# Formation Python – 13 juin 2017

Laurent Bourneuf
laurent.bourneuf@ac-nantes.fr

Vincent Labbé
vincent.labbe@ac-nantes.fr

Nicolas Winspeare
nicolas.winspeare@ac-nantes.fr

Document à jour au 14 juin 2017

**Résumé**

Le programme de mathématiques de seconde est aménagé pour la rentrée 2017. Le document d'aménagement indique : « Un langage de programmation simple d'usage est nécessaire pour l'écriture des programmes. Le choix du langage se fera parmi les langages interprétés, concis, largement répandus, et pouvant fonctionner dans une diversité d'environnements. »

Le langage Python présente toutes les caractéristiques requises. Il est utilisé dans l'enseignement supérieur, dans la recherche et dans les entreprises. Ce document propose des ressources concernant Python pour des professeurs de mathématiques de lycée.

## Table des matières

# 0 L'algorithmique au collège (avec Scratch ou équivalent)

## 0.1 Les programmes et les documents d'accompagnement

Sur le site de l'académie de Nantes, page programmes, on trouvera des liens vers les documents de la réforme du collège de la rentrée 2016.

- Programme – Les mathématiques et la technologie (pour l'algorithmique) du cycle 4 : PDF.
- Documents ressource du cycle 4 – Algorithmique et programmation : PDF.
- Document ressource de l'académie de Nantes – Algorithmique et programmation, un levier pour développer des compétences mathématiques : page web et PDF.

### Attendus de fin de cycle

Écrire, mettre au point et exécuter un programme simple

| **Connaissances et compétences associées** | **Exemples de situations, d'activités et de ressources pour l'élève** |
|---|---|
| Décomposer un problème en sous-problèmes afin de structurer un programme ; reconnaître des schémas.<br>Écrire, mettre au point (tester, corriger) et exécuter un programme en réponse à un problème donné.<br>Écrire un programme dans lequel des actions sont déclenchées par des événements extérieurs.<br>Programmer des scripts se déroulant en parallèle.<br>» Notions d'algorithme et de programme.<br>» Notion de variable informatique.<br>» Déclenchement d'une action par un évènement, séquences d'instructions, boucles, instructions conditionnelles. | Jeux dans un labyrinthe, jeu de Pong, bataille navale, jeu de nim, tic tac toe.<br>Réalisation de figure à l'aide d'un logiciel de programmation pour consolider les notions de longueur et d'angle.<br>Initiation au chiffrement (Morse, chiffre de César, code ASCII…).<br>Construction de tables de conjugaison, de pluriels, jeu du cadavre exquis…<br>Calculs simples de calendrier.<br>Calculs de répertoire (recherche, recherche inversée...).<br>Calculs de fréquences d'apparition de chaque lettre dans un texte pour distinguer sa langue d'origine : français, anglais, italien, etc. |

### Repères de progressivité :

En 5e, les élèves s'initient à la programmation événementielle. Progressivement, ils développent de nouvelles compétences, en programmant des actions en parallèle, en utilisant la notion de variable informatique, en découvrant les boucles et les instructions conditionnelles qui complètent les structures de contrôle liées aux événements.

FIGURE 1 – Programme de mathématiques, page 379

## Attendus de fin de cycle

» Comprendre le fonctionnement d'un réseau informatique
» Écrire, mettre au point et exécuter un programme.

(…)

| **Écrire, mettre au point et exécuter un programme** | |
|---|---|
| Analyser le comportement attendu d'un système réel et décomposer le problème posé en sous-problèmes afin de structurer un programme de commande.<br>Écrire, mettre au point (tester, corriger) et exécuter un programme commandant un système réel et vérifier le comportement attendu.<br>Écrire un programme dans lequel des actions sont déclenchées par des événements extérieurs.<br>» Notions d'algorithme et de programme.<br>» Notion de variable informatique.<br>» Déclenchement d'une action par un évènement, séquences d'instructions, boucles, instructions conditionnelles.<br>» Systèmes embarqués.<br>» Forme et transmission du signal.<br>» Capteur, actionneur, interface. | Concevoir, paramétrer, programmer des applications informatiques pour des appareils nomades.<br><br>Observer et décrire le comportement d'un robot ou d'un système embarqué. En décrire les éléments de sa programmation<br><br>Agencer un robot (capteurs, actionneurs) pour répondre à une activité et un programme donnés.<br><br>Écrire, à partir d'un cahier des charges de fonctionnement, un programme afin de commander un système ou un système programmable de la vie courante, identifier les variables d'entrée et de sortie.<br><br>Modifier un programme existant dans un système technique, afin d'améliorer son comportement, ses performances pour mieux répondre à une problématique donnée.<br><br>Les moyens utilisés sont des systèmes pluri-technologiques réels didactisés ou non, dont la programmation est pilotée par ordinateur ou une tablette numérique. Ils peuvent être complétés par l'usage de modélisation numérique permettant des simulations et des modifications du comportement. |

## Repères de progressivité

En 5e : traitement, mise au point et exécution de programme simple avec un nombre limité de variables d'entrée et de sortie, développement de programmes avec des boucles itératives.

En 4e : traitement, mise au point et exécution de programme avec introduction de plusieurs variables d'entrée et de sortie

En 3e : introduction du comptage et de plusieurs boucles conditionnels imbriqués, décomposition en plusieurs sous-problèmes

FIGURE 2 – Programme de technologie, pages 363 et 364

## 0.2 Ce qui a été vu au collège et que nous n'aborderons pas en mathématiques au lycée

Voici ce qui ne rentre pas dans le cadre du programme de mathématiques du lycée (mais qui pourrait être abordé en ICN ou en ISN)

- Programmation événementielle (par exemple le programme réagit aux touches directionnelles du clavier, ou quand des lutins s'envoient des messages et réagissent de façon adaptée)
- Programmation parallèle (par exemple différents scripts amenés à se dérouler en parallèle sont déclenchés par des événements différents)

Exemples issues du document d'accompagnement :

- Jeu à 3 personnages
- Jeu de pong.

# 1 Avant d'essayer Python

## 1.1 Ressources pour apprendre Python

Quelques ressources possibles :

- (Site web) Débuter avec Python au lycée (tutoriel Python pour lycéens) : http://python.lycee.free.fr/.
  Ce site est référencé sur les sites de mathématiques des académies de Rouen, de Limoges, et sur le site du département d'informatique de l'université d'Angers.
- (Site web) France IOI (voir plus loin) créé par une association agréée par le ministère de l'éducation nationale.
- (Livre) Manuel de spécialité ISN en terminale, *Informatique et sciences du numérique - Édition spéciale Python !* Une version numérique est disponible en fichier PDF (23,36 Mio) sur cette page.
- (Documentation PDF) Documentation EduPython. Contient de nombreux exemples intéressants pour les mathématiques [1]. Voir à la fin de ce document pour une présentation d'EduPython.
- (Livre) Gérard Swinnen, *Apprendre à programmer avec Python* 3 (5e édition). Liens : Site web, fichier PDF (6 Mio).

Il y a aussi beaucoup de pages de type « aide-mémoire Python » dans les livres de mathématiques de seconde.

## 1.2 Choix de l'environnement de travail : navigateur ou installation ?

Pour débuter, on n'est pas obligé d'installer Python !

En effet, on peut utiliser Python directement dans un navigateur. Il faut passer un site web qui propose ce service. Voici le principe [2] :

- Python est déjà installé sur un ordinateur appelé serveur, et on peut y accéder par un site web.
- Sur ce site web, on saisit notre code, ensuite on valide.
- Le code est exécuté par le serveur, qui retourne le résultat sous forme d'un texte qui s'affiche sur le site web.

Deux sites web assez complets [3] :

- https://repl.it/languages/python3
- https://trinket.io/features/python3

D'autres sites, au cas où les précédents seraient indisponibles :

- https://www.tutorialspoint.com/execute_python3_online.php
- http://rextester.com/l/python3_online_compiler
- http://www.pythontutor.com/ (Mode pas à pas intéressant)

1. Les exemples de la documentation EduPython utilisent souvent le module `lycee` qui est fourni avec EduPython mais que l'on peut utiliser sans EduPython. Les exemples sont à adapter pour éviter l'usage du module `lycee`, ou alors il faut utiliser le fichier `lycee.py`.

2. Attention à ne pas déposer sur ces sites des informations personnelles : tout peut y être conservé et éventuellement diffusé, partagé, etc.

3. En particulier on peut les utiliser avec les modules `matplotlib` et `turtle`. Voir dans ce document les sections concernant ces deux modules pour davantage d'informations.

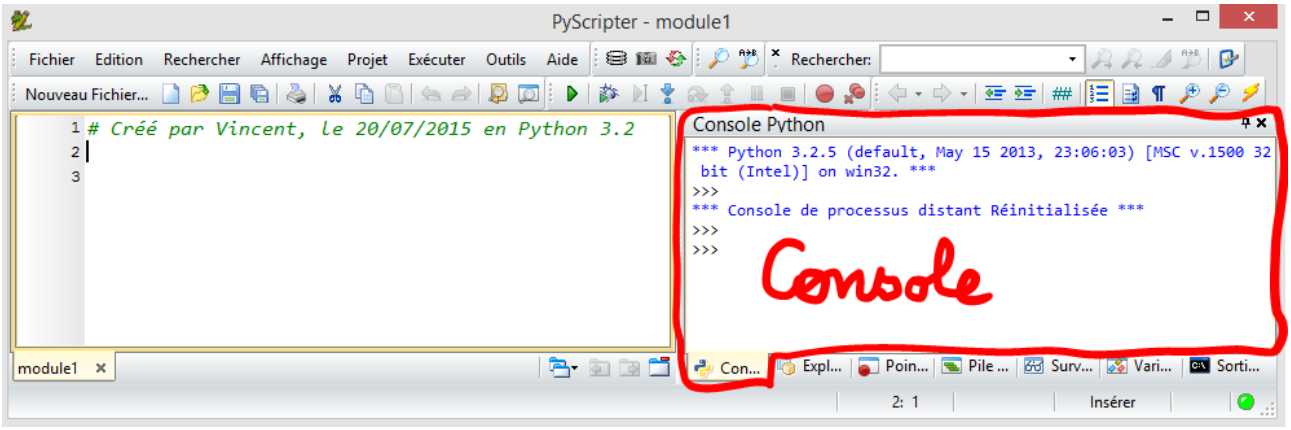
# 2 Utiliser la console et découvrir la syntaxe de Python

## 2.1 La console Python

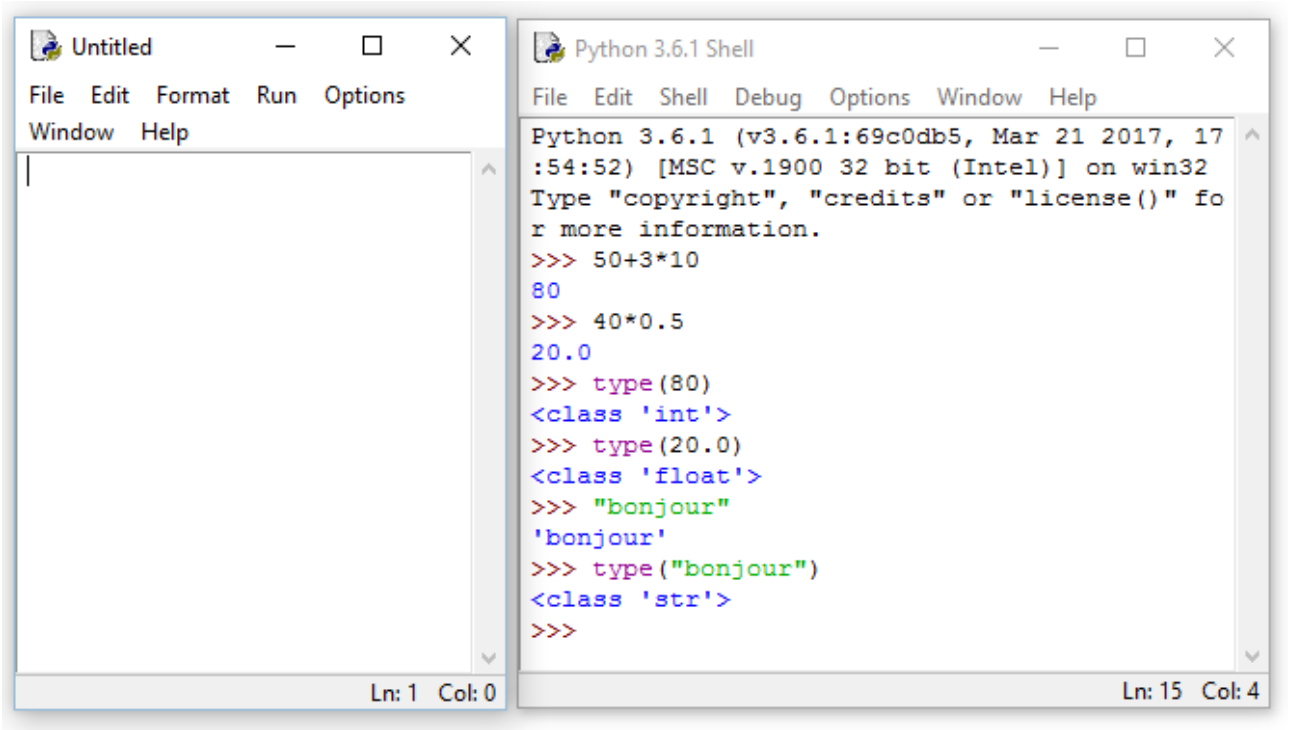
Dans un environnement Python, on dispose toujours d'une fenêtre appelée « console » que l'on peut utiliser pour faire des calculs, modifier des variables, évaluer des variables, exécuter des instructions. Dans la console, on exécute une instruction à la fois (ou un bloc à la fois) et un affichage se produit si cela a un sens.

C'est très pratique pour faire de petits essais et pour expérimenter.

On trouvera ci-dessous une capture d'écran de l'éditeur PyScripter qui est utilisé par la distribution EduPython. La console est à droite de l'écran.




Ci-dessous une capture d'écran de l'environnement de développement en Python appelé IDLE (pour Windows). La fenêtre de droite contient la console.



## 2.2 Python comme calculatrice

On peut directement entrer des calculs dans la console.

### 2.2.1 Les quatre opérations

On peut entrer directement des calculs. Les priorités opératoires sont respectées.

```
>>> 2+2
4
>>> 3+5*6
33
>>>(50-5*6)/4
5.0
>>> 8/5
1.6
```

### 2.2.2 Arithmétique

Les nombres entiers sont de type `int`, ceux avec une partie décimale sont de type `float` (les nombres flottants).
La commande `type()` donne le type d'une expression.

```
 >>> type(2)
<class 'int'>
>>> type(2.0)
<class 'float'>
```

La division donne toujours un résultat de type `float` mais on peut imposer un travail sur les entiers.

```
>>> 17/3
5.666666666666667
>>> 17//3 #On demande un quotient entier
5
>>> 17 % 3 #le reste de la division
2
```

### 2.2.3 Puissances

Les puissances d'un nombre s'écrivent avec **

```
>>>5**2
25
>>> 2**7
128
```

### 2.2.4 Les complexes

Python sait calculer avec les nombres complexes. Attention à la notation, le nombre écrit mathématiquement $2 + 3i$ sera noté `2+3j` dans Python.

```
>>> (2+1j)+(3+2j)
(5+3j)
>>> (2+1j)*(3+2j)
(4+7j)
>>>(2+1j)/(1+2j)
(0.8-0.6j)
```

## 2.3 Variables et affectation

Le signe = permet d'affecter une valeur à une variable. Il n'y a pas de déclaration préalable de variables en Python. Le type de variable est défini automatiquement suivant sa première affectation.

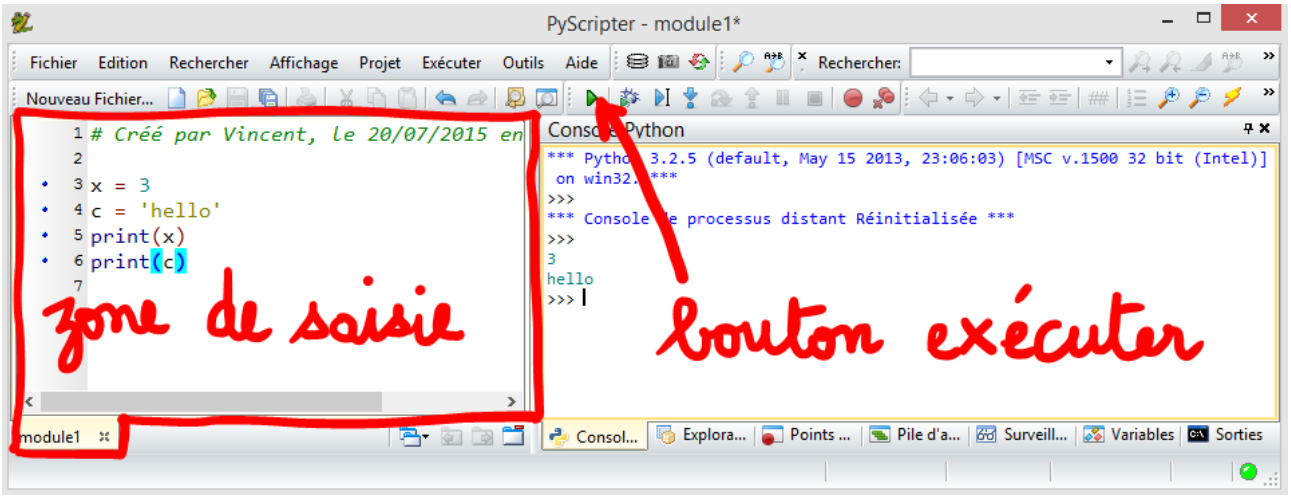
```
>>> a=5 # a est un entier
>>> b=7.2 # b est un nombre flottant
>>> a+b
12.2
>>> a*b
36.0
>>> 10+_ # le signe _ est comme le Rep de la calculatrice.
46.0
```

# 3 Écrire un algorithme dans l'éditeur de scripts Python

Lorsqu'on écrit directement les instructions dans la console, on utilise le mode interactif de Python. Chaque instruction est exécutée immédiatement après la saisie de la touche `Entrée`.

Il existe une autre façon de procéder. On peut écrire les instructions dans un fichier texte, dans une fenêtre appelée l'éditeur de script. Ensuite on demande à l'interpréteur d'exécuter toutes les instructions, les unes après les autres. L'avantage est qu'on peut sauvegarder les instructions dans le fichier que l'on peut réutiliser plus tard. Lorsqu'on a fait ceci, on dit qu'on a écrit un script Python.

Dans l'image ci-dessous (avec EduPython), un script a été saisi dans la fenêtre de gauche, il a été exécuté, et les résultats sont affichés dans la console à droite.




## 3.1 Saisir une variable, afficher

Pour écrire un algorithme et pouvoir l'exécuter, il faut sortir de la console. Pour cela, on crée un nouveau fichier. On écrira l'algorithme dans ce fichier puis on demandera l'exécution.
Attention : Le fichier doit être enregistré avant d'être exécuté.

### 3.1.1 Afficher une variable

Voici un algorithme qui affecte une valeur à une variable et qui l'affiche.

```
a=5
print(a)
```

L'affichage peut comporter un message aussi.

```
a=5
print("Le resultat est",a)
```

ou encore

```
a=5
print("Le nombre ",a," est la reponse.")
```

### 3.1.2 Saisir une valeur

L'instruction est `input()`, Dans les parenthèses, on écrit le message qui s'affiche.

```
a=input("Saisir a : ")
print ("a vaut ",a)
```

En fait cet algorithme cache un piège puisque la variable saisie n'est pas considérée comme un nombre par Python mais comme une chaine de caractères. On peut voir cela en modifiant l'algorithme.

```
a=input("Saisir a : ")
a=a+1
print ("a vaut ",a)
```

Cet algorithme donne une erreur !
Pour résoudre le problème, on convertit la chaine de caractères saisie en nombre entier ou flottant.

```
a=int(input("Saisir a : "))
a=a+1
print ("a vaut ",a)
```

ou

```
a=float(input("Saisir a : "))
a=a+1
print ("a vaut ",a)
```

Si on veut entrer une expression mathématique comme $\sqrt{3}$ comme valeur d'une variable, il faut s'assurer que Python comprend bien que la chaîne de caractères saisie au clavier est la représentation d'un nombre. Exemple :

```
from math import *
a=float(eval(input("Entrez a : ")))
print(a)
```

Dans la console, on verra :

```
Entrez a : sqrt(3)
1.7320508075688772
```

Si on n'utilise pas la fonction `eval()` qui évalue la chaine de caractères, le script produit une erreur.

**Idées d'exercices :**

1. Algorithme du volume d'une pyramide à base carrée (données : côté, hauteur).
2. Écrire un programme demandant à l'utilisateur son année de naissance et renvoyant son âge.
3. Un vendeur fait un devis pour un client souhaitant acheter deux articles en plusieurs exemplaires. Données : prix de l'article 1, quantité de l'article 1, prix de l'article 2, quantité de l'article 2. Sortie : prix total.
4. Des amis projettent un séjour d'une semaine à la montagne. La location de l'appartement coûte 600 euros et le forfait hebdomadaire pour les remontées mécaniques est de 200 euros par skieur. Ils ne savent pas encore s'ils partiront à 4, 5 ou 6 et désirent partager les frais. Écrire un programme qui demande le nombre de participants, affiche le coût total du séjour et la part revenant à chacun.
5. Algorithme sur la distance de deux points en repère orthonormé. Données : les coordonnées de points. Sortie : la distance.
6. Algorithme sur le milieu d'un segment dans un repère. Données : les coordonnées de points. Sortie : les coordonnées du milieu.

## 3.2 Instructions conditionnelles

Il s'agit de faire un test sur la valeur d'un variable et agir en conséquence. Les mots clés sont `if` et `else`. Attention !

- Il n'y a pas de mot `then`, il est sous-entendu.
- On décale d'une tabulation tout ce qui fait dans le « alors » et le « sinon ».
- Après chaque mot clé on trouve un « : »
- Un test d'égalité se fait avec deux signes égal : ==

```
a=int(input("Saisir a: "))
if a==5:
    print ("a vaut 5")
else:
    print("a est different de 5")
```

Le décalage avec des tabulations est primordial en Python. C'est une des difficultés majeures au début. Les deux algorithmes suivants sont différents. Testez-les !

```
a=int(input("Saisir a: "))
if a==0:
   print("a=0")
print("C'est gagne!")
```

```
a=int(input("Saisir a: "))
if a==0:
    print("a=0")
    print("C'est gagne!")
```

Ce décalage vaudra pour les boucles aussi.

**Idées d'exercices :**

1. Programme qui affiche Majeur/Mineur suivant l'âge saisi.
2. Programme qui affiche si un triangle est rectangle ou non selon les longueurs des trois côtés.
3. Programme qui affiche si un triangle est isocèle ou non selon les coordonnées des sommets (dans un repère orthonormé).
4. Programme qui affiche si un quadrilatère est un parallélogramme selon les coordonnées des sommets (dans un repère).
5. Programme qui affiche Admis/Oral/Recalé selon la moyenne à l'écrit de bac.
6. Programmer une fonction définie par intervalles.
7. Programme qui affiche le prix à payer pour des photocopies dans un magasin qui applique les tarifs suivants :
   - 0,15 euro l'unité pour 50 photocopies ou moins.
   - 0,10 euro l'unité pour 51 photocopies ou plus.

### 3.3 Les nombres aléatoires

Python n'a pas d'instruction naturelle pour créer un nombre aléatoire. Pour cela, comme pour beaucoup de fonctions avancées, il existe des bibliothèques qui permettent d'enrichir le langage.
La bibliothèque `random` contient trois fonctions qui peuvent être utiles dès la seconde mais en contient beaucoup d'autres.

- `randint(a,b)` qui donne un entier aléatoire $n$ tel que $a \leq n \leq b$
- `random()` qui donne un nombre flottant $x$ tel que $0 \leq x \leq 1$
- `uniform(a,b)` qui donne un nombre flottant $x$ tel que $a \leq x \leq b$

Il y a trois façons d'importer une bibliothèque, ces trois façons vont influencer la syntaxe du script.

1.
```
>>> import random
>>> random.randint(1,6)
5
```

Cette méthode est simple mais la syntaxe est longue.

2.
```
>>> from random import *
>>> randint(1,6)
4
```

Cette méthode raccourcit le script mais on peut oublier de quelle bibliothèque vient la fonction utilisée.

3.
```
>>> import random as rd
>>> rd.randint(1,6)
2
```

Cette dernière méthode est souvent utilisée, elle raccourcit la syntaxe tout en se rappelant que la fonction fait partie d'une bibliothèque.

**Idées d'exercices :**

1. Programme qui affiche Pile/Face au hasard.
2. Programme qui affiche Pierre/Feuille/Ciseaux au hasard.

### 3.4 La boucle Pour

1. Afficher les nombres de 1 à 10

**début**
  **pour** *i allant de 1 à 10* **faire**
    Afficher i
  **fin**
**fin**

```
for i in range(1,11):
    print(i)
```

`range(1,11)` contrairement à d'autres langages donne succesivement les nombre supérieurs ou égaux à 1 et strictement inférieurs à 11.
On peut arranger l'affichage en replaçant `print(i)` par `print(i, end="-")`

2. Afficher la table de 7

**début**
  **pour** *i allant de 1 à 8* **faire**
    P prend la valeur 7 × i
    Afficher P
  **fin**
  Afficher "C'est fini"
**fin**

```
for i in range(1,9):
    P=7*i
    print(P,end="  ")
print("C'est fini")
```

**Idées d'exercices :**

1. Un placement bancaire. On demande la valeur au bout d'un certain nombre d'année.
2. Faire la somme de tous les entiers compris entre deux entiers relatifs donnés.
3. Calculer $1+2+3+\cdots+n$ pour une valeur de $n$ saisie.
4. Calculer $1+2^2+3^2+\cdots+n^2$ pour une valeur de $n$ saisie.
5. Calculer $T_n = 1+\frac{1}{2^2}+\frac{1}{3^2}+\frac{1}{4^2}+\cdots+\frac{1}{n^2}$ pour une valeur de $n$ saisie. Calculer $\sqrt{6\times T_n}$. Que remarque-t-on ?
6. Programme qui affiche le nombre de diviseurs d'un entier $n$ saisi.
7. Programme qui simule 100 lancers d'une pièce de monnaie équilibrée et qui compte le nombre de Pile obtenus.

### 3.5 La boucle Tant que

1. Un algorithme qui donne le reste de la division par 7.

```
début
    Saisir U
    tant que u>=7 faire
        U prend la valeur U-7
    fin
    Afficher U
fin
```

```
u=int(input("u="))
while u>=7:
      u=u-7
print(u)
```

2. Recherche de la plus petite valeur de $n$ pour laquelle la somme des $n$ premiers entiers naturels dépasse 10 000

```
début
    n prend la valeur 0
    s prend la valeur 0
    tant que s<10000 faire
        n prend la valeur n+1
        s prend la valeur s+n
    fin
    Afficher n
fin
```

```
n,s=0,0 # On peut affecter deux variables d
    'un coup!
while s<10000:
      n+=1    #pareil que n=n+1
      s=s+n
print(n)
```

**Exercice 1**

Soit $(u_n)$ la suite définie par $u_{n+1} = 1.04u_n - 1$ avec $u_0 = 50$. Ecrire un programme qui détermine à partir de quel rang $u_n$ est supérieur ou égal à 5000.

**Exercice 2**

Jeu du lièvre et de la tortue. Un lièvre et une tortue font la course. On lance un dé cubique équilibré. Si le résultat est 1,2,3,4 ou 5 alors la tortue avant d'un pas. Si le résultat est 6 alors le lièvre gagne la course. La tortue gagne la course si elle arrive à avancer de 6 pas avant que le lièvre ne gagne.

1. Écrire un programme qui simule une partie et qui annonce le vainqueur.
2. Améliorer le programme pour qu'il simule 10000 parties et pour qu'il donne la fréquence de victoires de la tortue.

*Petite indication : `import random as rd` permettra d'utiliser l'instruction `rd.randint(1,6)` pour obtenir un lancer de dé.*

### 3.6 Les fonctions - Nouveauté en seconde 2017 ! !

On va introduire en seconde la fonction numérique dans les algorithmes. Prenons l'exemple de la fonction du second degré $f(x) = x^2 - x + 1$.
On souhaite compléter un tableau de valeurs et afficher les images des nombres de 1 à 10.

```
def f(x): #la variable peut porter le nom qu'on veut
    return x**2-x+1
```

```
for i in range(1,11):
    print(f(i))
```

Pour un affichage plus soigné, on peut remplacer `print(i)` par `print("f({})={}" .format(i,f(i)))`

La plupart des fonctions mathématiques sont absentes de Python mais on peut les retrouver avec le module `numpy` (et aussi avec le module `math`). La différence pratique essentielle est la capacité de `numpy` de faire les calculs sur des tableaux et sur des scalaires alors que le module `math` ne fera des calculs que sur les scalaires.

```
>>> import numpy as np
>>> np.sin(np.pi)
1.2246467991473532e-16
>>> x=[-np.pi,0,np.pi]
>>> np.cos(x)
array([-1.,  1., -1.])
```

**Idées d'exercices :**

1. Voir les nouveaux manuels de seconde qui proposent des exercices avec des fonctions en Python.
2. En fait, beaucoup d'exercices que nous donnons déjà peuvent être donnés sous la forme d'une fonction. Par exemple la fonction qui renvoie le volume d'une pyramide à base carrée, et dont les variables sont le côté du carré et la hauteur de la pyramide.

**Exercice 3**

Soit $f(x) = x^2 + 20x - 12$ définie sur $\mathbb{R}$. Rechercher par dichotomie avec une précision $10^{-4}$ la racine de $f$ qui se situe dans l'intervalle $[-15, 15]$.

**Remarque :**

Il existe une différence entre Python et Scratch au sujet des fonctions.

- Dans Python, nous venons de voir qu'une fonction retourne une valeur (ce qui n'est d'ailleurs pas forcément une obligation : une fonction peut ne rien retourner).
- Dans Scratch, il n'y a pas de fonction. À la place, il y a des blocs utilisateurs. Mais ces blocs ne retournent pas de valeur (on dit parfois en informatique que ce sont des *procédures*). On peut demander à un bloc utilisateur de modifier une *variable globale* (voir document ressource pages 10 et 11). On dit qu'on a utilisé un *effet de bord*. En faisant ceci, on arrive à faire comme si on pouvait utiliser des fonctions dans Scratch (mais cette démarche est moins satisfaisante du point de vue de l'apprentissage des concepts informatiques).

# 4 Types de variables non numériques

## 4.1 Les chaînes de caractères (aperçu rapide)

```
>>> chaine = "Bonjour"
>>> type(chaine)
<class 'str'>
>>> chaine[0]
'B'
>>> chaine[1]
'o'
>>> chaine[6]
'r'
>>> chaine[7]  # il y aura une erreur car les indices vont de 0 a 6...
Traceback (most recent call last):
```

```
  File "<pyshell#4>", line 1, in <module>
    chaine[7]
IndexError: string index out of range
>>> len(chaine) # instruction pour avoir la longueur d'une chaine de caracteres
7
>>> for k in range(len(chaine)):
    print(chaine[k], end='-')


B-o-n-j-o-u-r-
>>> for caractere in chaine:
    print(caractere, end='-')


B-o-n-j-o-u-r-
>>> chaine[0]='b' # un caractere d'une chaine de caracteres en Python ne peut pas
     etre modifie, donc on aura une erreur ici:
Traceback (most recent call last):
  File "<pyshell#15>", line 1, in <module>
    chaine[0]='b'
TypeError: 'str' object does not support item assignment
>>> chaine = "Bonjour"
>>> chaine = chaine + " a tous"
>>> chaine
'Bonjour a tous'
```

## 4.2 Les listes (aperçu rapide)

En Python, il n'y a pas de structure appelée *tableau*. À la place, il y a les *listes*.

```
>>> maListe = [45,'a', 21.3]
>>> maListe
[45, 'a', 21.3]
>>> type(maListe)
<class 'list'>
>>> maListe[0]
45
>>> maListe[1]
'a'
>>> maListe[2]
21.3
>>> maListe[3]
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    maListe[3]
IndexError: list index out of range
>>> maListe.append("bonjour")
>>> maListe
[45, 'a', 21.3, 'bonjour']
```

- On remarque que ces listes peuvent contenir des données de types différents. C'est ce qui limite leur utilité pour les opérations mathématiques.
- La méthode **.append()** ne fonctionne que sur une liste qui est déjà existante. Dans un algorithme, on peut déclarer une liste vide avec `maliste=[]`

**Exercice 4**

Ecrire un programme qui donne un échantillon de 10 notes de musique choisies au hasard.
*On pourra définir une liste : notes=['do','re','mi','fa','sol','la','si']*

On peut créer une liste à l'aide d'une boucle :

```
>>> maliste=[x for x in range(5)]
>>> maliste
[0, 1, 2, 3, 4]
>>> maliste=[x for x in range(0,30,5)]
>>> maliste
[0, 5, 10, 15, 20, 25]
```

Cette méthode ne permet pas d'obtenir facilement des nombres décimaux puisque le pas de `range()` doit être un entier. On ne peut pas facilement faire d'opérations sur ces listes.

```
>>> maliste=[x for x in range(5)]
>>> maliste
[0, 1, 2, 3, 4]
>>> maliste**2
Traceback (most recent call last):
  File "<pyshell#22>", line 1, in <module>
    maliste**2
TypeError: unsupported operand type(s) for ** or pow(): 'list' and 'int'
>>> maliste=[x for x in range(0,2,0.5)]
Traceback (most recent call last):
  File "<pyshell#23>", line 1, in <module>
    maliste=[x for x in range(0,2,0.5)]
TypeError: 'float' object cannot be interpreted as an integer
```

Pour contourner ces difficultés, il existe une bibliothèque `numpy` qui permet ces manipulations et qui permettra de construire des tableaux. Un tel tableau (*array*) est une sorte de liste de nombres. On peut faire des calculs mathématiques directement sur le tableau numpy et python comprendra qu'il faut faire le calcul sur tous les éléments. C'est pratique pour obtenir un tableau de valeurs d'une fonction par exemple.

```
>>> import numpy as np
>>> maliste=np.linspace(0,1,num=11)
>>> maliste
array([ 0. ,  0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9,  1. ])
>>> maliste**2
array([ 0.  ,  0.01,  0.04,  0.09,  0.16,  0.25,  0.36,  0.49,  0.64, 0.81,  1.
    ])
>>> maliste=np.linspace(-5,5,6)
>>> maliste
array([-5., -3., -1.,  1.,  3.,  5.])
>>> maliste*3
array([-15.,  -9.,  -3.,   3.,   9.,  15.])
```

L'argument `num=11` donne le nombre d'éléments du tableau Les deux premiers arguments sont le premier et dernier nombre de la liste. Pour un écart de 0.1 entre les éléments, il faut donc un tableau de 11 éléments.
Intuitivement, l'écart étant de 0.1, on pourrait penser qu'il faut 10 éléménts. Pour cela, il faut exclure le dernier nombre de la liste :

```
>>> maliste=np.linspace(0,1,10,endpoint=False)
>>> maliste
array([ 0. ,  0.1,  0.2,  0.3,  0.4,  0.5,  0.6,  0.7,  0.8,  0.9])
```

On peut calculer la somme, le produit ou la moyenne des éléments de la liste :

```
>>> maliste=np.linspace(1,11,10,endpoint=False)
>>> maliste
array([  1.,   2.,   3.,   4.,   5.,   6.,   7.,   8.,   9.,  10.])
>>> maliste=np.linspace(1,10,10)
```

```
>>> maliste.sum()
55.0
>>> maliste.prod()
3628800.0
>>> maliste.mean()
5.5
```

**Exercice 5**

Soit $f(x) = \dfrac{x}{x^2+1}$

Ecrire un algorithme qui donne le tableau de valeurs de $f$ sur l'intervalle [a,b] avec un pas de p où l'utilisateur choisit a,b et p.

# 5 France IOI (site web) : apprentissage et entraînement à l'algorithmique.

## 5.1 Présentation

France IOI est un site internet d'entraînement à la programmation et à l'algorithmique. Il est géré par une association agréée par le ministère de l'éducation nationale.

Voici les particularités de ce site :

- On peut l'utiliser pour apprendre à programmer. Une progression des apprentissages est proposée, avec des cours et des exercices à réaliser régulièrement. Ces exercices sont vérifiés de façon automatique par le site internet.
- Il y a aussi un système pour les professeurs, pour organiser le suivi d'élèves via un système de classes.
- Pas besoin d'installer de logiciel, tout se fait sur le site.
- Le langage Python est disponible.
- On peut suivre ses progrès et se positionner par rapport aux autres utilisateurs.
- Il y a un système d'entraide.

Après un essai de la quasi totalité du « Parcours Lycée », on peut dire que c'est vraiment accessible au début, et tout à fait progressif. Un utilisateur qui sait déjà programmer a intérêt à passer le début car il y a vraiment beaucoup d'exercices faciles au début du niveau 1. C'est un site qui prend le temps de bien poser les choses, donc parfois on a l'impression de ne pas avancer très vite. Cela peut constituer un inconvénient.

En fait, IOI signifie Olympiades Internationales d'Informatique. Ce site permet aussi permet d'entraîner des lycéens à ces Olympiades (le niveau des problèmes les plus difficiles est donc élevé...)

L'association France IOI participe également à l'organisation du concours Castor Informatique, qui vise à faire découvrir aux jeunes collégiens et lycéens l'informatique et les sciences du numérique.

## 5.2 Objectifs

On cherche ici à comprendre le fonctionnement du site France IOI. On se demandera si on peut envisager d'utiliser ce site avec tous les élèves, certains élèves en particulier, et de quelle façon.

## 5.3 Envisager une utilisation avec les élèves (en classe ou en dehors)

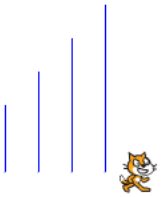
Lien vers le site `http://www.france-ioi.org/`

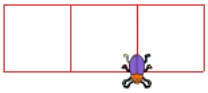
1. Créer un compte sur le site.
2. Rejoindre le groupe `Formation-Python-13-juin-2017` Le mot de passe est `formationpython2017math` (sans espaces).
3. Essayer quelques exercices de niveau 1 :
   - Dans la partie `1 - Affichage de texte, suite d'instructions`
     → exercice 4) Dans le fourré (déplacer le robot sur une grille avec des obstacles)
   - Dans la partie `2 - Répétitions d'instructions`
     → exercice 9) Vendanges (faire faire une tâche au robot avec des répétitions imbriquées)
   - Dans la partie `3 - Calculs et découverte des variables`
     → exercice 11) Course avec les enfants (utiliser un compteur pour le nombre d'itérations d'une boucle interne)
   - Dans la partie `3 - Calculs et découverte des variables`
     → exercice 12) Construction d'une pyramide (sommer les cubes des entiers impairs de 1 à 17)
4. Voir avec le vidéoprojecteur (dans le menu du groupe) les exercices essayés et résolus par chacun.

# 6 Programmer un dessin à l'écran

## 6.1 Ce qui a déjà été fait au collège avec Scratch

Il s'agit ici de la programmation d'un dessin à l'écran, comme dans le document d'accompagnement.





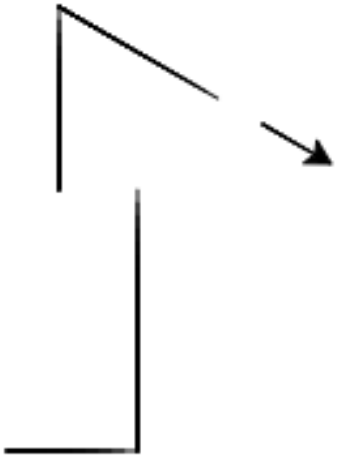
Le lutin chat trace 4 segments espacés de 25 pixels. Le premier segment a une longueur de 50 pixels. Chaque segment mesure 25 pixels de plus que le précédent.

Le lutin scarabée trace 3 carrés de 50 pixels de côté.

## 6.2 Le module turtle de Python

```
import turtle as t

# 1ere ligne
t.forward(50)
t.left(90)
t.forward(100)

# 2e ligne
t.penup()
t.left(90)
t.forward(30)
t.pendown()
t.right(90)
t.forward(70)
t.right(120)
t.forward(70)

# 3e ligne
t.penup()
t.forward(20)
t.pendown()
t.forward(30)
```

Il existe aussi un module « tortue » avec Python. Le script ci-contre permet de créer l'image ci-dessous.



Exemples de logiciels ou de sites web avec lesquels le script précédent a été testé et fonctionne :

- avec l'environnement IDLE.
- https://repl.it/DoYJ/222 (Remarque : on peut choisir « continue as anonymous »)
  Et plus généralement https://repl.it/languages/python_turtle
- https://trinket.io/python
- En dépannage, on pourra utiliser http://interactivepython.org/runestone/static/thinkcspy/PythonTurtle/SummaryofTurtleMethods.html

## 6.3 Utilisations en classe : idées de programmes à demander aux élèves

**Exercice 6**

Construire un hexagone de 100 pixels de côté.

**Exercice 7**

Construire un pentagone de 100 pixels de côté.

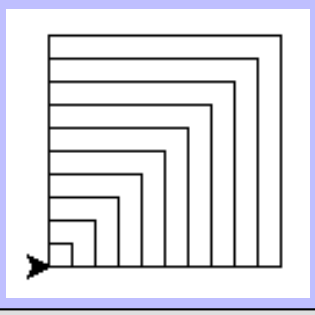
## Exercice 8

Tracer les 4 segments du « lutin chat » et les 3 carrés du « lutin scarabée » dont il était question précédemment.

On pourra utiliser l'instruction du type `t.goto(0, 140)` entre les deux constructions (si on veut construire les deux figures dans le même scripte). Cette instruction déplace la tortue au pixel de coordonnes `(0,140)`.

## Exercice 9

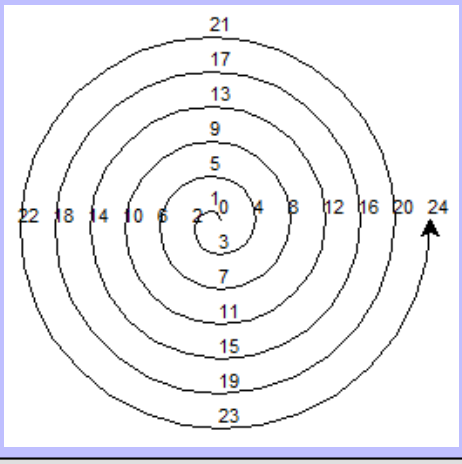
Écrire un programme qui affiche des carrés imbriqués comme dans la figure ci-dessous.



## Exercice 10

Écrire un programme qui affiche une spirale formée de quarts de cercles de même centre. On pourra numéroter les quarts de cercle à l'écran.

Aide : après `import turtle as tortue`, l'instruction `tortue.circle(rayon, angle)` trace un arc de cercle de rayon et d'angle donné à partir de la position actuelle de la tortue. Par ailleurs, l'instruction `tortue.write(v)` affiche le contenu de la variable `v`.

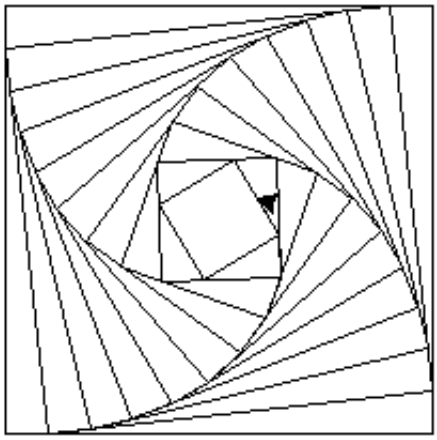



## Exercice 11

**Notion de limite de suites en première S**

ABCD est un carré de côté 10cm ; Chaque nouveau carré est construit en déplaçant chaque sommet de 1cm. On veut conjecturer la limite de la suite des aires. Ecrire un programme qui donne le dessin ci-dessous avec la suite des aires.

*On peut utiliser t.write(n) pour écrire le contenu de la variable n à la position de la tortue et p=t.pos() pour placer dans la variable p la position actuelle de la tortue pour pouvoir y revenir*

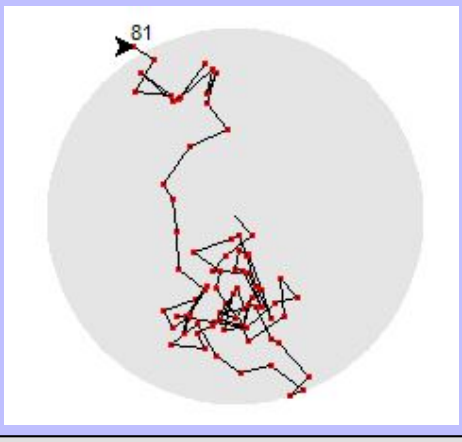


100
82.0
65.8892297237
51.6547935187
39.2805396572
28.7457018589
20.0226982016
13.0733522586
7.84193406893
4.24124337262

## Exercice 12

Une marche aléatoire.

On trace un disque de rayon 200 pixels. On commence au centre. On se déplace successivement d'un nombre de pixels aléatoire entre $-20$ et 20 horizontalement, et aussi verticalement. Si on sort du disque, on s'arrête et on affiche le nombre d'étapes effectuées.

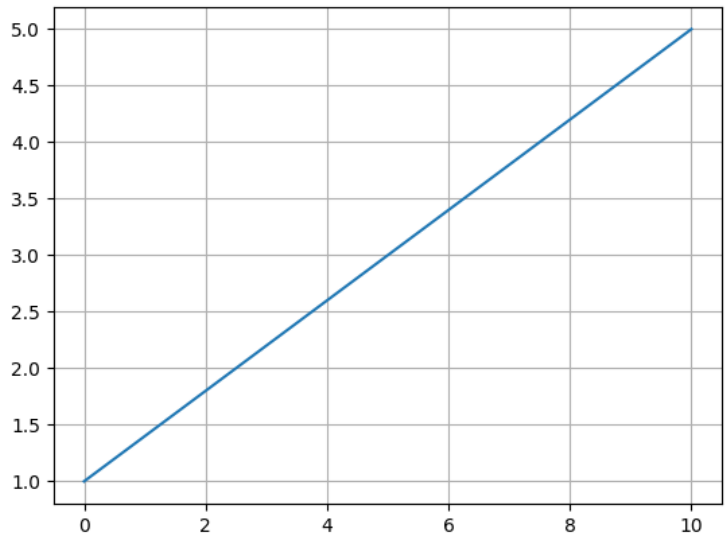


# 7 Les graphiques en Python : le module matplotlib

## 7.1 Avec une version de Python installée, ou avec un site web

Le plus simple est d'utiliser une version installée de Python, par exemple avec l'environnement IDLE.

```
# Avec une version de Python installee
import matplotlib.pyplot as plt
# debut de nos instructions
plt.plot([0,10], [1,5])
plt.grid()
# fin de nos instructions
plt.show()  # ouverture de fenetre avec le graphique
```




Mais on peut aussi utiliser un site web sur lequel le module `matplotlib` est disponible. Par exemple :

- https://repl.it/languages/python3 (pour utiliser `matplotlib`, cliquer à gauche sur l'icône « add new file »)
- https://trinket.io/features/python3

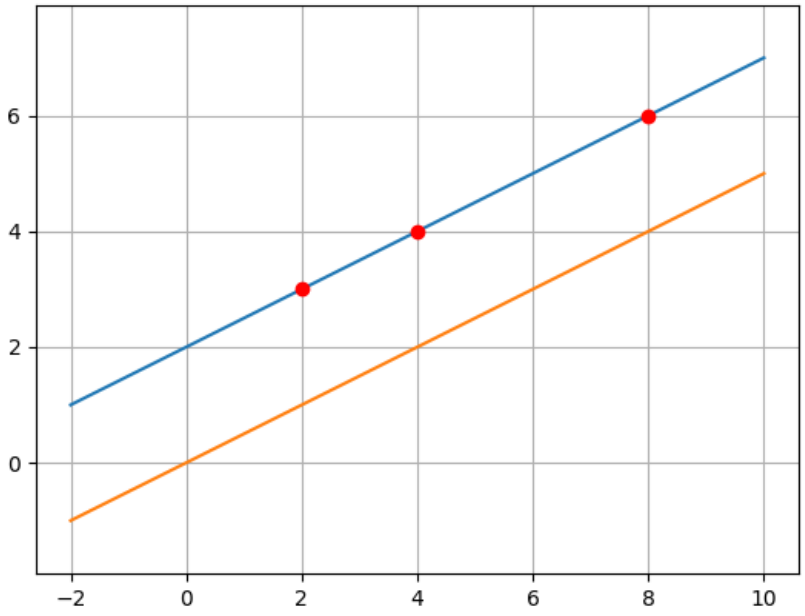
Dans le cas d'un site web, python n'ouvre pas de nouvelle fenêtre pour afficher le graphique. À la place, python écrira dans un fichier image, qu'il faudra ensuite visualiser dans le navigateur. Cela explique qu'il faut des instructions supplémentaires comme ci-dessous.

```
# En utilisant Python sur un site web
import matplotlib as mpl
mpl.use('Agg')
import matplotlib.pyplot as plt
fig = plt.figure()
plt.grid()
ax = fig.add_subplot(111)
# debut de nos instructions
plt.plot([0,10], [1,5])
# fin de nos instructions
fig.savefig('graph.png') # ecriture du graphique dans un fichier image
```

## 7.2 Segments, points (exemple, exercices)

Exemple :

```
# Avec une version de Python installee
import matplotlib.pyplot as plt
# segments :
plt.plot([-2,10], [1,7])
plt.plot([-2,10], [-1,5])
# nuage de points :
plt.plot([2,4,8], [3,4,6], 'ro')
# pour un repere orthonorme :
plt.axis('equal')
plt.grid()

plt.show()
```




**Exercice 13**

Soit $u$ la suite définie par $u_n = \dfrac{20}{n^2 - 14n + 51}$ pour tout entier naturel $n$.

1. Représenter graphiquement les termes de la suite.
2. Un élève affirme : « La suite $u$ est croissante ». Cet élève a-t-il raison ? Pourquoi ?

**Exercice 14**

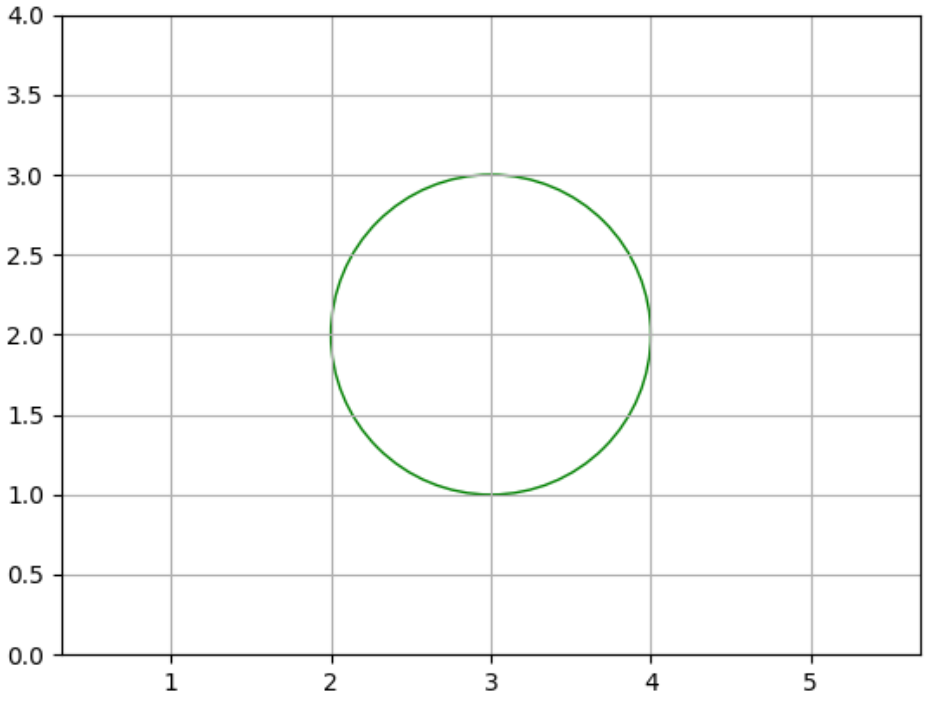
Soit $u$ la suite définie par $u_0 = 0,652$ et $u_{n+1} = 3,5u_n(1 - u_n)$ pour tout entier naturel $n$.
Représenter graphiquement les 100 premiers termes de la suite.

### 7.3 Exemple : cercle

On peut tracer un cercle avec `matplotlib`, mais cela demande un certain nombre d'instructions, comme le montre l'exemple ci-dessous. Cela rend les cercles moins pratiques pour une utilisation en classe.

```
# Avec une version de Python installee
import matplotlib.pyplot as plt

# important pour avoir un repere orthonorme:
plt.axis('equal')
# valeurs indicatives pour les axes:
plt.axis([0, 6, 0, 4])

plt.grid()  # pour afficher le quadrillage

# pour afficher un cercle
fig=plt.figure(1)
ax=fig.add_subplot(1,1,1)
cercle = plt.Circle((3,2), radius=1, color='g', fill=False)
ax.add_patch(cercle)

plt.show()
```

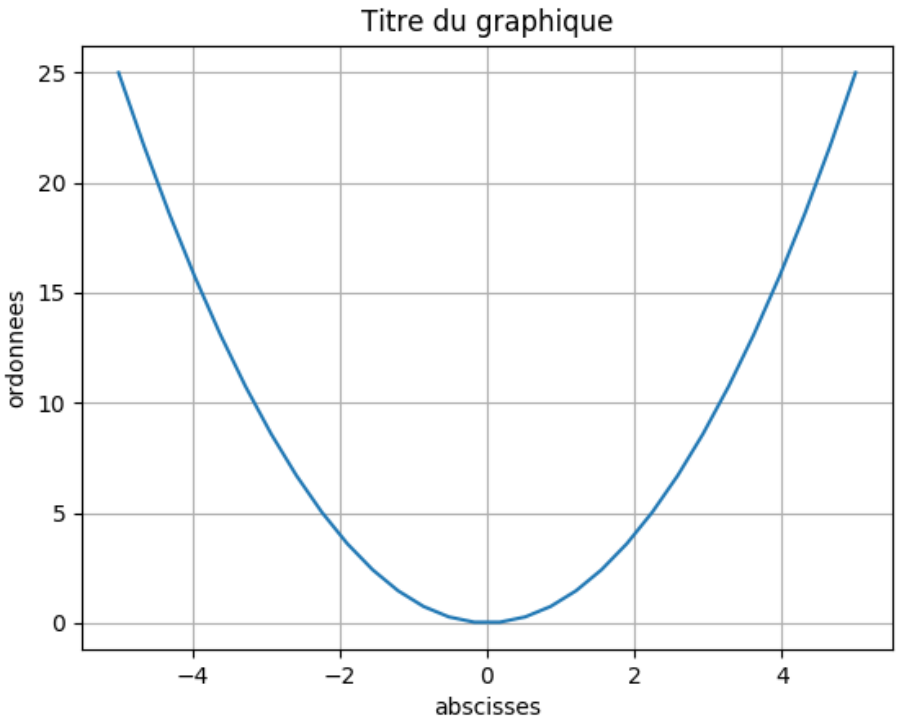
Voir à la fin du document un code plus court (et peut-être plus agréable) pour afficher un cercle.



## 7.4 Courbe représentative de fonction (exemples, exercice)

Premier exemple. On remarque que les axes habituels ne sont pas représentés ici.

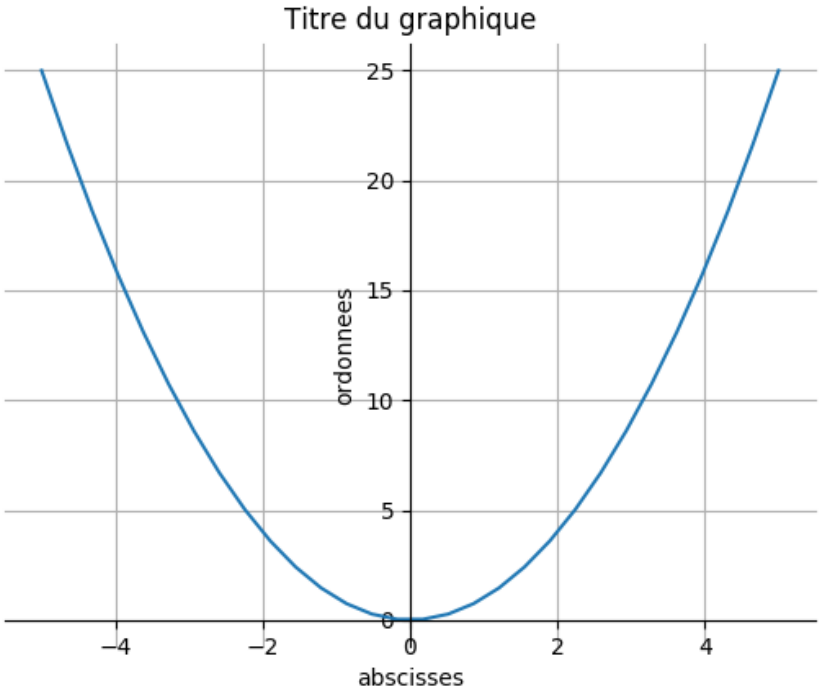
```
# Avec une version de Python installee
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-5, 5, 30)
print(x)
y = x**2
plt.plot(x, y)
plt.grid()
plt.title("Titre du graphique")
plt.xlabel("abscisses")
plt.ylabel("ordonnees")

plt.show()
```




Deuxième exemple, avec les axes cette fois.

```
# Avec une version de Python installee
import numpy as np
import matplotlib.pyplot as plt

x = np.linspace(-5, 5, 30)
print(x)
y = x**2
plt.plot(x, y)
plt.title("Titre du graphique")
plt.xlabel("abscisses")
plt.ylabel("ordonnees")
plt.grid()

ax = plt.gca()
```

```
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data',0))
ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data',0))

plt.show()
```




**Exercice 15**

Tracer la même courbe point par point sans utiliser de tableau. On utilisera une variable $x$ qui parcourt le domaine de définition $[-5;5]$.

**Exercice 16**

Pour tout entier $k$ compris entre $-5$ et 5, on note $f_k$ la fonction définie sur l'intervalle $[0;5]$ par

$$f_k(x) = k(x-2)+3.$$

1. Tracer avec Python la représentation graphique de toutes les fonctions $f_k$. On pourra utiliser la fenêtre graphique donnée par `plt.axis([0, 5, -15, 20])`
2. Quelle conjecture peut-on formuler concernant ces représentations graphiques ?
3. Justifier que la conjecture faite à la question précédente est vraie.

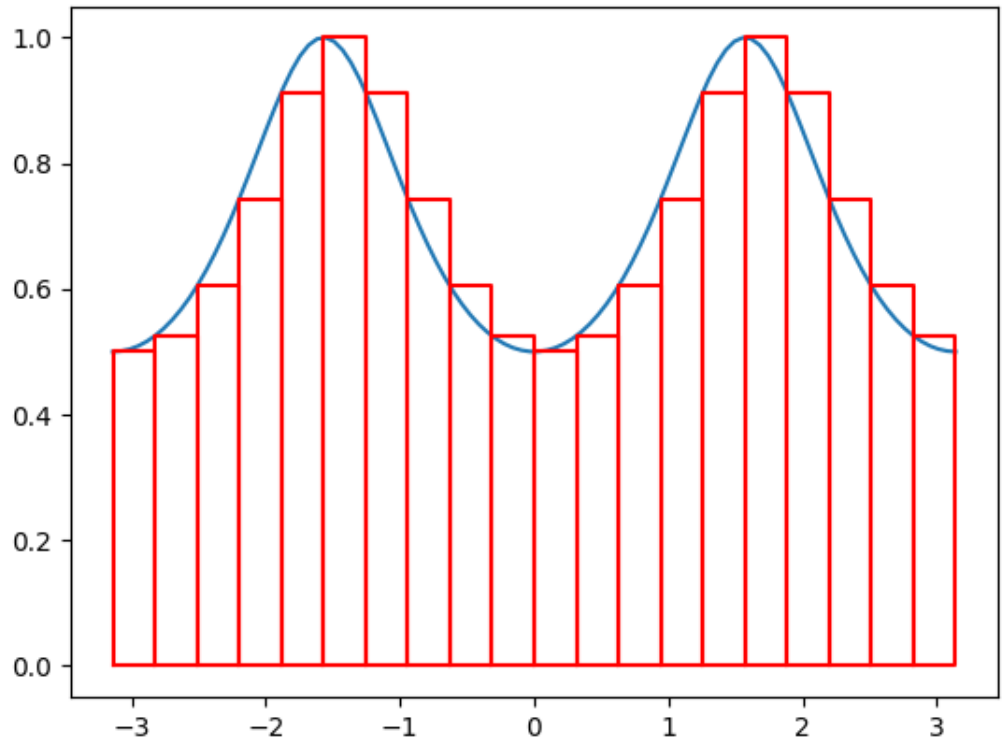
## Exercice 17

**Intégrale par méthode probabiliste**

Soit $f$ la fonction définie sur $[0,1]$ par $f(x) = \cos(\sqrt{x}) \times \mathrm{e}^{-x}$. On cherche une valeur approchée de $I = \displaystyle\int_0^1 f(x)\mathrm{d}x$. Pour cela, on crée $n$ points dont les coordonnées suivent des lois uniformes sur [0,1] et on compte le nombre de points sous la courbe. Ecrire un programme qui calcule une valeur approchée de I pour $n = 1000$ et qui place les points aléatoires dans un graphique avec la courbe de $f$.

## 7.5 Exemple : Méthode des rectangles

```
# Avec une version de Python installee
import numpy as np
import matplotlib.pyplot as plt
xmin = -np.pi
xmax = np.pi
nbr = 20 #nbre de rectangles

def f(a):
    return 1/(np.cos(a)**2+1)

xf=np.linspace(xmin,xmax,100)
yf=f(xf)

x=np.linspace(xmin,xmax,nbr+1)
y = f(x)
plt.plot(xf,yf,"-")
integrale = 0
for i in range(nbr):
    integrale = integrale + y[i]*(x[i+1]-x[i])   # dessin du rectangle
    x_rect = [x[i], x[i], x[i+1], x[i+1], x[i]] # abscisses des sommets
    y_rect = [0   , y[i], y[i]  , 0     , 0   ] # ordonnees des sommets
    plt.plot(x_rect, y_rect,"r")
print("integrale =", integrale)
plt.show()
```

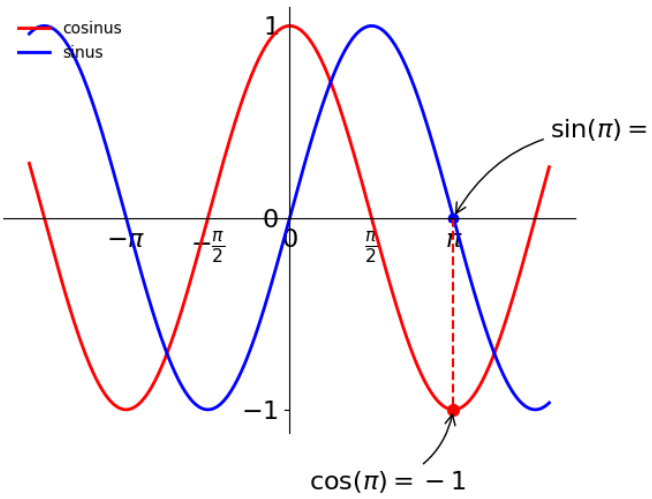



**Exercice 18**

Modifier le programme pour obtenir la méthode des trapèzes.

## 7.6 Exemple : Tracer une fonction : modifier le graphique

```
# Avec une version de Python installee
import matplotlib.pyplot as plt
import numpy as np

def f(t):
    return np.cos(t)

def g(t):
    return np.sin(t)

x=np.linspace(-5,5,500,endpoint=False)
lines1=plt.plot(x,f(x),label="cosinus")
lines2=plt.plot(x,g(x),label="sinus")

plt.setp(lines1,color='r',linewidth=2.0)
plt.setp(lines2,color='b',linewidth=2.0)

plt.legend(loc='upper left', frameon=False)

#les axes sont centres
ax=plt.gca()
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data',0))
ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data',0))

# On remplace les graduations sur l'axe des ordonnees
plt.yticks([-1,0,1],[r'$-1$',r'$0$',r'$1$'])

# On remplace les graduations sur l'axe des abscisses
plt.xticks([-np.pi,-np.pi/2,0,np.pi/2,np.pi],[r'$-\pi$',r'$-\frac{\pi}{2}$',r'$0$',r'$\frac
    {\pi}{2}$',r'$\pi$'])

t=np.pi
# une fleche pour le sinus de pi
plt.scatter([t],[g(t)],color='blue')
plt.annotate(r'$\sin(\pi)=0$',xy=(t,g(t)),xycoords='data',xytext=(60,50),textcoords='offset
    points',fontsize=16,arrowprops=dict(arrowstyle='->',connectionstyle="arc3,rad=.3"))

# les pointilles et le texte au bout de la fleche pour le cosinus
plt.scatter([t],[f(t)],50,color='red')
plt.plot([t,t],[0,f(t)],'r--')
plt.annotate(r'$\cos(\pi)=-1$',xy=(t,f(t)),xycoords='data',xytext=(-90,-50),textcoords='
    offset points',fontsize=16,arrowprops=dict(arrowstyle='->',connectionstyle="arc3,rad=.3"
    ))
for label in ax.get_xticklabels()+ax.get_yticklabels():
    label.set_fontsize(16)
    label.set_bbox(dict(facecolor='white',edgecolor='None',alpha=0.65))
plt.show()
```

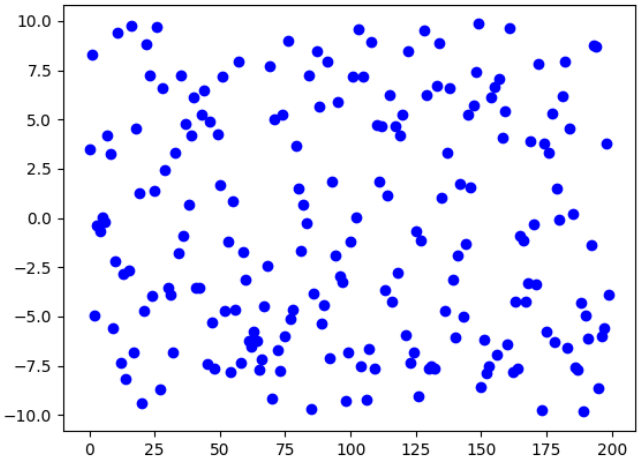



## 7.7 Points aléatoires (exemple, exercice)

```
# Avec une version de Python installee
import matplotlib.pyplot as plt
import random
for i in range(200):
    plt.plot(i,random.uniform(-10,10),'bo')
plt.show()
```

https://repl.it/EwS1/331

```
# En utilisant Python sur un site web
import matplotlib as mpl
mpl.use('Agg')
import matplotlib.pyplot as plt
fig = plt.figure()
ax = fig.add_subplot(111)

import random
for i in range(200):
    plt.plot(i,random.uniform(-10,10),'bo')

fig.savefig('graph.png')
```



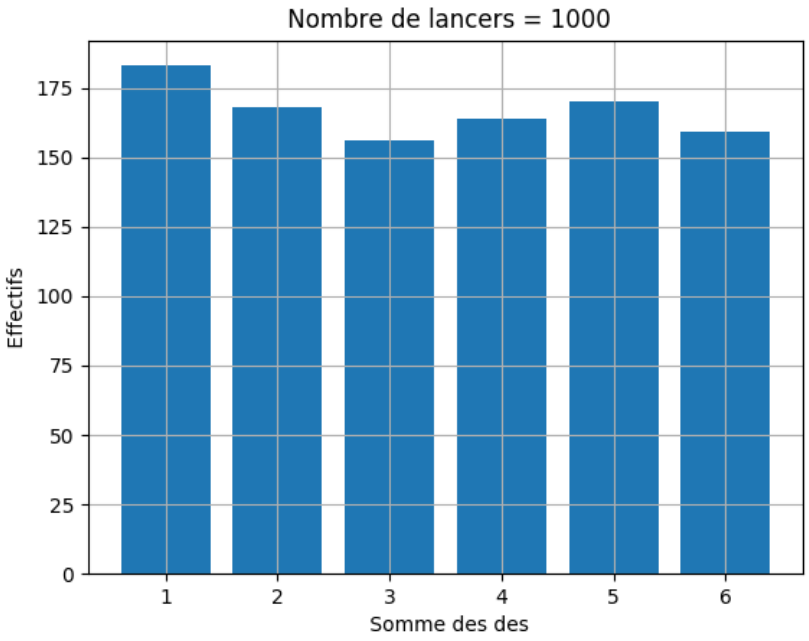
**Exercice 19**

Générer 2000 points d'abscisses aléatoires comprises en $-10$ et 10 et d'ordonnées aléatoires comprises en $-10$ et 10. Pour chaque point, s'il appartient au disque de centre $(0;0)$ et de rayon 5, le point s'affiche en bleu, sinon il s'affiche en vert.

## 7.8 Simulation de lancers de deux dés et diagramme en bâtons (exemple, exercice)

Exemple :

```
# Avec une version de Python installee
import matplotlib.pyplot as plt
import random
plt.grid ( True )
plt.xlabel (" Somme des des")
plt.ylabel (" Effectifs ")
effectif = 0
nb_tirages = 1000
plt.title (" Nombre de lancers = "+str( nb_tirages ))
liste_valeurs   = [1, 2, 3, 4, 5, 6]
liste_resultats = [0, 0, 0, 0, 0, 0]
for i in range (nb_tirages):
    de = random.randint (1, 6)
    liste_resultats [de-1] += 1
plt.bar( liste_valeurs , liste_resultats )
plt.show ()
```




**Exercice 20**

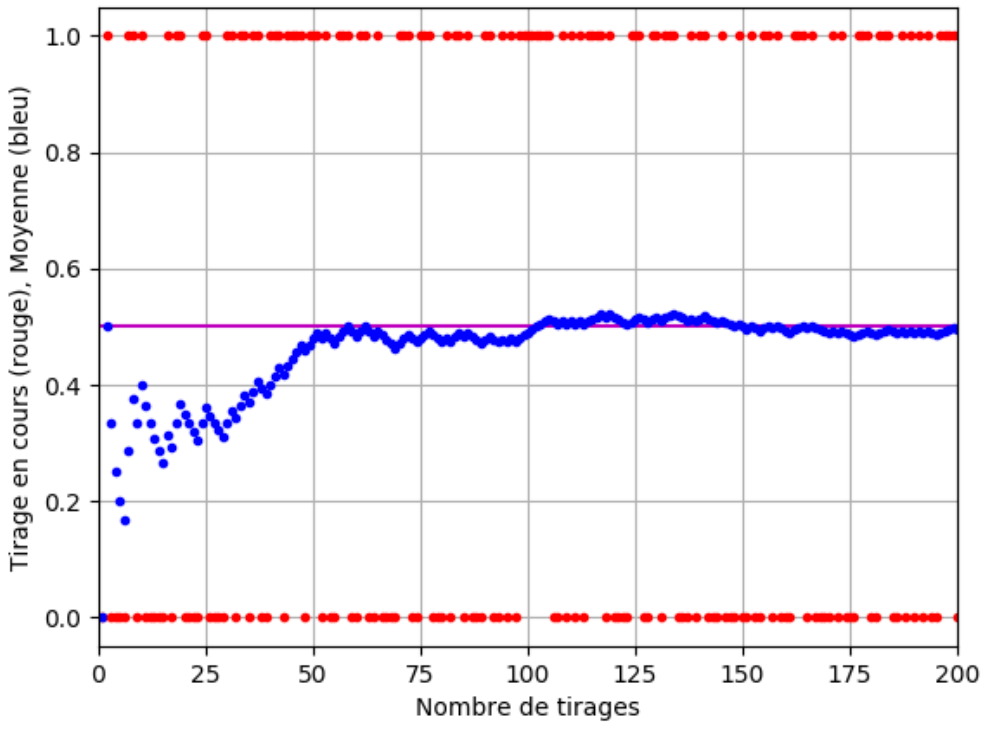
Simuler la somme de deux dés (1000 expériences) et afficher le diagramme en barres.

## 7.9 Évolution de la fréquence de pile dans un jeu de pile ou face (exercice : compléter l'algorithme)

### Exercice 21

Compléter les 4 lignes de l'algorithme suivant afin d'obtenir le graphique ci-dessous.

```
# Avec une version de Python installee
import matplotlib.pyplot as plt
import random as rd
nb_tirages=200
somme = 0
moyenne = 0
plt.axis([0,nb_tirages,-0.05,1.05])
plt.grid(True)
plt.xlabel('Nombre de tirages')
plt.ylabel('Tirage en cours (rouge), Moyenne (bleu)' )
plt.plot([0,nb_tirages],[0.5,0.5],'m-')
# a completer
    # a completer
    # a completer
    # a completer
    plt.plot(i,tirage,'r.')
    plt.plot(i,moyenne,'b.')
plt.show()
```



# 8 Pour les élèves les plus à l'aise

## 8.1 Project Euler (site web) : Défis mathématiques en lien avec les algorithmes

https://projecteuler.net/

- Problem 1 – Multiples of 3 and 5 – *résolu 660 000 fois.*
- Problem 2 – Even Fibonacci numbers – *résolu 534 000 fois.*
- Problem 9 – Special Pythagorean triplet – *résolu 260 000 fois.*
- Problem 14 – Longest Collatz sequence – *résolu 167 000 fois.*
- Problem 85 – Counting rectangles – *résolu 18 000 fois.*

## 8.2 Codingame (site web) : apprentissage de la programmation

https://www.codingame.com/home

Quelques problème dans lesquels on a besoin de mathématiques :
Entraînement, Puzzles Classiques - Facile :

- Onboarding (Conditions)
- Power of Thor - Episode 1 (Conditions)
- La descente (Conditions, extremum dans une liste de valeurs)

# 9 Le calcul formel en Python : le module sympy

Python peut développer, factoriser, simplifier des expressions (y compris trigonométriques) :

```
>>>from sympy import *
>>>x,y,z=symbols('x y z')
>>>factor(x**2+2*x+1)
(x + 1)**2
>>>simplify((x**3+x**2-x-1)/(x**2+2*x+1))
x-1
>>>expand((x+1)**3)
x**3 + 3*x**2 + 3*x + 1
>>>cancel(1/x+(3*x/2-2)/(x-4))
(3*x**2 - 2*x - 8)/(2*x**2 - 8*x)
>>>simplify(1/x+(3*x/2-2)/(x-4))
(3*x**2/2 - x - 4)/(x*(x - 4))
>>>trigsimp(sin(x)**4-2*cos(x)**2*sin(x)**2+cos(x)**4)
cos(4*x)/2 + 1/2
>>>expand_trig(sin(x+y))
sin(x)*cos(y) + sin(y)*cos(x)
```

Remarque : pour les puissances, python utilise ** et non pas le symbole ^

Avec l'instruction `init_printing()`, l'affichage est différent, cela peut être intéressant suivant les besoins :

```
>>> from sympy import *
>>> init_printing()
>>> x,y,z=symbols('x y z')
>>> factor(x**2+2*x+1)
       2
(x + 1)
>>> simplify((x**3+x**2-x-1)/(x**2+2*x+1))
x - 1
>>> expand((x+1)**3)
 3      2
x  + 3*x  + 3*x + 1
>>> cancel(1/x+(3*x/2-2)/(x-4))
   2
3*x  - 2*x - 8
--------------
     2
  2*x  - 8*x
>>> simplify(1/x+(3*x/2-2)/(x-4))
   2
3*x
---- - x - 4
 2
------------
 x*(x - 4)
>>> trigsimp(sin(x)**4-2*cos(x)**2*sin(x)**2+cos(x)**4)
cos(4*x)   1
-------- + -
   2       2
>>> expand_trig(sin(x+y))
sin(x)*cos(y) + sin(y)*cos(x)
```

Python peut résoudre des équations, calculer des dérivées, calculer des primitives et des intégrales :

```
>>> from sympy import *
>>> x=symbols('x')
>>> solve(x**2-x-1)
[1/2 + sqrt(5)/2, -sqrt(5)/2 + 1/2]
>>> diff(cos(x),x)
-sin(x)
>>> diff(3*x**2-5*x+7,x)
6*x - 5
>>> integrate(9*x**2+8*x+10,x)
3*x**3 + 4*x**2 + 10*x
>>> integrate(9*x**2+8*x+10,(x,0,1))
17
```

On peut aussi essayer `sympy` sur http://live.sympy.org/.

# 10 Jupyter (application web)

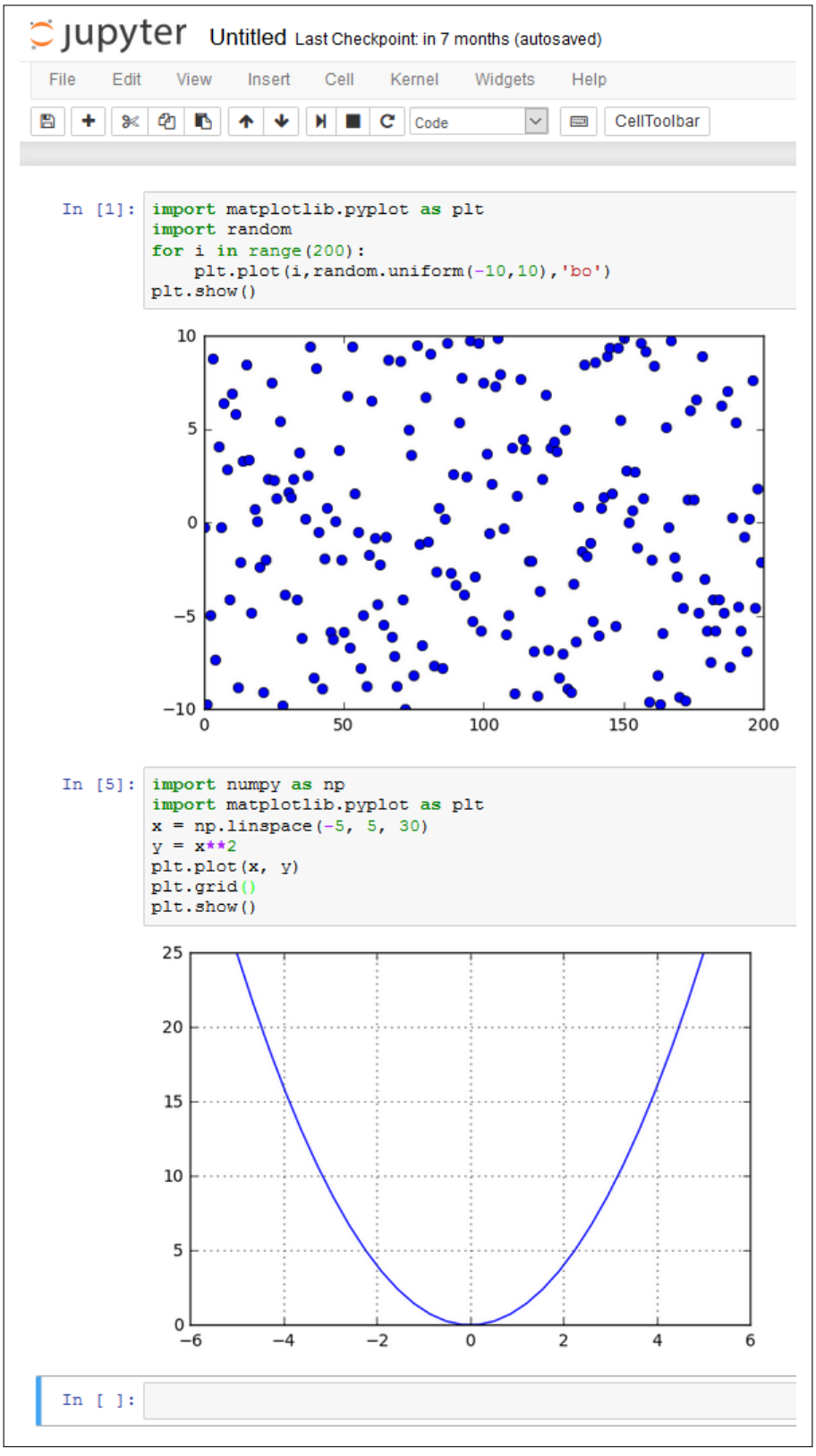
Jupyter est une application web qui propose une interface interactive pour programmer en Python.

Pour pouvoir l'utiliser, il faut avoir accès à un serveur Jupyter. Comme ce n'est actuellement pas encore très répandu, nous n'avons pas présenté cette possibilité dans un premier temps.

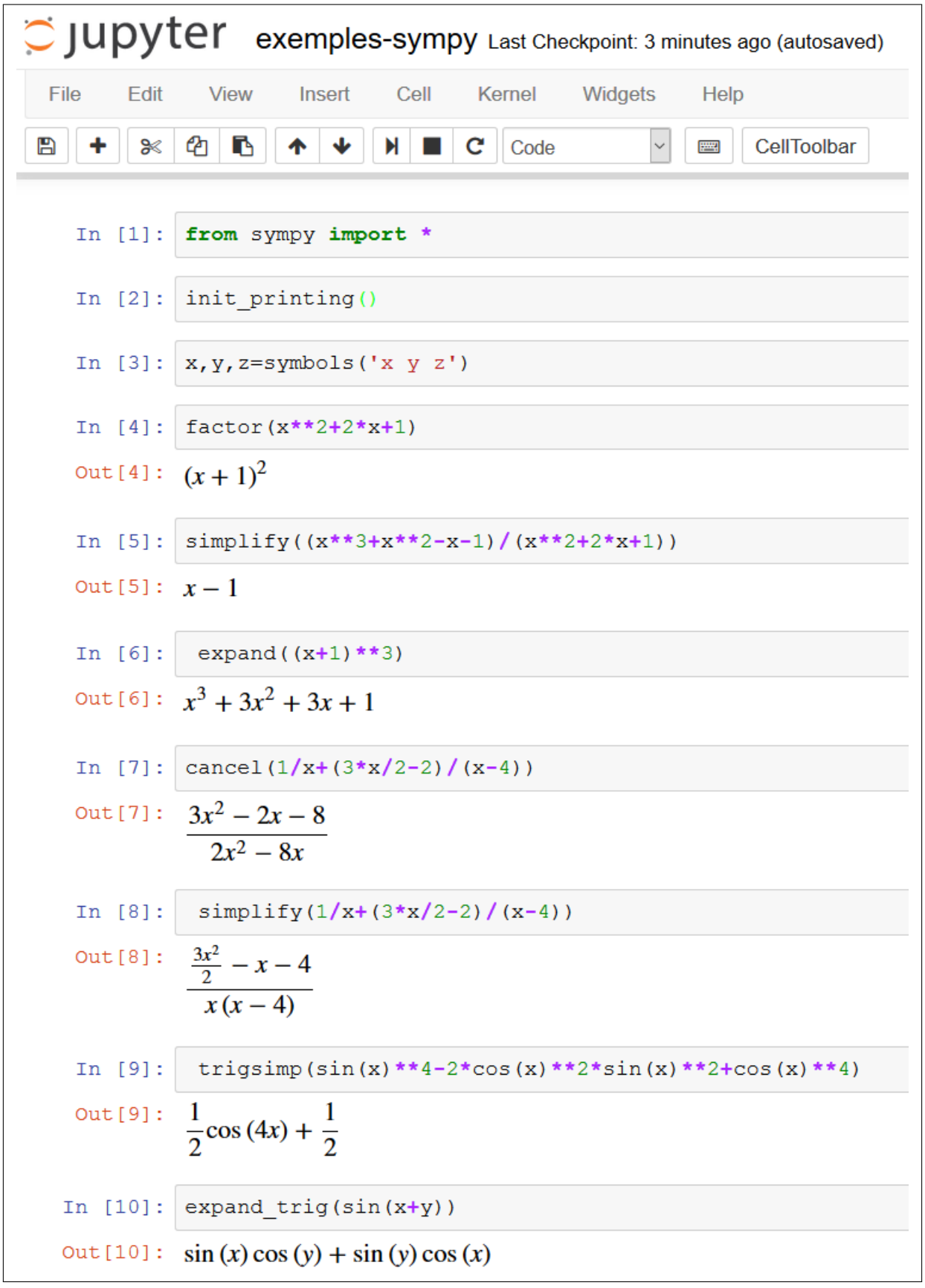
Il est possible que l'utilisation de Jupyter se développe beaucoup à l'avenir.

Pour essayer : https://try.jupyter.org/ (Ce lien fonctionne pour de petits tests uniquement).

Voici un premier exemple :



Sur l'image ci-dessous, on voit que l'affichage des résultats fournis par `sympy` est très lisible :



# 11 Installation de Python

Parfois l'utilisation de Python en ligne ne suffit pas. Voici quelques exemples d'installation possibles.

## 11.1 EduPython (pour Windows seulement)

C'est une version de Python qui peut fonctionner dans un dossier ou sur une clé USB. Elle contient une sélection de modules utiles qui sont déjà installés.

Actuellement en version 2.3. (Testé et utilisé pendant plus d'un an en classe en version 1.3). Python 3.4. Place une fois installé (926 Mo)

http://edupython.tuxfamily.org/

## 11.2 Python pour Windows (version officielle)

Parfois on préfère utiliser la distribution officielle de Python.

### 11.2.1 Instructions d'installation

Télécharger le fichier d'installation sur https://www.python.org/downloads/ puis exécuter le fichier téléchargé `python-3.6.1.exe`


Quelques modules préinstallés :

- `turtle`

Quelques modules non installés :

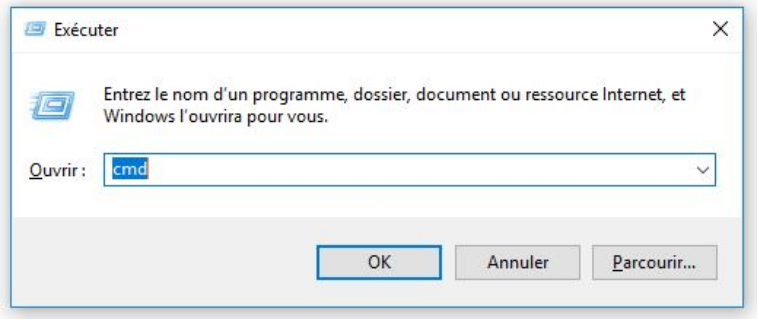
- `matplotlib`

### 11.2.2 Instructions d'installation des modules additionnels

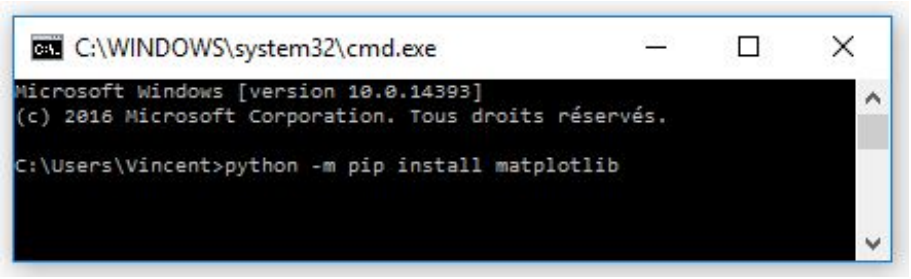
Pour installer des modules supplémentaires, on suivra les instructions officielles sur https://docs.python.org/3/installing/

Par exemple, si on veut installer `matplotlib`, la méthode actuelle est la suivante :

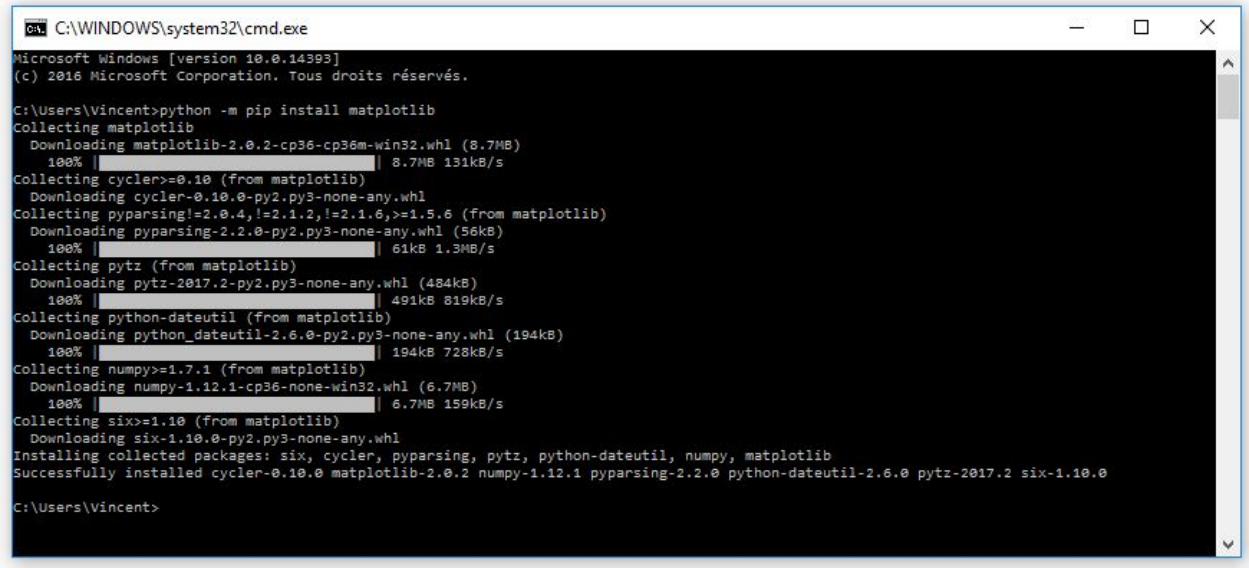
- Ouvrir un terminal par le raccourci clavier `Windows+R` puis saisir `cmd` et valider.




- Saisir `python -m pip install matplotlib` et valider.

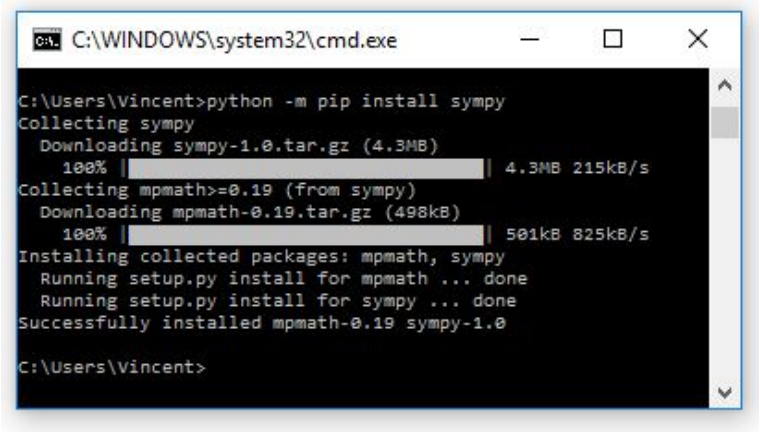



- Tout s'installe automatiquement.




On remarque que certains autres modules sont installés automatiquement (car `matplotlib` les utilise).

Installation de `sympy` :




## 11.3 Sous GNU/Linux

Nous n'avons pas pu faire de tutoriel dans ce cas (par manque de temps) mais normalement Python est déjà installé sous Linux. L'ajout des modules est assez simple a priori.

# 12 Solutions des exercices

**Attention :** Dans ce fichier PDF, les tabulations en début de ligne ne sont pas bien reconnues lorsqu'on fait un « copier-coller ». Il faut donc rajouter manuellement les indentations après avoir fait « coller ». C'est un problème connu [4], que nous n'avons pas pu contourner pour le moment.

**Exercice 1** Réponse n=135.

**Exercice 3.6**

```
def f(x):
    return x*x+20*x-12

def dichotomie(f,a,b,e):
    if f(a)*f(b)>0:
        print("La fonction ne s'annule pas dans l'intervalle [{:d}, {:d}]" .
    format(a,b))
        return False
    else:
        while(abs(b-a)>e):
            m=(a+b)/2
            if f(m)*f(b)>0:
                b=m
            else:
                a=m
    return m

print(dichotomie(f,-15,15,0.0001))
```

**Exercice 4**

```
import random
notes=['do','re','mi','fa','sol','la','si']
echantillon=[]
for i in range(10):
    d=random.randint(1,7)
    echantillon.append(notes[d-1])
print (echantillon)
```

**Exercice 5**

```
import numpy as np

def f(x):
    return x/(x**2+1)

a=float(input("Entrer la borne inferieure :"))
b=float(input("Entrer la borne superieure :"))
p=float(input("Entrer le pas :"))
maliste=np.linspace(a,b,(b-a)/p+1,endpoint=True)
print(maliste)
print(f(maliste))
```

**Exercice 6**

4. Par exemple ici et ici.

```
# hexagone
import turtle as t

for k in range(6):
    t.forward(100)
    t.left(60)
```

### Exercice 7

```
# pentagone
import turtle as t

for k in range(5):
    t.forward(100)
    t.left(72)
```

### Exercice 8

```
# Les 4 segments et les 3 carres
import turtle as t

# les 4 segments
longueurLigne = 50
for k in range(4):
    t.left(90)
    t.forward(longueurLigne)
    t.penup()
    t.right(180)
    t.forward(longueurLigne)
    t.left(90)
    t.forward(25)
    t.pendown()
    longueurLigne = longueurLigne + 25

# deplacement entre les 2 dessins
t.penup()
t.goto(0, 140)
t.pendown()

# les 3 carres
for numeroCarre in range(3):
    for numeroCote in range(4):
        t.forward(50)
        t.left(90)
    t.penup()
    t.forward(50)
    t.pendown()
```

https://repl.it/DoYJ/223

### Exercice 9

```
# carres imbriques
import turtle as t

longueurCote = 10
for numeroCarre in range(10):
    for numeroCote in range(4):
        t.forward(longueurCote)
```

```
        t.left(90)
    longueurCote = longueurCote + 10
```

### Exercice 10

```
# une spirale
import turtle as t

for i in range(25):
    t.circle(5*i,90)
    t.write(i)
```

### Exercice 11

```
import turtle as t
import numpy as np

t.degrees()
a=10 #le cote fid'origine est 10cm
m=20 #20 pixels par cm.

def square(a):
    for i in range(4):
        t.forward(a)
        t.left(90)

for i in range(10):
    p=t.pos()
    t.penup()
    t.goto(250,200-20*i)
    t.write(a*a)
    t.goto(p)
    t.pendown()

    square(a*m)
    b=np.sqrt(a**2-2*a+2)
    c=np.arctan(1/(a-1))
    c=np.degrees(c)
    t.forward(m)
    t.left(c)
    a=b
```

### Exercice 12

```
# une marche aleatoire
import turtle as t
import random as rd

x=0
y=0
t.dot(200,'gray90')
compteur=0
while x**2+y**2<10000 and compteur <1000:
    compteur=compteur+1
    x=x+rd.randint(-20,20)
    y=y+rd.randint(-20,20)
    t.goto(x,y)
    t.dot(4,'red')
t.write(compteur)
```

### Exercice 13

```
# Avec une version de Python installee
import matplotlib.pyplot as plt
for n in range(13):
    plt.plot(n, 20/(n**2-14*n+51),'ro')
plt.show()
```

Remarque : on peut penser que c'est une suite croissante si on raisonne sur les termes jusqu'au rang 7, mais ce raisonnement n'est pas valable. La suite décroît à partir du rang 7.

### Exercice 14

```
# Avec une version de Python installee
# suite logistique:
# k parametre dans [0;4]
# u(0) dans [0;1]
# u(n+1)=k*u(n)*(1-u(n)) pour n>=0

import matplotlib.pyplot as plt
u = 0.652
k = 3.5
for n in range(100):
    u = k*u*(1-u)
    print("n=",n,"  u(n)=",u)
    plt.plot(n, u,'ro')
plt.show()
```

### Exercice 15

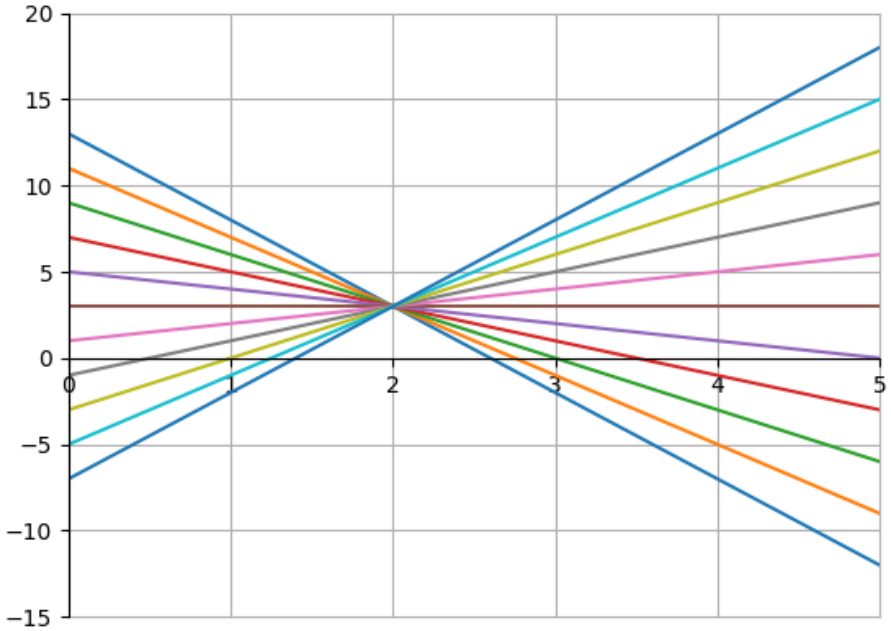
```
# trace d'une courbe point par point
# Avec une version de Python installee

import matplotlib.pyplot as plt

def f(x):
    return x**2

delta = 0.1 # ou 0.005
x = -5
while x <5 :
    plt.plot(x,f(x),'m.')  # ou 'm,'
    x = x + delta

plt.grid()
plt.title("Titre du graphique")
plt.xlabel("abscisses")
plt.ylabel("ordonnees")
plt.show()
```

### Exercice 16

1. Voici le code et le graphique obtenus.

```
# Avec une version de Python installee
# famille de fonctions f_k pour -5 <= k <=5
import numpy as np
import matplotlib.pyplot as plt

for k in range(-5,6):
```

```
    x = np.linspace(0, 5, 30)
    y = k*(x-2)+3
    plt.plot(x, y)
plt.axis([0, 5, -15, 20])
plt.grid()

ax = plt.gca()
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data',0))
ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data',0))

plt.show()
```




2. On a l'impression que les droites sont concourantes, au point de coordonnées $(2; 3)$.
3. Pour tout entier $k$ compris entre $-5$ et 5, $f_k(2) = k(2-2)+3 = 0+3 = 3$. Donc la conjecture est vérifiée.

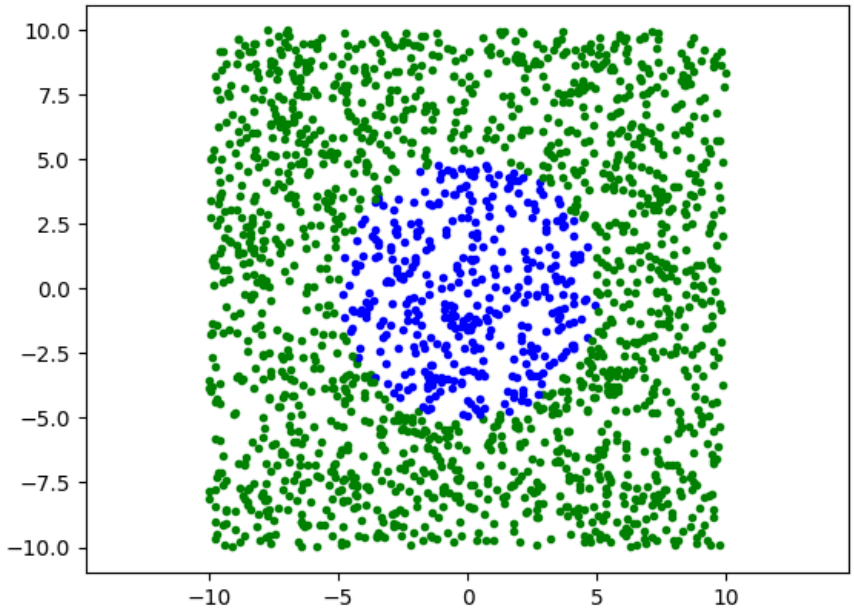
**Exercice 17**

```
# Avec une version de Python installee
import numpy as np
import matplotlib.pyplot as plt
import random as rd

def f(a):
    return np.cos(np.sqrt(a))*np.exp(-a)

fig = plt.figure()
ax = fig.add_subplot(111)
xx = np.linspace(0, 1, 40)
ax.plot(xx, f(xx))
```

```
plt.axis([0,1,0,1])
ax = plt.gca()
ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data',0))
ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data',0))

s=0
nbpoints=1000
for i in range(nbpoints):
    x=rd.uniform(0,1)
    y=rd.uniform(0,1)
    if y>f(x):
        ax.plot(x,y,'bo',markersize=2)
    else:
        ax.plot(x,y,'ro',markersize=2)
        s+=1

ax.annotate(r'$I \approx {}$'.format(s/nbpoints),xy=(0.5,0.8),xytext=(0.5,0.8),
    fontsize=30)
plt.show()
```

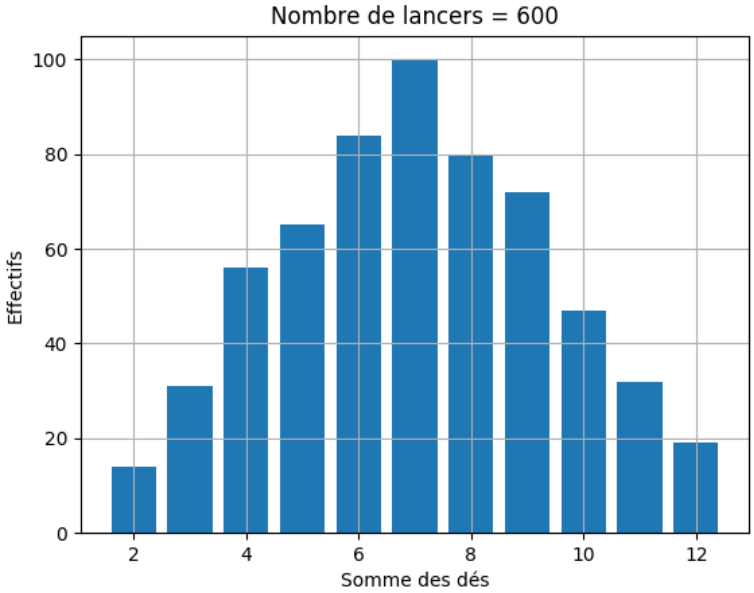
### Exercice 19

```
# Avec une version de Python installee
# Points aleatoires et couleur selon un disque
import matplotlib.pyplot as plt
import random

plt.axis([-10,10,-10,10])
plt.axis('equal')
for i in range(2000):
    x = random.uniform(-10, 10)
    y = random.uniform(-10, 10)
    if x**2+y**2<25 :
        plt.plot(x,y,'b.')
    else:
        plt.plot(x,y,'g.')

plt.show()
```




### Exercice 20

```
# Avec une version de Python installee
import matplotlib.pyplot as plt
import random

plt.grid(True)
plt.xlabel("Somme des des")
plt.ylabel("Effectifs")
effectif = 0
nb_tirages = 600
plt.title("Nombre de lancers = "+str(nb_tirages))

liste_valeurs   = [2,3,4,5,6,7,8,9,10,11,12]
liste_resultats = [0,0,0,0,0,0,0,0, 0, 0, 0]
for i in range(1,nb_tirages+1):
    de1 = random.randint(1, 6)
    de2 = random.randint(1, 6)
    somme = de1 + de2
    liste_resultats[somme-2] += 1
plt.bar(liste_valeurs,liste_resultats)
plt.show()
```




### Exercice 21

```
# voici les 4 lignes a completer
for i in range(1,nb_tirages+1):
    tirage = rd.randint(0, 1)
    somme = somme + tirage
    moyenne = somme/i
```

# 13 Compléments, après la formation

## 13.1 Attention : éviter Python 2

Nous avons utilisé Python 3 dans tout ce document. Il faut maintenant éviter d'utiliser Python 2 car il y a des incompatibilités. Par exemple, en Python 2, on écrit `print "Bonjour"` alors qu'en Python 3, on écrit `print("Bonjour")`. Python 3 est sorti en décembre 2008.

## 13.2 Le type booléen

Une expression est dite *booléenne* lorsqu'elle prend la valeur `True` ou bien la valeur `False`. On parlera aussi de variable de *type booléen*. Le type booléen est donc un autre type de données, que nous n'avions pas encore abordé dans ce document. Ci dessous, une console Python.

```
>>> b = 10 # ceci est une affectation (simple egal)
>>> b>8
True
>>> b==5 # affiche False, le test d'egalite s'ecrit avec un double egal
False
>>> b!=10 #  != signifie est different de
False
>>> 0<= b <=20
True
>>> True or False
True
>>> True and False
False
>>> not True
False
>>> type(True)
<class 'bool'>
```

## 13.3 L'utilisation de `elif`

On peut si besoin utiliser la structure suivante :

```
if condition:
    action1
elif condition2: # sinon si la condition2 est verifiee
    action2
else:         # sinon
    action3   # executer cette instruction
# remarque : il est possible de mettre plusieurs fois elif
```

## 13.4 Les pièges dans l'utilisation des nombres complexes

On rappelle qu'ici `1j` désigne le nombre complexe noté $i$ en mathématiques.

```
>>> 1j
1j
>>> 1J # on peut aussi utiliser J majuscule
1j
```

L'exemple ci-dessous montre que `j` tout seul ne veut rien dire (cela pourrait être une variable mais ici elle n'a pas été déclarée).

```
>>> j
Traceback (most recent call last):
  File "<pyshell#26>", line 1, in <module>
```

```
    j
NameError: name 'j' is not defined
```

Dans le premier calcul ci-dessous, `5j` est une notation spéciale et non une multiplication de `5` par `j`, c'est prioritaire par rapport a la division. Il n'y a pas d'erreur mais il est vrai que ces conventions sont *très subtiles* et risquent de poser des problèmes si on les utilise en classe.

```
>>> 2-4/5j     # premier calcul
(2+0.8j)
>>> 2-(4/5)*1j # deuxieme calcul
(2-0.8j)
```

## 13.5 Les coordonnées `(x,y)` dans le module `turtle`

Les coordonnées utilisées par l'instruction `t.goto(x,y)` sont bien celles utilisées habituellement en mathématiques, comme le montre l'exemple ci-dessous (à essayer dans la console). L'origine est centrée, les ordonnées croissantes correspondent bien à un déplacement vers le haut.

```
>>> import turtle as t
>>> t.goto(100,0)
>>> t.goto(100,50)
>>> t.goto(0,50)
>>> t.goto(0,50)
```

## 13.6 Un exemple pour montrer d'autres façons d'écrire un algorithme

```
# carres imbriques
import turtle as t

# ici on utilise une fonction qui trace un carre
def tracerCarre(longueurCote):
    for k in range(4):  # comme range(0,4), donc k va de 0 a 3 inclus.
        t.forward(longueurCote)
        t.left(90)

longueur = 10 # un nom de variable peut comporter plusieurs lettres
for numeroCarre in range(10):
    tracerCarre(longueur)
    longueur = longueur + 10
```

## 13.7 Afficher un cercle avec matplotlib : méthode plus concise

```
# Avec une version de Python installee

import matplotlib.pyplot as plt

plt.axis('equal') # pour avoir un repere orthonorme
plt.axis([0, 6, 0, 4]) # valeurs indicatives pour les axes
plt.grid()  # pour afficher le quadrillage

# pour afficher un cercle
cercle = plt.Circle((3,2), radius=1, color='g', fill=False)
plt.gcf().gca().add_artist(cercle)
# gcf() signiefie Get Current Figure, gca() signifie Get Current Axis

plt.show()
```

Méthode trouvée sur ce lien.