typage/entrée/sortie

Exo 1 : décrire les spécifications des deux premiers exemples du cours et proposer des jeux de tests.

Exo 2 : proposer un algorithme, organigramme et programme `python`, qui échange deux valeurs entre elles.

Exo 3 : on considère une valeur x connue. Afin de calculer l'image x^3+x^2+x on a besoin de 5 opérations (rq $x^3 = x \times x \times x$ correspond à 2 opérations). Proposer un calcul qui requiert moins d'opérations (il s'agit de la méthode de Ruffini-Horner).

Exo 4 : À partir du code IMC fournit en cours,

- proposer un programme `python` avec moins de lignes de code.
- préciser la spécification et un jeu de test pour le programme du calcul d'IMC.
- dérouler dans un tableau le programme sur votre jeu de tests
- proposer un algorithme, organigramme et programme `python` qui demande la taille avant de calculer l'IMC.

Exo 5 : Proposer un programme `python` qui demande la longueur d'un côté d'un carré et renvoie son aire. De même pour le volume d'un cube.

Exo 6 : En s'inspirant de l'exercice précédent, on souhaite créer un programme qui demande l'âge de l'utilisateur et calcule son âge en 2 050. Proposer une spécification, un algorithme, un organigramme, un code `python` ainsi qu'un jeu de tests.

Exo 7 : Écrire un programme `python` qui demande un nombre de jours et effectue la conversion en année (composée de 365j) et jours restants. Proposer un jeu de tests.

Rq : `a//b` renvoie le quotient de la division euclidienne de a par b en `python`, ainsi $15//7 = 2$. Comment obtenir le reste ?

Exo 8 : Écrire un programme `python` qui demande un nombre de secondes et effectue la conversion en jours, heures, minutes et secondes. Proposer un jeu de tests.

test conditionnel

Exo 9 : Proposer un code `python` qui demande un nombre et précise s'il est pair.

Rq : `a%b` renvoie le reste de la division euclidienne de a par b en `python`, ainsi $16\%3 = 1$ car $16 = 3\times5+1$.

Exo 10 : Proposer un code `python` qui demande l'âge de l'utilisateur et affiche "Bonjour" s'il a plus de 18 ans, "Salut" sinon. Écrire l'organigramme correspondant, proposer un jeu de tests et dérouler un exemple.

Exo 11 : Proposer un code `python` qui demande les coefficients d'un point $M(x;y)$ et vérifie si M appartient à la parabole d'équation $y = x^2$. Proposer un organigramme, un jeu de tests et un déroulé.

Exo 12 : Proposer un code `python` qui demande un nombre et qui vérifie si ce nombre est compris entre 0 et 20. S'il est à l'extérieur, il affiche un message d'erreur. S'il est supérieur à 15, il affiche "Bravo !". Écrire l'organigramme correspondant, proposer un jeu de tests et dérouler un exemple.

Exo 13 : On considère un triangle dont on fournit les longueurs des 3 côtés. Proposer un organigramme, un code `python` qui permet de déterminer si le triangle est isocèle, équilatéral ou autre. Proposer un jeu de tests et dérouler un exemple.

Exo 14 : En reprenant l'étude faite pour savoir si un triangle est rectangle, proposer un code `python` qui répond à la question lorsque les longueurs sont données dans l'ordre croissant.

Exo 15 : À partir des deux vecteurs $\overrightarrow{u} = \binom{a}{b}$ et $\overrightarrow{v} = \binom{a'}{b'}$ qui définissent un parallélogramme, proposer un code `python` qui demande les coordonnées des vecteurs puis fournit l'aire du parallélogramme ainsi définit (on rappelle que le déterminant des deux vecteurs fournit l'aire algébrique du parallélogramme).

Exo 16 : Écrire un programme qui calcule le déterminant ($\Delta = b^2 - 4ac$) d'une équation polynomiale du 2n degré $ax^2+bx+c=0$ et précise le nombre de solutions réelles. Proposer un jeu de tests et un déroulé.

Exo 17 : On propose un début de code qui demande un mot de passe. On souhaite

1. si le mot de passe entré est 'secret', le mot secret est affiché ;
2. si le mot de passe est une partie du mot "secret", un message annonce qu'il y a eut une erreur de frappe (*e.g.* si "secr" est entré on annonce une erreur de frappe) ;
3. sinon on affiche un message courtois de refus.

Compléter ce programme.

```
mot2passe=input("Entrer votre mot
↪  de passe")
if (mot2passe == "secret"):
   print("secret")
elif ( mot2passe ..... ) :
   print(".....")
else :
   print(".....")
```

Exo 18 : On donne l'algorithme ci-contre :

```
Variables
    km : entier
    prix : réel
Algorithme
    SAISIR km
    SI km < 250 ALORS
        prix ← 99
    SINON
        prix ← 99 + (km − 250) × 0,34
    FIN-SI
    AFFICHER prix
Fin Algorithme
```

1. En déroulant l'algorithme, quelle valeur est affichée en sortie si on entre : 112, 250 puis 321 ?

2. Déterminer l'organigramme correspondant

3. Une agence de location, Autoloc', possède un logiciel permettant de calculer le prix (en euro) de location d'une voiture à partir du nombre de kilomètres parcourus. Le logiciel utilise l'algorithme décrit ci-contre.

En déduire les informations manquantes dans l'encadré suivant :

AUTOLOC '
Tarif de location : .....€
Ce tarif permet de parcourir ..... kms
puis ..... € par km supplémentaire.

Certaines de ces voitures sont équipées de la climatisation, l'agence demande alors un supplément de 0,10€ pour chaque kilomètre parcouru, la climatisation augmentant sensiblement la consommation en carburant.

3. Modifier l'organigramme précédent afin que :

- On demande si le client souhaite la climatisation.
- Le montant total de la location soit calculé et affiché.

4. Proposer un jeu de tests, implémenter l'algorithme en `python` puis le tester sur votre jeu de tests.

Exo 19 : On considère trois nombres réels x, y et z.

1. proposer un algorithme qui permettent d'ordonner ces 3 nombres (minimiser le nombre de comparaisons),
2. proposer un jeu de tests puis écrire cet algorithme en langage python.

boucle finie

Exo 20 : écrire un code `python` qui affiche la table de multiplication de l'entier fournit par l'utilisateur.

Exo 21 : écrire un algorithme, un organigramme et un code `python` qui affiche les nombres de 0 à 10. Écrire un code `python` qui affiche les nombres pairs compris entre 10 et 100 compris. Écrire un code `python` qui affiche les nombres impairs de 99 à 1 compris.

Exo 22 : écrire un organigramme et un code `python` qui affiche les carrés des nombres compris entre 0 et 10.

Exo 23 : Écrire un code `python` qui affiche les carrées des nombres compris entre 0 et un nombre n fournit par l'utilisateur. Proposer un jeu de test et dérouler le code pour $n = 3$.

Exo 24 : on considère la fonction f définie par $f(x) = x^2 + 2$. Proposer un code `python` qui affiche les images par f des entiers compris entre -2 et 2. Dérouler le code.

Exo 25 : écrire un code `python` qui demande la valeur d'un réel x et un entier n puis calcule par multiplication successive x^n. Dérouler le programme pour $(x,n) = (3,4)$ puis $(-2,7)$.

Exo 26 : écrire un code `python` qui calcule la somme des termes compris entre deux entiers a et b fournit par l'utilisateur. N'existerait-il pas une formule mathématique pour cela (calculer 2 fois cette somme et regrouper les termes de manière astucieuse) ?

Exo 27 : factorielle : proposer un code `python` qui affiche selon l'entier n la valeur $1 \times 2 \times 3 \times \ldots n$. Si n est nul, 1 sera affiché.

Exo 28 : Soit (u_n) la suite de Fibonacci définie par $u_0 = 0$, $u_1 = 1$ et $u_{n+1} = u_n + u_{n-1}$ pour $n \geqslant 1$.

1. calculer u_2, u_3, u_4 et u_5,
2. proposer un algorithme qui demande à l'utilisateur un entier n et renvoie u_n,
3. écrire (ou implémenter) cet algorithme en `python`.
4. proposer un jeu de tests

Exo 29 : Une personne a mis 1 000€ dans un compte rémunéré à 2%. Combien y a-t-il au bout d'un an ? de deux ? de trois ? Proposer un code `python` qui calcule la somme d'argent disponible au bout de n années.

Exo 30 : Proposer un code `python` qui pour les 100 premiers entiers (0, 1, ... 99), affiche pair s'il est pair, triple s'il est multiple de trois.

Rq : a%b renvoie le reste de la division euclidienne de a par b, ainsi 15%7 renvoie 1.

Exo 31 : Proposer un code `python` qui liste tous les diviseurs entiers de n.

Exo 32 : Vladimir cherche à implémenter une bataille navale contre l'ordinateur.

```
import numpy.random
reussi=False
i=numpy.random.randint(0,3)
j=numpy.random.randint(0,3)
si=int(input("si"))
sj=int(input("sj"))
if si==i and sj==j:
    reussi=True
if reussi :
    print("Bravo, croiseur coule !")
else :
    print("Le croiseur a survecu")
```

Il considère que la mer est un carré de 3×3 et qu'un croiseur occupe une case au hasard.

Rq : la fonction `randint(n,m)` permet de tirer un nombre pseudo-aléatoire entier entre n et m-1.

1. proposer l'organigramme correspondant à ce programme,
2. l'implémenter sur python et le tester,
3. Vladimir aimerait avoir 5 tentatives pour trouver le croiseur ennemi. Il devra effectuer ses 5 tentatives puis seulement après il saura si le croiseur est coulé. Proposer une solution à l'aide d'une boucle POUR, tracer l'organigramme correspondant. Implémenter votre solution sous python.

boucle conditionnelle

Exo 33 :

1. Donner les organigrammes des programmes suivants et les traduire à l'aide de boucles `TANT-QUE`.

```
for i in range(13):
    print(i)
```

```
for i in range(-2,13):
    print(i)
```

```
for i in range(3,12,4):
    print(i)
```

2. Pour le 3ème algorithme, dérouler l'algorithme lorsqu'il s'agit d'une boucle `POUR` puis lorsqu'il s'agit d'une boucle `TANT-QUE`.

Exo 34 : Proposer un programme `python` qui demande à l'utilisateur le nombre n de notes qu'il a, puis toutes ses notes et lui affiche sa moyenne. Proposer un jeu de tests et dérouler pour $(n, n1, n2, n3) = (3, 10, 15, 11)$.

Exo 35 : Proposer un programme `python` qui génère deux chiffres aléatoires et demande à l'utilisateur la valeur du produit tant que la réponse est fausse. Comment renvoyer le nombre d'essais avant réussite ? Changer le code `python` afin que soit affiché ce nombre d'essai.

Exo 36 :

1. proposer un organigramme et un code `python` qui demande à l'utilisateur un nombre *epais* et un nombre *lim*. Le programme doit doubler le nombre *epais* jusqu'à ce qu'il dépasse le nombre *lim*. Il affiche alors la valeur *epais*. Proposer un jeu de tests et dérouler le programme pour $(epais, lim) = (1, 50)$
2. modifier votre organigramme et votre code afin qu'il compte le nombre de pliage effectué.
3. Application : une feuille de papier possède une épaisseur de 0,1mm et la distance terre-lune est d'environ 380 000km. Combier de fois selon vous, faut-il plier la feuille de papier pour dépasser la distance terre-lune ? Faites tourner votre programme et comparer.

Exo 37 : On souhaite jouer au jeu du "juste prix". Le maître du jeu (ou l'ordinateur) choisi un entier entre 0 et 100 et le joueur doit trouver cet entier avec le moins de coup possible. Selon sa réponse, le maître du jeu dit si le nombre recherché est plus petit, plus grand ou égal à sa proposition.

1. proposer un organigramme qui réponde au problème,
2. proposer une implémentation `python`.

Exo 38 : Un établissement scolaire souhaite vérifier si la quantité de CO_2 de l'air ne dépasse pas la valeur limite de 5 000 ppm (partie par million). Proposer un programme `python` qui lit des valeurs ; si ces valeurs restent en deçà de la limite rien ne se passe, sinon le programme s'arrête et lance un message d'alerte.

Exo 39 : On considère la suite de Syracuse (u_n) définie par $u_{n+1} = \begin{cases} 3u_n + 1 & \text{si } u_n \text{ est impair} \\ \frac{u_n}{2} & \text{si } u_n \text{ est pair} \end{cases}$.

1. proposer un organigramme qui demande à l'utilisateur la première valeur u_0 puis calcule et affiche chaque terme u_n jusqu'à obtenir 1.
2. implémenter cet algorithme en `python`.

rq : il n'a toujours pas été démontré que cette suite atteint, quelque soit la valeur initiale, le cycle 4, 2, 1.

fonction

Pour les fonctions, les spécifications seront TOUJOURS précisées.

Exo 40 : Proposer une fonction `zerof` qui étant donné 3 réels a, b et c définissant la fonction polynomiale du 2n degré $x \mapsto ax^2 + bx + c$ renvoie un booléen vrai s'il existe des racines, faux si elles n'existent pas. Proposer un jeu de tests.

Exo 41 : Proposer une fonction `fmax` qui prend deux flottants a et b en argument, renvoie le réel le plus grand. Proposer un jeu de tests.

Exo 42 : Faire de même pour `fmin`.

Exo 43 : Proposer une fonction `fabs` qui prend 1 réel a en argument et renvoie sa valeur absolue. Proposer un jeu de tests.

Exo 44 : Proposer une fonction `pythagore` qui prend trois longueurs (rangées dans l'ordre croissant) en argument et renvoie un booléen selon que le triangle soit rectangle ou non.

Exo 45 : Proposer une fonction `fmax3` qui prend en argument trois flottants et renvoie le plus grand. Proposer un jeu de tests.

Exo 46 : Proposer une fonction `isBissextile` qui prend en argument un entier annee et renvoie un booléen selon que l'année est bissextile ou non. Proposer un jeu de tests.

Exo 47 : Proposer une fonction `puissance` qui prend en argument un flottant x et un entier naturel n et renvoie x^n. On utilisera une boucle `for/while` à l'intérieur de la fonction. Proposer un jeu de tests, dérouler pour $(x,n) = (2,3)$.

Exo 48 : Proposer une procédure `convertime` qui prend en argument un entier nbs représentant le nombre de secondes et affiche la conversion en jours, heure, minutes et secondes. Proposer un jeu de tests.

Exo 49 : Prédire l'affichage des codes `python` suivants :

```
def f(a):
    print(a+y)
y=2
a=5
f(4)
```

```
def f(a):
    y=6
    print(a+y)
y=2
f(4)
```

```
def f(a):
    a=7
    y=2
    print(a+y)
f(4)
```

tableau

Exo 50 : On considère la variable `mot` de type `str`, écrire un code `python` qui affiche les lettres de `mot` de 2 en 2 (la 1è lettre, puis la 3è, etc.) De même, de la dernière lettre à la première.

Exo 51 : Soit un tableau `T`, écrire un code qui affiche les éléments du tableau à l'envers (en partant du dernier). Dérouler pour $T = [1, 3, -2, 8]$.

Exo 52 : Soit `T` un tableau de nombres, proposer un code qui calcule la somme des éléments de `T`. Dérouler pour $T = [1, 3, -2, 8]$.

Exo 53 : Soit `tab` un tableau de nombres, proposer un code qui calcule la somme des éléments de `tab` en multipliant chacun par sa position. *e.g.* si $tab = [3, 2, 5]$, on doit obtenir $3 \times 0 + 2 \times 1 + 5 \times 2 = 12$.

Exo 54 : Soit `tab` un tableau de nombres, proposer un code qui calcule la somme des éléments de `tab` en mettant chaque nombre à la puissance de sa position. *e.g.* si $tab = [3, 2, 5]$, on doit obtenir $3^0 + 2^1 + 5^2 = 28$.

Exo 55 : même exercice mais qui met la valeur absolue de la valeur.

Exo 56 : Soit `tab` un tableau de n nombres, proposer un code qui créer un nombveau tableau de taille $n - 1$ contenant la différence de chaque case contigües. *e.g.* si $tab = [3, 2, 5]$, on doit obtenir $[1, -3]$.

Exo 57 : De deux manières différentes (l'une avec for, l'autre avec while), écrire une fonction `epeler(mot)` qui affiche chacune des lettres d'un mot (ne pas oublier les spécifications). Faire un déroulé pour le mot="Chat".

Exo 58 : Écrire une fonction `positifsT(tab)` qui renvoie True si tous les nombres du tableau tab sont positifs, False sinon.

Exo 59 : Écrire une fonction `nb_valeur_sup(tab, a)` qui renvoie le nombre d'éléments de tab supérieur à a.

Exo 60 : même exercice avec inférieur.

Exo 61 : Écrire une fonction `nbChiffres(n)` qui donne le nombre de chiffres contenus dans l'entier n.

Exo 62 : Écrire une fonction `element_plus_grand(tab, val)` qui renvoie un tableau contenant tous les éléments de *tab* plus grand (ou égal) à *val*. Proposer un jeu de tests.

Exo 63 : Écrire une fonction `presque_egal(tab, val)` qui renvoie un tableau contenant tous les éléments de *tab* de valeur comprise entre $val - 1$ et $val + 1$. Proposer un jeu de tests.

Exo 64 : Écrire une fonction `occurrence(let, mot)` qui renvoie le nombre de fois que la lettre let est dans le mot.

Exo 65 : Écrire une fonction `occurrenceT(val, tab)` qui renvoie le nombre de fois que la valeur val se trouve dans le tableau tab. Proposer un jeu de tests.

Exo 66 : Écrire un code `python` qui créer un tableau de taille aléatoire entre 10 et 20, contenant les carrés 0, 1, 4, 9, ...

Exo 67 : Écrire une fonction `maxT(tab)` qui renvoie l'élément le plus grand du tableau tab.

Exo 68 : Même chose avec `minT(tab)`.

Exo 69 : Écrire une fonction `echangeT(tab, i, j)` qui échange le i^{è} élément avec le j^{è} si les indices i et j sont possibles. Proposer un jeu de tests.

Exo 70 : Écrire une fonction `moyenneT(tab)` qui calcule la moyenne des valeurs contenues dans tab. Proposer un jeu de tests.

Exo 71 : Écrire une fonction `inverseT(tab)` qui renvoie un nouveau tableau inverse de tab (*i.e.* le 1er élément de tab est le dernier du nouveau tableau, etc.)

Exo 72 : Écrire une fonction `concatT(t1, t2)` qui crée un nouveau tableau contenant tous les éléments du tableau t1 puis ceux du tableau t2.

Exo 73 : Écrire une fonction `melangeAleaT(t1, t2)` qui crée un nouveau tableau avec les éléments de t1 et t2 choisis aléatoirement.