Python en 2nde

Alain Busser

2011

Table des matières

Chapitre 1

Nombres

I/ Vrit

1°) Affectation

Lorsqu'on souhaite que la variable x soit gale 3, on place le nombre 3 dans x :

En *Python*, l'affectation se note par un signe "=".

```
x=3
print(x)
```

Des affectations peuvent se succder :

```
x=3
x=8*7
print(x)
```

2°) Propositions

Une **proposition** est une affirmation qui est soit vraie, soit fausse. Par exemple :

1: « $2+2=4$ » est vraie ;

2: « $-8>5$ » est fausse ;

3: « $2x+1=3$ » n'est pas une proposition.

```
print(2+2==4)
print(-8>5)
print(2*x+1==3)
```

Pour la distinguer de l'affectation, l'galit est note par le signe "=" ddoubl ; de plus, « suprieur ou gal » se note >=.

3°) Ngation

```
print(not True)
print(not False)
```

Exercice :

Donner les ngations des propositions suivantes :

1: « Le triangle ABC est isocle en A » :.....

2: « Les droites d_1 et d_2 sont parallles » :.....

3: « L'entier naturel n est impair » :.....

4°) Oprations logiques

a) Conjonction

La conjonction de deux propositions n'est vraie que si chacune des deux propositions est vraie.

```
print(2+2==4 and 2>3)
```

Exercice :

Simplifier les noncs des conjonctions suivantes :

1: « Le triangle ABC est isocle et il a un angle de $60°$ » :

.....

2: « Les diagonales de $ABCD$ sont perpendiculaires et elles ont le mme milieu » :.....

3: « Les droites d_1 et d_2 sont parallles et elles passent par le point P » :

.....

b) Disjonction

Ds qu'une des propositions d'une disjonction est vraie, toute la disjonction est vraie :

```
print(2+2==4 or 2>3)
```

II/ Nombres entiers

1°) Oprations en Python

Les oprations sont notes respectivement +, -, * et /. L'lvation une puissance se note avec l'astrisque ddouble **. Par exemple pour afficher le cube de 5, on peut faire

```
print(5**3)
```

Exercice :

Que va afficher le programme *Python* ci-dessous ?

```
print(2+3*7)
print(-3**2)
print((-3)**2)
print(5+1/2+3)
print((5+1)/(2+3))
```

2°) Reconnaissance

Python reconnat les nombres en affichant leur *type*. Ainsi, $\frac{3}{2}$ et « vrai » ne sont pas entiers, alors que 3 est entier :

```
print(type(3/2))
print(type(3))
print(type(True))
```

le type des propositions est une abrviation de *boolen*, d'aprs le nom de George Boole.

3°) Division euclidienne

a) Arrondi

L'arrondi entier par dfaut d'un nombre positif se note *int* en *Python* :

```
print(int(3/2))
print(int(3))
```

b) Quotient

Pour obtenir le quotient euclidien de 34 par 13, il suffit de faire

```
print(34//13)
```

c) Reste

Pour obtenir le reste euclidien de 34 par 13, on utilise le symbole "pourcent" :

```
print(34%13)
```

III/ Nombres rels

1°) criture

Le carr de 2 millimes est 4 millionnimes :

```
print(0.002**2)
```

L'affichage $4e-6$ veut dire 4×10^{-6} : C'est l'**criture scientifique** du rel. La prsence de l'exposant vite de fixer la place de la virgule : en *Python*, les rels sont dits **flottants**.

Bien que $\frac{6}{2}$ soit entier, *Python* ne le sait pas :

```
print(type(6/2))
```

Exercice d'algorithmique :

Comment faire pour que *Python* reconnaisse $\frac{6}{2}$ comme un entier ?

2°) Programmation objet

a) Proprits

Le rel π n'est pas connu par *Python*. Pour le charger, il faut le rcuprer auprs de l'objet *math* dont il est une **proprit** :

```
from math import pi
print(pi)
```

b) Mthodes

L'expression "racine carre" s'abrge *sqrt* (**sq**uare **r**oo**t**). Mais pour accder cette fonction en *Python*, on doit aussi l'importer depuis l'objet *math* dont elle est une **mthode** (ou algorithme) :

```
from math import sqrt
print(sqrt(2))
```

Le meilleur moyen pour faire des mathmatiques avec *Python*, c'est d'importer toutes les proprits et toutes les mthodes de l'objet *math* avec

```
from math import *
```

Dans ce cas, l'astrisque signifie « tout ».

Chapitre 2

Vocabulaire des vnements

I/ vnements

1°) Description

a) Notation

Un vnement est dcrit par la liste de ses issues possibles, note entre accolades.

La mme notation sera utilise pour donner la liste des solutions d'une quation ou inquation.

b) Exemples

1: On tire une carte d'un jeu de 32. L'vnement « la carte est une dame » se note $D = \{D\diamondsuit, D\heartsuit, D\spadesuit, D\clubsuit\}$.

2: Toujours avec un jeu de 32 cartes, l'vnement « la carte est un pique » se note $P = \{1\spadesuit, 7\spadesuit, 8\spadesuit, 9\spadesuit, 10\spadesuit, V\spadesuit, D\spadesuit, R\spadesuit\}$.

3: En lanant un d, l'vnement « le rsultat est pair » se note $A = \{2, 4, 6\}$, et l'vnement « le rsultat est plus petit que 5 » se note $B = \{1, 2, 3, 4\}$.

2°) Cas particuliers

1: L'**vnement certain** ou **univers** contient toutes les ventualits. On le note Ω. Par exemple, en lanant un d, $\Omega = \{1, 2, 3, 4, 5, 6\}$.

2: L'**vnement impossible** est au contraire celui qui ne contient aucune ventualit. Par exemple, en choisissant une carte au hasard dans un jeu de 32, l'vnement « la carte est un 37 de foie » est impossible.

L'vnement impossible est not $\{\ \}$ ou $\emptyset$.

3: Lorsqu'un vnement ne contient qu'une ventualit, on dit qu'il est **lmentaire**. Par exemple :

(a) Avec un jeu de cartes, l'vnement « la carte est l'as de pique » ou {1♠} est lmentaire.
(b) Avec un d, l'vnement « le d est tomb sur 6 » est lmentaire aussi : C'est {6}.

3°) Lancer de d en Python

a) Simulation

Pour lancer un d, on peut faire ceci :

```
from random import *
print(randint(1,6))
```

b) Univers

Pour faciliter la suite, on va stocker l'univers dans une variable appele *omega*.

```
omega=set(range(1,7))
print(omega)
```

c) Autres vnements

```
pair={2,4,6}
petit={1,2,3,4}
print(pair)
print(petit)
```

II/ Oprations

1°) Contraire

Le contraire d'un vnement A, not $\overline{A}$, est form des ventualits qui ne sont pas dans A. On l'obtient en enlevant A Ω :

```
omega=set(range(1,7))
pair={2,4,6}
impair=omega-pair
print(impair)
```

Exercice :
Donner le contraire des vnements suivants :

1: L'univers :.....

2: L'vnement impossible :.....

3: L'vnement « la carte est un pique » :.....

4: L'vnement « la carte est rouge » :.....

5: Quel est le contraire du contraire de A ?.....

2°) Conjonction

L'vnement « A et B » est form de toutes les ventualits communes A et B. On le note $A \cap B$.

Le symbole "∩" ressemble un "n", lettre centrale du mot anglais *and* qui s'abrge souvent par une esperluette &, elle-mme obtenue par une dformation calligraphique de "Et".

```
omega=set(range(1,7))
pair={2,4,6}
petit={1,2,3,4}
print(pair&petit)
```

Lorsque $A \cap B = \emptyset$, on dit que A et B sont **incompatibles**. C'est le cas en particulier de A et $\overline{A}$: On ne peut pas avoir la fois un vnement et son contraire.

Exercice : La rciproque est-elle vraie ?

3°) Disjonction

L'vnement « A ou B » est form de toutes les ventualits qui sont soit dans A, soit dans B. On le note $A \cup B$.

Le symbole "∪" ressemble un "u", qui se prononce souvent "ou" dans plusieurs langues.

```
omega=set(range(1,7))
pair={2,4,6}
petit={1,2,3,4}
print(pair|petit)
```

Chapitre 3

Intervalles

I/ Ensembles de nombres

1: L'ensemble des entiers naturels est not $\mathbb{N}$.

2: L'ensemble des entiers (relatifs) est not $\mathbb{Z}$.

3: L'ensemble des fractions est not $\mathbb{Q}$.

4: L'ensemble des rels est not $\mathbb{R}$.

L'initiale de Zahl, « nombre » en Allemand.

Scoop : $\pi \notin \mathbb{Q}$

Pour noter qu'un nombre x est entier, on crit $x \in \mathbb{Z}$. Pour noter qu'un nombre x n'est pas entier, on crit $x \notin \mathbb{Z}$.

II/ Inclusion

Pour dire que tous les entiers sont des fractions, on note $\mathbb{Z} \subset \mathbb{Q}$. On dit que $\mathbb{Z}$ est **inclus** dans $\mathbb{Q}$.

III/ Intervalles de rels

1°) Segments

L'ensemble de tous les nombres compris entre a et b est not $[a;b]$. Plus prcisment

1: $x \in [a;b]$ veut dire $a \leqslant x \leqslant b$;

2: $x \in [a;b[$ veut dire $a \leqslant x < b$;

3: $x \in]a;b]$ veut dire $a < x \leqslant b$;

4: $x \in]a;b[$ veut dire $a < x < b$;

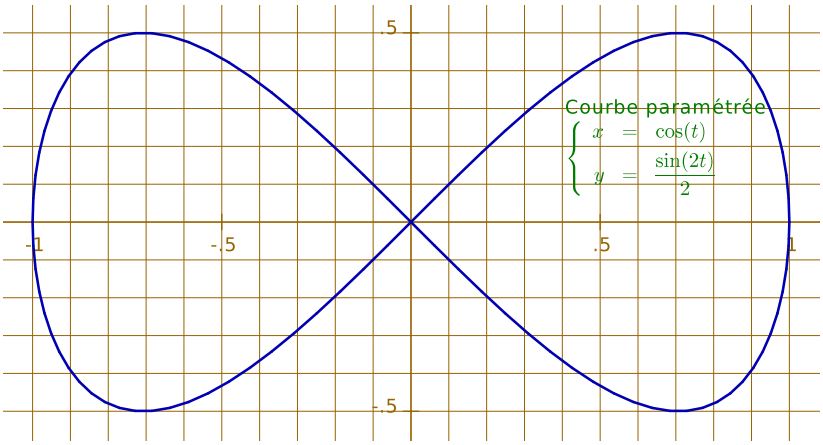
2°) Intervalles d'entiers

En *Python*, les intervalles d'entiers peuvent tre dfinis par *range*, suivi par les deux bornes de l'intervalle. L'appartenance s'crit *in*. Par exemple, les entiers entre 13 et 21 constituent l'« intervalle »

```
intervalle=range(13,21)
print(0 in intervalle)
print(13 in intervalle)
print(13.2 in intervalle)
print(20 in intervalle)
print(23 in intervalle)
print(21 in intervalle)
```

En bref, le "range" en question est mathmatiquement dcrit par $\mathbb{N} \cap [13; 21[$.

3°) Demi-droites



On prononce « plus l'infini ».

L'ensemble des tous les rels suprieurs a est not $]a; +\infty[$; l'ensemble de tous les rels suprieurs ou gaux a est $[a; +\infty[$. Ce sont aussi des intervalles.

L'ensemble des nombres infrieurs b est not $]-\infty; b[$.

L'ensemble des rels est lui-mme un intervalle : $\mathbb{R} =]-\infty; +\infty[$.

L'ensemble vide est aussi un intervalle : $]4; 4[= \emptyset$.

L'intersection de deux intervalles est toujours un intervalle. La runion de deux intervalles n'est pas forcment un intervalle.

IV/ Boucles en Python

Les *range* de *Python* servent avant tout faire quelque chose de rptitif, par exemple pour crire beaucoup de texte :

```
for n in range(5):
        print('ligne num ro '+str(n))
```

Chapitre 4

Gnralits sur les fonctions

I/ Exemple

Si on modifie le rayon x d'une boule, on modifie galement son volume : On dit que le volume V est *fonction* du rayon x.

En Python, on peut crire

```
from math import *
def V(x):
    return 4/3*pi*x**3
```

On rappelle que $V(x) = \frac{4}{3}\pi x^3$.

II/ Dfinitions

1°) Image

On dit que $V(x)$ est l'*image* de x.

Par exemple, l'image de 10 est $V(10) = \frac{4}{3}\pi \times 10^3 \simeq 4\,189$ cm^3 soit environ 4,2 litres.

2°) Antcdent

On dit que x est l'antcdent de $V(x)$ par V.

Pour trouver l'antcdent de 1000 par V, on doit rsoudre l'quation $\frac{4}{3}\pi x^3 = 1000$.

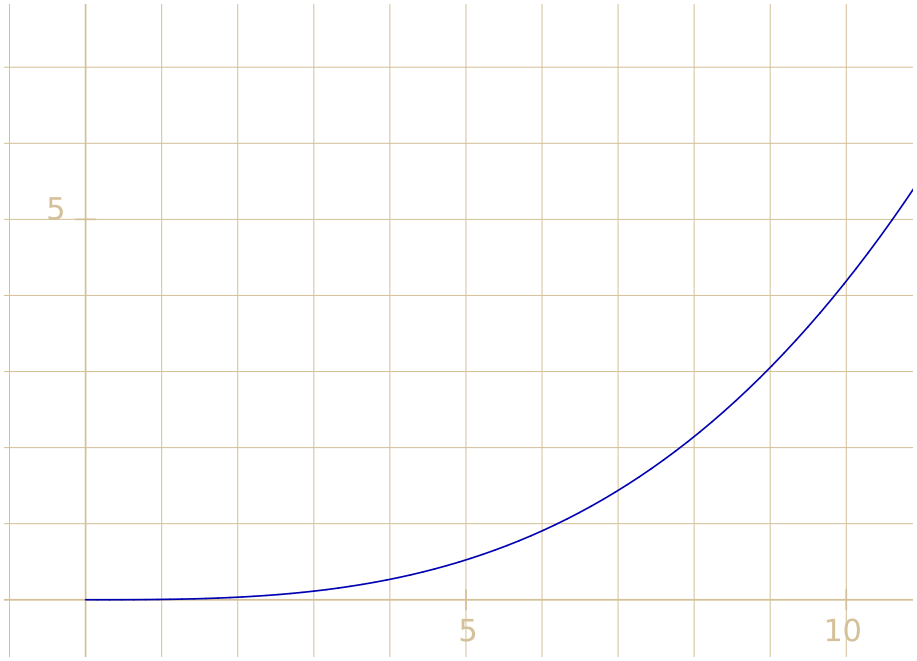
3°) Domaine

L'ensemble des rels qui ont une image par V s'appelle le *domaine de dfinition* de V. Ici c'est $[0; +\infty[$ parce qu'un rayon est toujours positif.

III/ Reprsentation graphique

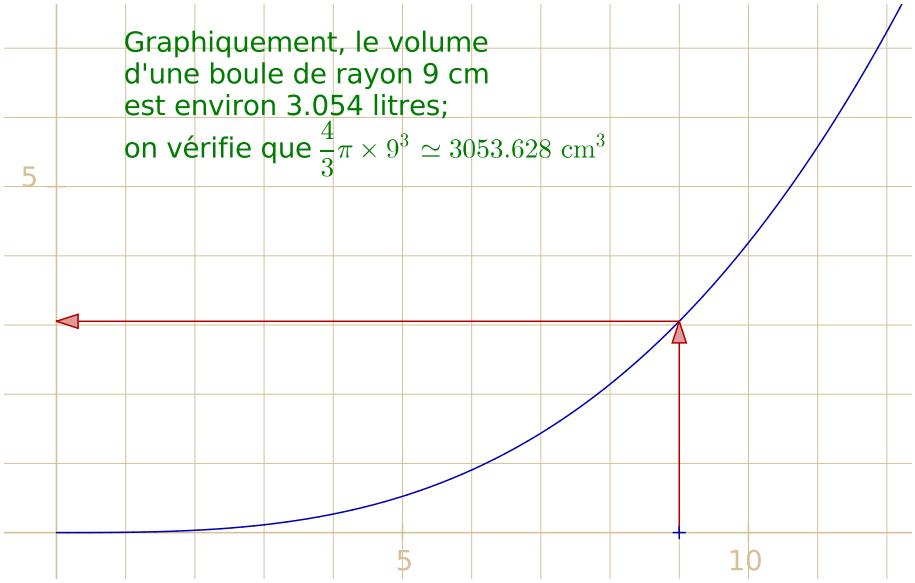
1°) Dfinition

Si, dans un repre, on place tous les points dont l'ordonne y est l'image de l'abscisse x, on obtient une courbe appele *reprsentation graphique* de la fonction dans le repre.

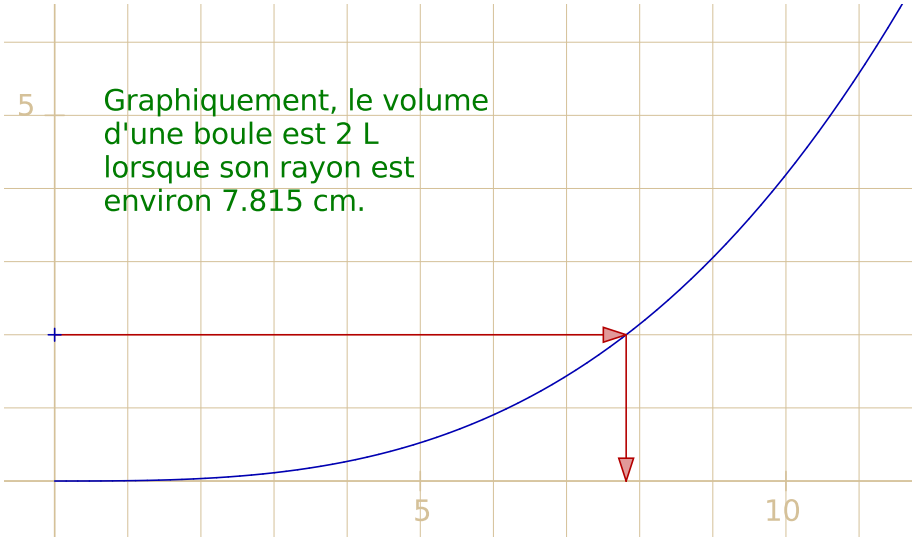


2°) Utilisation

a) Recherche d'images

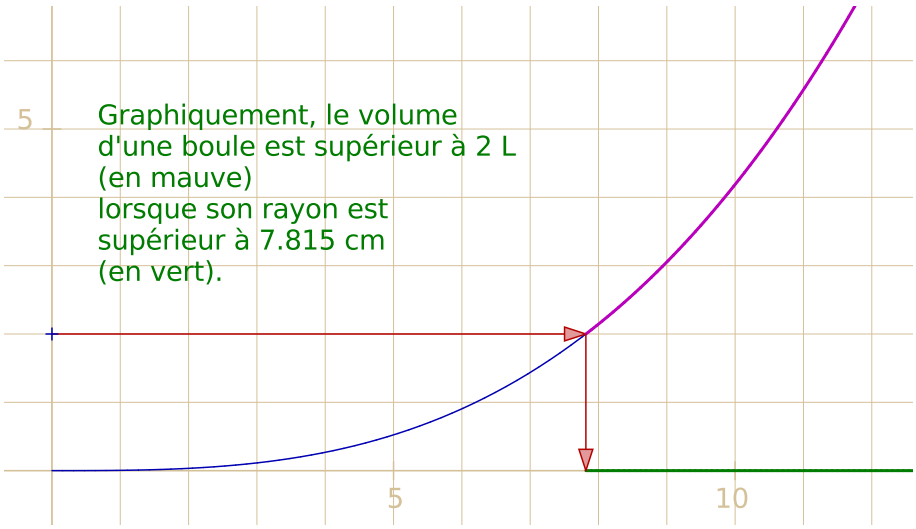


b) Recherche d'antcdents



Cette mthode permet de rsoudre des quations graphiquement.

c) Rsolution d'inquations



On trouve graphiquement que l'ensemble des solutions de $V(x) \geqslant 2$ est $[7,8; +\infty[$.

IV/ Variations

1°) fonction croissante

Plus le rayon d'une boule est grand, plus son volume est important : On dit que la fonction V est *croissante* sur $[0; +\infty[$.

Une fonction est croissante sur un intervalle I si, chaque fois que $a \leqslant b$ dans I, $f(a) \leqslant f(b)$.

2°) fonction dcroissante

Une fonction est dcroissante sur un intervalle I si, chaque fois que $a \leqslant b$ dans I, $f(a) \geqslant f(b)$.

3°) extrema

a) Maximum

On dit que M est le *maximum* de f sur $[a; b]$ si pour tout x de $[a; b]$ on a $f(x) \leqslant M$.

Pour trouver le maximum de la fonction $f(x) = x - \dfrac{x^3}{25}$ sur l'intervalle $[0;5]$ on peut, avec une assez bonne approximation, chercher la plus grande valeur d'un tableau de valeurs de f :

```
def f(x):
    return x-x**3/25

M=max[f(x/1000) for x in range(5000)]
```

b) Minimum

On dit que m est le *minimum* de f sur $[a;b]$ si pour tout x de $[a;b]$ on a $f(x) \leqslant m$.

Chapitre 5

Translations

I/ Implication

1°) Exemple

L'nonc « Si ABC est rectangle en A alors $AB^2 + AC^2 = BC^2$ » peut aussi s'crire

1: ABC est rectangle en A seulement si $AB^2 + AC^2 = BC^2$;

2: Pour que ABC soit rectangle en A, il est ncessaire que $AB^2 + AC^2 = BC^2$;

3: Pour que $AB^2 + AC^2 = BC^2$, il suffit que ABC soit rectangle en A ;

4: Le fait que ABC est rectangle en A implique que $AB^2 + AC^2 = BC^2$;

5: Si $AB^2 + AC^2 \neq BC^2$, alors ABC ne peut pas tre rectangle en A ;

6: Ou bien ABC n'est pas rectangle en A, ou alors $AB^2 + AC^2 = BC^2$.

2°) Dfinition

tant donnes deux propositions a et b, la proposition « a est faux ou b est vrai » s'appelle **implication** de a vers b et se note $a \Rightarrow b$.

Par exemple, le thorme ci-dessus s'abrge en

$$\widehat{BAC} = 90° \Rightarrow AB^2 + AC^2 = BC^2$$

3°) Rciproque

a) Dfinition

En inversant le sens de la flche dans une implication, on obtient la **rciproque** de celle-ci. Par exemple, la rciproque du thorme prcdent peut s'crire

$$\widehat{BAC} = 90° \Leftarrow AB^2 + AC^2 = BC^2$$

ou alors

$$AB^2 + AC^2 = BC^2 \Rightarrow \widehat{BAC} = 90°$$

Exercice :
Donner un exemple de thorme vrai, dont la rciproque est fausse.

b) quivalence logique

La proposition « $a \Rightarrow b$ et $b \Rightarrow a$ » se dit « a est quivalent b » et se note $a \Leftrightarrow b$. On dit souvent « a est vrai si et seulement si b est vrai ».

4°) Applications

a) Dmonstration

Lorsque a et $a \Rightarrow b$ sont tous les deux vrais, alors b est vrai aussi : On dit qu'on a dmontr b partir de a.

b) Contrapose

La **contrapose** de « a implique b » est « le contraire de b implique le contraire de a ».

Toute implication est quivalente sa contrapose.

c) Algorithmes et tests

En *Python*, il y a quelque chose qui ressemble « si a alors b », mais alors que a est toujours une proposition, b est une instruction (ou suite d'instructions) en *Python*, donc un algorithme effectuer, et pas une proposition : Au lieu de « si a est vrai alors b est vrai aussi » on dit plutt « si a est vrai alors il faut faire b ». Par exemple, pour jouer un jeu de d :

```
from random import *
lancer=randint(1,6)
if lancer==6:
        print('gagn ')
```

(on ne gagne que si le d est tomb sur 6). Peut-tre devrait-on aussi voir ce qu'il convient de faire dans les autres cas, pour cela on crit

```
from random import *
lancer=randint(1,6)
if lancer==6:
        print('gagn ')
else:
        print('perdu')
```

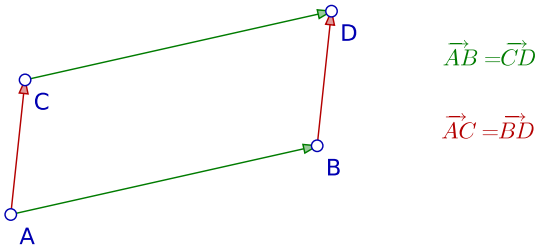
II/ Vecteurs

1°) Dfinitions

a) Notation

Le vecteur allant de A vers B est not $\overrightarrow{AB}$ et reprsent par un segment flch allant de A vers B.

b) galit



On dit que $\overrightarrow{AB} = \overrightarrow{CD}$ lorsque $ABDC$ est un paralllogramme.

c) Translation

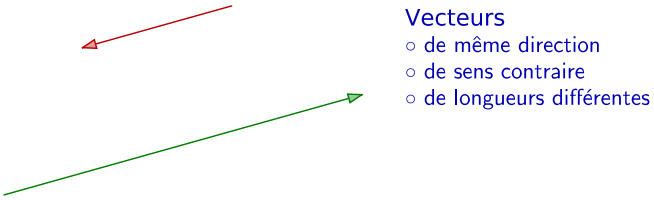
Dans le mme cas, on dit que la mme translation amne A en B et C en D. On dit que c'est la translation de vecteur $\overrightarrow{AB}$.

2°) Proprits

1: Lorsque $(AB) \parallel (CD)$, on dit que les vecteurs $\overrightarrow{AB}$ et $\overrightarrow{CD}$ ont la **mme direction**.

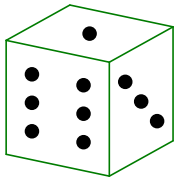
2: Si, en plus, ils ont la flche du mme ct, on dit qu'ils ont le mme sens. Par exemple, $\overrightarrow{AB}$ et $\overrightarrow{BA}$ sont de sens contraires (mais ils ont la mme direction).

3: La **longueur** du vecteur $\overrightarrow{AB}$ est la distance AB.



Chapitre 6

Probabilits



I/ Dfinitions

1°) Probabilit

La **probabilit** d'un vnement A, note $P(A)$, est un nombre compris entre 0 et 1, qui mesure les chances de ralisation de A. Ce nombre ne prsente d'intrt que lorsque A ne s'est pas encore ralis.

Il s'agit d'une fonction, mais elle n'est pas numrique puisque l'antcdent est un vnement, pas un nombre.

2°) quiprobabilit

On dit qu'il y a **quiprobabilit** lorsque tous les vnements lmentaires ont la mme probabilit.

Dans ce cas, la probabilit d'un vnement est le quotient de sa taille (le nombre d'ventualits qu'il contient), par la taille de l'univers :

```
omega=set(range(1,7))

def proba(evenement):
        return len(evenement)/len(omega)

pair={2,4,6}
petit={1,2,3,4}

print(proba(pair))
print(proba(petit))
```

len est le dbut du mot *length* qui veut dire "longueur" en Anglais.

Le second exemple n'est pas dcimal, donc on lui prfre l'criture en fraction. Avec *Python* on peut adapter l'exemple prcdent en

```
from fractions import *
```

```
omega=set(range(1,7))

def proba(evenement):
        return Fraction(len(evenement),len(omega))

petit={1,2,3,4}
print(proba(petit))
```

L'affichage prcdent tait une valeur approche de $\frac{4}{6} = \frac{2}{3}$, ce qu'on exprime parfois par « il y a deux chances sur trois d'avoir un nombre infrieur 5 en lanant un d quilibr ».

II/ Proprits

1°) Intervalle

Une probabilit est toujours comprise entre 0 et 1.

On vitera l'usage des pourcentages pour exprimer une probabilit ; cette notation est rserve aux statistiques.

2°) Cas particuliers

1: $P(\emptyset) = 0$

2: $P(\Omega) = 1$

3°) Disjonction

```
omega=set(range(1,7))

def proba(evenement):
        return len(evenement)/len(omega)

pair={2,4,6}
petit={1,2,3,4}

print(proba(pair|petit)+proba(pair&petit))
print(proba(pair)+proba(petit))
```

On admettra que dans le cas gnral,

$$P(A \cup B) + P(A \cap B) = P(A) + P(B)$$

4°) Thorme

$$P(\overline{A}) = 1 - P(A)$$

III/ Cas non quiprobable

1°) Exemple

Si le d est truqu, le 6 sort plus souvent que les autres rsultats, par exemple, en donnant les probabilits de chaque chiffre dans un tableau :

La donne de ces probabilits s'appelle la loi de probabilit de l'univers.

vnements	1	2	3	4	5	6
Probabilits	0,15	0,15	0,15	0,15	0,2	0,2

En *Python*, un tableau s'crit entre crochets, mais ses lments sont compts partir de 0, pas de 1. On va donc ajouter un 0 au dbut du tableau *loi* (la probabilit que le d donne 0 est nulle) :

```
omega=set(range(1,7))
loi=[0]+[0.15]*4+[0.2]*2

def proba(evt):
        return sum(loi[x] for x in evt)

petit=set(range(1,5))
pair=set(range(2,7,2))

print(proba(petit))
print(proba(pair))
```

2°) Cas gnral

La probabilit d'un vnement est la somme des probabilits des vnements lmentaires qui le constituent (ses issues).

Chapitre 7

Fonctions affines

I/ Dfinitions

1°) Fonction affine

Une fonction f est dite **affine** s'il existe deux rels a et b tels que $f(x) = ax + b$.

Exemple : Un transporteur demande 1,5 € par kilomtre parcouru, auxquels il ajoute 20 € pour l'assurance. Alors le prix est fonction affine de la distance.

```
def prix(distance):
        return 1.5*distance+20
```

2°) Coefficient directeur

Le nombre a s'appelle le coefficient directeur de la fonction.
Exemple : Les 1,5 € par kilomtre.

3°) Ordonne l'origine

Le nombre b s'appelle l'ordonne l'origine de la fonction.
Exemple : Les 20 €.

4°) Cas particuliers

a) Fonctions linaires

Si $b = 0$, $f(x) = ax$ est **linaire**.

b) Fonctions constantes

Si $a = 0$, $f(x) = b$ est **constante**.

II/ Reprsentation graphique

1°) Allure

La reprsentation graphique d'une fonction affine est une droite.

2°) Rciproque

Rciproquement, si une fonction est reprsente par une droite, elle est affine, et le nombre a s'appelle coefficient directeur de la droite, et b s'appelle l'ordonne l'origine de la droite.

3°) Dtermination

a) Ordonne l'origine

Comme l'image de 0 est b, la droite coupe l'axe des ordonnes au point de coordonnes $(0; b)$.

b) Coefficient directeur

Si x_1 et x_2 sont deux nombres,

$$a = \frac{f(x_2) - f(x_1)}{x_2 - x_1}$$

III/ Variations

1°) Cas o $a > 0$

Si $a > 0$, la fonction $f(x) = ax + b$ est croissante sur $\mathbb{R}$. Son tableau de variation ressemble

x	$-\infty$	$+\infty$
$ax + b$	$\nearrow$	

2°) Cas o $a < 0$

Si $a < 0$, la fonction $f(x) = ax + b$ est dcroissante sur $\mathbb{R}$. Son tableau de variation ressemble

x	$-\infty$		$+\infty$
$ax + b$		$\searrow$	

IV/ Signe

1°) Antcdent de 0

La fonction $ax + b$ s'annule lorsque $x = -\dfrac{b}{a}$.

2°) Tableau de signe

a) Cas o $a > 0$

x	$-\infty$		$-\frac{b}{a}$		$+\infty$
$ax + b$		$-$	0	$+$	

b) Cas o $a < 0$

x	$-\infty$		$-\frac{b}{a}$		$+\infty$
$ax + b$		$+$	0	$-$	

Chapitre 8

Coordonnes

I/ Point

1°) Repre

Un repre du plan est donn par trois points O, I et J non aligns. La droite (OI) s'appelle l'**axe des abscisses** et elle est gnralement horizontale, dirige vers la droite. La droite (OJ) s'appelle l'**axe des ordonnes** et elle est en gnral dirige vers le haut. Le point O est l'**origine** du repre. En posant $\vec{\imath} = \overrightarrow{OI}$ et $\vec{\jmath} = \overrightarrow{OJ}$, on note aussi $(O, \vec{\imath}, \vec{\jmath})$ le repre.

Si $(OI) \perp (OJ)$, le repre est dit **orthogonal**. Si, de plus, $OI = OJ$, le repre est **orthonorm**.

2°) Coordonnes

a) Dfinition

Un point est dtermin par ses deux coordonnes, l'abscisse et l'ordonne, notes respectivement x et y. Pour noter que le point M a pour coordonnes x et y, on crit $M(x; y)$. Par exemple, $O(0; 0)$, $I(1; 0)$ et $J(0; 1)$.

b) Programmation objet

Pour crer un objet *point* en *Python*, on va l'appeler *classe*, et dans la classe, on va placer des *mthodes* de l'objet. La premire est celle qui s'excute chaque fois qu'on cre un point, elle s'appelle toujours __init__ :

Le caractre "_" s'obtient en appuyant sur la touche 8 du haut du clavier.

```
class Point:
        def __init__(self,x,y):
                self.x=x
```

```
        self.y=y
```

Pour accder l'abscisse d'un objet *obj*, on crit *obj.x* ; ici l'objet est le point lui-mme, d'o le *self* (*soi* en Anglais).

Cette fonction ne fait que crer les coordonnes du point et les stocker dans les variables x et y. Pour crer le point $M(3;2)$ on fera donc

```
M=Point(3,2)
```

Une mthode d'affichage possible est celle-ci :

```
    def __str__(self):
        return '('+str(self.x)+';'+str(self.y)+')'
```

(le dcalage vers la droite (« indentation ») signifie que la dfinition de la mthode est l'intrieur de la « classe »)

II/ Milieu

1°) Coordonnes

On considre deux points $A\,(x_A;y_A)$, $B\,(x_B;y_B)$ et leur milieu $M\,(x_M;y_M)$

a) Abscisse

L'abscisse du milieu est la moyenne des abscisses

$$x_M = \frac{x_A + x_B}{2}$$

b) Ordonne

L'ordonne du milieu est la moyenne des ordonnes

$$y_M = \frac{y_A + y_B}{2}$$

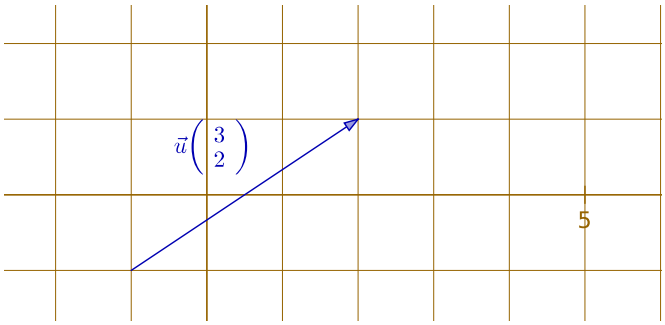
c) En Python

```
    def milieu(self,p):
        xM=(self.x+p.x)/2
        yM=(self.y+p.y)/2
        return Point(xM,yM)
```

III/ Vecteur

1°) Coordonnes

Un vecteur aussi a des coordonnes dans un repre. Pour crire que le vecteur $\vec{u}$ a pour coordonnes 3 et 2, on crit $\vec{u}(3;2)$ ou $\vec{u}\begin{pmatrix}3\\2\end{pmatrix}$.



a) Abscisse

L'abscisse du vecteur $\overrightarrow{AB}$ est la diffrence entre les abscisses de A et B.

$$x_{\overrightarrow{AB}} = x_B - x_A$$

b) Ordonne

L'ordonne de $\overrightarrow{AB}$ est la diffrence entre les ordonnes.

$$y_{\overrightarrow{AB}} = y_B - y_A$$

2°) En Python

a) Vecteur

Comme pour le point, on peut donc crer un objet *Vecteur* avec les mmes proprits x et y et la mme mthode d'affichage :

```
class Vecteur:
        def __init__(self,x,y):
                self.x=x
                self.y=y
        def __str__(self):
                return '('+str(self.x)+';'+str(self.y)+')'
```

b) Points

On peut aussi dfinir un vecteur partir de deux points, mais ici ce sera une mthode de l'objet *Point* :

```
        def vecteur(self,p):
                return Vecteur(p.x-self.x,p.y-self.y)
```

Exemple : Dans un repre, on donne $A(-3;5)$ et $B(3;-2)$. On demande les coordonnes, dans le mme repre, du milieu M de $[AB]$, ainsi que du vecteur $\overrightarrow{AB}$. On suppose que les classes prcdentes ont t enregistres dans des fichiers appels *points.py* et *vecteurs.py*.

```
from points import *
from vecteurs import *
A=Point(-3,5)
B=Point(3,-2)
print(A.milieu(B))
print(A.vecteur(B))
```

Chapitre 9

Espace

Chapitre 10

Addition des vecteurs

I/ Somme de vecteurs

1°) Coordonnes

tant donns deux vecteurs $\vec{u}$ et $\vec{v}$, on dfinit leur somme comme le vecteur $\vec{u}+\vec{v}$ dont

1: l'abscisse est la somme des abscisses de $\vec{u}$ et $\vec{v}$

$$x_{\vec{u}+\vec{v}} = x_{\vec{u}} + x_{\vec{v}}$$

2: l'ordonne est la somme des ordonnes

$$y_{\vec{u}+\vec{v}} = y_{\vec{u}} + y_{\vec{v}}$$

2°) Python

La mthode *__add__* correspond au symbole $\boxed{+}$, c'est donc celle-l qu'on va dfinir pour l'objet *Vecteur* (associ un autre vecteur) :

```
class Vecteur:
        def __init__(self,x,y):
                self.x=x
                self.y=y
        def __str__(self):
                return '('+str(self.x)+';'+str(self.y)+')'
        def __add__(self,v):
                return Vecteur(self.x+v.x,self.y+v.y)
```

II/ Addition des vecteurs

1°) Translations successives

Si on va de A vers B, puis de B vers C, on accomplit une translation de A vers C : $\overrightarrow{AB} + \overrightarrow{BC} = \overrightarrow{AC}$.

Cette relation porte le nom de Michel CHASLES, mathmaticien franais du 19e sicle.

2°) Exemple

On donne $A(-3;5)$, $B(3;-2)$ et $C(5;8)$ dans un repre. On peut vrifier que $\vec{w} = \overrightarrow{AC}$ est la somme de $\vec{u} = \overrightarrow{AB}$ et $\vec{v} = \overrightarrow{BC}$, tout simplement en comparant leurs coordonnes :

```
from points import *
from vecteurs import *
A=Point(-3,5)
B=Point(3,-2)
C=Point(5,8)
u=A.vecteur(B)
v=B.vecteur(C)
w=A.vecteur(C)
print(u+v)
print(w)
```

3°) paralllogramme

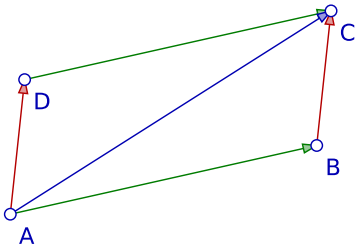
Soit $ABCD$ un paralllogramme. Alors les galits suivantes sont vraies :

1: $\overrightarrow{AB} = \overrightarrow{DC}$

2: $\overrightarrow{AB} + \overrightarrow{BC} = \overrightarrow{AC}$

3: $\overrightarrow{AB} + \overrightarrow{AD} = \overrightarrow{AC}$

La rgle du paralllogramme



Chapitre 11

Trinmes

I/ Fonction "carr"

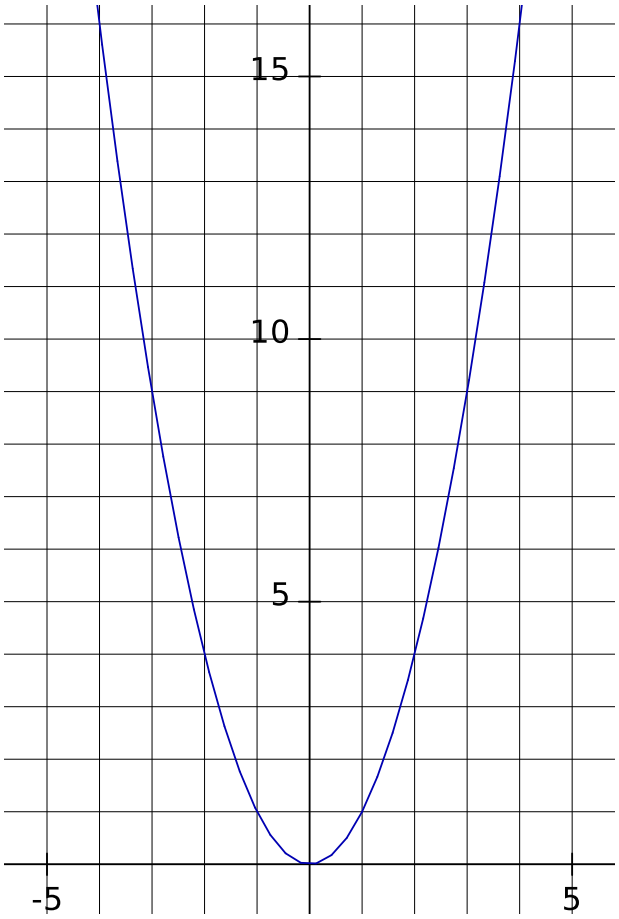
1°) Dfinition

Le carr d'un rel x est le produit de x par lui-mme : $x^2 = x \times x$.

```
def carr (x):
        return x**2
```

La fonction "carr" est dfinie sur $\mathbb{R}$.

2°) Reprsentation graphique



3°) Variations

a) Tableau de variations

x	$-\infty$		0		$+\infty$
x^2		$\searrow$	0	$\nearrow$	

b) Minimum

La fonction "carr" admet un minimum en 0, et ce minimum est 0 : $x^2 \geqslant 0$.

Un carr est donc toujours positif.

II/ Trinmes

1°) Dfinition

On dit qu'une fonction f est du second degr (ou *trinme*) s'il existe trois rels $a \neq 0$, b et c tels que $f(x) = ax^2 + bx + c$.

2°) Variations

a) Extremum

La fonction $f(x) = ax^2 + bx + c$ passe par un minimum ou un maximum en $x = -\dfrac{b}{2a}$. Sa reprsentation graphique est une **parabole** d'axe vertical (l'quation de l'axe est $x = -\dfrac{b}{2a}$)

b) Cas o $a > 0$

x	$-\infty$		$-\frac{b}{2a}$		$+\infty$
$f(x)$		$\searrow$	$f\left(-\frac{b}{2a}\right)$	$\nearrow$	

c) Cas o $a < 0$

x	$-\infty$		$-\frac{b}{2a}$		$+\infty$
$f(x)$		$\nearrow$	$f\left(-\frac{b}{2a}\right)$	$\searrow$	

Chapitre 12

Statistique

I/ Tableaux

Un tableau T est une liste de donnes, la premire se notant T[0], la suivante T[1], etc. Pour trier le tableau T, on crit T.sort(). Dans ce chapitre, on va faire des statistiques sur les 1000 premires dcimales de $\sqrt{2}$ (pour des indices allant de 0 999 dans le tableau).

En anglais, trier se dit *to sort*...

```
from decimal import *
getcontext().prec=1001
rac2=str(Decimal(2).sqrt())
rac2=rac2[2:]
chiffres=[int(c) for c in rac2]
```

II/ Moyenne

```
def moyenne(tableau):
        return sum(tableau)/len(tableau)

print(moyenne(chiffres))
```

En choisissant des chiffres compltement au hasard, leur moyenne est 4,5 et ici, on a $4,482 \simeq 4,5$.

L'erreur d'approximation est $\frac{4,5-4,482}{4,5} \simeq 0,004$ soit 0,4 %.

III/ Quantiles

1°) Mdiane

```
def mediane(tableau):
        trie=sorted(tableau)
        n=len(trie)
        if n%2==1:
                return trie[n//2]
        else:
                return (trie[n//2-1]+trie[n//2])/2

print(mediane(chiffres))
```

Parmi les 1000 premires dcimales de $\sqrt{2}$, le *chiffre mdian* est donc 5.

2°) Quartiles

a) Premier quartile

```
def Q1(tableau):
        trie=sorted(tableau)
        n=len(trie)
        return trie[n//4]
```

Parmi les 1000 premires dcimales de $\sqrt{2}$, le premier quartile est le chiffre 2.

b) Troisime quartile

```
def Q3(tableau):
        trie=sorted(tableau)
        n=len(trie)
        return trie[3*n//4]
```

Parmi les 1000 premires dcimales de $\sqrt{2}$, le troisime quartile est le chiffre 7.

IV/ Effectifs

1°) Comptage sous Python

a) Un effectif

Pour savoir combien de fois le 6 est sorti (l'effectif du 6), on peut faire

```
print(chiffres.count(6))
```

On apprend que le chiffre 6 figure 90 fois parmi les 1000 premires dcimales de $\sqrt{2}$.

b) Effectifs

Donc la liste des effectifs peut s'obtenir par

```
effectifs=[chiffres.count(n) for n in range(9)]
print(effectifs)
```

Exercice d'algorithmique :
crire un algorithme qui, partir des effectifs, donne les frquences.

2°) Effectifs cumuls

a) Effectifs cumuls croissants

```
effcum=[sum(effectifs[0:n]) for n in range(9)]
print(effcum)
```

Les effectifs cumuls se reprsentent en gnral par un polygone, mais le diagramme en btons est aussi possible.

b) Gnralisation

On dfinit galement les effectifs cumuls dcroissants, et les frquences cumules.

Les effectifs et frquences cumuls peuvent aider dterminer graphiquement les quartiles.

Chapitre 13

Fonctions homographiques

I/ Fonction "inverse"

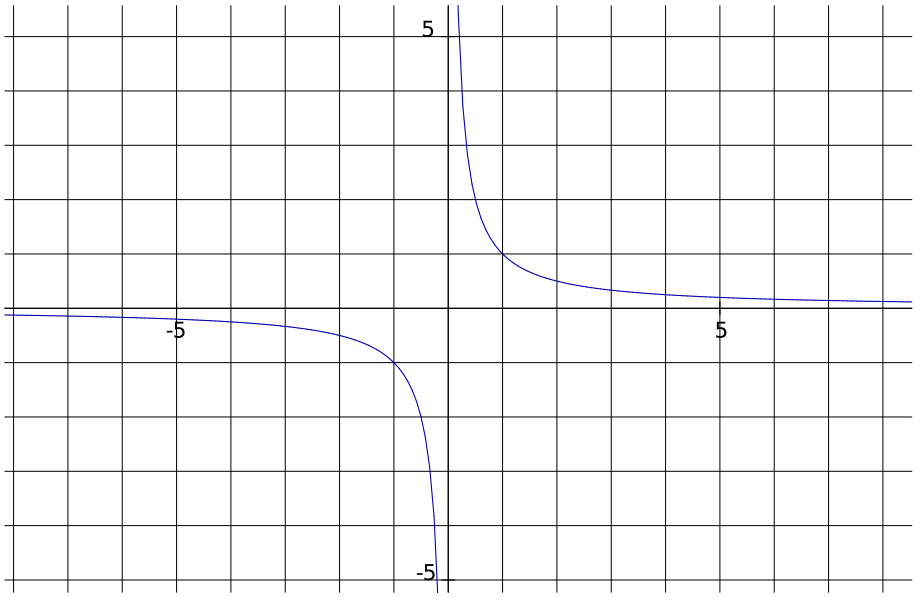
1°) Dfinition

L'inverse d'un rel x non nul est le quotient de 1 par x par lui-mme : $x^{-1} = \frac{1}{x}$.

```
def inverse(x):
        return x**(-1)
```

La fonction "inverse" est dfinie sur $]-\infty;0[\cup]0;+\infty[$.

2°) Reprsentation graphique



3°) Variations

a) Tableau de variations

x	$-\infty$		0		$+\infty$
$\frac{1}{x}$		$\searrow$		$\searrow$	

II/ Homographies

1°) Dfinition

On dit qu'une fonction h est homographique s'il existe quatre rels a, b, c et d tels que $h(x) = \dfrac{ax+b}{cx+d}$.

2°) Valeur interdite

a) Remarque

Si $x = -\dfrac{d}{c}$, le dnominateur s'annule : $-\dfrac{d}{c}$ est la **valeur interdite** de la fonction h.

b) Ensemble de dfinition

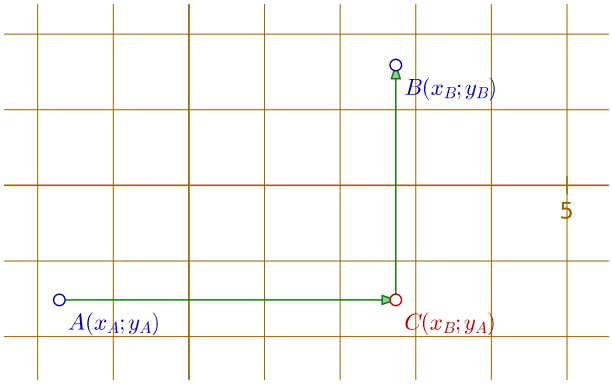
La fonction $h(x) = \dfrac{ax+b}{cx+d}$ est donc dfinie sur $\left]-\infty; -\dfrac{d}{c}\right[\cup \left]-\dfrac{d}{c}; +\infty\right[$.

Chapitre 14

Distances dans un repre orthonorm

On ne parle de mesures (distances ou angles) que dans un repre orthonorm.

I/ Pythagore



Dans un repre orthonorm, soient $A(x_A; y_A)$ et $B(x_B; y_B)$. Alors, en ajoutant le point $C(x_B; y_A)$, le vecteur $\overrightarrow{AC}$ a la mme abscisse que le vecteur $\overrightarrow{AB}$ et le vecteur $\overrightarrow{CB}$ a la mme ordonne que le vecteur $\overrightarrow{AB}$. Par ailleurs, le triangle ABC est rectangle en C **puisque le repre est orthonorm**, donc, d'aprs le thorme de Pythagore, le carr de la longueur de $\overrightarrow{AB}$ est la somme des carrs de ses coordonnes.

II/ longueur d'un vecteur

La longueur du vecteur $\vec{u}$ est donc donne par $\sqrt{x_{\vec{u}}^2 + y_{\vec{u}}^2}$.
On peut donc ajouter une proprit l'objet *Vecteur* de *Python* :

```
from math import *

class Vecteur:
        def __init__(self,x,y):
                self.x=x
                self.y=y
        def __str__(self):
                return '('+str(self.x)+';'+str(self.y)+')'
        def __add__(self,v):
                return Vecteur(self.x+v.x,self.y+v.y)
        def longueur(self):
                return hypot(self.x,self.y)
```

1°) Distance entre deux points

```
from math import *

class Point:
        def __init__(self,x,y):
                self.x=x
                self.y=y
        def __str__(self):
                return '('+str(self.x)+';'+str(self.y)+')'
        def milieu(self,p):
                xM=(self.x+p.x)/2
                yM=(self.y+p.y)/2
                return Point(xM,yM)
        def vecteur(self,p):
                return Vecteur(p.x-self.x,p.y-self.y)
        def distance(self,p):
                return self.vecteur(p).longueur()
```

Dans un repre orthonorm, la distance entre deux points A et B est donne par

Mais seulement dans un repre orthonorm, hein !

$$AB = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$

III/ Exemple

Dans un repre orthonorm, soient $A(3;2)$, $B(6;1)$ et $C(4;5)$. Dterminer la nature du triangle ABC.

```
from points import *
from vecteurs import *
A=Point(3,2)
B=Point(6,1)
C=Point(4,5)
d1=A.distance(B)
d2=A.distance(C)
d3=B.distance(C)
print(d1)
print(d2)
print(d1**2+d2**2)
print(d3**2)
```

Chapitre 15

Fonctions trigonomtriques

I/ Radians

1°) De degrs en radians

```
from math import *
print(radians(30))
print(pi/6)
```

2°) De radians en degrs

```
from math import *
print(degrees(pi/4))
```

II/ Fonctions trigonomtriques

1°) Cosinus

```
from math import *
print(cos(pi/6))
print(sqrt(3)/2)
```

2°) Sinus

```
from math import *
print(sin(pi/4))
print(sqrt(2)/2)
```

Chapitre 16

Direction

I/ Multiplication

1°) Dfinition

Le produit du vecteur $\vec{u}$ par le rel k est dfini comme le vecteur dont l'abscisse est $k \times x_{\vec{u}}$ et l'ordonne est $k \times y_{\vec{u}}$

2°) Nouvelle mthode

```
class Vecteur:
        def __rmul__(self,r):
                return Vecteur(self.x*r,self.y*r)
```

Pour utiliser cette mthode de l'objet vecteur, on utilise le symbole de multiplication :

```
from vecteurs import *
u=Vecteur(3,2)
print(2*u)
```

3°) Proprits

Lorsqu'un vecteur est le produit d'un autre vecteur par un nombre, on dit qu'ils sont **colinaires**.

Cela veut dire (en latin) qu'ils ont la mme direction

a) Droites parallles

Les droites (AB) et (CD) sont parallles si et seulement si les vecteurs $\overrightarrow{AB}$ et $\overrightarrow{CD}$ sont colinaires.

b) Points aligns

Les points A, B et C sont aligns si et seulement si les vecteurs $\overrightarrow{AB}$ et $\overrightarrow{AC}$ sont colinaires.

II/ Test de colinarit

1°) Produit en croix

Deux vecteurs $\vec{u}(x_{\vec{u}}, y_{\vec{u}})$ et $\vec{v}(x_{\vec{v}}, y_{\vec{v}})$ sont colinaires si et seulement si leurs coordonnes forment un tableau de proportionnalit, soit

$$x_{\vec{u}} \times y_{\vec{v}} = y_{\vec{u}} \times x_{\vec{v}}$$

2°) Test en Python

l'objet *Vecteur*, on peut ajouter une mthode de colinarit (avec un autre vecteur) :

```
from math import *

class Vecteur:
        def __init__(self,x,y):
                self.x=x
                self.y=y
        def __str__(self):
                return '('+str(self.x)+';'+str(self.y)+')'
        def __add__(self,v):
                return Vecteur(self.x+v.x,self.y+v.y)
        def longueur(self):
                return hypot(self.x,self.y)
        def __rmul__(self,r):
                return Vecteur(self.x*r,self.y*r)
        def colin(self,v):
                return self.x*v.y==self.y*v.x
```

3°) Exemple

On voudrait savoir si les points $A(2,1;1,3)$, $B(5,5;3,4)$ et $C(14,4;8,9)$ sont aligns. Pour cela on peut vrifier si les vecteurs $\overrightarrow{AB}$ et $\overrightarrow{AC}$ sont colinaires :

```
from points import *
from vecteurs import *
A=Point(2.1,1.3)
B=Point(5.5,3.4)
C=Point(14.4,8.9)
u=A.vecteur(B)
v=A.vecteur(C)
print(u.colin(v))
```

Chapitre 17

chantillonnage

I/ chantillon

1°) Dfinition

On constitue un **chantillon** en prlevant au hasard des lments d'un ensemble. On note N la taille de l'chantillon.

En latin, *alea* veut dire "les ds" ; en arabe, *al zahr* veut dire "les ds" ; en anglais, "au hasard" se dit *at random*.

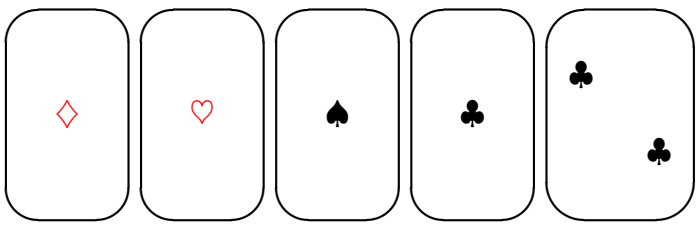
2°) Exemple

Au poker, une main de 5 cartes s'obtient en choisissant les cartes au hasard dans un jeu de 32 cartes :

```
from random import *
valeurs=['1','7','8','9','10','V','D','R']
couleurs=['\u2665','\u2666','\u2663','\u2660']
jeu=[v+'␣'+c for v in valeurs for c in couleurs]
print(jeu)

main=sample(jeu,5)
print(main)
```

En Anglais, "chantillon" se dit *sample*.



II/ Sondages

1°) Mthode

Pour effectuer un sondage, on prlve un chantillon au hasard (pour qu'il soit reprsentatif). On cherche estimer la proportion relle d'un caractre (par exemple, ceux qui sont pour un candidat) dans la population entire, partir de la proportion mesure dans l'chantillon. Pour cela on va inverser le problme, en estimant la probabilit que la proportion dans l'chantillon soit proche de la proportion relle. Et on va estimer cette probabilit par une mesure de frquence.

2°) Exemple

a) nonc

Dans une ville de 100000 habitants, 43000 ont l'intention de voter pour le maire sortant aux prochaines municipales, mais il ne le sait pas. Alors il va commanditer un sondage sur un chantillon de 100 personnes. 51 personnes parmi les 100 disent vouloir voter pour lui.

b) Simulation du sondage

On va simuler 1000 chantillons de 100 personnes chacun, et compter combien d'entre eux donnent l'impression que le maire sera rlu :

```
population=['contre']*57000+['pour']*43000
print(len(population))
print(sample(population,100))
```

Et pour compter les chantillons favorables au maire parmi les 1000 :

```
def favorables(ech):
        return [vote for vote in ech if vote=='pour']

def youpi():
        return len(favorables(sample(population,100)))>=50

p=len([n for n in range(1000) if youpi()])
print(p/1000)
```

III/ Intervalles de fluctuation

1°) Thorie

On admettra le rsultat suivant, o p dsigne la proportion de « pour » dans la population, et N la taille de l'chantillon :

La probabilit que la proportion de « pour » dans l'chantillon soit comprise entre $p - \frac{1}{\sqrt{N}}$ et $p + \frac{1}{\sqrt{N}}$ est 0,95.

2°) Intervalle de fluctuation

a) Dfinition

On dit que $\left[p - \frac{1}{\sqrt{N}}; p + \frac{1}{\sqrt{N}}\right]$ est un intervalle de fluctuation 95 % pour la proportion de « pour » dans la population.

b) Exemple

Avec $N = 100$, on a $\frac{1}{\sqrt{N}} = \frac{1}{10} = 0,1$ donc l'intervalle de fluctuation allait de 0,43-0,1=0,33 0,43+0,1=0,53. On constate qu'une partie de cet intervalle donne un succs au maire.

c) Pratique

En pratique, on ne connat pas p, alors on prend la place, la proportion dans l'chantillon. On parle alors d'***intervalle de confiance*** 95 % .

Exemple : Avec $p = 0,51$, on trouve comme intervalle de confiance

$$[0,41; 0,61]$$

Comme $0,43 \in [0,41; 0,61]$, le maire ne peut pas accuser l'intitut de sondage d'avoir trich.

Chapitre 18

Droites du plan

I/ Vecteur directeur

1°) Droite par deux points

```
import points
class Droite:
        def __init__(self,A,B):
                self.A=A
                self.B=B
```

Ici A et B sont des points.

2°) Vecteur

a) Dfinition

On dit que $\overrightarrow{AB}$ est un **vecteur directeur** de la droite (AB).

b) en Python

On peut ajouter une mthode l'objet *Droite* :

```
import points
import vecteurs
class Droite:
        def __init__(self,A,B):
                self.A=A
                self.B=B
        def directeur(self):
                return self.A.vecteur(self.B)
```

Deux droites sont parallles (ont la mme direction) si et seulement si elles ont un vecteur directeur en commun.

II/ quations cartsiennes

1°) Dfinition

On dit que l'quation $ax+by=c$ est une **quation cartsienne** de la droite d si $M(x_M, y_M) \in d \Leftrightarrow ax_M + by_M = c$.

2°) Mthode

```
        def __str__(self):
                a=-self.directeur().y
                b=self.directeur().x
                c=a*self.A.x-b*self.A.y
                eq='('+str(a)+')x+('+str(b)+')y='+str(c)
                return eq
```

III/ quation rduite

1°) Cas parallle l'axe des ordonnes

Si d est parallle (OJ), une de ses quations cartsiennes peut s'crire $x = c$: Cette quation est dite **rduite**.

2°) Autres cas

Sinon, d est la reprsentation graphique d'une fonction affine, et possde un coefficient directeur m et une ordonne l'origine p. Son quation rduite est alors $y = mx + p$.

a) Coefficient directeur

```
        def cd(self):
                return self.directeur().y/self.directeur().x
```

b) Ordonne l'origine

```
        def oalo(self):
                return self.A.y-self.cd()*self.A.x
```

c) quation rduite

```
        def reduite(self):
                if self.directeur().x==0:
                        eq='x='+str(self.A.x)
                else:
                        eq='y='
                        eq=eq+str(self.cd())+'x'
                        eq=eq+'+('+str(self.oalo())+')'
                        return eq
```

3°) Paralllisme

a) Thorme

Deux droites sont parallles si et seulement si elles ont le mme coefficient directeur.

b) En Python

```
        def parallele(self,autre):
                return self.cd()==autre.cd()
```

On pouvait aussi faire

```
def parallele(self,autre):
        return self.directeur().colin(autre.directeur())
```