
Reflexion: Language Agents with Verbal Reinforcement Learning

Anonymous Author(s)

Affiliation

Address

email

Abstract

1 Large language models (LLMs) have been increasingly used to interact with external
2 environments (e.g., games, compilers, APIs) as goal-driven agents. However,
3 it remains challenging for these language agents to quickly and efficiently learn
4 from trial-and-error as traditional reinforcement learning methods require extensive
5 training samples and expensive model fine-tuning. We propose *Reflexion*, a
6 novel framework to reinforce language agents not by updating weights, but instead
7 through linguistic feedback. Concretely, Reflexion agents verbally reflect
8 on task feedback signals, then maintain their own reflective text in an episodic
9 memory buffer to induce better decision-making in subsequent trials. Reflexion is
10 flexible enough to incorporate various types (scalar values or free-form language)
11 and sources (external or internally simulated) of feedback signals, and obtains
12 significant improvements over a baseline agent across diverse tasks (sequential
13 decision-making, coding, language reasoning). For example, Reflexion achieves
14 a 91% pass@1 accuracy on the HumanEval coding benchmark, surpassing the
15 previous state-of-the-art GPT-4 that achieves 80%. We also conduct ablation and
16 analysis studies using different feedback signals, feedback incorporation methods,
17 and agent types, and provide insights into how they affect performance.

18 1 Introduction

19 Recent works such as ReAct [30], SayCan [1], Toolformer [20], HuggingGPT [21], generative
20 agents [17], and WebGPT [15] have demonstrated the feasibility of autonomous decision-making
21 agents that are built on top of a large language model (LLM) core. These methods use LLMs to
22 generate text and ‘actions’ that can be used in API calls and executed in an environment. Since
23 they rely on massive models with an enormous number of parameters, such approaches have been
24 so far limited to using in-context examples as a way of teaching the agents, since more traditional
25 optimization schemes like reinforcement learning with gradient descent require substantial amounts
26 of compute and time.

27 In this paper, we propose an alternative approach called *Reflexion* that uses verbal reinforcement
28 to help agents learn from prior failings. Reflexion converts binary or scalar feedback from the
29 environment into verbal feedback in the form of a textual summary, which is then added as additional
30 context for the LLM agent in the next episode. This self-reflective feedback acts as a ‘semantic’
31 gradient signal by providing the agent with a concrete direction to improve upon, helping it learn
32 from prior mistakes to perform better on the task. This is akin to how humans iteratively learn to
33 accomplish complex tasks in a few-shot manner – by reflecting on their previous failures in order to
34 form an improved plan of attack for the next attempt. For example, in figure 1, a Reflexion agent
35 learns to optimize its own behavior to solve decision-making, programming, and reasoning tasks
36 through trial, error, and self-reflection.

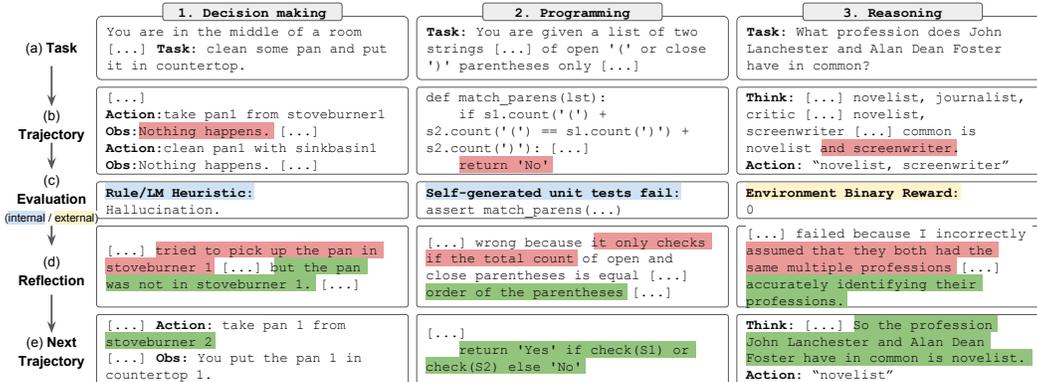


Figure 1: Reflexion works on decision-making 4.1, programming 4.3, and reasoning 4.2 tasks.

37 Generating useful reflective feedback is challenging since it requires a good understanding of where
38 the model made mistakes (i.e. the credit assignment problem [23]) as well as the ability to generate
39 a summary containing actionable insights for improvement. We explore three ways for doing
40 this – simple binary environment feedback, pre-defined heuristics for common failure cases, and
41 self-evaluation such as binary classification using LLMs (decision-making) or self-written unit
42 tests (programming). In all implementations, the evaluation signal is amplified to natural language
43 experience summaries which can be stored in long-term memory.

44 Reflexion has several advantages compared to more traditional RL approaches like policy or value-
45 based learning: 1) it is lightweight and doesn't require finetuning the LLM, 2) it allows for more
46 nuanced forms of feedback (e.g. targeted changes in actions), compared to scalar or vector rewards
47 that are challenging to perform accurate credit assignment with, 3) it allows for a more explicit and
48 interpretable form of episodic memory over prior experiences, and 4) it provides more explicit hints
49 for actions in future episodes. At the same time, it does have the disadvantages of relying on the
50 power of the LLM's self-evaluation capabilities (or heuristics) and not having a formal guarantee for
51 success. However, as LLM capabilities improve, we only expect this paradigm to get better over time.

52 We perform experiments on (1) decision-making tasks to test sequential action choices over long
53 trajectories, (2) reasoning tasks to test knowledge-intensive, single-step generation improvement,
54 and (3) programming tasks to teach the agent to effectively use external tools such as compilers
55 and interpreters. Across all three types of tasks, we observe Reflexion agents are better decision-
56 makers, reasoners, and programmers. More concretely, Reflexion agents improve on decision-making
57 AlfWorld [22] tasks over strong baseline approaches by an absolute 22% in 12 iterative learning
58 steps, and on reasoning questions in HotPotQA [28] by 20%, and Python programming tasks on
59 HumanEval [5] by as much as 11%.

60 To summarize, our contributions are the following:

- 61 • We propose Reflexion, a new paradigm for ‘verbal’ reinforcement that parameterizes a
62 policy as an agent’s memory encoding paired with a choice of LLM parameters.
- 63 • We explore this emergent property of *self-reflection* in LLMs and empirically show that
64 self-reflection is extremely useful to learn complex tasks over a handful of trials.
- 65 • We introduce LeetcodeHardGym, a code-generation RL gym environment consisting of 40
66 challenging Leetcode questions (‘hard-level’) in 19 programming languages.
- 67 • We show that Reflexion achieves improvements over strong baselines across several tasks,
68 and achieves state-of-the-art results on various code generation benchmarks.

69 2 Related work

70 **Reasoning and decision-making** Self-Refine [12] employs an iterative framework for self-
71 refinement to autonomously improve generation through self-evaluation. These self-evaluation

Related work on reasoning and decision-making					
Approach	Self refine	Hidden constraints	Decision making	Binary reward	Memory
Self-refine [12]	✓	✗	✗	✗	✗
Beam search [27]	✓	✓	✓	✓	✗
Reflexion (ours)	✓	✓	✓	✓	✓

Related work on programming					
Approach	Test execution	Debugging	Self-generated tests	Multiple languages	Self-reflection
AlphaCode [11]	✓	✗	✗	✓	✗
CodeT [4]	✓	✗	✓	✗	✗
Self-debugging [6]	✓	✓	✗	✗	✗
CodeRL [10]	✓	✓	✗	✗	✗
Reflexion (ours)	✓	✓	✓	✓	✓

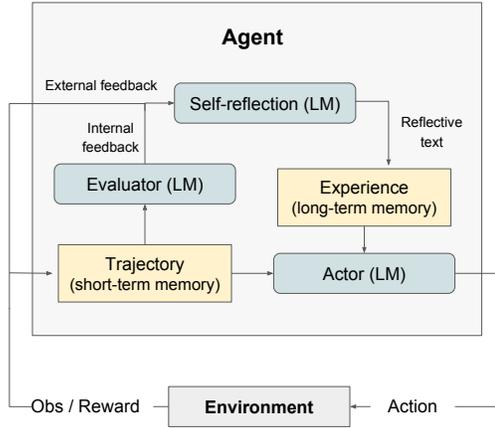
72 and self-improvement steps are conditioned on given task constraints, such as "How can this genera-
73 tion be written in a more positive way". Self-Refine is effective but is limited to single-generation
74 reasoning tasks. Pryzant et al. [19] performs a similar semantic prompt-writing optimization, but is
75 also limited to single-generation tasks. Paul et al. [18] fine-tune critic models to provide intermediate
76 feedback within trajectories to improve reasoning responses. Xie et al. [27] use stochastic beam
77 search over actions to perform a more efficient decision-making search strategy which allows the
78 agent to use foresight advantage due to its self-evaluation component. Yoran et al. [31] and Nair et al.
79 [14] use decider models to reason over several generations. In this paper, we show that several of
80 these concepts can be enhanced with *self-reflection* to build a persisting memory of self-reflective
81 experiences which allows an agent to identify its own errors and self-suggest lessons to learn from its
82 mistakes over time.

83 **Programming** Several past and recent works employ variations of test-driven development or
84 code debugging practices. AlphaCode [11] evaluates a set of generations on hidden test cases.
85 CodeT [4] uses self-generated unit tests that are used to score generated function implementations.
86 Self-Debugging [6] employs a debugging component that is used to improve existing implementations
87 given feedback from a code execution environment. CodeRL [10] sets the problem in an RL frame-
88 work using an actor-critic setup to debug programs given feedback from an execution environment.
89 AlphaCode, Self-Debugging and CodeRL are effective in fixing less-complex program bugs, but they
90 rely upon ground truth test cases that invalidate pass@1 eligibility, and do not use self-reflection to
91 bridge the gap between error identification and implementation improvement. CodeT does not access
92 hidden test cases but does not implement a self-learning step to improve code writing.

93 **Self-reflection** Several recent works recognize new cognitive abilities exhibited by LLMs that
94 were previously considered unique to humans. Kosinski [8] show notable performance on theory
95 of mind (ToM) tasks at the level of young to middle-aged human children. However, Ullman [24]
96 argue that ToM task completion accuracy is highly dependent on specific setup and will fail to
97 complete trivial variations to the original problem. Moghaddam and Honey [13] show that ToM task
98 accuracy can be improved using better prompting strategies. Wei et al. [25] introduce the concept of
99 emergent properties in LLMs and discuss scaling predictions for future models. To the best of our
100 knowledge, our work is the first to utilize *self-reflection* for practical use in autonomous behavior in
101 language agents for reasoning, decision-making, and programming tasks. Empirically, we show that
102 self-reflection steps are crucial to amplifying sparse feedback for iterative learning.

103 3 Reflexion: reinforcement via verbal reflection

104 We develop a modular formulation for Reflexion, utilizing three distinct models: an *Actor*, denoted as
105 M_a , which generates text and actions; an *Evaluator* model, represented by M_e , that scores the outputs
106 produced by M_a ; and a *Self-Reflection* model, denoted as M_{sr} , which generates verbal reinforcement
107 cues to assist the Actor in self-improvement. We provide a detailed description of each of these
108 models and subsequently elucidate their collaborative functioning within the Reflexion framework.



Algorithm 1 Reinforcement via self-reflection

```

Initialize Actor, Evaluator, Self-Reflection:
 $M_a, M_e, M_{sr}$ 
Initialize policy  $\pi_\theta(a_i|s_i), \theta = \{M_a, mem\}$ 
Generate initial trajectory using  $\pi_\theta$ 
Evaluate  $\tau_0$  using  $M_e$ 
Generate initial self-reflection  $sr_0$  using  $M_{sr}$ 
Set  $mem \leftarrow [sr_0]$ 
Set  $t = 0$ 
while  $M_e$  not pass or  $t < \text{max trials}$  do
  Generate  $\tau_t = [a_0, o_0, \dots, a_t, o_t]$  using  $\pi_\theta$ 
  Evaluate  $\tau_t$  using  $M_e$ 
  Generate self-reflection  $sr_t$  using  $M_{sr}$ 
  Append  $sr_t$  to  $mem$ 
  Increment  $t$ 
end while
return

```

Figure 2: (a) Diagram of Reflexion. (b) Reflexion reinforcement algorithm

109 **Actor** The Actor is built upon a large language model (LLM) that is specifically prompted to
 110 generate the necessary text and actions conditioned on the state observations. Analogous to traditional
 111 policy-based RL setups, we sample an action or generation, a_t , from the current policy π_θ at time
 112 t , receive an observation from the environment o_t . We explore various Actor models, including
 113 Chain of Thought [26] and ReAct [30]. These diverse generation models allow us to explore different
 114 aspects of text and action generation within the Reflexion framework, providing valuable insights
 115 into their performance and effectiveness. In addition, we also add a memory component mem that
 116 provides additional context to this agent (details on how this is populated are provided below).

117 **Evaluator** The Evaluator component of the Reflexion framework plays a crucial role in assessing
 118 the quality of the generated outputs produced by the Actor. It takes as input a generated trajectory
 119 and computes a reward score that reflects its performance within the given task context. Defining
 120 effective value and reward functions that apply to semantic spaces is difficult, so we investigate
 121 several variants of the Evaluator model. For reasoning tasks, we explore reward functions based
 122 on exact match (EM) grading, ensuring that the generated output aligns closely with the expected
 123 solution. In decision-making tasks, we employ pre-defined heuristic functions that are tailored to
 124 specific evaluation criteria. Additionally, we experiment with using a different instantiation of an
 125 LLM itself as an Evaluator, generating rewards for decision-making and programming tasks. This
 126 multi-faceted approach to Evaluator design allows us to examine different strategies for scoring
 127 generated outputs, offering insights into their effectiveness and suitability across a range of tasks.

128 **Self-reflection** The Self-Reflection model instantiated as an LLM, plays a crucial role in the
 129 Reflexion framework by generating verbal self-reflections to provide valuable feedback for future
 130 trials. Given a sparse reward signal, such as a binary success status (success/fail), the current trajectory,
 131 and its persistent memory mem , the self-reflection model generates nuanced and specific feedback.
 132 This feedback, which is more informative than scalar rewards, is then stored in the agent’s memory
 133 (mem). For instance, in a multi-step decision-making task, when the agent receives a failure signal, it
 134 can infer that a specific action a_i led to subsequent incorrect actions a_{i+1} and a_{i+2} . The agent can
 135 then verbally state that it should have taken a different action, a'_i , which would have resulted in a'_{i+1}
 136 and a'_{i+2} , and store this experience in its memory. In subsequent trials, the agent can leverage its past
 137 experiences to adapt its decision-making approach at time t by choosing action a'_i . This iterative
 138 process of trial, error, self-reflection, and persisting memory enables the agent to rapidly improve its
 139 decision-making ability in various environments by utilizing informative feedback signals.

140 **Memory** Core components of the Reflexion process are the notion of short-term and long-term
 141 memory. At inference time, the Actor conditions its decisions on short and long-term memory, similar
 142 to the way that humans remember fine-grain recent details while also recalling distilled important

143 experiences from long-term memory. In the RL setup, the trajectory history serves as the short-term
144 memory while outputs from the Self-Reflection model are stored in long-term memory. These two
145 memory components work together to provide context that is specific but also influenced by lessons
146 learned over several trials, which is a key advantage of Reflexion agents over other LLM action
147 choice works.

148 **The Reflexion process** Reflexion is formalized as an iterative optimization process in 1. In the
149 first trial, the Actor produces a trajectory τ_0 by interacting with the environment. The Evaluator then
150 produces a score r_0 which is computed as $r_t = M_e(\tau_0)$. r_t is only a scalar reward for trial t that
151 improves as task-specific performance increases. After the first trial, to amplify r_0 to a feedback form
152 that can be used for improvement by an LLM, the Self-Reflection model analyzes the set of $\{\tau_0, r_0\}$
153 to produce a summary sr_0 which is stored in the memory mem . sr_t is a verbal experience feedback
154 for trial t . The Actor, Evaluator, and Self-Reflection models work together through trials in a loop
155 until the Evaluator deems τ_t to be correct. As mentioned in 3, the memory component of Reflexion
156 is crucial to its effectiveness. After each trial t , sr_t , is appended mem . In practice, we bound mem
157 by a maximum number of stored experiences, Ω (usually set to 1-3) to adhere to max context LLM
158 limitations.

159 4 Experiments

160 We evaluate various natural language RL setups on decision-making, reasoning, and code generation
161 tasks. Specifically, we challenge an agent to perform search-based question answering on HotPotQA
162 [28], multi-step tasks in common household environments in AlfWorld [22], and code writing tasks
163 in competition-like environments with interpreters and compilers in HumanEval [5], MBPP [2],
164 and LeetcodeHard, a new benchmark. Most notably, Reflexion improves performance over strong
165 baselines by 22% in AlfWorld, 20% in HotPotQA, and 11% on HumanEval.

166 4.1 Sequential decision making: ALFWorld

167 AlfWorld is a suite of text-based environments that challenge an agent to solve multi-step tasks
168 in a variety of interactive environments based on TextWorld [7]. Following Yao et al. [30], we
169 run the agent in 134 AlfWorld environments across six different tasks, including finding hidden
170 objects (e.g., finding a spatula in a drawer), moving objects (e.g., moving a knife to the cutting
171 board), and manipulating objects with other objects (e.g., chilling a tomato in the fridge). We use
172 ReAct [30] as the action generator as Yao et al. [30] has shown success in long trajectory decision-
173 making using explicit intermediate thoughts. AlfWorld tasks naturally require a self-evaluation step
174 as the environment can only signal if a task is complete. To achieve fully autonomous behavior,
175 we implement two self-evaluation techniques: natural language classification using an LLM and a
176 hand-written heuristic. The heuristic is simple: if the agent executes the same action and receives the
177 same response for more than 3 cycles, or if the number of actions taken in the current environment
178 exceeds 30 (inefficient planning), we self-reflect. In the baseline runs, if self-reflection is suggested,
179 we skip the self-reflection process, reset the environment, and start a new trial. In the Reflexion runs,
180 the agent uses self-reflection to find its mistake, update its memory, reset the environment, and start a
181 new trial. To avoid very long prompt windows that may exceed the maximum limit, we truncate the
182 agent’s memory to the last 3 self-reflections (experiences).

183 To avoid syntactic errors, we provide two domain-specific few-shot trajectories to the agent. We use
184 the same few-shot trajectory examples as Yao et al. [30] with GPT-3 for the LLM. AlfWorld tasks,
185 ReAct few-shot prompts, and Reflexion examples are included in the appendix.

186 **Results** ReAct + Reflexion significantly outperforms ReAct by completing 130 out of 134 tasks
187 using the simple heuristic to detect hallucinations and inefficient planning. Further, ReAct + Reflexion
188 learns to solve additional tasks by learning in 12 consecutive trials. In the ReAct-only approach, we
189 see that performance increase halts between trials 6 and 7.

190 **Analysis** A common error in baseline failed AlfWorld trajectories is when an agent thinks that it
191 has possession of an item but does not actually have the item. The agent proceeds to execute several
192 actions in a long trajectory and is not able to backtrack its actions to find the mistake. Reflexion
193 eliminates almost all of these cases by using self-reflection to distill long, failed trajectories into

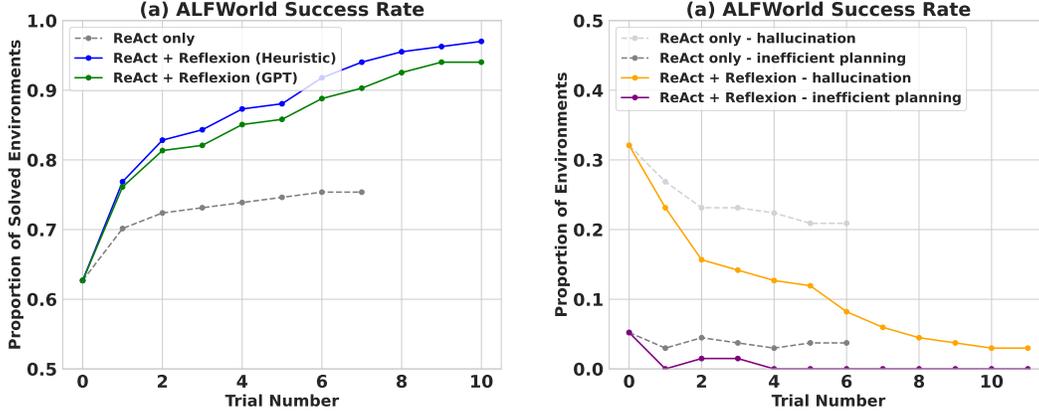


Figure 3: (a) ALFWorld performance across 134 tasks showing cumulative proportions of solved tasks using self-evaluation techniques of (Heuristic) and (GPT) for binary classification. (b) Classification of ALFWorld trajectories by reason of failure.

194 relevant experiences that can be used as "self-hints" in the future. There are two main cases in which
 195 long-term memory helps an agent in ALFWorld: 1) An early mistake in a long trajectory can be easily
 196 identified. The agent can suggest a new action choice or even a new long-term plan. 2) There are too
 197 many surfaces/containers to check for an item. The agent can exploit its experience memory over
 198 several trials to thoroughly search a room. In 3, the learning curve suggests that the learning process
 199 occurs over several experiences, meaning that the agent is successfully balancing cases 1 and 2 shown
 200 in the immediate spike in the improvement between the first two trials, then a steady increase over
 201 the next 11 trials to a near-perfect performance. On the other hand, 3 shows a ReAct-only agent
 202 converging at a hallucination rate of 22% with no signs of long-term recovery.

203 4.2 Reasoning: HotpotQA

204 HotPotQA [28] is a Wikipedia-based dataset with 113k question-and-answer pairs that challenge
 205 agents to parse content and reason over several supporting documents. To test improvement in
 206 reasoning *only* ability, we implement Reflexion + Chain-of-Thought (CoT) [26] for step-by-step
 207 $Q \rightarrow A$ and $Q, C_{gt} \rightarrow A$ implementations, where Q is the question, C_{gt} is the ground truth context
 208 from the dataset, and A is the final answer. Since CoT is not a multi-step decision-making technique,
 209 we give C_{gt} to the agent so that we can isolate the reasoning behavior over large sections of the
 210 provided text. To test holistic question and answering ability, which requires reasoning and action
 211 choice, we implement a Reflexion + ReAct [30] agent that can retrieve relevant context using a
 212 Wikipedia API and infer answers using step-by-step explicit thinking. For CoT implementations, we
 213 use 6-shot prompting; for ReAct, we use 2-shot prompting, and for self-reflection, we use 2-shot
 214 prompting. All examples can be found in the appendix.

215 Robustly evaluating natural language answers is a long-standing problem in NLP. Therefore, between
 216 trials, we use exact match answer grading using the environment to give a binary success signal to
 217 the agent. After each trial, the self-reflection loop is employed to amplify the binary signal, similar to
 218 the decision-making setup 4.1 in ALFWorld with a memory size of 3 experiences.

219 **Results** Reflexion outperforms all baseline approaches by significant margins over several learning
 220 steps. Furthermore, ReAct-only, CoT-only, and CoT (GT)-only implementations fail to probabilistically
 221 improve on any tasks, meaning that no failed tasks from the first trial from any of the baseline
 222 approaches were able to be solved in subsequent trials using a temperature of 0.7 In the Reflexion runs,
 223 we allowed the agent to gather experience and retry on failed tasks until it produced 3 consecutive
 224 failed attempts on the particular task. Naturally, the CoT (GT) achieved higher accuracy scores as it
 225 was given access to the ground truth context of the question. Still, the CoT (GT) agent is unable to
 226 correctly infer the correct answer for 39% of the questions, but Reflexion helps the agent to correct
 227 its mistakes without access to the ground truth answer to improve its accuracy by 14%.

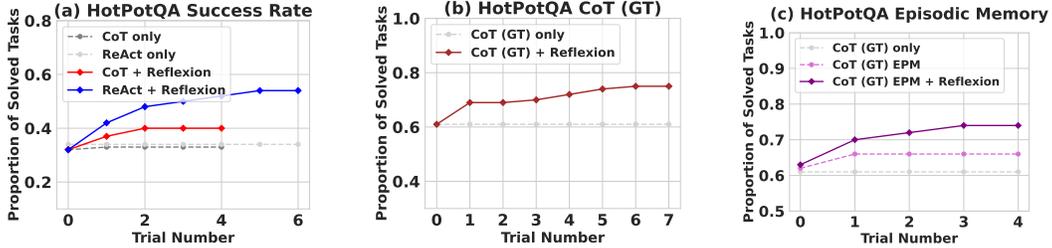


Figure 4: Chain-of-Thought (CoT) and ReAct. Reflexion improves search, information retrieval, and reasoning capabilities on 100 HotPotQA questions. (a) Reflexion ReAct vs Reflexion CoT (b) Reflexion CoT (GT) for reasoning only (c) Reflexion vs episodic memory ablation.

228 **Analysis** We perform an ablation experiment to isolate the advantage of the self-reflective step for
 229 reasoning using CoT (GT) as the baseline approach 4. Recall that CoT (GT) uses Chain-of-Thought
 230 reasoning with provided ground truth context, which tests reasoning ability over long contexts. Next,
 231 we add an element of episodic memory (EPM) by including the most recent trajectory. For the
 232 Reflexion agent, we implement the standard self-reflection step as a final pass. Intuitively, we test if
 233 the agent is iteratively learning more effectively by using verbal explanation using language written
 234 in the first person. 4 shows that self-reflection improves learning by an 8% absolute boost over
 235 the episodic memory learning advantage. This result supports the argument that refinement-only
 236 approaches are not as effective as self-reflection-guided refinement approaches.

237 4.3 Programming

238 We evaluate the baseline and Reflexion approaches on Python and Rust code writing on MBPP
 239 [2], HumanEval [5], and LeetcodeHardGym, our new dataset. MBPP and HumanEval measure
 240 function body generation accuracy given natural language descriptions. We use a benchmark language
 241 compiler, MultiPL-E [3], to translate subsets of HumanEval and MBPP to the Rust language. MultiPL-
 242 E is a collection of small compilers that can be used to translate Python benchmark questions to 18
 243 other languages. We include experiments for Rust code generation to demonstrate that Reflexion
 244 implementations for code generation are language-agnostic and can be used for interpreted and
 245 compiled languages. Lastly, we introduce a new benchmark, LeetcodeHardGym, which is an
 246 interactive programming gym that contains 40 Leetcode hard-rated questions that have been released
 247 after October 8, 2022, which is the pre-training cutoff date of GPT-4 [16].

248 The task of programming presents a unique opportunity to use more grounded self-evaluation practices
 249 such as self-generated unit test suites. Thus, our Reflexion-based programming task implementation is
 250 eligible for pass@1 accuracy reporting. To generate a test suite, we use Chain-of-Thought prompting
 251 [26] to produce diverse, extensive tests with corresponding natural language descriptions. Then, we
 252 filter for syntactically valid test statements by attempting to construct a valid abstract syntax tree
 253 (AST) for each proposed test. Finally, we sample n tests from the collection of generated unit tests
 254 to produce a test suite T , denoted as $\{t_0, t_1, \dots, t_n\}$. We set n to a maximum of 6 unit tests. Aside
 255 from the unit test suite component, the setup for the learning loop for a Reflexion programming agent
 256 is identical to the reasoning and decision-making agents with a max memory limit of 1 experience.

Benchmark + Language	Prev SOTA Pass@1	SOTA Pass@1	Reflexion Pass@1
HumanEval (PY)	65.8 (CodeT [4] + GPT-3.5)	80.1 (GPT-4)	91.0
HumanEval (RS)	–	60.0 (GPT-4)	68.0
MBPP (PY)	67.7 (CodeT [4] + Codex [5])	80.1 (GPT-4)	77.1
MBPP (RS)	–	70.9 (GPT-4)	75.4
Leetcode Hard (PY)	–	7.5 (GPT-4)	15.0

Table 1: Pass@1 accuracy for various model-strategy-language combinations. The base strategy is a single code generation sample. All instruction-based models follow zero-shot code generation.

Benchmark + Language	Base	Reflexion	TP	FN	FP	TN
HumanEval (PY)	0.80	0.91	0.99	0.40	0.01	0.60
MBPP (PY)	0.80	0.77	0.84	0.59	0.16	0.41
HumanEval (RS)	0.60	0.68	0.87	0.37	0.13	0.63
MBPP (RS)	0.71	0.75	0.84	0.51	0.16	0.49

Table 2: Overall accuracy and test generation performance for HumanEval and MBPP. For Rust, HumanEval is the hardest 50 problems from HumanEval Python translated to Rust with MultiPL-E [3]. TP: unit tests pass, solution pass; FN: unit tests fail, solution pass; FP: unit tests pass, solution fail; TN: unit tests fail, solution fail.

257 **Results** Reflexion outperforms all baseline accuracies and sets new state-of-the-art standards on
 258 all benchmarks for Python and Rust except for MBPP Python 1. We further investigate the inferior
 259 performance of Reflexion on MBPP Python.

260 **Analysis** We acknowledge that self-reflecting code-generation agents are bound to their ability to
 261 write diverse, comprehensive tests. Therefore, in the case in which the model generates a flaky test
 262 suite, it is possible that all tests pass on an incorrect solution and lead to a false positive label on a
 263 code completion [9]. On the other hand, if the model produces an incorrectly written test suite, it
 264 is possible for some of the tests to fail on a correct solution, leading to a self-reflection generation
 265 that is conditioned on a false negative code completion. Given the implementation of Reflexion,
 266 false negatives are preferred over false positives as the agent may be able to use self-reflection to
 267 identify the incorrect test(s) and prompt itself to keep the original code completion intact. On the
 268 other hand, if an invalid test suite returns a false positive completion (all internal test cases pass
 269 but the implementation is incorrect), the agent will prematurely report an invalid submission. In 2,
 270 various conditions are measured to analyze performance beyond pass@1 accuracy. Previously, we
 271 displayed the inferior performance of Reflexion to the baseline GPT-4 on MBPP Python. In 2, we
 272 observe a notable discrepancy between the false positive labels produced by internal test execution,
 273 $P(\text{not pass@1 generation correct} | \text{tests pass})$. That is, the probability that a submission will fail given
 274 that it passes all unit tests. For HumanEval and MBPP Python, the baseline pass@1 accuracies are
 275 relatively similar, 82% and 80%, respectively. However, the false positive test execution rate for
 276 MBPP Python is 16.3% while the rate for HumanEval Python is a mere 1.4%, leading to 91% overall
 277 accuracy 1.

Approach	Test Generation	Self-reflection	Pass@1 (Acc)
Base model	False	False	0.60
Test generation omission	False	True	0.52
Self-reflection omission	True	False	0.60
Reflexion	True	True	0.68

Table 3: Pass@1 accuracy for various compromised approaches on the Reflexion approach using GPT-4 as the base model on HumanEval Rust - 50 hardest problems

278 **Ablation study** We test the composite approach of Reflexion for test generation and self-reflection
 279 cooperation on a subset of the 50 hardest HumanEval Rust problems. Our Rust compiler environment
 280 provides verbose error logs and helpful debugging hints, therefore serving as a good playground
 281 for compromised approaches. First, we omit internal test generation and execution steps, which
 282 test the agent to self-reflect without guidance from current implementations. 3 shows an inferior
 283 52% vs 60% (baseline) accuracy, which suggests that the agent is unable to determine if the current
 284 implementation is correct without unit tests. Therefore, the agent must participate in all iterations of
 285 the run without the option to return early, performing harmful edits to the implementation.

286 Next, we test self-reflection contribution by omitting the natural language explanation step following
 287 failed unit test suite evaluations. Intuitively, this challenges the agent to combine the tasks of
 288 error identification and implementation improvement across all failed unit tests. Interestingly, the
 289 compromised agent does not improve performance over the baseline run. We observe that the test
 290 generation and code compilation steps are able to catch syntax and logic errors, but the implementation
 291 fixes do not reflect these indications. These empirical results suggest that several recent works that

292 propose *blind* trial and error debugging techniques without self-reflection are ineffective on harder
293 tasks such as writing complex programs in Rust.

294 **5 Limitations**

295 At its core, Reflexion is an optimization technique that uses natural language to do policy optimization.
296 Policy optimization is a powerful approach to improve action choice through experience, but it may
297 still succumb to non-optimal local minima solutions. In this study, we limit long-term memory to
298 a sliding window with maximum capacity, but we encourage future work to extend the memory
299 component of *Reflexion* with more advanced structures such as vector embedding databases or
300 traditional SQL databases. Specific to code generation, there are many practical limitations to test-
301 driven development in specifying accurate input-output mappings such as non-deterministic generator
302 functions, impure functions that interact with APIs, functions that vary output according to hardware
303 specifications, or functions that invoke parallel or concurrent behavior that may be difficult to predict.

304 **6 Broader impact**

305 Large language models are increasingly used to interact with external environments (e.g. the Internet,
306 software, robotics, etc.) and humans. Our work has the potential of reinforcing and empowering
307 these agents toward greater automation and work efficiency, but it also amplifies the risks when these
308 agents were put into misuse. We believe that this direction of research will need more effort in safety
309 and ethical considerations.

310 On the other hand, reinforcement learning has suffered from its black-box policy and optimiza-
311 tion setups in which interpretability and alignment have been challenging. Our proposed “verbal”
312 reinforcement learning might address some of the issues and turn autonomous agents more inter-
313 pretable and diagnosable. For example, in the case of tool-usage that may be too hard for humans to
314 understand, self-reflections could be monitored to ensure proper intent before using the tool.

315 **7 Conclusion**

316 In this work, we present *Reflexion*, an approach that leverages verbal reinforcement to teach agents
317 to learn from past mistakes. We empirically show that Reflexion agents significantly outperform
318 currently widely-used decision-making approaches by utilizing self-reflection. In future work,
319 Reflexion could be used to employ more advanced techniques that have been thoroughly studied in
320 traditional RL settings, such as value learning in natural language or off-policy exploration techniques.

321 **8 Reproducibility**

322 We highly advise others to use isolated execution environments when running autonomous code
323 writing experiments as the generated code is not validated before execution.

324 References

- 325 [1] Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Gopalakrishnan,
326 K., Hausman, K., Herzog, A., et al. (2022). Do as i can, not as i say: Grounding language in
327 robotic affordances. *arXiv preprint arXiv:2204.01691*.
- 328 [2] Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C.,
329 Terry, M., Le, Q., et al. (2021). Program synthesis with large language models. *arXiv preprint*
330 *arXiv:2108.07732*.
- 331 [3] Cassano, F., Gouwar, J., Nguyen, D., Nguyen, S., Phipps-Costin, L., Pinckney, D., Yee, M.-H., Zi,
332 Y., Anderson, C. J., Feldman, M. Q., Guha, A., Greenberg, M., and Jangda, A. (2022). Multipl-e:
333 A scalable and extensible approach to benchmarking neural code generation.
- 334 [4] Chen, B., Zhang, F., Nguyen, A., Zan, D., Lin, Z., Lou, J.-G., and Chen, W. (2022). Codet: Code
335 generation with generated tests. *arXiv preprint arXiv:2207.10397*.
- 336 [5] Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y.,
337 Joseph, N., Brockman, G., et al. (2021). Evaluating large language models trained on code. *arXiv*
338 *preprint arXiv:2107.03374*.
- 339 [6] Chen, X., Lin, M., Schärli, N., and Zhou, D. (2023). Teaching large language models to
340 self-debug. *arXiv preprint arXiv:2304.05128*.
- 341 [7] Côté, M.-A., Kádár, A., Yuan, X., Kybartas, B., Barnes, T., Fine, E., Moore, J., Hausknecht, M.,
342 El Asri, L., Adada, M., et al. (2019). Textworld: A learning environment for text-based games. In
343 *Computer Games: 7th Workshop, CGW 2018, Held in Conjunction with the 27th International*
344 *Conference on Artificial Intelligence, IJCAI 2018, Stockholm, Sweden, July 13, 2018, Revised*
345 *Selected Papers 7*, pages 41–75. Springer.
- 346 [8] Kosinski, M. (2023). Theory of mind may have spontaneously emerged in large language models.
347 *arXiv preprint arXiv:2302.02083*.
- 348 [9] Lam, W., Winter, S., Wei, A., Xie, T., Marinov, D., and Bell, J. (2020). A large-scale longitudinal
349 study of flaky tests. *Proc. ACM Program. Lang.*, 4(OOPSLA).
- 350 [10] Le, H., Wang, Y., Gotmare, A. D., Savarese, S., and Hoi, S. C. H. (2022). Coderl: Mastering
351 code generation through pretrained models and deep reinforcement learning. *Advances in Neural*
352 *Information Processing Systems*, 35:21314–21328.
- 353 [11] Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., Eccles, T., Keeling,
354 J., Gimeno, F., Dal Lago, A., et al. (2022). Competition-level code generation with alphacode.
355 *Science*, 378(6624):1092–1097.
- 356 [12] Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegrefe, S., Alon, U., Dziri, N.,
357 Prabhunoye, S., Yang, Y., et al. (2023). Self-refine: Iterative refinement with self-feedback. *arXiv*
358 *preprint arXiv:2303.17651*.
- 359 [13] Moghaddam, S. R. and Honey, C. J. (2023). Boosting theory-of-mind performance in large
360 language models via prompting. *arXiv preprint arXiv:2304.11490*.
- 361 [14] Nair, V., Schumacher, E., Tso, G., and Kannan, A. (2023). Dera: Enhancing large language
362 model completions with dialog-enabled resolving agents. *arXiv preprint arXiv:2303.17071*.
- 363 [15] Nakano, R., Hilton, J., Balaji, S., Wu, J., Ouyang, L., Kim, C., Hesse, C., Jain, S., Kosaraju, V.,
364 Saunders, W., et al. (2021). Webgpt: Browser-assisted question-answering with human feedback.
365 *arXiv preprint arXiv:2112.09332*.
- 366 [16] OpenAI (2023). Gpt-4 technical report. *ArXiv*.
- 367 [17] Park, J. S., O’Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. (2023).
368 Generative agents: Interactive simulacra of human behavior. *arXiv preprint arXiv:2304.03442*.

- 369 [18] Paul, D., Ismayilzada, M., Peyrard, M., Borges, B., Bosselut, A., West, R., and Faltings,
370 B. (2023). Refiner: Reasoning feedback on intermediate representations. *arXiv preprint*
371 *arXiv:2304.01904*.
- 372 [19] Pryzant, R., Iter, D., Li, J., Lee, Y. T., Zhu, C., and Zeng, M. (2023). Automatic prompt
373 optimization with "gradient descent" and beam search. *arXiv preprint arXiv:2305.03495*.
- 374 [20] Schick, T., Dwivedi-Yu, J., Dessì, R., Raileanu, R., Lomeli, M., Zettlemoyer, L., Cancedda, N.,
375 and Scialom, T. (2023). Toolformer: Language models can teach themselves to use tools. *arXiv*
376 *preprint arXiv:2302.04761*.
- 377 [21] Shen, Y., Song, K., Tan, X., Li, D., Lu, W., and Zhuang, Y. (2023). Hugginggpt: Solving ai
378 tasks with chatgpt and its friends in huggingface. *arXiv preprint arXiv:2303.17580*.
- 379 [22] Shridhar, M., Yuan, X., Côté, M.-A., Bisk, Y., Trischler, A., and Hausknecht, M. (2021).
380 ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proceedings*
381 *of the International Conference on Learning Representations (ICLR)*.
- 382 [23] Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT
383 Press, second edition.
- 384 [24] Ullman, T. (2023). Large language models fail on trivial alterations to theory-of-mind tasks.
385 *arXiv preprint arXiv:2302.08399*.
- 386 [25] Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M.,
387 Zhou, D., Metzler, D., et al. (2022a). Emergent abilities of large language models. *arXiv preprint*
388 *arXiv:2206.07682*.
- 389 [26] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., and Zhou, D. (2022b). Chain of
390 thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.
- 391 [27] Xie, Y., Kawaguchi, K., Zhao, Y., Zhao, X., Kan, M.-Y., He, J., and Xie, Q. (2023). Decomposi-
392 tion enhances reasoning via self-evaluation guided decoding. *arXiv preprint arXiv:2305.00633*.
- 393 [28] Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D.
394 (2018). HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Conference*
395 *on Empirical Methods in Natural Language Processing (EMNLP)*.
- 396 [29] Yao, S., Chen, H., Yang, J., and Narasimhan, K. (preprint). Webshop: Towards scalable
397 real-world web interaction with grounded language agents. In *ArXiv*.
- 398 [30] Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K., and Cao, Y. (2023). ReAct:
399 Synergizing reasoning and acting in language models. In *International Conference on Learning*
400 *Representations (ICLR)*.
- 401 [31] Yoran, O., Wolfson, T., Bogin, B., Katz, U., Deutch, D., and Berant, J. (2023). Answering
402 questions by meta-reasoning over multiple chains of thought. *arXiv preprint arXiv:2304.13007*.

403 **A Additional Information and Examples**

404 **B Decision-making**

Environment: You are in the middle of a room. Looking quickly around you, you see a bed 1, a desk 2, a desk 1, a drawer 6, a drawer 5, a drawer 4, a drawer 3, a drawer 2, a drawer 1, a garbagecan 1, a laundryhamper 1, a safe 1, a shelf 6, a shelf 5, a shelf 4, a shelf 3, a shelf 2, and a shelf 1.

Task: examine the mug with the desklamp.

Trial #1

> think: To solve the task, I need to find and take a mug, then find and use a desklamp.

> think: First I need to find a mug. A mug is more likely to appear in drawer (1-6), desk (1-2), shelf (1-6), garbagecan (1), laundryhamper (1). I can check one by one, starting with drawer 1.

> go to drawer 1

The drawer 1 is closed.

...

> open drawer 6

You open the drawer 6. The drawer 6 is open. In it, you see a keychain 2.

> go to desk 1

On the desk 1, you see a creditcard 3, a desklamp 1, a laptop 2, a mug 1, a pen 1, and a pencil 1.

> think: Now I find a mug (1). Next, I need to take it.

> take mug 1 from desk 1

You pick up the mug 1 from the desk 1.

> think: Now I take a mug (1). Next, I need to find a desklamp. A desklamp is more likely to appear in desk (1-2), sidetable (1-2), shelf (1-6), bed (1), drawer (1-6). I can check one by one, starting with desk 1.

> go to desk 1

Nothing happens.

> go to desk 2

On the desk 2, you see a alarmclock 1, a bowl 1, a mug 3, a pencil 3, and a pencil 2.

> think: Now I find a desklamp (1). Next, I need to use it.

> use desklamp 1

Nothing happens.

> use desklamp 1

Nothing happens.

Status: Fail

Reflection: In this environment, my plan was to find a mug then find and use a desklamp. However, the task says to examine the mug with the desklamp. I should have looked for the desklamp first, then looked for the mug. I noticed that the desklamp was found on desk 1. In the next trial, I will go to desk 1, find the lamp, then look for the mug and examine it with the desklamp.

Trial #2

> go to desk 1

On the desk 1, you see a creditcard 3, a desklamp 1, a laptop 2, a mug 1, a pen 1, and a pencil 1.

> think: To solve the task, I need to find and take a mug, then find and use a desklamp.

> take mug 1 from desk 1

You pick up the mug 1 from the desk 1.

> think: To solve the task, I need to find and take a mug, then find and use a desklamp.

> use desklamp 1

You turn on the desklamp 1.

Status: Success

Figure 5: [Top] An AlfWorld trajectory in which the agent failed due to inefficient planning. In the reflection, the agent recognizes that it should have looked for the desklamp then the mug, not the mug then the desklamp. [Bottom] The agent is able to correct its reasoning trace and execute a sequence of actions in a concise manner.

405 **B.1 WebShop Limitation**

406 In 5, we briefly state that Reflexion struggles to overcome local minima choices that require extremely
407 creative behavior to escape. We observe this shortcoming in an experiment on WebShop [29].
408 WebShop is a web-based problem-solving benchmark that tests agents to navigate an e-commerce
409 website to locate and purchase products given requests from clients. We test a two-shot ReAct +
410 Reflexion agent in 100 environments. However, after only four trials, we terminate the runs as the
411 agent does not show signs of improvement 6. Further, the agent does not generate helpful, intuitive
412 self-reflections after failed attempts. We conclude that Reflexion is unable to solve tasks that require
413 a significant amount of diversity and exploration. In AlfWorld, the agent is able to adequately explore
414 new environments because the permissible actions can be seen in the observations. In HotPotQA,
415 the agent faces a similar WebShop search query task but is more successful as the search space for
416 Wikipedia articles is more diverse and requires less precise search queries. A common problem for
417 e-commerce search engines is properly handling ambiguity in natural language search interpretations.
418 Thus, WebShop presents a task that requires very diverse and unique behavior from a Reflexion agent.

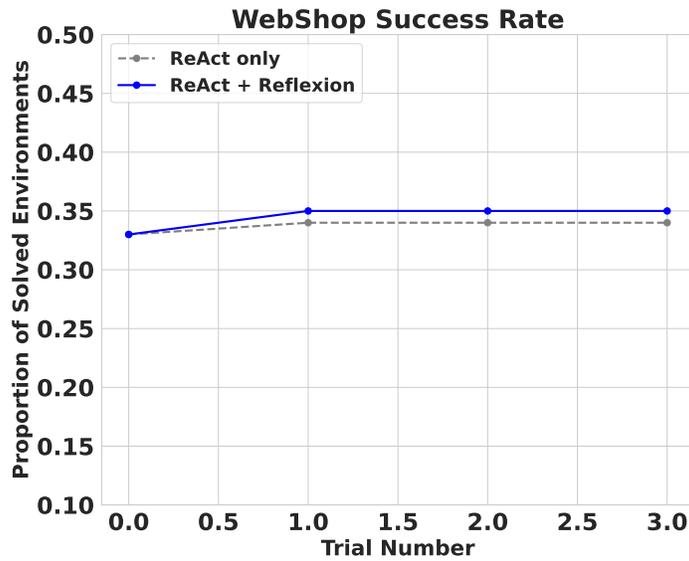


Figure 6: Reflexion vs React performance on WebShop across 100 customer shopping requests. ReAct + Reflexion fails to significantly outperform ReAct.

419 **C Programming**

420 Programming LLM calls require strict instructions to produce function bodies *only*, due to the
421 extensive dialogue training of the LLMs. A few programming examples are reported below with
422 instructions **highlighted in blue** and templates. See the full implementation at <https://github.com/noahshinn024/reflexion>.
423

424 **C.0.1 Programming function implementation example (HumanEval Python)**

425 Sample function signature:

426

```
427 def minSubArraySum(nums):  
428 1     """  
429 2     Given an array of integers nums, find the minimum sum of  
430 3     any  
431     non-empty sub-array of nums.  
432 4     Example  
433 5     minSubArraySum([2, 3, 4, 1, 2, 4]) == 1  
434 6
```

```
435 7     minSubArraySum([-1, -2, -3]) == -6
436 8     """
```

438 Sample function body implementation:

```
439
440
441 1     min_sum = float('inf')
442 2     for i in range(len(nums)):
443 3         current_sum = 0
444 4         for j in range(i, len(nums)):
445 5             current_sum += nums[j]
446 6             if current_sum < min_sum:
447 7                 min_sum = current_sum
448 8     return min_sum
```

450 C.0.2 Reflexion Actor instruction

451 You are a Python writing assistant. You will be given your previous implementation of a function,
452 a series of unit tests results, and your self-reflection on your previous implementation. Apply the
453 necessary changes below by responding only with the improved body of the function. Do not include
454 the signature in your response. The first line of your response should have 4 spaces of indentation so
455 that it fits syntactically with the user provided signature. You will be given a few examples by the
456 user.

457 Reflexion Actor generations follow the form:

458 (Instruction)

459 (Function implementation)

460 (Unit test feedback)

461 (Self-reflection)

462 (Instruction for next function implmentation)

463 Full examples are shown in <https://github.com/noahshinn024/reflexion>

464 C.0.3 Reflexion Self-reflection instruction and example

465 You are a Python writing assistant. You will be given your previous implementation of a function,
466 a series of unit tests results, and your self-reflection on your previous implementation. Apply the
467 necessary changes below by responding only with the improved body of the function. Do not include
468 the signature in your response. The first line of your response should have 4 spaces of indentation so
469 that it fits syntactically with the user provided signature. You will be given a few examples by the
470 user. Reflexion Self-Reflection generations follow the form:

471 (Instruction)

472 (Function implementation)

473 (Unit test feedback)

474

475 C.0.4 Reflexion programming *no* Self-Reflection ablation example

476 Reflexion *no* Self-Reflection ablation Actor generations follow the form:

477 (Instruction)

478 (Function implementation)

479 (Unit test feedback)

480 (Self-reflection)

481 (Instruction for next function implmentation)

482 C.0.5 Reflexion programming *no* test generation ablation example

483 Reflexion *no* test generation ablation Actor generations follow the form:

484 (Instruction)

485 (Function implementation)
486 ~~(Unit test feedback)~~
487 (Self-reflection)
488 (Instruction for next function implementation)

489 **D Reasoning**

490 **D.1 Full example**

Trial #1: Question: Grown-Ups starred the actor who was best known for which role on "'Allo 'Allo!'?"

Thought 1: I need to search Grown-Ups and "'Allo 'Allo!", find the actor who starred in Grown-Ups, then find the role he was best known for in "'Allo 'Allo!".

Action 1: Search[Grown-Ups]

Observation 1: Grown-Ups is a 1980 British BBC television film devised and directed by Mike Leigh. It stars Lesley Manville, Philip Davis, Brenda Blethyn, Janine Duvitski, Lindsay Duncan and Sam Kelly. It was edited by Robin Sales and produced by Louis Marks for the BBC, and originally shown on BBC 2 on 28 November 1980.

Thought 2: Grown-Ups starred Lesley Manville, Philip Davis, Brenda Blethyn, Janine Duvitski, Lindsay Duncan and Sam Kelly. I need to search "'Allo 'Allo!" and find which actor was best known for which role.

Action 2: Search["'Allo 'Allo!"]

Observation 2: Could not find ["'Allo 'Allo!"]. Similar: ["'Allo 'Allo!", "List of 'Allo 'Allo! characters", 'Gorden Kaye', "'Allo 'Allo! (series 2)", 'Allô, allô', 'Francesca Gonshaw', "'Allo 'Allo! (series 4)", 'Kim Hartman', "'Allo 'Allo! (series 6)", 'Jeremy Lloyd']

Thought 3: To find the actor who was best known for which role on "'Allo 'Allo!", I can search Gorden Kaye.

Action 3: Search[Gorden Kaye]

Observation 3: Gordon Irving Kaye (7 April 1941 – 23 January 2017), known professionally as Gorden Kaye, was an English actor, best known for playing womanising café owner René Artois in the television comedy series 'Allo 'Allo!.

Thought 4: Gorden Kaye was best known for playing womanising café owner René Artois in the television comedy series "'Allo 'Allo!". So the answer is Rene Artois.

Action 4: Finish[Rene Artois]

Observation 4: Answer is INCORRECT

Reflection: I searched the wrong title for the show, "'Allo 'Allo!", which resulted in no results. I should have searched the show's main character, Gorden Kaye, to find the role he was best known for in the show.

Trial #2: Question: Grown-Ups starred the actor who was best known for which role on "'Allo 'Allo!'?"

Thought 1: I need to search the actor who was best known for a role on "'Allo 'Allo!" and find out what role they were best known for.

Action 1: Search[Grown-Ups]

Observation 1: Grown-Ups is a 1980 British BBC television film devised and directed by Mike Leigh. It stars Lesley Manville, Philip Davis, Brenda Blethyn, Janine Duvitski, Lindsay Duncan and Sam Kelly. It was edited by Robin Sales and produced by Louis Marks for the BBC, and originally shown on BBC 2 on 28 November 1980.

Thought 2: The paragraph does not mention the actor who was best known for a role on "'Allo 'Allo!". I need to search the actor's name instead.

Action 2: Search[Sam Kelly]

Observation 2: Roger Michael Kelly (19 December 1943 – 14 June 2014), known by the stage name Sam Kelly, was an English actor who appeared in film, television, radio and theatre. He is best known for his roles as Captain Hans Geering in 'Allo 'Allo!, Warren in Porridge, Sam in On the Up, and Ted Liversidge in Barbara.

Thought 3: Sam Kelly is best known for his role as Captain Hans Geering in "'Allo 'Allo!", so the answer is Captain Hans Geering.

Action 3: Finish[Captain Hans Geering]

Observation 3: Answer is CORRECT

Figure 7: Two HotPotQA trials within the same environment and task. The Reflexion + ReAct agent uses self-reflection to determine a better search method for the next trial.

491 **D.2 Chain-of-Thought + Reflexion**

Trial #1: Question: What profession does John Lanchester and Alan Dean Foster have in common?

Thought 1: Let's think step by step. John Lanchester is a novelist, journalist, and critic. Alan Dean Foster is a novelist and screenwriter. So the profession John Lanchester and Alan Dean Foster have in common is novelist and screenwriter.

Action 1: Finish[novelist and screenwriter]

Observation 1: Answer is INCORRECT

Reflection: My reasoning for the profession John Lanchester and Alan Dean Foster have in common failed because I incorrectly assumed that they both had the same profession. In the future, when attempting this question, I should focus on better researching the individual backgrounds of both authors to ensure that I am accurately identifying their professions. Additionally, I should consider the possibility that they may have multiple professions in common.

Trial #2: Question: What profession does John Lanchester and Alan Dean Foster have in common?

Thought 1: Let's think step by step. John Lanchester is a novelist, journalist, and financial journalist. Alan Dean Foster is a novelist and screenwriter. So the profession John Lanchester and Alan Dean Foster have in common is novelist.

Action 1: Finish[novelist]

Observation 1: Answer is CORRECT

492 **D.3 HotPotQA Chain-of-Thought (GT) + Reflexion**

Context: The Battle of White Plains was a battle in the New York and New Jersey campaign of the American Revolutionary War fought on October 28, 1776, near White Plains, New York. Following the retreat ...

Trial #1: Question: What was a series of battles during the Revolutionary War, for control of New York City and the state of New Jersey, fought on October 28, 1776 near White Plains, New York?

Thought 1: The question is asking about a series of battles during the Revolutionary War for control of New York City and the state of New Jersey, fought on October 28, 1776 near White Plains, New York. From the context, I know that this was the Battle of White Plains.

Action 1: Finish[Battle of White Plains]

Observation 1: Answer is INCORRECT

Trial #2: Question: What was a series of battles during the Revolutionary War, for control of New York City and the state of New Jersey, fought on October 28, 1776 near White Plains, New York?

Thought 1: The question is asking for a series of battles during the Revolutionary War, for control of New York City and the state of New Jersey, fought on October 28, 1776 near White Plains, New York. Based on the context, I know that this series of battles was the New York and New Jersey campaign.

Action 1: Finish[The New York and New Jersey campaign]

Observation 1: Answer is CORRECT

Reflection: Upon reflecting on the incorrect answer I provided, I realize that I may not have provided enough context to accurately answer the question. The question asked for a series of battles, but I only provided the name of one battle. To better answer this question in the future, I will make sure to provide more context, such as the name of the campaign, which was the New York and New Jersey campaign, and the fact that it was a series of battles. Additionally, I will make sure to include the dates of the battles, as well as the locations, to ensure that my answer is comprehensive and accurate.

493 D.4 HotPotQA episodic memory (EPM) ablation prompts

494 D.4.1 (EPM) Chain-of-Thought + Reflexion

Trial #1: Question: Which of Jonny Craig and Pete Doherty has been a member of more bands ?

Thought 1: Let's think step by step. Jonny Craig has been a member of six bands, including Emarosa, Dance Gavin Dance, and Slaves. Pete Doherty has been a member of seven bands, including The Libertines, Babyshambles, and The Puta Madres. So Pete Doherty has been a member of more bands.

Action 1: Finish[Pete Doherty]

Observation 1: Answer is INCORRECT

Reflection: My reasoning for which musician has been a member of more bands failed because I did not take into account that Jonny Craig has been a member of more bands in the past, even though Pete Doherty is currently a member of more bands. In the future, when attempting this question, I should focus on researching the past and current bands of both musicians to ensure that I am accurately comparing their band memberships.

Trial #2: Question: Which of Jonny Craig and Pete Doherty has been a member of more bands ?

Thought 1: Let's think step by step. I will research the past and current bands of both musicians to ensure that I am accurately comparing their band memberships. After researching, I have found that Jonny Craig has been a member of seven bands, including Emarosa, Dance Gavin Dance, and Slaves. Pete Doherty has been a member of seven bands, including The Libertines, Babyshambles, and The Puta Madres. So Jonny Craig has been a member of more bands.

Action 2: Finish[Jonny Craig]

Observation 2: Answer is CORRECT

495 D.4.2 (EPM) Chain-of-Thought (GT) + Reflexion

Context: Hari Bahadur Basnet is a Nepalese politician. He is the head of the Foreign Relations Department of the Rastriya Janashakti Party. Basnet holds a M.Sc. in Engineering. A Master of Science (Latin: "Magister Scientiae" ; abbreviated MS, M.S., MSc, M.Sc., MSci, M.Sci., ScM, Sc.M., SciM or Sci.M.) is a master's degree in the field of science...

Trial #1: Question: The head of the Foreign Relations Department of the Rastriya Janashakti Party holds a degree that can be abbreviated MS, M.S., or ScM, in what field?

(Previous trajectory)

Thought 1: The question is asking what field of study the head of the Foreign Relations Department of the Rastriya Janashakti Party holds a degree in. From the context, I know that the degree is a Master of Science degree, which is typically granted for studies in sciences, engineering, and medicine.

Action 1: Finish[Sciences, Engineering, and Medicine]

Observation 1: Answer is INCORRECT

Reflection: For the first trial, I misunderstood the question. I thought the question was asking for a category of degrees, rather than a specific degree. I learned from my mistake and in the second trial, I focused more closely on the question to understand that it was asking for a specific field of study.

Trial #2: Question: The head of the Foreign Relations Department of the Rastriya Janashakti Party holds a degree that can be abbreviated MS, M.S., or ScM, in what field?

(Previous trajectory)

Thought 1: The question is asking what field of study the head of the Foreign Relations Department of the Rastriya Janashakti Party holds a degree in. From the context, I know that the degree is a Master of Science degree, which is typically granted for studies in sciences, engineering, and medicine. However, the question is asking for the field of study, not the degree. Based on the context, I know that Hari Bahadur Basnet holds a M.Sc. in Engineering.

Action 1: Finish[Engineering]

Observation 1: Answer is CORRECT