# Programmes Processing: blocs, itérations, fonctions

Valeur d'accueil et reconversion en informatique 1 (VARI1)

Daniel Porumbel

**nombreaux slides dus à**

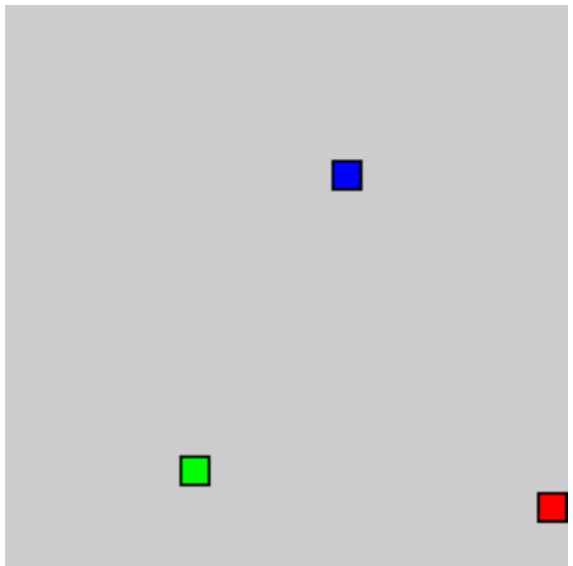
Pierre Cubaud

# Quel est le résultat du code ?

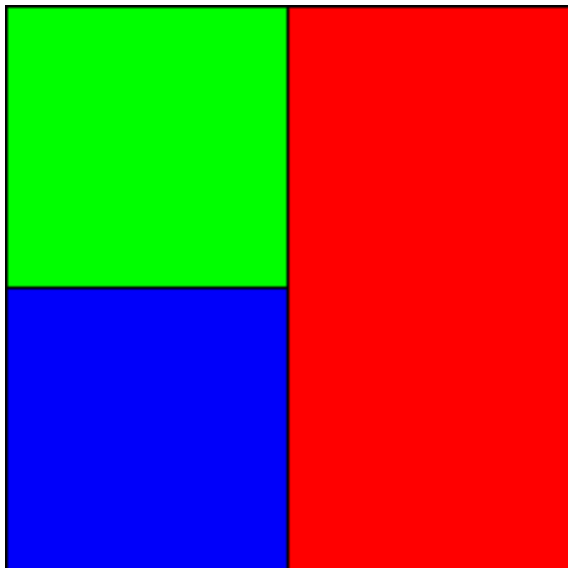
```
size(200,200);
fill(255,0,0);
rect(random(200),random(200), 10, 10);
fill(0,255,0);
rect(random(200),random(200), 10, 10);
fill(0,0,255);
rect(random(200),random(200), 10, 10);
```

# Quel est le résultat du code ?

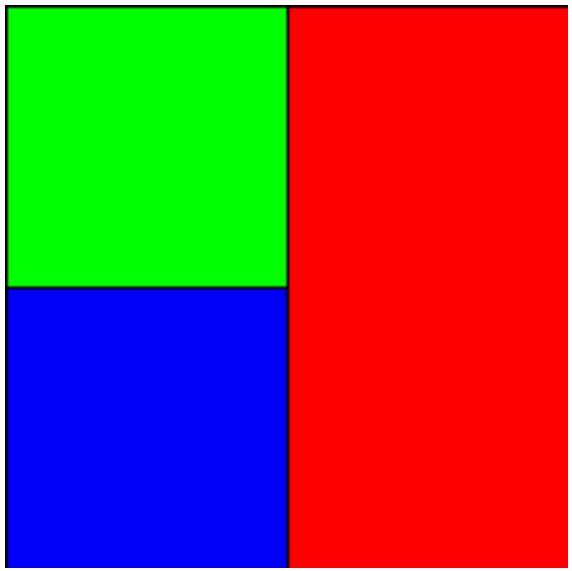
```
size(200,200);
fill(255,0,0);
rect(random(200),random(200), 10, 10);
fill(0,255,0);
rect(random(200),random(200), 10, 10);
fill(0,0,255);
rect(random(200),random(200), 10, 10);
```



# Comment réaliser ce dessin ?



# Comment réaliser ce dessin ?



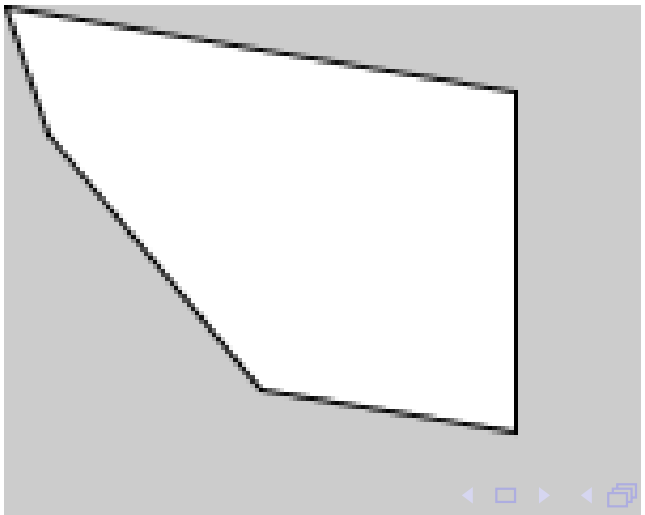
```
size(200,200);
fill(0,255,0);
rect(0,0,100,100);
fill(255,0,0);
rect(100,0,100,200);
fill(0,0,250);
rect(0,100,100,100);
```

# Les extrémités des lignes et les joins

```
size(350,120);
stroke(255,0,0);
strokeWeight(20);
strokeCap(ROUND);
line(10,10,100,100);
strokeCap(SQUARE);
line(100,10,10,100);
strokeJoin(BEVEL);
rect(200,20,100,60);
```

# Des polygones

```
size(150,150);
beginShape();
vertex(0,0);
vertex(120,20);
vertex(120,100);
vertex(60,90);
vertex(10,30);
vertex(0,0);
endShape();
```



## Les commentaires en Java

Il est très important de commenter ses programmes :
pour les autres ou pour soi-même plus tard

```
// commentaire sur une ligne

/*
 commentaires
 sur plusieurs lignes
*/

/**
 commentaire qui va etre reconnu par javadoc
*/
```

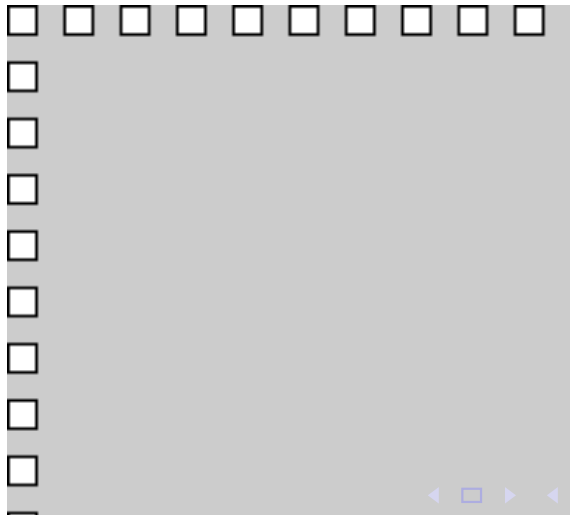
Il existe aussi des outils pour l'annotation (@author etc)

# Quel est le résultat du code ?

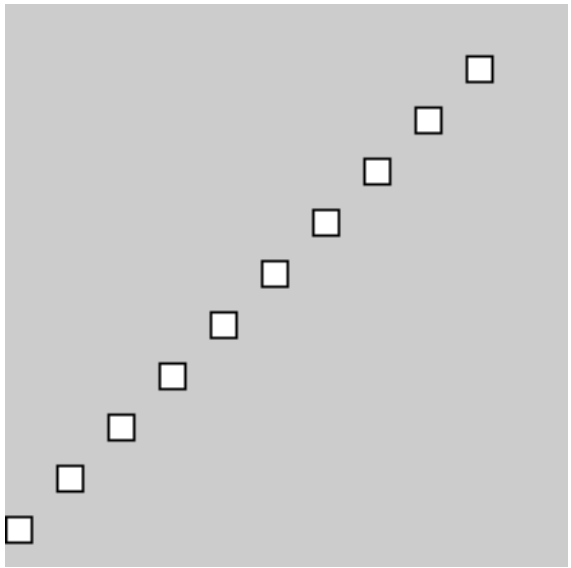
```
size(200,200);
for(int i=0;i<10;i++){
    int cx = i*20;
    rect(cx,0,10,10);
}
for(int i=0;i<10;i++){
    int cy = i*20;
    rect(0,cy,10,10);
}
```

# Quel est le résultat du code ?

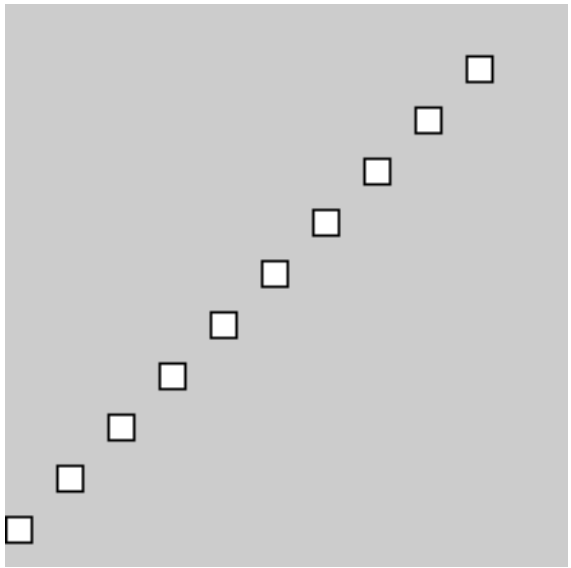
```
size(200,200);
for(int i=0;i<10;i++){
    int cx = i*20;
    rect(cx,0,10,10);
}
for(int i=0;i<10;i++){
    int cy = i*20;
    rect(0,cy,10,10);
}
```



# Comment réaliser une diagonale montante ?



# Comment réaliser une diagonale montante ?



```
size(220,220);
for(int i=0;i<10;i++){
    int cx = i*20;
    int cy = 200-i*20;
    rect(cx,cy,10,10);
}
```

**Le test** : if-then-(else)

```
if (heure==21){
 println("le cours est fini!");
 salaireProf = salaireProf+10;
}
```

pileface

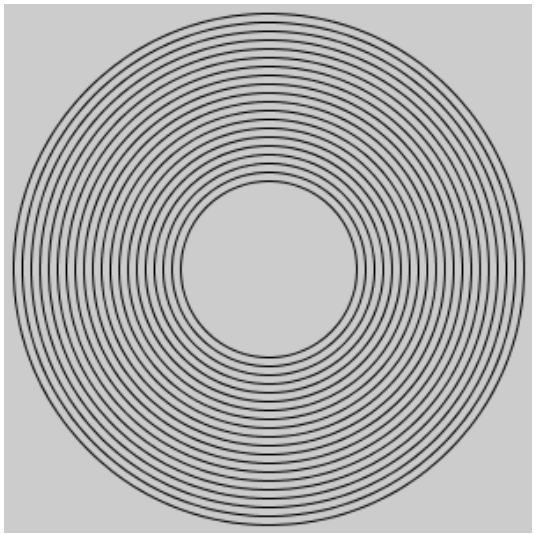
```
if (random(0,1)<0.5)
 println("pile");
else
 println("face");
```

imbrication :

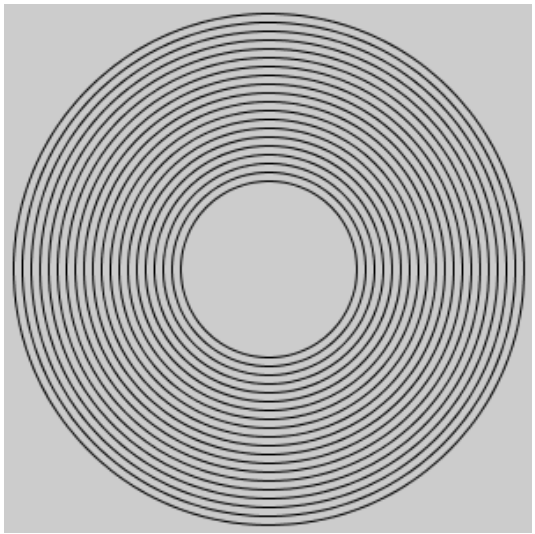
equation2

```
// delta=b*b-4*a*c;
if (delta>0)
 println("deux solutions");
else if (delta<0)
  println("pas de solution");
else
  println("une solution");
```

# Plusieurs cercles de même centre



# Plusieurs cercles de même centre



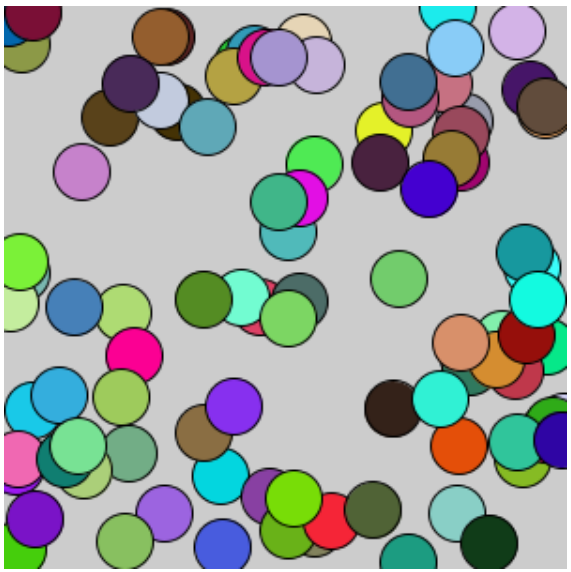
```
size(300,300);
noFill();
for(int i=100;i<300;i=i+10){
    ellipse(150,150, i,i);
}
```

# Quel est le résultat du code ?

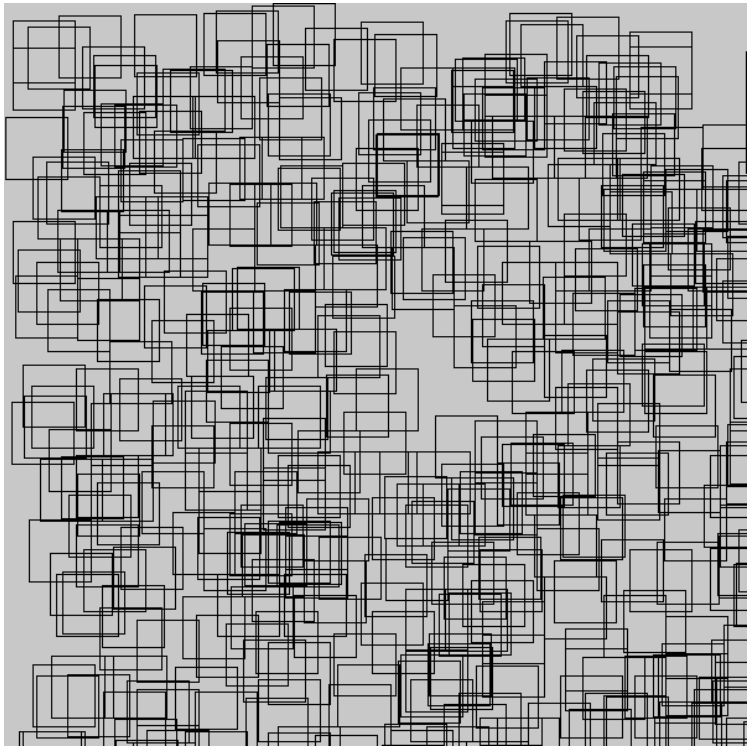
```
size(300,300);
for(int i=0;i<100;i=i+1){
    fill(random(255), random(255), random(255));
    ellipse(random(300), random(300), 30,30);
}
```

# Quel est le résultat du code ?

```
size(300,300);
for(int i=0;i<100;i=i+1){
    fill(random(255), random(255), random(255));
    ellipse(random(300), random(300), 30,30);
}
```



# Un premier programme itératif



repetition

```
size(600,600);
noFill();
int i=0;
do {
  float x = random(0,600);
  float y = random(0,600);
  rect(x,y,30,30);
  i = i+1;
} while (i<1000);
```

i joue le rôle d'un compteur

Deux autres variantes pour les itérations :

repetitionV2

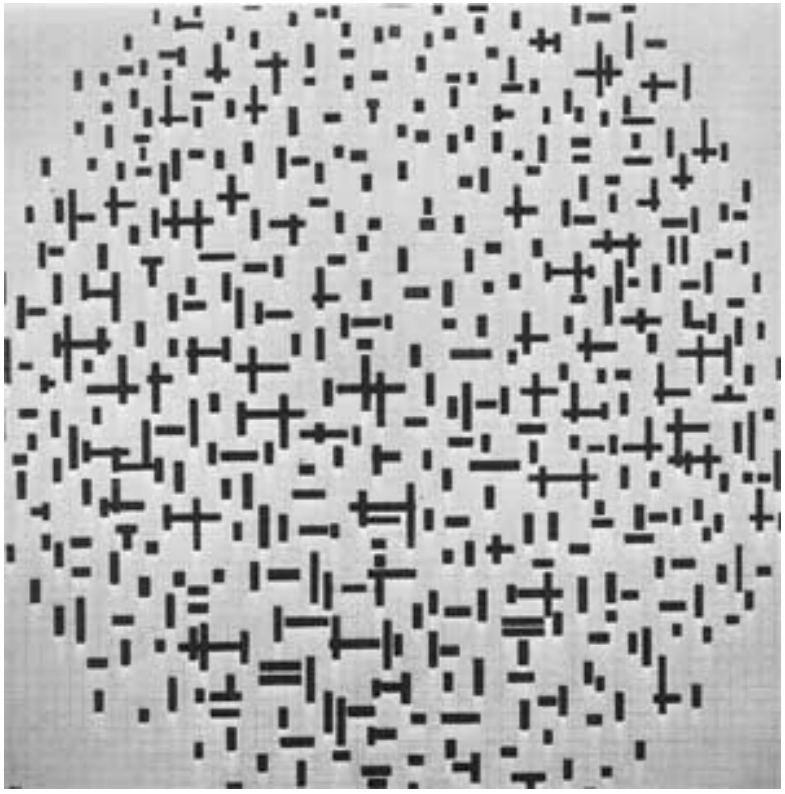
```
size(600,600);
noFill();
int i=0;
while (i<1000) {
  float x = random(0,600);
  float y = random(0,600);
  rect(x,y,30,30);
  i = i+1;
}
```

repetitionV3

```
size(600,600);
noFill();
for (int i=0;i<1000;i++) {
  float x = random(0,600);
  float y = random(0,600);
  rect(x,y,30,30);
}
```

Remarque : i++ raccourci pour i=i+1

## Test et itération



-tirer au sort la direction horizontale ou verticale
-tirer au sort x et y
-tester si dans le cercle
-répéter ...

une solution possible :

mondrian

```
size(600,600);
background(220);
fill(0);
for(int i=0;i<1000;i++){
  float x = random(0,600);
  float y = random(0,600);
  if (sqrt((x-300)*(x-300)+(y-300)*(y-300))
   // tirage au sort direction de trace
   if (random(0,1)<0.5)
     rect(x,y,5,20);
   else
     rect(x,y,20,5);
  }
}
```

# Boucles à retenir 1

- La plus classique boucle `for`

```
1       for(int i=0;i<100;i++){
2           ...//faire quelque chose (100 fois)
3       }
```

- La plus classique boucle `while`

```
1       int i = 0;
2       while(i<100){
3           ...//faire quelque chose
4           i = i+1;
5       }
```

# Boucles à retenir 2

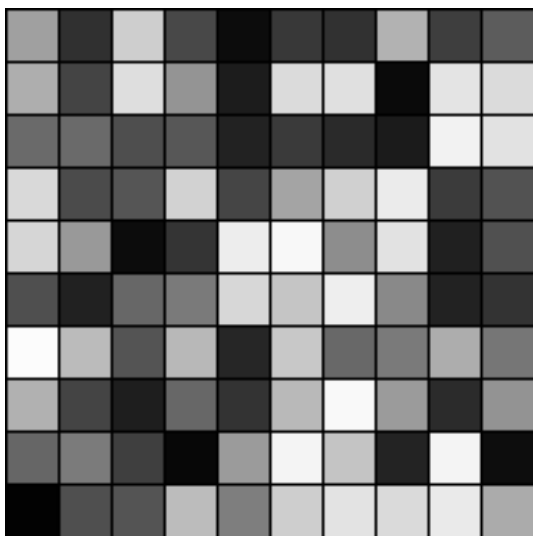
- Une boucle `for` imbriqué

```
1       size(200,200);
2       for(int i=0;i<10;i++){
3           for(int j=0;j<10;j++){
4               fill(random(255));
5               rect(i*20,j*20,20,20);
6           }
7       }
```

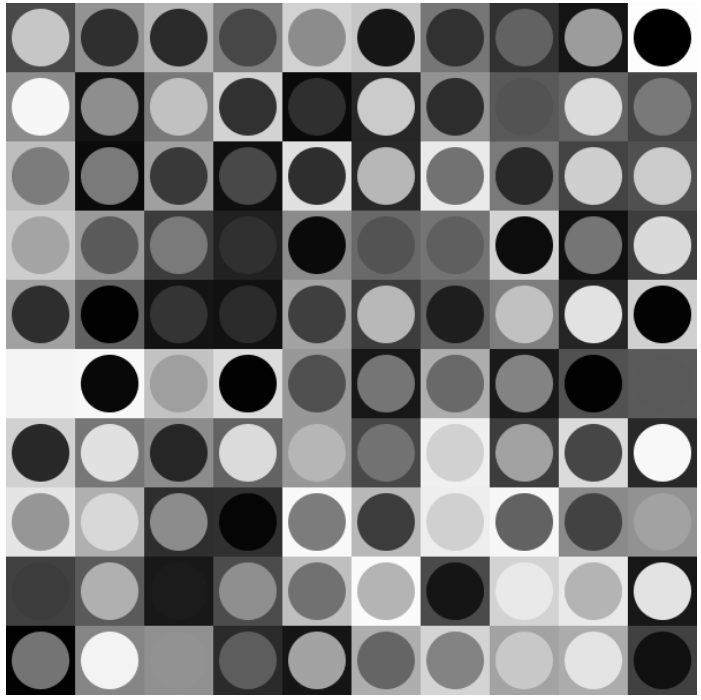
# Boucles à retenir 2

- Une boucle `for` imbriqué

```
1       size(200,200);
2       for(int i=0;i<10;i++){
3           for(int j=0;j<10;j++){
4               fill(random(255));
5               rect(i*20,j*20,20,20);
6           }
7       }
```



**Boucles imbriquées** : exemple de la trame



(en hommage à Julio LeParc)


leparc | Processing 2.0a3
STANDARD
leparc
size(600,600);
rectMode(CENTER);
ellipseMode(CENTER);
smooth();
noStroke();
for (int x=30; x<600; x+=60){
for (int y=30; y<600; y+=60){
fill(random(0,255));
rect(x,y,60,60);
fill(random(0,255));
ellipse(x,y,50,50);
}
}
save("leparc.png");
Done Saving.
1

# Une boucle sur des angles de cercle 1

Comment tracer une cercle de centre (100,100) et rayon 100 :

```
size(300,300);
float rayon = 100;
for(float i=0;i<2*PI;i=i+0.01){
    point(cos(i)*rayon+100,sin(i)*rayon+100);
}
```

# Une boucle sur des angles de cercle 2

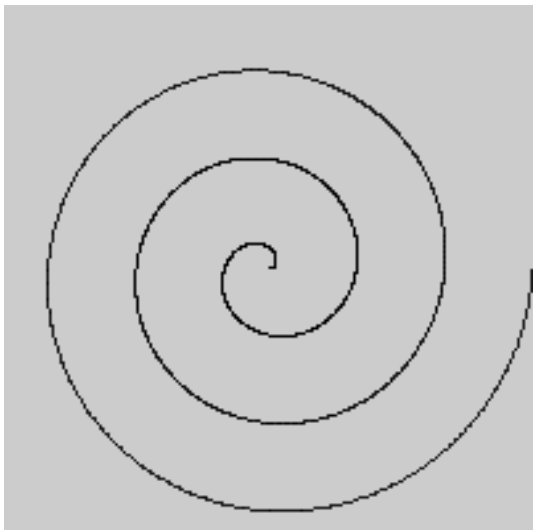
```
size(300,300);
float rayon = 100;
for(float i=0;i<6*PI;i=i+0.01){
    rayon = 100-100*(i/(6*PI));
    point(cos(i)*rayon+100,sin(i)*rayon+100);

}
```

# Une boucle sur des angles de cercle 2

```
size(300,300);
float rayon = 100;
for(float i=0;i<6*PI;i=i+0.01){
    rayon = 100-100*(i/(6*PI));
    point(cos(i)*rayon+100,sin(i)*rayon+100);

}
```



# Fonctions

Lorsqu'on appelle une fonction, il y a trois étapes.

**L'entrée** on fait « rentrer » des informations dans la fonction (on lui donne les données avec lesquelles travailler).

**Les calculs** grâce aux informations qu'elle a reçues en entrée, la fonction travaille.

**La sortie** une fois qu'elle a fini ses calculs, la fonction renvoie un résultat.

# Fonctions

Lorsqu'on appelle une fonction, il y a trois étapes.

**L'entrée** on fait « rentrer » des informations dans la fonction (on lui donne les données avec lesquelles travailler).

**Les calculs** grâce aux informations qu'elle a reçues en entrée, la fonction travaille.

**La sortie** une fois qu'elle a fini ses calculs, la fonction renvoie un résultat.


Une fonction qui renvoie rien (void) est aussi appelée méthode

# Fonctions 2 : `setup()` et `draw()`

## La méthode `setup()`

- appelée au début du programme
- on y met, par exemple, l'appel à `size(...)`

## La méthode `draw()`

- appelée de manière répétitive
- le nombre d'appels par seconde peut être modifié par la méthode `frameRate(nombre)`
- Pour faire une animation :
  - appeler `background(200)` au début de `draw()`
  - faire un dessin qui dépends d'une variable globale

# Fonctions 3

Écrire une fonction `somme(`$n$`)` qui calcule la somme

$$1 + 2 + 3 + \cdots + n$$

# Fonctions 3

Écrire une fonction `somme(`$n$`)` qui calcule la somme

$$1 + 2 + 3 + \cdots + n$$

```
int somme(int n){
    int s = 0;
    for(int i=0;i<n;i++){
        s = s + i;
    }
    return s;
}
```

# Fonctions 3

Écrire une fonction `somme(`$n$`)` qui calcule la somme

$$1 + 2 + 3 + \cdots + n$$

```
int somme(int n){
    int s = 0;
    for(int i=0;i<n;i++){
        s = s + i;
    }
    return s;
}
```

Écrire une fonction `factoriel(`$n$`)` qui calcule

$$n! = 1 \cdot 2 \ldots n$$

# Fonctions 3

Écrire une fonction `somme(`$n$`)` qui calcule la somme

$$1 + 2 + 3 + \cdots + n$$

```
int somme(int n){
    int s = 0;
    for(int i=0;i<n;i++){
        s = s + i;
    }
    return s;
}
```

Écrire une fonction `factoriel(`$n$`)` qui calcule

$$n! = 1 \cdot 2 \ldots n$$

# Fonctions 4

1. Écrire une fonction qui détermine si un nombre donné en entrée est premier ; la fonction doit renvoyer un boolean

# Fonctions 4

1. Écrire une fonction qui détermine si un nombre donné en entrée est premier ; la fonction doit renvoyer un boolean
2. Écrire un programme qui afiche les nombres premiers inférieurs à 100

# Quel est le résultat du code ?

```
1 void setup(){
2       size(200,200);
3       frameRate(30);
4 }
5 int i = 0;
6 void draw(){
7    fill(random(255),random(255),random(255));
8    rect(random(200),random(200)
9          ,random(50),random(50));
10 }
```

# Quel est le résultat du code ?

```
1 void setup(){
2     size(200,200);
3     frameRate(10);
4 }
5 int i = 0;
6 void draw(){
7     background(200);
8     ellipse(i,i,50,50);
9     i++;
10 }
```

# Quel est le résultat du code ?

```
1 void setup(){
2     size(200,200);
3     frameRate(10);
4 }
5 int i = 0;
6 void draw(){
7     background(200);
8     line(100,100, 100+100*sin(i*0.1),
9                   100+100*cos(i*0.1));
10    i++;
11 }
```

# Faire tourner une pyramide

- Chaque point de la base décrit un cercle
- La projection du cercle est une elipse : hauteur$=\frac{1}{2}$largeur

```
void setup() {
    size(300,300);
}
float i = 0;
void draw() {
    background(204);
    i++;
    float xa = 100 - 100*sin(i*0.1);
    float ya = 100 - 50*cos(i*0.1);
    float xb = 100 - 100*sin(i*0.1+2*PI/3);
    float yb = 100 - 50*cos(i*0.1 +2*PI/3);
    float xc = 100 - 100*sin(i*0.1+2*PI/3+2*PI/3);
    float yc = 100 - 50*cos(i*0.1 +2*PI/3+2*PI/3);

    line(xa,ya,xb,yb) ; //dessin
    line(xb,yb,xc,yc) ; //de la
    line(xc,yc,xa,ya) ; //base
    line(xa,ya,100,0) ;
    line(xb,yb,100,0) ; //le sommet=(100,0)
    line(xc,yc,100,0);
}
```