LYCÉE Jules Haag CPGE PT / PTSI Votre avenir avec un temps d'avance	**TP d'Informatique** Rappel sur les structures de contrôle - Liste – Tri à bulles	**TP2**

Partie du programme :
1. Programme du premier semestre
1.1 Rappels des notions de programmation en langage python vues au lycée

L'objectif de ce TP est de transposer des algorithmes de résolution d'un problème posé, en script écrit dans le langage de développement python.

Ressources **:**
- **Cours 2 : Listes et Tuples en langage Python**
- **Cours 3 : Structures de contrôle**

1. Rappels :

Les **structures de contrôle** décrivent l'enchaînement des instructions. **Elles permettent des traitements séquentiels, conditionnels ou répétitifs (itératifs).**

Il existe **3 types de structures** de contrôle :

- **Conditionnelles : « Si …. Alors … Sinon…. Ou sinon si…. »**

La conditionnelle permet d'exécuter une séquence d'instruction, seulement si une condition est vraie. La condition est nécessairement une expression booléenne. Naturellement, il est possible d'imbriquer les structures de contrôle conditionnelles les unes à l'intérieur des autres.

Exemples :

```
if condition:
    sequence
suite
```

```
if condition:
    sequence1
else:
    sequence2
suite
```

```
if condition1:
    sequence1
elif condition2:
    sequence2
elif condition3:
    sequence3
else:
    sequence4
suite
```

On peut ajouter autant de clause « Sinon Si » (**elif** en Python, contraction de else if) que de conditions à tester si les précédentes ont renvoyé faux.

- **La répétition « Tant Que » :**

La répétition « Tant Que » permet d'exécuter une séquence d'instructions tant qu'une condition est vraie.

```
while condition:
    sequence
suite
```

- **La répétition « Pour Chaque » :**

La répétition « Pour Chaque » permet d'exécuter une séquence d'instructions un nombre connu de fois, pour chaque élément...

```
for element in iterable:
    instructions
suite
```

Si on connaît avant de démarrer la boucle le nombre d'itérations à exécuter, on choisit une **boucle for.**
Au contraire, si la décision d'arrêter la boucle ne peut se faire que par un test, on choisit une boucle **while**.

Dans un premier temps, nous allons répéter une séquence n fois. Pour cela, nous allons itérer sur un **range** (ou gamme, portée, échelle). Un range est défini par trois nombres entiers (int) : **un début, une fin et un pas**. Il contient tous les entiers, à partir de début (inclus), jusqu'à fin (exclus), en avançant d'un pas.

```
list(range(4,10,2))                  # range(début, fin , pas) : 4,6,8
[4, 6, 8]

list(range(4,10))                  # range(début, fin) pas=1 : 4,5,6,7,8,9,10
[4, 5, 6, 7, 8, 9]

list(range(10))                  # range(fin) début=0 pas=1 : 1,2,3,4,5,6,7,8,9,10
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

Le pas est par défaut 1

- Le début est par défaut 0
- Le pas est par défaut 1

```
for i in range(n):
    print(i**2)
```

En pratique, dans IDLE, vous créez un nouveau script (*Menu File, puis New File*) que vous enregistrez sous un nom « nom de votre script ».py sur votre clef usb. Une fois rédigé, le programme sera exécuté en cliquant sur *Run module* dans le menu *Run* ou en appuyant sur F5. Le résultat sera automatiquement affiché dans la fenêtre de l'interpréteur.

2. Utilisation de la structure conditionnelle : Si …. Alors … Sinon…. Ou sinon si….

2.1. Lecture

Indiquer dans les programmes suivants, à chaque ligne commençant par #, ce que fait l'instruction qui suit :
(Vous pouvez aussi les tester avec IDLE)

Exemple 1 :

```
from math import sqrt
# ..........................................................................
# ..........................................................................
print("\n donner un nombre : ")
x = float(input("x ? "))

# ..........................................................................
# ..........................................................................
if x >= 0:
    y = sqrt(x)
    print("La racine de {:.2f} est {:.3f}".format(x, y))
else:
    print("On ne peut pas prendre la racine d'un reel negatf !")

print("Au revoir")
```

Infos pratique :
en début de ligne
Commentaires dans le programme

dans un print
\n est un saut de ligne
\t est une tabulation

Exemple 2 :

```
#..........................................................................
#..........................................................................
print("\n quel âge avez-vous ?")
a=int(input("a = "))

#..........................................................................
#..........................................................................
if a < 18 :
    print("votre place de cinéma coûte 4€")
elif a>=18 and a<25 :
    print("votre place de cinéma coûte 5€")
else:
    print("votre place de cinéma coûte 7€")
```

Exemple 3 :

```
#..........................................................................
#..........................................................................
p_seuil, v_seuil = 2.3, 7.41
#..........................................................................
#..........................................................................
print("\nSeuil pression : {} bars \t Seuil volume :{} cm3 \n".format(p_seuil,v_seuil))
#..........................................................................
#..........................................................................
pression = float(input("Pression courante = "))
volume = float (input("Volume courant = "))
#..........................................................................
#..........................................................................
#..........................................................................
#..........................................................................
if (pression > p_seuil) and (volume > v_seuil):
    print ("pression ET volume trop élevés. Stoppez !")
elif pression > p_seuil :
    print ("Il faut diminuer la pression avant d'augmenter le volume")
elif volume > v_seuil :
    print ("Vous devez diminuer le volume")
else:
    print("Tout va bien !")
```

2.2. Écriture

PROGRAMME 1 : RÉSOLUTION D'UNE ÉQUATION DU 2ND DEGRÉ

1. Créer un script resol_degre2.py
2. Écrire un programme qui donne les racines réelles d'une équation du second degré, comme suit :
   - Demande à l'utilisateur la saisie des coefficients a, b, c
   - Calcule le discriminant
   - Suivant les cas, affiche les résultats ou la mention « l'équation n'a pas de solution réelle ».

3. Utilisation de la répétition « Tant Que » :

3.1. Lecture

Indiquer dans le programme suivant, à chaque ligne commençant par #, ce que fait l'instruction qui suit :
(Vous pouvez aussi les tester avec IDLE)
Exemple 4 :

```
#.....................................................................
a, b = 0, 10
#.....................................................................
#.....................................................................
#.....................................................................
while a < b:
    print(a, end=" ")
    a = a + 1

print("\n\nAutre exploitation :")
#.....................................................................
#.....................................................................
#.....................................................................
while b:
    b = b - 1
    if b % 2 != 0:
        print(b, end=" ")
print()
```

Infos pratique :
% calcule le reste d'une opération de division

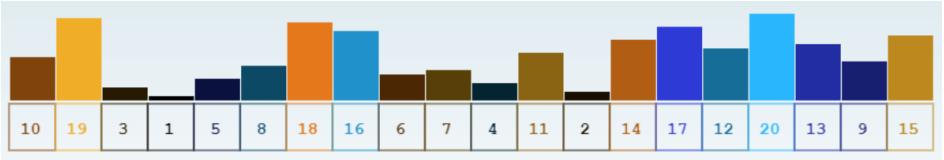
3.1. Écriture

PROGRAMME 2 : SAISIE NOMBRE

1. Créer un script saisie_nbre.py
2. Écrire un programme qui demande la saisie d'un entier tant qu'il ne se trouve pas dans l'intervalle [1,10]. Affichez la saisie quand elle est correcte.

PROGRAMME 3 : RECHERCHE D'UN MAXIMUM DANS UNE LISTE DE NOMBRES

1. Créer un script maximum.py
2. Créer la liste suivante : L1=[10,19,3,1,5,8,18,16,6,7,4,11,2,14,17,12,20,13,9,15]

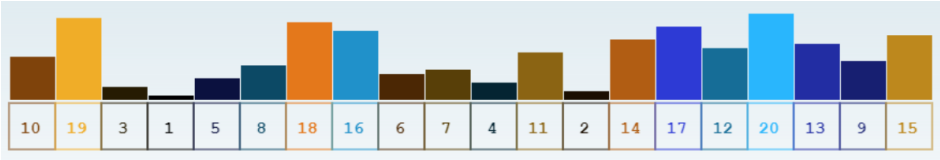


3. Écrire un programme qui renverra le maximum de cette liste en comparant les éléments de manière consécutive en utilisant des structures de contrôle conditionnelle. Tester votre programme.
4. Modifier votre programme pour indiquer dans le même temps l'indice de ce maximum dans la liste initiale. Si plusieurs nombres sont égaux au maximum, le programme renverra l'indice le plus petit. Tester en remplaçant le nombre 5 par le nombre 20.

PROGRAMME 4 : TRI À BULLES OU PAR PROPAGATION D'UNE LISTE

Le tri à bulles consiste à ordonner (dans l'ordre croissant en général) deux éléments consécutifs d'une liste ou d'un tableau en les comparant et d'effectuer permutation si besoin. On réitère l'opération tant qu'une permutation est possible.

1. Créer un script tri_bulle_liste.py
2. Reprendre la liste suivante : L1=[10,19,3,1,5,8,18,16,6,7,4,11,2,14,17,12,20,13,9,15]



3. Écrire un début de programme qui permet de trier (ordre croissant) cette liste par la méthode de tri à bulles seulement pour un parcours de la liste (sans itération).
4. Terminer le tri à bulles en réitérant votre 1er tri jusqu'à ce que la liste soit complètement ordonnée (ordre croissant).

4. Utilisation de la répétition « Pour Chaque » :

4.1. Lecture

Indiquer dans le programme suivant, à chaque ligne commençant par #, ce que fait l'instruction qui suit :
(Vous pouvez aussi les tester avec IDLE)

Exemple 5 :

```
#.....................................................................
#.....................................................................
for i in range(1, 11):
    if i == 5:
        continue
    print(i, end=" ")
print()

#.....................................................................
#.....................................................................
for i in range(1, 11):
    if i == 5:
        break
    print(i, end=" ")
print()
```

Exemple 6 :

```
from math import sin
# programme principal ------------------------------------------
#.....................................................................
#.....................................................................
#.....................................................................
#.....................................................................
for x in range(-3, 4):
    try:
        print("\n (sin({}))/({})={:.3f}".format(x,x,sin(x)/x))
    except :
        print("\n on ne divise pas par zéro... ")
print()
```

Exemple 7 :

```
#.....
#.....
#.....
#.....
#.....
#.....
for n in range(2, 8):
    for x in range(2, n):
        if n % x == 0:
            print(n, "egale", x, "*", n/x)
            break
    else:
        print(n, "est un nombre premier")
```

4.2. Écriture

PROGRAMME 5 : MOYENNE DES NOMBRES D'UNE LISTE

1. Créer un script moyenne.py

   Les notes des étudiants d'une classe à un devoir sont les suivantes :

   17,10,14,12,15,4,13,18,15,5,7,20,5,9,19,8,10,11,6,4,3,7
2. Écrire un programme qui affiche la moyenne des résultats de la classe, initialement saisie dans une liste.
3. Compléter ce programme pour qu'il affiche aussi l'écart-type.

   On rappelle que l'écart-type de (x_1,x_2,…,x_n) est $\sigma = \sqrt{\frac{1}{n}\sum_{k=1}^{n} x_k^2 - \left(\frac{1}{n}\sum_{k=1}^{n} x_k\right)^2}$
4. Compléter ce programme qui détermine et affiche le nombre d'étudiants qui ont eu la moyenne.

PROGRAMME 6 : PUISSANCE MOTEUR ÉLECTRIQUE CC

1. Créer un script puissance.py

Des mesures de tension et d'intensité aux bornes d'un moteur qui démarre sont données ci-dessous :

Tension=[0.424,0.490,0.530,0.570]

Intensité=[1.162,2.449,3.307,5.053]

2. Écrire un script qui créer une liste appelée puissance puis qui affiche son contenu. Utilisez l'écriture formatée pour afficher la valeur de chaque puissance avec deux décimales.

Aide pratique pour créer une liste :

- La syntaxe nom_liste.append(a) permet de rajouter l'élément « a » à la fin de la liste nom_liste.
- Lorsque la longueur de la liste à créer est connue à l'avance, il est possible de créer une liste vide par nom_liste=[0]*n, puis de remplacer son contenu en la parcourant.

5. Exercices supplémentaires :

Exercice 1 : trou

Un technicien de laboratoire utilise un robot pipeteur utilisant un plateau horizontal contenant 200 trous disposés sur 10 lignes et 20 colonnes.
La machine indique au technicien qu'il doit sortir le tube dans le trou cible 8, 3 : colonne 8 et ligne 3.
La machine l'aide à atteindre le tube pour le saisir.

1. Créer un script robot.py
2. Écrire un programme qui aide le technicien à atteindre le trou cible :
   a. Le technicien choisit une colonne, puis une ligne (sur le système réel il approche sa main)
   b. Votre programme envoie un message avec une couleur au technicien :
      - « Vert », si le technicien a trouvé le trou cible sur le plateau
      - « Orange », si le technicien a choisi un trou sur la même ligne ou la même colonne que le trou cible.
      - « Rouge », sinon.

Exercice 2 : recherche nombre

1. Créer un script recherche_nbre.py
2. Écrire un programme qui propose de trouver un nombre entre 1 et 1000 par essais successifs en affichant si le nombre saisi est trop grand ou trop petit.
   - Le nombre à trouver est dans un premier temps fixé par une valeur littérale dans le code.
     - Le programme s'exécute tant que l'utilisateur n'a pas trouvé le bon résultat
     - Le programme affiche au fur et à mesure le nombre de tentatives effectuées
     - Le programme limite à 10 tentatives et sort avec un message d'erreur si on dépasse les 10 essais.
   - Remplacer la valeur à deviner par une valeur aléatoire.

Aide pratique :
La fonction **random.randint(1,1000)** permet de générer un nombre aléatoire entier compris entre 1 et 1000

Exercice 3 : table multiplication

1. Créer un script table_multi.py
2. Écrire un programme qui affiche sous forme d'un tableau les tables de multiplication au format **5 x 3 = 15** organisé en 10 lignes (1 à 10) et 11 colonnes (0 à 10)
3. Modifier le programme pour qu'il demande le nombre de ligne et de colonnes avant d'afficher le tableau

Exercice 4 : Opération sur des listes

On considère les deux listes :

- L1 = [i for i in range(5)]
- L2 = [i**2 for i in range(8) if (i**2)%3==1] # si le reste de la division de i^2 par 3 vaut 1.

1. Afficher la concaténation des deux listes : L1+L2. Vous noterez dans votre programme en commentaires, ce qu'effectue cette opération.
2. Créer et afficher une nouvelle liste L3, dont les termes correspondent à la somme terme à terme de deux listes L1 et L2, à condition qu'elles soient de même taille.
3. Afficher le produit 3*L1. Vous noterez dans votre programme en commentaires, ce qu'effectue cette opération.
4. Créer et afficher une nouvelle liste L4, dont les termes correspondent aux termes de la liste L1 multipliés par un facteur k. Vous testerez pour k=3.
5. Créer une liste L5 dont les termes correspondent aux termes classés dans le sens inverse de la liste L1.
6. Créer et afficher une liste L3' construite comme L3 mais avec L2 et L5.
7. Afficher la valeur maxi de L5.