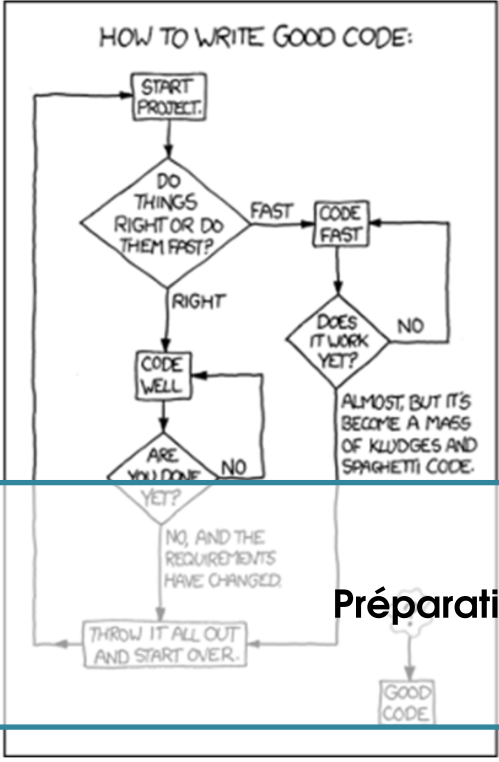
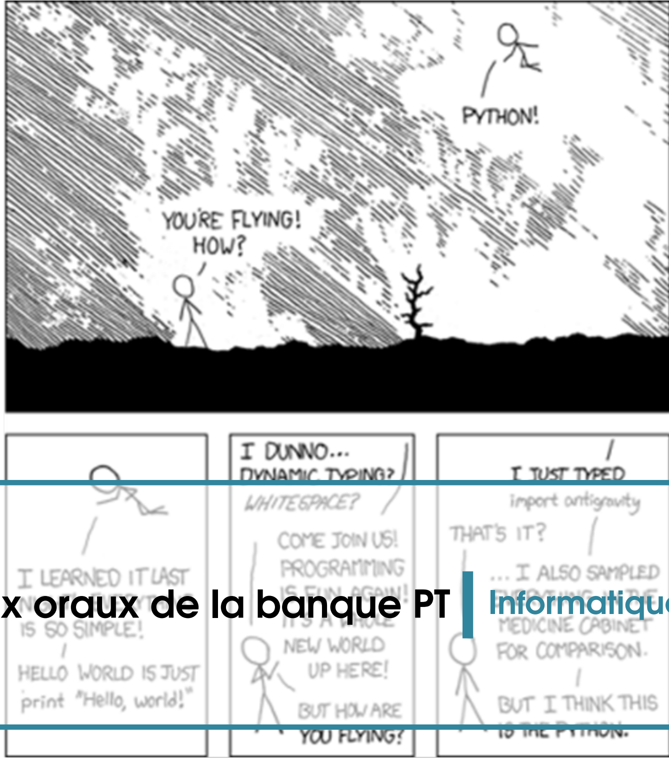



http://xkcd.com/353/
http://xkcd.com/844/

**Oral**

**PT – PT⋆**

Préparation aux oraux de la banque PT | Informatique

# Exercices

# Préparation aux oraux de la banque PT
# Épreuve de « Mathématique et Algorithmique »

## Exercice 1

1. Soit l'entier $n = 1234$. Quel est le quotient, noté $q$, dans la division euclidienne de $n$ par 10 ? Quel est le reste ? Que se passe-t-il si on recommence la division par 10 à partir de $q$ ?
2. Écrire la suite d'instructions calculant la somme des cubes des chiffres de l'entier 1234.
3. Écrire une fonction `somcube`, d'argument `n`, renvoyant la somme des cubes des chiffres du nombre entier `n`.
4. Trouver tous les nombres entiers inférieurs à 1000 égaux à la somme des cubes de leurs chiffres.
5. En modifiant les instructions de la fonction `somcube`, écrire une fonction `somcube2` qui convertit l'entier `n` en une chaîne de caractères permettant ainsi la récupération de ses chiffres sous forme de caractères. Cette nouvelle fonction renvoie toujours la somme des cubes des chiffres de l'entier `n`.

## Exercice 2

On cherche à calculer une valeur approchée de l'intégrale d'une fonction donnée par des points dont les coordonnées sont situées dans un fichier.

1. Le fichier `ex_001.csv`, situé dans le sous-répertoire `data` du répertoire de travail, contient une quinzaine de lignes selon le modèle suivant :

```
0.0;1.00988282142
0.1;1.07221264497
```

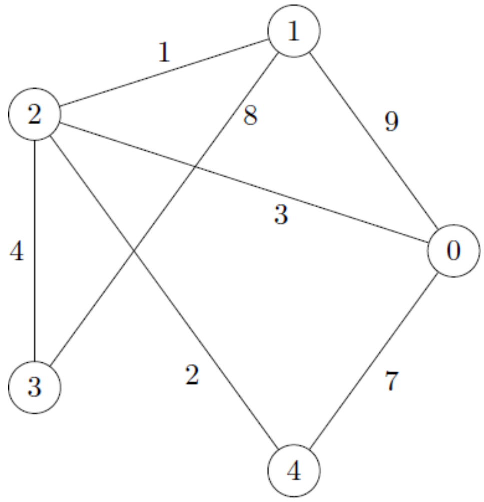
   Chaque ligne contient deux valeurs flottantes séparées par un point-virgule, représentant respectivement l'abscisse et l'ordonnée d'un point. Les points sont ordonnés par abscisses croissantes. Ouvrir le fichier en lecture, le lire et construire la liste `LX` des abscisses et la liste `LY` des ordonnées contenues dans ce fichier.
2. Représenter les points sur une figure.
3. Les points précédents sont situés sur la courbe représentative d'une fonction `f`. On souhaite déterminer une valeur approchée de l'intégrale `I` de cette fonction sur le segment où elle est définie. Écrire une fonction `trapeze`, d'arguments deux listes `y` et `x` de même longueur `n`, renvoyant :

$$\sum_{i=1}^{n-1}(x_i - x_{i-1})\frac{y_i + y_{i-1}}{2}.$$

   `trapeze(LY,LX)` renvoie donc une valeur approchée de l'intégrale $I$ par la méthode des trapèzes.
4. En utilisant la méthode d'intégration numérique `trapz` de la sous-bibliothèque `scipy.integrate` du langage Python ou la méthode `inttrap` du logiciel Scilab, retrouver la valeur approchée de l'intégrale $I$.

## Exercice 3

On considère le graphe `G` suivant, où le nombre situé sur l'arête joignant deux sommets est leur distance, supposée entière :




1. Construire la matrice $\left(M_{ij}\right)_{0\leq i,j\leq 4}$, matrice de distances du graphe `G`, définie par :
   « pour tous les indices $i$, $j$, $M_{ij}$ représente la distance entre les sommets $i$ et $j$, ou encore la longueur de l'arête reliant les sommets $i$ et $j$ ».
   On convient que, lorsque les sommets ne sont pas reliés, cette distance vaut -1. La distance du sommet $i$ à lui-même est, bien sûr, égale à 0.
2. Écrire une suite d'instructions permettant de dresser à partir de la matrice `M` la liste des voisins du sommet 4.
3. Écrire une fonction `voisins`, d'argument un sommet $i$, renvoyant la liste des voisins du sommet $i$.
4. Écrire une fonction `degre`, d'argument un sommet $i$, renvoyant le nombre des voisins du sommet $i$, c'est-à-dire le nombre d'arêtes issues de $i$.
5. Écrire une fonction `longueur`, d'argument une liste `L` de sommets de `G`, renvoyant la longueur du trajet d'écrit par cette liste `L`, c'est-à-dire la somme des longueurs des arêtes empruntées. Si le trajet n'est pas possible, la fonction renverra -1.

## Exercice 4

Soit un entier naturel $n$ non nul et une liste `t` de longueur $n$ dont les termes valent 0 ou 1. Le but de cet exercice est de trouver le nombre maximal de 0 contigus dans `t` (c'est-à-dire figurant dans des cases consécutives). Par exemple, le nombre maximal de zéros contigus de la liste `t1` suivante vaut 4 :

| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|---|
| t1[i] | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 |

| i | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|
| t1[i] | 0 | 1 | 1 | 0 | 0 | 0 | 0 |

1. Écrire une fonction nombreZeros(t,i), prenant en paramètres une liste t, de longueur n, et un indice i compris entre 0 et $n-1$, et renvoyant :

$$\begin{cases} 0, \text{ si } t[i] = 1 \\ \text{le nombre de zéros consécutifs dans t} \\ \text{à partir de t[i] inclus, si t[i]} = 0. \end{cases}$$

Par exemple, les appels `nombreZeros(t1,4)`, `nombreZeros(t1,1)` et `nombreZeros(t1,8)` renvoient respectivement les valeurs 3, 0 et 1.

2. Comment obtenir le nombre maximal de zéros contigus d'une liste `t` connaissant la liste des `nombreZeros(t,i)` pour $0 \leq i \leq n-1$ ? En déduire une fonction `nombreZerosMax(t)`, de paramètre `t`, renvoyant le nombre maximal de 0 contigus d'une liste `t` non vide. On utilisera la fonction `nombreZeros`.
3. Quelle est la complexité de la fonction `nombreZerosMax(t)` construite à la question précédente ?
4. Trouver un moyen simple, toujours en utilisant la fonction `nombreZeros`, d'obtenir un algorithme plus performant.

## Exercice 5

Soient $n$ un entier naturel strictement positif et $p$ un réel compris entre 0 et 1. On considère $X$ et $Y$ deux variables aléatoires à valeurs dans $\mathbb{N}$ sur un espace probabilisé donné. $X$ suit une loi de Poisson de paramètre $\lambda = np$ et $Y$ suit une loi binomiale de paramètres $(n, p)$.

1. Définir une fonction `Px`, d'arguments $k$, $n$ et $p$, renvoyant la valeur de $P(X = k)$. $k!$ (factorielle k) s'obtient par `factorial(k)` en Python (bibliothèque math) et `prod(1 : k)` en Scilab. Déterminer, pour $n = 30$ et $p = 0,1$, la liste des valeurs de $P(X = k)$ pour $k \in \mathbb{N}$, $0 \leq k \leq 30$.
2. Définir une fonction `Py`, d'arguments $k$, $n$ et $p$, renvoyant la valeur de $P(Y = k)$. On pourra utiliser `comb` de la sous-bibliothèque `scipy.misc` en Python et `binomial` en Scilab.
   Déterminer, pour $n = 30$ et $p = 0,1$, la liste des valeurs de $P(Y = k)$ pour $k \in \mathbb{N}$, $0 \leq k \leq 30$.
3. Soit $k \in \mathbb{N}$. On rappelle que, sous certaines conditions sur $n$ et $p$, la probabilité $P(Y = k)$ peut être approchée par $P(X = k)$. Déterminer une fonction `Ecart` d'arguments $n$ et $p$, renvoyant le plus grand des nombres $|P(Y = k) - P(X = k)|$, pour $0 \leq k \leq n$.
4. Soit $e$ un réel strictement positif. Déterminer une fonction `N`, d'arguments $e$ et $p$, renvoyant le plus petit entier $n$ tel que `Ecart(n, p)` soit inférieur ou égal à $e$.
5. Faire l'application numérique dans les quatre cas suivants :
   - $p = 0,075$ avec $e = 0,008$ et $e = 0,005$ ;
   - $p = 0,1$ avec $e = 0,008$ et $e = 0,005$. Interpréter le dernier résultat.

## Exercice 6

On considère la fonction $g$ définie sur $[0,2[$ par :

$$g(x) = \begin{cases} x & \text{pour } 0 \leq x < 1 \\ 1 & \text{pour } 1 \leq x < 2 \end{cases}$$

1. Définir la fonction $g$. Tracer sa courbe représentative sur $[0,2[$, c'est-à-dire la ligne brisée reliant les points $(x, g(x))$ pour $x$ variant de 0 à 1,99 avec un pas de 0,01.
2. Définir une fonction $f$ donnée de manière récursive sur $[0,+\infty[$ par

$$g(x) = \begin{cases} g(x) & \text{pour } 0 \leq x < 2 \\ \sqrt{x} f(x-2) & \text{pour } x \geq 2 \end{cases}$$

3. Tracer la courbe représentative de $f$ sur $[0,6]$.
4. Écrire les instructions permettant de calculer, à $10^{-2}$ près, la plus petite valeur $\alpha > 0$ telle que $f(\alpha) > 4$.

## Exercice 7

On considère le code Python de la fonction `d` suivante :

**Python**
```python
def d(n):
    L =[1]
    for nombre in range(2,n +1):
    if n%nombre == 0:
        L. append(nombre)
    return L
```

1. Quel est le résultat de l'appel `d(4)` ? Puis de l'appel `d(10)` ? Que fait la fonction `d` ?
2. Un diviseur non-trivial d'un entier `n` est un diviseur de `n` différent de 1 et de `n`. Écrire une fonction `DNT`, d'argument `n`, renvoyant la liste des diviseurs non-triviaux de l'entier `n`.
3. Écrire une fonction `sommeCarresDNT`, d'argument `n`, renvoyant la somme des carrés des diviseurs non-triviaux de l'entier `n`.
4. Écrire la suite des instructions permettant d'afficher tous les nombres entiers inférieurs à 1000 et égaux à la somme des carrés de leurs diviseurs non-triviaux. Que peut-on conjecturer ?

## Exercice 8

Soit n un entier vérifiant $n \leq 26$. On souhaite écrire un programme qui code un mot en décalant chaque lettre de l'alphabet de $n$ lettres. Par exemple pour $n = 3$, le décalage sera le suivant :

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| Avant décalage | a | b | c | ... | x | y | z |
| Après décalage | d | e | f | ... | a | b | c |

Le mot `oralensam` devient ainsi `rudohqvdp`.

1. Définir une chaîne de caractères contenant toutes les lettres dans l'ordre alphabétique (caractères en minuscule).
2. Écrire une fonction `decalage`, d'argument un entier `n`, renvoyant une chaîne de caractères contenant toutes les lettres dans l'ordre alphabétique, décalées de `n`, comme indiqué ci-dessus.
3. Écrire une fonction `indices`, d'arguments un caractère `x` et une chaîne de caractères `phrase`, renvoyant une liste contenant les indices de `x` dans `phrase` si `x` est une lettre de phrase et une liste vide sinon.
4. Écrire une fonction codage d'arguments un entier `n` et une chaîne de caractères `phrase`, renvoyant `phrase` codé avec un décalage de `n` lettres.
5. Comment peut-on décoder un mot codé ?

Oral

## Exercice 9

On pose $M = 20$ et $m = 10$. À un nombre $c$ quelconque, on associe la suite $(u_n)_n \geq 0$ définie par $u_0 = 0$ et $u_{n+1} = u_n^2 + c$ pour $n \geq 0$.

S'il existe, on note $k$ le plus petit entier tel que l'on ait $0 \leq k \leq m$ et $|u_k| > M$. On définit alors la fonction $f$ par

$$f : c \mapsto \begin{cases} k \text{ s'il existe} \\ m+1 \text{ sinon.} \end{cases}$$

1. Donner le code définissant la fonction $f$.
2. Tracer l'allure de la courbe représentative de la fonction $f$ sur $[-2;2]$, en créant une liste `LX` de 401 valeurs équiréparties entre -2 et 2 inclus et en utilisant les fonctions `plot` et show de la sous- bibliothèque `matplotlib.pyplot`.
3. Construire le tableau des valeurs $f(x+iy)$ où $x$ prend 101 valeurs comprises entre -2 et 0,5 et $y$ prend 101 valeurs entre -1,1 et 1,1. *On rappelle que le nombre complexe $i$ est représenté par* `1j`. *Par exemple, le complexe $1+2i$ est représenté par* `1+2j`.
4. Tracer l'image que code ce tableau. On pourra utiliser les fonctions `imshow` et `show` de la sous- bibliothèque `matplotlib.pyplot`. Quels paramètres peut-on modifier pour obtenir une meilleure résolution ?