REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE SUPERIEURE EN SCIENCES APPLIQUEES
-TLEMCEN-

وزارة التعليم العالي والبحث العلمي

المدرسة العليا في العلوم التطبيقية
-تلمسان-

Programmation python
Séries de TD

Département du second cycle

1ere année Electronique, option informatique industrielle

Semestre 1

Par : KARAOUZENE Zoheir

Maitre assistant à l'ESSA-Tlemcen

Email: zoheir_karaouzene@yahoo.fr

Chers étudiants,

Il me fait plaisir de vous présenter ce recueil de séries de Travaux Dirigés (TD) accompagné d'exercices résolus, spécialement conçu pour les étudiants de première année en électrotechnique, option informatique industrielle. Ce document a pour objectif de vous accompagner dans votre apprentissage en vous fournissant des exercices pratiques et des solutions détaillées, afin de renforcer vos compétences et votre compréhension dans ce domaine passionnant.

Contenu du Document :

1. Séries de TD : Les séries de TD sont organisées de manière progressive, couvrant les différents concepts et thématiques essentiels à votre formation en électrotechnique et en informatique industrielle. Chaque série est soigneusement élaborée pour vous permettre de développer vos compétences de manière cohérente et structurée.

2. Exercices Résolus : Pour chaque série de TD, vous trouverez une sélection d'exercices résolus, présentés de manière claire et détaillée. Ces solutions servent non seulement de référence pour vérifier vos réponses, mais aussi de guide pour comprendre les méthodes de résolution et les concepts sous-jacents.

Objectifs Pédagogiques :

- Renforcer votre compréhension des principes fondamentaux de l'électrotechnique et de l'informatique industrielle.
- Développer vos compétences analytiques et votre capacité à résoudre des problèmes concrets..

Utilisation du Document :

Ce document est conçu pour être un outil d'apprentissage flexible et adaptatif. Vous pouvez l'utiliser de différentes manières :

- En tant que support lors des séances de TD pour approfondir les concepts abordés en classe.
- En tant que ressource d'entraînement pour vous exercer régulièrement et consolider vos acquis.
- En tant que référence pour préparer vos examens et évaluer votre niveau de maîtrise des différents sujets.

Conclusion :

Ce recueil de séries de TD avec exercices résolus est un précieux compagnon pour votre parcours académique en électrotechnique, option informatique industrielle. Je vous encourage vivement à l'exploiter pleinement et à en tirer le meilleur parti possible pour atteindre vos objectifs d'apprentissage. N'oubliez pas que la persévérance et l'engagement sont les clés du succès dans vos études. Je vous souhaite beaucoup de succès dans votre parcours universitaire

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE SUPERIEURE EN SCIENCES APPLIQUEES
-TLEMCEN-

وزارة التعليم العالي والبحث العلمي

المدرسة العليا في العلوم التطبيقية
-تلمسان-

Département du second cycle **1ere année Electronique**

Python

TD1 : Introduction à Python

Exercice 1

Écrivez un programme Python qui affiche "Hello, World!" à l'écran. Modifiez le programme pour demander à l'utilisateur son nom, puis affichez un message de salutation personnalisé**.**

Exercice 2

Écrivez un programme qui demande à l'utilisateur deux nombres, puis effectuez les opérations suivantes : addition, soustraction, multiplication et division. Affichez les résultats.

Exercice 3

Déclarez une variable nom avec votre nom, une variable age avec votre âge, et une variable ville avec votre ville de résidence.

Affichez ces trois variables à l'écran en utilisant la fonction print().

Calculez et affichez l'année de naissance en utilisant l'âge.

Exercice 4

Écrivez un programme Python qui permet à l'utilisateur de saisir les informations suivantes sur un voyage en voiture :

La distance totale du voyage en kilomètres.

La consommation de carburant de la voiture en litres par 100 kilomètres.

Le coût du litre de carburant.

Ensuite, utilisez ces informations pour calculer et afficher le coût total du voyage.

Exercice 5

Écrivez un programme Python qui calcule la somme des nombres pairs de 1 à n, où n est un nombre entier introduit par l'utilisateur.

Exercice 6

Écrivez un programme Python qui calcule la factorielle d'un nombre entier que l'utilisateur entre.

Exercice 7

Écrivez un programme Python qui demande à l'utilisateur d'introduire une chaine de caractère, le programme affiche ensuite la longueur de cette chaine.

Exercice 8

Créez un programme en Python visant à déterminer le Plus Grand Commun Diviseur (PGCD) de deux nombres entiers.

Exercice 9

Écrivez un programme Python qui prend une liste d'entiers initialisée dans le programme et affiche tous les nombres pairs de cette liste.

Solution (Exercice 1 et 2)

Exercice 1 :

```
# Afficher "Hello, World!"
print("Hello, World!")

# Demander à l'utilisateur son nom
nom = input("Quel est votre nom ? ")

# Afficher un message de salutation personnalisé
print(f"Bonjour, {nom} !")
```

Exercice 2

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

# Effectuer l'addition
addition = nombre1 + nombre2

# Effectuer la soustraction
soustraction = nombre1 - nombre2

# Effectuer la multiplication
multiplication = nombre1 * nombre2

# Vérifier si le deuxième nombre n'est pas zéro avant de faire la division
if nombre2 != 0:
    division = nombre1 / nombre2
else:
    division = "Division par zéro impossible"

# Afficher les résultats
print(f"Addition : {nombre1} + {nombre2} = {addition}")
print(f"Soustraction : {nombre1} - {nombre2} = {soustraction}")
print(f"Multiplication : {nombre1} * {nombre2} = {multiplication}")
print(f"Division : {nombre1} / {nombre2} = {division}")
```

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE SUPERIEURE EN SCIENCES APPLIQUEES
-TLEMCEN-

وزارة التعليم العالي والبحث العلمي

المدرسة العليا في العلوم التطبيقية
-تلمسان-

Département du second cycle **1ere année Electronique**

Python

TD2 : Les fonctions en python

Exercice 1

1-Écrivez une fonction somme_liste qui prend une liste de nombres en entrée et renvoie la somme de ces nombres.

2- Ecrivez une autre fonction qui retourne la somme des éléments pairs et la somme des éléments impaires d'une liste d'entiers donnée en argument

Exercice 2

Écrivez une fonction est_pair qui prend un nombre en entrée et renvoie True s'il est pair et False s'il est impair.

Exercice 3

Écrivez une fonction récursive factorielle qui calcule la factorielle d'un nombre donné.

Exercice 4

Écrivez une fonction affiche_info_personne qui prend en entrée un nom et un âge (avec un âge par défaut de 18) et affiche une phrase de la forme "Ahmed a 25 ans."

Exercice 5

Écrivez une fonction nombres_premiers qui prend un nombre entier n en entrée et renvoie une liste de tous les nombres premiers jusqu'à n.

Exercice 6

Ecrivez une fonction en Python qui prend en paramètres une chaîne de caractères ainsi que deux indices i et j, et renvoie la partie de cette chaîne allant du caractère à l'indice i jusqu'au caractère à l'indice j inclus

Exercice 7

Écrivez une fonction en Python qui prend une chaîne de caractères comme seul argument. Cette fonction choisira aléatoirement deux indices i et j dans la chaîne et échangera les caractères correspondants à ces indices. La fonction doit retourner la nouvelle chaine

Exercice 8

Élaborez une fonction en Python qui a pour but d'afficher la date actuelle.
Créez une autre fonction qui affiche le jour de la semaine

Solutions (Exercice 1 et 8)
Exercice1
1-
```
def somme_liste(liste):
    somme = 0
    for nombre in liste:
        somme += nombre
    return somme
```

2-
```
def somme_liste_pair_impair(liste):
    sPairs = 0
    sImpaires=0
    for nombre in liste:
        if nombre%2==0:
            sPairs += nombre
        else :
            sImpaires+=nombre

    return sPairs,sImpaires
```

Exercice 8 :
1-
```
import datetime

def afficher_date_aujourdhui():
    date_aujourdhui = datetime.date.today()
    return date_aujourdhui

# Appeler la fonction pour afficher la date d'aujourd'hui
date = afficher_date_aujourdhui()
print("Date d'aujourd'hui :", date)
```

2-

```
import datetime

def afficher_jour_actuel():
    # Obtenir la date actuelle
    date_actuelle = datetime.date.today()

    # Obtenir le jour de la semaine sous forme de texte (par exemple, "lundi", "mardi", etc.)
    jour_de_la_semaine = date_actuelle.strftime("%A")

    return jour_de_la_semaine

# Appeler la fonction pour afficher le jour actuel
jour = afficher_jour_actuel()
print("Le jour actuel est :", jour)
```

L'instruction `date_actuelle.strftime("%A")` permet d'obtenir le nom complet du jour de la semaine (en anglais) sous forme de texte. Si vous souhaitez obtenir des formats différents, voici quelques autres options couramment utilisées :

1. `%a` : Obtient l'abréviation du jour de la semaine (en anglais) en trois lettres. Par exemple, "Mon" pour lundi.

2. `%d` : Obtient le jour du mois sous forme de nombre, avec un zéro initial si nécessaire. Par exemple, "05" pour le 5ème jour du mois.

3. `%m` : Obtient le mois sous forme de nombre, avec un zéro initial si nécessaire. Par exemple, "02" pour février.

4. `%Y` : Obtient l'année sous forme de nombre à quatre chiffres. Par exemple, "2023" pour l'année 2023.

5. `%B` : Obtient le nom complet du mois (en anglais) sous forme de texte. Par exemple, "February" pour février.

Si vous souhaitez afficher le jour de la semaine en français, vous pouvez utiliser la locale française avec `locale.setlocale(locale.LC_TIME, 'fr_FR')` de la bibliothèque locale (donc vous devez ajouter import locale), puis utiliser `%A` pour obtenir le nom complet du jour en français. Pour d'autres formats en français, vous pouvez utiliser `%a` pour l'abréviation, `%d` pour le jour du mois, `%m` pour le mois, `%Y` pour l'année, ou `%B` pour le mois complet en français.

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE SUPERIEURE EN SCIENCES APPLIQUEES
-TLEMCEN-

وزارة التعليم العالي والبحث العلمي

المدرسة العليا في العلوم التطبيقية
-تلمسان-

Département du second cycle **1ere année Electronique**

Python

TD3 : Listes, Tuples et Dictionnaires

Exercice1 :

1. Créez une liste contenant trois fruits de votre choix. Affichez chaque fruit en utilisant une boucle `for`.

2. Ajoutez un nouveau fruit à la liste créée dans la première question.

3. Supprimez le deuxième fruit de la liste.

4. Inversez l'ordre des éléments dans la liste.

5. Créez une nouvelle liste vide, puis copiez les éléments de la première liste dans la nouvelle liste sans utiliser la méthode `copy()`.

Exercice 2 :

1. Créez un tuple contenant les six premiers mois de l'année. Essayez d'ajouter un nouveau mois au tuple. Pourquoi cela ne fonctionne-t-il pas ?

2. Accédez au troisième mois dans le tuple et affichez-le.

3. Créez un deuxième tuple contenant le reste des mois.

4. Créez un nouveau tuple en combinant deux tuples existants.

5. Convertissez un tuple des entiers en liste, apportez des modifications à la liste, puis convertissez-la à nouveau en tuple.

5. Parcourez le tuple des mois et affichez les mois qui contiennent la lettre 'r'.

Exercice 3 :

1. Créez un dictionnaire représentant un étudiant avec des clés telles que "nom", "prénom", "âge", et "note_moyenne".

2. Ajoutez une nouvelle clé "matricule" au dictionnaire et attribuez-lui une valeur.

3. Affichez toutes les clés du dictionnaire.

4. Supprimez la clé "note_moyenne" du dictionnaire.

5. Créez une liste de dictionnaires représentant plusieurs étudiants, puis affichez les noms et âges de tous les étudiants.

Exercice 4

Soit la liste des dictionnaires suivante (chaque dictionnaire représente un étudiant)

```
students = [
    {'nom': 'Hamdan', 'prenom': 'Akram', 'date_naissance': '1998-05-15'},
    {'nom': 'Saad', 'prenom': 'Amine', 'date_naissance': '2000-03-20'},
    {'nom': 'Boukhel', 'prenom': 'Sanae', 'date_naissance': '1999-08-10'},
]
```

1- Donner la fonction qui permet de classer les étudiants en ordre croissant dans la même liste
2- Donner la fonction qui permet de classer les étudiants en ordre croissant dans une liste intermédiaire
3- Comment ajouter un nouvel étudiant ?

Exercice 5

1. Créez une liste de noms d'étudiants (liste_noms) avec au moins 5 noms.
2. Créez un tuple de notes (tuple_notes) avec les notes correspondantes pour chaque étudiant (assurez-vous que le tuple est de la même longueur que la liste des noms).
3. Créez un dictionnaire (dict_etudiants) qui associe chaque nom à sa note.
4. Affichez la liste des noms d'étudiants avec leurs notes correspondantes
5. Trouvez et affichez le nom de l'étudiant avec la meilleure note

Solutions (exercice 1)

Exercice 1

1. Créez une liste contenant 5 fruits de votre choix. Affichez chaque fruit en utilisant une boucle `for`.

```
fruits = ["pomme", "banane", "orange", "fraise", "raisin"]

for fruit in fruits:
    print(fruit)
```

2. Ajoutez un nouveau fruit à la liste créée dans la première question.

```
fruits.append("kiwi")
```

3. Supprimez le deuxième fruit de la liste.

```
del fruits[1]
```

4. Inversez l'ordre des éléments dans la liste.

```
fruits.reverse()
```

5. Créez une nouvelle liste vide, puis copiez les éléments de la première liste dans la nouvelle liste sans utiliser la méthode `copy()`.

```
nouvelle_liste = []
for fruit in fruits:
    nouvelle_liste.append(fruit)
```

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
الجمهورية الجزائرية الديمقراطية الشعبية

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

ECOLE SUPERIEURE EN SCIENCES APPLIQUEES
-TLEMCEN-

وزارة التعليم العالي والبحث العلمي

المدرسة العليا في العلوم التطبيقية
-تلمسان-

Département du second cycle **1ere année Electronique**

Python

TD4 : Les interfaces graphiques

Exercice1 :

Créez une application en utilisant Tkinter qui permet à l'utilisateur de saisir des éléments dans un champ de texte. Les éléments saisis doivent être ajoutés à une liste. Ajoutez deux boutons : "Ajouter" pour ajouter un élément à la liste et "Afficher" pour afficher les éléments de la liste dans un label.

Instructions :
1. Créez une fenêtre principale avec un champ de texte (Texte) , deux boutons ("Ajouter" et "Afficher"), et un label.
2. Utilisez une fonction pour gérer l'ajout d'un élément à la liste lorsque le bouton "Ajouter" est cliqué.
3. Utilisez une fonction pour afficher les éléments de la liste dans le label lorsque le bouton "Afficher" est cliqué.
4. Assurez-vous de gérer les cas où le champ de texte est vide lors de l'ajout à la liste.
L'instruction : Texte.delete(0, tk.END) permet d'effacer le champ de saisie après l'ajout
5. Personnalisez l'interface utilisateur selon vos préférences en termes de mise en page, de couleurs, etc.

Conseils :
- Utilisez les classes Tkinter telles que `Entry`, `Button`, et `Label` pour créer les widgets.
- Utilisez une liste pour stocker les éléments ajoutés.
- Vous pouvez utiliser la méthode `join()` pour créer une chaîne à partir des éléments de la liste à afficher dans le label.
- N'oubliez pas de lancer la boucle principale de Tkinter avec `fenetre.mainloop()`.

Exercice 2 :

Créez une application en utilisant Tkinter qui affiche une fenêtre au centre de l'écran. La fenêtre doit contenir un label affichant un message de bienvenue.

Instructions :

1. Créez une fenêtre Tkinter.
2. Utilisez une fonction pour déterminer les dimensions de la fenêtre nécessaire pour la centrer sur l'écran.
3. Placez un label à l'intérieur de la fenêtre affichant un message de bienvenue.
4. Personnalisez le message de bienvenue et l'apparence de la fenêtre selon vos préférences.

Conseils :

- Utilisez la méthode `geometry` pour définir les dimensions et la position de la fenêtre.
- Vous pouvez utiliser la méthode `winfo_screenwidth()` et `winfo_screenheight()` pour obtenir les dimensions de l'écran.
- N'oubliez pas de lancer la boucle principale de Tkinter avec `fenetre.mainloop()`.

Solution :

Exercice 1 :

```
import tkinter as tk

def ajouter_element():
    element = Texte.get()
    if element:  # S'assurer que l'entrée n'est pas vide
        liste_elements.append(element)
        Texte.delete(0, tk.END)  # Effacer le champ de saisie après l'ajout

def afficher_liste():
    label_result.config(text=','.join(liste_elements))

# Création de la fenêtre principale
fenetre = tk.Tk()
fenetre.title("Saisie de liste Tkinter")

# Champ de texte pour saisir un élément
Texte = tk.Entry(fenetre)
Texte.pack(pady=10, padx=10)

# Bouton pour ajouter un élément à la liste
ajouter_button = tk.Button(fenetre, text="Ajouter", command=ajouter_element)
ajouter_button.pack(pady=5, padx=10)

# Bouton pour afficher la liste
afficher_button = tk.Button(fenetre, text="Afficher", command=afficher_liste)
afficher_button.pack(pady=5, padx=10)

# Liste pour stocker les éléments
liste_elements = []

# Label pour afficher la liste
label_result = tk.Label(fenetre, text="")
label_result.pack(pady=10, padx=10)

# Lancement de la boucle principale
fenetre.mainloop()
```

Département du second cycle **1ere année Electronique**

Python

TD5 : La gestion des exceptions en Python

Exercice 1 :

Écrivez un programme qui demande à l'utilisateur de saisir deux nombres, puis effectue la division du premier par le deuxième. Utilisez **try** et **except** pour gérer les erreurs potentielles, telles que la division par zéro ou une saisie invalide.

Exercice 2 :

Écrivez un programme qui crée une liste de nombres et demande à l'utilisateur de saisir l'index d'un élément qu'il souhaite afficher. Utilisez **try** et **except** pour gérer les erreurs si l'index est en dehors des limites de la liste.

Exercice 3 :

Écrivez un programme qui utilise un dictionnaire pour stocker des informations sur des personnes (nom, âge, adresse, etc.). Demandez à l'utilisateur de saisir le nom d'une personne, puis affichez les détails correspondants à cette personne. Utilisez **try** et **except** pour gérer les erreurs si le nom n'est pas présent dans le dictionnaire.

Exemple de représentation des éléments :

```
informations_personnes = {
   "Anis": {"âge": 25, "adresse": "123 rue A, Ville B"},
   "Omar": {"âge": 30, "adresse": "456 rue C ville D"}
}
```

Exercice 4 :

Écrivez un programme qui utilise une liste de dictionnaires pour stocker des informations sur des personnes (nom, âge, adresse, etc.), chaque personne est stocker par un dictionnaire. Demandez à l'utilisateur de saisir l'index de la personne dans la liste, puis affichez les détails correspondants à cette personne. Utilisez **try** et **except** pour gérer les erreurs si le nom n'est pas présent dans le dictionnaire.

Exemple de la liste :

```
informations_personnes = [{
   "nom":"Lamia","âge": 19, "adresse": "ORAN"},
   {"nom":"Ahmed", "âge": 30, "adresse": "ALGER"}
]
```

Solution (exercice 1 et 2)

Exercice 1 :

```
try:
    num1 = float(input("Entrez le premier nombre : "))
    num2 = float(input("Entrez le deuxième nombre : "))
    resultat = num1 / num2
    print("Le résultat de la division est :", resultat)
except ZeroDivisionError:
    print("Erreur : Division par zéro.")
except ValueError:
    print("Erreur : Entrée invalide. Veuillez saisir des nombres.")
```

Exercice 2 :

```
liste_nombres = [10, 20, 30, 40, 50]

try:
    index_demande = int(input("Entrez l'index de l'élément à afficher : "))
    valeur = liste_nombres[index_demande]
    print(f"L'élément à l'index {index_demande} est {valeur}.")
except IndexError:
    print("Erreur : Index hors limites.")
except ValueError:
    print("Erreur : Veuillez entrer un index valide (entier).")
```