Travaux Pratiques

Structures de Contrôle Avancées en Python

if...else, match...case, for, while

M. DIALLO
L2 Informatique
muustafa.dllo@gmail.com

2025

Objectifs du TP

Ce TP vise à maîtriser les structures de contrôle fondamentales en Python de manière approfondie :

- Les structures conditionnelles imbriquées
- Les boucles complexes et imbriquées
- Les algorithmes mathématiques avancés
- La manipulation de structures de données
- L'optimisation des structures de contrôle

Consignes Générales

- Respectez les bonnes pratiques de programmation (noms de variables explicites, commentaires)
- Testez vos programmes avec plusieurs jeux de données
- Gérez les cas d'erreur quand c'est pertinent
- Utilisez l'indentation correcte (4 espaces)
- Optimisez vos algorithmes pour éviter les calculs inutiles

1 Exercice 1 : Vérificateur de dates

Écrivez un programme qui demande à l'utilisateur de saisir une date sous la forme JJ/MM/AAAA et vérifie si la date est valide. Le programme doit prendre en compte :

- Les années bissextiles (divisible par 4 mais pas par 100, ou divisible par 400)
- Le nombre correct de jours pour chaque mois
- Les dates historiques (après 1582 pour le calendrier grégorien)

Exemple d'exécution :

```
Entrez une date (JJ/MM/AAAA) : 29/02/2023
Date invalide : fevrier 2023 n'a pas 29 jours.

Entrez une date (JJ/MM/AAAA) : 29/02/2024
Date valide !
```

2 Exercice 2 : Solveur d'équation du second degré

Créez un programme qui résout une équation du second degré de la forme $ax^2+bx+c = 0$. Le programme doit :

- Gérer tous les cas (a = 0, discriminant négatif/nul/positif)
- Afficher les solutions sous forme exacte quand possible
- Donner les solutions complexes quand le discriminant est négatif
- Gérer les erreurs de saisie

Exemple d'exécution :

```
Coefficient a : 1
Coefficient b : -3
Coefficient c : 2
L'equation a deux solutions reelles :
x1 = 2.0
x2 = 1.0
```

3 Exercice 3 : Conjecture de Syracuse

La suite de Syracuse est définie par :

- Si n est pair : n = n/2
- Si n est impair : n = 3n + 1

Écrivez un programme qui :

1. Calcule la suite de Syracuse pour un nombre donné
2. Trouve le temps de vol (nombre d'étapes pour arriver à 1)
3. Trouve l'altitude maximale (plus grand nombre atteint)
4. Pour tous les nombres de 1 à N, trouve celui qui a le plus long temps de vol

Exemple d'exécution :

```
Nombre de depart : 27
Suite : 27 -> 82 -> 41 -> 124 -> ... -> 1
Temps de vol : 111 etapes
Altitude maximale : 9232
```

4 Exercice 4 : Cryptage par décalage (Chiffre de César)

Implémentez un programme de cryptage/décryptage par décalage (chiffre de César). Le programme doit :

— Demander à l'utilisateur s'il veut crypter ou décrypter

— Demander un décalage (clé)

— Traiter un message complet (conserver les espaces et la ponctuation)

— Gérer les majuscules et minuscules optional : Inclure une option de force brute pour décrypter sans connaître la clé

Exemple d'exécution :

```
Voulez-vous (C)rypter ou (D)ecrypter ? C
Entrez le decalage : 3
Entrez le message : Hello World
Message crypte : Khoor Zruog
```

5 Exercice 5 : Tri par sélection et insertion

Sans utiliser la fonction `sorted()` ou la méthode `sort()`, implémentez deux algorithmes de tri :

1. Tri par sélection
2. Tri par insertion

Le programme doit :

— Générer une liste aléatoire de N nombres

— Afficher la liste non triée

— Trier avec les deux algorithmes

— Afficher les listes triées

— Comparer le nombre d'opérations effectuées par chaque algorithme

Exemple d'exécution :

```
Liste non triee : [64, 34, 25, 12, 22, 11, 90]
Tri par selection (7 elements) : 21 comparaisons
Tri par insertion (7 elements) : 14 comparaisons
```

6 Exercice 6 : Jeu de devinette

Créez un jeu où :

— L'ordinateur choisit un nombre aléatoire entre 1 et 100

— L'utilisateur a 7 tentatives pour deviner

— À chaque tentative, indiquez si le nombre est "trop grand" ou "trop petit"

— Affichez un message de victoire ou de défaite

Modules à importer : `import random`

7 Exercice 7 : Triangle de Pascal

Générez et affichez le triangle de Pascal jusqu'à la ligne n (donnée par l'utilisateur).
Rappel : Chaque nombre est la somme des deux nombres au-dessus de lui.
Exemple pour n=5 :

```
    1
   1 1
  1 2 1
 1 3 3 1
1 4 6 4 1
```

8 Exercice 8 : Système de notes

Créez un système de gestion de notes avec un menu permettant de :

1. Ajouter une note
2. Afficher toutes les notes
3. Calculer la moyenne
4. Trouver la note maximale et minimale
5. Compter les notes par mention (Très Bien >= 16, Bien >= 14, Assez Bien >= 12, Passable >= 10, Insuffisant < 10)
6. Quitter le programme

Utilisez une boucle `while` pour le menu principal et `match...case` pour traiter les choix.

9 Exercice 9 : Nombres premiers

Écrivez un programme qui :

1. Détermine si un nombre donné est premier
2. Affiche tous les nombres premiers jusqu'à n
3. Trouve les n premiers nombres premiers
4. Décompose un nombre en facteurs premiers

Optimisation : Utilisez `break` pour arrêter les vérifications dès qu'un diviseur est trouvé.

10 Exercice 10 : Générateur de motifs

Créez un programme qui génère différents motifs selon le choix de l'utilisateur :
Motif 1 - Carré d'étoiles :

```
* * * * *
* * * * *
* * * * *
* * * * *
* * * * *
```

Motif 2 - Triangle rectangle :

```
*
* *
* * *
* * * *
* * * * *
```

Motif 3 - Losange :

```
  *
 * *
* * *
 * *
  *
```

Motif 4 - Sapin de Noël :

```
    *
   * *
  * * *
   * *
  * * *
 * * * *
  * * *
 * * * *
* * * * *
    *
    *
```

Utilisez `match...case` pour le choix du motif et des boucles imbriquées pour la génération.

11 Exercice 11 : Simulation de machine à états finis

Implémentez une machine à états finis qui reconnaît les nombres binaires :

— Divisible par 3

— Divisible par 5

— Divisible par 3 et 5 (divisible par 15)

Le programme doit lire un nombre binaire (chaîne de caractères de 0 et 1) et déterminer à quelle(s) catégorie(s) il appartient.

Exemple d'exécution :

```
Entrez un nombre binaire : 1111  (15 en decimal)
1111 est divisible par 3
1111 est divisible par 5
1111 est divisible par 15
```

12 Exercice 12 : Approximation de π par la méthode Monte Carlo

Implémentez l'approximation de π par la méthode Monte Carlo :

1. Générez N points aléatoires dans un carré de côté 2
2. Comptez combien tombent dans le cercle inscrit de rayon 1
3. Calculez $\pi \approx 4 \times \frac{\text{points dans le cercle}}{\text{total points}}$
4. Affichez l'approximation et l'erreur relative
5. Tracez le graphique des points (optionnel avec matplotlib)

Exemple d'exécution :

```
Nombre de points : 1000000
Approximation de pi : 3.141592
Erreur relative : 0.0000003%
```

13 Exercice 13 : Recherche dichotomique avancée

Implémentez une recherche dichotomique qui fonctionne sur :

1. Une liste triée de nombres
2. Une liste triée de chaînes de caractères
3. Une fonction monotone (trouver x tel que f(x) = c)

Le programme doit :

— Générer/charger les données
— Permettre la recherche d'un élément
— Afficher le nombre d'itérations nécessaires
— Gérer les cas où l'élément n'est pas trouvé
— Mesurer le temps d'exécution pour différentes tailles de données

Exemple d'exécution :

```
Liste de 1000000 elements generee.
Recherche de l'element : 42
Element trouve a l'indice 419999
Recherche effectuee en 20 iterations
Temps d'execution : 0.0001 secondes
```

Conseils

— Planifiez vos algorithmes sur papier avant de coder
— Testez avec des cas limites (valeurs extrêmes, cas d'erreur)
— Utilisez des fonctions pour modulariser votre code
— Pour les exercices mathématiques, vérifiez vos résultats avec des calculs manuels
— Optimisez votre code seulement après avoir une version fonctionnelle
— Documentez vos algorithmes avec des commentaires

Bonne programmation !