
	Projet 1	1 / 6
	La suite de Syracuse	

1 Présentation

1.1 Objectifs du projet

- faire preuve d'autonomie, d'initiative et de créativité ;
- présenter un problème ou sa solution, développer une argumentation ;
- coopérer au sein d'une équipe rechercher de l'information, partager des ressources ;
- rechercher de l'information ;
- écrire le programme informatique correspondant au cahier des charges ;
- créer un diaporama de présentation (en mode collaboratif, travail à plusieurs sur le même document).

1.2 Les outils à utiliser

- Programmation : Spyder
- Diaporama de présentation : Powerpoint office 365 version de l'ENT.

1.3 L'organisation

- Le projet se fait à deux ou à trois.
- Vous devrez rendre et ou présenter :
 - Le diaporama
 - Le code python

2 Quelques consignes

- Travailler à plusieurs ne veut pas dire faire exactement la même chose,
- Se répartir le travail en fonction des indications données dans le descriptif,
- Utiliser des fonctions pour structurer le code,
- Utiliser des noms de variables explicites mais de longueur correcte,
- Ecrire un ou plusieurs algorithmes avant de commencer à coder,
- Commenter le code informatique,
- Penser à faire des sauvegardes régulières et en plusieurs exemplaires.

3 Descriptif

3.1 Définition générale

En mathématiques, on appelle **suite de Syracuse** une suite d'entiers naturels définie de la manière suivante:

- On part d'un nombre entier plus grand que zéro :
 - s'il est pair, on le divise par 2
 - s'il est impair, on le multiplie par 3 et on ajoute 1
- En répétant l'opération, on obtient une suite d'entiers positifs dont chacun ne dépend que de son prédécesseur.

On a constaté, sans parvenir à le démontrer, que l'on finit toujours par obtenir 1, quel que soit le nombre choisi au départ (ensuite, la même séquence de nombres se répète : 1, 4, 2, 1...).

Exemple

Par exemple, la suite de Syracuse partant de **11**, est :

11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1.

3.2 Les termes associés

3.2.1 Vol

En se bornant à 1, on appelle cette suite finie d'entiers le **vol** de « l'entier de départ », car la suite de Syracuse fait penser au vol d'une feuille morte, qui finit toujours par retomber au sol, la suite étant celle des altitudes de la feuille.

Pour la suite de Syracuse partant de 11, la suite d'entiers finie est appelée :

- **vol de 11**

3.2.2 Etape

On appelle **étape** un nombre quelconque de la suite finie.

Pour la suite de Syracuse partant de 11:

- **17 est une étape du vol de 11**

3.2.3 Etape maximale

On remarque que la suite atteint une étape maximale, appelée **altitude maximale** du vol.

Pour la suite de Syracuse partant de 11:

- **Par exemple 52 est l'altitude maximale du vol de 11.**

3.2.4 Temps de vol

On appelle **temps de vol** le nombre d'étapes minimum pour atteindre 1.

Attention le terme initial n'est pas compté.

Pour la suite de Syracuse partant de 11:

- **Par exemple 14 est le temps de vol.**

3.2.5 Temps de vol en altitude

On appelle **temps de vol en altitude** le nombre d'étapes nécessaires pour redescendre en dessous de l'altitude initiale.

Pour la suite de Syracuse partant de 11:

- **Par exemple 8 est le temps de vol en altitude.**

4 Travail à faire

4.1 Le principe

Question 1. Faire un bref historique de « La suite de Syracuse ». Utiliser une frise chronologique.

Question 2. Retrouver la suite de Syracuse de l'entier 11.

Question 3. Donner la suite de Syracuse en partant d'un entier compris entre 1 et 40 ainsi que de tous les paramètres liés (nombre d'étapes, étape maximale...).

4.2 Programmation

Si vous êtes un groupe :

- de deux, chaque élève programme ou implémente trois fonctions sur les six.
- de trois, chaque élève programme ou implémente deux fonctions sur les six.

4.2.1 Implémentation des fonctions

Question 4. :

1. Chaque élève crée un fichier dans spyder
2. La première fonction est implémentée
 - elle est testée
 - elle est modifiée si le fonctionnement n'est pas celui attendu (jusqu'à obtention du fonctionnement attendu)
 - elle est validée
3. Toujours dans le même fichier on implémente :
 - la deuxième fonction en appliquant le même mode opératoire (voir 3.)
 - la troisième fonction en appliquant le même mode opératoire (voir 3.)
4. Chaque élève doit **rendre** son travail via l'ENT et « Travail à faire »

4.2.2 Détail des fonction

Fonction **saisiEntier()** qui :

- a. demande à l'utilisateur de saisir un entier avec les contraintes suivantes:
 - un message clair indique à l'utilisateur ce qui est attendu ;
 - la saisie au clavier du nombre doit être compris entre 2 et 200 ;
 - Si la valeur saisie n'est pas dans l'intervall [2 , 200], on demande à l'utilisateur de saisir à nouveau une valeur.

Fonction **etapeSuivante()** qui :

- b. reçoit un seul argument de type entier strictement positif,
- c. calcule l'entier suivant dans la suite de Syracuse et le retourne. Par exemple **etapeSuivante(5)** doit produire 16 et **etapeSuivante(16)** doit faire 8, etc ;
- d. renvoi la valeur de l'entier suivant calculé.

Fonction **vol()** qui :

- e. reçoit un seul argument de type entier strictement positif,
- f. crée une liste contenant la première valeur de la suite,
- g. en utilisant la fonction **etapeSuivante()**, calcule et stocke dans la liste créée les termes de la suite de Syracuse de l'entier initial jusqu'à 1 inclus ;
- h. renvoi les valeurs utiles à l'affichage des informations demandées par le programme principal.

Fonction **tempsVol()** qui :

- i. reçoit un seul argument de type liste ;
- j. qui calcule la valeur du temps de vol ;
- k. renvoi la valeur du temps de vol sous forme d'un entier ;

Fonction **altMaxi()** qui :

- l. reçoit un argument de type liste ;
- m. qui détermine l'altitude maximale de la suite qui peut être la valeur initiale ;
- n. renvoi la valeur.

Fonction **tempsVolAltitude()** qui :

- o. reçoit un argument de type liste ;
- p. qui détermine le temps de vol de la suite ;
- q. renvoi la valeur.

4.2.3 Mise en commun des fonctions implémentées

Question 5. :

1. **Créer** un fichier dans spyder
2. **Intégrer** l'ensemble des fonctions et les rendre opérationnelles.
3. **Créer** une fonction d'affichage qui permet d'afficher toutes les valeurs des termes associés.
4. Un élève du groupe doit **rendre** le travail via l'ENT et « Travail à faire »

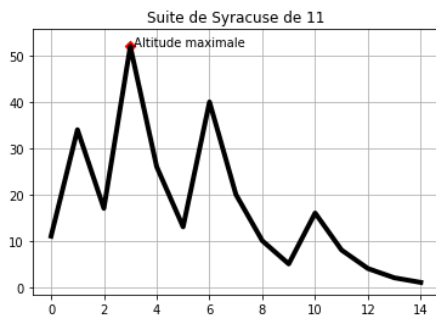
4.2.4 Affichage graphique

Question 6. :

Pour chaque élève :

1. **Dupliquer** le programme rendu précédemment de façon à ce que chaque élève dispose du fichier.
2. **Compléter** le programme de façon à afficher la courbe de la suite de Syracuse du nombre entier saisi. Il vous faut utiliser la bibliothèque « Matplotlib ». Un extrait de la documentation est fourni.

Voici un exemple de ce que vous pourriez obtenir :



3. Chaque élève doit **rendre** son travail via l'ENT et « Travail à faire »

4.3 Diaporama de présentation

Question 7. **Créer** pour le groupe un diaporama de présentation du projet dans sa globalité. Il devra comporter entre 6 et 8 diapositives maximum.

Vous trouverez des conseils pour créer un diaporama de qualité [ici](#).

<https://www.youtube.com/watch?v=J-9eaJOem1M>

Ressource d'utilisation de la bibliothèque Matplotlib

Vous pouvez aussi vous servir des ressources disponibles sur le net.



La bibliothèque *matplotlib* doit être appelée pour utilisation des graphiques.

Dans la suite de la fiche, nous supposons que la ligne suivante a été insérée au début du script.

```
import matplotlib.pyplot as plt
```

On suppose dans l'ensemble de la présente fiche que les listes *x* et *y* ont été déclarées au préalable avec les données à utiliser pour les graphiques.

Fonctions principales de *matplotlib*

(Consulter le site <https://matplotlib.org/> pour la notice complète)

Fonctions	Actions réalisées
<code>plt.clf()</code>	Supprimer les tracés précédents
<code>plt.plot(x, y, styleDuGraphe, linewidth=1, label = 'y = f(x)')</code>	Tracer la courbe représentant <i>y</i> en fonction de <i>x</i> avec le style <i>styleDuGraphe</i> , l'épaisseur <i>linewidth</i> , le nom de la courbe à afficher dans la légende étant <i>label</i>
<code>plt.xlabel('x - axe des abscisses')</code> <code>plt.ylabel('y - axe des ordonnées')</code>	Ajouter des libellés sur les axes
<code>plt.axis([-5.5,5.5,0,10])</code> ou <code>plt.xlim(-5.5,5.5)</code> <code>plt.ylim(0,10)</code>	Définir des valeurs minimales et maximales pour les abscisses (-5.5 et 5.5) et les ordonnées (0 et 10)
<code>plt.title('Représentation de y en fonction de x')</code> <code>plt.title(r"\$\Delta E = \frac{h}{\lambda} \times c\$ {\lambda} \$ (J)\$")</code>	Ajouter un titre au graphique <i>NB</i> : en ajoutant un <i>r</i> devant la chaîne de caractères, on peut afficher des formules mathématiques à l'aide de la syntaxe LATEX
<code>plt.grid()</code>	Ajouter une grille au graphique
<code>plt.text(2, 3.5, 'Point de fonctionnement')</code> <code>plt.annotate('Maximum', xy=(1.5, 1), xytext=(2,1.5), arrowprops=dict(facecolor='black', arrowstyle='->'))</code>	Ajouter du texte dans le graphe à la position souhaitée Ajouter une annotation à la position souhaitée <i>xytext</i> et trace une flèche jusqu'au point <i>xy</i>
<code>vecteur = plt.quiver(xVecteur, yVecteur, vecteurX, vecteurY, scale=echelleVecteur, color='r',angles='xy', units='xy')</code> <code>plt.quiverkey(vecteur, 0.1, 0.1, 2, label='échelle 2 m/s', coordinates='data')</code>	Tracer un vecteur au point d'application (<i>xVecteur</i> , <i>yVecteur</i>), <i>vecteurX</i> composante suivant <i>x</i> , <i>vecteurY</i> composante suivant <i>y</i> Trace l'échelle correspondant au vecteur <i>vecteur</i> , en position (0.1,0.1) sur le graphique et valeur de l'échelle.
<code>plt.legend()</code>	Ajouter une légende avec le nom des courbes
<code>plt.show()</code>	Afficher le graphe

Enjoliver les graphes

✓ Paramètres de la fonction plot

`plt.plot(x, y, styleDuGraphe)` où `styleDuGraphe` est une chaîne de caractères qui regroupe la couleur de la courbe, le marqueur de point et le style de liaison entre les points.

Chaîne	Marqueur de point
.	point
,	pixel
o	rond
v	triangle pointe en bas
^	triangle pointe en haut
<	triangle pointe à gauche
>	triangle pointe à droite
1	croix à 3 branches vers le bas
2	croix à 3 branches vers le haut
3	croix à 3 branches vers la gauche
4	croix à 3 branches vers la droite
s	carré
p	pentagone
*	étoile
h	hexagone
H	hexagone
+	plus
P	plus plein
x	croix
X	croix pleine
d	carreau
D	carreau plus grand
	barre verticale
_	barre horizontale

Chaîne	Couleur en anglais	Couleur en français
b	blue	bleu
g	green	vert
r	red	rouge
c	cyan	cyan
m	magenta	magenta
y	yellow	jaune
k	black	noir
w	white	blanc

Voir la palette complète sur https://matplotlib.org/gallery/color/named_colors.html

Chaîne	Style de ligne
-	ligne continue
--	tirets
:	ligne en pointillé
-.	tirets points

Exemple : `plt.plot(x, y, 'r+:')` → trace un graphe dont les points sont rouges, en forme de + et reliés par des lignes en pointillé.

À noter !

La fonction `plot` découpe l'option `styleDuGraphe` en morceaux :

- ✓ **Une couleur** (une seule lettre acceptée), que l'on peut aussi donner avec le mot clé `color=' '` (plus de contrainte sur le nom de la couleur). De nombreuses autres fonctions de pyplot (`plt.grid`, `plt.xlabel`, `plt.ylabel`, `plt.title`, ...) utilisent ce mot clé permettant l'usage de toute la palette des couleurs.
- ✓ **Un style de marqueur de point**, que l'on peut aussi donner avec le mot clé `marker=' '`
- ✓ **Un style de ligne**, que l'on donne avec le mot clé `linestyle=' '`, souvent abrégé en `ls=' '`

Si rien n'est précisé, Matplotlib utilise simplement le paramètre par défaut pour ces 3 styles.

✓ Types de graphes

`plt.plot()` : pour tracer des courbes
`plt.scatter()` : pour tracer des points
`plt.bar()` : pour des diagrammes à barre
`plt.pie()` : pour des camemberts
`plt.hist()` : pour les histogrammes

✓ Créer des grilles de graphes

Il est possible de créer des grilles de graphes, solution très pratique pour empiler des graphes qui doivent être regardés ensemble mais qui n'ont pas les mêmes ordres de grandeurs en matière d'abscisses et d'ordonnées.

On utilise l'instruction `subplot()` qui va décrire une grille. Cette commande prend plusieurs arguments :

1. Nombre de lignes de la grille de graphe
2. Nombre de colonnes de la grille de graphe
3. Index du graphe dans la grille (la numérotation se fait de gauche à droite et de haut en bas)
4. Options

Exemple avec une grille de 1 colonne et 2 lignes (la couleur de fond du second graphe sera cyan) :

```
plt.subplot(1, 2, 1)
plt.plot(x, y, 'k-.', linewidth=2)
plt.subplot(1, 2, 2, facecolor='c')
plt.plot(t, y, 'r+:', linewidth=1)
plt.show()
```