Séquence 1 : Algorithme & Programmation en Python

Historique

Objectifs de la séquence :

- Savoir la différence entre un algorithme et un programme en python
- Variables/Affectation
- Entrées/sorties
- Fonctions
- Bibliothèques
- Instructions conditionnelles
- Boucles bornées et non bornées

I. Algorithme ou langage naturel - programme

Définition : Un **algorithme** est un énoncé d'une suite d'instructions **finie** permettant de donner la réponse à un problème. On utilise des mots en français et peu de symboles.

Voici un algorithme donné en français vu en $3^{ème}$

L1	Choisir un nombre x
L2	Le multiplier par 5
L3	Lui ajouter 3
L4	L'élever au carré
L5	Afficher la valeur de x obtenue

A l'aide de ce programme de calcul compléter le tableau ci-dessous pour les valeurs de x donné

x (L1)	x (L2)	x (L3)	x (L4)	x (L5)
4				
0				
-1				

Voici la correspondance en algorithme et le compléter

L1	Saisir x
L2	$x \leftarrow 5x$
L3	
L4	
L5	Afficher x

Définition : Un **programme** est un texte qui décrit un algorithme que l'on souhaite faire exécuter par une machine. Ce texte est écrit dans un langage informatique. Le langage utilisé au lycée est **Python**

II. Python : installation, écriture et exécution

Python est distribué sous licence libre et présente une syntaxe épurée et simplifiée. Il est **multiplateforme** ce qui permet de l'installer sur ordinateur (version 3.7) mais il existe aussi sur calculatrice (Numworks, casio ou TI) ou sur des éditeurs en ligne (edupython, repl.it ….)

2.1 Installation sur PC :

Python téléchargeable depuis le site internet dont voici l'adresse : https://www.python.org/downloads/

Nous utiliserons la **version 3.7.1** qui est sortie (octobre 2018)

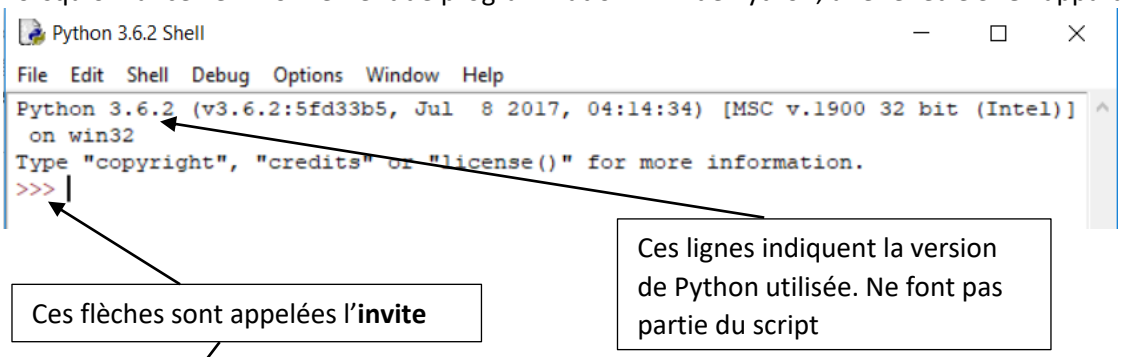
Il faut cependant penser à télécharger les **bibliothèques** pour avoir accès aux fonctions mathématiques, tracer des représentations graphiques…. (Pil, Numpy, Pyparsing, Pillow, matplotlib, Pygame, tkinter, turtle)

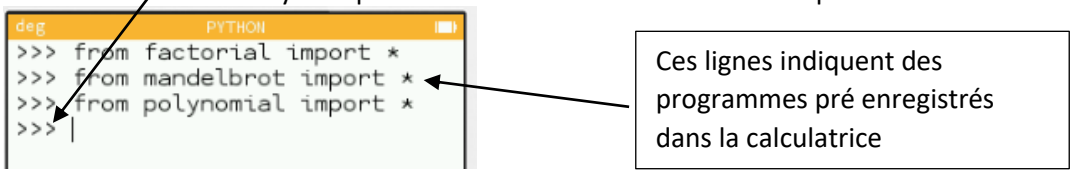
2.2 Où écrire le programme ? Comment l'exécuter ?

Après installation du logiciel, vous avez plusieurs exécutables qui s'installent notamment **IDLE (integrated development environment)**
IDLE est un programme qui aide à écrire, modifier et sauvegarder des scripts. Tous les utilisateurs de Python n'utilisent pas IDLE, certains préfèrent des plateformes
Ici nous utiliserons **IDLE** ou la calculatrice **Numworks**

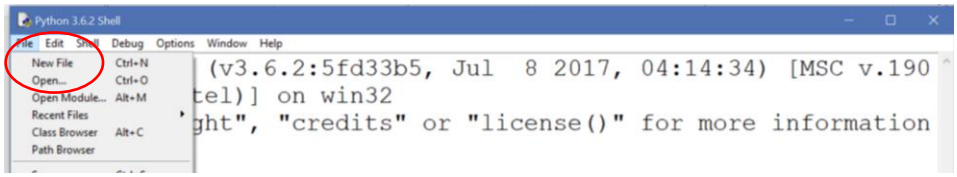
<table>
<tr>
<td>L'interpréteur Shell

Permet de tester de courtes portions de code et de les exécuter au fur et à mesure que l'on tape.

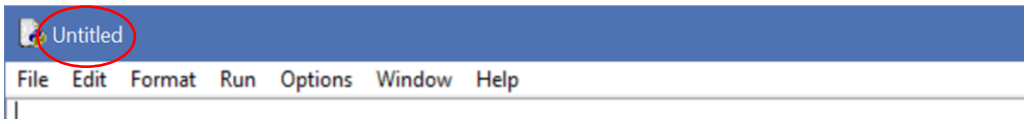
Inconvénient : il ne permet pas de sauvegarder ni d'exécuter un programme entier.</td>
<td>Ordinateur :
Lorsqu'on lance l'environnement de programmation IDLE de Python, une fenêtre Shell apparaît.
 

Numworks :
Il faut lancer le Menu Python puis sélectionner console d'exécution pour obtenir le Shell
 
</td>
</tr>
<tr>
<td>Script

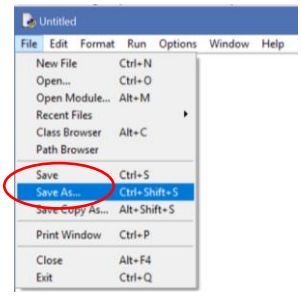
Pour exécuter un programme plus long et que l'on veut modifier plus tard</td>
<td>Ordinateur :
Il faut utiliser ce qui s'appelle la fenêtre de l'éditeur.

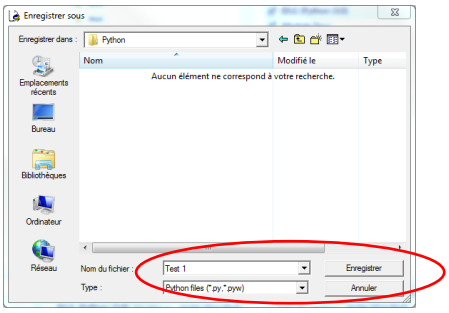
On peut ouvrir une fenêtre d'édition avec la commande « New file » dans le menu « file »
 
</td>
</tr>
</table>

Une nouvelle fenêtre s'ouvre : « **Untitled** ».



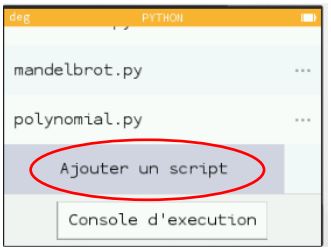
Puis l'enregistrer au format « **.py** » en lui donnant un nom.

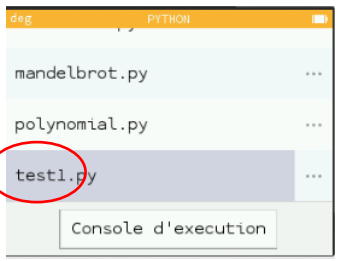




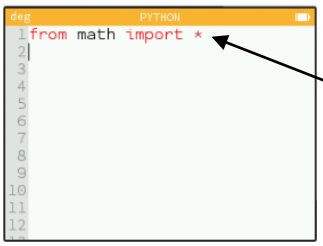
Numworks :

Il faut lancer le Menu Python puis sélectionner « **ajouter un script** » qui se trouve au bas de la liste des programmes. Validez en appuyant sur la touche **O** . Un nouveau script apparaît dans la liste. Vous pouvez à ce moment-là entrer un nom pour ce script





Pour écrire dans un script, il vous suffit de placer la sélection sur le nom de ce script et d'appuyer sur **O**. L'éditeur s'ouvre et vous pouvez écrire vos programmes à l'intérieur.

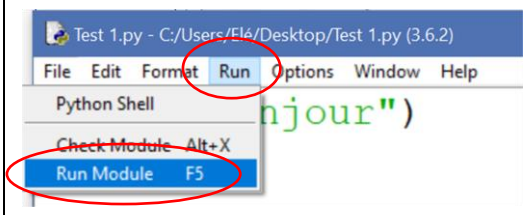


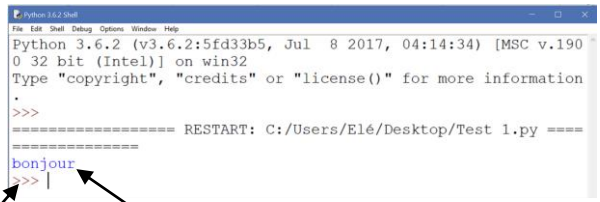
Ces lignes indiquent l'importation d'une bibliothèque

Execution du script

Ordinateur :

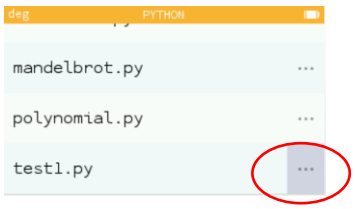
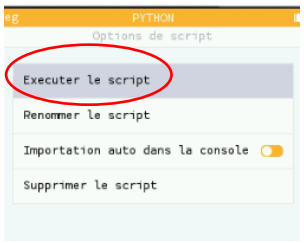
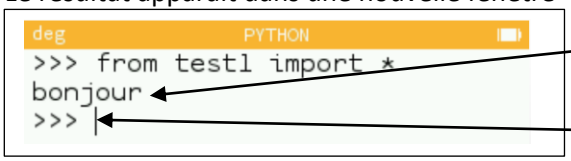
Pour exécuter le programme, on utilise la commande « **Run module** » dans le menu « **Run** » (où l'on appuie sur F5). Le mot **module** désigne un fichier Python. Le résultat apparait à l'écran dans la fenêtre « **shell** »





Une fois le programme exécuter l'invite réapparait

Résultat affiché

	Numworks : Pour exécuter le programme, il vous suffit de placer la sélection sur le nom de ce script et de déplacer le curseur vers la droite (à l'écran « … ») et d'appuyer sur la touche **O** Un nouveau menu apparait, sélectionner « **exécuter le script** » et **O**   Le résultat apparait dans une nouvelle fenêtre  Résultat affiché Une fois le programme exécuter l'invite réapparait

2.3 Débogage

Si on reçoit un message d'erreur (en anglais) lors de l'exécution du programme, il faut vérifier que l'on a correctement écrit le code. Parmi les erreurs les plus courantes on a :

- Les guillemets
- Les parenthèses
- L'orthographe
- Majuscules et minuscules

2.4 Commentaires

Un commentaire est un texte, écrit dans le corps du programme et ignoré lors de l'exécution du programme. Un commentaire aide à la bonne compréhension du programme.

Instructions	Langage naturel	Python
Commentaires	//	#

III. Variables et Affectations

3.1 Notion de Variable & type de variables

Une **variable** est une zone de mémoire dans l'ordinateur repérée par son **nom**. Celle-ci peut changer de valeur au cours de l'exécution.
On peut donner n'importe quel ***nom à une variable*** à partir du moment où il n'y a pas d'espaces, ni certains caractères spéciaux, ni certains mots-clés désignant quelque chose en python.

Le type d'une variable identifie la nature de son contenu : nombre entier, nombre décimal, texte…

- « **int** » : nombres entiers
- « **float** » : nombres décimaux
- « **eval** » : les nombres réels
- « **str** » : chaine de caractères (une chaine de caractère sera toujours encadrée par des guillemets ou des apostrophes)
- « **bool** » : variable booléenne (Vrai ou faux)
- « **list** » : liste ou tableau d'éléments qui peuvent être de type différents Les listes peuvent être modifié au cours du programme.
  Syntaxe : nom_List=[…, …, …]
- « **tuple** » : regroupement de valeurs. On ne peut pas modifier sa valeur au cours du programme
  Syntaxe : nom_tuple=(…, …, …)
- ……

3.2 Création et affectation

L'affectation consiste à donner une valeur à la variable. Lorsqu'on fait cette affectation, en python il n'est pas nécessaire de définir le type de variable. L'affectation peut être imposé ou donné par l'utilisateur.

Instructions	Langage naturel	Python
Affecter 2 à la variable x	$x \leftarrow 2$	$x = 2$
Affecter à la variable y la variable x	$y \leftarrow x$	$y = x$
Donner une valeur à x	Saisir x	x =eval(input(«donner une valeur à x »)) #pour saisir un nombre réel x =input(«donner une valeur à x ») #pour saisir une chaine de caractère
Affichage variable x **Affichage texte** **Affichage texte et variable** x	Afficher x Afficher « … »	print (x) print (« … ») print (« … », x)
Retour à la ligne		« \n »

Remarques :

- **la fonction input()**

Elle permet une interaction avec l'utilisateur : elle va stopper l'exécution du programme et attendre que l'utilisateur saisisse une donnée. Le programme reprend lorsque l'utilisateur appuie sur la touche entrée.

- **La fonction print()**

Elle sert à afficher du texte et/ou le contenu d'une variable

3.3 Opérations

Les opérations avec les variables sont courantes : additionner, multiplier
Selon le type de variable, certaines opérations seront impossibles.

Instructions	Langage naturel	Python
Opérations de base : addition, multiplication, soustraction, division	+ ; × ; − ; ÷	+ ; * ; − ; /
Opérateurs mathématiques Puissance Division entière Reste de la division entière	 ∧ Div Mod	 ** // %
Opérateurs de comparaison : Inférieure stricte Inférieure ou égale Supérieure strict Supérieure ou égale	 < ≤ > ≥	 < <= > >=
Outils de comparaison Egal Différent	 = ≠	 == !=
Opérateurs logiques Et logique Ou logique	 ∧ ∨	 and or

3.4 Exercices

(1) Dialoguer avec son ordinateur
Ecrire un programme permettant à l'ordinateur de :

- Demander votre prénom
- Demander votre âge
- D'afficher en une seule phrase votre prénom et votre âge
- D'afficher grâce à votre âge votre année de naissance

(2) Dans chaque code ci-dessous, on définit deux variables, a et b, puis on fait une opération avec ces deux variables.

a="bonjour" b=3 print(a+b)	a="bonjour" b=3 print(a*b)	a=5 b=12.2 print(a+b)
a=['lun','mar','mer'] b=['jeu'] print(a+b)	a=['lun','mar','mer'] b='jeu' print(a+b)	a=51.3 b=4 print(a//b)
a=51 b=4 print(a%b)	a=['lun','mar','mer'] b=3 print(a*b)	a="bonjour" b="à tous!" print(a+b)

a) Identifier le type de variables a et b de chaque code.
b) Déterminer si les opérations proposées sont réalisables ou si elles génèrent une erreur. Si elles sont réalisables, quel est le type de résultat ?

(3) Quelles variables renvoient les codes suivants ? les tester.

a='bonjour' b=3 print(a==b)	a='bonjour' b=3 print(a!=b)	a='bonjour' b='bonjour' print(a==b)	a=3.0 b=3 print(a==b)

IV. Les fonctions

4.1 Notion de fonction

Une fonction est **un morceau de programme autonome** qui réalise une tâche précise et bien définie.

Une fonction est définie par :

- Un **nom** (choisir un nom qui indique clairement ce que fait la fonction)
- Ses arguments ou **paramètres** qui porteront les valeurs communiquées par le programme principal à la fonction au moment de son appel et seront écrit dans les parenthèses.
- Eventuellement une **valeur de retour** communiquée au programme par la fonction en fin d'exécution.

On peut utiliser une fonction dans le shell en donnant des valeurs aux paramètres dans le bon ordre

Remarques :

- Les variables utilisées dans une fonction sont **locales** et n'ont pas d'incidences sur le programme principal. Elles sont détruites dès que l'on sort de la fonction.
- Toutes les fonctions n'ont pas nécessairement besoin de paramètres
- On peut écrire une fonction2 qui fait appel à une fonction1 à certaines conditions :
  - La fonction1 doit être définie avant la fonction2
  - La fonction1 contient nécessairement une valeur de retour qui sera récupéré par la fonction2

  Pour appeler la fonction1 dans la fonction2, il suffit de stocker dans une variable la fonction1 avec ses paramètres.

4.2 Syntaxe

En langage naturel

```
Fonction nomfonction (paramètres) :
    #bloc d'instructions
    retourner résultat   # renvoie le résultat
Fin Fonction
```

En python

```
def nomfonction(parametres):
  #bloc des instructions
  return resultat
  #renvoie le resultat
```

Remarques :

- La programmation d'une fonction commence toujours par **def** suivie du **nom** de la fonction
- Suivi des **paramètres** de la fonction écrits entre parenthèses.
- Cette ligne finie toujours par « **:** »
  Les deux points marquent le démarrage du bloc d'instructions définissant la fonction. Toutes ses instructions sont indentées c'est-à-dire décaler vers la droite par rapport à la première ligne.
- Eventuellement une **valeur de retour** communiquée au programme par la fonction en fin d'exécution.
- Pour renvoyer la valeur de la variable on peut écrire **return …** ou **return(…)**
-

4.3 Exercices

(1) L'abonnement à un service de stockage de données en ligne pour une entreprise coûte 200€ d'ouverture de compte à l'inscription, puis 340€ par mois. On veut calculer le prix total payé par l'entreprise en fonction du nombre de mois d'abonnement.

a) Quel nom pourrait-on donner à cette fonction ?

b) Combien de paramètres a-ton besoin ? Nommez-les.

c) Que faut-il retourner ?

d) Le programmer en python.

(2) On souhaite écrire une fonction permettant de calculer l'aire d'un rectangle.

a) Quel nom pourrait-on donner à cette fonction ?

b) Combien de paramètres a-ton besoin ? Nommez-les.

c) Que faut-il retourner ?

d) Le programmer en python.

(3) On souhaite écrire une deuxième fonction permettant de calculer le volume d'un parallélépipède rectangle mais en faisant appelle à la fonction « aire_rectangle ».

a) Quel nom pourrait-on donner à cette fonction ?

b) Combien de paramètres a-ton besoin ? Nommez-les

c) Que faut-il retourner ?

d) Le programmer en python.

V. Les instructions conditionnelles

5.1 Définition :

On appelle « instructions conditionnelles » des lignes de code qui seront exécutées sous certaines conditions. Le programme fera un test et il effectuera certaines instructions suivant la réponse au test.

5.2 Syntaxe :

Pour aiguiller dans différentes directions l'exécution d'un programme, il est possible d'avoir recours à des « instructions conditionnelles » qui permettent de déterminer si les instructions qui suivent doivent être ou non exécutées.

Instructions	Langage naturel	Python

Instruction conditionnelle	Si **conditions** alors
	instructions 1
	Sinon
	instructions 2
	Fin Si
		`if conditions:`
		`  instructions 1`
		`else:`
		`  instructions 2`

	Si **conditions 1** alors
	instructions 1
	Sinon
	Si **conditions 2** alors
	instructions 2
	Sinon
	instructions 3
	Fin Si
	Fin Si
		`if conditions 1:`
		`  instructions 1`
		`elif conditions 2:`
		`  instructions 2`
		`else:`
		`  instructions 3`

Remarques :

- Le mot français **« Si »** du langage naturel se traduit par le mot anglais **« if »** en langage python. L'instruction se termine par **« : »** et un passage à la ligne avec un décalage appelé **« indentation »**
- Le mot français **« Alors »** du langage naturel ne se traduit pas par un mot anglais en langage python mais par un retour à la ligne et une indentation.

- Le mot français **« sinon »** du langage naturel se traduit par le mot anglais **« else »** en langage python. L'instruction se termine par **« : »** et un passage à la ligne.
- Le mot français **« Fin Si »** du langage naturel ne se traduit pas en anglais. En langage python il n'y a pas d'instruction pour indiquer la fin de l'instruction conditionnelle
- Le mot français **« Sinon Si »** du langage naturel se traduit par le mot anglais **« elif »** en langage python. L'instruction se termine par **« : »** et un passage à la ligne.
- Lorsque la situation à tester est plus compliquée, il est possible de combiner plusieurs conditions grâce aux opérateurs logiques : **and, or**
- On peut utiliser pour écrire des conditions, les opérateurs de comparaison : **> ; >= ; < ; <=** mais aussi « **in** » qui permet de parcourir un mot.

5.3 Exercices :

(1) On souhaite écrire un programme en python, qui demande à un utilisateur de saisir un mot, puis teste si au moins une fois le caractère « e » se trouve dans ce mot ou non.

(2) La loi interdit à un jeune de moins de 13 ans de s'inscrire sur les réseaux sociaux. Entre 13 et 15 ans, l'inscription nécessite l'autorisation d'un responsable légal. Au-delà de 15 ans, l'inscription est autorisée sans condition.
Ecrire une fonction en python, qui affiche les modalités d'inscription en fonction de l'âge de l'utilisateur.

(3) Le code PIN de votre smartphone est un entier stocké dans la variable « codepin »

a) Jean trouve un téléphone dans la rue. Il souhaite le déverrouiller mais ne connait pas le code PIN.
Ecrire un programme en python, permettant à Jean de saisir un code PIN pour essayer de déverrouiller le téléphone. Attention Jean n'a que 3 essais pour trouver le code PIN.
Après les 3 essais le programme affiche un message que le téléphone est bloqué.
On suppose que le codepin du smartphone est 989796

b) Que pourrait-on envisager s'il fallait le simplifier ?

VI. Les boucles bornées

6.1 Définition :

Il est parfois utile dans un programme de répéter une ou plusieurs instructions un nombre défini de fois. Lorsque le nombre de répétitions est connu à l'avance on utilise alors la boucle bornée « pour »

6.2 Syntaxe :

<table>
<tr><td>Instruction</td><td colspan="4"></td></tr>
<tr><td rowspan="4">Boucle bornée</td><td>Langage naturel</td><td colspan="3">Pour i allant de p à n
instructions
Fin Pour</td></tr>
<tr><td rowspan="3">Python</td><td>

```
for i in range(n+1):
    instructions
```

</td><td>La variable « i » prend les valeurs entières de 0 à n ($n \in \mathbb{N}$)</td><td>

```
for i in range(4):
    print(i)
```

</td><td>La variable i prend les valeurs entières de 0 à 3 donc 4 valeurs</td></tr>
<tr><td>

```
for i in range(p,n+1):
    instructions
```

</td><td>La variable « i » prend les valeurs entières de p à n ($n \in \mathbb{N}$ et $p \in \mathbb{N}$)</td><td>

```
for i in range(5,9):
    print(i)
```

</td><td>La variable i prend les valeurs entières de 5 à 8 donc 4 valeurs</td></tr>
<tr><td>

```
for i in range(p,n+1,k):
    instructions
```

</td><td>La variable « i » prend les valeurs entières de p à n avec un pas de k ($n \in \mathbb{N}$, $p \in \mathbb{N}$ et $k \in \mathbb{N}$)</td><td>

```
for i in range(3,10,2):
    print(i)
```

</td><td>La variable i prend les valeurs entières de 3 à 9 avec un pas de 2 donc 4 valeurs</td></tr>
</table>

La fonction **range ( )** crée une liste de nombres. Cette liste détermine combien de fois la boucle va s'exécuter.

Remarques :

- La variable i contrôle le nombre de répétitions. Ce compteur est aussi appelé « pas » ou « incrément »
- L'exécution des instructions ne s'effectue qu'entre les valeurs minimum et maximum données au départ.
- Le mot français **« pour »** du langage naturel se traduit par le mot anglais **« for »** en langage python.
  L'instruction se termine par **« : »** et un passage à la ligne avec une **« indentation »**
- Le mot français **« Fin Pour »** du langage naturel ne se traduit pas en anglais. En langage python, pour indiquer la fin de la boucle, on sort du bloc d'instruction.

6.3 Exercices :

(1) On a placé 5000€ sur un compte bancaire rémunéré à 2,5% chaque année. Compléter la fonction suivante pour qu'elle retourne la somme disponible au bout de 10 ans.

```
def compte():
    capital=.............
    for i in range (2,.....):
        capital=.............
    return capital
```

(2) Une voiture perd 10% de sa valeur chaque année. On cherche à connaitre son prix au bout de n années sachant qu'à l'achat elle coûtait 35 000€.
A l'aide d'une boucle bornée, écrire une fonction python permettant de connaître le prix de la voiture au bout de n années.

VII. Les boucles non bornées

7.1 Définition :

Lorsqu'on ne connait pas à l'avance le nombre de répétitions à effectuer, on utilise une boucle non bornée. A chaque tour de boucle, on teste une condition et cela permet de décider si on exécute à nouveau les instructions de la boucle ou si on sort de la boucle.

7.2 Syntaxe :

Instructions	Langage naturel	Python
Instruction boucle non bornée	Tant que condition(s) Instruction(s) Fin Tant que	`while condition(s):` `  instruction(s)`

Remarques :

- Le mot français **« tant que »** du langage naturel se traduit par le mot anglais **« while »** en langage python. L'instruction se termine par **« : »** et un passage à la ligne avec un décalage appelé **« indentation »**
- Le mot français **« Fin Tant que »** du langage naturel ne se traduit pas en anglais. En langage python, pour indiquer la fin de la boucle, on sort du bloc d'instruction.

7.3 Exercices :

(1) Une application de course à pied sur smartphone propose à l'utilisateur de rentrer les distances parcourues chaque jour. Lorsque l'utilisateur a atteint son objectif fixé de 45 km, le décompte s'arrête.
Ecrire un programme en python qui calcule la somme des distances parcourues tant que l'utilisateur n'a pas atteint son objectif, puis afficher le message « Félicitations »

(2) **Déverrouiller un smartphone – version 2**
On souhaite améliorer le programme du déverrouillage du téléphone fait avec la **condition si**.
Ecrire un programme en python utilisant la boucle non bornée, permettant à Jean de saisir un code PIN pour essayer de déverrouiller le téléphone. Attention Jean n'a que 3 essais pour trouver le code PIN. Le programme doit avertir l'utilisateur du nombre d'essai restant.
Après les 3 essais le programme affiche un message que le téléphone est bloqué.
On suppose que le code pin du smartphone est 989796

VIII. Les bibliothèques ou modules

8.1 Définition et syntaxe :

Sous python certaines notions mathématiques sont de base avec le langage (exemple : les commandes de calculs….). Cependant certaines sont regroupées dans une bibliothèque qu'il faut importer pour y avoir accès.

8.2 La bibliothèque « math » :

La bibliothèque « **math** » permet d'utiliser des fonctions mathématiques ainsi que certaines constantes.
Syntaxe : **from** math **import** *

Fonction	Effet
sqrt(x)	Renvoie $\sqrt{x}$
sin(x) ; **cos**(x) ; **tan**(x)	Fonctions trigonométriques
floor(x)	Renvoie la partie entière de x
abs(x)	Renvoie x si $x \geq 0$ et $-x$ sinon (appelé valeur absolue)
pi	Renvoie le nombre π
gcd (a,b)	Retourne le plus grand diviseur commun des entiers a et b

8.3 La bibliothèque « random » :

La bibliothèque « **random** » permet de générer des nombres aléatoires.
Syntaxe : **from** random **import** *

Fonction	Effet
randint(a,b)	Renvoie un **entier** choisi aléatoirement dans $[a;b]$
random ( )	Renvoie un flottant choisi dans l'intervalle $[0;1]$
uniform(a,b)	Renvoie un **flottant** choisi dans l'intervalle $[a;b]$

8.4 La bibliothèque « matplotlib » et « numpy » :

Les bibliothèques « **numpy** » et « **matplotlib.pyplot** » permettent de tracer et d'afficher des graphiques.

Les noms des bibliothèques étant trop long on leur donne un nom pour simplifier

Syntaxe pour importer ces bibliothèques :

- import matplotlib.pyplot as **plt**
- import numpy as **np**

Nom pour simplifie l'appellation des bibliothèques

as : est-ce qu'on appelle un alias, permet de renommer la bibliothèques

numpy as np	Effet
x = np.linspace (x_{min} , x_{max} , k)	Fait varier x entre x_{min} et x_{max} avec k valeurs

matplotlib.pyplot as plt	Effet
plt.axis ([x_{min} , x_{max} , y_{min} , y_{max}])	Détermine les dimensions des axes
plt.grid ()	Affiche une grille
plt.plot (x , y) plt.plot(x , y , « couleur * »)	Trace la courbe Trace un nuage de point et change la couleur de la courbe (1ère lettre du mot anglais)
plt.show()	Ouvre la fenêtre et affiche la courbe

8.5 La bibliothèque « pillow » :

La bibliothèque « **pillow** » permet d'utiliser des outils pour les images numériques : afficher, retoucher…

Syntaxe : **from** PIL **import** Image

Fonction	Effet
Nom_image=**Image.new**("…")	Création d'une nouvelle image
Nom_image=**Image.open**("…")	Charge une image en mémoire
Nom_image**.convert** (…)	Change le mode de l'image L : 256 nuances de gris / RGB : couleurs rouge, vert, bleue
Nom_image**.show**( )	Ouvre l'image dans un outil externe
Nom_image**.save**( )	Sauvegarde l'image dans le format spécifié

8.6 Exercices :

(1) Ecrire un programme en Python qui calcule l'aire d'un disque et la volume d'un cône.

(2) Ecrire un programme en Python qui simule le lancer de deux dés à 6 faces et qui affiche la somme des numéros des faces obtenue lors du lancer

(3) Ecrire un programme en Python permettant de convertir le logo de NDG en noir et blanc

(4) Ecrire un programme en Python qui permet d'afficher la représentation graphique d'une fonction affine suivant les valeurs de a et b données par l'utilisateur. On choisira de tracer cette courbe sur l'intervalle $[-10;10]$ et on choisira d'afficher 50 points.

Modifier le programme de façons à afficher un nuage de point rouge (Attention les couleurs sont en anglais)