

TD-TP n° 3

Cette feuille d'exercices suit le cours 3 et reprend aussi des notions des cours précédents.

Selon les indications données, les exercices sont à traiter

- en mode TD (papier, crayon),
- en mode TP (Thonny, jupyter-notebook)

Exercice 1. (TD) Analyse de code

Dans cet exercice, vous disposez du code d'un programme, mais vous ne pouvez pas le recopier pour l'exécuter.

```
1 # DONNÉES / ENTRÉES
2 a = int(input("entrez a : "))
3 b = int(input("entrez b : "))
4 c = int(input("entrez c : "))
5
6 # CALCUL
7 if c < a and c == b :
8     d = 1
9 else :
10     if c < a or c == b :
11         d = 2
12     else :
13         d = 3
14         if b < c :
15             d = 4
16         else :
17             d = 5
18     d = a + d
19
20 # RÉSULTAT / SORTIE
21 print("d =", d)
```

1. Faites la trace de l'exécution du programme dans le cas où l'utilisateur entre les valeurs 7, puis 3, puis 5.
2. Donnez un exemple d'entrées pour lesquelles le programme affiche **d = 1**.
3. Donnez un exemple d'entrées pour lesquelles le programme affiche **d = 10**.
4. Donnez un exemple d'entrées pour lesquelles le programme affiche une valeur de **d** négative.

Exercice 2. (TD) Analyse de code 2

Même contexte, avec le code donné en page suivante.

1. Faites la trace de l'exécution du programme dans le cas où l'utilisateur entre les valeurs 100, puis 6, puis 20.
2. Donnez un exemple d'entrées pour lesquelles le programme affiche **P**.

3. Même question pour **Q**, **R** et **S**.

```
1 # DONNÉES / ENTRÉES
2 a = int(input("entrez a : "))
3 b = int(input("entrez b : "))
4 c = int(input("entrez c : "))
5
6 # CALCUL, RÉSULTAT / SORTIE
7 if a < c :
8     if not a < b :
9         print('P')
10    else :
11        print('Q')
12 else :
13     if b < c and b < a :
14         print('R')
15     else :
16         print('S')
17
```

Exercice 3. (TD) Analyse de code 3

Voici quatre programmes.

```
1 # programme 1
2 note = int(input("note : "))
3 if 10 <= note :
4     print("reçu")
5 else :
6     print("ajourné")
7
```

```
1 # programme 2
2 note = int(input("note : "))
3 if 10 <= note :
4     print("reçu")
5 if note <= 10 :
6     print("ajourné")
7
```

```
1 # programme 3
2 note = int(input("note : "))
3 if 10 < note :
4     print("reçu")
5 if note < 10 :
6     print("ajourné")
7
```

```
1 # programme 4
2 note = int(input("note : "))
3 if 10 <= note :
4     print("reçu")
5 if note < 10 :
6     print("ajourné")
7
```

1. Expliquez pourquoi ils ne font pas la même chose.
2. Quel programme est à privilégier ? Argumentez.

Exercice 4. (TD-TP) Remise pour gros achats

Un client achète **qte** exemplaires d'un produit dont le prix unitaire est **pu**. Le nombre d'exemplaires **qte** est entier. Le prix unitaire **pu** n'a pas de raison d'être entier.

Les 50 premiers exemplaires d'une commande sont facturés au prix normal, les 50 exemplaires suivants bénéficient d'une remise de 10%, et les autres exemplaires (à partir du 101ème) bénéficient d'une remise de 20%.

1. Exprimez, en fonction de **pu**, le prix payé par le client pour **qte** = **17**, **50**, **94** et **123**.
2. Écrivez un programme qui demande les valeurs de **qte** et **pu** puis qui calcule et affiche la somme à payer par le client.

3. Testez le programme sur les valeurs précédentes (question 1).

Exercice 5. (TD) Questions sur le cours du matin

- a) Comment les chaîne vide, tuple vide et liste vide sont-elles représentées ?
- b) Quel est, en entier positif ou nul, l'indice du premier caractère d'une chaîne ? du dernier ?
- c) Quel est, en entier négatif, l'indice du premier caractère d'une chaîne ? du dernier ?
- d) On part de **mot** = "algorithmes". Que valent les extractions suivantes :

mot[0]	mot[7]	mot[10]	mot[-1]
mot[-11]	mot[-7]	mot[4 : -3]	mot[-8 : 9]
mot[-7 : -2]	mot[3 :]	mot[: -5]	mot[-5 :]
mot[: 4]	mot[4 : 13]	mot[len(mot) - 1]	mot[-len(mot)]

- e) Quelle est la valeur des expressions suivantes ?
 - a) **mot_1** = **mot**[-5 :8] + **mot**[-2] + **mot**[4] + **mot**[-3 :]
 - b) **mot_2** = **mot_1**[:-2] + **mot**[-11 :-9] + **mot**[5] + **mot**[-1] + **mot**[-3 :10]
 - b) **mot_3** = **mot_2**[4 :6] + **mot_2**[:2] + **mot_2**[-3]

Exercice 6. (TD-TP) Listes

Voici deux programmes.

```
1 # programme 1
2
3 liste = []
4 for i in range(1, 5) :
5     liste = liste + [i]
6
7 print("liste =", liste)
```

```
1 # programme 2
2
3 liste = []
4 for i in range(1, 5) :
5     liste = [liste + [i]]
6
7 print("liste =", liste)
```

A. Sans recopier les programmes sur Thonny ou autre, répondez aux questions suivantes :

- (a) Qu'affiche le programme 1 ?
- (b) Qu'affiche le programme 2 ?
- (c) Pour chaque cas, que vaut len(liste) après exécution du programme ?
- (d) Pour quel(s) programme(s) les expressions suivantes retournent-elles une valeur (et pas une erreur) ? Et dans ce cas, quelle valeur ?
 - a) liste[0]
 - b) liste[1]
 - c) liste[0][0]

B. Cette fois vous pouvez recopier et modifier les programmes pour les tester.
Comment obtenir l'affichage suivant :

```
liste = [1, 2, 2, 3, 3, 4, 4, 5]
```

Exercice 7. (TP) Livret A

On suppose qu'un livret A contient 1500 €. Le taux d'intérêts annuel du livret A est 2.4 %.

Écrivez un programme qui affiche le montant du livret pour chacune des huit prochaines années. Le visuel doit être conforme à :

```
après 1 année(s) : capital = 1536.00 €
après 2 année(s) : capital = 1572.86 €
après 3 année(s) : capital = 1610.61 €
après 4 année(s) : capital = 1649.27 €
après 5 année(s) : capital = 1688.85 €
après 6 année(s) : capital = 1729.38 €
après 7 année(s) : capital = 1770.89 €
après 8 année(s) : capital = 1813.39 €
```

Vous pourrez chercher comment afficher un float avec 2 décimales significatives.

Exercice 8. (TP) Motifs géométriques

1. Écrivez un programme qui demande à l'utilisateur un entier **n** en entrée et affiche une ligne de **n** caractères "*".

Par exemple, pour l'entrée 8, il affichera : `*****`

2. Écrivez un programme qui demande à l'utilisateur un entier **n** et un caractère **c** en entrée et affiche une ligne de **n** caractères semblables au caractère **c**.

Par exemple, pour l'entrée 6 et "A", il affichera : `AAAAAA`

3. Écrivez un programme qui demande à l'utilisateur un entier **n** et un caractère **c** en entrée et affiche un carré de côté **n** composé de **n** lignes de **n** caractères semblables à **c**.

Par exemple, pour l'entrée 4 et X, il affichera :

```
XXXX
XXXX
XXXX
XXXX
```

4. Écrivez un programme qui demande à l'utilisateur un entier **n** et un caractère **c** en entrée et affiche un triangle rectangle comme dans les deux exemples ci-dessous.

Pour l'entrée 4 et X, il affichera

```
X
XX
XXX
XXXX
```

Pour l'entrée 5 et +, il affichera

```

+
++
+++
++++
+++++
```