Programmer

1. Qu'est-ce que programmer ?

Programmer, c'est expliquer à la machine ce qu'elle doit faire pour résoudre le problème d'un utilisateur

2. Les 3 intervenants

Le programmeur ne s'adresse pas directement à l'utilisateur.

3. Les 3 étapes à suivre

- Ecrire l'algorithme
- Traduire dans un langage de programmation
  (compiler)
- Exécuter

4. L'algorithme (mode d'emploi)

On peut comparer un algorithme à un mode d'emploi prenons l'exemple du tube de Pattex :

- **Nettoyer** les surfaces à encoller
- **Etendre** la colle uniformément
- **Laisser sêcher** 10 à 15 minutes
- **Assembler** les matériaux
- **Presser** fortement

C'est la décomposition du problème en tâches simples que l'ordinateur est capable d'effectuer.

Attention l'ordre des tâches a énormément d'importance

!! Cette étape ne nécessite de travailler sur un ordinateur !!

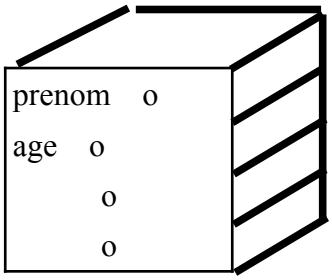
5. Que peut faire un ordinateur

5.1 demander au clavier une donnée (à homer) et la placer dans une variable

prenom = **input**("votre prenom svp = ") **str transforme en string une expression**

age = **int**(**input**("votre age svp =")) **int transforme en entier une expression**

qu'est-ce qu'une variable ?



On peut comparer une variable à un ..... d'une armoire.
Celui-ci porte une étiquette : le nom de la variable.

Sous Python, les noms de variables doivent en outre obéir à quelques règles simples :

• Un nom de variable est une séquence de lettres (a à z , A à Z) et de chiffres (0 à 9), qui doit toujours commencer par une lettre.

• Seules les lettres ordinaires sont autorisées. Les lettres accentuées, les cédilles, les espaces, les caractères spéciaux tels que $, #, @, etc. sont interdits, à l'exception du caractère _ (souligné).

• La casse est significative (les caractères majuscules et minuscules sont distingués). Attention : Joseph, joseph, JOSEPH sont donc des variables différentes. Soyez attentifs !

• N'utilisez pas de mots réservés (input, print, etc.)

Prenez l'habitude d'écrire l'essentiel des noms de variables en caractères minuscules (y compris la première lettre). Il s'agit d'une simple convention, mais elle est largement respectée. N'utilisez les majuscules qu'à l'intérieur même du nom, pour en augmenter éventuellement la lisibilité, comme dans tableDesMatieres, par exemple.

5.2 Initialiser une variable (imposer une valeur)

```
i = 0
mot="code"
```

5.3 calculer une expression

peri = (long + larg) * 2 opérateurs (* / + - a**2=> a^2) !!! PEMDAS !!!

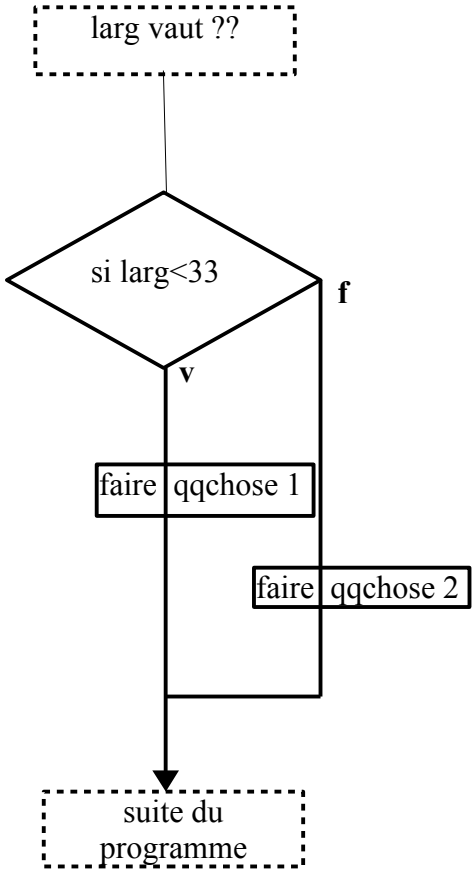
5.4 afficher une variable, un message, une variable et un message

print ("bonjour") =>
print (maxi) =>
print (f"le prix de cet ordinateur est de {prix} €") =>

5.5 Comparer si une variable est ..... par rapport à une valeur ou une autre variable. Si cela est vrai **alors** faire quelque chose1 **sinon** faire autre chose2

Opérateurs de comparaison :

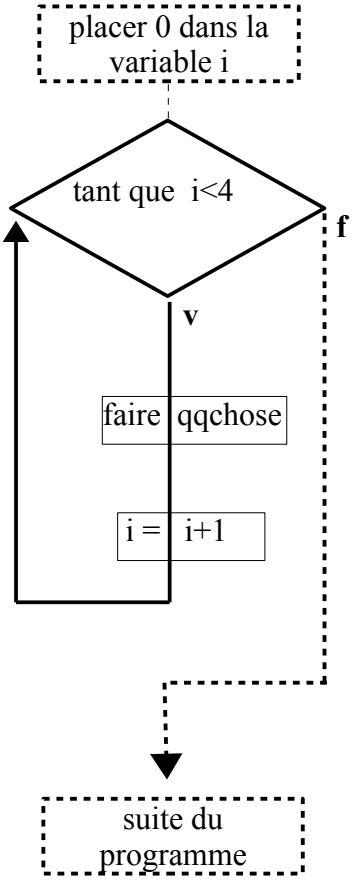
<	>
2 x = égal	!= différent
<=	>=



si la variable larg vaut .....alors la condition est ..... et l'ordinateur fait qqchose.....

```
if (larg < 33) :
        print(" larg est plus petit que 33")
else :
        print(" larg est plus grand que 33")
```

5.6 Répéter tant que



debuging

i	i<4	faire qqchose

```
i=0
while (i<4) :
    print("bonjour")
    i=i+1
print("fin")
```

5.7 Compter + 1

```
cpt = cpt + 1
```

5.8 Accumuler des valeurs dans une variable

```
accu = accu + x   x est une variable
```

Résumé des instructions 2022

1	Initialiser une variable (affecter)	`long= 0 mot="code"`

2	Lire une variable	`age=int(input("votre age svp"))`
		`prenom=input("votre prénom svp)`

3	Calculer	`surf = long * larg`

4	Afficher une variable	`print(surf)`
	Afficher un message	`print("surface")`
	Afficher message & variable	`print(f"la surface est de {surf} cm² ")`

5	Si …….. Alors ……	`if (surf < 50) :`
		`  ???????????`
		`else :`
		`  ???????????`

6	Répéter Tant que	`while (i<=17) :`
		`  ???????????`
		`  i=i+1`

7	Utiliser un compteur	`cpt = cpt + 1`

8	Utiliser un accumulateur	`accu = accu + nbr`

9	tirer un nb au hasard	`from random import randrange # debut de script`
		`nbr = randrange(1,11)`

10	nbr multiple de 3	`if (nbr % 3 == 0) :`

égal	==
différent	!=
exposant	2**3

6. Quelques scripts pour débuter

Créez un répertoire début et sauvez vos exercices sous ***exe1.py, exe2.py, ...***

1. Ecrire un script qui permet de calculer le volume d'une sphère 4/3 pi R^3
   pi vaut 3.141592 (1, 2, 3, 4)

2. Ecrire un script qui permet d'afficher les 10 premiers termes de la table de multiplication par 3. (1, 6, 4)

3. Ecrire un script qui permet d'afficher les 10 premiers termes de n'importe quelle table de multiplication, utilisez i dans votre multiplication. (1, 2, 6, 3, 4)

4. Ecrire un script qui permet d'afficher 55 nombres tirés au hasard et compris entre 0 et 100 (1, 6, 9, 4)

5. Ecrire un script qui permet à l'utilisateur d'introduire 5 nombres, d'en faire la somme et de l'afficher. Utilisez une boucle !! (1, 6, 2, 8, 4)
   utilise un accumulateur

6. Ecrire un script qui permet à l'utilisateur d'introduire un nombre et qui lui indique s'il est pair ou impair. Pas besoin de boucle
   % (modulo) représente le reste de la division (entier)
   ex ***7%2*** donne comme résultat **1** et **6%2** donne comme résultat **0**
   (1, 2, 5, 4)

7. Ecrire un script qui permet de tirer 200 nombres au hasard, compris entre 0 et 100 et qui n'affiche que les impairs. (1,6, 9, 5, 4)

8. Ecrire un script qui permet à l'utilisateur d'introduire des nombres et qui les additionne au fur et à mesure. Le script s'arrête lorsque l'utilisateur introduit un nombre négatif. A ce moment l'ordinateur affiche la somme. (1, 6, 2, 8, 4)

9. Ecrire un script qui permet de tirer 200 nombres au hasard, compris entre 0 et 100 et qui affiche ces nombres **s'ils ne sont pas** des multiples de 3. *(1, 6, 9, 5, 4)*

7. Devoir sur les boucles

Créez dans votre répertoire programmation un sous répertoire **dboucles2022**, dans lequel vous sauvegarderez vos programmes.

1. Etablir un programme qui permet de tirer 25 nombres au hasard compris entre 1 et 100. D'afficher uniquement les multiples de 9 et de les comptabiliser (comptabiliser = combien il y en a) **(mult9.py).** *(1, 6, 9, 5, 4, 7, 4)*

2. Ecrivez un programme qui permet de transformer une température en degrés celcius en degrés farenheit**.** Tf = (Tc * $\frac{9}{5}$) +32 **(celcius.py)** (1, 2, 3, 4)

   Ex.

   Introduisez la temperature (celcius) 37
   La temperature en farenheit est de **98.6**

3. Ecrivez un programme qui permet d'obtenir la figure suivante :
   `print("@"*3)` donne `@@@`

```
@@
@@@@
@@@@@@
@@@@@@@@
@@@@@@@@@@
```

   (arobas.py) (1, 6, 4)

4. Etablir un programme qui permet d'obtenir la figure reprise ci-dessous. Nous demanderons à l'utilisateur ((Homer) d'introduire la largeur et la hauteur de la figure **(figure.py)** (1, 2, 4, 6, 3, 1, 4, 4)

   Introduisez la largeur svp 12
   Introduisez la hauteur svp 5

```
XXXXXXXXXXXX
X          X
X (espaces)X   } Boucle => 5-2
X          X
XXXXXXXXXXXX
```

```
print(f"X{espace}X")
```

5. Etablir un programme qui tire des nombres aléatoirement compris entre 1 et 100, jusqu'à ce que le nombre 75 sorte. A ce moment, l'ordinateur indique le nombre de tirages qui ont été nécessaires pour obtenir ce 75. **(jusqua.py)** (1, 6, 9, 4, 7, 4)

6. Afficher la table de multiplication de 3 mais à l'envers (1, 6, 4)
   30
   27
   ....
   0

8. Les listes

Longueur d'une liste

```
jours=['lundi', 'mardi', 'mercredi', 'jeudi', 'vendredi']
n=len(jours)  => n contient .........
```

L'indiçage d'une liste

```
jours=['lundi', 'mardi', 'mercredi', 'jeudi', 'vendredi']

print (jours[0])      => lundi

print (jours[2])     =>mercredi

print (jours[4])     =>vendredi

print (jours[5]) : index out of range
```

Passer en revue liste qui s'appelle jours, revient à regarder : jours[0] jours[1] jours[2] jours[3] jours[4] => jour[i] dans une boucle.

```
jours= ['lundi', 'mardi', 'mercredi', 'jeudi', 'vendredi]
i=0
n=len(jours)
while i<n:
     jours[i] => va prendre les valeurs de la liste
     i=i+1
```

Les listes sont modifiables

```
jours=['lundi', 'mardi', 'mercredi', 'jeudi', 'vendredi']
jours[0]= 'monday'

print jours => ['monday', 'mardi', 'mercredi', 'jeudi', 'vendredi']
```

Ajouter un élément à une liste

```
jours=['lundi', 'mardi', 'mercredi', 'jeudi', 'vendredi']
jours.append('samedi')
```

Supprimer un élément d'une liste

```
jours=['lundi', 'mardi', 'mercredi', 'jeudi', 'vendredi']
del(jours[3])
```

9. Exercices sur les listes

Créez dans le répertoire programmation un sous répertoire listes201? dans lequel vous sauvegarderez vos programmes.

1) Soit 1 liste de nombres

nb=[15,89,5,99,23,78,12,33,68,16]

Afficher les cotes supérieures à 30 et comptabiliser les (Sauvez sous liste1.py)

2) Etablir un programme qui permet de construire une liste composée de 25 nombres tirés au hasard et compris entre 1 et 100. Partez d'une liste vide et utilisez la méthode **.append()**

3) Soit 1 liste de nombres

nb=[25,89,65,99,13,78,12,33,68,17]

Trouver et afficher le maximum de la liste

4) Soit 1 liste de nombres

nb=[15,89,5,99,23,78,12,33,68,16]

Trouver la moyenne
Indiquer combien il y a de cotes supérieures à 30. (Sauvez sous liste1.py)

5) Soit 1 liste

ventes=["Bruxelles",12897, "Charleroi",11782, "Namur", 17651,"Liège",17670]

Faites la moyenne des ventes. Partez **absolument** de cette liste. Votre programme doit fonctionner si l'on ajoute une ville et un montant.

6) Soit 1 liste

capitales=['Bruxelles','Paris','Londres','Berlin','Luxembourg','Madrid','Rome']

Afficher les mots et leur longueur (il faut utiliser 2 len un pour la liste et un pour le mot)

Ex Bruxelles 9
Paris 5
.....…

7) Soit 1 liste de nombres

nb=[25,89,65,99,13,78,12,33,68,17]

afficher **l'indice** du plus grand nombre, la réponse ici est 3

8) Soit une liste

vecteur=[27,62]

Echangez les valeurs au sein de la liste. => vecteur devient [62,27]

bonus) Quel jour êtes-vous né ?

La date de naissance doît être comprise entre 1900 et 1999

1°) **Ajoute** :

- **A** : les deux derniers chiffres de ton année de naissance ;
- **B** : l'entier juste en-dessous du quart de A ;
- **M** : un nombre associé au mois de ta naissance (janv.=0, fév.=3 mars=3, avril=6, mai=1, juin=4, juill.=6, août=2, sept=5, oct.=0, nov.=3, déc.=5) ; **(utilisez une liste)**
- **J** : le numéro du jour du mois de ta naissance.

*2°) **Divise le résultat par 7;***
*3°) Regarde **le reste de la division** : si c'est un 0, tu es né un dimanche ; si c'est un 1, tu es né un lundi ; si c'est un 2, tu es né un mardi ; ... etc **(utilisez une liste)***

***Exemple** : Si tu es né le 16 mai 1979.*
A = 79 ; B = 19 ; M = 1 ; J = 16 ; A +B + M + J = 115 .
115 divisé par 7, il reste 3.
Tu es né un mercredi !

***Remarque** : Cet algorithme ne passe pas l'an 2000 !!!*
Il faut alors retrancher 2 au reste de la division ou ajouter 5 si le reste est plus petit que 2 (Exemple : Le 1er janvier 2000 était un samedi et donne 1 comme reste au lieu de 6).

10. Devoir sur les listes

Créez dans le répertoire programmation un sous répertoire dlistes2013 dans lequel vous sauvegarderez vos programmes.

1) Soit 2 listes

facturegsm =[25,**55**,20,**63**,**120**,45,**67**,**81**,**96**,26,**56**,36]
mois=[‘janvier’,’février’,’mars’,….]

établir un programme qui permet

- de compter le nombre de mois où les dépenses sont supérieures à 50 (ici 7)
- d’afficher les mois ou les dépenses sont supérieures à 50
- moyenne mensuelle des dépenses de communication

Sauvez sous liste1.py

2) Tirez 200 nombres au hasard compris entre 1 et 100, et ajoutez les dans la liste **lst50** s'ils sont supérieurs à 50**.** Partez d'une liste vide et affichez à la fin la liste complète.

Sauvez sous liste2.py

3) Soit 1 liste

fruits=['pomme','poire','ananas','kiwi','banane','mangue','peche','prune']

affichez les mots inférieurs ou égal à 5 lettres, combien y en a t-il ?

Sauvez sous liste3.py

4) Soit 1 liste de nombres

nb=[25,89,65,99,13,78,12,33,68,17]
0 1 2 3 4 **5 6** 7 **8** 9

Affichez la **position !!!** des nombres pairs. Ici ce sera 5 6 8

Sauvez sous liste4.py

5) Soit 1 liste de nombres

nb=[25,89,65,99,13,78,12,33,68,17]

Affichez le minimum (cf. Maximum)
On initialise mini=nb[0] et pas mini=0

Sauvez sous liste6.py

11. Les string

11.1 Définition: ……

11.2. A retenir

```
m= str(input("introduisez un mot"))

x="bon"
v="jour"
z= x+v  => z contiendra bonjour ( + concaténation ou
fusion)

z="ordinateur"
n=len(z)  => n contient 10
```

```
nom= "Juliette"

print nom[0] : J…
print nom[3] : i…….
print nom[7] : e….
print nom[8] : index out of range.
print nom[0:3] Jul       : signifie jusque
```

11.3. Passer en revue une chaîne de caractères qui s'appelle nom, revient à regarder : nom[0] nom[1] nom[2] nom[3] nom[4] nom[5] nom[6] nom[7]

```
mot= "Juliette"
i=0
n=len(mot)
while i<n:
     mot[i] #  mot[i] représente 1 lettre du mot
     i=i+1
```

```
mot="ordinateur"
for lettre in mot :
     lettre  => représente chaque lettre du mot
```

11.4. Les chaînes sont non modifiables

```
m="butman"
m[1]="a"   # =>  interdit
```

11.5 Remplacer un caractère par un autre dans un string20

```
mot="ananas"
mot=mot.replace("a","i")
```

nouvelle variable

11.6 Inverser un string

```
txt = "Hello World" [::-1]
print(txt)
```

Créez dans le répertoire programmation un sous répertoire string20?? dans lequel vous sauvegarderez vos programmes.

12. Exercices sur les strings

1) Réalise un programme qui compte le nombre de "e" dans une phrase.

2) Réalise un programme python qui épelle ton nom

Exemple
Quel est ton nom? : PAUL

Ton nom s'écrit donc:

P
A
U
L

3) Idem mais ton nom s'affiche en commençant par la fin

L U A P (sans passer à la ligne)

4) Réalise un programme qui compte les voyelles dans une phrase

voy="aeiouy"
et on regarde si une lettre de la phrase est dans (instruction **in**) voy .

5) Réalise un programme qui permet de supprimer les espaces dans une phrase **(! 2 variables)**

Ex: il fait froid
ilfaitfroid

utilise un accumulateur dans lequel tu places uniquement les lettres.
Ou utilise replace.

6) Etablir un programme qui permet d'introduire un numéro de compte bancaire européen et qui en extrait le n° banque, le digit

ex : BE**68**001156575**71**
n° banque : 68
digit 71

7) Soit une variable **m** qui contient "la t§l§vision est cass§e, il faut la r§parer". Etablir un programme qui passe en revue cette chaîne et qui en construit une autre **m2** où les § sont remplacés par é. (2 versions avec ou sans replace())

très proche du programme qui élimine les espaces.

8) Etablir un programme qui permet d'introduire un mot et qui teste s'il s'agit d'un palindrome.

Ex RADAR

13. Des chiffres et des lettres

Créez dans le répertoire programmation un sous répertoire lettreschiffres2013 dans lequel vous sauvegarderez vos programmes.

De nombreuses applications nécessitent d'utiliser un mélange de chiffres et de lettres or ces deux types sont incompatibles. C'est pourquoi nous allons utiliser des fonctions permettant la conversion.
La fonction **str()** qui permet de transformer une expression numérique en string.
La fonction **float()** ou **int()** qui permettent de transformer une expression alphanumérique en nombre.

1) Application : établir un programme qui permet à l'utilisateur d'introduire le nombre d'ordinateurs qu'il désire dans son réseau ainsi que le numéro d'ordre du premier ordinateur. Le programme constitue alors deux listes. Celle des noms des ordinateurs et celle des adresses Ip.
Voici comment devront se présenter les listes si l'utilisateur veut 5 ordinateurs et que le numéro d'ordre du premier est égal à 14

liste_ordi=['ordi14','ordi15','ordi16',...,'ordi18']
liste_ip=['192.168.0.14', '192.168.0.15', '192.168.0.16',....,'192.168.0.18']

bonus : l'adresse 192.168.0.0 ainsi 192.168.0.255 ne peuvent être utilisée, et bien entendu on ne peut aller au delà de 192.168.0.254 adresse ip .maximale dans un réseau de classe C.

2) Établir un programme qui permet à l'utilisateur d'introduire le nombre de mots de passe qu'il désire et qui constitue ensuite une liste de ces mots de passe comme suit:

liste_pswd= ['vtuc48','stbf31',.....]

chaque mot de passe est constitué de 4 lettres et 2 chiffres.

14. Les listes à 2 dimensions et la boucle for

14.1. Exemples :

0 1 2 3 => ligne
liste=[[11,24,33],[47,52,66],[71,89,91],[10,73,47]]
0 1 2 =>colonne

grille=[[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0],[0,0,0,0,0,0]]

panier=[[3,7],[2,10],[1,4],.....]

14.2 Passer en revue une liste

liste[ligne] [colonne]

Passer en revue une liste de liste consiste à regarder liste[0] [0], liste[0] [1], liste[0] [2], liste[1] [0], liste[1] [1], liste[1] [2], etc . Pour cela nous allons utiliser une boucle **for.**

```
liste=[ [11,24,3],[47,52,66],[71,89,91],[10,73,47] ]
for i in range(0,4): # ligne
     for j in range (0,3):  # colonne
         print(liste[i][j])
```

14.3. Exercices (une matrice est une liste à 2 dimensions)

Créez dans le répertoire programmation un sous répertoire matrice201? dans lequel vous sauvegarderez vos programmes.

1. Soit une matrice **exematr.py**, je vous demande de faire la somme de la matrice.

2. Soit une matrice **exematr.py**, je vous demande de calculer la somme de la 2ème colonne ainsi que de la 3ème ligne. (pas de boucles imbriquées une seule boucle est nécessaire)

3. Soit une matrice **exematr.py**, je vous demande d'afficher cette matrice ligne par ligne. (on utilise un seul indice pas deux)
[11,34,39,22]
[73,84,21,93]
.....

4. Créez une matrice carrée de n sur n, n étant fourni par l'utilisateur et initialisez la avec des zéros.

```
lst=[]   #matrice de 3 lignes 4 colonnes
for i in range(0,3):
     lst.append([0]*4)
```

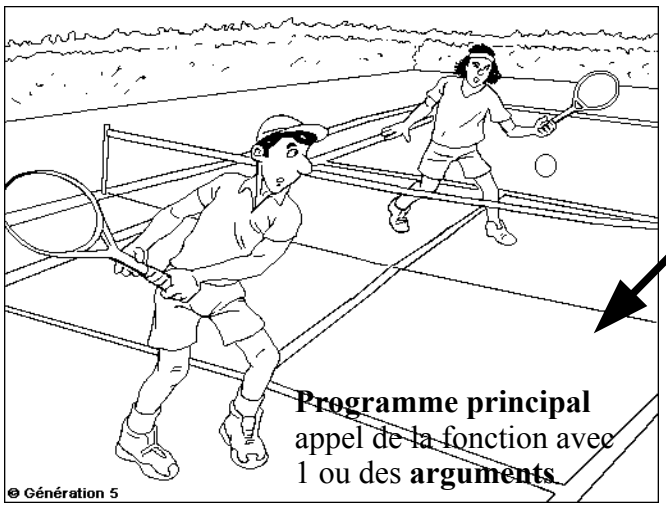
5. Soit cette liste ventes=[['Marc',2600],['Julie',7800],['Paul',3600],['Luc',9500],['Lola',1100]] .
Ecrire un script qui permet d'afficher les pourcentages des ventes de chaque collaborateur.

Marc 10,569 %
.....

15. Les fonctions

15.1. Une fonction c'est du code réutilisable

Appel de la FONCTION pour le prénom



Au niveau de la **fonction** réception du ou des arguments placés dans 1 ou plusieurs **paramétres**

return

Programme principal
appel de la fonction avec 1 ou des **arguments**

15.2. Concrètement en python

```
def nomFonction(paramètres) :
      ......................
      ......................
      ......................
      ......................

# *** pgr principal ***

print(nomFonction(arguments))
```

15.3. Mais ce n'est pas si simple ....

Teste ce programme2, que veut-on démontrer ?

```
def Nonmutable(z):
    z=z*2
    print ("z vaut ",z)


# ------- Programme principal
-----------------------------------

z=int(input("un nb svp "))
Nonmutable(z)
print ("z vaut ",z)
```

Réponse : .....
.....
.....
.....
.....

15.4. Teste ce programme3, que peut-on dire par rapport au programme2 ?

```
def Mutable(lst):
    lst[0]=lst[0]*2
    lst[1]=lst[1]*2
    print ("lst vaut ",lst)


# ------- Programme principal
-----------------------------------

j=[]
j.append(2)
j.append(3)
print ("j vaut ",j)
Mutable(j)
print ("j vaut ",j)
```

Réponse : .....
.....
.....
.....
.....

14.5. Teste ce programme et explique à quoi sert un return

```
def Volume(h,r):
    vol=3.1415*h*r**2
    return vol


# ------- Programme principal
-----------------------------------

hauteur= int(input("introduisez la hauteur de la boite"))
rayon= int(input("introduisez le rayon"))
print ("le volume est de ", Volume(hauteur,rayon))
```

Réponse : .....
.....
.....
.....
.....
.....

Mutable ou Nonmutable ? Thats's the question.

Un objet est **nonmutable** lorsqu'il est **impossible de le modifier.**

Un objet **est mutable** lorsqu'il est **possible de le modifier.**

Les objets **nonmutables** sont les valeurs numériques, les chaînes de caractères, et les tuples.

Les objets **mutables** sont les listes, les dictionnaires. Généralement les autres objets (ceux que vous créez) sont aussi mutables.

Passage de Paramètres en python

En Python tous les paramètres sont passés par référence (références à des objets), ces références étant associées aux noms des paramètres lors de l'appel.

Mais!!!

- Si un paramètre est une valeur **nonmutable**, la fonction n'aura aucun moyen de modifier l'original, l'appelant conserve la valeur telle qu'il l'a transmise.

- Si un paramètre est une valeur **mutable**, la fonction pourra modifier l'original, et avoir ainsi un effet de bord (désiré/prévu ou non) vis à vis de l'appelant qui a fourni ce paramètre.

NB il existe des fonctions sans return
NB il existe des fonctions sans paramètres

15.6. Exercices

*Créez dans votre répertoire programmation un sous répertoire **fonctions** dans lequel vous sauvegarderez vos programmes.*

Réalisez la fonction **surfCarre()**qui permet de calculer la surface d'un carré.

```
#-------------- pgr principal-----------------------------
cote=int(input("introduisez le coté du carré"))
print(surfCarre(cote))
```

sauve sous **fonction1.py**

Réalisez la fonction **intComp** qui retourne le capital final (cf) obtenu si je place un capital de départ (cd) à un taux (t) pendant un certain nombre d'années (n). taux est exprimé en % 5 % => 0.05
formule = **$cf= cd . (1+t)^n$**
ex. = ***10000 € * (1+1%) puissance 20 = 12 201 €***

```
# ------------------pgr principal-------------------------
print( intComp(10000,0.01,20))
```

sauve sous **fonction2.py**

Réalisez la fonction **motPasse()** qui permet de renvoyer un mot de passe aléatoire composé de n lettres et 2 chiffres

```
#-------------- pgr principal-----------------------------
passwd=motPasse(5)
print(passwd)  => srb68
```

sauve sous **fonction3.py**

Réalisez la fonction **verifIban()** qui permet de vérifier un numéro de compte bancaire belge. Cette fonction renverra True ou False.

```
#-------------- pgr principal-----------------------------
compte='BE68539007547034'
if verifIban(compte) :
    print('ok')
else :
    print('bad')
```

BE68 **5390 0754 70**34
extraire 5390 0754 70
transformer en nombre int()
appliquer modulo 97 on obtient 34 qui représente les 2 derniers digits BE68 5390 0754 70**34**

BE43 3400 2721 4501 BE89 0963 6391 1658

sauve sous **fonction4.py**

Réalisez la fonction **rechDom()**qui permet de rechercher le nom de domaine d'une adresse Email.

```
#-------------- pgr principal-----------------------------
email= "toto@voo.be"
print (rechDom(email)) => donnera voo.be
```

En utilisant la fonction split(), cet exercice est très simple.

```
flag ="noir;jaune;rouge"
lstflag=flag.split(";")
print(lstflag)          =>  ["noir","jaune","rouge"]
print(lstflag[2])       => rouge

              @ devient votre séparateur
```

sauve sous **fonction5.py**

14.7. Rappel sur les matrices

1.Rappel

```
matr                    représente toute la matrice
matr[0] [3]             représente un élément de la matrice
matr[0]                 représente une ligne de la matrice
nbligne = len(matr)     fournit le nombre de lignes de la matrice
nbcol = len(matr[0])    fournit le nombre de colonnes de la matrice
```

Parcourir une matrice : boucles imbriquées
Parcourir une ligne ou une colonne d'une matrice : simple boucle

2. Concevoir une fonction qui permet de faire la somme d'une matrice.

```
#-------------- pgr principal-----------------------------
matr=  [[11,34,39,22],\
       [73,84,21,93],\
       [69,80,18,75],\
       [54,72,12,41],\
       [34,27,10,14]]
print(sommeMatr(matr)) # (883)
```

sauve sous **matrice1.py**

3. Concevoir une fonction qui permet de faire la somme d'une colonne donnée d'une matrice.

```
#-------------- pgr principal-----------------------------
matr=  [[11,34,39,22],\
       [73,84,21,93],\
       [69,80,18,75],\
       [54,72,12,41],\
       [34,27,10,14]]
print(colMatr(matr,2))# (100)
```

sauve sous **matrice2.py**

4. Concevoir une fonction qui permet de faire la somme d'une ligne donnée d'une matrice.

```
#-------------- pgr principal-----------------------------
matr=  [[11,34,39,22],\
       [73,84,21,93],\
       [69,80,18,75],\
       [54,72,12,41],\
       [34,27,10,14]]
print(ligneMatr(matr,3)) # (179)
```

sauve sous **matrice3.py**

16. *Les modules*

En Python, un module est un fichier contenant du code Python qui peut être importé et utilisé dans un autre programme Python. Les modules sont utilisés pour organiser le code en fonctionnalités logiques et pour faciliter la réutilisation du code. (*)

16.1. Reprenons le programme des intérêts composés :

```
def intComp(cd,t,n):
    cf=cd*(1+t)**n
    return cf

#---------------Pgr principal ------------------
capitald= int(input("introduisez le capital de départ :
"))
taux = float(input("introduisez le taux : "))
duree = int(input("introduisez la durée : "))
print(intComp(capitald,taux,duree))
```

16.2. Créons à partir de ce fichier un autre fichier finance.py et transformons le :

```
def intComp(cd,t,n):
    cf=cd*(1+t)**n
    return cf


if __name__ == '__main__':
    capitald= int(input("introduisez le capital de départ
: "))
    taux = float(input("introduisez le taux : "))
    duree = int(input("introduisez la durée : "))
    print(intComp(capitald,taux,duree))
```

16.3. `if __name__ == '__main__'`

En Python, la construction `if __name__ == '__main__':` est utilisée pour déterminer si un script Python est exécuté en tant que programme principal ou s'il est simplement importé en tant que module dans un autre programme.(*)

16.4. Utilisons notre module dans un fichier simulation.py comme ceci :

```
from finance import *

capitald=10000
lst_taux=[0.01,0.02,0.03]
duree=20

for donnee in lst_taux:
    print(intComp(capitald,donnee,duree))
```

(*) from ChatGPT

16.5. Ajoutons au module finance une formule très complexe mais très utilisée (annuité)

Ajoutez dans le module finance le calcul de l'annuité d'un remboursement d'un prêt hypothécaire. Vous trouverez la doc à l'adresse suivante :
https://www.papernest.com/simulation-credit-immobilier/calcul-mensualites/annuite/

Ensuite testez : pour un prêt d'un montant de **250 000 €** sur **15 ans** avec un taux de **0,79%**, le calcul d'annuité est le suivant : 17739.33 €

Ensuite testez : ce même prêt sur 15 ans, 20 ans, 25ans et 30 ans.