```
import os

from dotenv import load_dotenv

import swarms.prompts.education as edu_prompts
from swarms import Agent, SequentialWorkflow
from swarm_models import OpenAIChat

# Load environment variables
load_dotenv()
api_key = os.getenv("OPENAI_API_KEY")
stability_api_key = os.getenv("STABILITY_API_KEY")

# Initialize language model
llm = OpenAIChat(
    openai_api_key=api_key, temperature=0.5, max_tokens=3000
)

# User preferences (can be dynamically set in a real application)
user_preferences = {
    "subjects": "Cognitive Architectures",
    "learning_style": "Visual",
    "challenge_level": "Moderate",
}

# Formatted prompts from user preferences
```

```
curriculum_prompt = edu_prompts.CURRICULUM_DESIGN_PROMPT.format(
    **user_preferences
)
interactive_prompt = edu_prompts.INTERACTIVE_LEARNING_PROMPT.format(
    **user_preferences
)
sample_prompt = edu_prompts.SAMPLE_TEST_PROMPT.format(
    **user_preferences
)
image_prompt = edu_prompts.IMAGE_GENERATION_PROMPT.format(
    **user_preferences
)

# Initialize agents for different educational tasks
curriculum_agent = Agent(llm=llm, max_loops=1, sop=curriculum_prompt)
interactive_learning_agent = Agent(
    llm=llm, max_loops=1, sop=interactive_prompt
)
sample_lesson_agent = Agent(llm=llm, max_loops=1, sop=sample_prompt)

# Create Sequential Workflow
workflow = SequentialWorkflow(max_loops=1)

# Add tasks to workflow with personalized prompts
workflow.add(curriculum_agent, "Generate a curriculum")
workflow.add(
```

```
    interactive_learning_agent, "Generate an interactive lesson"
)
workflow.add(sample_lesson_agent, "Generate a practice test")

# Execute the workflow for text-based tasks
workflow.run()

# Generate an image using Stable Diffusion
# Output results for each task
for task in workflow.tasks:
    print(
        f"Task Description: {task.description}\nResult:"
        f" {task.result}\n"
    )

# Output image result
print("Image Generation Task: Generate an image for the interactive")
```