let's just write a concise plan of what i want to do , for now so we have a structured approach to research

1- i want to explore the skeleton attack and the crescendo attack
2- i want to explore how to make underlying custom components for the following :
a- converters , this should be simple
b- Orchestrators (one shot , multi turn)
c- how to adjust the underlying components of attacks like Crescendo and the skeleton to run my system prompts and configurations for the possibility of making my own set of tools that would be buil on PyRit
d- how to operate the DataSet
e- how to use the Variant prompt converter which genreates prompts
f- how to use Human in the loop converter , selectiively converting option
make a plan of research for a step by step approach
pull information from the documentation only and for the info you'll pull each section will have a link to the doc you pulled the info from.

Perfect. Now I have the **authoritative** documentation. Let me build your structured research plan with **only verified documentation links**.

PyRIT Red Teaming Research Plan - Structured Approach

PHASE 1: Attack Patterns (Weeks 1-2)

1.1 Crescendo Attack (Multi-Turn)

- **What it is:** An orchestrator that uses backtracking (rewinding failed conversation turns) to maintain the victim's trust while escalating requests.
- **Core Components:**
  - `objective_target` → The victim system
  - `adversarial_chat` → The attacker LLM (brain)
  - `adversarial_chat_system_prompt_path` → Specialist persona (path to file, not string)
  - `max_turns` → Conversation limit
  - `max_backtracks` → Refusal rewind limit (unique to Crescendo)

    - `objective_scorer` → Pass/fail judge
- **Documentation:** Multi-Turn Orchestrators (CrescendoOrchestrator)

1.2 Skeleton Key Attack

- **What it is:** A jailbreak strategy that instructs the target to "ignore safety filters via a behavior update request." Not a unique orchestrator, but a **system prompt strategy** or attack pattern.
- **Implementation:** Use any orchestrator (Crescendo, RedTeaming, PAIR) with a Skeleton Key-specific `adversarial_chat_system_prompt_path`.
- **Documentation:** Confirmed in Microsoft security blog + PyRIT updates (as of June 2024). Not yet a dedicated orchestrator class—use as a persona/strategy.
- **Reference:** Microsoft Skeleton Key Mitigation

PHASE 2: Custom Components (Weeks 3-5)

2A: Custom Converters (Simplest)

- **Goal:** Transform prompts before they hit the target (Base64, emojis, leetspeak, translation, obfuscation).
- **Implementation Steps:**
    1. Inherit from `PromptConverter` abstract base class
    2. Implement `convert_async(prompt: str, **kwargs) → ConverterResult`
    3. Return a `ConverterResult` object with the transformed prompt
- **Example Pattern:**

```
from pyrit.prompt_converter import PromptConverter, ConverterResult

class CustomConverter(PromptConverter):
    async def convert_async(self, prompt: str, **kwargs) -> ConverterResult:
        # Your transformation logic
        transformed = prompt.upper()  # Example
        return ConverterResult(output=transformed, output_type="text")
```

- **Documentation:** Using Prompt Converters

2B: Custom Orchestrators (One-Shot vs. Multi-Turn)

One-Shot Orchestrator:

- **Component:** `PromptSendingOrchestrator`
- **Use Case:** Send a batch of prompts to a target once, collect results (no conversation history).
- **Key Parameters:**

    - `prompt_target` → Where prompts go
    - `prompts_list` → List of attack prompts
    - No backtracking or history management
- **Documentation:** Orchestrator Overview - PromptSendingOrchestrator

Multi-Turn Orchestrator (Custom):

- **Inheritance:** Subclass `MultiTurnOrchestrator`
- **Must Implement:**
    - `run_attack_async(objective: str, memory_labels: Optional[dict] = None) → OrchestratorResult`
    - `objective_target`, `adversarial_chat`, `max_turns` parameters
    - Conversation loop logic (attacker generates → target responds → scorer evaluates)
- **Built-in Subclasses:** `CrescendoOrchestrator`, `PairOrchestrator`, `RedTeamingOrchestrator`, `TreeOfAttacksWithPruningOrchestrator`
- **Documentation:** Orchestrators Overview - MultiTurnOrchestrator Base Class

2C: Adjusting Attack Components (Crescendo + Skeleton Key)

System Prompt Customization:

- **Parameter:** `adversarial_chat_system_prompt_path` (takes `pathlib.Path`, not string)
- **For Crescendo:**

```
orchestrator = CrescendoOrchestrator(
    objective_target=victim_llm,
    adversarial_chat=attacker_llm,
    adversarial_chat_system_prompt_path=pathlib.Path("./prompts/arabic_persona.txt"),
    # Your custom system prompt here
    max_turns=10,
    max_backtracks=5
)
```

- **For Skeleton Key:** Pass a Skeleton Key system prompt (role instruction to ignore safety).
- **Scorer Customization:** Swap `objective_scorer` (e.g., `SelfAskTrueFalseScorer` → `ContentClassifierScorer`).
- **Documentation:** Multi-Turn Orchestrators - Configuration

2D: Operating the DataSet

- **Primary Component:** `SeedPromptDataset` (loads prompts from files/memory).
- **Key Methods:**
    - `get_values()` → Retrieve all prompts
    - `get_prompt(index)` → Get specific prompt

- **Use Case:** Store jailbreak templates, then iterate through them for batch attacks.
- **Pattern:**

```
from pyrit.datasets import SeedPromptDataset

dataset = SeedPromptDataset(data_source="path/to/prompts.csv")
for prompt in dataset.get_values():
    # Use in orchestrator
```

- **Documentation:** Using Datasets

2E: Variation Converter (LLM-Powered Prompt Rewriting)

- **Component:** `VariationConverter` (uses an LLM endpoint to rewrite prompts).
- **Purpose:** Instead of fixed transformations, use an LLM to generate variations (e.g., "Make this more polite", "Translate to Arabic", "Make this subtle").
- **Requires:** An `OpenAIChatTarget` or similar chat endpoint.
- **Usage Pattern:**

```
from pyrit.prompt_converter import VariationConverter

variation = VariationConverter(
    chat_target=your_llm,
    variation_prompt="Rewrite this as a casual question"
)
converted = await variation.convert_async("Your original prompt")
```

- **Documentation:** Variation Converter

2F: Human-in-the-Loop (HITL) - Selective Conversion

- **Scorer Component:** `HumanInTheLoopScorer` (pause and manually grade outputs).
- **Converter Component:** `HumanInTheLoopConverter` (pause and manually edit prompts before sending).
- **Interactive UI Option:** `HumanInTheLoopScorerGradio` (Gradio-based web interface for manual review).
- **Workflow:**
    1. Orchestrator generates a prompt
    2. HITL Converter pauses and asks human for approval/editing
    3. Human modifies or approves
    4. Modified prompt sent to target
    5. HITL Scorer pauses to let human judge success
- **Usage Pattern:**

```
from pyrit.score import HumanInTheLoopScorer
from pyrit.prompt_converter import HumanInTheLoopConverter

hitl_converter = HumanInTheLoopConverter()
hitl_scorer = HumanInTheLoopScorer()

orchestrator = CrescendoOrchestrator(
    ...,
    prompt_converters=[hitl_converter],
    objective_scorer=hitl_scorer
)
```

- **Documentation:** Human in the Loop Scorer

PHASE 3: Integration & Tool Building (Weeks 6-8)

3.1 Building Your Custom PyRit Toolkit

- Combine custom converters + custom orchestrators + specialized system prompts
- Create a **specialist registry** (Arabic red-teamer, social engineer, technical hacker, etc.)
- Integrate datasets for batch attack campaigns
- Add HITL for sensitive/high-stakes testing

3.2 Recommended Research Order

1. **Week 1:** Read Multi-Turn Orchestrators blog + Crescendo docs (understand the pattern)
2. **Week 2:** Build 2-3 custom converters (simple, low risk)
3. **Week 3:** Build one custom one-shot orchestrator
4. **Week 4:** Build one custom multi-turn orchestrator (extend Crescendo logic)
5. **Week 5:** Experiment with Variation Converter + custom system prompts
6. **Week 6:** Integrate HITL for selective prompt editing
7. **Week 7-8:** Build your specialist toolkit (Arabic + role-based personas)

Quick Reference: Documentation Links

Topic	Link
Crescendo & Multi-Turn Overview	Multi-Turn Orchestrators
Orchestrator Base Classes	Orchestrator Overview
Custom Converters	Using Prompt Converters
Datasets	Using Datasets
Variation Converter	Variation Converter

Topic	Link
Human in Loop	Human in the Loop Scorer
Skeleton Key (Context)	Microsoft Security Blog - Skeleton Key