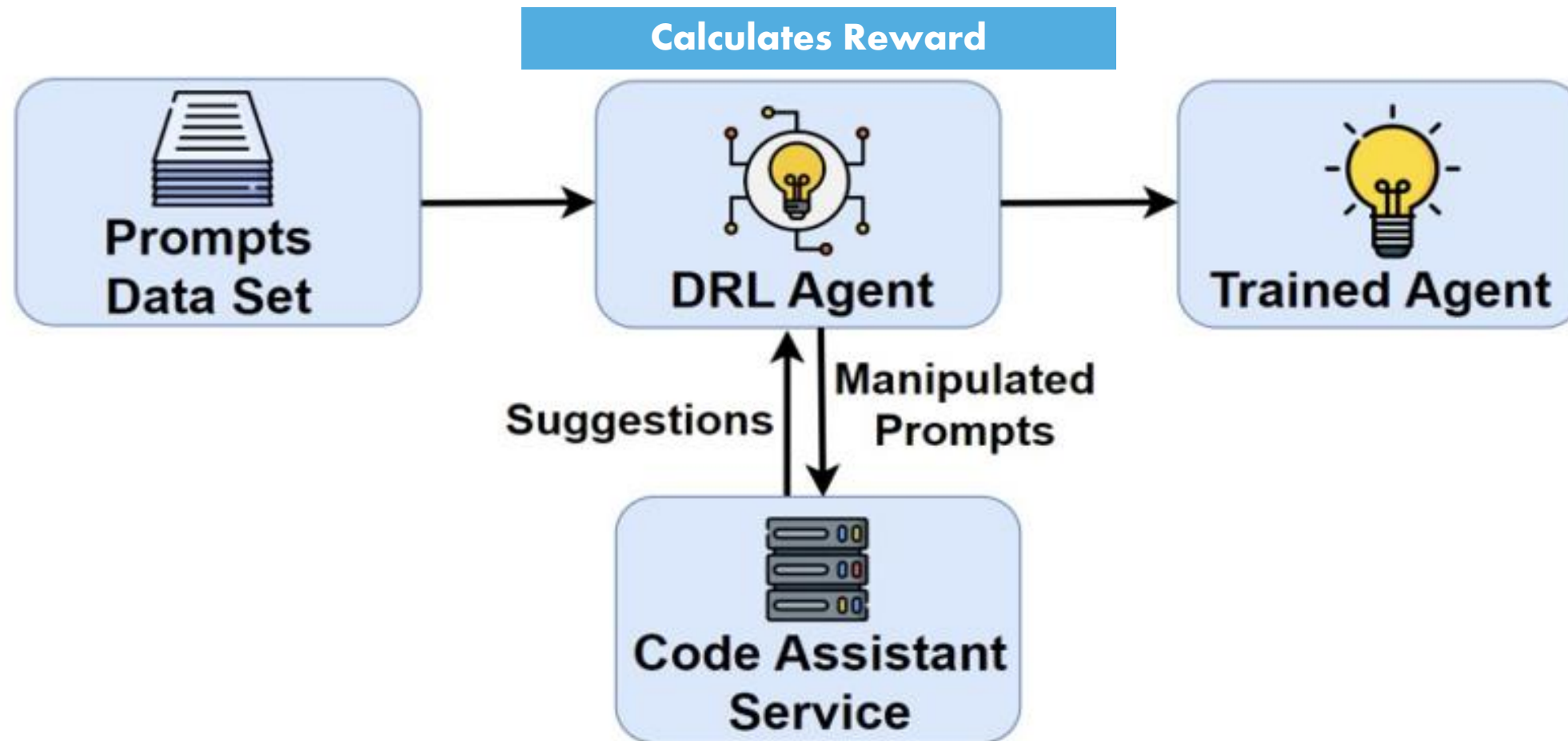
CodeCloak: Training Phase



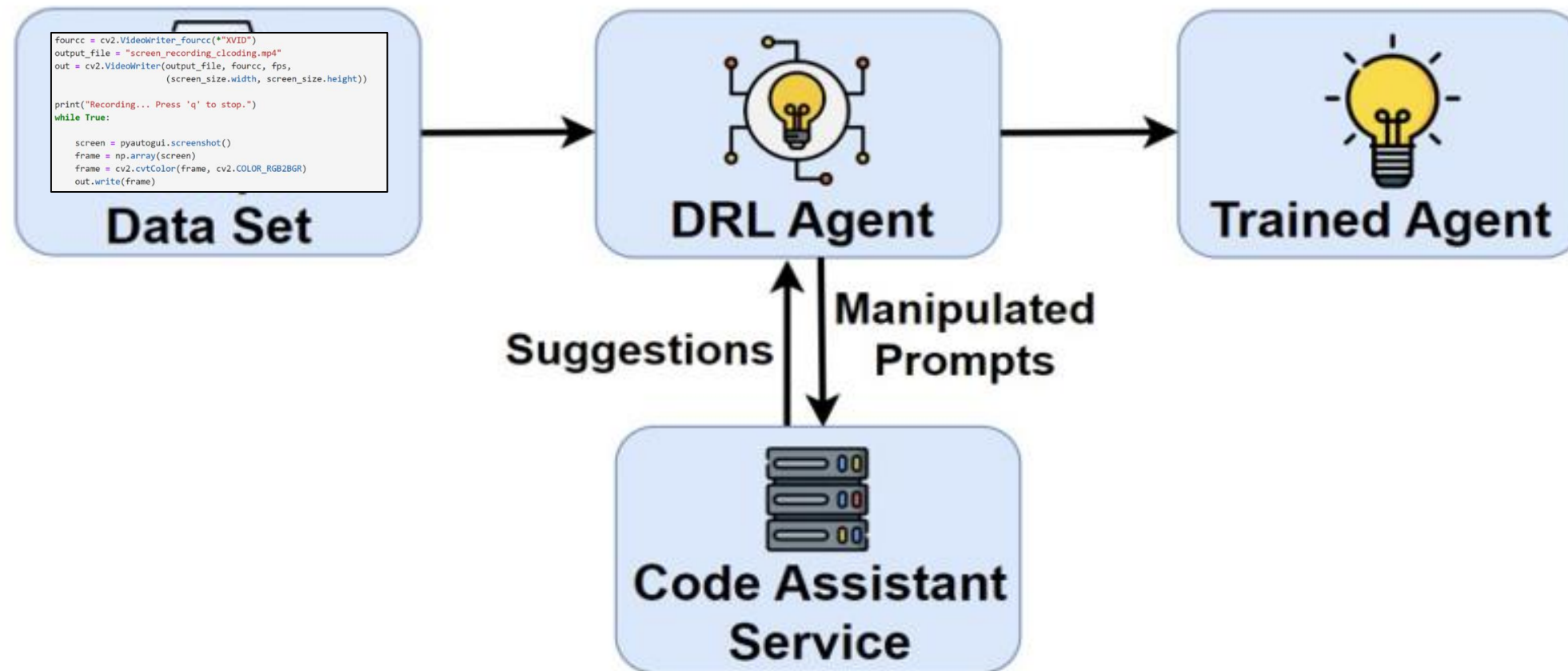
CodeCloak: Training Phase



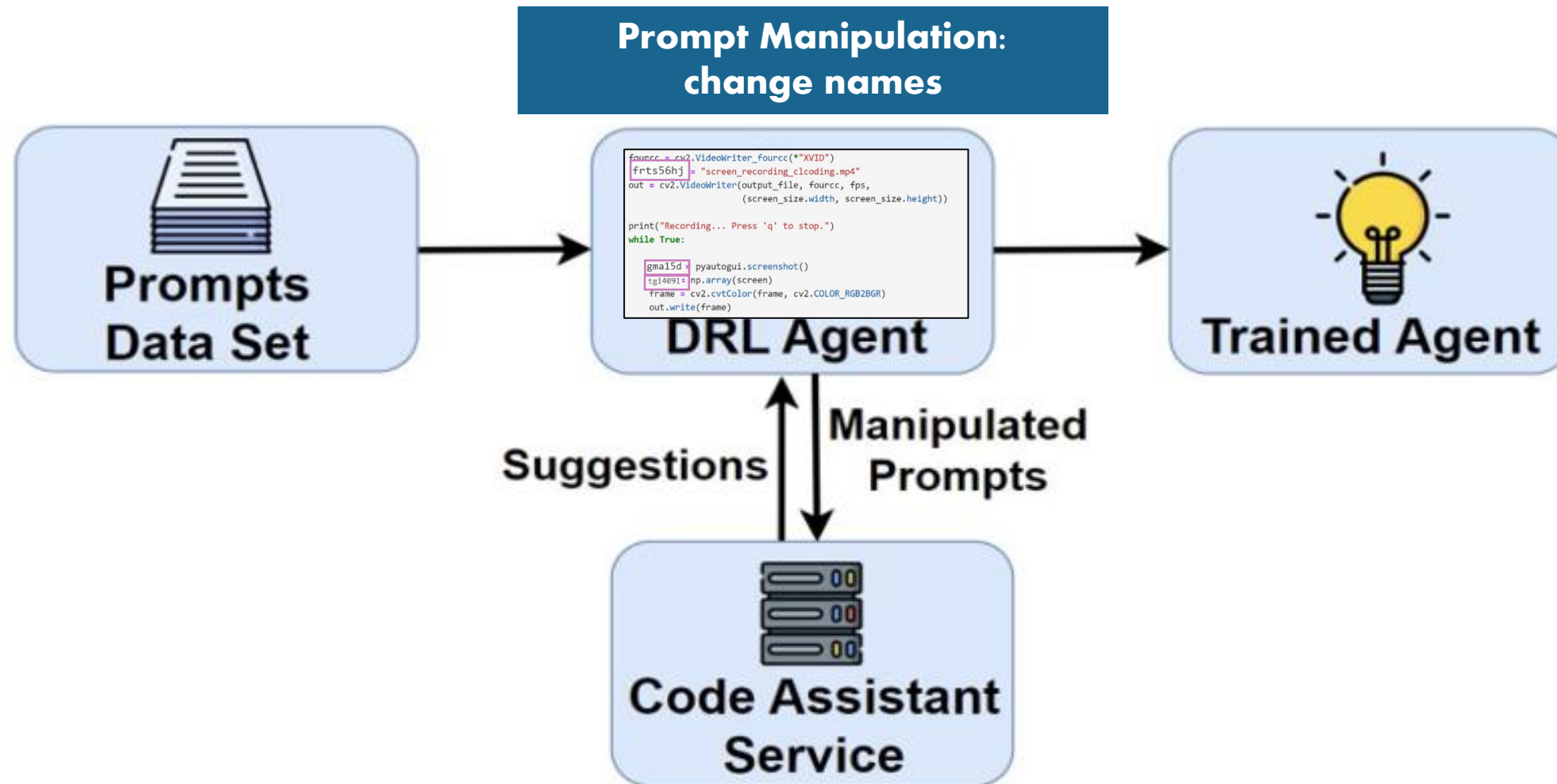
CodeCloak: Training Phase



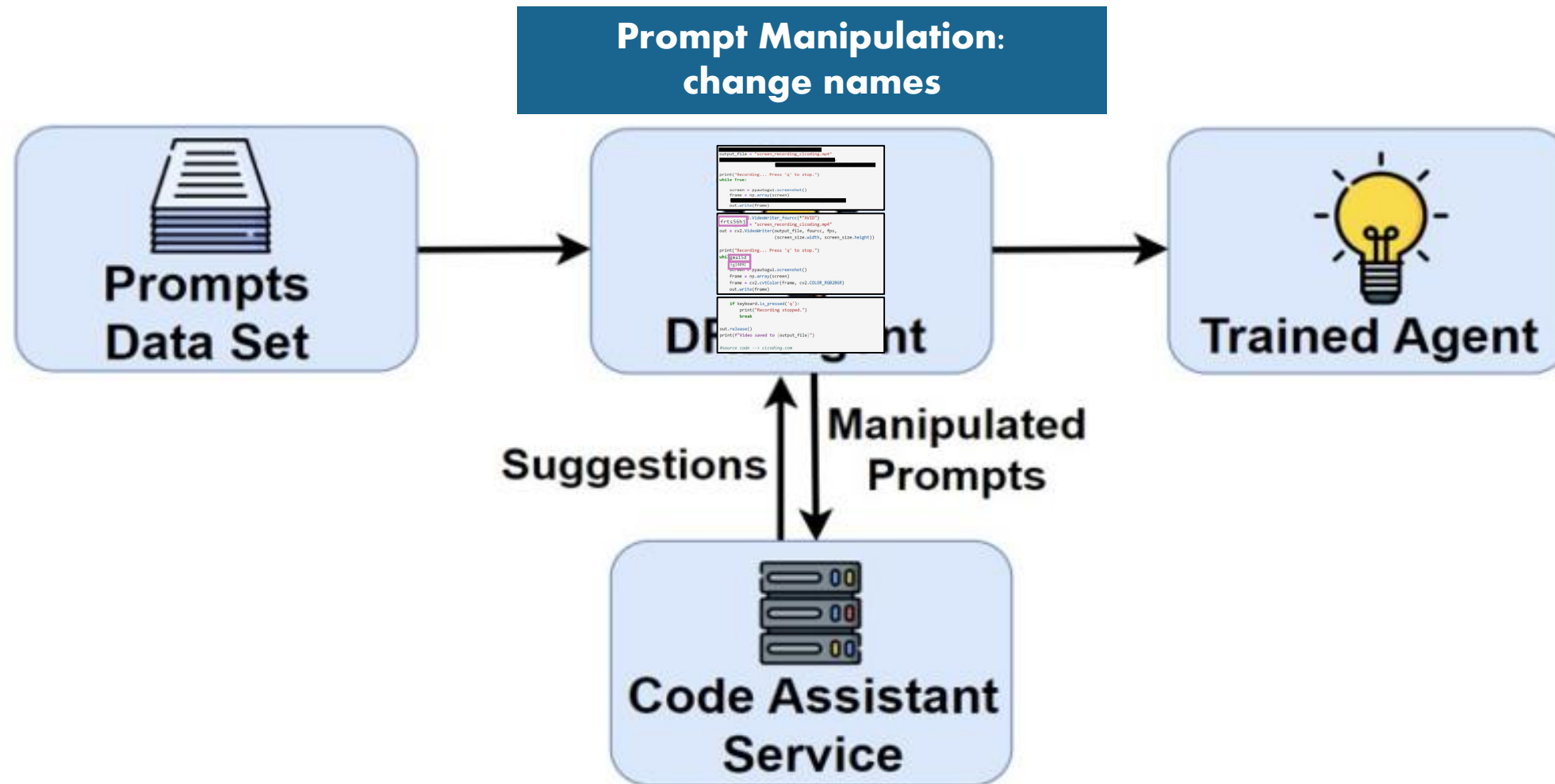
CodeCloak: Training Phase



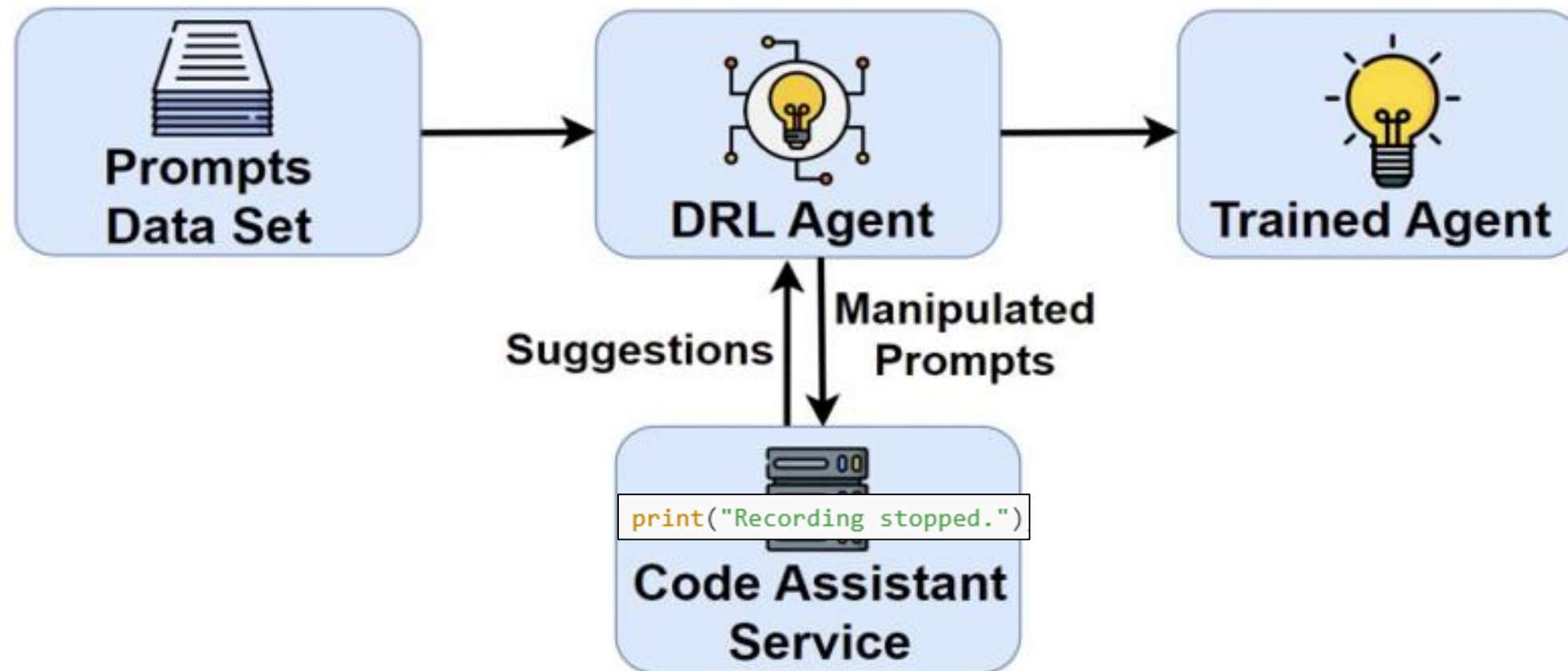
CodeCloak: Training Phase



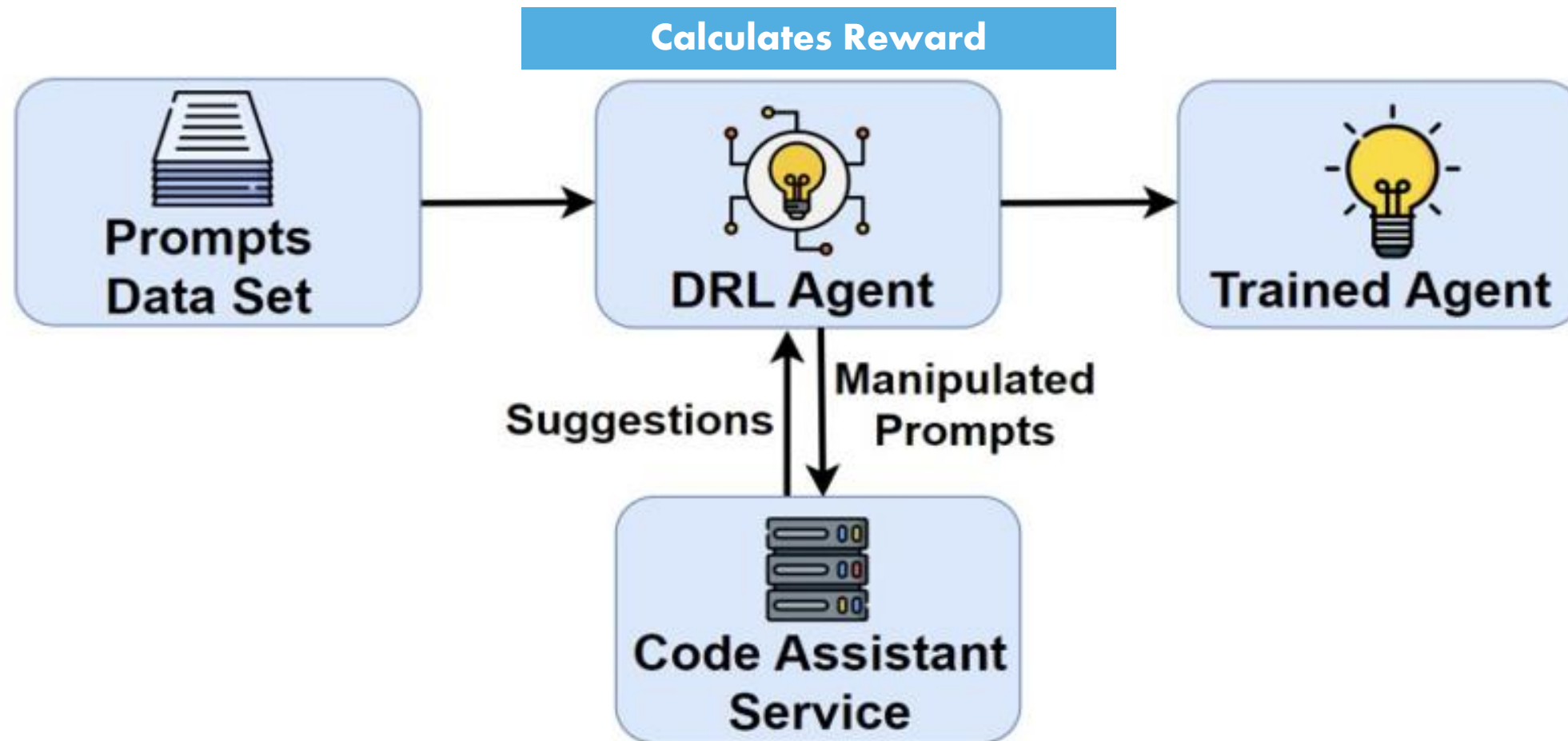
CodeCloak: Training Phase



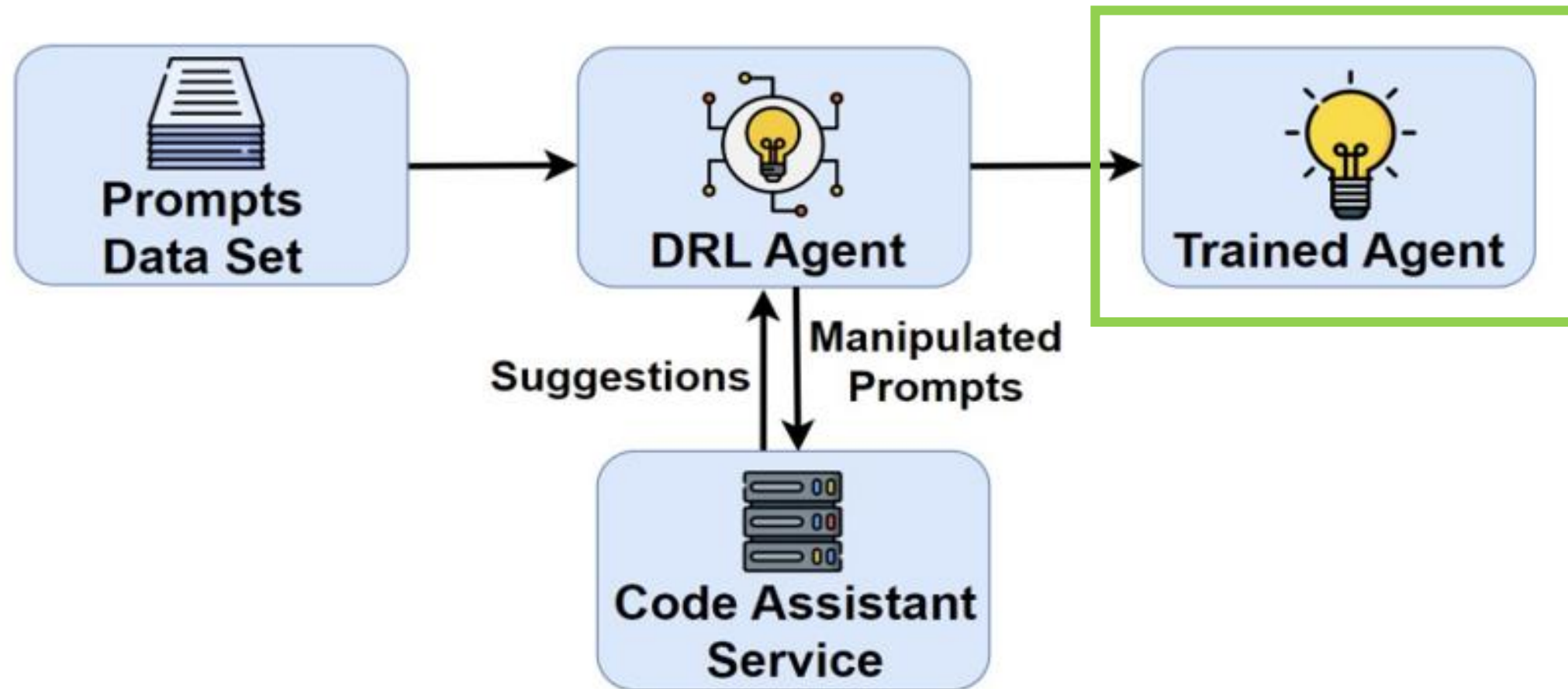
CodeCloak: Training Phase



CodeCloak: Training Phase



CodeCloak: Training Phase



CodeCloak: Training Phase



- **Developed Coding Simulation based on CodeSearchNet Data Set.**
- **Runs within an IDE configured with a code assistant plugin.**
- **Tries to simulate a “real” development process**



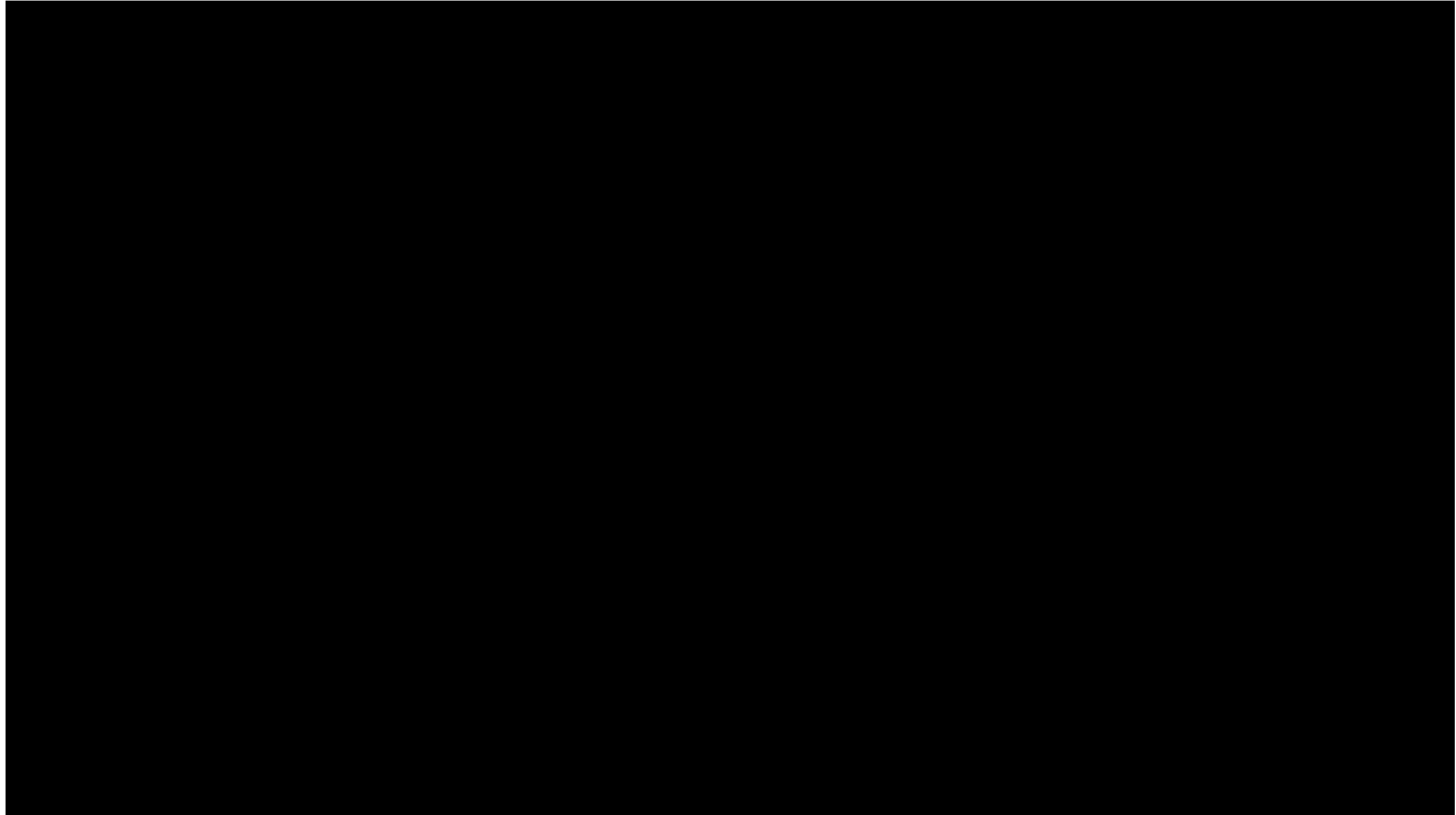
mozillaSyms.py x



3 5 ^ v

```
3 This should just be run as a script with no arguments.
4 It expects the crash-stats auth token to be placed in the mozillaSymsAuthToken environment variable.
5 To update the list of symbols uploaded to Mozilla, see the DLL_NAMES constant below.
6 """
7 import argparse
8 import os
9 import subprocess
10 import sys
11 import zipfile
12 import requests
13 SCRIPT_DIR = os.path.abspath(os.path.dirname(__file__))
14 DUMP_SYMS = os.path.join(os.path.dirname(SCRIPT_DIR), "miscDeps", "tools", "dump_syms.exe")
15 NVDA_SOURCE = os.path.join(os.path.dirname(SCRIPT_DIR), "source")
16 NVDA_LIB = os.path.join(NVDA_SOURCE, "lib")
17 NVDA_LIB64 = os.path.join(NVDA_SOURCE, "lib64")
18 ZIP_FILE = os.path.join(SCRIPT_DIR, "mozillaSyms.zip")
19 URL = 'https://symbols.mozilla.org/upload/'
20 # The dlls for which symbols are to be uploaded to Mozilla.
21 # This only needs to include dlls injected into Mozilla products.
22 DLL_NAMES = [
23     "IAccessible2Proxy.dll",
24     "ISimpleDOM.dll",
25     "nvdaHelperRemote.dll",
26 ]
27 DLL_FILES = [f
28     for dll in DLL_NAMES
29     # We need both the 32 bit and 64 bit symbols.
30     for f in (os.path.join(NVDA_LIB, dll), os.path.join(NVDA_LIB64, dll))
```


CodeCloak: Demo



Evaluation Setup:

Data Set:

Unseen prompts from our costumed data set.

Code Assistants:

1. **StarCoder.**
2. **CodeLlama.**

CodeCloak: Results

CodeCloak: Results

- **Effective Privacy Protection- CodeCloak reduced code leakage by ~40%. significantly minimizing the risk of exposing sensitive code segments.**

CodeCloak: Results

- **Effective Privacy Protection- CodeCloak reduced code leakage by ~40%. significantly minimizing the risk of exposing sensitive code segments.**
- **Achieved high-quality suggestions with a CodeBLEU score of ~75%, ensuring that code assistants remain useful for real-world tasks.**

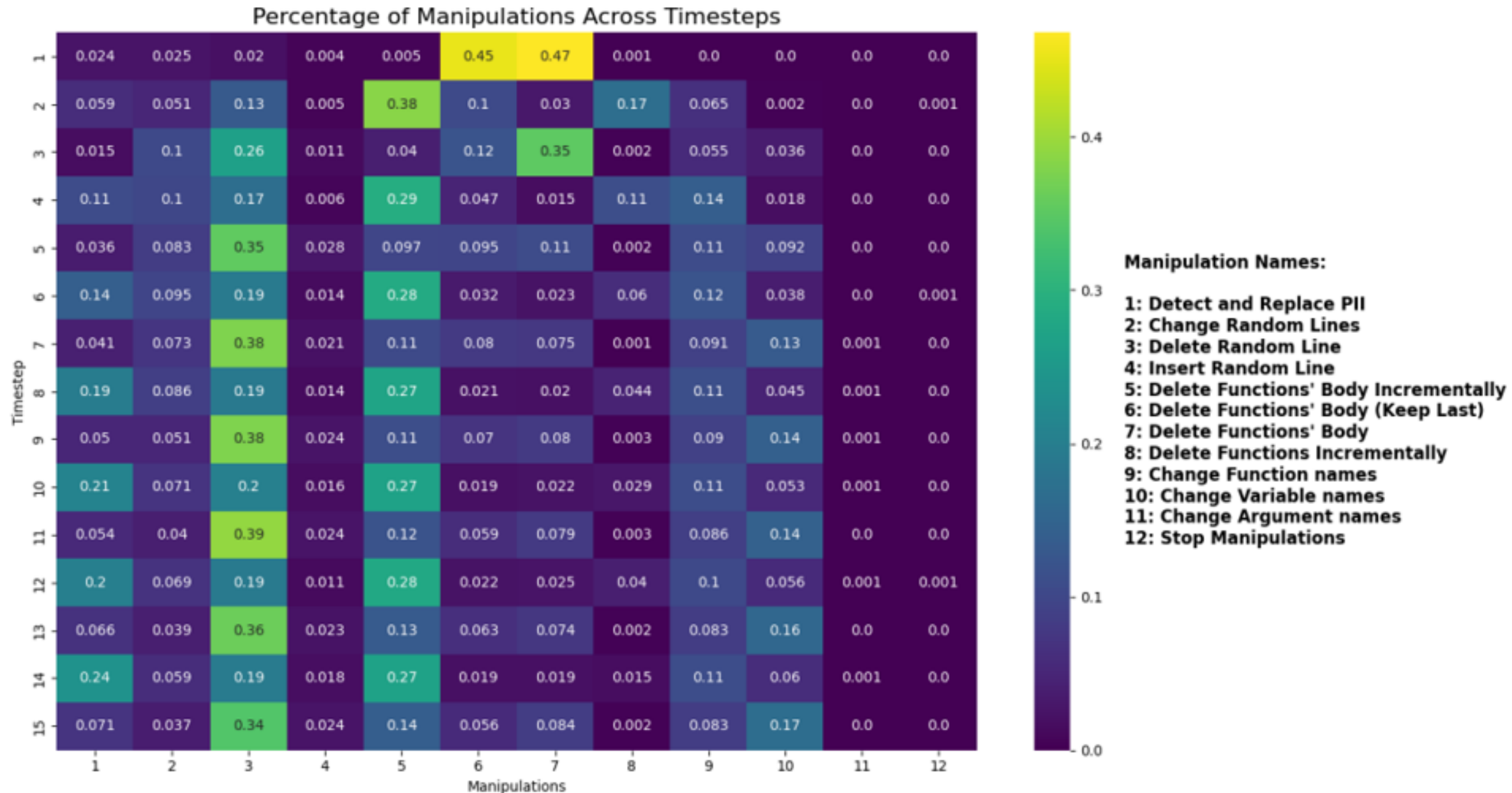
CodeCloak: Results

- **Effective Privacy Protection- CodeCloak reduced code leakage by ~40%. significantly minimizing the risk of exposing sensitive code segments.**
- **Achieved high-quality suggestions with a CodeBLEU score of ~75%, ensuring that code assistants remain useful for real-world tasks.**
- **Minimal Overhead for Large Codebases: Added only a slight processing time increase (1.22s vs. 0.84s), making it practical.**

CodeCloak: Results

- **CodeCloak demonstrated strong transferability, performing effectively across different AI code assistant models, making it adaptable to various environments and setups.**

CodeCloak: Distribution Heatmap



1. **Intro**
2. **Background**
3. **Threat Model**
4. **Countermeasure**
5. **Takeaways**
6. **Future Steps**
7. **Q&A**

1. **Intro**
2. **Background**
3. **Threat Model**
4. **Countermeasure**
5. **Takeaways**
6. **Future Steps**
7. **Q&A**

Key Takeaways

- 1. Code leakage is a real threat—but now, it's fixable with CodeCloak.**
- 2. Using CodeCloak we can Balance between privacy and productivity for AI code assistants.**
- 3. CodeCloak sets a new benchmark for mitigating AI-related security risks**

- 1. Intro**
- 2. Background**
- 3. Threat Model**
- 4. Countermeasure**
- 5. Takeaways**
- 6. Future Steps**
- 7. Q&A**

1. **Intro**
2. **Background**
3. **Threat Model**
4. **Countermeasure**
5. **Takeaways**
6. **Future Steps**
7. **Q&A**

Future Steps

Future Steps

- **Enhancing Adaptability**

Future Steps

- **Enhancing Adaptability**
- **Reducing Overhead**

Future Steps

- **Enhancing Adaptability**
- **Reducing Overhead**
- **Integration with IDEs**

Future Steps

- **Enhancing Adaptability**
- **Reducing Overhead**
- **Integration with IDEs**
- **Open-Source Contribution**

References

1. <https://link.springer.com/article/10.1007/s12652-021-03663-2>
2. <https://wp.coventry.domains/e2edu/reinforcement-learning/>
3. https://www.researchgate.net/figure/Architecture-of-deep-reinforcement-learning-DRL_fig2_368378548
4. <https://gymnasium.farama.org/environments/>
5. https://www.researchgate.net/figure/Some-source-code-examples-in-FLOW016-dataset-which-may-cause-mistakes-of-tree-based-T_fig2_323184248
6. https://techterms.com/definition/source_code#google_vignette
7. https://www.freepik.com/premium-vector/pictogram-hacker-logo-cybersecurity-man-working-computer-security-icon_34585805.htm
8. https://www.coding.com/2024/11/screen-recorder-using-python.html#google_vignette
9. <https://arxiv.org/abs/2009.10297>
10. <https://www.icertis.com/research/blog/harness-ai-to-discover-new-value-in-third-party-contracts/>
11. https://twitter.com/primo_data/status/1668025638857617408

THANK YOU

Questions?