TP2 : Dashboard d'Analyse de Marché BTP

Projet Complet d'Automatisation

Scraping + Analyse + Visualisation + Rapports

Niveau : École d'Ingénieurs - Première année

Durée : 2h00

Prérequis : TD3, TD4

20 octobre 2025

Table des matières

1 Introduction

1.1 Objectifs du TP2

Ce TP a pour objectif de créer un **dashboard professionnel** d'analyse de marché pour le secteur du BTP. Vous allez :

— Développer un système automatisé de collecte de données

— Implémenter une architecture orientée objet (classe Python)

— Réaliser des analyses approfondies avec indicateurs avancés

— Générer des visualisations professionnelles multiples

— Produire des rapports automatisés

— Gérer les exports multi-formats (CSV, Excel, TXT, PNG)

1.2 Rappels des TD précédents

TD3 : Structure HTML, BeautifulSoup, extraction de données locales
TD4 : Scraping web, Pandas, visualisations, analyses statistiques

1.3 Compétences visées

À la fin de ce TP, vous serez capable de :

— Concevoir et implémenter un projet Python complet
— Structurer du code avec une architecture orientée objet
— Automatiser un pipeline de données (collecte → analyse → rapport)
— Produire des livrables professionnels
— Interpréter et communiquer des résultats d'analyse

2 Contexte du projet

2.1 Problématique

Une entreprise de génie civil souhaite **optimiser ses achats de matériaux** en :

— Surveillant les prix du marché en temps réel
— Identifiant les meilleures opportunités d'achat
— Comparant les fournisseurs
— Anticipant les variations de prix
— Générant des rapports automatiques pour la direction

2.2 Solution proposée

Vous allez développer **MarketBTP Analyzer**, un dashboard automatisé qui :

1. Collecte les données du catalogue MarketBTP
2. Analyse les tendances de prix par catégorie et région
3. Identifie les opportunités (bon prix + disponible + bonne note)
4. Compare la compétitivité des fournisseurs
5. Génère un rapport de synthèse
6. Exporte tous les résultats

2.3 Données source

Site MarketBTP : http://www.malomatique.free.fr/MarketBTP/index.html

- 60 produits BTP sur 3 pages
- 5 catégories de matériaux
- 8 fournisseurs
- 3 régions françaises
- Données : prix, notes, délais, disponibilité

3 Cahier des charges

3.1 Fonctionnalités obligatoires

Votre dashboard doit obligatoirement :

1. **Scraper automatiquement** les 3 pages du catalogue
2. **Nettoyer et valider** les données collectées
3. **Analyser** :
   - Prix moyens par catégorie
   - Prix moyens par fournisseur
   - Prix moyens par région
   - Distribution des notes
   - Analyse de disponibilité
4. **Identifier des opportunités** selon des critères définis
5. **Générer 6 visualisations minimum** :
   - Dashboard principal (6 graphiques en 1)
   - Prix par catégorie
   - Distribution des notes
   - Top fournisseurs
   - Disponibilité
   - Prix par région
   - Distribution des prix
6. **Produire un rapport texte** formaté avec :
   - Vue d'ensemble
   - Statistiques clés
   - Produits extrêmes
   - Analyse par catégorie
   - Opportunités identifiées
   - Recommandations
7. **Exporter** :
   - CSV complet des données
   - CSV des opportunités
   - Rapport TXT
   - Tous les graphiques en PNG

3.2 Architecture du projet

Votre projet doit suivre une **architecture orientée objet** avec une classe principale :

```python
class DashboardMarketBTP:
    """Classe pour analyser le marche des materiaux BTP"""

    def __init__(self, base_url):
        """Initialisation avec l'URL du catalogue"""
        pass

    def scraper_catalogue(self):
        """Collecte toutes les donnees du catalogue"""
        pass

    def nettoyer_donnees(self):
        """Nettoie et formate les donnees"""
        pass

    def analyser_tendances(self):
        """Analyse les tendances du marche"""
        pass

    def identifier_opportunites(self):
        """Identifie les meilleures opportunites"""
        pass

    def comparer_fournisseurs(self):
        """Compare les fournisseurs"""
        pass

    def generer_visualisations(self):
        """Cree le dashboard complet"""
        pass

    def generer_rapport(self):
        """Genere un rapport texte de synthese"""
        pass

    def exporter_donnees(self):
        """Exporte tous les resultats"""
        pass

    def executer_analyse_complete(self):
        """Execute le pipeline complet d'analyse"""
        pass
```

Listing 1 – Structure de la classe DashboardMarketBTP

4 Implémentation détaillée

4.1 Code de départ

```python
import requests
from bs4 import BeautifulSoup
import pandas as pd
import matplotlib.pyplot as plt
import re
import time
from datetime import datetime

# Configuration matplotlib
```

```
plt.rcParams['font.sans-serif'] = ['DejaVu Sans']

# Fonctions utilitaires (a completer)
def extraire_prix(texte_prix):
    """Extrait le prix numerique"""
    match = re.search(r'(\d+\.?\d*)', texte_prix)
    return float(match.group(1)) if match else 0.0

def compter_etoiles(texte_note):
    """Compte les etoiles pleines"""
    return texte_note.count('\u2605')

def scraper_page(url):
    """Scrape une page et retourne les produits"""
    response = requests.get(url)
    if response.status_code != 200:
        return []

    soup = BeautifulSoup(response.text, 'html.parser')
    produits = []
    cards = soup.find_all('div', class_='product-card')

    for card in cards:
        produit = {
            'Type': '',
            'Nom': '',
            'Fournisseur': '',
            'Prix': 0.0,
            'Unite': '',
            'Note': 0,
            'Disponibilite': '',
            'Delai': '',
            'Region': ''
        }

        # Extraction des donnees (a completer comme TD4)
        # ...

        produits.append(produit)

    return produits

class DashboardMarketBTP:
    """Classe pour analyser le marche des materiaux BTP"""

    def __init__(self, base_url):
        self.base_url = base_url
        self.pages = ['index.html', 'page-2.html', 'page-3.html']
        self.df = None
        self.date_analyse = datetime.now()

    def scraper_catalogue(self):
        """Collecte toutes les donnees du catalogue"""
        print("Collecte des donnees en cours...")
        tous_produits = []

        for i, page in enumerate(self.pages, 1):
            url = self.base_url + page
            print(f"  Page {i}/3...")
            produits = scraper_page(url)
            tous_produits.extend(produits)
            time.sleep(1)  # Pause pour ne pas surcharger

```

```
        self.df = pd.DataFrame(tous_produits)
        print(f"  -> {len(self.df)} produits collectes")

    def nettoyer_donnees(self):
        """Nettoie et formate les donnees"""
        print("\nNettoyage des donnees...")
        taille_avant = len(self.df)

        # Supprimer les doublons
        self.df = self.df.drop_duplicates()

        # Supprimer les produits sans prix
        self.df = self.df[self.df['Prix'] > 0]

        print(f"  -> {taille_avant - len(self.df)} entrees supprimees")
        print(f"  -> {len(self.df)} produits valides")

    def analyser_tendances(self):
        """Analyse les tendances du marche"""
        print("\n" + "="*60)
        print("ANALYSE DES TENDANCES")
        print("="*60)

        # A completer : analyses par categorie, fournisseur, region

        tendances = self.df.groupby('Type').agg({
            'Prix': ['mean', 'min', 'max', 'count'],
            'Note': 'mean'
        }).round(2)

        print("\nTendances par categorie:")
        print(tendances)

        return tendances

    def identifier_opportunites(self):
        """Identifie les meilleures opportunites"""
        print("\n" + "="*60)
        print("IDENTIFICATION DES OPPORTUNITES")
        print("="*60)

        # Criteres : En stock + Prix < Q1 + Note >= 4
        q1_prix = self.df['Prix'].quantile(0.25)

        opportunites = self.df[
            (self.df['Disponibilite'] == 'En stock') &
            (self.df['Prix'] < q1_prix) &
            (self.df['Note'] >= 4)
        ].copy()

        print(f"\nCriteres : En stock + Prix < {q1_prix:.2f} + Note >= 4")
        print(f"Opportunites trouvees : {len(opportunites)}")

        if len(opportunites) > 0:
            economie = self.df['Prix'].mean() - opportunites['Prix'].mean()
            print(f"Economie potentielle : {economie:.2f} euros en moyenne")

        return opportunites

    def comparer_fournisseurs(self):
        """Compare les fournisseurs"""
        print("\n" + "="*60)
        print("COMPARAISON DES FOURNISSEURS")
```

```
        print("="*60)

        comparison = self.df.groupby('Fournisseur').agg({
            'Prix': 'mean',
            'Note': 'mean',
            'Nom': 'count'
        }).rename(columns={'Nom': 'Nb_produits'}).round(2)

        comparison = comparison.sort_values('Prix')
        print(comparison)

        return comparison

    def generer_visualisations(self):
        """Cree le dashboard complet"""
        print("\nGeneration du dashboard...")

        # Dashboard avec 6 graphiques
        fig, axes = plt.subplots(2, 3, figsize=(18, 12))
        fig.suptitle('Dashboard MarketBTP - Analyse Complete',
                     fontsize=20, fontweight='bold', y=0.995)

        # 1. Prix par categorie
        prix_cat = self.df.groupby('Type')['Prix'].mean().sort_values()
        axes[0,0].barh(range(len(prix_cat)), prix_cat.values,
                       color='steelblue')
        axes[0,0].set_yticks(range(len(prix_cat)))
        axes[0,0].set_yticklabels(prix_cat.index)
        axes[0,0].set_xlabel('Prix moyen (euros)')
        axes[0,0].set_title('Prix moyen par categorie', fontweight='bold')
        axes[0,0].grid(axis='x', alpha=0.3)

        # 2. Distribution des notes
        note_counts = self.df['Note'].value_counts().sort_index()
        axes[0,1].bar(note_counts.index, note_counts.values,
                      color='orange', edgecolor='black')
        axes[0,1].set_xlabel('Note (/5)')
        axes[0,1].set_ylabel('Nombre de produits')
        axes[0,1].set_title('Distribution des notes', fontweight='bold')
        axes[0,1].grid(axis='y', alpha=0.3)

        # 3-6 : A completer (top fournisseurs, disponibilite,
        #        prix par region, distribution des prix)

        plt.tight_layout()
        plt.savefig('dashboard_complet.png', dpi=300, bbox_inches='tight')
        print("  -> dashboard_complet.png cree")
        plt.close()

    def generer_rapport(self):
        """Genere un rapport texte de synthese"""
        print("\nGeneration du rapport...")

        rapport = []
        rapport.append("="*70)
        rapport.append("RAPPORT D'ANALYSE MARKETBTP")
        rapport.append(f"Date : {self.date_analyse.strftime('%d/%m/%Y %H:%M')}")
        rapport.append("="*70)
        rapport.append("")

        # Vue d'ensemble
        rapport.append("1. VUE D'ENSEMBLE")
        rapport.append(f"   Nombre total de produits : {len(self.df)}")
```

```
        rapport.append(f"   Prix moyen : {self.df['Prix'].mean():.2f} euros")
        # A completer...

        # Sauvegarder
        contenu = '\n'.join(rapport)
        with open('rapport_marketbtp.txt', 'w', encoding='utf-8') as f:
            f.write(contenu)

        print("  -> rapport_marketbtp.txt cree")
        print("\n" + contenu)

    def exporter_donnees(self):
        """Exporte tous les resultats"""
        print("\nExport des donnees...")

        # CSV complet
        self.df.to_csv('marketbtp_complet.csv',
                       index=False, encoding='utf-8')
        print("  -> marketbtp_complet.csv")

        # Opportunites
        opportunites = self.identifier_opportunites()
        if len(opportunites) > 0:
            opportunites.to_csv('opportunites.csv',
                                index=False, encoding='utf-8')
            print("  -> opportunites.csv")

    def executer_analyse_complete(self):
        """Execute le pipeline complet d'analyse"""
        print("="*70)
        print("DASHBOARD MARKETBTP - ANALYSE COMPLETE")
        print("="*70)

        self.scraper_catalogue()
        self.nettoyer_donnees()
        self.analyser_tendances()
        self.identifier_opportunites()
        self.comparer_fournisseurs()
        self.generer_visualisations()
        self.generer_rapport()
        self.exporter_donnees()

        print("\n" + "="*70)
        print("ANALYSE TERMINEE AVEC SUCCES")
        print("="*70)

# Programme principal
def main():
    base_url = 'http://www.malomatique.free.fr/MarketBTP/'
    dashboard = DashboardMarketBTP(base_url)
    dashboard.executer_analyse_complete()

if __name__ == "__main__":
    main()
```

Listing 2 – dashboard_btp.py - Structure complete

5 Travail à réaliser

5.1 Tâches obligatoires

1. **Compléter la fonction** `scraper_page()`

   - Extraire tous les champs de données (Type, Nom, Fournisseur, Prix, etc.)
   - Gérer les cas où des données sont manquantes
   - Nettoyer les textes (strip, replace)
2. **Implémenter `analyser_tendances()`**
   - Analyses par catégorie (prix moyen, min, max, nombre)
   - Analyses par fournisseur
   - Analyses par région
   - Afficher les résultats formatés
3. **Compléter `generer_visualisations()`**
   - Graphique 3 : Top 5 fournisseurs (nombre de produits)
   - Graphique 4 : Disponibilité (pie chart)
   - Graphique 5 : Prix par région (bar chart)
   - Graphique 6 : Distribution des prix (histogramme)
4. **Enrichir `generer_rapport()`**
   - Ajouter les statistiques clés
   - Ajouter les produits extrêmes
   - Ajouter l'analyse par catégorie
   - Ajouter les opportunités
   - Ajouter des recommandations
5. **Tester et valider**
   - Exécuter le programme complet
   - Vérifier tous les exports
   - Valider la cohérence des résultats

5.2 Fonctionnalités bonus (optionnelles)

- Ajouter des graphiques supplémentaires (scatter plots, boxplots)
- Implémenter un système de scoring des fournisseurs
- Ajouter une analyse de corrélation prix/note
- Générer un export Excel avec plusieurs feuilles
- Créer une interface simple en ligne de commande
- Ajouter un système de logs détaillés

6 Livrables attendus

6.1 Code source

- `dashboard_btp.py` : Programme principal complet et commenté
- Code propre, structuré et documenté (docstrings)
- Gestion d'erreurs robuste

6.2 Données et analyses

- `marketbtp_complet.csv` : Données complètes scrapées
- `opportunites.csv` : Opportunités identifiées
- `rapport_marketbtp.txt` : Rapport de synthèse

6.3 Visualisations

Minimum 6 graphiques PNG haute résolution (300 dpi) :

- `dashboard_complet.png` : Dashboard principal (6 en 1)
- Possibilité d'ajouter des graphiques individuels supplémentaires

6.4 Documentation (optionnelle)

- `README.md` : Instructions d'utilisation
- Explication des choix techniques
- Guide d'interprétation des résultats

7 Critères d'évaluation

7.1 Fonctionnalité (40%)

- Le programme s'exécute sans erreur
- Toutes les fonctionnalités obligatoires sont implémentées
- Les données scrapées sont correctes et complètes
- Les analyses sont pertinentes et correctes

7.2 Qualité du code (30%)

- Architecture orientée objet respectée
- Code propre et lisible
- Commentaires et docstrings présents
- Gestion d'erreurs appropriée
- Bonnes pratiques Python respectées

7.3 Analyses et visualisations (20%)

- Analyses statistiques correctes
- Graphiques clairs et informatifs
- Qualité professionnelle des visualisations
- Rapport de synthèse complet et structuré

7.4 Originalité et dépassement (10%)

- Fonctionnalités bonus implémentées
- Créativité dans les analyses
- Documentation complète
- Interface utilisateur améliorée

8 Conseils et bonnes pratiques

8.1 Méthodologie

1. **Commencer simple** : Faites fonctionner les bases avant d'ajouter des fonctionnalités
2. **Tester régulièrement** : Testez chaque fonction individuellement
3. **Déboguer méthodiquement** : Utilisez des print() pour comprendre les problèmes
4. **Consulter la documentation** : Pandas et Matplotlib ont une excellente doc
5. **Réutiliser le TD4** : Le code du TD4 est une excellente base

8.2 Pièges à éviter

— Ne pas gérer les erreurs réseau (connexion, timeout)
— Oublier le délai entre les requêtes (time.sleep)
— Ne pas nettoyer les données avant analyse
— Graphiques illisibles (taille de police, labels)
— Code non commenté et difficile à comprendre

8.3 Ressources utiles

— Documentation Pandas : https://pandas.pydata.org/docs/
— Matplotlib gallery : https://matplotlib.org/stable/gallery/index.html
— Stack Overflow pour les questions spécifiques
— Vos notes des TD3 et TD4

9 Conclusion

Ce TP vous permet de mettre en pratique toutes les compétences acquises lors des TD précédents dans un projet complet et professionnel. Un tel dashboard est directement applicable dans le monde de l'entreprise pour :

— Optimiser les achats
— Surveiller le marché
— Prendre des décisions data-driven
— Automatiser la veille concurrentielle

Prenez le temps de bien comprendre chaque étape et n'hésitez pas à expérimenter. Le web scraping et l'analyse de données sont des compétences essentielles pour l'ingénieur moderne !

Checklist finale avant soumission :

☐ Le programme s'exécute du début à la fin sans erreur
☐ Tous les fichiers sont générés (CSV, TXT, PNG)
☐ Le code est commenté et documenté
☐ Les résultats sont cohérents et validés
☐ Le rapport est complet et structuré
☐ Les graphiques sont de qualité professionnelle

Fin du TP2 - Durée estimée : 2h00

Bon courage et bonne programmation !