

TD n° 1

Exercice 1. évaluez les expressions suivantes

1. $3 + 5 * 2 - 7 // 5 * 4$
2. $3 + 5 * 2 - 7 / 5 * 4$
3. $(3 + 5) * 2 - 7 // (5 * 4)$
4. $((3 + 5) * (2 - 7) // 5) * 4$
5. $((((3 + 5) * 2) - 7) // 5) * 4$
6. $3 + (5 * (2 - (7 // (5 * 4))))$

Exercice 2. évaluez les expressions suivantes

1. $1 > 2$ or $4 \% 2 == 0$
2. $1 > 2$ and $4 // 5 == 0$
3. "c" in "char" and "s" not in "str"
4. "a" in "char"[0 : 2] or "r" in "string"[1 : 4]

Exercice 3. Échange

1. Quels affichages sont produits par l'exécution du programme suivant ?

```
1 a = 4
2 b = 7
3 print("a =", a, "et b =", b)
4 a = a + b
5 b = a - b
6 a = a - b
7 print("a =", a, "et b =", b)
```

2. Reprenez la question précédente avec comme valeur initiales $a = 15$ et $b = -3$.
3. Proposez une preuve générale du résultat qui s'est vérifié sur les 2 exemples.
4. Proposez maintenant un programme qui échange les valeurs des variables a et b et utilisant une variable auxiliaire c , et sans aucune opération arithmétique comme aux lignes 4 à 6.

Exercice 4. Permutation

Vous devez maintenant écrire un programme Python permettant une permutation des valeurs de 3 variables (variables a , b et c , de valeurs initiales x , y et z).

Vous devez, en quelques instructions (des affectations du type de celles des lignes 4, 5 et 6 de l'exercice 3), et sans utiliser aucune variable auxiliaire, permuter les valeurs de a , b et c .

Vous pourrez structurer votre démarche dans un tableau, comme le tableau suivant, où les deux premières instructions Python ont été écrites, ainsi que leur effet sur les valeurs des variables.

Il vous reste à compléter les autres lignes, pour avoir les valeurs finales.

x, y et z désignent 3 valeurs entières quelconques

instruction	a	b	c
a = x ; b = y ; c = z	x	y	z
b = a + b	x	x + y	z
	z	x	y

Les exercices à suivre posent des questions basiques d'arithmétique.

Quelques notions de base, qui seront régulièrement utilisées :

- Soit $n \in \mathbb{N}^*$. On dit que l'entier d est un **diviseur** de n si $d \in \mathbb{N}^*$ et si la division de n par d donne un reste nul. Par exemple 6 et 24 sont des diviseurs de 24.
- On parle de diviseur **strict** quand le diviseur en question est strictement inférieur au nombre. Par exemple, les diviseurs stricts de 15 sont 1, 3 et 5 (donc 15 est exclus).
- Un nombre n est dit **premier** si $n > 1$ et si son seul diviseur strict est 1. Par exemple, 17 est premier.

Exercice 5. diviseurs stricts d'un entier

Écrivez un programme qui, à partir d'une certaine valeur de la variable n , construit et affiche la liste des diviseurs stricts de n . Vous utiliserez une structure conditionnelle (if...) et la méthode **append** des listes, vue en CM.

Voici une ébauche du code, à compléter :

```
1 # la valeur de n, à adapter
2 n = 60
3
4 # la liste des diviseurs stricts de n, qui contient initialement 1
5 divStricts = [1]
6
7 # votre code à partir d'ici
8
9
10
11
12
13 # affichage final
14 print("les diviseurs stricts de", n, "sont :", divStricts)
```

Voici deux exemples d'exécution :

```
les diviseurs stricts de 40 sont : [1, 2, 4, 5, 8, 10, 20]
```

```
les diviseurs stricts de 17 sont : [1]
```

Exercice 6. nombre de diviseurs stricts

Modifiez le programme précédent pour qu'il donne aussi le nombre de diviseurs stricts de n . Vous pourrez adapter l'affichage au nombre de diviseurs stricts comme ci-dessous :

```
les diviseurs stricts de 40 sont : [1, 2, 4, 5, 8, 10, 20]
le nombre 40 a 7 diviseurs stricts
```

```
les diviseurs stricts de 17 sont : [1]
le nombre 17 a 1 seul diviseur strict
```

Exercice 7. test de primalité (pour savoir si un nombre est "premier")

Continuez d'enrichir le programme précédent pour qu'il affiche si le nombre n est un nombre premier ou non. S'il n'est pas premier, le programme donnera comme justification la valeur du plus petit diviseur autre que 1.

Voici deux exemples d'exécution :

```
les diviseurs stricts de 15 sont : [1, 3, 5]
le nombre 15 a 3 diviseurs stricts
il n'est pas premier, car il est divisible par 3
```

```
les diviseurs stricts de 13 sont : [1]
le nombre 13 a 1 seul diviseur strict
il est donc premier
```

Exercice 8. essai sur la répartition des nombres premiers

Écrivez un programme qui, à partir d'un entier $n \geq 2$ supérieur ou égal à 2, va calculer et afficher le nombre et la liste des nombres premiers inférieurs ou égaux à n .

Illustration :

```
il y a 6 nombres premiers inférieurs ou égaux à 13
voici leur liste : [2, 3, 5, 7, 11, 13]
```

```
il y a 4 nombres premiers inférieurs ou égaux à 10
voici leur liste : [2, 3, 5, 7]
```

```
il y a 10 nombres premiers inférieurs ou égaux à 29
voici leur liste : [2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
```

Culture : il est démontré que, pour $n \rightarrow +\infty$, le nombre de nombres premiers inférieurs ou égaux à n est équivalent à $\frac{n}{\ln n}$. Vous pouvez, pour les 3 exemples donnés, tester cette équivalence.

Exercice 9. somme des diviseurs

Écrivez un programme qui, à partir d'une valeur donnée $n \geq 2$, affiche la somme des diviseurs stricts de n . Voici deux exemples (vous prendrez le temps de vérifier que ces deux valeurs sont exactes en retrouvant à la main les diviseurs stricts) :

```
la somme des diviseurs stricts de 40 est 50
```

```
la somme des diviseurs stricts de 12 est 16
```

Exercice 10. nombres abondants, déficients et parfaits

Un entier $n \geq 2$ est dit :

- **parfait** s'il est égal à la somme de ses diviseurs stricts (par exemple 6) ;
- **abondant** s'il est inférieur à la somme de ses diviseurs stricts (par exemple 12) ;
- **déficient** s'il est supérieur à la somme de ses diviseurs stricts (par exemple 9 ou 17).

Complétez le programme précédent pour qu'il affiche en plus la catégorie du nombre n . Voici trois exemples :

```
la somme des diviseurs stricts de 12 est 16  
12 est un nombre abondant
```

```
la somme des diviseurs stricts de 10 est 8  
10 est un nombre déficient
```

```
la somme des diviseurs stricts de 6 est 6  
6 est un nombre parfait
```

Exercice 11. les nombres parfaits

Écrivez un programme qui, à partir d'un entier $n \geq 2$ donné, construit et affiche la liste des nombres parfaits inférieurs ou égaux à n . Deux exemples :

```
il y a 4 entiers parfaits inférieurs ou égaux à 10000  
les voici : [6, 28, 496, 8128]
```

```
il y a 3 entiers parfaits inférieurs ou égaux à 500  
les voici : [6, 28, 496]
```

Exercice 12. nombres amis

Deux nombres sont dits amis si chacun d'eux est égal à la somme des diviseurs stricts de l'autre. Vous pouvez par exemple vérifier que 24 et 36 ne sont pas amis, que 20 et 30 ne sont pas amis, mais que 220 et 284 le sont.

On exclura ce cas par la suite, mais de qui un nombre parfait serait-il l'ami ?

Écrivez un programme qui prend en charge deux entiers n et p supérieurs ou égaux à 2, et qui dit oui ou non les deux entiers sont amis.

Deux exemples :

```
16 et 12 ne sont pas amis
```

```
284 et 220 sont amis
```

Exercice 13. nombres amis v2

Améliorez le programme précédent pour qu'il prenne en charge un paramètre entier $n \geq 2$ et qu'il affiche toutes les paires d'entiers amis (donc des tuples) inférieurs ou égaux à n .

Exemple :

```
il y a 5 paires d'amis inférieurs ou égaux à 10000  
voici leur liste : [(220, 284), (1184, 1210), (2620, 2924), (5020, 5564), (6232, 6368)]
```