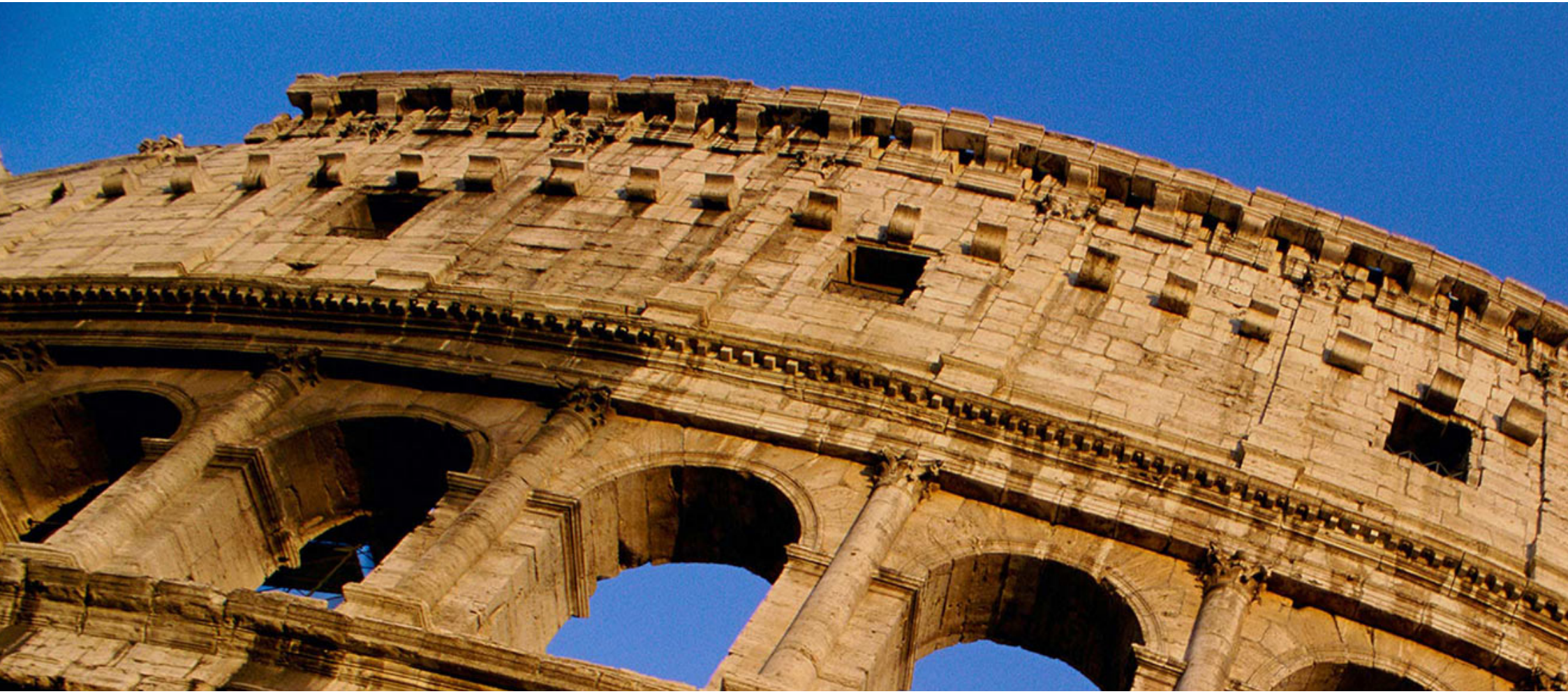


Algorithmique et programmation
Mise en œuvre en Python

Concepts fondamentaux

Introduction : qu'est-ce qu'un algorithme ?

- C'est une séquence d'instructions permettant d'atteindre un but qu'on s'est fixé (le problème à résoudre…)

- Ces instructions consistent à **modifier** l'existant, à **répéter** des opérations, à **tester** certaines conditions et à faire des **choix**.

Hey, mais c'est ce qu'on fait tous les jours !

Exemples d’algorithmes

- Une recette de cuisine est un algorithme qui utilise des ingrédients, des récipients et des techniques culinaires pour obtenir un plat.

Éplucher et découper en morceaux 4 Golden.

Faire une compote : les mettre dans une casserole avec un peu d'eau (1 verre ou 2). Bien remuer. Quand les pommes commencent à ramollir, ajouter un sachet ou un sachet et demi de sucre vanillé. Ajouter un peu d'eau si nécessaire.

Pendant que la compote cuit, éplucher et couper en quatre les deux dernières pommes, puis, couper les quartiers en fines lamelles.

Préchauffer le four à 210°C (thermostat 7).

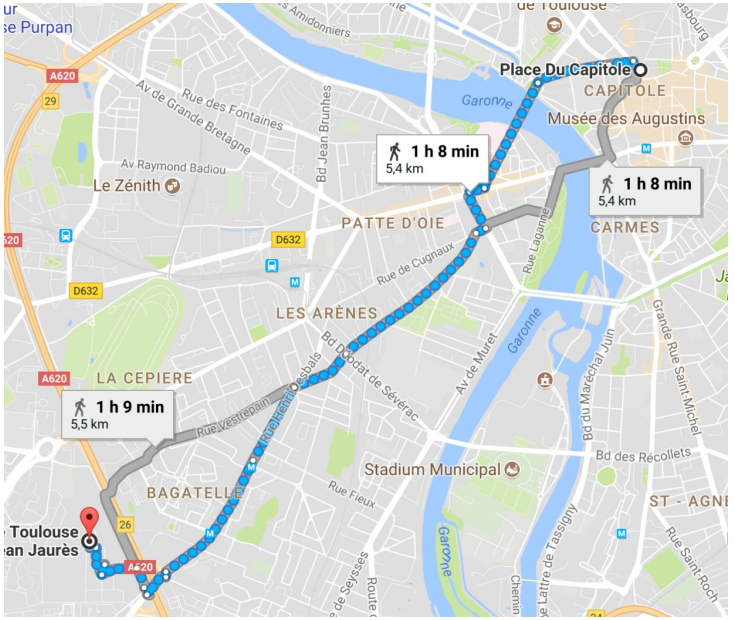
Laisser un peu refroidir la compote et étaler la pâte brisée dans un moule et la piquer avec une fourchette.

Verser la compote sur la pâte et placer les lamelles de pommes.

Mettre au four et laisser cuire pendant 30 min max. Surveiller la cuisson. Vous pouvez ajouter un peu de sucre vanillé sur la tarte pendant que çà cuit pour caraméliser un peu.

Exemples d’algorithmes

- Un trajet est un algorithme permettant, à partir d’un point de départ et d’un moyen de transport, d’arriver à l’endroit voulu en passant par différentes étapes.



Exemples d'algorithmes

- Un « algorithme informatique » est un algorithme qui utilise des valeurs, des emplacements mémoire et des instructions pour résoudre un problème.

```
lire a
lire b
lire c
delta = b * b - 4 * a * c
si delta > 0 alors …
```

Pour résumer, un algorithme est simplement l'expression d'une démarche dans un langage adapté au problème.

Algorithmes et processeurs

- Les algorithmes ne sont que des **plans**.

- Ils sont réalisés par des **processeurs** : c'est le cuisinier qui exécute la recette, c'est le conducteur qui fait le trajet, c'est un processeur qui exécute l'algorithme informatique.

- Les processeurs ont des **compétences** différentes : il y a de grands chefs et des apprentis cuisiniers, il y a des processeurs de bas niveau et des processeurs évolués.

Qu’est-ce qu’un programme ?

- Un **programme** est la traduction d’un algorithme dans le langage que comprendra le processeur chargé de l’exécuter.

- On voit donc que le **niveau de détail** de l’algorithme final dépend des compétences du processeur concerné.

Exemple d'une recette de cuisine : compétence d'un apprenti cuisinier

Éplucher et découper en morceaux 4 Golden.

Faire une compote : les mettre dans une casserole avec un peu d'eau (1 verre ou 2). Bien remuer. Quand les pommes commencent à ramollir, ajouter un sachet ou un sachet et demi de sucre vanillé. Ajouter un peu d'eau si nécessaire.

Pendant que la compote cuit, éplucher et couper en quatre les deux dernières pommes, puis, couper les quartiers en fines lamelles.

Préchauffer le four à 210°C (thermostat 7).

Laisser un peu refroidir la compote et étaler la pâte brisée dans un moule et la piquer avec une fourchette.

Verser la compote sur la pâte et placer les lamelles de pommes.

Mettre au four et laisser cuire pendant 30 min max. Surveiller la cuisson. Vous pouvez ajouter un peu de sucre vanillé sur la tarte pendant que çà cuit pour caraméliser un peu.

Qu’est-ce qu’un programme ?

- Principe des **raffinages successifs** : on part d’un algorithme grossier et on raffine jusqu’à avoir des instructions directement compréhensibles par le processeur/langage cible.

Faire une compote

les mettre dans une casserole avec un peu d'eau (1 verre ou 2). Bien remuer. Quand les pommes commencent à ramollir, ajouter un sachet ou un sachet et demi de sucre vanillé. Ajouter un peu d'eau si nécessaire.

L’algorithme ne dépend pas du processeur : il dépend uniquement du problème à résoudre. Mais son niveau de détail dépend des compétences du processeur.

Peter, le petit traceur

- Nous allons dans un premier temps travailler avec un processeur particulier que nous avons appelé Peter

- Il est capable de tracer des figures géométriques simples (traits, carrés, rectangles, …)

- nous allons écrire des algorithmes de tracé de figures géométriques réalisables par Peter

Peter, le petit traceur

- Peter est constitué physiquement d'un stylet posé sur un bras.
- Ses compétences lui permettent d'effectuer des tracés géométriques simples sur une feuille posée à plat sous le stylet.
- Le stylet peut être en position levée ou baissée (il touche ou pas la feuille)
- Le bras se déplace selon les 4 points cardinaux au dessus d’une feuille immobile.
- Au départ, le stylet est levé, au centre de la feuille ; l’orientation du bras est vers le nord.

Compétences de Peter

- Avant d'effectuer un tracé, il faut initialiser Peter avec la commande **init(titre) :**
  - titre sera le titre du dessin ; exemple : init("Mon beau dessin")
  - le stylet est alors placé au centre de la feuille, en position levée et il est dirigé vers le nord ;
  - la couleur de tracé est noire.
- Pour tracer un trait, il faut mettre le stylet en contact avec la feuille :
  - la commande **baisser()** permet de mettre en contact ;
  - **lever()** permet de remettre le stylet en position levée

Compétences de Peter

- La direction courante du stylet peut être directement modifiée à l'aide des commandes **est()**, **ouest()**, **nord()** et **sud()**.

- Les commandes **droite()** et **gauche()** permettent de changer la direction de 90° vers la droite ou la gauche.

- La commande **pivoter(nb_degrés)** change la direction du nombre de degrés indiqué **vers la droite**

  - exemple : pivoter(30)

Compétences de Peter

- Quel que soit son contact avec la feuille, le stylet avance d'un certain nombre de pas selon la direction courante avec la commande **avancer(nb_pas)**
  - exemple : avancer(4)
- Le stylet peut être replacé au centre de la feuille avec la commande **centrer()**. En ce cas, sa nouvelle direction est le **nord**. Attention si le stylet est baissé !!!
- La couleur du tracé peut être modifiée par la commande **couleur(coul)**, où **coul** est une chaîne de caractères valant "blanc", "noir", "rouge", "vert", "bleu", "jaune" ou "violet"
  - exemple : couleur("vert")
- Le tracé reste affiché tant que l'on n'a pas fait la commande **quitter()**.

Compétences de Peter (résumé de la liste des commandes)

- init(titre)
- centrer(), baisser() et lever()
- est(), ouest(), nord(), sud()
- droite(), gauche()
- pivoter(nb_degrés)
- avancer(nb_pas)
- couleur(coul)
- quitter()

Utilisation de Peter

- Notre première approche des algorithmes va consister à utiliser les compétences de Peter pour lui faire tracer des figures géométriques simples.

- Pour ce faire, nous écrirons les "recettes" indiquant comment, à partir d'un état initial, on fera appel aux compétences de Peter pour obtenir une figure particulière.

- Ceci nous permettra aussi d'introduire les premiers concepts de la programmation : registres, séquence d'instructions, répétitions, etc.

Premier exemple

Écrire la "recette" indiquant à Peter comment tracer un carré violet de 200 x 200 pas

Premier exemple

- Si le traceur Peter a toutes les compétences que l'on vient de présenter et que l'on veut tracer un carré violet de 200 x 200 pas, il faut :

    1. Initialiser le traceur (il est donc au centre, orienté vers le nord et levé, de couleur noire)
    2. Changer la couleur en violet
    3. Baisser le stylet
    4. Avancer de 200 pas (donc vers le nord), tourner à droite (donc vers l'est), avancer de 200 pas, tourner à droite (donc vers le sud), avancer de 200 pas, tourner à droite (donc vers l'ouest), avancer de 200 pas.
    5. Lever le stylet.

- Ceci est **l'algorithme** (dans le niveau de détail des compétences de Peter). Il faut le traduire dans le langage de Peter pour obtenir le **programme**.

Premier exemple

- L'algorithme :

1. Initialiser le traceur (il est donc au centre, orienté vers le nord et levé, de couleur noire)
2. Changer la couleur en violet
3. Baisser le stylet
4. Avancer de 200 pas (donc vers le nord), tourner à droite (donc vers l'est), avancer de 200 pas, tourner à droite (donc vers le sud), avancer de 200 pas, tourner à droite (donc vers l'ouest), avancer de 200 pas.
5. Lever le stylet.

- Le programme :

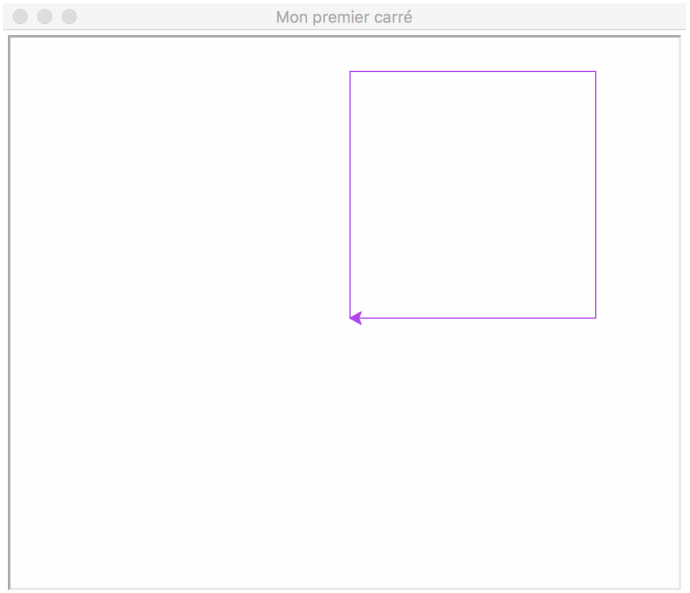
```
from robot import *
init("Mon premier carré")
centrer()
couleur("violet")
baisser()
avancer(200)
droite()
avancer(200)
droite()
avancer(200)
droite()
avancer(200)
lever()
quitter()
```

Premier exemple

- Le programme :

```
from robot import *
init("Mon premier carré")
centrer()
couleur("violet")
baisser()
avancer(200)
droite()
avancer(200)
droite()
avancer(200)
droite()
avancer(200)
lever()
quitter()
```

- et le résultat de son exécution :



Remarques

- Ce programme « marche » mais on voit bien qu'il a quelques défauts :
  - On répète 4 fois la valeur 200. Si on veut maintenant tracer un carré de 300x300, il faudra faire 4 modifications.
  - Le tracé du carré consiste à **répéter** 4 fois la séquence avancer/droite…

```
from robot import *
init("Mon premier carré")
centrer()
couleur("violet")
baisser()
avancer(200)
droite()
avancer(200)
droite()
avancer(200)
droite()
avancer(200)
lever()
quitter()
```

Variables et répétitions

- Le premier problème peut se résoudre aisément avec un **registre mémoire** (également appelé **« variable »**) : on **stocke** le nombre de pas dans un registre et on utilise ensuite ce registre. Si l'on veut un carré plus grand, il suffit de modifier la valeur de ce registre ; les autres instructions resteront inchangées.

- Pour le deuxième problème,

  - on préférera simplement dire **faire 4 fois la séquence avancer/droite** car en informatique, on a horreur de se répéter… Mais il faut trouver un moyen de **compter** le nombre de fois où l'on fait quelque chose.

  - Il faut donc un compteur. Mais qu'est-ce qu'un compteur sinon un registre mémoire allant de 1 à la valeur voulue ?

  - C'est exactement ce que permet de faire la boucle **pour** (qui s'appelle **for** en programmation).

Premier exemple amélioré

- Notre traceur a donc besoin de trois nouvelles compétences :
  - Pouvoir **stocker** une valeur dans un registre.
  - Pouvoir **lire** la valeur contenue dans un registre
  - Pouvoir **répéter** une séquence d'actions un certain nombre de fois.

Premier exemple amélioré

- D'où le programme final :

```
from robot import *
init("Mon premier carré")
nb_pas = 200
centrer()
couleur("violet")
baisser()
for nb_cotes in range(4):
    avancer(nb_pas)
    droite()
lever()
quitter()
```

Définition et initialisation d'un registre

Mise en place d'une boucle qui répétera 4 fois les deux

Le programme initial :

```
from robot import *
init("Mon premier carré")
centrer()
couleur("violet")
baisser()
avancer(200)
droite()
avancer(200)
droite()
avancer(200)
droite()
avancer(200)
lever()
quitter()
```

Remarques

- L'initialisation d'un registre se fait simplement avec le signe égal. Attention à choisir un nom de registre « parlant ».

- La lecture de la valeur d'un registre est automatique : il suffit de mettre le nom du registre et il sera remplacé par sa valeur.

```
from robot import *
init("Mon premier carré")
nb_pas = 200
centrer()
couleur("violet")
baisser()
for nb_cotes in range(4):
    avancer(nb_pas)
    droite()
lever()
quitter()
```

Remarques

attention à la syntaxe !

c'est la forme que nous utiliserons dans l'algorithme

- **for cpteur in range(x):** se lit « *pour cpteur variant de 0 à x non compris* ».
  - Autrement dit, on *répète x fois* la suite d'instructions contenue dans le for.
- Avec **for cpteur in range(deb, fin):** *cpteur* varie de *deb* à *fin non compris*. Nous étudierons cette forme plus tard…

```
from robot import *
init("Mon premier carré")
nb_pas = 200
centrer()
couleur("violet")
baisser()
for nb_cotes in range(4):
    avancer(nb_pas)
    droite()
lever()
quitter()
```

attention à la syntaxe !

Exercice

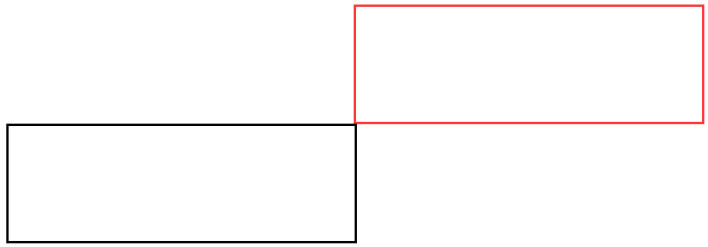
Écrire l'algorithme pour tracer un carré noir de 10 x 10 pas. Le traduire dans le langage de Peter.

Exercice

Écrire l'algorithme pour tracer un rectangle vert de 10 x 50 pas. Le traduire dans le langage de Peter.

Exercice

Écrire l'algorithme pour tracer la figure suivante. Le traduire dans le langage de Peter.



Bilan

- Vous savez utiliser des registres mémoire pour stocker des valeurs que vous pouvez relire ensuite.
- Vous savez faire des boucles avec des compteurs.
- Vous connaissez le langage et les compétences d'un processeur particulier : Peter le traceur.

Premier exemple : suite

- Dans le prochain exercice, on va supposer que Peter ne sait avancer que d’un seul pas à la fois. Comment faire le même tracé que précédemment ?

L’algorithme est le même (c'est le même problème à résoudre). Mais il faut modifier son niveau de détail pour s'adapter aux compétences du processeur (Peter).

Premier exemple : suite

- Comment tracer un carré de **nb_pas** par **nb_pas** sachant que Peter ne sait avancer que d'un pas à la fois ?

- En mettant un pied devant l'autre ! C'est-à-dire en répétant l'opération **avancer()** qui, maintenant, n'avance que d'un seul pas. Peter a donc perdu en compétence, mais on va voir que ce n'est pas un problème.

- Par rapport à l'algorithme précédent, il suffit de détailler l'instruction **avancer(nb_pas)** : "pour avancer de **nb_pas**, il faut faire **nb_pas"**, autrement dit, "avancer **nb_pas** fois d'un pas"…

- On utilisera donc un compteur de pas :

```
for cpteur_pas in range(nb_pas):
    avancer()
```

Premier exemple : suite

- Le programme final devient donc :

```
from robot import *
init("Mon premier carré")
nb_pas = 200
centrer()
couleur("violet")
baisser()
for nb_cotes in range(4):
    for cpteur_pas in range(nb_pas):
        avancer()
    droite()
lever()
quitter()
```

On trace 4 côtés

On avance de nb_pas

le programme précédent :

```
from robot import *
init("Mon premier carré")
nb_pas = 200
centrer()
couleur("violet")
baisser()
for nb_cotes in range(4):
    avancer(nb_pas)
    droite()
lever()
quitter()
```