

## TRAVAUX PRATIQUES

**Exercice 1** – Calculer  $1 - 0.2 - 0.2 - 0.2 - 0.2 - 0.2$ . Que constatez-vous?

**Exercice 2** – Prévoir le type des expressions suivantes et le résultat.

- |  |                                    |
|--|------------------------------------|
| • $2**3.0+4$                             | • $5\%3*5$                         |
| • $2>=3$ or $2**4*5**2//20==20$          | • <code>not False and False</code> |
| • <code>True or 4&gt;3 and 3&gt;4</code> | • <code>float(2)**3</code>         |

**Exercice 3** – On exécute le code suivant, donner les valeurs des variables après exécution.

```
>>> a=5
>>> b=a+2
>>> a=b
>>> c=a==b
>>> a=10
>>> d=a!=b
```

**Exercice 4** – Écrire des expressions booléennes exprimant le fait que les quantités qui suivent sont positives. Vérifiez avec Python si elles s'évaluent en `True` ou `False`. N'oubliez pas d'importer ce dont vous avez besoin de `numpy`.

- |  |  |
|--|--|
| • $\sqrt{1 + \sqrt{1 + \sqrt{1 + 1}}}$ | • $\frac{\sqrt{3}}{2} - \frac{e}{\pi}$ |
|--|--|

**Exercice 5** – Donner une expression booléenne dépendant des variables  $a$ ,  $b$ ,  $c$  et  $d$  exprimant le fait que  $a$  est supérieure ou égale à  $b$ ,  $c$  et  $d$ . Tester éventuellement en affectant préalablement des valeurs aux variables.

**Exercice 6** – Écrire un programme qui fait l'échange de deux variables  $x$  et  $y$ .

**Exercice 7** – Écrire un programme qui permet de calculer la moyenne de trois nombres  $a$ ,  $b$  et  $c$  et qui vous affiche le résultat sous la forme : "La moyenne de 1, 2 et 3 vaut 2.0" (par exemple si  $a = 1$ ,  $b = 2$  et  $c = 3$ .)

**Exercice 8** – Tracer le graphe de la fonction  $x \mapsto x^{-x}$  sur l'intervalle  $[0, 5]$ . Continuer en traçant les fonctions usuelles suivantes :

- La fonction carrée sur l'intervalle  $[-3, 3]$ .
- La fonction cube sur l'intervalle  $[-3, 3]$ .
- La fonction valeur absolue sur l'intervalle  $[-3, 3]$ .
- La fonction racine carrée sur l'intervalle  $[0, 5]$ .
- La fonction inverse sur l'intervalle  $[-2, 2] \setminus \{0\}$ .

## Instructions conditionnelles

**Exercice 9** – On suppose deux variables `taille` et `poids` déjà affectées. L'indice de masse corporelle (abrégié IMC, égal à  $\frac{\text{poids}}{\text{taille}^2}$ ) d'un individu donne une indication sur sa santé. Calculer l'IMC que vous stockerez dans une variable `imc`, et indiquer par un message à l'écran si l'individu est en sur-poids ( $\text{IMC} > 25$ ) ou en sous-poids ( $\text{IMC} < 18$ ).

**Exercice 10** – On suppose que  $a$  et  $b$  sont des variables booléennes. Indiquer les instructions à effectuer pour calculer `not a`, `a and b` et `a or b`, sans utiliser ces opérateurs mais à l'aide d'instructions conditionnelles (le moins possible!). Stocker les résultats dans `non_a`, `a_et_b` et `a_ou_b`.

**Exercice 11** – On suppose  $x$ ,  $y$  et  $z$  affectées, ces variables sont des entiers. Indiquer un code permettant de tester si au moins deux des entiers sont égaux (on affichera un message à l'écran, du style "il y en a deux égaux" ou "ils sont tous distincts").

**Exercice 12** – On suppose que  $a$ ,  $b$  et  $c$  sont des variables contenant trois nombres. Stocker dans la variable  $d$  le plus petit, sans utiliser la fonction `min` de Python.

**Exercice 13** (Suite de l'exercice précédent) – Calculer le triplet  $t$  contenant les trois nombres précédents, mais triés dans l'ordre croissant. Il y a 6 possibilités.

## Boucles

**Exercice 14** – Que vaut  $s$  après la boucle suivante, en supposant les variables  $x$  et  $N$  affectées ( $N$  contient un entier positif) ?

```
>>> s=0
>>> y=1
>>> for i in range(N):
...     s=s+y
...     y=y*x
```

**Exercice 15** – Écrire un algorithme ou un programme affichant la différence de deux carrés consécutifs de  $1^2 - 0^2$  à  $100^2 - 99^2$ . Que remarque-t-on?

**Exercice 16** – À l'aide de boucles `for`, calculer les sommes suivantes :

$$\sum_{i=0}^{100} i^2, \quad \sum_{i=0}^{100} \sum_{j=0}^{100} ij, \quad \sum_{i=0}^{100} \sum_{j=i}^{100} ij \quad \text{et} \quad \sum_{i=0}^{100} \sum_{j=0}^i ij$$

**Exercice 17** – Vérifiez expérimentalement les formules suivantes (on coupera la somme après un "grand" nombre de termes, comme 10000 ou 100000).

- $\frac{\pi^2}{6} = \frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \dots + \frac{1}{n^2} + \dots$
- $\ln(2) = 1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \dots + \frac{(-1)^n}{n+1} + \dots$
- $\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \dots + \frac{(-1)^n}{2n+1} + \dots$

**Exercice 18** – On peut montrer que  $\sum_{i=0}^n i^3 = \frac{n^2(n+1)^2}{4}$ . On suppose la variable `n` affectée, elle contient un entier naturel. Écrire un code permettant de vérifier la proposition (on pourra par exemple afficher un message à l'écran du type "la conjecture est vraie pour  $n =$ ").

**Exercice 19** (Approximation de  $\sqrt{2}$ ) – La suite définie par  $u_0 = 1$  et  $u_{n+1} = \frac{1}{2} \left( u_n + \frac{2}{u_n} \right)$  converge (rapidement) vers  $\sqrt{2}$ . Écrire un script affichant à l'écran les approximations successives de  $\sqrt{2}$  obtenues à l'aide de cette suite (un petit nombre suffit pour atteindre la précision maximale par défaut, moins de dix étapes sont nécessaires). Vérifier que même si mathématiquement, on n'a jamais  $u_n = u_{n+1}$ , ceci se produit avec des flottants à partir d'un certain rang.

**Exercice 20** – La suite de Fibonacci est définie par  $F_0 = 0$ ,  $F_1 = 1$  et pour tout  $i \geq 2$ ,  $F_i = F_{i-2} + F_{i-1}$ . En utilisant d'abord seulement trois variables et une boucle, calculer  $F_{100}$ .

**Exercice 21** – Écrire un code permettant d'afficher à l'écran tous les triplets pythagoriciens (triplets d'entiers  $(a, b, c)$  tels que  $a^2 + b^2 = c^2$ ), où  $1 \leq a \leq b \leq c \leq 1000$ . (Cela peut prendre 4/5 minutes.)

**Exercice 22** – Quelle est la somme des entiers entre 0 et 10000 (tous deux inclus) qui sont divisibles par 3 ou par 5? *Réponse : 23341668.*

**Exercice 23** – Quelle est la somme des entiers entre 0 et 10000 (tous deux inclus) qui sont divisibles par 3 ou par 5 mais pas par 15? *Réponse : 20010003.*

## Fonctions

**Exercice 24** – Une entreprise commercialise deux forfaits téléphoniques :

- Forfait A : 1.20€ par heure d'appel,
- Forfait B : illimité pour 20€ par mois.

Écrire une fonction prenant en entrée le temps d'appel mensuel de l'utilisateur, affichant le prix payé pour chacun des deux forfaits et renvoyant le forfait à privilégier (A ou B).

**Exercice 25** – Sans utiliser `numpy` ni l'opérateur `**`, définir une fonction `absolue` prenant en entrée un entier et retournant sa valeur absolue (on ne demande pas de vérifier que le paramètre passé à la fonction est un entier).

De même, définir une fonction `fact` prenant en entrée un entier positif et retournant sa factorielle définie par  $n! = \prod_{i=1}^n i$ .

Enfin, définir une fonction `puissance` prenant deux arguments  $x$  et  $n$  et retournant  $x^n$ . (On supposera que  $n$  est un entier positif, et on posera  $x^0 = 1$  pour tout réel  $x$ ).

**Exercice 26** – Écrire une fonction `max2(a,b)` prenant en paramètre deux entiers et retournant le maximum des deux. Remarque : cette fonction existe déjà en Python sous le nom `max`, on ne l'utilisera pas ici, on comparera simplement `a` et `b`.

En déduire une fonction `max3(a,b,c)` retournant le maximum de 3 entiers, et utilisant `max2`. Remarque : la fonction `max` de Python peut également être utilisée à la place de `max3`.

**Exercice 27** (La conjecture de Syracuse) – Pour  $u_0 \in \mathbb{N}^*$ , on considère la suite définie par

$$u_{n+1} = \begin{cases} 3u_n + 1 & \text{si } n \text{ est impair,} \\ \frac{u_n}{2} & \text{si } n \text{ est pair.} \end{cases}$$

On conjecture que pour tout  $u_0 \in \mathbb{N}^*$ , il existe un indice  $n$  tel que  $u_n = 1$  (à partir de cet entier  $n$ , la suite prend périodiquement les valeurs 1, 4 et 2).

Écrire une boucle `while` permettant de vérifier cette conjecture pour  $u_0$  fixé. Testez-la! Remarque : on n'a ici besoin que d'une seule variable pour calculer successivement les termes de la suite, qu'on pourra appeler  $u$ . On appelle temps de vol à partir de  $u_0$  le plus petit  $n$  tel que  $u_n$  vaut 1. Calculez ce temps de vol pour  $u_0 = 15$  et  $u_0 = 127$ . On affichera les valeurs de la suite obtenues avant d'atteindre 1, à l'aide de `print`.

**Exercice 28** – Quel nombre inférieur ou égal à  $10^6$  possède le plus grand temps de vol (défini dans l'exercice 27)? Il n'y a qu'une solution.

Réponse : 837799.

**Exercice 29** (Le juste prix!) – On va créer un jeu du *juste prix*, aussi connu sous le nom de "C'est plus, c'est moins". Le principe est le suivant : l'ordinateur choisi un nombre au hasard entre 1 et 1000. Vous essayez de deviner ce nombre, et pour ce faire, vous proposez des entiers au clavier. À chaque proposition, l'ordinateur vous dit si l'entier qu'il a choisi est plus grand ou plus petit que votre proposition. Le jeu s'arrête lorsque vous donnez l'entier choisi, avec un message de félicitations. Tout d'abord, votre programme doit choisir un entier au hasard. On va pour cela importer la fonction `randint` du module `numpy.random`. Le code suivant affiche à l'écran un nombre au hasard entre 1 et 1000.

```
>>> from random.numpy import randint
>>> print(randint(1,1000))
```

Écrire une fonction `nombre_au_hasard()` (ne prenant pas d'argument) qui renvoie un entier aléatoire entre 1 et 1000.

Programmez maintenant le jeu proprement dit, sous la forme d'une fonction `juste_prix()` (ne prenant pas d'argument). Pour demander à l'utilisateur de donner un nombre, vous pouvez utiliser la fonction `input`. Cette fonction renvoie ce que l'utilisateur tape au clavier, sous forme de chaîne de caractères. Pour obtenir un entier, il faut convertir cette chaîne via la fonction `int()`. On peut placer un message comme paramètre à `input()`, sous forme de chaîne de caractères. Le code suivant convertit donc ce que l'utilisateur entre au clavier en entier et le stocke dans la variable `n`.

```
>>> n=int(input("Un nombre entre 1 et 1000 ?"))
```

Voici un exemple de déroulement :

```
>>> juste_prix()
Un nombre entre 1 et 1000 ? 500
C'est plus !
Un nombre entre 1 et 1000 ? 750
C'est plus !
Un nombre entre 1 et 1000 ? 875
C'est plus !
Un nombre entre 1 et 1000 ? 930
C'est plus !
Un nombre entre 1 et 1000 ? 965
C'est moins !
Un nombre entre 1 et 1000 ? 947
C'est moins !
Un nombre entre 1 et 1000 ? 938
C'est plus !
Un nombre entre 1 et 1000 ? 943
Bien joué !
```

**Exercice 30** (Juste prix à l'envers) – À l'inverse, écrivez une fonction `juste_prix_2()`. L'ordinateur vous propose des nombres et vous répondez par "oui", "plus" ou "moins". Dégagez une stratégie "optimale" pour le programme, celui-ci devant être capable de détecter si vous mentez.

```
>>> juste_prix_2()
501 ? moins
251 ? plus
376 ? moins
314 ? plus
345 ? moins
330 ? moins
322 ? plus
326 ? moins
324 ? moins
323 ? moins
Tu as triché !
```