```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler

# Constants
DAYS = 365
HOURS_PER_DAY = 24
SEQUENCE_LENGTH = 24  # For GRU, typical sequence length could be 24 hours (1 day)
MONTHS = 12
WEEKS_PER_MONTH = 5
TEMP_RANGE = {
    1: (0, 10),   # January
    2: (0, 10),   # February
    3: (5, 15),   # March
    4: (10, 20),  # April
    5: (15, 25),  # May
    6: (20, 30),  # June
    7: (20, 30),  # July
    8: (20, 30),  # August
    9: (15, 25),  # September
    10: (10, 20), # October
    11: (5, 15),  # November
    12: (0, 10)   # December
}

# List of holidays (days of the year)
holiday_dates = [
    26, 85, 89, 102, 108, 112, 144, 169, 198, 228, 240, 259, 276, 286, 305, 320, 359
]

# Generate Data
data = []

for day in range(DAYS):
    day_of_week = day % 7
    month = (day // 30) % 12 + 1
    week_of_month = (day // 7) % 5 + 1

    is_weekend = int(day_of_week in [5, 6])
    is_holiday = int(day + 1 in holiday_dates)

    temp = np.random.uniform(*TEMP_RANGE[month])

    for hour_of_day in range(HOURS_PER_DAY):
        if month in [6, 7, 8]:
            base_consumption = 120
        elif month in [12, 1, 2]:
            base_consumption = 80
        else:
            base_consumption = 100

        if hour_of_day < 6:
            consumption = base_consumption * 0.7
        elif hour_of_day >= 18:
            consumption = base_consumption * 1.2
        else:
            consumption = base_consumption

        if is_weekend:
            consumption *= 1.2

        consumption += np.random.normal(0, 5)


        data.append([
            day_of_week, hour_of_day, month, week_of_month, temp, consumption,
            is_holiday, is_weekend
        ])

columns = [
    'Day_of_Week', 'Hour_of_Day', 'Month_of_Year', 'Week_of_Month', 'Temperature',
    'Energy_Consumption','Is_Holiday', 'Is_Weekend'
]
df = pd.DataFrame(data, columns=columns)
```

```
!pip install gymnasium
!pip install stable-baselines3 matplotlib
```

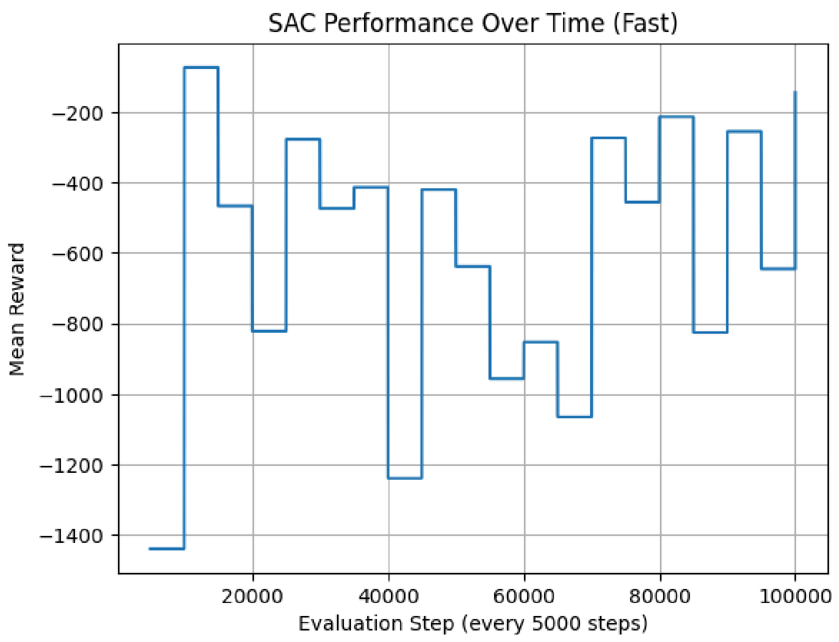
```
Collecting gymnasium
  Downloading gymnasium-0.29.1-py3-none-any.whl.metadata (10 kB)
Requirement already satisfied: numpy>=1.21.0 in /usr/local/lib/python3.10/dist-packages (from gymnasium) (1.26.4)
Requirement already satisfied: cloudpickle>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from gymnasium) (2.2.1)
Requirement already satisfied: typing-extensions>=4.3.0 in /usr/local/lib/python3.10/dist-packages (from gymnasium) (4.12.
Collecting farama-notifications>=0.0.1 (from gymnasium)
  Downloading Farama_Notifications-0.0.4-py3-none-any.whl.metadata (558 bytes)
Downloading gymnasium-0.29.1-py3-none-any.whl (953 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 953.9/953.9 kB 26.7 MB/s eta 0:00:00
Downloading Farama_Notifications-0.0.4-py3-none-any.whl (2.5 kB)
Installing collected packages: farama-notifications, gymnasium
Successfully installed farama-notifications-0.0.4 gymnasium-0.29.1
Collecting stable-baselines3
  Downloading stable_baselines3-2.3.2-py3-none-any.whl.metadata (5.1 kB)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.7.1)
Requirement already satisfied: gymnasium<0.30,>=0.28.1 in /usr/local/lib/python3.10/dist-packages (from stable-baselines3)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from stable-baselines3) (1.26.4)
Requirement already satisfied: torch>=1.13 in /usr/local/lib/python3.10/dist-packages (from stable-baselines3) (2.4.1+cu12
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.10/dist-packages (from stable-baselines3) (2.2.1)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from stable-baselines3) (2.1.4)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.3.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (10.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.1.4)
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (2.8.2)
Requirement already satisfied: typing-extensions>=4.3.0 in /usr/local/lib/python3.10/dist-packages (from gymnasium<0.30,>=
Requirement already satisfied: farama-notifications>=0.0.1 in /usr/local/lib/python3.10/dist-packages (from gymnasium<0.30
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.7->matplotlib)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch>=1.13->stable-baselines3) (
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch>=1.13->stable-baselines3) (1.1
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch>=1.13->stable-baselines3) (
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch>=1.13->stable-baselines3) (3.
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch>=1.13->stable-baselines3) (20
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->stable-baselines3) (2
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas->stable-baselines3)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.13->stabl
Requirement already satisfied: mpmath<1.4,>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from sympy->torch>=1.13->sta
Downloading stable_baselines3-2.3.2-py3-none-any.whl (182 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 182.3/182.3 kB 7.6 MB/s eta 0:00:00
Installing collected packages: stable-baselines3
Successfully installed stable-baselines3-2.3.2
```

```python
# Necessary imports
import gymnasium as gym  # Use gymnasium's gym
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from numpy.random import default_rng  # Make sure this is included for RNG

# Environment class
class TransformerEnv(gym.Env):
    def __init__(self, df):
        super(TransformerEnv, self).__init__()
        self.df = df
        self.scaler = MinMaxScaler()

        # Scale the data for more effective learning
        self.df[['Temperature', 'Energy_Consumption']] = self.scaler.fit_transform(
            self.df[['Temperature', 'Energy_Consumption']]
        )

        # Define observation and action space
        self.observation_space = gym.spaces.Box(
            low=0, high=1, shape=(6,), dtype=np.float32
        )  # Normalized state space
        self.action_space = gym.spaces.Box(
            low=-1, high=1, shape=(1,), dtype=np.float32
        )  # Action: Adjust energy consumption

        # Initialize state
        self.current_step = 0
        self.state = self._get_state()
        self.rng = None  # Random number generator

    def seed(self, seed=None):
        self.rng = default_rng(seed)  # Initialize the random number generator

    def _get_state(self):
        if self.current_step < len(self.df):
            # State is composed of Day_of_Week, Hour_of_Day, Month_of_Year, Temperature, Is_Holiday, Is_Weekend
            row = self.df.iloc[self.current_step]
            state = np.array([
                row['Day_of_Week'] / 6.0,
                row['Hour_of_Day'] / 23.0,
                row['Month_of_Year'] / 12.0,
                row['Temperature'],  # Already scaled
                row['Is_Holiday'],
                row['Is_Weekend']
            ], dtype=np.float32)  # Ensure the state is of type float32
            return state
        else:
            return np.zeros(6)  # Return an empty state when out of bounds

    def reset(self, seed=None, options=None):
        self.seed(seed)  # Seed the environment's random number generator
        self.current_step = 0
        self.state = self._get_state()
        return self.state, {}  # Gymnasium requires the reset method to return a tuple

    def step(self, action):
        # Apply the action to adjust energy consumption
        if self.current_step >= len(self.df):
            done = True
            return self.state, 0, done, False, {}  # Avoid stepping further when data is exhausted

        actual_consumption = self.df.iloc[self.current_step]['Energy_Consumption']
        adjusted_consumption = actual_consumption + action[0] * 10  # Action scaling

        # Calculate reward: inverse of the difference between adjusted and actual consumption
        reward = -np.abs(adjusted_consumption - actual_consumption)

        # Move to the next step
        self.current_step += 1
        done = self.current_step >= len(self.df)  # Check if end of data is reached
        truncated = False  # No truncation logic, so this remains False

        # Update state
        if not done:
            self.state = self._get_state()

        return self.state, reward, done, truncated, {}  # Include truncated in the return tuple

    def render(self, mode='human'):
        if self.current_step < len(self.df):
            row = self.df.iloc[self.current_step]
            print(f"Day: {row['Day_of_Week']}, Hour: {row['Hour_of_Day']}, Consumption: {row['Energy_Consumption']}, Temperature
        else:
            print("Out of data range.")
```

```
import matplotlib.pyplot as plt
from stable_baselines3 import SAC
from stable_baselines3.common.callbacks import EvalCallback
import gymnasium as gym

# Optimized Reward Callback
class FastRewardCallback(EvalCallback):
    def __init__(self, eval_env, log_path, eval_freq=5000, n_eval_episodes=5, verbose=1):
        super(FastRewardCallback, self).__init__(eval_env, best_model_save_path=log_path,
                                                 log_path=log_path, eval_freq=eval_freq,
                                                 n_eval_episodes=n_eval_episodes, verbose=verbose)
        self.episode_rewards = []

    def _on_step(self) -> bool:
        result = super()._on_step()
        # Collect the mean reward at each evaluation step
        self.episode_rewards.append(self.last_mean_reward)
        return result

# Create the environment
env = TransformerEnv(df)  # Your custom environment
eval_env = TransformerEnv(df)  # Evaluation environment

# Create SAC model with fast training settings
model = SAC("MlpPolicy", env, verbose=1, learning_rate=1e-3, batch_size=128, train_freq=4)

# Fast callback: eval every 5000 steps, fewer episodes
fast_reward_callback = FastRewardCallback(eval_env, log_path="./logs", eval_freq=5000, n_eval_episodes=5)

# Train the model with fewer timesteps for faster execution
model.learn(total_timesteps=100000, callback=fast_reward_callback)
```

```
Using cuda device
Wrapping the env with a `Monitor` wrapper
Wrapping the env in a DummyVecEnv.
/usr/local/lib/python3.10/dist-packages/stable_baselines3/common/evaluation.py:67: UserWarning: Evaluation environment
  warnings.warn(
Eval num_timesteps=5000, episode_reward=-1438.78 +/- 0.00
Episode length: 8760.00 +/- 0.00
---------------------------------
| eval/              |          |
|    mean_ep_length  | 8.76e+03 |
|    mean_reward     | -1.44e+03 |
| time/              |          |
|    total_timesteps | 5000     |
| train/             |          |
|    actor_loss      | 10.9     |
|    critic_loss     | 0.0635   |
|    ent_coef        | 0.699    |
|    ent_coef_loss   | 0.0053   |
|    learning_rate   | 0.001    |
|    n_updates       | 1224     |
---------------------------------
New best mean reward!
Eval num_timesteps=10000, episode_reward=-73.59 +/- 0.00
Episode length: 8760.00 +/- 0.00
--------------------------------
| eval/              |          |
|    mean_ep_length  | 8.76e+03 |
|    mean_reward     | -73.6    |
| time/              |          |
|    total_timesteps | 10000    |
| train/             |          |
|    actor_loss      | 19.5     |
|    critic_loss     | 0.055    |
|    ent_coef        | 0.715    |
|    ent_coef_loss   | 0.0475   |
|    learning_rate   | 0.001    |
|    n_updates       | 2474     |
--------------------------------
New best mean reward!
Eval num_timesteps=15000, episode_reward=-466.12 +/- 0.00
Episode length: 8760.00 +/- 0.00
--------------------------------
| eval/              |          |
|    mean_ep_length  | 8.76e+03 |
|    mean_reward     | -466     |
| time/              |          |
|    total_timesteps | 15000    |
| train/             |          |
|    actor_loss      | 26.9     |
|    critic_loss     | 0.0363   |
|    ent_coef        | 0.717    |
|    ent_coef_loss   | 0.063    |
|    learning_rate   | 0.001    |
|    n_updates       | 3724     |
--------------------------------
Eval num_timesteps=20000, episode_reward=-821.28 +/- 0.00
Episode length: 8760.00 +/- 0.00
```

```
# Plot rewards
plt.plot(fast_reward_callback.episode_rewards)
plt.xlabel('Evaluation Step (every 5000 steps)')
plt.ylabel('Mean Reward')
plt.title('SAC Performance Over Time (Fast)')
plt.grid(True)
plt.show()
```



```
def evaluate_and_render(env, model, num_episodes=10):
    for episode in range(num_episodes):
        obs, _ = env.reset()  # Extract only the observation
        done = False
        total_reward = 0

        print(f"Episode {episode + 1}:")
        while not done:
            action, _ = model.predict(obs, deterministic=True)
            obs, reward, done, _, _ = env.step(action)  # Handle the additional elements returned by step()

            total_reward += reward

            # Render the current state (if useful)
            if env.current_step < len(env.df):
                env.render()

        print(f"Total Reward for Episode {episode + 1}: {total_reward}")


evaluate_and_render(env, model, num_episodes=10)
```

```
Streaming output truncated to the last 5000 lines.
Day: 2.0, Hour: 17.0, Consumption: 0.5299765626108851, Temperature: 0.6924403516037891
Day: 2.0, Hour: 18.0, Consumption: 0.7998441221686055, Temperature: 0.6924403516037891
Day: 2.0, Hour: 19.0, Consumption: 0.6360275674172047, Temperature: 0.6924403516037891
Day: 2.0, Hour: 20.0, Consumption: 0.7462953060785695, Temperature: 0.6924403516037891
Day: 2.0, Hour: 21.0, Consumption: 0.6825261796233231, Temperature: 0.6924403516037891
Day: 2.0, Hour: 22.0, Consumption: 0.7248089569471676, Temperature: 0.6924403516037891
Day: 2.0, Hour: 23.0, Consumption: 0.7049710950805947, Temperature: 0.6924403516037891
Day: 3.0, Hour: 0.0, Consumption: 0.30729978132806124, Temperature: 0.7536625858669002
Day: 3.0, Hour: 1.0, Consumption: 0.30882382988719337, Temperature: 0.7536625858669002
Day: 3.0, Hour: 2.0, Consumption: 0.32030897308924966, Temperature: 0.7536625858669002
Day: 3.0, Hour: 3.0, Consumption: 0.3798749740818732, Temperature: 0.7536625858669002
Day: 3.0, Hour: 4.0, Consumption: 0.23831440656470249, Temperature: 0.7536625858669002
Day: 3.0, Hour: 5.0, Consumption: 0.34050105121739993, Temperature: 0.7536625858669002
Day: 3.0, Hour: 6.0, Consumption: 0.5725950066592056, Temperature: 0.7536625858669002
Day: 3.0, Hour: 7.0, Consumption: 0.5548209537244938, Temperature: 0.7536625858669002
Day: 3.0, Hour: 8.0, Consumption: 0.5059913041021743, Temperature: 0.7536625858669002
Day: 3.0, Hour: 9.0, Consumption: 0.5630435618287574, Temperature: 0.7536625858669002
Day: 3.0, Hour: 10.0, Consumption: 0.5057561933912309, Temperature: 0.7536625858669002
Day: 3.0, Hour: 11.0, Consumption: 0.5820188989435306, Temperature: 0.7536625858669002
Day: 3.0, Hour: 12.0, Consumption: 0.6142839562390675, Temperature: 0.7536625858669002
Day: 3.0, Hour: 13.0, Consumption: 0.594944091548081, Temperature: 0.7536625858669002
```

```
Day: 3.0, Hour: 14.0, Consumption: 0.5586597340681381, Temperature: 0.7536625858669002
Day: 3.0, Hour: 15.0, Consumption: 0.5982105624332169, Temperature: 0.7536625858669002
Day: 3.0, Hour: 16.0, Consumption: 0.546932282665981, Temperature: 0.7536625858669002
Day: 3.0, Hour: 17.0, Consumption: 0.5607218367257133, Temperature: 0.7536625858669002
Day: 3.0, Hour: 18.0, Consumption: 0.7053442536384886, Temperature: 0.7536625858669002
Day: 3.0, Hour: 19.0, Consumption: 0.6661244834653195, Temperature: 0.7536625858669002
Day: 3.0, Hour: 20.0, Consumption: 0.750578606905465, Temperature: 0.7536625858669002
Day: 3.0, Hour: 21.0, Consumption: 0.7372815317588264, Temperature: 0.7536625858669002
Day: 3.0, Hour: 22.0, Consumption: 0.687108817877012, Temperature: 0.7536625858669002
Day: 3.0, Hour: 23.0, Consumption: 0.7060331266002738, Temperature: 0.7536625858669002
Day: 4.0, Hour: 0.0, Consumption: 0.3313346472754132, Temperature: 0.7282465696757568
Day: 4.0, Hour: 1.0, Consumption: 0.3093303983033857, Temperature: 0.7282465696757568
Day: 4.0, Hour: 2.0, Consumption: 0.3612669706120881, Temperature: 0.7282465696757568
Day: 4.0, Hour: 3.0, Consumption: 0.30743406318594013, Temperature: 0.7282465696757568
Day: 4.0, Hour: 4.0, Consumption: 0.30223732572823525, Temperature: 0.7282465696757568
Day: 4.0, Hour: 5.0, Consumption: 0.29503646483060775, Temperature: 0.7282465696757568
Day: 4.0, Hour: 6.0, Consumption: 0.5619641745130785, Temperature: 0.7282465696757568
Day: 4.0, Hour: 7.0, Consumption: 0.6671993010233509, Temperature: 0.7282465696757568
Day: 4.0, Hour: 8.0, Consumption: 0.5441296053396798, Temperature: 0.7282465696757568
Day: 4.0, Hour: 9.0, Consumption: 0.505307207746325, Temperature: 0.7282465696757568
Day: 4.0, Hour: 10.0, Consumption: 0.6039829998323932, Temperature: 0.7282465696757568
Day: 4.0, Hour: 11.0, Consumption: 0.5388163683302415, Temperature: 0.7282465696757568
Day: 4.0, Hour: 12.0, Consumption: 0.5564328595731398, Temperature: 0.7282465696757568
Day: 4.0, Hour: 13.0, Consumption: 0.5379707994516856, Temperature: 0.7282465696757568
Day: 4.0, Hour: 14.0, Consumption: 0.5541671818561434, Temperature: 0.7282465696757568
Day: 4.0, Hour: 15.0, Consumption: 0.5386758096264167, Temperature: 0.7282465696757568
Day: 4.0, Hour: 16.0, Consumption: 0.6083679092049514, Temperature: 0.7282465696757568
Day: 4.0, Hour: 17.0, Consumption: 0.5165717789735227, Temperature: 0.7282465696757568
Day: 4.0, Hour: 18.0, Consumption: 0.6870915473866079, Temperature: 0.7282465696757568
Day: 4.0, Hour: 19.0, Consumption: 0.6890543708203962, Temperature: 0.7282465696757568
Day: 4.0, Hour: 20.0, Consumption: 0.7173752334801717, Temperature: 0.7282465696757568
Day: 4.0, Hour: 21.0, Consumption: 0.7139278258979267, Temperature: 0.7282465696757568
Day: 4.0, Hour: 22.0, Consumption: 0.6699548827388273, Temperature: 0.7282465696757568
Day: 4.0, Hour: 23.0, Consumption: 0.6997848084682788, Temperature: 0.7282465696757568
Day: 5.0, Hour: 0.0, Consumption: 0.43781326927135855, Temperature: 0.9428203270715642
```