1 Affectation

Enoncé : Un libraire propose une réduction de 3.5% sur le prix hors taxes (HT) d'un livre à 12.35 € HT. Sachant que la taxe sur la valeur ajoutée (TVA) sur les livres est de 5.5%, proposer une instruction de type « affectation » qui permettra de calculer le prix final toutes taxes comprises (TTC) pour le client.

Méthode : On commence par s'abstraire des données spécifiques, en particulier numériques, pour ne considérer que les variables associées.

Il s'agit ici de calculer le prix TTC (noté `ttc`) d'un produit connaissant son prix HT (noté `ht`), la TVA (notée `tva` et exprimée sous la forme d'un pourcentage) sur ce type de produit et la réduction éventuelle (notée `r` et exprimée sous la forme d'un pourcentage du prix HT) proposée par le vendeur sur ce produit.

Sachant qu'une réduction se soustrait du prix HT initial et qu'une taxe s'ajoute au prix hors taxes, on a ainsi : `ttc = ht*(1-r/100)*(1+tva/100)` .

Résultat : Appliquer la méthode précédente revient simplement ici à initialiser les variables `ht`, `r` et `tva` aux données du problème particulier.

Compte-tenu des valeurs de l'énoncé, le code PYTHON ci-contre permet de calculer le prix final TTC recherché.

Listing 1 – **prix d'un livre**

```
1 ht, r, tva = 12.35, 3.5, 5.5
2 ttc = ht*(1-r/100)*(1+tva/100)
```

Vérification : On peut utiliser une méthode d'encadrement du prix TTC pour vérifier que la valeur obtenue est du bon ordre de grandeur.

Etant donné que `r < tva`, le prix TTC sera supérieur au prix HT et inférieur au prix TTC sans réduction : `ht < ttc < ht*(1+tva/100)`.

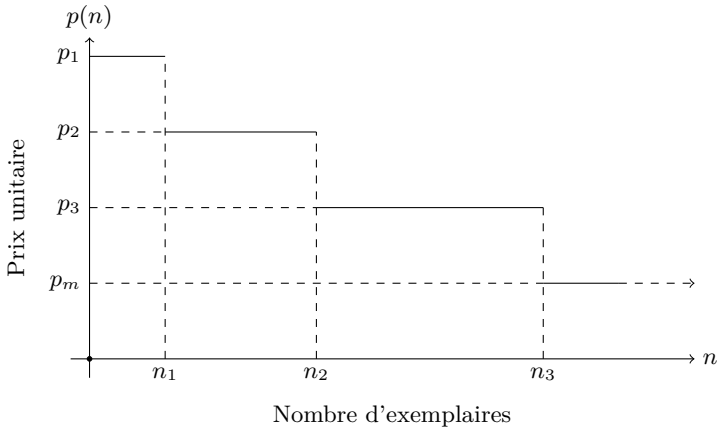
2 Tests

Enoncé : Une grande surface propose un service de photocopies aux conditions suivantes : les 100 premières pages sont facturées 0.1 € la page, les 500 suivantes sont facturées 0.08 € la page et au-delà, la page est facturée à 0.05 €. Proposer une instruction de type « alternative multiple » qui permettra de calculer le prix total des photocopies pour un document de n pages.

Méthode : On commence par s'abstraire des données spécifiques, en particulier numériques, pour ne considérer que les variables associées.

Il s'agit ici de calculer le prix total `pt` de `nt` exemplaires d'un produit sachant que le prix unitaire `p` du produit dépend du nombre d'exemplaires selon le principe suivant.

Les `n1` premiers exemplaires sont facturés au prix unitaire `p1`, de `n1` à `n2` exemplaires le prix unitaire est `p2`, de `n2` à `n3` exemplaires le prix unitaire est `p3`, et ainsi de suite ; au-delà de n_m exemplaires le prix unitaire est `pm`.



Ainsi, si un client veut acheter `nt` exemplaires, l'alternative multiple suivante permet de déterminer le prix total `pt` :

```
if   nt < n1 : pt = nt*p1
elif nt < n2 : pt = n1*p1 + (nt-n1)*p2
elif nt < n3 : pt = n1*p1 + (n2-n1)*p2 + (nt-n2)*p3
...
else         : pt = n1*p1 + (n2-n1)*p2 + (n3-n2)*p3 + ... + (nt-nm)*pm
```

Résultat : Appliquer la méthode précédente revient simplement ici à identifier le nombre de plages de prix concernées, ici 3 : `n1 = 100`, `n2 = 100+500 = 600` avec les prix unitaires `p1 = 0.1`, `p2 = 0.08` et `pm = 0.05`.

Le code PYTHON ci-contre permet de calculer le prix total recherché pour une valeur de `nt` donnée.

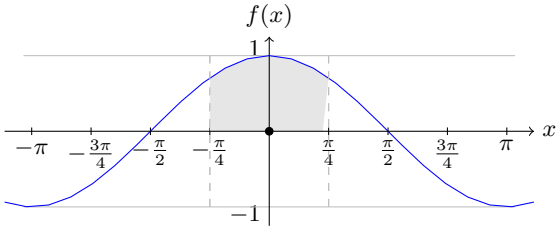
Listing 2 – **prix de photocopies**

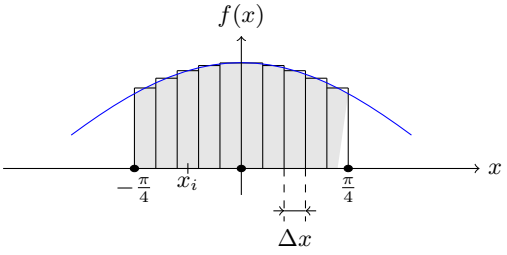
```
1 n1, n2 = 100, 600
2 p1, p2, pm = 0.1, 0.08, 0.05
3 if   nt < n1 : pt = nt*p1
4 elif nt < n2 : pt = n1*p1 + (nt-n1)*p2
5 else         : pt = n1*p1 + (n2-n1)*p2 + (nt-n2)*pm
```

Vérification : On peut ici tester certaines valeurs calculables « à la main » :
`nt = 1000 = 100+500+400` ⇒ `pt = 100*0.1 + 500*0.08 + 400*0.05 = 10+40+20 = 70`,
`nt = 500 = 100+400` ⇒ `pt = 100*0.1 + 400*0.08 = 10+32 = 42`.

3 Boucles

Enoncé : Proposer une instruction de type « boucle » qui permettra de calculer l'intégrale $I = \int cos(x)dx$ sur l'intervalle $[-\frac{\pi}{4}, \frac{\pi}{4}]$ par la méthode des rectangles.





Méthode : Il s'agit ici d'intégrer une fonction continue f de $\mathbb{R} \to \mathbb{R}$ sur un intervalle $[a, b]$. On supposera que f à toutes les bonnes propriétés mathématiques pour être intégrable sur l'intervalle considéré.

On cherche donc à calculer l'intégrale $I = \int_a^b f(x)dx$ qui représente classiquement l'aire comprise entre la courbe représentative de f et les droites d'équations $x = a$, $x = b$ et $y = 0$. Les méthodes d'intégration numérique consistent essentiellement à trouver une bonne approximation de cette aire.

Dans la méthode des rectangles proposée, on subdivise l'intervalle d'intégration de longueur $b - a$ en n parties égales de longueur $\Delta x = \dfrac{b-a}{n}$. Soient x_1, x_2, ..., x_n les points milieux de ces n intervalles. Les n rectangles formés avec les ordonnées correspondantes ont pour surface $f(x_1)\Delta x$, $f(x_2)\Delta x$, ..., $f(x_n)\Delta x$. L'aire sous la courbe est alors assimilée à la somme des aires de ces rectangles, soit

$$I = \int_a^b f(x)dx \approx (f(x_1) + f(x_2) + \cdots + f(x_n))\,\Delta x = \Delta x \cdot \sum_{i=1}^{i=n} f(x_i)$$

C'est la formule dite des rectangles qui repose sur une approximation par une fonction *en escalier*.

Le calcul attendu revient donc à évaluer d'abord la somme $s = \sum f(x_i)$ puis à la multiplier par la largeur $w = \Delta x$ des petits rectangles. Pour calculer s « de tête », on évalue d'abord $f(x_1)$ que l'on mémorise : $s = f(x_1)$, puis on rajoute $f(x_2)$ et on mémorise cette somme intermédiaire : $s = s + f(x_2)$ $(= f(x_1) + f(x_2))$, puis on rajoute $f(x_3)$ et on mémorise cette nouvelle somme intermédiaire $s = s + f(x_3)$ $(= f(x_1) + f(x_2) + f(x_3))$ et ainsi de suite jusqu'à rajouter enfin $f(x_n)$: $s = s + f(x_n)$ $(= f(x_1) + f(x_2) + f(x_3) + \ldots + f(x_n))$. On a alors la somme recherchée $(s = \sum f(x_i))$ et il ne reste plus qu'à multiplier cette somme s par la largeur w pour obtenir la valeur de l'intégrale I.

Les 4 étapes de construction de la boucle recherchée sont donc les suivantes :

- pour l'invariant, on vérifie qu'à chaque étape k, s_k est toujours égal à la somme des k premiers termes $f(x_k)$: $\forall k, s_k = \sum_{i=1}^{k} f(x_i)$;
- l'initialisation consiste à définir w, à se positionner sur le premier rectangle $x_1 = a + w/2$ et à initialiser s_1 en conséquence : $s_1 = f(x_1)$;
- la progression consiste à se déplacer sur le rectangle suivant : $x_{k+1} = x_k + w$ et à augmenter s_k de la nouvelle valeur $f(x_{k+1})$ afin de conserver l'invariant : $s_{k+1} = s_k + f(x_{k+1})$;
- la condition d'arrêt est obtenue pour $x_k > b - w/2$.

La boucle prend ainsi la forme :

```
« initialisation »
while not « condition d'arrêt » :
    « progression »
```

$w = (b-a)/2$, $x = x_1 = a + w/2$, $s = f(x_1)$
while not $x > b - w/2$:
    $x = x_{k+1} = x_k + w$
    $s = s_{k+1} = s_k + f(x_{k+1})$

Résultat : On applique la méthode précédente à la fonction $f : x \mapsto \cos(x)$ dans l'intervalle $\left[a = -\dfrac{\pi}{4}, b = \dfrac{\pi}{4}\right]$.

Le code PYTHON ci-contre permet de calculer l'intégrale recherchée à condition de ne pas oublier de définir f, a, b et n avant d'exécuter la boucle précédente. On prendra par exemple $n = 100$.

Listing 3 – **intégration numérique**

```
from math import *
f, a, b = cos, -pi/4, pi/4
n = 100
w = (b - a)/n
x = a + w/2
s = f(x)
while not (x > (b - w/2)):
    x = x + w
    s = s + f(x)
s = w*s
```

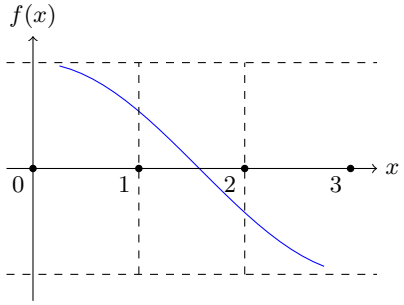
Vérification : On sait calculer analytiquement l'intégrale demandée :

$$I = \int_{-\frac{\pi}{4}}^{\frac{\pi}{4}} \cos(x)dx = \Big[\sin(x)\Big]_{-\frac{\pi}{4}}^{\frac{\pi}{4}} = sin(\frac{\pi}{4}) - sin(-\frac{\pi}{4}) = \frac{\sqrt{2}}{2} - (-\frac{\sqrt{2}}{2}) = 2\frac{\sqrt{2}}{2} = \sqrt{2}$$

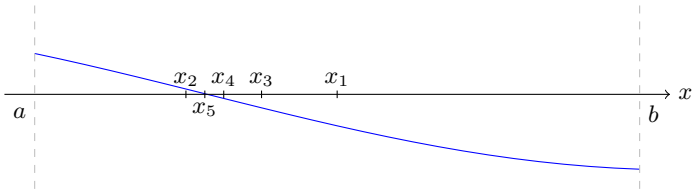
Ainsi, doit-on trouver une valeur proche de $\sqrt{2} \approx 1.414$.

4 Instructions imbriquées

Enoncé : Ecrire un algorithme qui permettra de déterminer le zéro de $\cos(x)$ dans $[1, 2]$ selon une méthode par dichotomie.



Indications : Soient $[x_1, x_2]$ l'intervalle de recherche et $x_m = (x_1 + x_2)/2$ le point milieu de cet intervalle. Si $f(x_1) \cdot f(x_m) < 0$, le zéro recherché est dans $[x_1, x_m]$, sinon le zéro est dans $[x_m, x_2]$. On réitère le procédé sur le nouvel intervalle de recherche jusqu'à ce que la longueur de l'intervalle soit suffisamment petite. Le milieu de ce dernier intervalle sera le zéro recherché.



Méthode :

Résultat :

Le code PYTHON ci-contre permet de calculer le zéro de la fonction cosinus dans l'intervalle $[1, 2]$. Pour le seuil de précision, on prendra par exemple $s = 10^{-9}$.

Listing 4 – **zéro d'une fonction**

```
from math import *
f, a, b = cos, 1, 2
s = 1.e-9
x1, x2 = a, b
x = (x1 + x2)/2
while not ((x2 - x1) <= s):
    if f(x1)*f(x) < 0 : x2 = x
    else              : x1 = x
    x = (x1 + x2)/2
```

Vérification : On connaît les zéros de la fonction cosinus : ce sont tous les multiples de $\pi/2$; ainsi, dans l'intervalle $[1, 2]$ doit-on trouver une valeur proche de $\pi/2 \approx 1.57$.

5 Exécution d'une séquence d'instructions

Enoncé : On considère la séquence d'instructions suivantes :

```
n = 0
while n <= k:
    j = 0
    while j <= n:
        num, den = 1, 1
        i = 1
        while i <= j:
            num = num * (n - i + 1)
            den = den * i
            i = i + 1
        c = num//den
        print(c,end=' ')
        j = j + 1
    print()
    n = n + 1
```

Qu'affiche cette séquence pour `k = 6` ?

Réponse : On n'attend pas ici une réponse concernant la méthode (M). En guise de vérification (V), on pourra proposer un nom à cet algorithme.

Méthode : Pour suivre « à la main » l'exécution d'une série de boucles imbriquées, on calcule les valeurs de toutes les variables impliquées à la fin de chaque itération de chacune des boucles.

Résultat : L'affichage attendu est le suivant :

```
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
```

Vérification : Il s'agit du triangle de Pascal à l'ordre $k = 6$ qui donne les coefficients du binôme $(x + y)^k$.

$$(x+y)^k = \sum_{n=0}^{k} \left(\begin{array}{c} k \\ n \end{array} \right) x^{k-n} y^n = \sum_{n=0}^{k} \frac{k!}{n!(k-n)!} x^{k-n} y^n$$

La version proposée calcule directement le numérateur `num` et le dénominateur `den` du coefficient `c` en tenant compte de la simplification suivante :

$$c = \left(\begin{array}{c} k \\ n \end{array} \right) = \frac{k!}{n!(k-n)!} = \frac{k \cdot (k-1) \cdots (k-n+2) \cdot (k-n+1)}{1 \cdot 2 \cdot 3 \cdots n}$$

Ainsi, on a les 7 premières identités :

$$\begin{aligned}
(x+y)^0 &= 1 \\
(x+y)^1 &= 1 \cdot x^1 + 1 \cdot y^1 \\
(x+y)^2 &= 1 \cdot x^2 + 2 \cdot x^1 y^1 + 1 \cdot y^2 \\
(x+y)^3 &= 1 \cdot x^3 + 3 \cdot x^2 y^1 + 3 \cdot x^1 y^2 + 1 \cdot y^3 \\
(x+y)^4 &= 1 \cdot x^4 + 4 \cdot x^3 y^1 + 6 \cdot x^2 y^2 + 4 \cdot x^1 y^3 + 1 \cdot y^4 \\
(x+y)^5 &= 1 \cdot x^5 + 5 \cdot x^4 y^1 + 10 \cdot x^3 y^2 + 10 \cdot x^2 y^3 + 5 \cdot x^1 y^4 + \cdot y^5 \\
(x+y)^6 &= 1 \cdot x^6 + 6 \cdot x^5 y^1 + 15 \cdot x^4 y^2 + 20 \cdot x^3 y^3 + 15 \cdot x^2 y^4 + 6 \cdot x^1 y^5 + 1 \cdot y^6
\end{aligned}$$