以下の手順を最初から最後まで全部自動で実行してください。途中で止まらず、全て完了してから報告してください。日本語で応答してください。

【重要ルール】

- どのステップでエラーが出ても、自分で原因を分析して修正し、そのステップをやり直してから次に進んでください。絶対に途中で止まらないでください。

- Pythonを実行するときは必ず ./venv/bin/python3 または ./venv/bin/pip を使ってください（venvのフルパス指定）。

- brewのPATHは /opt/homebrew/bin/brew を使ってください。

- 全ステップ完了後に「環境構築が完了しました」と報告してください。途中報告は不要です。

■ 1. Python 3.13のインストール

/opt/homebrew/bin/python3.13 --version で確認。なければ以下でインストール:

/opt/homebrew/bin/brew install python@3.13

※Python 3.14はSSLの互換性問題があるため、必ず3.13を使ってください。

■ 2. Python仮想環境の作成とライブラリインストール

/opt/homebrew/bin/python3.13 -m venv venv

./venv/bin/pip install --upgrade pip

./venv/bin/pip install tweepy requests

インストール後、以下でインポート確認:

./venv/bin/python3 -c "import tweepy; import requests; print('OK')"

■ 3. 以下のファイルを全てそのまま作成してください。コードは一切変更しないでください。

【config.py】

"""

```
X自動投稿パイプライン - 設定ファイル

"""



X_API_KEY = "ここにX_API_KEYを貼る"

X_API_SECRET = "ここにX_API_SECRETを貼る"

X_ACCESS_TOKEN = "ここにX_ACCESS_TOKENを貼る"

X_ACCESS_TOKEN_SECRET = "ここにX_ACCESS_TOKEN_SECRETを貼る
"



AI_PROVIDER = "anthropic"

ANTHROPIC_API_KEY = "ここにANTHROPIC_API_KEYを貼る"



POSTS_PER_DAY = 3



POST_SCHEDULE = [

    "08:00",
```

```
    "12:30",
    "19:00",
]

ACCOUNT_PROFILE = {
    "handle": "@あなたのアカウント名",
    "genre": "あなたのジャンル",
    "tone": "断言・言い切り",
    "target": "あなたのターゲット層",
    "credential": "あなたの実績をここに書く",
    "positioning": "あなたのポジショニング",
    "goal_30day": "30日間の目標",
    "ng_expressions": "使ってはいけない表現をカンマ区切り",
    "rules": "守るべきルールを列挙",
    "category_weights": "ノウハウ・Tips:30, 実績・証拠:20, 煽り・問題提起:20, ツール紹介:15, マインドセット:15",
```

```
    "top_patterns": "伸びた投稿の共通パターン",
}
```

【config.py ここまで】

【x_poster.py】

```
import tweepy
from config import (
    X_API_KEY,
    X_API_SECRET,
    X_ACCESS_TOKEN,
    X_ACCESS_TOKEN_SECRET,
)


def get_client():
```

```
    client = tweepy.Client(
        consumer_key=X_API_KEY,
        consumer_secret=X_API_SECRET,
        access_token=X_ACCESS_TOKEN,
        access_token_secret=X_ACCESS_TOKEN_SECRET,
    )
    return client



def post_tweet(text):
    try:
        client = get_client()
        response = client.create_tweet(text=text)
        tweet_id = response.data["id"]
        me = client.get_me()
        handle = me.data.username
```

```
        print(f"投稿成功: https://x.com/{handle}/status/{tweet_id}")
        return {"success": True, "tweet_id": tweet_id, "error": None}
    except tweepy.TooManyRequests:
        print("レート制限に達しました。15分後に再試行してください。")
        return {"success": False, "tweet_id": None, "error": "rate_limit"}
    except tweepy.Forbidden as e:
        print(f"投稿が拒否されました: {e}")
        return {"success": False, "tweet_id": None, "error": f"forbidden: {e}"}
    except Exception as e:
        print(f"エラー: {e}")
        return {"success": False, "tweet_id": None, "error": str(e)}


if __name__ == "__main__":
    print("=" * 50)
    print("X API 接続テスト")
```

```
    print("=" * 50)

    try:

        client = get_client()

        me = client.get_me()

        print(f"認証成功: @{me.data.username} ({me.data.name})")

    except Exception as e:

        print(f"認証失敗: {e}")

        print("  config.py のAPIキーを確認してください")
```

【x_poster.py ここまで】

【prompts.py】

```
import random

from config import ACCOUNT_PROFILE


P = ACCOUNT_PROFILE
```

```
POST_CATEGORIES = {
    "実績・証拠": {"weight": 20, "time_slots": ["12:30", "19:00"]},
    "ノウハウ・Tips": {"weight": 30, "time_slots": ["08:00", "19:00"]},
    "煽り・問題提起": {"weight": 20, "time_slots": ["12:30", "19:00"]},
    "ツール紹介": {"weight": 15, "time_slots": ["08:00", "19:00"]},
    "マインドセット": {"weight": 15, "time_slots": ["08:00", "19:00"]},
}

if P.get("category_weights"):
    for pair in P["category_weights"].split(","):
        pair = pair.strip()
        if ":" in pair:
            name, w = pair.rsplit(":", 1)
            name = name.strip()
            w = int(w.strip())
```

```
        for key in POST_CATEGORIES:
            if name in key or key in name:
                POST_CATEGORIES[key]["weight"] = w
                break
```

```
SYSTEM_PROMPT = f"""あなたは{P['handle']}として投稿するゴーストライターだ。

【ジャンル】{P['genre']}
【口調】{P['tone']}
【ターゲット】{P['target']}
【ポジショニング】{P['positioning']}
【実績】{P['credential']}

【文体ルール】
・140字以内で完結させろ。途中で切れる文章は絶対禁止。
```

・改行を効果的に使え。

・絵文字は最小限。

・ハッシュタグ禁止。URL禁止。★●■などの装飾記号禁止。

・同じ内容・同じフレーズの繰り返し禁止。

【NG表現】{P['ng_expressions']}

【ルール】{P['rules']}

【伸びるパターン】{P['top_patterns']}

投稿文のみを出力しろ。説明や補足は一切不要。"""

CATEGORY_PROMPTS = {

"実績・証拠": f"""{P['handle']}の実績（{P['credential']}）を自然に織り交ぜた投稿を作れ。

自慢にならず、事実ベースで権威性を出せ。

{{phase_context}}Day{{day_number}}の投稿だ。

140字以内で完結する投稿文を出力しろ。途中で切れる文章は禁止。投稿文のみを出力。""",

"ノウハウ・Tips": f"""{P['genre']}に関する実用的なノウハウやTipsの投稿を作れ。

ターゲット（{P['target']}）がすぐ使える内容にしろ。

{{phase_context}}Day{{day_number}}の投稿だ。

140字以内で完結する投稿文を出力しろ。途中で切れる文章は禁止。投稿文のみを出力。""",

"煽り・問題提起": f"""{P['genre']}の領域で問題提起する投稿を作れ。

「まだやってないの？」「知らないと損する」系の角度で、ターゲットの行動を促せ。

{{phase_context}}Day{{day_number}}の投稿だ。

140字以内で完結する投稿文を出力しろ。途中で切れる文章は禁止。投稿文のみを出力。""",

"ツール紹介": f"""{P['genre']}で使えるAIツールや機能を紹介する投稿を作れ。

具体的なツール名を入れて、{P['positioning']}の視点で語れ。

{{phase_context}}Day{{day_number}}の投稿だ。

140字以内で完結する投稿文を出力しろ。途中で切れる文章は禁止。投稿文のみを出力。""",

"マインドセット": f"""{P['genre']}に取り組む上でのマインドセットを語れ。

{P['handle']}のスタンス（{P['positioning']}）を滲ませろ。

{{phase_context}}Day{{day_number}}の投稿だ。

140字以内で完結する投稿文を出力しろ。途中で切れる文章は禁止。投稿文のみを出力。""",

}

def get_phase_context(day_number):

if day_number <= 7:

return "【フェーズ：価値提供期】フォロワーに役立つ情報を全力で出す時期。"

```
    elif day_number <= 14:
        return "【フェーズ：権威性構築期】実績や数字を見せて信頼を積む時期。"
    elif day_number <= 21:
        return "【フェーズ：期待感醸成期】次のコンテンツへの期待を高める時期。"
    else:
        return "【フェーズ：ローンチ期】行動喚起を強めて成果に繋げる時期。"



def select_category_for_slot(time_slot, day_number):
    available = []
    weights = []
    for cat, info in POST_CATEGORIES.items():
        if time_slot in info["time_slots"]:
            available.append(cat)
            weights.append(info["weight"])
    if not available:
```

```
        available = list(POST_CATEGORIES.keys())

        weights = [info["weight"] for info in POST_CATEGORIES.values()]

    return random.choices(available, weights=weights, k=1)[0]




def build_prompt(category, day_number):

    phase_context = get_phase_context(day_number)

    template = CATEGORY_PROMPTS.get(category,
CATEGORY_PROMPTS["ノウハウ・Tips"])

    return template.format(day_number=day_number,
phase_context=phase_context)
```

【prompts.py ここまで】

【post_generator.py】

```
import json, subprocess, tempfile, os, time

from config import ANTHROPIC_API_KEY
```

```
from prompts import SYSTEM_PROMPT, build_prompt,
select_category_for_slot




def generate_post_anthropic(category, day_number):
    user_prompt = build_prompt(category, day_number)
    payload = {
        "model": "claude-sonnet-4-20250514",
        "max_tokens": 300,
        "system": SYSTEM_PROMPT,
        "messages": [{"role": "user", "content": user_prompt}],
    }
    tmp = tempfile.NamedTemporaryFile(
        mode="w", suffix=".json", delete=False, encoding="utf-8"
    )
    json.dump(payload, tmp, ensure_ascii=False)
```

```
    tmp.close()
    try:
        cmd = (
            f'/usr/bin/curl -s https://api.anthropic.com/v1/messages '
            f'-H "x-api-key: {ANTHROPIC_API_KEY}" '
            f'-H "anthropic-version: 2023-06-01" '
            f'-H "content-type: application/json" '
            f'-d @{tmp.name}'
        )
        r = subprocess.run(cmd, shell=True, capture_output=True, text=True,
timeout=120)
        if not r.stdout.strip():
            return None
        data = json.loads(r.stdout)
        if "content" in data:
            return data["content"][0]["text"].strip()
```

```
        return None
    except:
        return None
    finally:
        os.unlink(tmp.name)


def is_complete_post(text):
    if not text or text == "投稿生成に失敗":
        return False
    if len(text) > 140:
        return False
    bad_endings = [
        "完", "自", "他", "読ん", "書いて", "感動",
        "こ", "プ", "なりた", "人にな", "仕組み作って",
    ]
```

```
    for be in bad_endings:
        if text.endswith(be):
            return False
    last_char = text[-1]
    good_endings = list(
        "るただいくすぞなよねかれんむえうろめてにをはもがぜさけせつみりし？！…。らので"
    )
    if last_char in good_endings:
        return True
    return False




def generate_post(category, day_number):
    text = None
    for attempt in range(5):
```

```
    text = generate_post_anthropic(category, day_number)
    if text and is_complete_post(text):
        return text
    if text and len(text) > 140:
        print(f"  {len(text)}字 再生成({attempt+1}/5)")
        time.sleep(1)
if text and len(text) > 140:
    lines = text.split("\n")
    result = ""
    for line in lines:
        if len(result + line + "\n") <= 140:
            result += line + "\n"
        else:
            break
    return result.strip() if result.strip() else text[:140]
return text if text else "投稿生成に失敗"
```

```
def generate_daily_posts(day_number, time_slots):
    posts = []
    used_categories = []
    for slot in time_slots:
        for _ in range(10):
            category = select_category_for_slot(slot, day_number)
            if category not in used_categories or len(used_categories) >= 4:
                break
        used_categories.append(category)
        text = generate_post(category, day_number)
        posts.append({
            "time": slot,
            "category": category,
            "text": text,
```

```
            "day": day_number,
        })
        print(f"  [{slot}] {category} ({len(text)}字): {text[:50]}…")
    return posts


def generate_batch_posts(start_day, end_day, time_slots):
    all_posts = []
    for day in range(start_day, end_day + 1):
        print(f"Day {day} の投稿を生成中…")
        daily = generate_daily_posts(day, time_slots)
        all_posts.extend(daily)
    return all_posts


def save_posts_to_json(posts, filepath="post_queue.json"):
```

```
    with open(filepath, "w", encoding="utf-8") as f:
        json.dump(posts, f, ensure_ascii=False, indent=2)
    print(f"{len(posts)}件の投稿を {filepath} に保存")



def load_posts_from_json(filepath="post_queue.json"):
    with open(filepath, "r", encoding="utf-8") as f:
        return json.load(f)
```

【post_generator.py ここまで】

■ 4. config.pyの設定

全ファイルの作成が完了したら、以下を行ってください：

A. ~/Downloads/ フォルダから account_profile.py を探して読み込み、その中の ACCOUNT_PROFILE の内容で config.py の ACCOUNT_PROFILE を上書きしてください。見つからない場合はユーザーに場所を聞いてください。

B. 上書きが完了したら、ユーザーに以下のように案内してください：

「環境構築が完了しました。AI人格プロファイルも自動で読み込みました。

あと必要なのはAPIキーだけです。

以下の5つをこのチャットに貼り付けてください：

1. X API Key

2. X API Key Secret

3. X Access Token

4. X Access Token Secret

5. Anthropic API Key

全部まとめて貼り付けてOKです。自動でconfig.pyに書き込みます。」

C. ユーザーがAPIキーを貼り付けたら、config.pyの該当箇所を自動で書き換えてください。

D. 書き換え後、./venv/bin/python3 x_poster.py で接続テストを実行し、結果を報告してください。