TP Python – Météo avancée et génération d'un QR Code

Contexte

Ce TP fait suite au précédent travail sur la récupération de données météo avec l'API **Open-Meteo** en Python.

Les notions suivantes sont **considérées comme acquises** :

- appel à une API avec `requests`
- structure d'une réponse JSON
- extraction de données simples
- génération basique d'un QR code

Dans ce TP, vous allez organiser un **projet Python complet**, exploiter des **prévisions horaires**, traduire des **codes météo**, et générer un **QR Code météo** lisible sur smartphone.

Objectifs

- organiser un projet Python en plusieurs fichiers
- factoriser et configurer des paramètres (ville, coordonnées, API)
- exploiter des données météo actuelles et horaires
- traduire des codes météo numériques
- gérer les erreurs lors d'appels à une API
- générer un QR code contenant des informations dynamiques

Prérequis techniques

- Python 3
- Environnement : **Thonny**
- Bibliothèques Python :
  - `requests`
  - `qrcode`
  - `pillow`

> Si nécessaire :

```
pip install requests qrcode[pil] pillow
```

Création de l'arborescence du TP (à créer avec un script Python)

Organisation du TP

Le travail est découpé en plusieurs scripts, chacun ayant un rôle précis :

Fichier	Rôle
`meteo_actuelle.py`	Requête météo actuelle
`meteo_actuelle_V2.py`	Version configurable
`meteo_code.py`	Traduction des codes météo
`meteo_previsions.py`	Prévisions horaires
`meteo_qrcode.py`	QR code météo final

Arborescence attendue

```
📂 TP_Meteo_Python/
│
├── meteo_actuelle.py
├── meteo_actuelle_V2.py
├── meteo_code.py
├── meteo_previsions.py
├── meteo_qrcode.py
│
├── 📁 images/
│
├── README.md
└── 1_TP_meteo.pdf          # A copier-coller ici
```

Script de création automatique de l'arborescence - `create_project.py`

> Compléter les parties `TODO` afin de créer automatiquement l'arborescence du projet.

```python
import os

BASE_DIR = "TP_Meteo_Python"

# TODO 1 : compléter la liste des fichiers à créer
files = [
    # "meteo_actuelle.py",
    # ...
]

# TODO 2 : compléter la liste des dossiers à créer
folders = [
    # "images"
]

# TODO 3 : créer le dossier principal

# TODO 4 : créer les sous-dossiers
```

```
# TODO 5 : créer les fichiers vides
```

---

1. Météo actuelle – `meteo_actuelle.py`

1.1. Objectif

Récupérer la météo actuelle pour une localisation donnée et afficher la réponse complète de l'API.

1.2. Localisation utilisée

- Ville : **Rodez**
- Latitude : `44.35258`
- Longitude : `2.57338`

1.3. Travail demandé

Compléter le fichier `meteo_actuelle.py` :

> *Les coordonnées doivent être complétées avant exécution.*

```
import requests

BASE_URL = "https://api.open-meteo.com/v1/forecast"

lat =      # latitude de Rodez
lon =      # longitude de Rodez

params = {
    "latitude": lat,
    "longitude": lon,
    "current_weather": True,
    "timezone": "Europe/Paris"
}

response = requests.get(BASE_URL, params=params)

if response.status_code != 200:
    print("Erreur lors de l'appel à l'API météo")
    exit()

response = response.json()
print(response)
```

Question

Quels champs sont présents dans `current_weather` ?

2. Extraction et affichage des données météo actuelles

2.1. Travail demandé

À partir de la réponse JSON :

1. Extraire :
    - la température
    - la vitesse du vent
    - la direction du vent
2. Afficher ces informations sous une forme lisible

2.2. Lever et coucher du soleil

- Lien utile (à ouvrir avec le navigateur Firefox) : [text](https://api.open-meteo.com/v1/forecast?latitude=44.35258&longitude=2.57338&current_weather=true&daily=sunrise,sunset&timezone=auto)

Modifier la requête API afin d'obtenir :

- l'heure du lever du soleil
- l'heure du coucher du soleil

```
    "daily": ["sunrise", "sunset"]
```

3. Version configurable – `meteo_actuelle_V2.py`

3.1. Objectif

Cette version vise à rendre le programme plus **lisible**, **modifiable** et **réutilisable** pour une autre ville.

3.2. Travail demandé

1. Créer une copie du fichier précédent nommée `meteo_actuelle_V2.py`
2. Ajouter une **section de configuration** contenant :
    - le nom de la ville
    - la latitude
    - la longitude
    - le fuseau horaire
    - l'URL de l'API
3. Modifier le programme pour utiliser ces variables

Solution

```
import requests
import datetime as dt
```

```
# Configuration
# Localisation
CITY = "Rodez"
LAT = 44.35258
LON = 2.57338
TIMEZONE = "Europe/Paris"

# API Open-Meteo
BASE_URL = "https://api.open-meteo.com/v1/forecast"

params = {
    "latitude": LAT,
    "longitude": LON,
    "current_weather": True,
    "daily": ["sunrise", "sunset"],
    "timezone": TIMEZONE
}

response = requests.get(BASE_URL, params=params)

if response.status_code != 200:
    print("Erreur lors de l'appel à l'API météo")
    exit()

data = response.json()

current = data["current_weather"]
daily = data["daily"]

# Formatage de l'heure actuelle
heure = dt.datetime.fromisoformat(current["time"]).strftime("%H:%M")

temperature = current["temperature"]          # °C
vitesse_vent = current["windspeed"]           # km/h
direction_vent = current["winddirection"]     # degrés
weathercode = current["weathercode"]

lever_soleil = daily["sunrise"][0]
coucher_soleil = daily["sunset"][0]

print(f"\n--- Météo à {CITY} ---")
print(f"Heure actuelle : {heure}")
print(f"Température : {temperature:.1f}°C")
print(f"Vitesse du vent : {vitesse_vent:.1f} km/h")
print(f"Direction du vent : {direction_vent}°")
print(f"Lever du soleil : {lever_soleil}")
print(f"Coucher du soleil : {coucher_soleil}")
```

4. Traduction du weathercode – `meteo_code.py`

Comme vu lors des TP sur la météo en Javascript, l'API fournit un code météo numérique.

4.1. Objectif

Associer un **code météo numérique** à une description textuelle.

4.2. Travail demandé

1. Créer le fichier `meteo_code.py` et compléter le dictionnaire suivant :
2. Tester l'affichage d'une description météo

```
weather_codes = {
    0: "Ciel dégagé",
    1: "TODO : à compléter ......",
    2: "TODO : à compléter ......",
    3: "TODO : à compléter ......",
    61: "Pluie faible",
    63: "TODO : à compléter ......",
    65: "TODO : à compléter ......",
    71: "TODO : à compléter ......",
    73: "TODO : à compléter ......",
    75: "Neige forte",
    80: "TODO : à compléter ......",
    81: "Averses modérées",
    82: "TODO : à compléter ......",
    95: "Orage"
}

# TODO : tester avec d'autres valeurs de weathercode
weathercode = 61  # valeur de test

description = weather_codes.get(
    weathercode,
    f"Temps inconnu (code {weathercode})"
)

print(f"Conditions météo : {description}")
```

4.3. Travail demandé

- Ajouter de nouveaux codes météo
- Tester avec différentes valeurs de `weathercode`

5. Prévisions horaires – `meteo_previsions.py`

5.1. Objectif

Exploiter les **prévisions horaires** fournies sous forme de **listes synchronisées**.

```
import requests

LAT = # à compléter
```

```
LON = # à compléter
TIMEZONE = "Europe/Paris"
BASE_URL = "https://api.open-meteo.com/v1/forecast"

params = {
    "latitude": LAT,
    "longitude": LON,
    "hourly": [
        "temperature_2m",
        "precipitation",
        "windspeed_10m"
    ],
    "forecast_days": 1,
    "timezone": TIMEZONE
}

response = requests.get(BASE_URL, params=params)

if response.status_code != 200:
    print("Erreur lors de l'appel à l'API météo")
    exit()

forecast = response.json()

# Les données horaires sont fournies sous forme de listes
# times  → [heure1, heure2, heure3, ...]
# temperatures  → [temp1, temp2, temp3, ...]
times = forecast["hourly"]["time"][:6]
temperatures = forecast["hourly"]["temperature_2m"][:6]
precipitations = forecast["hourly"]["precipitation"][:6]

print("Prévisions – prochaines 6 heures\n")

for t, temp, p in zip(times, temperatures, precipitations):
    print(f"{t} → {temp}°C | pluie : {p} mm")
```

Questions

- Pourquoi ces données sont-elles stockées dans des listes ?
- À quoi correspond l'indice `[0]` ?

---

6. Génération d'un QR Code météo – `meteo_qrcode.py`

6.1. Objectif

Synthétiser l'ensemble du travail dans un **QR Code météo complet**.

Le QR Code contiendra :

- la météo actuelle
- les prévisions des prochaines heures

- la date et la ville

```
import os
import requests
import qrcode
from datetime import datetime
from meteo_code import weather_codes

os.makedirs("images", exist_ok=True)

# Configuration
CITY = "Rodez"
LAT = 44.35258
LON = 2.57338
TIMEZONE = "Europe/Paris"
BASE_URL = "https://api.open-meteo.com/v1/forecast"

# dictionnaire weather_codes importer depuis `meteo_code.py` : from
meteo_code import weather_codes

# 1. Récupération de la météo actuelle
params_current = {
    "latitude": LAT,
    "longitude": LON,
    "current_weather": True,
    "timezone": TIMEZONE
}

current = requests.get(BASE_URL, params=params_current).json()
["current_weather"]

# 2. Récupération des prévisions horaires
params_forecast = {
    "latitude": LAT,
    "longitude": LON,
    "hourly": ["temperature_2m", "precipitation", "windspeed_10m"],
    "forecast_days": 1,
    "timezone": TIMEZONE
}

forecast = requests.get(BASE_URL, params=params_forecast).json()

# 3. Extraction des données actuelles
temperature = current["temperature"]
vitesse_vent = current["windspeed"]
weathercode = current["weathercode"]
description = weather_codes.get(weathercode, f"Temps inconnu (code
{weathercode})")

# Prévisions limitées (ex : 6 prochaines heures)
times = forecast["hourly"]["time"][:6]
temperatures = forecast["hourly"]["temperature_2m"][:6]
precipitations = forecast["hourly"]["precipitation"][:6]
```

```
# 4. Date actuelle
date = datetime.now().strftime("%d/%m/%Y %H:%M")

# 5. Construction du texte du QR code
# TODO: Partie à compléter
qr_text = f"""Météo - {CITY}
Date : ...

Température : ...
Vent : ...
Temps : ...

Prévisions :
"""

for t, temp, p in zip(times, temperatures, precipitations):
    heure = t[11:16]
    qr_text += f"\n{heure} → {temp}°C | pluie : {p} mm"

# 6. Génération et sauvegarde du QR code
qr = qrcode.QRCode(
    version=2,  # taille du QR (1 à 40)
    error_correction=qrcode.constants.ERROR_CORRECT_Q,
    box_size=10,
    border=4,
)

qr.add_data(qr_text)
qr.make(fit=True)

img = qr.make_image(fill_color="black", back_color="white")
img.save("images/meteo_rodez_qr.png")

print(f"QR code généré : images/meteo_rodez_qr.png")
print(f"\nContenu du QR code :\n{qr_text}")
```

6.2. Travail demandé

- Limiter les prévisions à **3 heures**
- Ajouter la **date**
- Ajouter le **nom de la ville**
- Ajouter une ligne :

  ```
  Bonne journée ☀
  ```

- Tester le QR code avec un smartphone

Question

- Quelle différence voyez-vous entre l'utilisation en Python et en JavaScript ?

---

7. Travail attendu

- Tous les scripts fonctionnent
- Le QR code est généré correctement et lisible après scan
- Le code est structuré et commenté

---

8. Checklist

8.1. Avant de tester :

- [ ] Installer les dépendances : `pip install requests qrcode[pil] pillow`
- [ ] Créer l'arborescence avec `create_project.py`
- [ ] Vérifier la connexion Internet

8.2. Tests à effectuer :

- [ ] `meteo_actuelle.py` → affiche les données brutes
- [ ] `meteo_actuelle_V2.py` → affiche les données formatées
- [ ] `meteo_code.py` → traduit les codes météo
- [ ] `meteo_previsions.py` → affiche 6 prochaines heures
- [ ] `meteo_qrcode.py` → génère `images/meteo_rodez_qr.png`

8.3. Vérifications :

- [ ] Aucune variable indéfinie
- [ ] Les chemins de fichiers sont corrects
- [ ] Les noms de variables sont cohérents
- [ ] Le QR code est scannable
