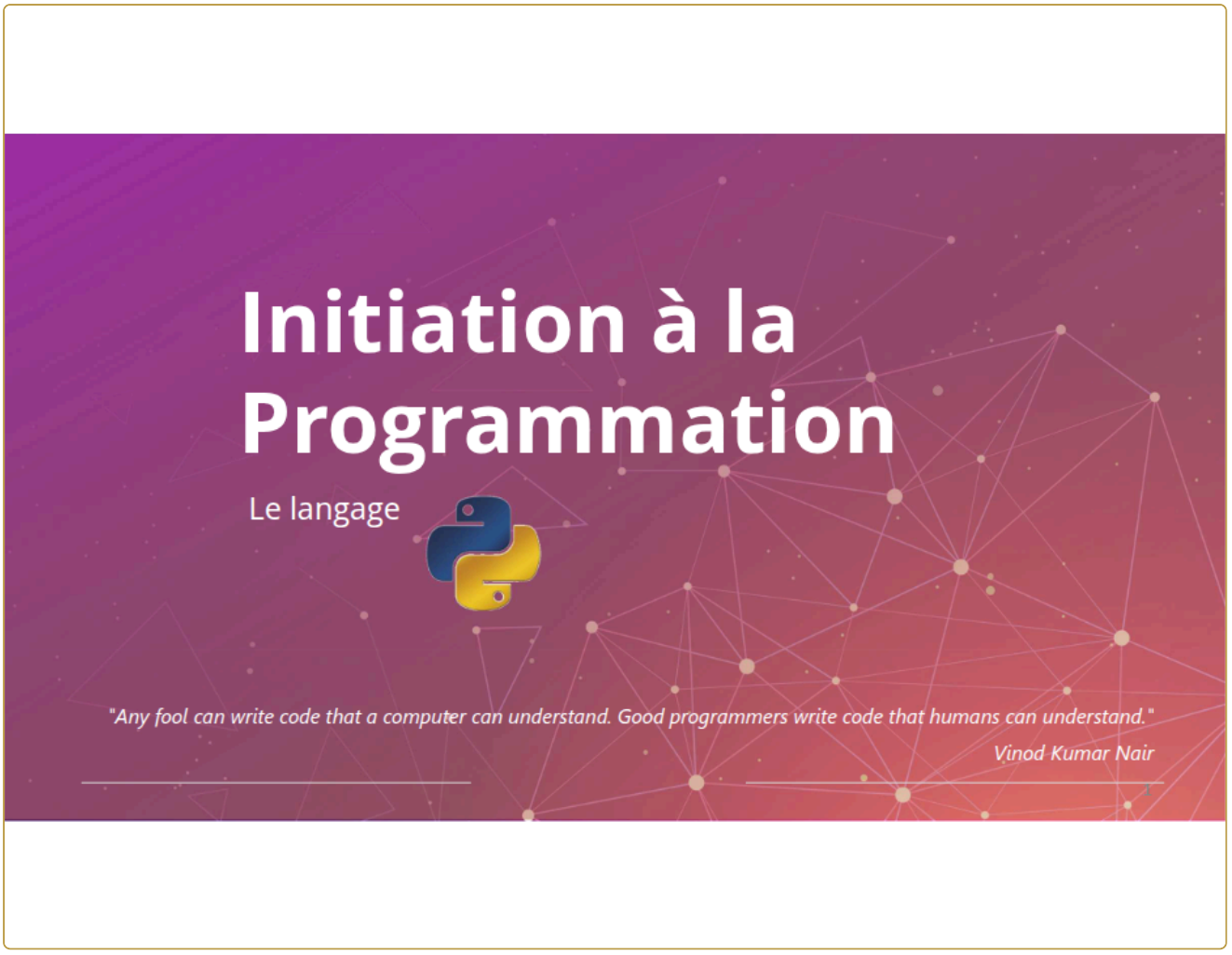
Note Python

Initiation_Programmation_Le langage Python.pdf



Le langage (les entrées/sorties)

1. print('Je suis nouveau en python !')

```
❯ python3 ./script1.pi
Je suis nouveau en python !
```

2. Print("Mamadou")

```
❯ python3 ./script1.pi
Mamadou
```

3. Mon nom est pris pour une variable

```
❯ python3 ./script1.pi
Traceback (most recent call last):
  File "/home/mamadou/env-i/Construction/./script1.pi", line 1, in <module>
    print(Mamadou)
          ^^^^^^^
NameError: name 'Mamadou' is not defined
```

4. Il comprend qu'il y a une erreur et qu'il manque des parenthèses

```
❯ python3 ./script1.pi
  File "/home/mamadou/env-i/Construction/./script1.pi", line 1
    print"Mamadou"
    ^^^^^^^^^^^^^^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print( ... )?
```

5. Plusieurs print sur la meme ligne ne sont pas autorisé

```
❯ python3 ./script1.pi
  File "/home/mamadou/env-i/Construction/./script1.pi", line 2
    print("ligne 1") print("ligne 1")
                     ^^^^^
SyntaxError: invalid syntax
```

6. Nouveau est ajouté juste après "suis"

```
❯ python3 ./script1.pi
je suis nouveau
```

Recherchez l'utilisation de l'argument sep dans la fonction print().

```
valeur1 = 'test1'
valeur2 = 'test2'
print(valeur1, valeur2, ..., sep="séparateur")
```

```
❯ python3 ./script1.pi
test1séparateurtest2séparateurEllipsis
```

Il est possible de préciser sur combien de caractères le résultat doit être écrit et comment se fait l'alignement (à gauche (<), à droite (>) ou centré (^)).

```
print(f"{10:>10d}"); print(f"{1000:>10d}")
```

```
❯ python3 ./script1.pi
        10
      1000
```

La fonction input() : est capable de lire les données saisies par l’utilisateur et de renvoyer les mêmes données au programme en cours d’exécution.

```
mon_nom = input("Saisir votre nom complet : ")
print(type(mon_nom))
```

```
❯ python3 ./script1.pi
Saisir votre nom complet : mamadou
<class 'str'>
❯ python3 ./script1.pi
Saisir votre nom complet : 2
<class 'str'>
```

La fonction input() lit les données saisies par l'utilisateur et les envoie au programme en cours d'exécution.

```
maVariable = input("Saisissez un chiffre : ")
monResultat = maVariable ** 2
print(maVariable, "au carré est :", monResultat)
```

```
❯ python3 ./script1.pi
Saisissez un chiffre : 2
Traceback (most recent call last):
  File "/home/mamadou/env-i/Construction/./script1.pi", line 2, in <module>
    monResultat = maVariable ** 2
                  ~~~~~~~~~~~^^~~
TypeError: unsupported operand type(s) for ** or pow(): 'str' and 'int'
```

Ecrire un programme en Python qui demande l'âge d'un enfant à l'utilisateur. Ensuite, il l'informe de sa catégorie :

- « Poussin » de 6 à 7 ans
- « Pupille » de 8 à 9 ans
- « Minime » de 10 à 11 ans
- « Cadet » après 12 ans

```
print("Ecrire 0 pour la fin")



while True:

age = int(input("Quel est l'age de l'enfant ? : "))



if age == 7 or age == 6 :

print("L'enfant est un poussin car il a : ", age)

elif age == 8 or age == 9 :

print("L'enfant est une pupille car il a : ", age)
```

```
elif age == 10 or age == 11 :

print ("L'enfant est minime car il a : ", age)

elif age > 11 :

print("Trop vieux, ", age)

elif age == 0 :

print("Fin des questions")

break
```

Ecrire un programme en Python qui permette

1. Entrer deux nombres.
2. Choisir une opération (+, -, , /).
3. Selon l'opération choisie, effectuer le calcul correspondant.
4. Gérer le cas spécial de la division par zéro. 5. Afficher le résultat.

```
print("Pour quitter la boucle écrire 'end'")



while True:

A = int(input(("Entrer le premier nombre : ")))

B = int(input(("Entrer le 2e nombre : ")))

R = 0



choix = input("Choisir une opération entre : +, -, * et / :")



if choix == "+":

R = A + B

print(f"Le résultat est : ",R)

elif choix == "-":
```

```
R = A - B

print(f"Le résultat est : ",R)

elif choix == "*" :

R = A * B

print(f"Le résultat est : ",R)

elif choix == "/" :

R = A / B

print(f"Le résultat est : ",R)

elif choix == "end":

break

else:

print("Il faut bien répondre a la question")
```

---

scripting_python_P2.pdf

1

2

Manipulation de fichiers exemple :

```
f = None
fichier_a_lire = "exemple_1.txt"

try:
    # 1 OUVERTURE : Spécification du mode 'r' (Lecture) et de l'encodage
'utf-8'
```

```
    # Si le fichier avait été encodé en 'latin-1', nous aurions du
utiliser 'latin-1' ici.
    f = open(fichier_a_lire, 'r',encoding='utf-8')

    # 2 Opétation : Lecture de tout le contenu
    contenu = f.read()

    print("--- Contenu lu avec succès ---")
    print(contenu)


    # Gestion d'erruer
except FileNotFoundError:
    print(f"Erreur : Le fichier '{fichier_a_lire}' n'a pas pu être
trouvé")
except UnicodeDecodeError:
    # Cette erreur sruvient si l'encodage 'utf-8' ne correspond pas au
fichier réel
    print("Erreur : L'encodage 'utf-8' spécifié n'a pas pu décoder les
octets du fichier")

    # 3 Fermeture Manuelle : Le bloc 'finally' garantit que f.clase() sera
exécuté.
finally:
    if f is not None:
        f.close()
        print("\nLe fichier a été fermé manuellement dans le bloc
'finally'.")
```

Modes d'ouverture :

```
# Lecture seule

with open("exemple.txt", "r", encoding="utf-8") as fichier:

contenu = fichier.read()

print(f'Contenu du fichier {contenu}')



# Écriture seule, création si non existant

with open("exemple.txt", "w", encoding="utf-8") as fichier:

fichier.write("Bonjour, monde !")
```

```
# Ajout de contenu à un fichier existant

with open("exemple.txt", "a", encoding="utf-8") as fichier:

fichier.write("\nNouvelle Lligne ajoutée.")



# Lecture et écriture

with open("exemple.txt", "r+", encoding="utf-8") as fichier:

donnees = fichier.read()

print(f'Contenu du fichier {donnees}')

fichier.write("\nFin de l'exercice")
```

Python et les nombres aléatoires :

```
python

import random

# Liste de fruits
fruits = ["pomme", "banane", "orange", "kiwi", "mangue"]

# Tirage aléatoire
fruit_choisi = random.choice(fruits)

print("Fruit choisi :", fruit_choisi)
```

Exercice :
Ecrire un script Python qui génère un mot de passe qui respecte les recommandations de la CNIL. Le script demande à l'utilisateur la longueur souhaitée pour le mot de passe.

```
import random

while True:
    mot_de_passe = input("Ecrire un mot de passe qui contient une
majuscule, 12 caractères et minimum un chiffre : ")

    if (any("A" <= char <= "Z" for char in mot_de_passe)
        and any("0" <= char <= "9" for char in mot_de_passe)
        and len(mot_de_passe) >= 12):
        print("Mot de passe accepte")
        break
```

```
    else:
        print("mot de passe valide.")
```

TP Thème 1 – "CrackMe" : Jeu de mots de passe faibles

ÉNONCÉ
Un site mal sécurisé contient une liste de mots de passe faibles (parmi les 10 plus utilisés).

Vous devez écrire un petit script Python qui :

1. Sélectionne un mot de passe aléatoire parmi une liste prédéfinie
2. Invite l'utilisateur à deviner ce mot de passe
3. Donne un indice si la tentative est incorrecte
4. Affiche le nombre de tentatives à la fin

```
import random

pw =
["123456","password","admin","123456789","qwerty","abc123","letmein","welc
ome","monkey","football"]

secretV = random.choice(pw)

essaieV = 0

while True:

guess = input("Devine : ").strip()

essaieV += 1

if guess == secretV:

print(f"Bravo Trouvé en {essaieV} essais : {secretV}")

break

h = random.choice(("long","start","common"))

if h == "long":

print("Indice :", "plus long" if len(secretV) > len(guess) else "plus
court" if len(secretV) < len(guess) else "même longueur")

elif h == "start":

print("Indice :", "même première lettre" if guess and guess[0] ==
```

```
secretV[0] else "pas la même première lettre")

else:

print("Indice :", len(set(guess) & set(secretV)), "caractères communs")
```

TP : Génération de logs synthétiques (version texte)

Complétez ce script pour générer un fichier de logs "en clair" contenant des événements simulés de différents serveurs et services. Objectifs :

- Utiliser la bibliothèque random
- Générer des événements réalistes (host, process, niveau, message…)
- Produire un fichier de logs texte lisible

```
import random

from datetime import datetime, timedelta

from pathlib import Path


# ----------------------------

# Configuration générale

# ----------------------------

N_EVENTS = 100 # Nombre de logs à générer

DAYS_RANGE = 1 # Étendue de temps (en jours)

OUT_FILE = "synthetic_system.log"


# Ajouter les erreurs, sources

HOSTS = ["srv-web01", "srv-web02", "db-master", "db-replica", "fw1"]

PROCS = ["sshd", "nginx", "postgres", "systemd", "cron"]

LEVELS = ["INFO", "WARNING", "ERROR", "CRITICAL"]


MESSAGES = [
```

```
"connexion acceptée depuis {ip}",
"échec d'authentification utilisateur {user}",
"latence élevée: {lat} ms",
"erreur I/O sur disque {dev}",
"redémarrage du service {proc}",
"paquet dropped par firewall"
]

# ------------------------------
# Fonctions utilitaires
# ------------------------------

def random_ip():
"""Retourne une adresse IPv4 aléatoire."""
# générer 4 entiers entre 1 et 254 et les concaténer avec "."
parts = [str(random.randint(1, 254)) for _ in range(4)]
return ".".join(parts)

def random_user():
"""Retourne un utilisateur choisi aléatoirement."""
users = ["alice", "bob", "admin", "guest", "monitor"]
# choisir un élément de la liste users avec random.choice
return random.choice(users)


def random_dev():
"""Retourne un périphérique disque aléatoire."""
devices = ["/dev/sda1", "/dev/nvme0n1", "/dev/sdb"]
```

```
# choisir un élément de la liste devices avec random.choice

return random.choice(devices)


# ----------------------------

# Boucle de génération

# ----------------------------

def main():

# Point de départ temporel (aujourd'hui - DAYS_RANGE)

start_time = datetime.now() - timedelta(days=DAYS_RANGE)

events = []


# Calcul de l'intervalle en secondes (utilisé pour répartir les
timestamps)

max_seconds = int(DAYS_RANGE * 24 * 3600)


for i in range(N_EVENTS):

# générer un delta aléatoire en secondes (sur l'intervalle DAYS_RANGE)

# utilisation de random.randint pour obtenir un entier de secondes

delta_seconds = random.randint(0, max_seconds)



ts = start_time + timedelta(seconds=delta_seconds)

ts_iso = ts.isoformat(sep=' ', timespec='seconds')



# choisir un host, un process, et un niveau

host = random.choice(HOSTS)

proc = random.choice(PROCS)

level = random.choice(LEVELS)
```

```
# choisir un modèle de message et le compléter avec format()

template = random.choice(MESSAGES)

msg = template.format(

ip=random_ip(),

user=random_user(),

lat=f"{random.uniform(10, 2000):.1f}",

dev=random_dev(),

proc=proc

)


# Format d’un log texte :

# 2025-09-29 10:23:45 srv-web01 nginx[1234]: INFO connexion acceptée
depuis 192.168.1.10

line = f"{ts_iso} {host} {proc}[{random.randint(1000,9999)}]: {level}
{msg}"



events.append((ts, line))



# Trier les événements par timestamp

events.sort(key=lambda x: x[0])

lines = [line for _, line in events]



# Écrire dans un fichier texte

with Path(OUT_FILE).open("w", encoding="utf-8") as f:

for line in lines:
```

```
    f.write(line + "\n")


print(f"{len(lines)} événements générés dans {OUT_FILE}")



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

Initiation_programmation_Python_P3

1

2

page 34 :

```
import re

# Expression régulière pour valider un email

email_pattern = r'^[a-zA-Z0-9_.+-]+@[a-zA-Z0-9-]+\.[a-zA-Z8-9-.]+$'
```

```
email = input("Saisissez votre adresse email : " )

if re.match(email_pattern, email):
    print ("Adresse email valide.")
else:
    print("Adresse email invalide.")
```

page 35 :

```
import re

test_ipv4 = r'^[0-9]+.[0-9]+.[0-9]+.[0-9]+$'
ipv4 = input("saisissez une adresse ip")

if re.match(test_ipv4, ipv4):
    print("Adresse mail valide")
else :
    print("Adresse ip pas valide")
```

TP2_Module2_Python

TP Thème 2 – Analyse de logs SSH simulés (auth.log)

Objectifs pédagogiques :

- Lire et parcourir un fichier de log
- Extraire des informations pertinentes (IPs, utilisateurs, erreurs)
- Compter et classer les tentatives de connexion suspectes
- Visualiser les IPs les plus actives (bonus)

Consignes

Contexte : Vous êtes analyste sécurité dans une entreprise. Vous recevez une copie d'un fichier de log SSH et devez identifier des comportements suspects : IPs avec de nombreuses tentatives d'accès échouées.

Fichier fourni

Un fichier **auth.log** est mis à disposition. Il contient un mélange de tentatives échouées et réussies de connexion SSH.

Partie 1 – Analyse textuelle (script simple)

1. Ouvrir le fichier auth.log
2. Extraire toutes les lignes contenant **"Failed password"**
3. Extraire les adresses IP de ces lignes à l'aide d'une **expression régulière**
4. Compter le nombre d'occurrences de chaque IP
5. Afficher les **5 IPs** ayant généré le plus d'échecs

Partie 2 – Visualisation (script avancé)

1. Utiliser la bibliothèque *matplotlib* (utilisez **pip install matplotlib** si nécessaire)
2. Créer un **graphique de barres** représentant les IPs avec le plus grand nombre d'échecs
3. Comparer les IPs ayant échoué et celles ayant réussi (bonus)
4. Ajouter une **légende, un titre**, et des **axes lisibles**

Bonus – Analyse avancée

- Filtrer les tentatives réussies (Accepted password) et les comparer aux échecs
- Exporter les résultats dans un fichier CSV ou JSON

Partie 1 – Analyse textuelle (script simple) :

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import re
from collections import Counter
```

```
def analyse_log():
    fichier_log = "auth.log" # Nom du fichier dans le dossier courant

    try:
        with open(fichier_log, "r", encoding="utf-8") as f:
            lignes = f.readlines()

    except FileNotFoundError:
        print(f"[ERREUR] Le fichier '{fichier_log}' est introuvable dans
le dossier courant.")
        return


    # Étape 1 : extraire les lignes contenant "Failed password"
    lignes_failed = [ligne for ligne in lignes if "Failed password" in
ligne]

    # Étape 2 : extraire les adresses IP avec une expression régulière
    pattern_ip = re.compile(r'(\d{1,3}(?:\.\d{1,3}){3})')

    ips = [pattern_ip.search(ligne).group() for ligne in lignes_failed if
pattern_ip.search(ligne)]

    # Étape 3 : compter le nombre d'occurrences de chaque IP
    compteur_ips = Counter(ips)

    # Étape 4 : afficher les 5 IPs ayant généré le plus d'échecs
    print("\n TOP 5 des IPs avec le plus d'échecs de connexion SSH ===")
    for ip, count in compteur_ips.most_common(5):
        print(f"{ip} -> {count} tentatives échouées")

    print("\nNombre total de tentatives échouées :", len(ips))

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

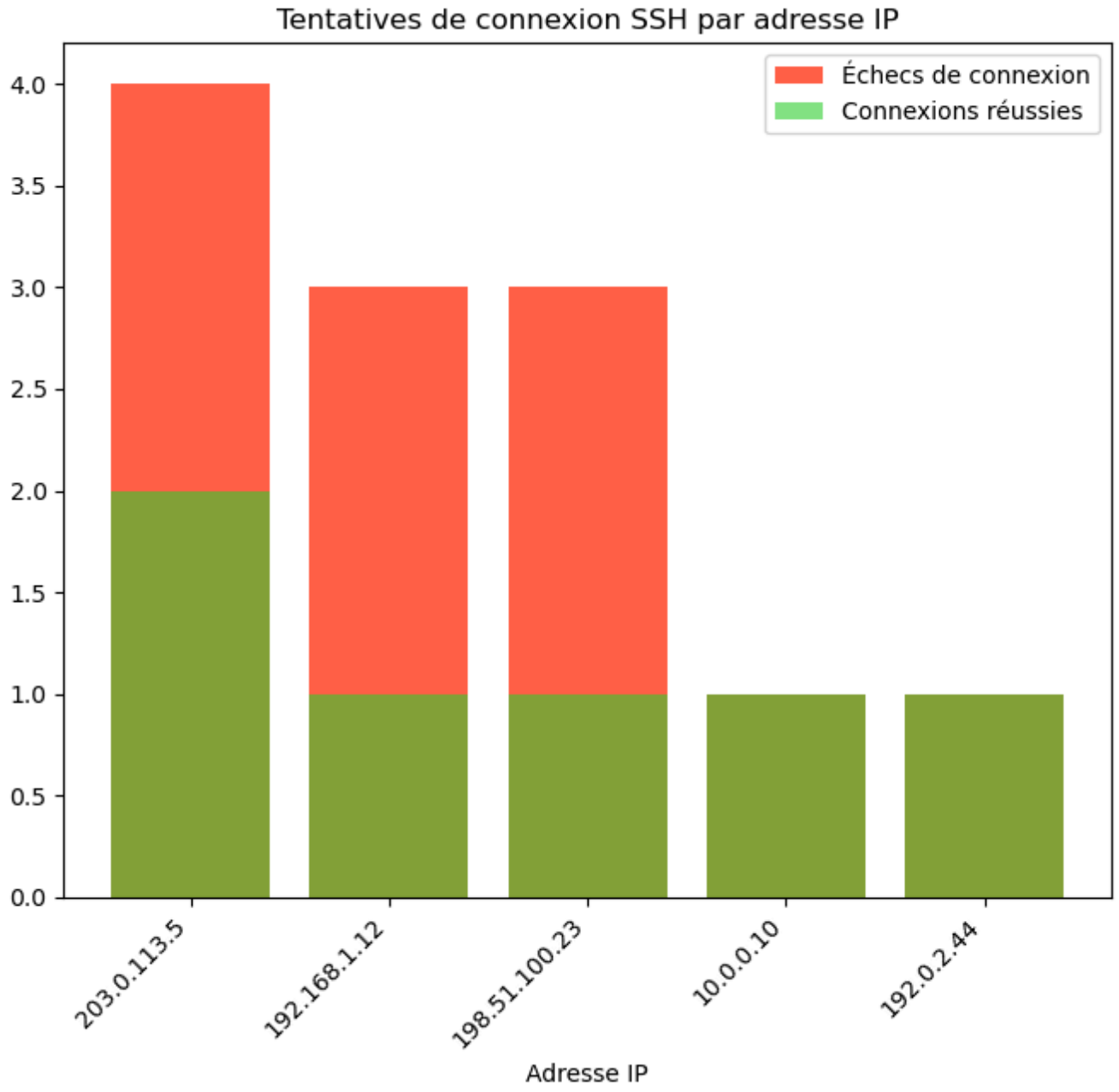
Partie 2 – Visualisation (script avancé) :

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import re
from collections import Counter
import matplotlib.pyplot as plt

def analyse_log(fichier_log="auth.log"):
    try:
        with open(fichier_log, "r", encoding="utf-8") as f:
            lignes = f.readlines()
```

```
    except FileNotFoundError:
        print(f"[ERREUR] Le fichier '{fichier_log}' est introuvable.")
        return None, None

    lignes_failed = [l for l in lignes if "Failed password" in l]
    lignes_success = [l for l in lignes if "Accepted password" in l]

    pattern_ip = re.compile(r'(\d{1,3}(?:\.\d{1,3}){3})')
    ips_failed = [pattern_ip.search(l).group() for l in lignes_failed if
pattern_ip.search(l)]
    ips_success = [pattern_ip.search(l).group() for l in lignes_success if
pattern_ip.search(l)]

    compteur_failed = Counter(ips_failed)
    compteur_success = Counter(ips_success)

    return compteur_failed, compteur_success

def visualiser_ips(compteur_failed, compteur_success):
    if not compteur_failed:
        print("[INFO] Aucun échec détecté, rien à visualiser.")
        return

    top_failed = compteur_failed.most_common(10)
    ips, valeurs = zip(*top_failed)

    plt.figure(figsize=(10, 6))
    plt.bar(ips, valeurs, color='tomato', label='Échecs de connexion')

    valeurs_success = [compteur_success.get(ip, 0) for ip in ips]
    plt.bar(ips, valeurs_success, color='limegreen', alpha=0.6,
label='Connexions réussies')

    plt.title("Tentatives de connexion SSH par adresse IP")
    plt.xlabel("Adresse IP")
    plt.ylabel("Nombre de tentatives")
    plt.xticks(rotation=45, ha='right')
    plt.legend()
    plt.tight_layout()

    plt.show()

if __name__ == "__main__":
    fichier = "auth.log"
    compteur_failed, compteur_success = analyse_log(fichier)
    visualiser_ips(compteur_failed, compteur_success)
```

Fichier analyse_log :

```
May 10 14:01:59 ubuntu sshd[12345]: Failed password for invalid user root
from 192.168.1.12 port 55874 ssh2
May 10 14:02:02 ubuntu sshd[12345]: Failed password for user1 from
203.0.113.5 port 55875 ssh2
May 10 14:02:05 ubuntu sshd[12345]: Failed password for user1 from
203.0.113.5 port 55876 ssh2
May 10 14:02:10 ubuntu sshd[12345]: Failed password for admin from
10.0.0.10 port 55877 ssh2
May 10 14:02:15 ubuntu sshd[12345]: Failed password for invalid user test
from 198.51.100.23 port 55878 ssh2
May 10 14:02:20 ubuntu sshd[12345]: Failed password for root from
192.168.1.12 port 55879 ssh2
May 10 14:02:25 ubuntu sshd[12345]: Failed password for root from
192.168.1.12 port 55880 ssh2
May 10 14:02:30 ubuntu sshd[12345]: Failed password for user1 from
203.0.113.5 port 55881 ssh2
May 10 14:02:35 ubuntu sshd[12345]: Failed password for root from
192.0.2.44 port 55882 ssh2
May 10 14:02:40 ubuntu sshd[12345]: Failed password for user1 from
203.0.113.5 port 55883 ssh2
May 10 14:02:45 ubuntu sshd[12345]: Failed password for invalid user guest
from 198.51.100.23 port 55884 ssh2
May 10 14:02:45 ubuntu sshd[12345]: Failed password for invalid user guest
from 198.51.100.23 port 55884 ssh2
May 10 14:03:00 ubuntu sshd[12346]: Accepted password for user1 from
203.0.113.5 port 55900 ssh2
May 10 14:03:05 ubuntu sshd[12347]: Accepted password for admin from
192.168.1.12 port 55901 ssh2
May 10 14:03:10 ubuntu sshd[12348]: Accepted password for user2 from
10.0.0.10 port 55902 ssh2
May 10 14:03:15 ubuntu sshd[12349]: Accepted password for root from
192.0.2.44 port 55903 ssh2
May 10 14:03:20 ubuntu sshd[12350]: Accepted password for user1 from
203.0.113.5 port 55904 ssh2
May 10 14:03:25 ubuntu sshd[12351]: Accepted password for guest from
198.51.100.23 port 55905 ssh2
```



TP3_Module3_Python

TP Thème 3 - Apache Logs – Analyse d'erreurs 404

Objectif : Extraire, traiter et visualiser les IPs responsables d'erreurs 404.

Contexte :

Un fichier **access.log** vous a été remis par l'administrateur système. Il contient les requêtes HTTP d'un serveur Apache. Vous devez l'analyser pour identifier les IPs générant le plus d'erreurs 404.

Étapes du TP :

1. **Chargement et parsing du fichier**
    - Charger **access.log** dans un **DataFrame**.
    - Extraire les colonnes suivantes : **ip**, **datetime**, **method**, **url**, **status**, **user_agent**.
    - Nettoyer ou ignorer les lignes malformées.
2. **Filtrage des erreurs 404**
    - Isoler les lignes où le status est **404**.
3. **Top 5 des IPs fautives**
    - Grouper par IP.
    - Compter et trier pour afficher les 5 IPs générant le plus d'erreurs.
4. **Visualisation**
    - Créer un histogramme (bar chart) avec **matplotlib**.
        - **Axe X** : IPs
        - **Axe Y** : Nombre d'erreurs 404
        - Personnaliser le graphique (titre, couleurs, labels).

Bonus : Détection de bots

- Filtrer les lignes dont le **user_agent** contient "bot", "crawler" ou "spider".
- Identifier les IPs suspectes.
- Calculer le pourcentage d'erreurs 404 provenant de bots.

Résultat attendu

- Un script Python contenant :
    - Le code structuré (fonctions recommandées).

```
Ouvrir le fichier log,
with open("/Users/lp1/Downloads/access.log", "r") as f:
    lignes = f.readlines()

Expression régulière pour trouver les lignes dont le status est 404,
pattern = re.compile(r'\s404\s')

Filtrer les lignes avec l'erreur 404,
```

```
lignes_404 = [ligne for ligne in lignes if re.search(pattern, ligne)]

Afficher le résultat,
for ligne in lignes_404:
    print(ligne.strip())

     Compter les 5 IP qui ressortent le plus avec l'erreur 404

     from collections import Counter
ips = [re.match(r'^(\d+.\d+.\d+.\d+)', ligne).group(1) for ligne in
lignes_404]
top_5 = Counter(ips).most_common(5)
print(top_5)

    Visualisation sur Matplotlib :

import matplotlib.pyplot as plt

Extraire les IPs et leurs comptes,
ips, comptes = zip(*Counter([re.match(r'^(\d+.\d+.\d+.\d+)',
ligne).group(1) for ligne in lignes_404]).most_common(5))

Créer l'histogramme,
plt.bar(ips, comptes, color='red')
plt.xlabel("IPs")
plt.ylabel("Nombre d'erreurs 404")
plt.show()
```