Atelier 3  Fonctions en Python

**Activité « Histoire de moyenne »**

**Contenus mathématiques :**

* Calcul de moyennes pondérées

**Contenus Python :**

* Créations et utilisation de fonctions avec arguments

**Prérequis**

* Instructions de base en Python
* Importation d’un module
* Définition d’une fonction

1. **a)** Rappeler la formule permettant de calculer la moyenne de deux notes dont on connait les coefficients.

**b)** Créer une fonction ***moyenne*** en langage Python, qui permet de calculer la moyenne pondérée de deux notes, et qui prends comme paramètres les deux notes et leurs coefficients.

|  |
| --- |
| def moyenne(n1,c1,n2,c2):  return (n1\*c1+n2\*c2)/(c1+c2) |
| Console  >>> moyenne(10,2,17,1)  12.333333333333334  >>> |

1. ***Situation***

La fin du trimestre arrive, Alice a un dernier contrôle en SVT cette semaine et elle se demande quelle note obtenir pour avoir 12 de moyenne.

Alice réfléchit : « j’ai déjà eu trois devoirs, à coefficient 2, deux interro coefficient 1 et un DM coefficient 0,5 mais je ne me souviens plus exactement de mes notes ; par contre la prof nous a donné nos moyennes provisoires, j’ai 10,7 … le prochain devoir sera coefficient 2. Si j’ai 10, quelle sera ma nouvelle moyenne ? »

1. Comment utiliser la fonction *moyenne* précédente pour connaître la nouvelle moyenne ?
2. Tester avec comme note au nouveau devoir 10, puis 11, puis 12. A-t-elle atteint son objectif ?

|  |
| --- |
| Console :  >>> moyenne(10.7,8.5,10,2)  10.566666666666666  >>> moyenne(10.7,8.5,11,2)  10.757142857142856  >>> moyenne(10.7,8.5,12,2)  10.947619047619046  >>> |

1. Quelle équation faudrait-il résoudre pour connaître la note à obtenir pour avoir plus de 11 ? La résoudre.
2. Si maintenant l’objectif d’Alice est d’avoir 12, il faudrait de nouveau résoudre une équation pour trouver la note à obtenir.

On veut donc écrire une fonction ***note\_manquante*** en langage Python, qui permet de calculer la note à obtenir pour qu’Alice atteigne son objectif, et qui prend pour argument la moyenne visée.

1. Est-ce que Alice peut atteindre la moyenne qu’elle veut ?

Écrire une fonction ***objectif\_atteint*** en langage Python, qui prend pour argument la moyenne visée, et qui permet d’avertir Alice si son objectif est atteignable ou non, et si oui, de lui donner la note à obtenir.

|  |
| --- |
| from lycee import \*  def moyenne(n1,c1,n2,c2):  return (n1\*c1+n2\*c2)/(c1+c2)  def note\_manquante(moy\_visee):  return (10.5\*moy\_visee-93.5)/2  def objectif\_atteint(moy\_visee):  if note\_manquante(moy\_visee)>20:  print("Tu ne peux pas atteindre une moyenne de",moy\_visee)  else:  print("Tu peux atteindre une moyenne de",moy\_visee,"avec une note de :",note\_manquante(moy\_visee)) |

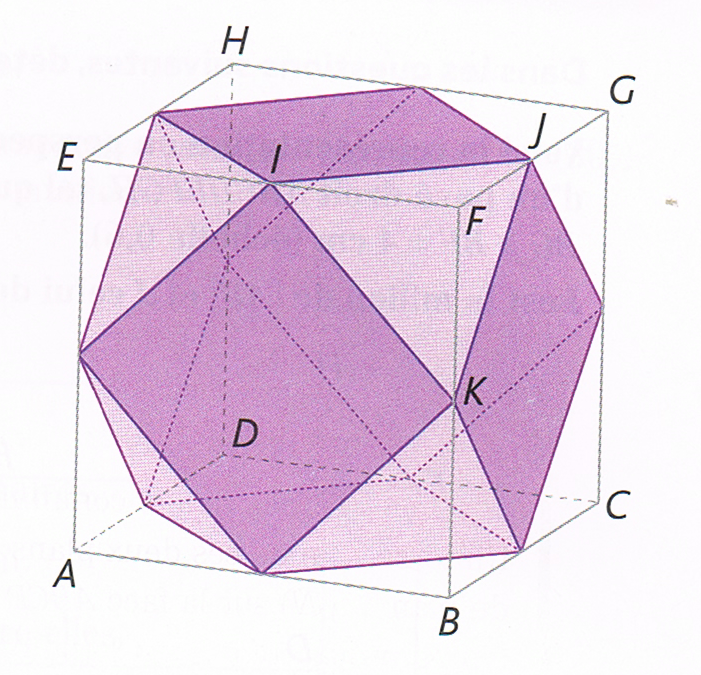
**Activité « Des aires et des volumes »**

**Contenus mathématiques :**

* Calcul de volumes des solides étudiés en collège

**Contenus Python :**

* Créations et utilisation de fonctions avec arguments

**Prérequis**

* Instructions de base en Python
* Importation d’un module
* Définition d’une fonction

On considère la figure suivante :

*ABCDEFGH* est un cube d’arête 4 cm.

*I*, *J* et *K* sont les milieux respectifs des arêtes [*FE*], [*FG*] et [*FB*].

1. Créer les fonctions suivantes en langage Python :
   * ***hypo***, qui permet de calculer la longueur de l’hypoténuse en fonction des longueurs des deux autres côtés dans un triangle rectangle ;
   * ***aire\_tri***, qui permet de calculer l’aire d’un triangle en fonction des longueurs de la base et de la hauteur ;
   * ***vol\_tetra***, qui permet de calculer le volume d’un tétraèdre en fonction de l’aire de base et de la hauteur relative à cette base.
2. **Volume du cuboctaèdre**

Écrire une fonction en langage Python, qui permet de calculer le volume du cuboctaèdre ci-dessus, et utilisant une des fonctions créées précédemment.

1. On considère un triangle équilatéral de côté *c*.
2. Écrire la longueur d’une hauteur en fonction de *c*.
3. En déduire l’expression de l’aire du triangle équilatéral en fonction de *c*.
4. Créer une fonction ***aire\_tri\_equi***, qui permet de calculer l’aire d’un triangle équilatéral de côté *c*.
5. **Aire du cuboctaèdre**

Écrire une fonction en langage Python, qui permet de calculer l’aire du cuboctaèdre ci-contre, et utilisant certaines des fonctions créées précédemment.

\**indication : créer une variable intermédiaire, dans la fonction, permettant de calculer la longueur du côté des faces carrées*

|  |  |
| --- | --- |
| from lycee import \*  def hypo(b,c):  return sqrt(b\*\*2+c\*\*2)  def aire\_tri\_equi(c):  return (sqrt(3)\*c\*\*2/8)  def vol\_tetra(b,h):  return (b\*h)/3  def vol\_cuboc(c):  return (c\*\*3-8\*vol\_tetra(c/2,c/2))  def aire\_cuboc(c):  cote=hypo(c/2,c/2)  return (6\*cote\*\*2+8\*aire\_tri\_equi(cote)) | Console  >>> hypo(2,2)  2.8284271247461903  >>> aire\_tri\_equi(2.8)  1.6974097914174995  >>> vol\_tetra(2,2)  1.3333333333333333  >>> vol\_cuboc(4)  53.333333333333336  >>> aire\_cuboc(4)  61.856406460551035 |

**Activité « Polygones imbriqués »**

**Contenus mathématiques :**

* Calcul de volumes des solides étudiés en collège

**Contenus Python :**

* Module Turtle
* Créations et utilisation de fonctions avec arguments

**Prérequis**

* Instructions de base en Python
* Importation d’un module
* Définition d’une fonction

Turtle est un module de Python permettant de faire des dessins

**Les principales fonctions du module turtle**

***Déplacements :***

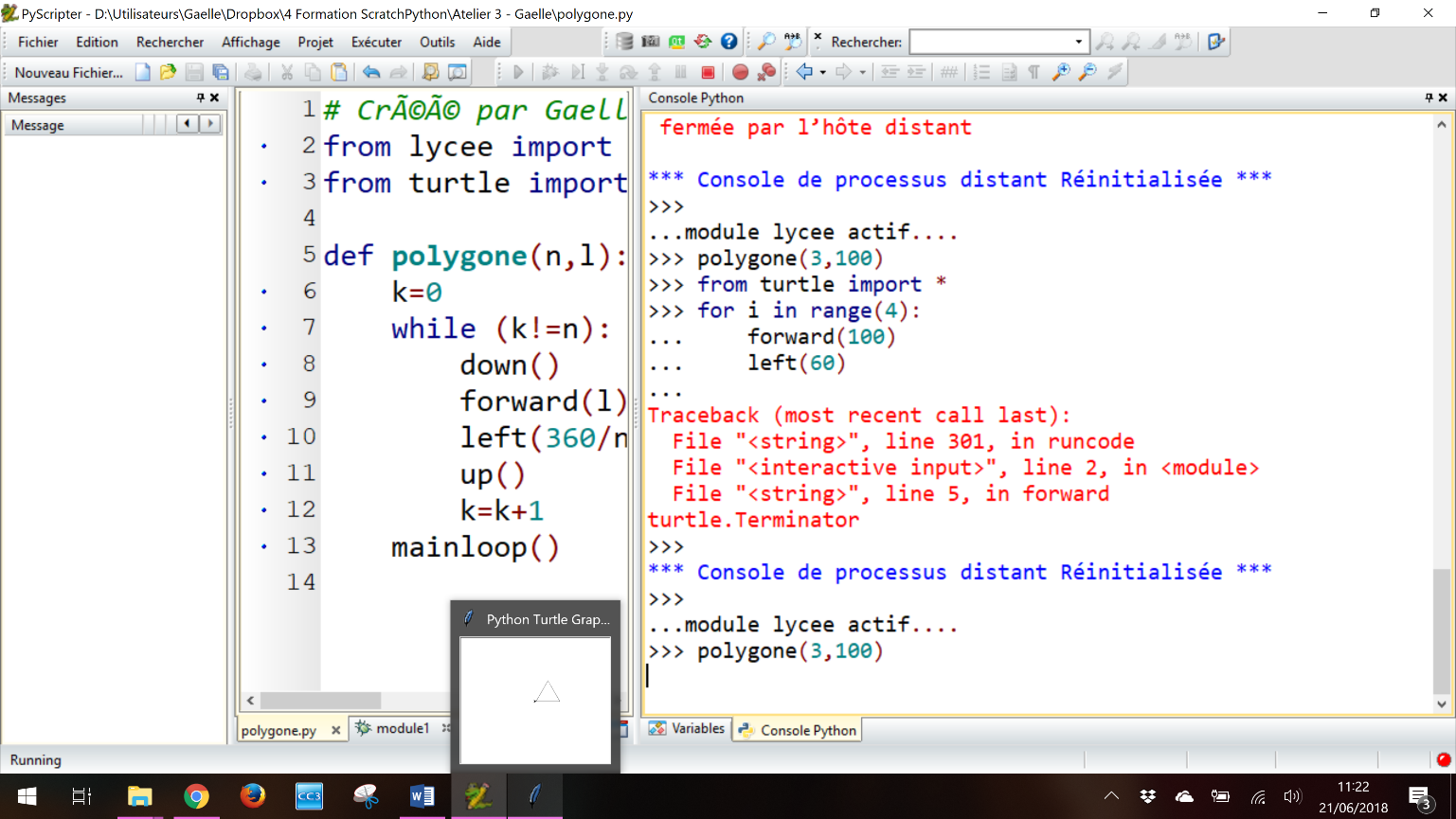
* **forward(d)** : avancer d'une distance d (en pixels)
* **backward(d)** : reculer d’une distance d (en pixels)
* **goto(*x,y*)** : positionner la tortue au point de coordonnées (*x ; y*)
* **left(*a*)** : fait pivoter la tortue d'un angle *a* degrés vers la gauche
* **right(*a*)** : fait pivoter la tortue d’un angle de *a* degrés vers la droite

***Tracés :***

* **up()** : relève le crayon (pour pouvoir avancer sans dessiner)
* **down()** : abaisse le crayon (pour pouvoir recommencer à dessiner)
* **width(épaisseur)** : choisit l'épaisseur du tracé (en pixels)
* **reset() :** nettoie la fenêtre de dessin, réinitialise la tortue ; elle est située alors au centre de l’écran de dessin tournée vers la droite.
* **color(c)** : la couleur par défaut est le noir, on peut la changer en mettant une couleur prédéfinie c : ‘red’, ‘green’, ‘blue’, ‘yellow’, . . .

Les fonctions données ci-dessus ne sont qu’une petite partie des possibilités . . . on peut aussi aller voir la documentation officielle python : http ://docs.python.org/3.2/library/turtle.html.

Lorsque vous exécuter le programme, une fenêtre de dessin s’ouvre :



Avec Edupython, il faut arrêter le processus lié à l’ouverture de cette fenêtre à la fin de vos programmes, avec l’instruction :

**mainloop()**

Et fermer la fenêtre avant de relancer le programme.

1. Écrire une fonction **polygone(n,l)**, qui permet de tracer un polygone régulier à **n** côtés de longueur **l** (n ≥ 3).

|  |  |
| --- | --- |
| from turtle import \*  def polygone(n,l):  k=0  while (k!=n):  down()  forward(l)  left(360/n)  up()  k=k+1  mainloop() | Console  >>> polygone(3,100)    >>> polygone(6,70) |

1. Écrire un code Python utilisant la fonction polygone précédente, permettant d’obtenir les figures suivantes, avec pour arguments le nombre de côtés, la longueur du côté de la petite figure, la couleur, le nombre de figures et le pas.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Nombre de côté : 3  Longueur de côté : 30  Couleur : red  Nombre de figures : 7  Pas : 15 | Nombre de côté : 5  Longueur de côté : 25  Couleur : blue  Nombre de figures : 10  Pas : 10 | Nombre de côté : 8  Longueur de côté : 10  Couleur : green  Nombre de figures : 10  Pas : 5 |

|  |  |
| --- | --- |
| from turtle import \*  def polygone(n,l):  k=0  while (k!=n):  forward(l)  left(360/n)  k=k+1  *#mainloop() si appel de la fonction seule*  def fig\_imbriq(n,l,col,k,pas):  i=0  color(col)  while (i!=k):  polygone(n,l)  l=l+pas  i=i+1  mainloop() | Console  >>> fig\_imbriq(5,60,'red',6,20) |