Cours / TP Python – Affichage de données météo et génération d'un QR Code

Introduction

Dans ce TP, nous allons récupérer des données météo depuis l'API **Open-Meteo**, les traiter en **Python** et les afficher sous différentes formes, jusqu'à la génération d'un **QR Code** contenant les prévisions météo.

Objectifs

- interroger une **API météo**
- exploiter une réponse au format **JSON**
- afficher des données météo en Python
- produire des **prévisions horaires**
- générer un **QR Code** contenant ces informations

> Les notions d'API et de QR Code ont déjà été vues en **JavaScript / HTML**.

Dans ce TP, nous les manipulons en **Python**.

Outils utilisés

- Python 3
- Bibliothèque `requests`
- API **Open-Meteo**
- Bibliothèque `qrcode`
- Environnement : **Thonny Python**

Rappel : API & JSON (similarité avec JavaScript)

- Une API fournit des données via le protocole **HTTP**
- La réponse est généralement au format **JSON**
- En JavaScript : `fetch(...)`
- En Python : `requests.get(...)`

QR code et emojis

> Dans ce TP, les QR codes sont générés avec **Python**.
>
> Contrairement à la version JavaScript vue dans un autre TP, **les emojis sont correctement interprétés lors du scan** et s'affichent directement avec le texte.

Exemple

```
# -*- coding: utf-8 -*-
import qrcode
```

```
data = (
    "🛒 FICHE PRODUIT\n"
    "————————————\n"
    "💻 MacBook Pro\n"
    "🧠 M5 • 💾 16 Go • 💿 512 Go\n"
    "💰 1 799,00 €\n"
    "📦 En stock\n"
    "🎨 Couleur : Gris sidéral\n"
    "📍 Magasin : Paris\n"
    "🚚 Livraison : 2–3 jours\n"
)

qr = qrcode.QRCode(
    version=None,  # taille automatique
    error_correction=qrcode.constants.ERROR_CORRECT_Q,
    box_size=10,
    border=4,
)

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

img = qr.make_image(fill_color="black", back_color="white")
img.save("qr_produit.png")
```

- Les retours à la ligne se font avec `\n`
- Les emojis sont intégrés directement dans la chaîne de caractères
- Le contenu du QR code doit rester **court et lisible** pour l'utilisateur final

> Scannez le QR code généré et observez le rendu du texte et des emojis.

Création de l'arborescence du TP par script Python

Installation requise :

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

Créer l'arborescence suivante en utilisant un script Python :

```
📂 TP_Meteo_Python/
│
├── meteo_actuelle.py
├── meteo_actuelle_V2.py
├── meteo_code.py
├── meteo_previsions.py
├── meteo_qrcode.py
│
├── 📁 images/
```

```
│
├── README.md
└── C_TP_meteo.pdf          # A copier-coller ici
```

Rôle de chaque fichier et dossier

Fichier `meteo_actuelle.py`

- Requête simple à l'API
- Affichage brut de la réponse JSON
- Découverte de la structure des données

Fichier `meteo_actuelle_V2.py`

- Version configurable
- Extraction des données météo actuelles
- Lever / coucher du soleil
- Gestion minimale des erreurs API

Fichier `meteo_code.py`

- Dictionnaire de traduction des `weathercode`
- Test et affichage de la description météo

Fichier `meteo_previsions.py`

- Récupération des **prévisions horaires**
- Affichage des 6 prochaines heures
- Manipulation des listes et de `zip()`

Fichier `meteo_qrcode.py`

- Construction du texte météo
- Génération et sauvegarde du **QR Code**
- Réutilisation des données météo

Dossier `images/`

- Dossier dédié aux fichiers générés automatiquement (afin d'éviter de polluer la racine du projet)

Script **à compléter** : `create_project.py`

```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
```

Requête météo simple – météo actuelle

L'API Open-Meteo permet d'obtenir gratuitement des données météo.

Localisation utilisée

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

Travail demandé

1. Compléter le fichier `meteo_actuelle.py` par le code ci-dessous

```
import requests

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

lat =  # latitude
lon =  # longitude

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 sont les champs disponibles dans `current_weather` ?

Extraction et affichage des données météo actuelles

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

Lever et coucher du soleil

Modifier la requête API afin d'obtenir :

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

Version configurable du programme (V2)

Créer une copie du fichier précédent nommée :

```
meteo_actuelle_V2.py
```

et créer une section de configuration contenant :

- le nom de la ville
- les coordonnées GPS
- l'URL de l'API
- le fuseau horaire

> Objectif : Rendre le programme plus **lisible**, **modifiable** et **réutilisable**.

Solution (V2)

```
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
t = current["time"]  # Maintenant t existe
heure = dt.datetime.fromisoformat(t).strftime("%H:%M")
print("Heure actuelle :", heure)

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"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}")
```

Traduction du weathercode (description du temps)

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

Travail demandé

1. Créer un dictionnaire Python permettant de traduire ce code en texte
2. Afficher la description correspondante

Exemple :

`0 → Ciel dégagé`

- Créer le fichier Python `meteo_code.py` et compléter le code suivant :

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

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

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

Questions

- Ajouter de nouveaux codes météo
- Afficher le code si la description est inconnue

Prévisions horaires `meteo_previsions.py`

Nous allons maintenant exploiter les données horaires fournies par l'API **sous forme de listes**.

```
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()

# Exemple : afficher les 6 prochaines heures
print("Prévisions – prochaines 6 heures\n")

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

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

# Exemple (formatage différent) : 6 prochaines heures
print("Prévisions – prochaines 6 heures\n")

for t, temp, p in zip(times[:6], temperatures[:6], precipitations[:6]):
    print(f"{t[-5:]} → {temp:>4}°C | pluie : {p} mm")
```

Questions

- Pourquoi `forecast["hourly"]["time"]` est une liste ?
- À quoi correspond l'indice `[0]` ?

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

Le QR Code contiendra les données météo actuelles ainsi que les prévisions des prochaines heures.

Installation de la bibliothèque :

```
!pip install qrcode[pil]
```

```
import requests
import qrcode
from datetime import datetime

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

```
# Codes météo
# TODO: Partie à compléter (à partir de la réponse à l'exercice précédent)
weather_codes = {
    0: "Ciel dégagé",
    1: "...",
    2: "",
}

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

try:
    response = requests.get(BASE_URL, params=params_current, timeout=10)
    response.raise_for_status()
    data_current = response.json()
    current = data_current["current_weather"]
except Exception as e:
    print(f"Erreur lors de la récupération de la météo actuelle : {e}")
    exit(1)

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

try:
    response = requests.get(BASE_URL, params=params_forecast, timeout=10)
    response.raise_for_status()
    forecast = response.json()
except Exception as e:
    print(f"Erreur lors de la récupération des prévisions : {e}")
    exit(1)

# 3. Extraction des données
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]
```

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

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

Exercices

- Limiter le QR à 3 heures
- Ajouter la date
- Ajouter le nom de la ville scannée

Questions

- Quels sont les avantages d'une API par rapport à un fichier local ?
- Quelle différence voyez-vous entre l'utilisation en Python et en JavaScript ?
- Que se passerait-il si l'API ne répond pas ?
