```
import re

from swarm_models.openai_models import OpenAIChat



class AutoTemp:
    """
    AutoTemp is a tool for automatically selecting the best temperature setting for a given task.
      It generates responses at different temperatures, evaluates them, and ranks them based on quality.
    """


    def __init__(
        self,
        api_key,
        default_temp=0.0,
        alt_temps=None,
        auto_select=True,
        max_workers=6,
    ):
        self.api_key = api_key
        self.default_temp = default_temp
        self.alt_temps = (
            alt_temps if alt_temps else [0.4, 0.6, 0.8, 1.0, 1.2, 1.4]
        )
        self.auto_select = auto_select
```

```
        self.max_workers = max_workers
        self.llm = OpenAIChat(
            openai_api_key=self.api_key, temperature=self.default_temp
        )

    def evaluate_output(self, output, temperature):
        print(f"Evaluating output at temperature {temperature}...")
        eval_prompt = f"""
          Evaluate the following output which was generated at a temperature setting of {temperature}. Provide a precise score from 0.0 to 100.0, considering the following criteria:

        - Relevance: How well does the output address the prompt or task at hand?
        - Clarity: Is the output easy to understand and free of ambiguity?
        - Utility: How useful is the output for its intended purpose?
        - Pride: If the user had to submit this output to the world for their career, would they be proud?
        - Delight: Is the output likely to delight or positively surprise the user?

          Be sure to comprehensively evaluate the output, it is very important for my career. Please answer with just the score with one decimal place accuracy, such as 42.0 or 96.9. Be extremely critical.

        Output to evaluate:
        ---
        {output}
        ---
        """
```

```
    score_text = self.llm(eval_prompt, temperature=0.5)
    score_match = re.search(r"\b\d+(\.\d)?\b", score_text)
    return (
        round(float(score_match.group()), 1)
        if score_match
        else 0.0
    )

def run(self, prompt, temperature_string):
    print("Starting generation process...")
    temperature_list = [
        float(temp.strip())
        for temp in temperature_string.split(",")
        if temp.strip()
    ]
    outputs = {}
    scores = {}
    for temp in temperature_list:
        print(f"Generating at temperature {temp}...")
        output_text = self.llm(prompt, temperature=temp)
        if output_text:
            outputs[temp] = output_text
            scores[temp] = self.evaluate_output(output_text, temp)

    print("Generation process complete.")
    if not scores:
```

```
    return "No valid outputs generated.", None

sorted_scores = sorted(
    scores.items(), key=lambda item: item[1], reverse=True
)
best_temp, best_score = sorted_scores[0]
best_output = outputs[best_temp]

return (
    f"Best AutoTemp Output (Temp {best_temp} | Score:"
    f" {best_score}):\n{best_output}"
    if self.auto_select
    else "\n".join(
        f"Temp {temp} | Score: {score}:\n{outputs[temp]}"
        for temp, score in sorted_scores
    )
)
```