Une introduction à Python 3

Hubert Raymondaud - Yannick Le Bastard

24 novembre 2019

1 Exercices d'initiation

Exercice 0 : Donnez pour chacun des cas le type (ou message d'erreur obtenu) si l'on saisit :

1. type(56+9)
2. type(3*4)
3. type(3/4)
4. type(3.*4)
5. type('5')
6. type('5'+2)
7. type('ABC'*2)
8. type('5'+str(2))
9. type([11,'Hello',56.3])

Exercice 1 : Un étudiant saisit dans le shell (ne le faîtes surtout pas !) les instructions :
>>>a=5
>>>b=2
puis :
>>>a=a+b
>>>b=a-2*b
Question 1 : Quelles valeurs sont affectées à a et à b ?

Se ravisant, il saisit :
>>>a=5
>>>b=2
puis :
>>>a,b=a+b,a-2*b
Question 2 : Quelles valeurs sont affectées à a et à b ?

Exercice 2 : écrire un script qui étant donné un triangle ABC rectangle en A, demande à l'utilisateur de saisir les mesures des côtés AB et AC et renvoie la mesure de l'hypoténuse BC. On invoquera la fonction racine carrée par la commande : from math import sqrt dès la première ligne du script.

Exercice 3 : écrire un script qui donne la table de multiplication par N, où N est un entier naturel saisi par l'utilisateur. Par exemple, pour N=4 il s'affichera :
0 fois 4 = 0
1 fois 4 = 4
⋮
10 fois 4 = 40

Exercice 4 : Sans les saisir sur machine, devinez le résultat renvoyé par les lignes qui suivent.

1. list(range(2,14,3))
2. list(range(14,2,-3))
3. list(range(2,14,-3))
4. list(range(-2,-14,-3))
5. list(range(-2,13,3))

Exercice 5 : En utilisant une ou plusieurs boucles for, calculer :

1. $S = 1 + \frac{1}{2} + \frac{1}{2\times 4} + \frac{1}{2\times 4\times 6} + \cdots + \frac{1}{2\times 4\times\cdots\times 20}$
2. $S = 1 + \cfrac{1}{1+\cfrac{1}{1+\ddots}}$ avec N barres de fractions, où N est saisi par l'utilisateur. Tester avec N=3.
3. $S = \sum_{1\leq i<j\leq 10} ij$
4. La solution X de $LX = Y$, avec L la matrice 4×4 dont la diagonale est constituée de 2, la sous-diagonale de -1 et $Y = (1,1,1,1,1)^T$. Généraliser au cas d'une matrice $N\times N$ constituée des mêmes éléments.

Exercice 6 : En utilisant une ou plusieurs boucles for, calculer :

1. $S = 1 + \frac{1}{2} + \frac{1}{2\times 4} + \frac{1}{2\times 4\times 8} + \cdots + \frac{1}{2\times 4\times\cdots\times 2^N}$. Tester avec N=5,
2. $S = \sum_{i=1}^{9}\prod_{k=1}^{i}\frac{1}{k(k+1)}$,
3. La somme des N premiers termes de la suite de Fibonacci. On rappelle que cette dernière est définie par $u_0 = u_1 = 1$ et pour tout entier $n\in\mathbb{N}$, $u_{n+2} = u_{n+1} + u_n$. Tester avec N=7.
   Indication : pour définir les u_n, penser à l'affectation parallèle.

Exercice 7 : while ou for ou les deux...

1. Calculer $S = 3 + 3\times 2 + 3\times 4 + 3\times 8 + \cdots + 196608$ (si pas de calcul préalable, ne soyez pas for ...)
2. Déterminer le plus petit entier naturel N tel que $\sum_{i=1}^{N}\frac{1}{i}\geq 5$

Exercice 8 : Écrire un script qui demande à l'utilisateur de saisir une chaine de caractères de longueur 10 (l'instruction len(ch) renvoie la longueur de la chaine ch). Tant que cette condition n'est pas vérifiée, l'utilisateur doit recommencer la saisie. En cas de saisie correcte, il s'affichera "Saisie correcte".

Exercice 9 : En conditionnelle...
Préliminaire : le symbole modulo % donne le reste de la division euclidienne de deux entiers naturels a et b. Ainsi 4%2 renvoie 0 tandis que 5%2 renvoie 1. Un entier a est divisible par un entier b (non nul) si a%b est égal à 0.
Une année N est dite *bissextile* si N est un multiple de 4. Elle ne l'est cependant pas si N est un multiple de 100, à moins que N ne soit un multiple de 400.
Écrire un script qui demande à l'utilisateur de saisir une année N et qui renvoie un message annonçant si l'année est bissextile ou non.
Indication : il est recommandé de faire un diagramme pour modéliser les différents cas.

Exercice 10 : Un grand classique qui plait ...
Ouvrir un fichier vierge que vous nommerez devin.py ; saisir en première ligne la commande : from random import randint
L'instruction randint(a,b) permet de générer un entier aléatoire compris entre les bornes entières (incluses) a et b.

1. Écrire un script qui génère un entier aléatoire alea compris entre 1 et 100 et demande à l'utilisateur de le deviner. Tant que ce dernier ne l'aura pas trouvé :
   Si l'utilisateur saisit un entier N trop grand (resp. trop petit), l'ordinateur affichera "plus petit" (resp. "trop grand").
   En cas de victoire, il s'affichera "Gagné", ainsi que le nombre d'essais utilisés.
2. Modifier le script précédent pour limiter le nombre d'essais à 5. Il sera affiché "Perdu" si l'utilisateur n'a pas trouvé le nombre mystère avant ces cinq essais ainsi que le nombre qu'il fallait trouver.

Exercice 11 : Modifier l'exemple de l'ordre alphabétique pour comparer deux chaines de caractères en lettres minuscules de longueurs quelconques. Par exemple, si l'utilisateur saisit 'jeu' et 'jeunesse', il s'affichera jeu $<$ jeunesse.

Exercice 12 : Le but est de convertir un entier naturel compris entre 1 et 3999 en nombre romains (rechercher le principe sur internet). Pour rappel :

un	deux	trois	quatre	cinq	six	sept
I	II	III	IV	V	VI	VII
huit	neuf	dix	cinquante	cent	cinq-cents	mille
VIII	IX	**X**	L	**C**	D	M

Écrire un script répondant à la question. Le tester ensuite sur :
(a) 2397 (b) 3912 (c) 3812 (d) 756

Exercice 13 : Écrire un script qui demande à l'utilisateur de saisir une chaine de caractères ch et qui renvoie :

1. cette chaine écrite à l'envers. Par exemple, BOUCHON deviendra NOHCUOB,
2. Même chose mais 2 par 2 à l'envers. Par exemple, BOUCHON deviendra NHUB.

Exercice 14 : Le codage ASCII permet d'attribuer à un caractère un entier. De même, à chaque entier correspond un caractère. Nous disposons pour ceci de deux fonctions duales : **ord**(caractère) et **chr**(nombre).

1. Testez sous Pyzo les commandes suivantes :
   (a) ord('a') puis chr(97)
   (b) ord('z') puis chr(122)
   (c) ord('A') puis chr(65)
   (d) ord('Z') puis chr(90)
2. Écrire un script qui demande à l'utilisateur de saisir une chaine ch de lettres minuscules non accentuées et qui les convertit en majuscules,
3. Écrire un script qui demande à l'utilisateur de saisir une chaine ch de lettres majuscules non accentuées et qui les convertit en minuscules,
4. Écrire un script qui effectue ces deux opérations quand les casses sont mélangées,
5. Tester ch.upper() pour ch en minuscules et ch.lower() pour ch en majuscules. Vous nous détestez ?

Exercice 15 : Même exercice que le 13 mais avec le slicing (une seule commande suffit ! ! !)

Exercice 16 : Écrire un script qui demande à :

1. la machine de générer un mot aléatoire de 5 voyelles,
2. à l'utilisateur de saisir une voyelle.

Si la voyelle est correcte *i.e* dans le mot choisi par l'ordinateur, elle apparait dans le mot à la place où elle se trouve.
Recommencer jusqu'à temps de trouver le mot complet.
Début du programme :

```
from random import randint

#creation du mot mystere choisi par l'ordinateur
L=['a','e','i','o','u','y'] #liste de voyelles
mot_mystere=""              #mot vide au depart
for i in range(5):
    mot_mystere=mot_mystere+L[randint(0,5)]
Lmot=[""]*5                 #votre mot en liste (pourquoi ?)
```

Exercice 17 : Recréer des méthodes !

1. Re-créer explicitement la méthode min(maliste), qui étant donné une liste L de cinq nombres saisis par l'utilisateur, renvoie le minimum d'entre eux,
2. Re-créer explicitement la méthode maliste.sort(), qui étant donné une liste L de cinq nombres saisis par l'utilisateur, renvoie la liste ordonnée dans l'ordre croissant,
3. **Sans se servir des méthodes sort() et reverse()**, écrire un script qui étant donné une liste L de cinq nombres saisis par l'utilisateur, renvoie la liste ordonnée dans l'ordre décroissant (on autorise la méthode remove()).

Exercice 18 : Comme nous l'avons vu précédemment, la commande : from random import randint permet d'appeler la fonction randint(a,b) qui génère un entier pseudo-aléatoire compris entre a et b inclus.

1. En utilisant cette instruction, générer une liste de cinq entiers aléatoires compris entre 1 et 6 et afficher la liste,

2. Écrire un script qui compte le nombre d'ex-aequo dans la liste précédente et les répertorie (dans l'ordre croissant) dans une nouvelle liste,
3. Écrire un script qui sépare en deux listes : entiers pairs et entiers impairs, les entiers de la liste de la question 1. Les nombres de ces listes seront classés par ordre croissant.

Exercice 19 : Calculer en utilisant le moins de commandes possibles (1 ou 2) les sommes :

1. $S = 1 + 3 + 5 + 7 + \cdots + 201$,
2. $S = \sum_{i=1}^{N} \frac{1}{i^2}$. Tester avec N=15,
3. Sans utiliser le slicing, calculer $S = \sum_{i=4}^{10} \frac{1}{(2i)^2}$ et $T = \sum_{i=4}^{10} \frac{1}{(2i+1)^2}$,
4. La question précédente avec le slicing,
5. Déterminer la norme euclidienne d'un vecteur $(a_1, \ldots, a_N)$ de $\mathbb{R}^N$. On rappelle que $\|(a_1, \ldots, a_N)\| = \left(\sum_{i=1}^{N} a_i^2\right)^{\frac{1}{2}}$.
   Tester avec $\vec{a} = (1, -2, \sqrt{2})$.

Exercice 20 : Listes et chaines de caractères.

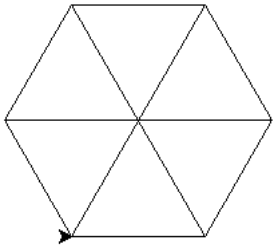
1. Écrire un script Python qui demande à l'utilisateur de saisir une chaine de caractères en minuscules, et qui renvoie ses éléments ordonnés par ordre alphabétique.
   Indication : Penser au codage ASCII
2. Un brin d'ADN est une séquence ordonnée de nucléotides symbolisés par les lettres A (adénine), T (thymine), G (guanine) et C (cytosine). Ces nucléotides peuvent s'apparier mais pas n'importe comment : A et T sont complémentaires ; G et C le sont aussi.
   Écrire un script qui génère un brin aléatoire d'ADN de taille N choisie par l'utilisateur, et qui renvoie sa séquence complémentaire.
   On considérera un brin d'ADN comme une variable de type string. Pour AAATGCATG en entrée, on aura en sortie :
   AAATGCATG
   | | | | | | | | |
   TTTACGTAC

Exercice 21 : Créer un script qui génère une liste aléatoire de cinq consonnes et cinq voyelles mélangées, puis qui renvoie une nouvelle liste alternant consonnes et voyelles dans leur ordre d'apparition.
Par exemple, si la liste générée est : ['a','o','e','f','i','t','r','y','z','z'], on aura en sortie :
['a','f','o','t','e','r','i','z','y','z'].
Indication : la commande shuffle(maliste) permet de permuter aléatoirement les objets de maliste. On l'importe via la commande from random import shuffle

Exercice 22 : Quelques petits dessins ...

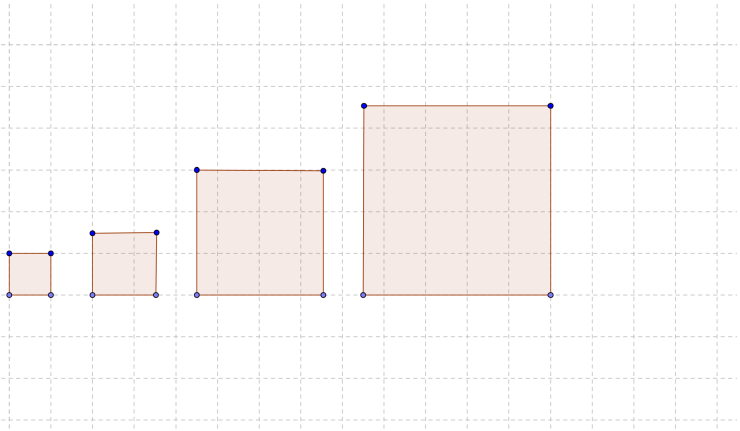
1. Dessiner un carré de côté 80,
2. un pentagone de côté 70,
3. La figure suivante, de côté 100, en utilisant seulement deux boucles :

screenshot001.png

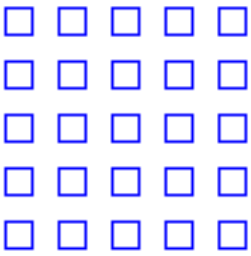


4. dix carrés rouges de côtés 10 dont les bases sont espacées de 10,

5. une série de quatre carrés bleus dont les côtés augmentent de 15 en 15. Le premier carré a un côté de 10 et l'espace entre chaque carré est de 10,



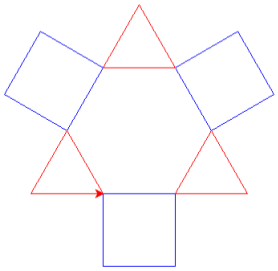
6. La figure suivante (choix de la couleur et de la taille laissées au lecteur) :



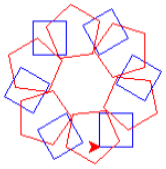
Exercice 23 : Écrire un script Python sous la forme d'une fonction qui permet de dessiner un polygone régulier à N côtés, d'une taille T et rempli de la couleur C.
Se servir de ce script pour créer une fonction motif se basant sur la fonction précédente et qui dessine la figure suivante (la mesure du côté du carré bleu est de 40) :

Dessiner ensuite les figures suivantes :

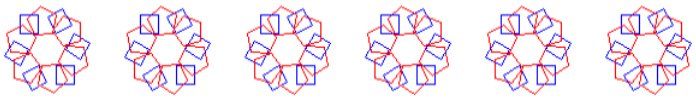
1. carrés et triangles de côté 80



2. carrés et pentagones de côté 30 (indication : bases espacées de 30 horizontalement)



3. Se placer d'abord à $(-250;0)$ puis espacer les motifs de 100 entre eux.



Exercice 24 : Le script suivant amène à un message d'erreur. Corrigez-le afin qu'il fonctionne.

```
# on se donne un R.O.N (O;i;j)et deux points A(xA;yA) et B(xB;yB)
# on souhaite savoir si un point M(xM ; yM) appartient au cercle de diametre [AB]

def saisie():
    xA=float(input("Abscisse de A ? "))
    yA=float(input("Ordonnee de A ? "))
    xB=float(input("Abscisse de B ? "))
    yB=float(input("Ordonnee de B ? "))
    xM=float(input("Abscisse de M ? "))
    yM=float(input("Ordonnee de M ? "))

def distance2(x1,y1,x2,y2):
    return (x1-x2)**2+(y1-y2)**2

def Est_sur_C():
    saisie()
    if distance2(xA,yA,xB,yB)==distance2(xA,yA,xM,yM)+distance2(xM,yM,xB,yB):
        return 1
    else:
        return 0

if Est_sur_C()==1:
    print("M appartient au cercle de diametre [AB]")
else:
    print("M n'appartient pas au cercle de diametre [AB]")
```

Exercice 25 : Nous rappelons qu'un entier naturel $p \geq 2$ est *premier* si ses seuls diviseurs entiers naturels sont 1 et lui-même. Ainsi, 2 est le seul nombre pair qui soit premier. Tous les autres nombres premiers sont impairs.
Deux entiers a et b sont *premiers entre eux* s'ils n'ont pas de diviseur commun autre que 1.
Théorème utile : Un entier naturel $n \geq 2$ est premier si et seulement si il est premier avec tous les entiers $k \in [1; \sqrt{n}]$.
Ceci revient à dire qu'aucun $k \in [1; \sqrt{n}]$ ne divise n.

Nous pouvons prouver, sachant que tout entier $n \geq 2$ a un facteur premier, que l'ensemble des nombres premiers est infini.

1. En vous servant des résultats précédents, créez une fonction $listePremiers(n)$ qui demande à l'utilisateur de saisir un entier naturel $n \geq 2$ et qui renvoie la liste de tous les nombres premiers inférieurs ou égaux à n.
2. Se servir de cette liste pour construire la fonction $decompositionP(n)$ qui demande à l'utilisateur de saisir un entier naturel $n \geq 2$ et qui renvoie sa décomposition en facteurs premiers sous la forme d'une liste.
   Par exemple, si on saisit $n = 24$, la liste renvoyée sera $[2, 2, 2, 3]$.

2 Exercices thématiques

2.1 Géométrie

Exercice x : On se donne quatre points distincts A, B, C et D.

1. Écrire un script Python qui détermine si les points A, B et C sont alignés.
2. Écrire un script Python qui détermine si les droites (AB) et (CD) sont strictement parallèles ou confondues ou sécantes.
3. Soient $A(2; -3)$, $B(1, 4)$, $C(-2; 14)$ et $D(-5; 35)$. Justifier que les droites points (AB) et (CD) sont strictement parallèles.

Exercice y : Écrire un script Python qui étant donnés quatre points A, B, C et D détermine si le quadrilatère ABCD est un trapèze.

Exercice z : Écrire un script Python qui étant donnés quatre points distincts A, B, C et D détermine la nature du quadrilatère ABCD : trapèze, parallélogramme, losange, rectangle, carré, quelconque.

2.2 Probabilités - statistiques

Exercice x : Modéliser le lancer de deux dés équilibrés. On calculera la différence entre le plus grand et le plus petit des deux nombres obtenus, ainsi que la distribution de probabilité approchée de la variable aléatoire D "différence" (effectuer N= 10 000 expériences).
Vous pourrez tracer cette distribution approchée avec matplotlib.

Exercice xx : On lance une pièce truquée. La probabilité de faire pile est de $\frac{1}{3}$. Si l'on fait pile on tire une boule dans une urne contenant 4 boules blanches et 6 boules noires indiscernables au toucher. Si l'on fait face, on tire une boule dans une urne contenant 2 boules blanches et 8 boules noires également indiscernables.
Si au bout du compte on a obtenu une boule blanche, on gagne 5 euros ; sinon on perd 2 euros.

Quel est le gain moyen à ce jeu ?
Vous écrirez pour ceci un script Python modélisant une partie puis simulerez 10 000 parties pour estimer le gain moyen.

Exercice xxx : On lance trois dés parfaits à 6 faces. Si la somme S obtenue est inférieure ou égale à 7, on perd la partie ; si S est supérieure ou égale à 15, on gagne la partie ; sinon on relance les dés.

1. Écrire une fonction unePartie() qui modélise une partie de ce jeu et retourne 1 si gain et 0 si perte ;
2. Écrire une fonction TempsAttente() qui renvoie le nombre de lancers de dés effectués au cours d'une partie jusqu'à temps qu'elle se finisse ;
3. Écrire une fonction frequence(N) qui calcule la fréquence de gain et le temps moyen d'attente sur N parties jouées par l'utilisateur ;
4. En utilisant matplotlib, donner la distribution simulée des variables aléatoires X (0 si perte, 1 si gain d'une partie) et Y : "temps d'attente avant la fin d'une partie". On pourra effectuer N = 50 000 parties.