Utilisation de python au lycée :

Pout information voici le lien vers le document officiel décrivant les attendus du lycée en termes de programmation. Bien qu'il soit long à lire, sa lecture est vivement conseillée, elle permettra de comprendre ce que l'on doit attendre d'un élève en termes d'algorithmique. De plus le document officiel présente quelques activités à faire avec les élèves.

Document officiel

Pour résumer :

Les compétences attendues sont :

- Lire et comprendre un algorithme dans le but de l'utiliser.
- Compléter / corriger un algorithme.
- Ecrire un algorithme simple.
- Notion de fonction.
- Boucles bornées et non bornées
- Travail sur les listes (1ére et terminales)

La liste des algorithme attendus par niveau est la suivante :

I. EN SECONDE

A. *Arithmétique*

- Déterminer par balayage un encadrement de $\sqrt{2}$ d'amplitude inférieure ou égale à 10^{-n}
- Déterminer si un entier naturel a est multiple d'un entier naturel b
- Pour des entiers a et b donnés, déterminer le plus grand multiple de a inférieur ou égal à b
- Déterminer si un entier naturel est premier.
- Déterminer la première puissance d'un nombre positif donné supérieure ou inférieure à une valeur donnée.

B. *Géométrie analytique*

- Étudier l'alignement de trois points dans le plan.
- Déterminer une équation de droite passant par deux points donnés.

C. *Fonctions*

- Pour une fonction dont le tableau de variations est donné, algorithmes d'approximation numérique d'un extremum (balayage, dichotomie).
- Algorithme de calcul approché de longueur d'une portion de courbe représentative de fonction.

Une fonction distance de deux points est donc souhaitable

D. *Statistiques*

Connaissances

Fonctions à un ou plusieurs arguments.

- Fonction renvoyant un nombre aléatoire. Série statistique obtenue par la répétition de l'appel d'une telle fonction.

Capacités associées

- Écrire des fonctions simples ; lire, comprendre, modifier, compléter des fonctions plus complexes. Appeler une fonction.
- Lire et comprendre une fonction renvoyant une moyenne, un écart type. Aucune connaissance sur les listes n'est exigée.
- Écrire des fonctions renvoyant le résultat numérique d'une expérience aléatoire, d'une répétition d'expériences aléatoires indépendantes.

Penser aux pourcentages

E. Compétences attendues

- la notion de fonction ;
- la programmation comme production d'un texte dans un langage informatique.

- décrire des algorithmes en langage naturel ou dans un langage de programmation ;
- en réaliser quelques-uns à l'aide d'un programme simple écrit dans un langage de programmation textuel ;
- interpréter, compléter ou modifier des algorithmes plus complexes.

II. EN PREMIERE

A. Suites

L'étude des suites est l'occasion d'une sensibilisation à l'idée de limite. Toute formalisation est exclue, mais sur des exemples, on s'attachera à en développer une intuition en s'appuyant sur des calculs numériques, des algorithmes de recherche de seuil.

- Exemples de modes de génération d'une suite : explicite $u_n = f(n)$, par une relation de récurrence $u_{n+1} = f(u_n)$, par un algorithme, par des motifs géométriques. Notations : $u(n)$,

- Calculer des termes d'une suite définie explicitement, par récurrence ou par un algorithme.

Exemples d'algorithme

- Calcul de termes d'une suite, de sommes de termes, de seuil.
- Calcul de factorielle.
- Liste des premiers termes d'une suite : suites de Syracuse, suite de Fibonacci.

B. Fonctions

Exemple d'algorithme

- Méthode de Héron pour l'approximation numérique des racines en utilisant la forme canonique.

Exemple d'algorithme

- Écrire la liste des coefficients directeurs des sécantes pour un pas donné.

Exemple d'algorithme

- Méthode de Newton, en se limitant à des cas favorables.

Pour trouver des racines

Exemple d'algorithme

- Construction de l'exponentielle par la méthode d'Euler. Détermination d'une valeur approchée de e à l'aide de la suite $\left(\left(1+\frac{1}{n}\right)^n\right)$.

- Approximation de π par la méthode d'Archimède.

- Méthode de Monte Carlo : estimation de l'aire sous la parabole, estimation du nombre π.

C. Statistiques

- Algorithme renvoyant l'espérance, la variance ou l'écart type d'une variable aléatoire.
- Fréquence d'apparition des lettres d'un texte donné, en français, en anglais.

III. EN TERMINALE

En maths expertes

– Algorithme d'Euclide de calcul du PGCD de deux nombres et calcul d'un couple de Bézout.

– Crible d'Ératosthène.

– Décomposition en facteurs premiers.

En maths de complément

Modèles d'évolution

- Calcul des termes d'une suite.
- Recherche de seuils.
- Méthode d'Euler

Approche historique de la fonction logarithme

- Algorithme de Briggs.
- Approximation de ln(*2)* par dichotomie selon l'algorithme de Brouncker.

Calculs d'aires

- Calcul d'un terme de rang donné d'une suite.
- Recherche d'une valeur approchée de précision donnée.

Répétition d'expériences indépendantes, échantillonnage

- Dans le cadre de la loi binomiale : calcul de coefficients binomiaux (triangle de Pascal), de probabilités ; détermination d'un intervalle I pour lequel la probabilité P(X ∈ I) est inférieure à une valeur donnée α, ou supérieure à 1 - α.
- Simulation avec Python d'une variable aléatoire (de la loi de Bernoulli, d'une loi uniforme discrète, etc.) d'un échantillon de taille n d'une variable aléatoire. Fonction Python renvoyant une moyenne pour un échantillon. Série des moyennes pour *N* échantillons de taille n d'une variable aléatoire d'espérance μ et d'écart type σ. Calcul de l'écart type σ de la série des moyennes des échantillons observés, à comparer à $\sigma / \sqrt{n}$. Calcul de la proportion des cas où l'écart entre la moyenne m et μ est $k\sigma / \sqrt{n}$ inférieur ou égal à ou à ks, pour k = 2 ou k = 3.

Temps d'attente

- Simulation d'une variable aléatoire de loi géométrique à partir du schéma de Bernoulli.
- Simulation d'une loi exponentielle à partir d'une loi uniforme.
- Demi-vie d'un échantillon de grande taille d'atomes radioactifs.

Suites numériques, modèles discrets

- Recherche de seuils.
- Pour une suite récurrente $u_{n+1} = f(u_n)$, calcul des termes successifs.
- Recherche de valeurs approchées de constantes mathématiques, par exemple π, ln2, $\sqrt{2}$.

Fonctions : continuité, dérivabilité, limites, représentation graphique

- Méthodes de recherche de valeurs approchées d'une solution d'équation du type $f(x)=k$ *:* balayage, dichotomie, méthode de Newton.
- Algorithme de Briggs pour le calcul de logarithmes.
- Sur des exemples, résolution approchée d'une équation différentielle par la méthode d'Euler.

Intégration

- Méthode des rectangles, des trapèzes.
- Méthode de Monte-Carlo pour un calcul d'aire.

Lois discrètes

- Simulation d'une variable de Bernoulli ou d'un lancer de dé (ou d'une variable uniforme sur un ensemble fini) à partir d'une variable aléatoire de loi uniforme sur [*0,1*].
- Simulation du comportement de la somme de n variables aléatoires indépendantes et de même loi.

Spé math (sujets en plus)

Combinatoire et dénombrement

- Pour un entier n donné, génération de la liste des coefficients $\binom{k}{n}$ à l'aide de la relation de Pascal.
- Génération des permutations d'un ensemble fini, ou tirage aléatoire d'une permutation.
- Génération des parties à 2, 3 éléments d'un ensemble fini.

Succession d'épreuves indépendantes, schéma de Bernoulli

- Simulation de la planche de Galton.
- Problème de la surréservation. Étant donné une variable aléatoire binomiale X et un réel strictement positif α, détermination du plus petit entier k tel que $P(X > k) \leqslant \alpha$.
- Simulation d'un échantillon d'une variable aléatoire.

Concentration, loi des grands nombres

- Calculer la probabilité de $(|S_n - pn| > \sqrt{n})$, où S_n est une variable aléatoire qui suit une loi binomiale $\mathcal{B}(n,p)$. Comparer avec l'inégalité de Bienaymé-Tchebychev.
- Simulation d'une marche aléatoire.

Impressions personnelles sur l'algorithmique : Il est clairement intéressant de faire faire de l'algorithmique à nos élèves, comme nous l'avons déjà évoqué cela leur permet d'apprendre à structurer leur raisonnement. Cependant la liste d'algorithme officielle me parait un peu longue pour être faite au complet, y compris avec des élèves ayant un bon niveau. Et ce pour plusieurs raisons. Tout d'abord, au niveau du matériel, bien qu'une partie des activités puissent se réaliser en « débranché », il est quand même plus intéressant de pouvoir travailler sur ordinateur. Ce qui implique de passer une heure complète de cours sur ordinateur les salles info étant généralement peu nombreuses et peu adaptées à un cours classique. Mais aussi à cause du contenu, les programmes sont très chargés et il est difficile de terminer dans les temps. Donc amputer le programme de quelques algorithmes pour gagner un peu de temps me semble être un bon compromis.

Exemple de progression python :

Voici un exemple de progression sur la géométrie analytique qui reprend *des points du programme* et qui permet aux élèves de percevoir la notion de fonction :

Alignement et distances :

- Créer une fonction permettant de calculer la distance entre deux points.
- Créer un programme utilisant la fonction précédente qui détermine si *trois points sont alignés.*

Vecteurs :

- Créer un programme qui calcule les coordonnées d'un vecteur.
- Créer une fonction qui calcule le déterminant de deux vecteurs en faisant appel au programme précédent.
- Créer une fonction qui utilise le déterminant pour voir si deux vecteurs sont colinéaires.
- Créer un programme qui utilise les fonctions précédentes pour savoir si *des points sont alignés ou des droites parallèles.*

Nature d'un quadrilatère.

- Créer une fonction qui calcule les coordonnées du milieu d'un segment.
- Créer une fonction utilisant la précédente pour déterminer si un quadrilatère est un parallélogramme.
- Utiliser les fonctions précédentes pour déterminer la nature d'un quadrilatère.

Voici maintenant quelques exemples d'activités réalisables en classe.

Activité 1 : Géométrie et algorithmique. (Durée 2h)

G1 : géométrie dans le plan et algorithmique

A retenir

La notion de fonction :

En python on peut définir des fonctions. Ce sont des morceaux de programmes que l'on pourra réutiliser plus tard. Par exemple :

```
def carré(n)
    Return (n**2)
```

Ici on définit la fonction « carré », qui calcule le carré du nombre que la fonction recevra (et que l'on nomme argument)

Par exemple si on tape dans l'exécuteur :

carré(3) il retourne 9.

A toi de jouer !

Objectif : définir une fonction nommée distance qui a en argument les coordonnées de deux points (donc *xa,ya,xb,yb*). Cette fonction doit retourner la distance entre les deux points.

1) Ecrire la formule de calcul de la distance entre deux points A (x_A ; y_A) et B(x_B ; y_B)

A préparer la séance précédente.

2) Ecrire le programme que vous allez entrer sur l'ordinateur (Ecrire au crayon pour pouvoir effacer en cas de besoin). Pour information : $\sqrt{x}$ s'écrit sqrt(*x*) (sqrt pour SQuare RooT, littéralement racine carrée en anglais)

Il est possible de différentier ici en proposant une version à compléter du programme, ou plusieurs programmes complets et laisser l'élève choisir le bon

3) Testez votre programme avec A (1 ; 2) et B (4 ; 6) quel résultat obtenez-vous ? (Appelez l'enseignant pour vérifier votre réponse)

Avoir fait l'exercice la séance précédente pour pouvoir comparer les résultats.

A retenir :

Réutiliser les fonctions :

Une fois une fonction définie, elle peut être utilisée dans une autre partie du programme.

Si on reprend notre fonction « carré », et que l'on souhaite écrire un programme qui additionne le carré de deux nombres le programme ressemblerait à ça.

```
def somme() :
n1=float(input('Entrez votre premier nombre : '))
n1=float(input('Entrez votre second nombre : '))
S= carré(nombre1) + carré(nombre2)
print(S)
```

Si on tape dans l'exécuteur la commande somme(). L'ordinateur demandera d'entrer un premier puis un second nombre. Ensuite il calculera le carré du premier nombre, et l'additionnera au carré du second nombre.

A toi de jouer !

Objectif : créer une fonction qui détermine si trois points sont alignés ou non.

1) Complétez la phrase suivante en évoquant les distances :

Trois points A, B et C sont alignés dans cet ordre si et seulement si ….

2) Complétez le programme suivant pour qu'il calcule les distances entre chacun des points et qu'il détermine si les points sont alignés.

```
def alignement() :
    xa=float(input(« xa =  »)
    ya=float(input(« ya =  »)
    xb=float(input(« xb =  »)
    ……..
    ……..
    ……..
    d1=distance(xa,ya,xb,yb)
    d2=…….
    d3=……
    if d1+d2==d3 :
        print(« les points …….. alignés »)
    elif  ……. :
        ……
    elif ……..…… :
        ……
    else :
        ……
```

3) Entrez ce programme dans Edupython et testez le avec A(0 ; 0) B(2 ; 1) et C (6 ; 3). Ecrivez la réponse fournie par le programme et appelez l'enseignant pour continuer.

4) Objectif ++ : Définir une fonction qui calcule les coordonnées du milieu d'un segment et utiliser cette fonction pour déterminer si un quadrilatère est un parallélogramme.

Cette question est facultative et ne sera traitée que par les élèves les plus rapides, qui ne recevront pas d'aide pour traiter le problème. Cela permet de différentier par un objectif plus poussé. Cette question sera traitée en DM, les élèves devront écrire le programme ce qui permettra d'évaluer s'ils ont acquis la notion de fonction. Une fois le DM corrigé une séance en salle info permettra de tester son programme sur ordinateur avec différentes valeurs. Les élèves auront alors la possibilité de corriger leurs programmes. Ils enregistreront et enverrons leurs programmes afin que je puisse à nouveau les évaluer.

Analyse activité :

Pré requis : Avoir déjà utilisé python (déclarer et entrer une variable, connaitre la notion de test)

Objectifs et compétences travaillées :

- Introduire la notion de fonction
- Raisonnement par étape
- Voir dans un contexte différent un problème d'alignement
- Savoir écrire et tester un programme
- Savoir compléter un programme.

Evaluation des acquis : cf. remarques sur la dernière question.

Activité 2 : Algo et suites pour recherche de seuil (Niveau 1ére)

Soit le programme suivant :

```
def u(n) :
    return (-n*n+21*n+75)

def seuil() :
    n = 0
    while u(n)>u(n-1) :
        n = n+1
    print("Le premier entier 'n', tel que u(n) > 0 est : ",n)
```

1) Que renvoie la fonction u(n) si n = 24 et si n = 25.
2) Donnez l'expression de (u_n) en fonction de n.
3) Que renvoie la fonction seuil si elle est exécutée ?
4) Proposez une modification du programme qui permette à l'utilisateur de choisir le seuil.
5) Proposez une modification de l'algorithme pour qu'il donne la valeur maximale de cette suite ainsi que son rang.

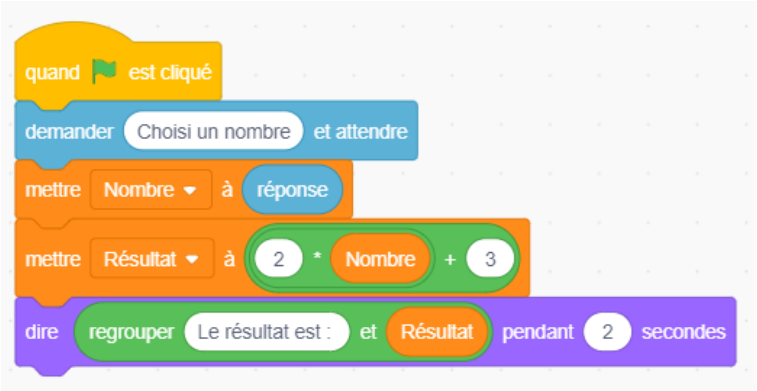
Analyse de l'activité

Cette activité est pensée pour être faite en débranché, c'est un petit exercice qui ne devrait pas prendre plus de 15 min, il permet de travailler les compétences de compréhension et de modification d'un algorithme. La cinquième question est surtout présente pour gérer la différentiation. L'objectif est que tous les élèves puissent répondre à la question 4 qui permettra de vérifier qu'ils comprennent le fonctionnement de l'algorithme. On peut même demander aux élèves les plus rapides de programmer l'algorithme sur leurs calculatrices ou sur l'ordinateur de la classe, afin de pouvoir utiliser l'algorithme pour faire la correction.

Activité 3 : Passage de scratch à python.

A la découverte de Python :

Voici un algorithme programmé à l'aide de scratch.

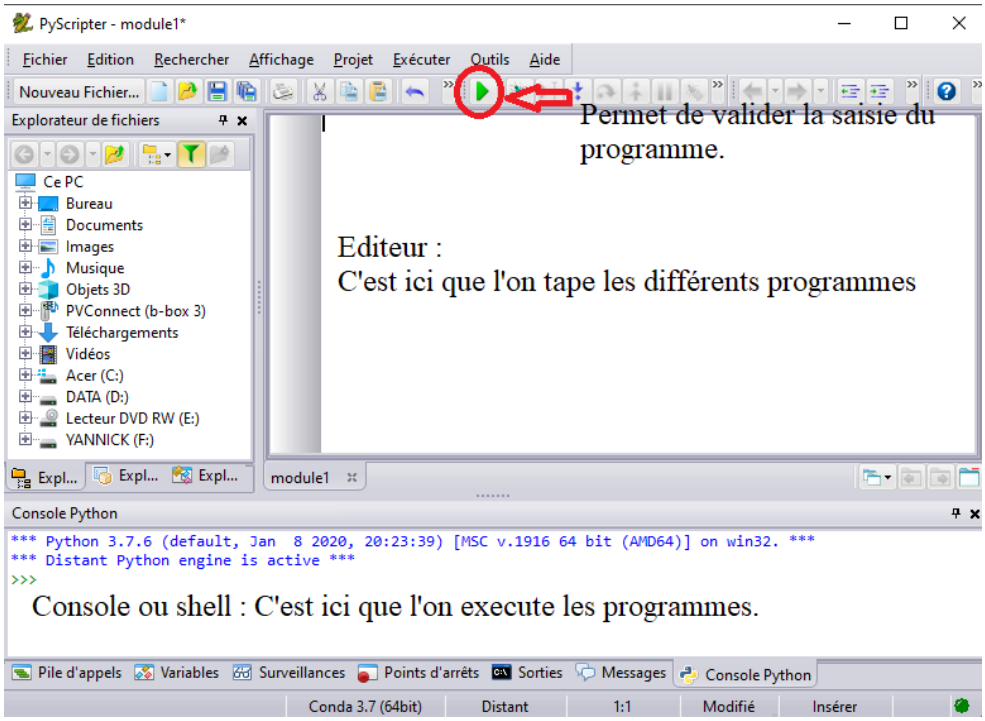


1) Quel résultat obtenez-vous si vous répondez 5, si vous répondez 3, si vous répondez -6.

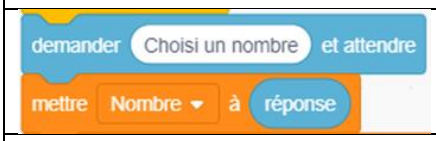
L'objectif est de s'assurer que tout le monde comprenne bien l'algorithme, on laisse quelques minutes aux élèves le temps de lire l'énoncé, ensuite les questions 1 et 2 sont corrigées au tableau. On pourra écrire les nombres utilisés dans une couleur différente lors de la correction de la question 1 pour faciliter la compréhension de la question 2. En effet si il sera assez facile pour les élèves de savoir appliquer l'algorithme, le généraliser avec une formule littéral est loin d'être évident.

2) Si on nomme N le nombre choisi et R le résultat, exprimez R en fonction de N. (Ce qui signifie, donnez le calcul qui permet de trouver la valeur de R à partir de N)

Lancer le programme édupython, vous obtenez ceci.



Le programme précédent se traduit de la manière suivante en python :

	`def Algo1() :`	En scratch c'est la condition pour exécuter le script en python, c'est le nom du programme.
	`N=int(input("N = "))`	On demande d'entrer (input) un nombre entier (int). Et on écris un message à l'écran (N =)
	`R=2*N+3`	R prend la valeur 2*N + 3. Le signe = n'a pas le même sens qu'en mathématiques il signifie juste que l'on « range le résultat du calcule dans la variable ».
	`print("Le résultat est : ",R)`	print demande l'affichage et le message à afficher est entre guillemets.

Chaque instruction est séparée du bord par un espace nommé indentation.

A toi de jouer !

Faire un nouvel algorithme nommé Algo 2 qui calcule le triple du nombre choisi et qui lui soustrait 5 et qui affiche le résultat.

Tester l'algorithme avec 10.

Le but de cette question est de vérifier que les élèves ont bien acquis les connaissances.

Faire un nouvel algorithme nommé Algo 3 qui additionne le carré de deux nombres A et B. *(objectif facultatif pour gérer la différentiation.)*

Activité 3 : Python et calculatrice.

2D **Algorithmique**

Exercice 1 :
Créer un programme python sur la calculatrice nommé Exo1. Pour cela :
Appuyer sur :



Sélectionner 2 : Python App
Appuyer sur :



Pour créer un nouveau programme. Et entrer le nom **Exo1**. Valider la création en appuyant sur la touche graphe.

1) Nous allons maintenant créer une première fonction que l'on nommera **algo1**. Cette fonction prend un nombre au choix, le multiplie par 2, lui ajoute 1 et le met au carré.

Listes des touches à utiliser :



: Permet d'écrire le def ou le return



: Permet d'écrire le =



: Permet d'écrire les lettres (en vert sur le clavier)

Voici à quoi devrai ressembler votre écran à la fin :

```
def algo1(a):
  a=2*a
  a=a+1
  a=a*a
  return (a)_
```

Le a entre parenthèses a côté du nom de programme signifie que la valeur a se choisie par l'utilisateur au moment ou il exécutera le programme.

Pour tester le programme il faut appuyer sur 

. Un nouvel écran s'ouvre. Il faut taper algo1(1) :

```
>>> # Shell Reinitialized
>>> # L'exécution de EXO1
>>> from EXO1 import *
>>> algo1(1)
```

Le algo1, fait appel à la fonction que nous avons écrite et nommée algo1.
Le chiffre entre parenthèses est la valeur que l'on choisit pour a.
Donc ici la calculatrice exécute le programme avec a = 1 au départ.

Pour lancer l'exécution appuyez sur enter. La calculatrice affichera le résultat 9 car.
2 x 1 = 2
2 + 1 = 3
3 x 3 = 9
Tester ce programme avec comme valeur initiale pour a : 3 puis 5.

Résultat obtenu pour 3 :
Résultat obtenu pour 5 :

2) Appuyer sur la touche trace pour retourner sur l'écran d'édition des fonctions. Créer alors une fonction nommée algo2 qui multiplie par 3 le nombre choisi et lui ajoute 5. Tester cet algorithme avec comme valeur initiale 3 puis 5.
Résultat obtenu pour 3 :
Résultat obtenu pour 5 :

Appuyer sur 2nd puis mode pour quitter l'éditeur. Confirmer avec graphe.

Exercice 2 : Découverte des boucles.

1) Créer un nouveau programme nommé Exo2. Dans ce programme créer la fonction algo 3. Le but de cette fonction est de multiplier un nombre a choisi par l'utilisateur du programme par 2 autant de fois que souhaité par l'utilisateur.

Liste des touches utiles :

f(x) : Cette touche permet d'accéder aussi aux différentes fonctions via les touches  de la calculatrice.

Voici à quoi devrait ressembler votre programme :

```
def algo3(a,b):
  for i in range(0,b,1):
    a=a*2
    print(a)
    print(i)
  return (a)
```

a) Expliquez ce quel effet on les lignes print(a) et print(i)

Une traduction de la ligne for i in range(0,b,1) serait pour i compris entre 0 et b, par pas de 1 exécuter les instructions ci-dessous. Pour mieux comprendre ce qui se passe dans le programme voici un détail des calculs effectués, si on choisit au départ a = 1 et b = 3.
Etape1: i = 0 or 0 < 3 donc on calcule a x 2 la nouvelle valeur de a est donc 2. On affiche a puis i. On ajoute alors 1 à i.
Etape2: i = 1 or 1 < 3 donc on calcule a x 2 la nouvelle valeur de a est donc 4. On affiche a puis i. On ajoute alors 1 à i.
Etape3: i = 2 or 2 < 3 donc on calcule a x 2 la nouvelle valeur de a est donc 8. On affiche a puis i. On ajoute alors 1 à i.
Etape4: i = 3 ce qui n'est pas strictement inférieur à 3, donc le programme s'arrête et affiche la dernière valeur de a.

b) Prédire le résultat final obtenu si on exécute le programme : algo3(6,3). Puis vérifier le résultat obtenu.
Résultat obtenu pour algo3(6,3) :
Testez a nouveau ce programme mais avec a = 2 et b = 4, résultat obtenu :

2) Créer une nouvelle fonction nommée algo4. Le but de cette fonction est d'ajouter 3 à un nombre choisi par l'utilisateur jusqu'à ce que la valeur de ce nombre dépasse 10, la fonction affiche alors le nombre de fois ou elle a dû ajouter 3.

Votre écran devrait ressembler à ça :

```
def algo4(a):
  n=0
  while a<10:
    a=a+3
    n=n+1
  return (n)
```

Voici ce qui ce passe si l'on exécute algo4(2) :
n prend la valeur 0.
Etape 1 : a = 2 or 2 < 10 donc on calcule a + 3 = 5 et n + 1 = 1.
Etape 2 : a = 5 or 5 < 10 donc on calcule a + 3 = 8 et n + 1 = 2.
Etape 3 : a = 8 or 8 < 10 donc on calcule a + 3 = 11 et n + 1 = 3.
Etape 4 : a = 11 or 11 > 10 donc on stoppe la boucle et on affiche n donc ici 3.
(Pour info : while signifie tant que)

a) Prédire le résultat qui va s'afficher pour a = 1 et le tester vérifier votre résultat avec le programme.

b) Prédire le résultat qui va s'afficher si on tape algo4(-2), puis vérifier avec le programme.

3) Ecrivez un programme qui divise par 2 un nombre a choisi par l'utilisateur autant de fois que nécessaire pour que le résultat soit inférieur ou égal à 1. Et qui affiche en sortie le nombre de fois où le programme a effectué la division. Nous nommerons ce programme Algo5.
   a) Tester ce programme en choisissant 20 comme valeur. Résultat obtenu :
   b) Prédire quel serait le résultat si on tape Algo5(11) puis tester avec votre programme :

Exercice 3 : Choisir des valeurs.

1) Créez le programme suivant :

```
def algo6():
    print("a=")
    a=int(input())+3
    return (a)
```

Ensuite exécutez le en tapant algo6(). a = s'affiche mais rien d'autre ne se passe. Tapez 5 et appuyez sur entrée. Vous obtenez :

```
>>> algo6()
a=
5
8
```

Expliquez l'effet de la ligne a=int(input()).

Pour info en anglais le mot pour désigner un nombre entier est integer. Et le mot pour une entrée est input.

2) A l'aide des connaissances que vous venez d'acquérir, créez un programme qui permet à l'utilisateur de choisir un nombre de valeurs. Puis de les entrer une à une afin d'en calculer la moyenne.

A vous de jouer :

Concevez une activité que vous pourriez donner en classe à vos élèves (niveau au choix) utilisant python. Vous réaliserez le document qui serait fourni à la classe pour cette activité.