

Solutions des Exercices Python

Ce document contient les solutions détaillées de tous les exercices du cours Python pour débutants.

Solution Exercice 1 : Recherche sur Python

Réponses attendues :

1. En quelle année Python a-t-il été créé et par qui ?

- Python a été créé en 1989 (développement commencé en décembre 1989, première version publiée en 1991) par Guido van Rossum.

2. Pourquoi le langage s'appelle-t-il "Python" ?

- Le nom vient de la série télévisée britannique "Monty Python's Flying Circus" que Guido van Rossum adorait. Cela n'a rien à voir avec le serpent.

3. Citez trois domaines d'application de Python

- Développement web (Django, Flask)
- Data Science et Machine Learning (Pandas, TensorFlow)
- Automatisation et scripts système
- (Autres réponses acceptables : IA, jeux vidéo, applications desktop, IoT, etc.)

4. Quelle est la différence majeure entre Python 2 et Python 3 ?

- Python 3 est une refonte majeure non rétrocompatible avec Python 2. Python 2 n'est plus maintenu depuis 2020. Les différences incluent la fonction print(), la gestion des chaînes Unicode, la division d'entiers, etc.

5. Trouvez deux entreprises célèbres qui utilisent Python

- Exemples : Google, Netflix, NASA, Instagram, Spotify, Dropbox, Reddit, YouTube, etc.

Solution Exercice 2 : Premier Programme

```
# Programme de bienvenue
```

```
# Auteur : [Votre nom]
```

```
# Affichage du message de bienvenue
print("Bienvenue dans le cours Python !")
```

```
# Affichage du prénom
print("Jean")
```

```
# Affichage du message d'apprentissage
print("J'apprends à programmer")
```

Solution Exercice 3 : Variables et Types

```
# Demander les informations à l'utilisateur
prenom = input("Quel est votre prénom ? ")
age = int(input("Quel âge avez-vous ? "))
taille = float(input("Quelle est votre taille en mètres ? "))
est_etudiant_str = input("Êtes-vous étudiant ? (True/False) ")
est_etudiant = est_etudiant_str == "True"
```

```
# Afficher les informations avec leurs types
print(f"\nPrénom : {prenom} (type: {type(prenom)})")
print(f"Âge : {age} (type: {type(age)})")
print(f"Taille : {taille} (type: {type(taille)})")
print(f"Étudiant : {est_etudiant} (type: {type(est_etudiant)})")
```

```
# Calculer l'âge dans 10 ans
age_futur = age + 10
print(f"\nDans 10 ans, vous aurez {age_futur} ans.")
```

Version alternative avec validation :

```
# Demander les informations avec gestion d'erreurs basique
```

try:

```
prenom = input("Quel est votre prénom ? ")
age = int(input("Quel âge avez-vous ? "))
taille = float(input("Quelle est votre taille en mètres ? "))

# Demander si étudiant
reponse = input("Êtes-vous étudiant ? (oui/non) ").lower()
est_etudiant = reponse == "oui"

# Afficher les informations
print(f"\n{'='*50}")
print(f"Prénom : {prenom} (type: {type(prenom)})")
print(f"Âge : {age} (type: {type(age)})")
print(f"Taille : {taille} (type: {type(taille)})")
print(f"Étudiant : {est_etudiant} (type: {type(est_etudiant)})")
print(f"{'='*50}")

# Calculer l'âge dans 10 ans
age_futur = age + 10
print(f"\nDans 10 ans, vous aurez {age_futur} ans.")
```

except ValueError:

```
    print("Erreur : Veuillez entrer des valeurs valides.")
```

Solution Exercice 4 : Calculatrice Simple

```
# Demander deux nombres
nombre1 = float(input("Entrez le premier nombre : "))
nombre2 = float(input("Entrez le deuxième nombre : "))
```

```
print("\n=== RÉSULTATS DES OPÉRATIONS ===")
```

```
# Toutes les opérations arithmétiques
```

```
addition = nombre1 + nombre2
```

```
soustraction = nombre1 - nombre2
```

```
multiplication = nombre1 * nombre2
```

```
division = nombre1 / nombre2 if nombre2 != 0 else "Impossible (division par zéro)"
```

```
division_entiere = nombre1 // nombre2 if nombre2 != 0 else "Impossible"
```

```
modulo = nombre1 % nombre2 if nombre2 != 0 else "Impossible"
```

```
puissance = nombre1 ** nombre2
```

```
print(f"{nombre1} + {nombre2} = {addition}")
```

```
print(f"{nombre1} - {nombre2} = {soustraction}")
```

```
print(f"{nombre1} * {nombre2} = {multiplication}")
```

```
print(f"{nombre1} / {nombre2} = {division}")
```

```
print(f"{nombre1} // {nombre2} = {division_entiere}")
```

```
print(f"{nombre1} % {nombre2} = {modulo}")
```

```
print(f"{nombre1} ** {nombre2} = {puissance}")
```

```
# Demander l'âge
```

```
print("\n=== VÉRIFICATIONS ===")
```

```
age = int(input("Quel âge avez-vous ? "))
```

```
# Vérifier si majeur et peut voter
```

```
est_majeur = age >= 18
```

```
peut_voter = age >= 18
```

```
print(f"Majeur : {est_majeur}")
```

```
print(f"Peut voter : {peut_voter}")
print(f"Majeur ET peut voter : {est_majeur and peut_voter}")

# Vérifier si enfant ou adolescent
est_enfant = age < 12
est_adolescent = 12 <= age <= 17
print(f"Enfant (< 12) : {est_enfant}")
print(f"Adolescent (12-17) : {est_adolescent}")
print(f"Enfant OU adolescent : {est_enfant or est_adolescent}")
```

Solution Exercice 5 : Système de Notes

```
# Demander la note
note = float(input("Entrez une note entre 0 et 20 : "))

# Vérifier la validité de la note
if note < 0 or note > 20:
    print("Erreur : La note doit être entre 0 et 20")
else:
    # Déterminer la mention
    print("\n=== RÉSULTAT ===")

    if note >= 16:
        print("Mention : Très bien")
    elif note >= 14:
        print("Mention : Bien")
    elif note >= 12:
        print("Mention : Assez bien")
    elif note >= 10:
```

```
    print("Mention : Passable")
else:
    print("Mention : Insuffisant")

# Vérifier la réussite
if note >= 10:
    print("✓ Vous avez réussi !")
else:
    print("X Vous devez repasser l'examen")

# Vérifier si mention obtenue
if note >= 12:
    print("✓ Vous avez obtenu une mention")
else:
    print("X Pas de mention obtenue")
```

Version avec gestion d'erreurs :

```
try:
    note = float(input("Entrez une note entre 0 et 20 : "))

    if note < 0 or note > 20:
        print("⚠ Erreur : La note doit être entre 0 et 20")
    else:
        print("\n" + "="*40)
        print("BULLETIN DE NOTES".center(40))
        print("="*40)
        print(f"Note obtenue : {note}/20")
        print("-"*40)
```

```
# Mention

if note >= 16:

    mention = "Très bien"

elif note >= 14:

    mention = "Bien"

elif note >= 12:

    mention = "Assez bien"

elif note >= 10:

    mention = "Passable"

else:

    mention = "Insuffisant"

print(f"Mention : {mention}")

print("-"*40)
```

```
# Réussite

if note >= 10:

    print("Résultat : ✓ RÉUSSI")

    print("Vous pouvez passer à l'année suivante")

else:

    print("Résultat : ✗ ÉCHEC")

    print("Vous devez repasser l'examen")
```

```
# Mention obtenue

if note >= 12:

    print("Félicitations : Vous avez une mention !")

print("="*40)
```

```
except ValueError:
```

```
    print("⚠ Erreur : Veuillez entrer un nombre valide")
```

Solution Exercice 6 : Tables de Multiplication

```
# Demander le nombre
```

```
try:
```

```
    nombre = int(input("Entrez un nombre entre 1 et 10 : "))
```

```
# Vérifier la validité
```

```
if nombre < 1 or nombre > 10:
```

```
    print("Le nombre doit être entre 1 et 10")
```

```
else:
```

```
    print(f"\n=== TABLE DE MULTIPLICATION DE {nombre} ===\n")
```

```
    somme_totale = 0
```

```
# Afficher la table
```

```
for i in range(1, 11):
```

```
    resultat = i * nombre
```

```
    somme_totale += resultat
```

```
    print(f"{i} x {nombre} = {resultat}")
```

```
print(f"\nSomme totale : {somme_totale}")
```

```
# Demander si continuer
```

```
continuer = True
```

```
while continuer:
```



```
reponse = input("\nVoulez-vous voir une autre table ? (oui/non) : ").lower()
```

```
if reponse == "oui":
```

```
    nombre = int(input("Entrez un nombre entre 1 et 10 : "))
```

```
    if 1 <= nombre <= 10:
```

```
        print(f"\n=== TABLE DE MULTIPLICATION DE {nombre} ===\n")
```

```
        somme_totale = 0
```

```
        for i in range(1, 11):
```

```
            resultat = i * nombre
```

```
            somme_totale += resultat
```

```
            print(f"{i} x {nombre} = {resultat}")
```

```
        print(f"\nSomme totale : {somme_totale}")
```

```
    else:
```

```
        print("Nombre invalide")
```

```
elif reponse == "non":
```

```
    continuer = False
```

```
    print("Au revoir !")
```

```
else:
```

```
    print("Répondez par 'oui' ou 'non'")
```

```
except ValueError:
```

```
    print("Veuillez entrer un nombre entier valide")
```

Version avec tableau formaté :

```
nombre = int(input("Entrez un nombre entre 1 et 10 : "))
```

```
if 1 <= nombre <= 10:
```

```
    print(f"\n{'='*30}")
```

```
    print(f"TABLE DE MULTIPLICATION DE {nombre}".center(30))
```

```
print('='*30)

somme = 0

for i in range(1, 11):
    resultat = i * nombre
    somme += resultat
    print(f"{i:2d} x {nombre:2d} = {resultat:3d}")

print('='*30)

print(f"SOMME : {somme}".center(30))

print('='*30)

else:
    print("Nombre invalide")
```

Solution Exercice 7 : Gestion de Notes

Créer une liste vide pour les notes

```
notes = []
```

```
print("=== SAISIE DES NOTES ===\n")
```

Demander 5 notes

```
for i in range(5):
```

```
    while True:
```

```
        try:
```

```
            note = float(input(f"Entrez la note {i+1} (entre 0 et 20) : "))
```

```
            if 0 <= note <= 20:
```

```
                notes.append(note)
```

```
                break
```

```
    else:

        print("⚠ La note doit être entre 0 et 20")

    except ValueError:

        print("⚠ Veuillez entrer un nombre valide")


# Afficher toutes les notes

print("\n=== NOTES SAISIES ===")

print(f"Notes : {notes}")


# Calculer les statistiques

moyenne = sum(notes) / len(notes)

note_min = min(notes)

note_max = max(notes)

nb_reussites = sum(1 for note in notes if note >= 10)


print("\n=== STATISTIQUES ===")

print(f"Moyenne : {moyenne:.2f}/20")

print(f>Note minimale : {note_min}/20")

print(f>Note maximale : {note_max}/20")

print(f"Nombre de notes >= 10 : {nb_reussites}/{len(notes)}")


# Trier les notes

notes.sort()

print(f"\nNotes triées (ordre croissant) : {notes}")


# Supprimer une note

print("\n=== SUPPRESSION D'UNE NOTE ===")

try:
```

```

note_a_supprimer = float(input("Quelle note voulez-vous supprimer ? "))

if note_a_supprimer in notes:
    notes.remove(note_a_supprimer)
    print(f"✓ Note {note_a_supprimer} supprimée")

# Nouvelle liste et moyenne
print(f"\nNouvelle liste : {notes}")

if len(notes) > 0:
    nouvelle_moyenne = sum(notes) / len(notes)
    print(f"Nouvelle moyenne : {nouvelle_moyenne:.2f}/20")
else:
    print("La liste est maintenant vide")
else:
    print("⚠ Cette note n'est pas dans la liste")
except ValueError:
    print("⚠ Entrée invalide")

```

Solution Exercice 8 : Gestion de Coordonnées

```

import math

# Liste pour stocker les coordonnées
coordonnees = []

print("=== SAISIE DES COORDONNÉES ===\n")

# Demander 3 coordonnées
for i in range(3):
    x = float(input(f"Point {i+1} - Entrez x : "))

```

```

y = float(input(f"Point {i+1} - Entrez y : "))
coordonnees.append((x, y))

# Afficher tous les points
print("\n=== POINTS SAISIS ===")
for i, (x, y) in enumerate(coordonnees, 1):
    print(f"Point {i} : ({x}, {y})")

# Calculer les distances
print("\n=== DISTANCES PAR RAPPORT À L'ORIGINE (0, 0) ===")
distances = []

for i, (x, y) in enumerate(coordonnees, 1):
    distance = math.sqrt(x**2 + y**2)
    distances.append(distance)
    print(f"Point {i} : {distance:.2f} unités")

# Créer un tuple avec les distances
tuple_distances = tuple(distances)
print(f"\nTuple des distances : {tuple_distances}")

# Trouver le point le plus proche
index_min = distances.index(min(distances))
point_proche = coordonnees[index_min]
print(f"\n=== POINT LE PLUS PROCHE ===")
print(f"Point {index_min + 1} : {point_proche}")
print(f"Distance : {distances[index_min]:.2f} unités")

```

```

# Démonstration de l'unpacking

print("\n=== DÉMONSTRATION DE L'UNPACKING ===")

for i, coord in enumerate(coordonnees, 1):
    x, y = coord # Unpacking
    print(f"Point {i} décomposé : x = {x}, y = {y}")

# Tentative de modification d'un tuple (erreur)

print("\n=== TENTATIVE DE MODIFICATION D'UN TUPLE ===")

try:
    premier_point = coordonnees[0]
    print(f"Tentative de modifier {premier_point}...")
    premier_point[0] = 999 # Ceci va générer une erreur
except TypeError as e:
    print(f"X Erreur : {e}")
    print("Les tuples sont immuables !")

```

Solution Exercice 9 : Carnet d'Adresses

```

# Dictionnaire pour stocker les contacts

contacts = {}

def afficher_menu():
    print("\n" + "="*50)
    print("CARNET D'ADRESSES".center(50))
    print("="*50)
    print("1. Ajouter un contact")
    print("2. Afficher tous les contacts")
    print("3. Rechercher un contact")
    print("4. Modifier un numéro de téléphone")

```

```
print("5. Supprimer un contact")  
print("6. Afficher le nombre de contacts")  
print("7. Contacts par ville")  
print("8. Quitter")  
print("="*50)
```

```
def ajouter_contact():  
    nom = input("Nom du contact : ").strip()  
    if nom in contacts:  
        print("⚠ Ce contact existe déjà")  
        return  
  
    telephone = input("Téléphone : ").strip()  
    email = input("Email : ").strip()  
    ville = input("Ville : ").strip()  
  
    contacts[nom] = {  
        "telephone": telephone,  
        "email": email,  
        "ville": ville  
    }  
    print(f"✓ Contact '{nom}' ajouté avec succès")
```

```
def afficher_contacts():  
    if not contacts:  
        print("⚠ Aucun contact dans le carnet")  
        return
```

```
print("\n" + "="*50)

print("LISTE DES CONTACTS".center(50))

print("="*50)
```

```
for nom, info in contacts.items():

    print(f"\nNom : {nom}")

    print(f" Téléphone : {info['telephone']}")

    print(f" Email : {info['email']}")

    print(f" Ville : {info['ville']}")

    print("-"*50)
```

```
def rechercher_contact():

    nom = input("Nom du contact à rechercher : ").strip()
```

```
    if nom in contacts:

        info = contacts[nom]

        print("\n=== CONTACT TROUVÉ ===")

        print(f"Nom : {nom}")

        print(f"Téléphone : {info['telephone']}")

        print(f"Email : {info['email']}")

        print(f"Ville : {info['ville']}")
```

```
    else:

        print("⚠ Contact non trouvé")
```

```
def modifier_telephone():

    nom = input("Nom du contact : ").strip()
```

```
    if nom in contacts:
```



```
nouveau_tel = input("Nouveau numéro de téléphone : ").strip()
```

```
contacts[nom]['telephone'] = nouveau_tel
```

```
print(f"✓ Numéro de téléphone de '{nom}' modifié")
```

```
else:
```

```
print("⚠ Contact non trouvé")
```

```
def supprimer_contact():
```

```
    nom = input("Nom du contact à supprimer : ").strip()
```

```
    if nom in contacts:
```

```
        confirmation = input(f"Êtes-vous sûr de vouloir supprimer '{nom}' ? (oui/non) : ")
```

```
        if confirmation.lower() == "oui":
```

```
            del contacts[nom]
```

```
            print(f"✓ Contact '{nom}' supprimé")
```

```
        else:
```

```
            print("Suppression annulée")
```

```
    else:
```

```
        print("⚠ Contact non trouvé")
```

```
def afficher_nombre_contacts():
```

```
    nombre = len(contacts)
```

```
    print(f"\n 📞 Nombre total de contacts : {nombre}")
```

```
def contacts_par_ville():
```

```
    ville = input("Entrez le nom de la ville : ").strip()
```

```
    contacts_ville = {nom: info for nom, info in contacts.items()
```

```
        if info['ville'].lower() == ville.lower()]}
```

```
if contacts_ville:

    print(f"\n== CONTACTS À {ville.upper()} ==")

    for nom, info in contacts_ville.items():

        print(f"\n{nom}")

        print(f" Téléphone : {info['telephone']}")

        print(f" Email : {info['email']}")

else:

    print(f"⚠️ Aucun contact trouvé à {ville}")
```

Programme principal

```
def main():

    while True:

        afficher_menu()

        choix = input("\nVotre choix : ").strip()

        if choix == "1":

            ajouter_contact()

        elif choix == "2":

            afficher_contacts()

        elif choix == "3":

            rechercher_contact()

        elif choix == "4":

            modifier_telephone()

        elif choix == "5":

            supprimer_contact()

        elif choix == "6":

            afficher_nombre_contacts()
```

```
elif choix == "7":
    contacts_par_ville()
elif choix == "8":
    print("\nAu revoir ! 🤖")
    break
else:
    print("⚠ Choix invalide")

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

Solution Exercice 10 : Analyse de Texte

Demander les deux phrases

```
phrase1 = input("Entrez la première phrase : ")
```

```
phrase2 = input("Entrez la deuxième phrase : ")
```

Convertir en sets de mots (en minuscules pour comparaison)

```
mots1 = set(phrase1.lower().split())
```

```
mots2 = set(phrase2.lower().split())
```

```
print("\n" + "="*60)
```

```
print("ANALYSE DES PHRASES".center(60))
```

```
print("="*60)
```

Afficher les mots uniques de chaque phrase

```
print("\n--- Mots uniques de la phrase 1 ---")
```

```
print(mots1)
```

```
print("\n--- Mots uniques de la phrase 2 ---")
```

```
print(mots2)
```

```
# Mots communs (intersection)
```

```
mots_communs = mots1 & mots2
```

```
print("\n--- Mots communs aux deux phrases (n) ---")
```

```
print(mots_communs)
```

```
print(f"Nombre : {len(mots_communs)}")
```

```
# Mots uniquement dans phrase 1 (différence)
```

```
mots_seulement_phrase1 = mots1 - mots2
```

```
print("\n--- Mots uniquement dans la phrase 1 (phrase1 - phrase2) ---")
```

```
print(mots_seulement_phrase1)
```

```
print(f"Nombre : {len(mots_seulement_phrase1)}")
```

```
# Mots uniquement dans phrase 2
```

```
mots_seulement_phrase2 = mots2 - mots1
```

```
print("\n--- Mots uniquement dans la phrase 2 (phrase2 - phrase1) ---")
```

```
print(mots_seulement_phrase2)
```

```
print(f"Nombre : {len(mots_seulement_phrase2)}")
```

```
# Union (tous les mots)
```

```
tous_mots = mots1 | mots2
```

```
print("\n--- Tous les mots uniques (U) ---")
```

```
print(tous_mots)
```

```
print(f"Nombre total de mots uniques : {len(tous_mots)}")
```

```
# Mot le plus long
```

```

if tous_mots:

    mot_plus_long = max(tous_mots, key=len)

    print(f"\n--- Mot le plus long ---")

    print(f"'{mot_plus_long}' ({len(mot_plus_long)} caractères)")


print("\n" + "="*60)


# Statistiques supplémentaires

print("\n--- Statistiques détaillées ---")

print(f"Mots dans phrase 1 : {len(mots1)}")

print(f"Mots dans phrase 2 : {len(mots2)}")

print(f"Mots en commun : {len(mots_communs)}")

print(f"Similarité : {len(mots_communs) / len(tous_mots) * 100:.1f}%")

```

Version avec analyse avancée :

```

def analyser_phrases():

    phrase1 = input("Entrez la première phrase : ")
    phrase2 = input("Entrez la deuxième phrase : ")


    # Nettoyer les phrases (enlever ponctuation)

    import string

    phrase1_nettoyee = phrase1.translate(str.maketrans("", "", string.punctuation))
    phrase2_nettoyee = phrase2.translate(str.maketrans("", "", string.punctuation))


    # Convertir en sets

    mots1 = set(phrase1_nettoyee.lower().split())
    mots2 = set(phrase2_nettoyee.lower().split())


    # Toutes les analyses

```

```
intersection = mots1 & mots2
```

```
union = mots1 | mots2
```

```
diff1 = mots1 - mots2
```

```
diff2 = mots2 - mots1
```

```
diff_sym = mots1 ^ mots2
```

```
# Affichage
```

```
print("\n" + "="*70)
```

```
print("ANALYSE COMPARATIVE DE TEXTES".center(70))
```

```
print("="*70)
```

```
print(f"\nPhrase 1 : \"{phrase1}\"")
```

```
print(f"Phrase 2 : \"{phrase2}\"")
```

```
print(f"\n{'Catégorie':<30}{'Nombre':<10} Mots")
```

```
print("-"*70)
```

```
print(f"{'Mots phrase 1':<30}{len(mots1):<10}{sorted(mots1)}")
```

```
print(f"{'Mots phrase 2':<30}{len(mots2):<10}{sorted(mots2)}")
```

```
print(f"{'Mots communs (n)':<30}{len(intersection):<10}{sorted(intersection)}")
```

```
print(f"{'Seulement phrase 1':<30}{len(diff1):<10}{sorted(diff1)}")
```

```
print(f"{'Seulement phrase 2':<30}{len(diff2):<10}{sorted(diff2)}")
```

```
print(f"{'Diff. symétrique ( $\oplus$ ):<30}{len(diff_sym):<10}{sorted(diff_sym)}")
```

```
print(f"{'Union (U)':<30}{len(union):<10}{sorted(union)}")
```

```
# Statistiques
```

```
print("\n" + "="*70)
```

```
print("STATISTIQUES".center(70))
```

```
print("="*70)
```

```

similarite = len(intersection) / len(union) * 100 if union else 0
print(f"Similarité : {similarite:.1f}%")

if union:
    mot_long = max(union, key=len)
    mot_court = min(union, key=len)
    print(f"Mot le plus long : '{mot_long}' ({len(mot_long)} lettres)")
    print(f"Mot le plus court : '{mot_court}' ({len(mot_court)} lettres)")

print("="*70)

if __name__ == "__main__":
    analyser_phrases()

```

Solution Exercice 11 : Traitement de Texte

```

# Demander la phrase
phrase = input("Entrez une phrase : ")

print("\n" + "="*60)
print("ANALYSE DE TEXTE".center(60))
print("="*60)

# Phrase originale
print(f"\nPhrase originale : {phrase}")

# Majuscules et minuscules
print(f"\nEn MAJUSCULES : {phrase.upper()}")

```

```
print(f"En minuscules : {phrase.lower()}")
```

```
# Nombre de caractères
```

```
nb_caracteres_total = len(phrase)
```

```
nb_caracteres_sans_espaces = len(phrase.replace(" ", ""))
```

```
print(f"\n--- Statistiques de caractères ---")
```

```
print(f"Nombre de caractères (avec espaces) : {nb_caracteres_total}")
```

```
print(f"Nombre de caractères (sans espaces) : {nb_caracteres_sans_espaces}")
```

```
# Nombre de mots
```

```
mots = phrase.split()
```

```
nb_mots = len(mots)
```

```
print(f"Nombre de mots : {nb_mots}")
```

```
# Compter les voyelles
```

```
voyelles = "aeiouyAEIOUY"
```

```
nb_voyelles = sum(1 for caractere in phrase if caractere in voyelles)
```

```
print(f"Nombre de voyelles : {nb_voyelles}")
```

```
# Inverser la phrase
```

```
phrase_inversee = phrase[::-1]
```

```
print(f"\nPhrase inversée : {phrase_inversee}")
```

```
# Remplacer les espaces
```

```
phrase_avec_underscores = phrase.replace(" ", "_")
```

```
print(f"Espaces remplacés par _ : {phrase_avec_underscores}")
```



```
# Premier et dernier mot
```

```
if mots:
```

```
    print(f"\nPremier mot : {mots[0]}")
```

```
    print(f"Dernier mot : {mots[-1]}")
```

```
# Vérifier si palindrome
```

```
phrase_sans_espaces = phrase.replace(" ", "").lower()
```

```
est_palindrome = phrase_sans_espaces == phrase_sans_espaces[::-1]
```

```
print(f"\n--- Test de palindrome ---")
```

```
print(f"Est un palindrome : {'Oui ✓' if est_palindrome else 'Non ✗'}")
```

```
print("\n" + "="*60)
```

Version avec analyse détaillée :

```
import string
```

```
def analyser_texte(phrase):
```

```
    """Analyse complète d'un texte"""
```

```
    print("\n" + "="*70)
```

```
    print("ANALYSE DÉTAILLÉE DE TEXTE".center(70))
```

```
    print("="*70)
```

```
# 1. Transformations de casse
```

```
print("\n--- Transformations de casse ---")
```

```
print(f"Original   : {phrase}")
```

```
print(f"MAJUSCULES  : {phrase.upper()}")
```

```
print(f"minuscules  : {phrase.lower()}")
```

```
print(f"Titre      : {phrase.title()}")
print(f"Capitalisé : {phrase.capitalize()}")
print(f"Inversé    : {phrase.swapcase()}")
```

2. Statistiques de caractères

```
print("\n--- Statistiques de caractères ---")
nb_total = len(phrase)
nb_sans_espaces = len(phrase.replace(" ", ""))
nb_lettres = sum(c.isalpha() for c in phrase)
nb_chiffres = sum(c.isdigit() for c in phrase)
nb_espaces = phrase.count(" ")

print(f"Caractères totaux      : {nb_total}")
print(f"Caractères (sans espace) : {nb_sans_espaces}")
print(f"Lettres                  : {nb_lettres}")
print(f"Chiffres                  : {nb_chiffres}")
print(f"Espaces                  : {nb_espaces}")
```

3. Analyse des voyelles et consonnes

```
print("\n--- Voyelles et consonnes ---")
voyelles = "aeiouyAEIOUY"
nb_voyelles = sum(1 for c in phrase if c in voyelles)
nb_consonnes = sum(1 for c in phrase if c.isalpha() and c not in voyelles)

print(f"Voyelles : {nb_voyelles}")
print(f"Consonnes : {nb_consonnes}")
```

Détail des voyelles

```
compteur_voyelles = {}
for voyelle in "aeiouy":
    compte = phrase.lower().count(voyelle)
    if compte > 0:
        compteur_voyelles[voyelle] = compte

print("Détail voyelles :", compteur_voyelles)
```

4. Analyse des mots

```
print("\n--- Analyse des mots ---")
mots = phrase.split()
nb_mots = len(mots)

print(f"Nombre de mots : {nb_mots}")
```

```
if mots:
    print(f"Premier mot : '{mots[0]}'")
    print(f"Dernier mot : '{mots[-1]}'")
```

```
longueurs = [len(mot) for mot in mots]
print(f"Mot le plus long : '{max(mots, key=len)}' ({max(longueurs)} lettres)")
print(f"Mot le plus court : '{min(mots, key=len)}' ({min(longueurs)} lettres)")
print(f"Longueur moyenne : {sum(longueurs)/len(longueurs):.1f} lettres")
```

5. Transformations

```
print("\n--- Transformations ---")
print(f"Inversée : {phrase[::-1]}")
print(f"Sans espaces : {phrase.replace(' ', '')}")
```

```
print(f"Espaces → underscore : {phrase.replace(' ', '_')}")
```

```
print(f"Espaces → tirets : {phrase.replace(' ', '-')}")
```

```
# 6. Test palindrome
```

```
print("\n--- Test palindrome ---")
```

```
phrase_nettoyee = ''.join(c.lower() for c in phrase if c.isalnum())
```

```
est_palindrome = phrase_nettoyee == phrase_nettoyee[::-1]
```

```
print(f"Phrase nettoyée : '{phrase_nettoyee}'")
```

```
print(f"Est un palindrome : {'Oui ✓' if est_palindrome else 'Non X'}")
```

```
# 7. Mots les plus fréquents
```

```
print("\n--- Fréquence des mots ---")
```

```
frequence = {}
```

```
for mot in mots:
```

```
    mot_lower = mot.lower().strip(string.punctuation)
```

```
    frequence[mot_lower] = frequence.get(mot_lower, 0) + 1
```

```
# Trier par fréquence
```

```
mots_tries = sorted(frequence.items(), key=lambda x: x[1], reverse=True)
```

```
print("Mots les plus fréquents :")
```

```
for mot, freq in mots_tries[:5]: # Top 5
```

```
    print(f" '{mot}' : {freq} fois")
```

```
print("\n" + "="*70)
```

```
# Programme principal
```

```
if __name__ == "__main__":
```

```
    phrase = input("Entrez une phrase à analyser : ")
```

```
analyser_texte(phrase)
```

```
# Exemples de palindromes pour tester
```

```
print("\n 💡 Exemples de palindromes à tester :")
```

```
print(" - 'Elu par cette crapule'")
```

```
print(" - 'Esope reste ici et se repose'")
```

```
print(" - 'kayak'")
```

```
print(" - 'radar'")
```

Solution Exercice 12 : Calculatrice avec Fonctions

```
def addition(a, b):
```

```
    """
```

```
    Additionne deux nombres.
```

```
    Args:
```

```
        a (float): Premier nombre
```

```
        b (float): Deuxième nombre
```

```
    Returns:
```

```
        float: La somme de a et b
```

```
    """
```

```
    return a + b
```

```
def soustraction(a, b):
```

```
    """
```

```
    Soustrait b de a.
```

```
    Args:
```

a (float): Nombre de départ

b (float): Nombre à soustraire

Returns:

float: La différence a - b

"""

return a - b

def multiplication(a, b):

"""

Multiplie deux nombres.

Args:

a (float): Premier nombre

b (float): Deuxième nombre

Returns:

float: Le produit de a et b

"""

return a * b

def division(a, b):

"""

Divise a par b.

Args:

a (float): Numérateur

b (float): Dénominateur

Returns:

float: Le quotient a / b

None: Si division par zéro

"""

if b == 0:

print("⚠ Erreur : Division par zéro impossible")

return None

return a / b

def puissance(a, b):

"""

Calcule a élevé à la puissance b.

Args:

a (float): Base

b (float): Exposant

Returns:

float: $a ** b$

"""

return a ** b

def moyenne(*nombres):

"""

Calcule la moyenne de n'importe quel nombre d'arguments.

Args:

*nombres: Nombre variable d'arguments numériques

Returns:

float: La moyenne des nombres

None: Si aucun nombre fourni

"""

if len(nombres) == 0:

print("⚠️ Aucun nombre fourni")

return None

return sum(nombres) / len(nombres)

def calculer_tout(a, b):

"""

Effectue toutes les opérations et retourne un dictionnaire.

Args:

a (float): Premier nombre

b (float): Deuxième nombre

Returns:

dict: Dictionnaire contenant tous les résultats

"""

resultats = {

"addition": addition(a, b),

"soustraction": soustraction(a, b),

"multiplication": multiplication(a, b),

"division": division(a, b),

"puissance": puissance(a, b)


```
}
```

```
return resultats
```

```
def afficher_menu():
```

```
    """Affiche le menu des opérations disponibles."""
```

```
    print("\n" + "="*50)
```

```
    print("CALCULATRICE".center(50))
```

```
    print("="*50)
```

```
    print("1. Addition")
```

```
    print("2. Soustraction")
```

```
    print("3. Multiplication")
```

```
    print("4. Division")
```

```
    print("5. Puissance")
```

```
    print("6. Moyenne de plusieurs nombres")
```

```
    print("7. Toutes les opérations")
```

```
    print("8. Quitter")
```

```
    print("="*50)
```

```
def programme_principal():
```

```
    """
```

```
    Fonction principale qui gère le menu et les interactions.
```

```
    """
```

```
    print("📊 Bienvenue dans la Calculatrice Python !")
```

```
while True:
```

```
    afficher_menu()
```

```
    choix = input("\nChoisissez une opération (1-8) : ").strip()
```

```
if choix == "8":  
  
    print("\n👋 Merci d'avoir utilisé la calculatrice. Au revoir !")  
  
    break
```

```
if choix not in ["1", "2", "3", "4", "5", "6", "7"]:  
  
    print("⚠️ Choix invalide. Veuillez choisir entre 1 et 8.")  
  
    continue
```

```
try:  
  
    if choix == "6":  
  
        # Moyenne de plusieurs nombres  
  
        nb_nombres = int(input("Combien de nombres ? "))  
  
        nombres = []  
  
        for i in range(nb_nombres):  
  
            nombre = float(input(f"Nombre {i+1} : "))  
  
            nombres.append(nombre)  
  
  
        resultat = moyenne(*nombres)  
  
        if resultat is not None:  
  
            print(f"\n✓ Moyenne de {nombres} = {resultat:.2f}")
```

```
else:  
  
    # Autres opérations (nécessitent 2 nombres)  
  
    a = float(input("Entrez le premier nombre : "))  
  
    b = float(input("Entrez le deuxième nombre : "))  
  
  
    if choix == "1":  
  
        resultat = addition(a, b)
```

```
print(f"\n✓ {a} + {b} = {resultat}")
```

```
elif choix == "2":
```

```
    resultat = soustraction(a, b)
```

```
    print(f"\n✓ {a} - {b} = {resultat}")
```

```
elif choix == "3":
```

```
    resultat = multiplication(a, b)
```

```
    print(f"\n✓ {a} × {b} = {resultat}")
```

```
elif choix == "4":
```

```
    resultat = division(a, b)
```

```
    if resultat is not None:
```

```
        print(f"\n✓ {a} ÷ {b} = {resultat}")
```

```
elif choix == "5":
```

```
    resultat = puissance(a, b)
```

```
    print(f"\n✓ {a} ^ {b} = {resultat}")
```

```
elif choix == "7":
```

```
    resultats = calculer_tout(a, b)
```

```
    print(f"\n=== RÉSULTATS POUR {a} ET {b} ===")
```

```
    print(f"Addition      : {resultats['addition']}")
```

```
    print(f"Soustraction   : {resultats['soustraction']}")
```

```
    print(f"Multiplication : {resultats['multiplication']}")
```

```
    if resultats['division'] is not None:
```

```
        print(f"Division       : {resultats['division']}")
```

```
    print(f"Puissance      : {resultats['puissance']}")
```

```

except ValueError:

    print("⚠ Erreur : Veuillez entrer des nombres valides")


except Exception as e:

    print(f"⚠ Une erreur s'est produite : {e}")


# Demander si continuer

continuer = input("\nVoulez-vous effectuer un autre calcul ? (oui/non) : ").lower()

if continuer != "oui":

    print("\n👋 Au revoir !")

    break


# Point d'entrée du programme

if __name__ == "__main__":

    programme_principal()

```

(Les solutions continuent pour les exercices 13-16 dans la partie suivante en raison de la longueur...)

Solution Exercice 13 : Générateur de Mots de Passe

Fichier : generateur_mdp.py

```
"""
```

Module de génération et gestion de mots de passe.

Auteur: Cours Python

```
"""
```

```
import random
```

```
import string
```

```
from datetime import datetime
```

```
import os
```

```
def generer_mdp_simple(longueur=12):
```

```
    """
```

Génère un mot de passe simple avec lettres et chiffres.

Args:

longueur (int): Longueur du mot de passe (défaut: 12)

Returns:

str: Mot de passe généré

```
    """
```

```
    caracteres = string.ascii_letters + string.digits
```

```
    mot_de_passe = ''.join(random.choice(caracteres) for _ in range(longueur))
```

```
    return mot_de_passe
```

```
def generer_mdp_fort(longueur=16):
```

```
    """
```

Génère un mot de passe fort avec lettres, chiffres et symboles.

Args:

longueur (int): Longueur du mot de passe (défaut: 16)

Returns:

str: Mot de passe fort généré

```
    """
```

```
    if longueur < 8:
```

```

longueur = 8 # Minimum recommandé

# Garantir au moins un de chaque type
mdp = [
    random.choice(string.ascii_uppercase),
    random.choice(string.ascii_lowercase),
    random.choice(string.digits),
    random.choice(string.punctuation)
]

# Compléter avec des caractères aléatoires
tous_caracteres = string.ascii_letters + string.digits + string.punctuation
mdp.extend(random.choice(tous_caracteres) for _ in range(longueur - 4))

# Mélanger
random.shuffle(mdp)

return ''.join(mdp)

def evaluer_force(mdp):
    """
    Évalue la force d'un mot de passe.

    Args:
        mdp (str): Mot de passe à évaluer

    Returns:
        tuple: (force, score) où force est 'Faible', 'Moyen' ou 'Fort'
    """

```

et score est un entier sur 100

"""

score = 0

commentaires = []

Longueur

longueur = len(mdp)

if longueur >= 12:

score += 25

elif longueur >= 8:

score += 15

else:

commentaires.append("Trop court (minimum 8 caractères)")

Majuscules

if any(c.isupper() for c in mdp):

score += 15

else:

commentaires.append("Aucune majuscule")

Minuscules

if any(c.islower() for c in mdp):

score += 15

else:

commentaires.append("Aucune minuscule")

Chiffres

if any(c.isdigit() for c in mdp):

```

        score += 15
    else:
        commentaires.append("Aucun chiffre")

# Symboles
if any(c in string.punctuation for c in mdp):
    score += 20
else:
    commentaires.append("Aucun symbole")

# Diversité des caractères
types_uniques = len(set(mdp))
if types_uniques > longueur * 0.7:
    score += 10

# Déterminer la force
if score >= 80:
    force = "Fort"
elif score >= 50:
    force = "Moyen"
else:
    force = "Faible"

return force, score, commentaires

def sauvegarder_mdp(mdp, nom_service, fichier="mots_de_passe.txt"):
    """
    Sauvegarde un mot de passe dans un fichier avec horodatage.

```


Args:

mdp (str): Mot de passe à sauvegarder

nom_service (str): Nom du service associé

fichier (str): Nom du fichier de sauvegarde

Returns:

bool: True si succès, False sinon

"""

try:

maintenant = datetime.now().strftime("%Y-%m-%d %H:%M:%S")

with open(fichier, "a", encoding="utf-8") as f:

f.write(f"{maintenant} | {nom_service} | {mdp}\n")

return True

except Exception as e:

print(f"Erreur lors de la sauvegarde : {e}")

return False

def lire_historique(fichier="mots_de_passe.txt"):

"""

Lit l'historique des mots de passe sauvegardés.

Args:

fichier (str): Nom du fichier à lire

Returns:

list: Liste des entrées d'historique

"""

```
if not os.path.exists(fichier):
```

```
    return []
```

```
try:
```

```
    with open(fichier, "r", encoding="utf-8") as f:
```

```
        lignes = f.readlines()
```

```
    return [ligne.strip() for ligne in lignes]
```

```
except Exception as e:
```

```
    print(f"Erreur lors de la lecture : {e}")
```

```
    return []
```

Tests du module

```
if __name__ == "__main__":
```

```
    print("=== Tests du module generateur_mdp ===\n")
```

Test mot de passe simple

```
mdp_simple = generer_mdp_simple(10)
```

```
print(f"Mot de passe simple : {mdp_simple}")
```

```
force, score, commentaires = evaluer_force(mdp_simple)
```

```
print(f"Force : {force} ({score}/100)")
```

```
print(f"Commentaires : {commentaires}\n")
```

Test mot de passe fort

```
mdp_fort = generer_mdp_fort(16)
```

```
print(f"Mot de passe fort : {mdp_fort}")
```

```
force, score, commentaires = evaluer_force(mdp_fort)
```

```

print(f"Force : {force} ({score}/100)")

print(f"Commentaires : {commentaires}\n")

# Test évaluation

test_mdps = ["abc123", "Abc123!@#", "M0tD3P@ss3TresF0rt!2024"]

for mdp in test_mdps:

    force, score, _ = evaluer_force(mdp)

    print(f"{mdp:30} → {force:8} ({score}/100)")

```

Fichier : main.py

```

"""

```

Programme principal du générateur de mots de passe.

```

"""

```

```

import generateur_mdp as gmdp

```

```

from datetime import datetime

```

```

def afficher_menu():

```

```

    """Affiche le menu principal."""

```

```

    print("\n" + "="*60)

```

```

    print("GÉNÉRATEUR DE MOTS DE PASSE".center(60))

```

```

    print("="*60)

```

```

    print("1. Générer un mot de passe simple")

```

```

    print("2. Générer un mot de passe fort")

```

```

    print("3. Évaluer un mot de passe existant")

```

```

    print("4. Sauvegarder un mot de passe")

```

```

    print("5. Afficher l'historique")

```

```

    print("6. Quitter")

```

```

    print("="*60)

```

```

def generer_simple():
    """Interface pour génération simple."""
    try:
        longueur = int(input("Longueur souhaitée (défaut 12) : ") or "12")
        mdp = gmdp.generer_mdp_simple(longueur)

        print(f"\n✓ Mot de passe généré : {mdp}")

        # Évaluer automatiquement
        force, score, commentaires = gmdp.evaluer_force(mdp)
        print(f"Force : {force} ({score}/100)")
        if commentaires:
            print(f"Suggestions : {', '.join(commentaires)}")

        # Proposer de sauvegarder
        sauvegarder = input("\nVoulez-vous sauvegarder ce mot de passe ? (oui/non) : ")
        if sauvegarder.lower() == "oui":
            service = input("Nom du service : ")
            if gmdp.sauvegarder_mdp(mdp, service):
                print("✓ Mot de passe sauvegardé")

    except ValueError:
        print("⚠ Veuillez entrer un nombre valide")

def generer_fort():
    """Interface pour génération forte."""
    try:

```

```

longueur = int(input("Longueur souhaitée (défaut 16, min 8) : ") or "16")

mdp = gmdp.generer_mdp_fort(longueur)

print(f"\n✓ Mot de passe généré : {mdp}")

# Évaluer automatiquement
force, score, commentaires = gmdp.evaluer_force(mdp)
print(f"Force : {force} ({score}/100)")
if commentaires:
    print(f"Suggestions : {' '.join(commentaires)}")

# Proposer de sauvegarder
sauvegarder = input("\nVoulez-vous sauvegarder ce mot de passe ? (oui/non) : ")
if sauvegarder.lower() == "oui":
    service = input("Nom du service : ")
    if gmdp.sauvegarder_mdp(mdp, service):
        print("✓ Mot de passe sauvegardé")

except ValueError:
    print("⚠ Veuillez entrer un nombre valide")

def evaluer_existant():
    """Évalue un mot de passe existant."""
    mdp = input("Entrez le mot de passe à évaluer : ")

    force, score, commentaires = gmdp.evaluer_force(mdp)

    print("\n" + "="*60)

```

```
print("ÉVALUATION".center(60))  
  
print("="*60)  
  
print(f"Mot de passe : {'*' * len(mdp)} (longueur: {len(mdp)})")  
  
print(f"Force : {force}")  
  
print(f"Score : {score}/100")
```

```
if commentaires:
```

```
    print("\nPoints à améliorer :")
```

```
    for commentaire in commentaires:
```

```
        print(f"    • {commentaire}")
```

```
else:
```

```
    print("\n✓ Excellent mot de passe !")
```

```
  
print("="*60)
```

```
def sauvegarder():
```

```
    """Sauvegarde un mot de passe."""
```

```
    service = input("Nom du service : ")
```

```
    mdp = input("Mot de passe : ")
```

```
  
    if gmdp.sauvegarder_mdp(mdp, service):
```

```
        print("✓ Mot de passe sauvegardé avec succès")
```

```
    else:
```

```
        print("X Échec de la sauvegarde")
```

```
  
def afficher_historique():
```

```
    """Affiche l'historique des mots de passe."""
```

```
    historique = gmdp.lire_historique()
```

```

if not historique:

    print("\n△ Aucun historique disponible")

    return


print("\n" + "="*80)

print("HISTORIQUE DES MOTS DE PASSE".center(80))

print("="*80)

print(f"{'Date':<20} {'Service':<30} {'Mot de passe':<30}")

print("-"*80)


for ligne in historique:

    parties = ligne.split(" | ")

    if len(parties) == 3:

        date, service, mdp = parties

        # Masquer partiellement le mot de passe

        mdp_masque = mdp[:3] + '*' * (len(mdp) - 6) + mdp[-3:] if len(mdp) > 6 else '*' *
len(mdp)

        print(f"{date:<20} {service:<30} {mdp_masque:<30}")


print("="*80)

print(f"Total : {len(historique)} mots de passe sauvegardés")


def main():

    """Fonction principale."""

    print("🔒 Bienvenue dans le Générateur de Mots de Passe Sécurisé")

    print(f>Date : {datetime.now().strftime('%d/%m/%Y %H:%M')}")

```

```
while True:

    afficher_menu()

    choix = input("\nVotre choix : ").strip()

    if choix == "1":

        generer_simple()

    elif choix == "2":

        generer_fort()

    elif choix == "3":

        evaluer_existant()

    elif choix == "4":

        sauvegarder()

    elif choix == "5":

        afficher_historique()

    elif choix == "6":

        print("\n👋 Au revoir et restez en sécurité !")

        break

    else:

        print("⚠ Choix invalide")

    input("\nAppuyez sur Entrée pour continuer...")
```

```
if __name__ == "__main__":

    main()
```

La suite des solutions pour les exercices 14, 15 et 16 sera dans un document complémentaire en raison de la longueur. Voulez-vous que je continue avec ces dernières solutions ?

