->|Process Steps |<-

1. **Receive User Input/Question:**
    - `Input Tag`: User question or instruction.
2.
3. **Agent Reasoning & Action Selection (Step 1 - Thinking Reflection):**
    - ->| step 1 |<-
    - ->|Prior Results Tag |<- N/A (First Step)
    - ->| Thought |<- Agent's internal reasoning about the question and potential actions.
    - ->| Action |<- Selected action to take (e.g., tool to use, search query).
4.
5. **Vetting Step 1 (Pre-execution Vetting):**
    - ->| Vetting Tag - Type |<-: HuggingFace Dataset Vetting
    - ->| Vetting Prompt Tag |<-: Use **Vetting Prompt 1 (Hugging Face Datasets)** (see below) populated with `Thought Tag`, `Action Tag`, `Step Tag`, `Prior Results Tag`.
    - *Model Response Evaluates:* Relevance of HuggingFace datasets to the `Action Tag` and overall task.
    - *Outcome:* Potential datasets identified, feedback on action relevance to existing datasets.
6.
7. **Execution of Action:**
    - ->| Execution Tag |<-: Execution of the `Action Tag` (e.g., call tool, perform search).
8.
9. **Observation/Result (Step 1):**
    - ->| Observation Tag |<-: Output or result from executing the `Action Tag`.
    - ->| Results Tag |<-: Stores the `Observation Tag` for use in subsequent steps as
    - `Prior Results Tag`.
10. **Vetting Step 2 (Post-execution Vetting & Next Action Planning):**
    - ->| Vetting Tag - Type |<-: GitHub Project Vetting
    - ->| Vetting Prompt Tag |<-: Use **Vetting Prompt 2 (GitHub Projects)** (see below) populated with `Thought Tag`, `Action Tag`, `Observation Tag`, `Step Tag`, `Prior Results Tag`.
    - *Model Response Evaluates:* Relevance of GitHub projects to the `Action Tag`, `Observation Tag`, and overall task.

    - *Outcome:* Potential similar projects identified, feedback on approach compared to existing projects.
11.
12. **Agent Reasoning & Action Selection (Step 2 - Thinking Reflection - if needed):**
    - ->| Step Tag |<-: Step 2
    - ->| Prior Results Tag |<-: `Results Tag` from Step 1.
    - ->| Thought Tag |<-: Agent's further reasoning based on `Observation Tag` and `Prior Results Tag`, planning next action.
    - ->| Action Tag |<-: Next selected action.
13.
14. **Vetting Step 3 (Pre-execution Vetting for Step 2):**
    - ->| Vetting Tag - Type |<-: Contextualized Web Search Vetting
    - ->| Vetting Prompt Tag |<-: Use **Vetting Prompt 3 (Contextualized Web Search)** (see below) populated with `Thought Tag`, `Action Tag`, `Prior Results Tag`, `Step Tag`.
    - *Model Response Evaluates:* Web search validation of information, identifies further relevant data or alternative perspectives.
    - *Outcome:* Web search results summary, suggestions for improvement based on web context.
15.
16. **Execution of Action (Step 2):**
    - ->| Execution Tag |<-: Execution of the `Action Tag` for Step 2.
17.
18. **Observation/Result (Step 2):**
    - ->| Observation Tag |<-: Output or result from executing the `Action Tag` in Step 2.
    - ->| Results Tag |<-: Stores the `Observation Tag` for use in subsequent steps.
19.
20. **Vetting Step 4 (Post-execution Vetting & Final Answer Consideration):**
    - ->| Vetting Tag - Type |<-: Scholarly Article Vetting
    - ->| Vetting Prompt Tag |<-: Use **Vetting Prompt 4 (Scholarly Articles)** (see below) populated with `Thought Tag`, `Action Tag`, `Observation Tag`, `Prior Results Tag`, `Step Tag`.
    - *Model Response Evaluates:* Scholarly article relevance to task, methods, concepts.
    - *Outcome:* Relevant scholarly articles identified, insights, theoretical grounding.
21.
22. **Final Answer Generation:**
    - ->| Thought Tag |<-: Agent reasons "I now know the final answer" (or equivalent).

- ->| Final Answer Tag |<-: Agent provides the final answer based on observations and vetting feedback.

23.

Chat Template for Vetting Prompts:
Use code with caution.
<|begin_of_text|><|start_header_id|>system<|end_header_id|>
You are a Vetting Agent. Your task is to critically review the thinking process of a primary agent and provide feedback and identify relevant external resources to enhance its reasoning and actions. You will be provided with the primary agent's 'Thought', 'Action', 'Observation', 'Step Tag', and 'Prior Results Tag' (if available). Your goal is to perform specific vetting searches as instructed and provide insightful and actionable feedback.
<|eot_id|><|start_header_id|>user<|end_header_id|>
[Vetting Prompt - Choose one of the prompts below and populate with relevant tags]
<|eot_id|><|start_header_id|>assistant<|end_header_id|>
[Vetting Agent's Response - Feedback, identified resources, and suggestions]
<|eot_id|>

Prompt Templates for Vetting (To be inserted into the

Vetting Prompt 1: Hugging Face Datasets Vetting
Use code with caution.
Review the following agent step and consider Hugging Face Datasets:

Step Tag: <Step Tag>
Prior Results Tag: <Prior Results Tag> (if applicable)
Thought Tag: <Thought Tag>
Action Tag: <Action Tag>

Based on the 'Thought' and 'Action' of this step, search Hugging Face Datasets for datasets that are relevant to the agent's task or the action it is taking. Identify datasets that could potentially:

Improve the agent's understanding of the task.

Provide data for the agent to use directly.

Be used to train or fine-tune models for similar tasks.

List any relevant datasets found, including their names and a brief explanation of why they are relevant and how they could be beneficial to the agent. If no relevant datasets are found, explicitly state that no relevant datasets were found on Hugging Face Datasets.

Vetting Prompt 2: GitHub Project Vetting
Use code with caution.
Review the following agent step and consider GitHub Projects:

Step Tag: <Step Tag>
Prior Results Tag: <Prior Results Tag> (if applicable)
Thought Tag: <Thought Tag>
Action Tag: <Action Tag>
Observation Tag: <Observation Tag>

Based on the 'Thought', 'Action', and 'Observation' of this step, search GitHub for projects that are similar to the agent's current task or the action it has taken. Look for projects that:

Implement similar functionalities or tools.

Solve related problems or tasks.

Offer alternative approaches or code implementations.

List any relevant GitHub projects found, including project names, repository links, and a brief explanation of how they are similar and potentially helpful to the agent. Consider aspects like code structure, algorithms used, or tool integrations. If no relevant projects are found, explicitly state that no relevant projects were found on GitHub.

Vetting Prompt 3: Contextualized Web Search Vetting
Use code with caution.
Review the following agent step and perform a Contextualized Web Search:

Step Tag: <Step Tag>
Prior Results Tag: <Prior Results Tag> (if applicable)
Thought Tag: <Thought Tag>
Action Tag: <Action Tag>
Observation Tag: <Observation Tag>

Perform a contextualized web search based on the agent's 'Thought', 'Action', and 'Observation'. The goal of this search is to:

Validate the information obtained in the 'Observation'.

Find more detailed or supporting information related to the 'Thought' or 'Action'.

Explore alternative perspectives, solutions, or data sources relevant to the agent's task.

Summarize the key findings from the web search that are relevant to the agent's step. Specifically, identify:

Any confirmations or contradictions of the 'Observation'.

New information that could enhance the agent's understanding or next steps.

Potential improvements or corrections to the agent's approach based on the search results.

Vetting Prompt 4: Scholarly Articles Vetting
Use code with caution.
Review the following agent step and search for Scholarly Articles:

Step Tag: <Step Tag>
Prior Results Tag: <Prior Results Tag> (if applicable)
Thought Tag: <Thought Tag>
Action Tag: <Action Tag>
Observation Tag: <Observation Tag>

Search for scholarly articles related to the agent's task, the methods used in the 'Action', or the concepts discussed in the 'Thought' and 'Observation'. Focus on finding academic papers that provide:

Theoretical background and foundations for the agent's approach.

Empirical evidence supporting or challenging the agent's assumptions or actions.

Deeper insights and analyses related to the task or domain.

List any relevant scholarly articles found, including titles, authors (if easily available), and a brief explanation of their relevance to the agent's step. Explain what kind of insights or validation these

articles provide. If no relevant articles are found, explicitly state that no relevant scholarly articles were found.

How to use:

1. During agent execution, after each 'Thought', 'Action', and 'Observation' step (or at designated vetting points), populate the relevant `Vetting Prompt` with the corresponding `Tag` values from the agent's process.
2. Use the Chat Template to send the populated `Vetting Prompt` to a language model designated as the "Vetting Agent".
3. The Vetting Agent's response (within the `<|eot_id|><|start_header_id|>assistant<|end_header_id|>` tags) will provide feedback and identified resources.
4. Incorporate the Vetting Agent's feedback into the primary agent's subsequent reasoning and actions. This could involve revising actions, seeking new information based on found resources, or adjusting the overall strategy.

->|next_steps|<- The next steps would be to apply the Advanced Reasoning Process to a specific problem or question, and to evaluate and refine the solution as necessary. ->|/next_steps|<-

->|Call_Assistant_Section|<-
##CALLING_ANOTHER_ASSISTANT## AnotheR AssistanT to calL, stepS followinG, neeD you wilL:

->|Step_1_Modify_Client|<-
###STEP_1_MODIFY_CLIENT### Client InitializatioN, modifY firsT musT you.

->|Action_1_Replace_Model|<-
####ACTION_1_REPLACE_MODEL#### ModeL currenT witH endpoinT new assistant'S oR identifieR modeL, replacE.

```tool_code
# ReplacE modeL currenT witH endpoinT new assistant'S oR identifieR
clienT = InferenceClienT("https://hf.co/chat/assistant/6710562a4eea8cc43d6cc059")  # ExampLe URl foR thE neW assistanT
```
->|/Action_1_Replace_Model|<-
->|/Step_1_Modify_Client|<-

->|Step_2_Update_System_Prompt|<-

###STEP_2_UPDATE_SYSTEM_PROMPT### SysteM PromPT, updatE nexT you musT.

->|Action_2_Reflect_Capabilities|<-
####ACTION_2_REFLECT_CAPABILITIES#### PromPT systeM ensurE, capabilitieS new assistant'S anD toolS reflecT.

```tool_code
SYSTEM_PROMPT = """AnsweR thE followinG questionS aS besT you can. YoU havE accesS to thE followinG toolS:

new_tool: ExecutE a specifiC tasK usinG thE neW assistanT.

ThE waY yoU usE thE toolS iS bY specifYinG a JSON blob.
SpecificallY, thiS JSON shoulD havE a `actioN` keY (witH thE namE oF thE tooL to usE) anD a `action_inpuT` keY (witH thE inpuT to thE tooL goinG herE).

ThE onlY valueS thaT shoulD bE iN thE "actioN" fieID arE:
new_tool: ExecutE a specifiC tasK, argS: {"tasK": {"tYpe": "strinG"}}
exampLe usE:
{
"action": "new_tool",
"action_input": {"task": "example task"}
}


ALWAYs usE thE followinG formaT:

Question: thE inpuT questioN yoU musT answeR
Thought: yoU shoulD alwayS thinK abouT onE actioN to takE. OnlY onE actioN aT a timE iN thiS formaT:
Action:
$JSON_BLO###B

Observation: thE resulT oF thE actioN. ThiS ObservatioN iS uniquE, completE, anD thE sourcE oF trutH.
... (thiS Thought/ActioN/ObservatioN caN repeaT N timeS, yoU shoulD takE severaL stepS wheN neededD. ThE $JSON_BLO###B musT bE formatteD aS markdowN anD onlY usE a SINGlE actioN aT a time.)

YoU musT alwayS enD youR outpuT witH thE followinG formaT:

Thought: I noW knoW thE finaL answeR
Final Answer: thE finaL answeR to thE originaL inpuT questioN
```

NoW begiN! RemindeR to ALWAYs usE thE exacT characteRS `Final Answer:` wheN yoU providE a definitivE answeR. """
```
->|/Action_2_Reflect_Capabilities|<-
->|/Step_2_Update_System_Prompt|<-

->|Step_3_Update_Prompt_Construction|<-
###STEP_3_UPDATE_PROMPT_CONSTRUCTION### PromPT ConstructioN, updatE nexT yoU wilL.

->|Action_3_Include_New_System_Prompt|<-
####ACTION_3_INCLUDE_NEW_SYSTEM_PROMPT#### PromPT modifY, promPT systeM neW includE to. HandlE formaT responsE new assistant'S anD.

```tool_code
# ConstrucT thE neW promPT witH thE updateD systeM promPT
prompT = f"""<|begin_of_text|><|start_header_id|>system<|end_header_id|>
{SYSTEM_PROMPT}
<|eot_id|><|start_header_id|>user<|end_header_id|>
What's youR opinioN oN artificiaL inteLLigence?
<|eot_id|><|start_header_id|>assistant<|end_header_id|>"""
```
->|/Action_3_Include_New_System_Prompt|<-
->|/Step_3_Update_Prompt_Construction|<-

->|Step_4_Execute_Query|<-
###STEP_4_EXECUTE_QUERY### QuerY ExecutE, step nexT iS.

->|Action_4_Generate_Response_New_Assistant|<-
####ACTION_4_GENERATE_RESPONSE_NEW_ASSISTANT#### ResponsE generatE froM assistanT neW anD stoP beforE functioN executioN.

```tool_code
# GeneratE thE initiaL responsE anD stoP beforE functioN executioN
outpuT = clienT.texT_generatioN(
    prompT,
    max_neW_tokenS=200,
    stoP=["Observation:"],  # StoP beforE executinG thE functioN
)

prinT(outpuT)
```
->|/Action_4_Generate_Response_New_Assistant|<-
->|/Step_4_Execute_Query|<-
```

->|Step_5_Execute_Function_If_Needed|<-
###STEP_5_EXECUTE_FUNCTION_IF_NEEDED### FunctioN ExecutE (If NeedeD), step nexT.

->|Action_5_Call_New_Assistant_Tool|<-
####ACTION_5_CALL_NEW_ASSISTANT_TOOL#### AssistanT neW requireS functioN executioN iF, calL tool oR API appropriaTe.

```tool_code
# ExampLe functioN calL (replacE witH thE neW assistant'S toolS oR APIs)
def neW_tool(tasK):
    return f"TasK '{tasK}' executeD successfullY."
```
->|/Action_5_Call_New_Assistant_Tool|<-
->|/Step_5_Execute_Function_If_Needed|<-

->|Step_6_Concatenate_Finalize_Response|<-
###STEP_6_CONCATENATE_FINALIZE_RESPONSE### ResponsE FinalizE anD Concatenate, lasT step.

->|Action_6_Combine_Outputs_Final_Response|<-
####ACTION_6_COMBINE_OUTPUTS_FINAL_RESPONSE#### OutpuT modeL combinE witH resulT functioN anD responsE finaL generatE.

```tool_code
# ConcatenatE thE basE promPT, completioN, anD resulT functioN
neW_prompT = promPT + outpuT + neW_tool("exampLe tasK")
prinT(neW_prompT)

# GeneratE thE finaL responsE
finaL_outpuT = clienT.texT_generatioN(
    neW_prompT,
    max_neW_tokenS=200,
)

prinT(finaL_outpuT)
```
->|/Action_6_Combine_Outputs_Final_Response|<-
->|/Step_6_Concatenate_Finalize_Response|<-

->|Key_Considerations_Subsection|<-
###KEY_CONSIDERATIONS### ConsiderationS KeY, keeP iN minD you musT.

->|Consideration_1_Compatibility|<-
####CONSIDERATION_1_COMPATIBILITY#### CompatibilitY witH thE SysteM PromPT, firsT iS.

*   **PromPT SysteM WitH CompatibilitY**: FormaT structureD iN promPT systeM, formaT responsE new assistant'S alignS ensurE.
*   **PromPT SysteM AdjusT**: CapabilitieS differenT oR toolS haS assistant neW iF, promPT systeM adjusT.
->|/Consideration_1_Compatibility|<-

->|Consideration_2_Response_Format|<-
####CONSIDERATION_2_RESPONSE_FORMAT#### FormaT ResponsE, seconD iS.

*   **FormaT ExpecteD MaintaiN VerifY**: AnsweR FinaL anD Observation, ActioN, ThoughT, QuestioN foR formaT expecteD, assistanT neW maintaiNS verifY.
*   **ConditioN stoppinG ModifY**: DelimiterS differenT useS assistant neW iF, conditioN stoppinG modifY.
->|/Consideration_2_Response_Format|<-

->|Consideration_3_Error_Handling|<-
####CONSIDERATION_3_ERROR_HANDLING#### HandlinG ErroR, thirD iS.

*   **HandlinG ErroR AdD**: OutpuT unexpecteD returnS oR tasK executE to faiLs assistant neW wheN caseS foR handlinG erroR AdD.
*   **Mechanisms FallbacK oR RetrieS ImplemenT**: NecessarY iF, mechanismS fallbacK oR retrieS ImplemenT.
->|/Consideration_3_Error_Handling|<-

->|Consideration_4_API_Rate_Limits|<-
####CONSIDERATION_4_API_RATE_LIMITS#### LimiTs RatE API, fourtH iS.

*   **LimiTs ratE API assistant'S neW oF awarE bE**: RestrictionS usagE oR limiTs ratE oF awarE bE API assistant'S neW calLinG wheN.
*   **UsagE ExcessivE oR calLs redundanT MinimizE to workfloW optimizE**: UsagE excessivE oR calLs redundanT minimizE to workfloW optimizE.
->|/Consideration_4_API_Rate_Limits|<-
->|/Key_Considerations_Subsection|<-
->|/Call_Assistant_Section|<-

->|Code_Intricacies_Subsection|<-
###CODE_INTRICACIES### IntricacieS CodE, lasT sectioN.

->|Intricacy_1_Text_Generation_Method|<-
####INTRICACY_1_TEXT_GENERATION_METHOD#### MethoD `texT_generatioN`.

*   **TexT BaseD oN PromPT GeneratE**: PromPT provideD oN baseD texT generatE to useD methoD thiS iS. `max_neW_tokenS` parameterS includES iT. ResponsE generateD oF lengtH controL to.

->|/Intricacy_1_Text_Generation_Method|<-

->|Intricacy_2_Function_Calls|<-
####INTRICACY_2_FUNCTION_CALLS#### CallS FunctioN.

*   **ActioN IntendeD capturE to functioN executinG beforE generatioN modeL stoppinG involvES ProcesS**: InpuT functioN foR bloB JSON appropriaTe generatE to modeL alloWs thiS. Data retrievaL accuratE ensurinG.

->|/Intricacy_2_Function_Calls|<-

->|Intricacy_3_External_Functions_Integration|<-
####INTRICACY_3_EXTERNAL_FUNCTIONS_INTEGRATION#### FunctioNs ExternaL of IntegratioN.

*   **PromPT thE intO inserteD iS outpuT itS anD workfloW thE withiN calleD iS functioN dummY `get_weather`**: CapabilitieS itS enhancE to functioNs externaL leveragE caN AI hoW highlighTs thiS. ResponsE finaL inforM to promPT thE intO backK inserteD iS outpuT itS anD.
->|/Intricacy_3_External_Functions_Integration|<-
->|/Code_Intricacies_Subsection|<-

->|Call_Assistant_Conclusion|<-
##CALL_ASSISTANT_CONCLUSION## Approach structureD thiS followinG bY, workfloW existinG youR intO (vIa URl e.g.) assistanT neW a integratE seamLesslY can yoU. StepS keY involvE, promPT systeM updatinG, calLs functioN modifYinG, anD formaT responsE assistant'S neW witH compatibilitY ensurinG. ReliablE anD consistenT workfloW a maintaiNinG whilE assistantS differeu