Programmation: Python

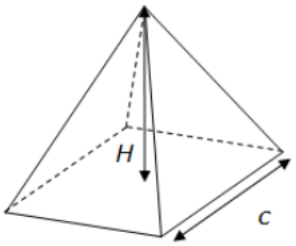
Langage Python:

On va utiliser Edupython
Python est un langage permettant de faire exécuter un algorithme par un ordinateur:
Programme = traduction d'un algorithme dans ce langage Python

Principales opérations:

Syntaxe:	Signification:
x+y	addition de x et y
x*y	produit de x et y
x/y	division de x par y
x**y	élévation de x à la puissance y
sqrt(x)	racine carrée de x
x//y	quotient de la division euclidienne de x par y
x%b	reste de la division euclidienne de x par y

Exemple:
Volume d'une pyramide à base carrée



Protocole:	*Algorithme:*	*Python:*	*Explications:*
Variables Les variables utilisées sont des nombres	Variables C H V		
Entrée	Entrée	print("Entrer c=")	- Affiche "Entrer c="
Saisie des variables Traitement	Saisir C	$c =$ float(input())	- On rentre la variable appelée C (côté) et celle-ci est stockée dans c
Opérations sur les variables	Saisir H	print("Entrer h=")	- Affiche "Entrer h="
	Traitement	$h =$ float(input())	- On rentre la variable appelée H (Hauteur) et celle-ci est stockée dans h
	V reçoit $\frac{C^2 \times H}{3}$	$v = (1/3) * c * * 2 * h$	- On procède au calcul $c * * 2$ équivaut à c^2 permet de calculer le volume
Sortie Affichage des résultats	Sortie Afficher V	print("v=",v)	- Affiche le résultat "v="

En langage Python, l'usage est de ne pas utiliser de lettres majuscules pour nommer les variables
Types de variable:
- *On utilise "float" pour réel (A=2,45)*
- *On utilise "int" pour entier relatif (A=2).*

n=int(input("entrer n")): on demande à l'utilisateur d'entrer un entier n, la valeur est affectée dans la variable n
x=float(input("entrer x")): on demande à l'utilisateur d'entrer un réel x, la valeur est affectée dans la variable x
x=eval(input("entrer n")): on demande à l'utilisateur d'entrer un nombre (qui peut être une fraction "a/b" ou une racine carrée "sqrt(a)"), la valeur est affectée dans la variable x

Si la variable A vaut "mathématiques", alors A est de type chaine de caractère, noté "str"

Attention:
On utilise l'instruction "input" ce qui provoque l'arrêt du programme afin que l'utilisateur puisse rentrer la donnée demandée

Variables :	On utilise les variables pour les mémoriser : Le fait de donner une valeur à une variable = *affectation* Exemple: On donne la valeur 2 à la variable A Algorithme : A ⟵ 2 et en Python: A = 2
Entrées / Sorties:	Entrées = données à traiter Exemple: on saisit les longueurs du côté et de la hauteur; ces valeurs sont affectées aux variables "c" et "h" . Sorties = résultat du problème à la suite de l'exécution du programme Exemple: le programme calcule et affiche le volume "v" en sorties

ATTENTION:
Ne pas confondre: " $a = b$ " et " $a == b$ "
- " $a = b$ " correspond à l'Affectation, c'est à dire que le contenu de la variable b est mis dans la variable a
- " $a == b$ " correspond à un test d'égalité, c'est à dire on vérifie si les contenus des variables a et b sont égaux.

Exemple:
On reprend l'exemple du volume d'une pyramide à base carrée:

Algorithme:	$1^{ère}$ possibilité:	2^{nde} possibilité:
Variables C H V		
Entrée Saisir C Saisir H	 c=5 h=3	 c = float(input(print("Entrer la longueur du côté"))) h = float(input(print("Entrer la longueur de la hauteur")))
Traitement V reçoit $\frac{C^2 \times H}{3}$	Volume=1/3*c**2*h	v = (1/3) * c * * 2 * h
Sortie Afficher V	print(Volume)	print("le volume est égal à ",v)
Explications:	Les trois premières lignes sont des affectations de variables. La $3^{ième}$ ligne est l'écriture d'une formule permettant un calcul combinant des variables.	On demande à l'utilisateur de saisir les valeurs nécessaires au lancement du programme. Si on souhaite utiliser des nombres décimaux, on écrira "float". Si on souhaite utiliser des nombres entiers, on écrira "int". On utilise des guillemets pour écrire du texte dans l'instruction input.

Les conditions:

Elles permettent de faire une action précise
Exemple: dire si un nombre est strictement supérieur à 5 ou non

Instructions conditionnelles:

1) Test "if ...." :
On effectue une action qui a été définie auparavant dans le programme et on regarde si elle vérifiée.

Reprenons l'exemple ci-dessus:

```
a=6
if a>5:
    print("a est strictement supérieur à 5")
if a<5:
    print("a est strictement inférieur à 5")
```

Explications:
Avec Python
"alors" est remplacé par les ":" , c'est obligatoire sinon erreur de syntaxe.
A la suite de l'instruction "if" , Python va effectuer les actions qui sont indentées (= décalées)
On peut se faire succèder plusieurs instructions "if"
On constate que la condition "if " est nécessaire mais insuffisante car il n'y a pas d'action de prévue au cas où a=5

Opérateurs de comparaison:

Opérateurs:	Signification:
x<y	x est strictement inférieur à y
x>y	x est strictement supérieur à y
x<=y	x est inférieur ou égal à y
x>=y	x est supérieur ou égal à y
x==y	x est égal à y
x!=y	x est différent de y

Tests:

En français :	En Python:
Si n est égal à 0	if (n==0):
Si n est strictement positif	if (n>0):
Si n est différent de 3	if (n!=3):
Si n est strictement compris entre 5 et 15	if (n>5) and (n<15):
Si n est divisible par 2	if (n%2==0):

2) Test "if ... else ...":
L'instruction "if ... else ..." offre une alternative du type "ou bien ... ou bien ...":

Reprenons l'exemple ci-dessus:

```
a=6
if a>5:
    print("a est strictement supérieur à 5")
else:
    print("a est strictement ou égal à 5")
```

Explications:
Après le "if", on peut mettre le "else" au même niveau que le "if" correspondant suivi des ":"
On constate que le cas a=5 est pris en compte
L'instruction "else" ne peut figurer *qu'une seule fois* , elle clôture le bloc de la condtion.

3) Test "if ... elif ...":
"elif" = "else if", ce qui veut dire "sinon si"
L'objectif est de trouver une condition intermédiaire permettant de faire la distinction entre les nombres supérieurs à 5, inférieurs à 5 ou égal à 5

Reprenons l'exemple ci-dessus:

```
a=6
if a>5:
    print("a est strictement supérieur à 5")
elif a<5:
    print("a est strictement inférieur à 5")
else:
    print("a est égal à 5")
```

Explications:
Après le "if", on peut mettre le "elif" au même niveau que le "if" correspondant suivi des ":"

Les boucles:

Elles permettent de répéter une action du programme un certain nombre de fois.

Boucle "Tant que ..." :	*Boucle "Pour ...":*

Cette instruction se traduit par "while" en Python, elle exécute une instruction ou un bloc d'instrucutions tant que la condition est vérifiée.
Pour savoir combien de fois la boucle while est exécutée, il peut être utile d'utiliser une variable qui servira de compteur, c'est-à-dire une variable (qu'il ne faut pas oublier d'initialiser) et incrémentée de 1 dans le bloc d'instructions
 	
Exemple:
$i=1$
while $i<5$:
  print(i)
  $i=i+1$
Le programme affiche:
1; 2; 3 et 4
Tant que (while) la condition $(i<5)$ est vraie, le bloc d'instructions est exécuté.
	Cette instruction se traduit par "for" en Python, elle répète un certain nombre de fois (nombre déterminé à l'avance) une instruction ou un bloc d'instrucutions.
	
	L'instruction "for i in range(...)" est suivie de ":"
	La fonction "range( )" permet d'énumérer le nombre de passage dans la boucle bornée.
	1) "for i in range(n):" i varie de 0 à n-1
	- range(6) donne la liste (0,1,2,3,4,5)
	2) "for i in range(n,m):" i varie de n à m-1
	- range(11,16) donne la liste (11,12,13,14,15)
	3) "for i in range(n,m,k):" i varie de n à m-1 avec un pas de k (si k est positif, les entiers sont rangés dans un ordre croissant; si k est négatif, ils sont rangés dans un ordre décroissant):
	- range(5,21,5) donne la liste (5,10,15,20)
	- range(11,18,-2) donne la liste(17,15,13,11)
	
	Exemple:
	S = 0
	for i in range(1,31,5)
	  S = S + i
	  print(S)
	Le programme affiche:
	1; 7; 18; 34; 55 et 81
	On veut calculer 1+6+11+16+21+26
	A chaque boucle for on ajoute i à S

Les fonctions:

Dans un programme on peut écrire des sous-programmes intermédiaires appelés *fonctions*
Le nom de la fonction est choisi en fonction de ce qu'elle apporte au programme.
Les fonctions sont une succession d'instructions regroupées dans un bloc, désigné par un nom.
Une fonction peut avoir aucun, un ou plusieurs paramètres, comme les fonctions en mathématiques .
De nombreuses fonctions sont disponibles, il faut les importer avec le mot "*import*":

from math import sqrt : on importe la fonction racine carrée

Syntaxe pour définir une fonction:

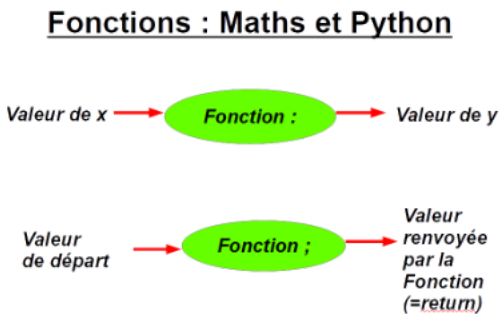
def nomdefonction(argument1,argument2,...): (*def* permet la construction d'une fonction)
(*arguments* sont des variables nécessaires à l'exécution de la fonction: on les liste et on les sépare par une virgule). Ne pas oublier les ":"

bloc_instruction(s)

return resultat (*return* permet à la fonction de renvoyer une ou plusieurs valeurs séparées par une virgule)

Avantages:
- Les fonctions rendent le programme plus lisible
- On peut appeler cette fonction autant de fois que cela est nécessaire.
- Le résultat obtenu par cette fonction est réutilisable dans un programme ou une autre fonction

Exemple:



Maths:	*Python:*
Une fonction f est un procédé qui à tout nombre réel x associe un seul nombre réel y x s'appelle la variable et y s'appelle l'image de x par la fonction f et se note $f(x)$ ou $y=f(x)$ Aire d'un carré: $c\hat{o}\ t\acute{e}\times c\hat{o}\ t\acute{e}$ $f(x)=x^2$ x | -2 | -1 | 0 | 1 | 2 $y=f(x)$ | 4 | 1 | 0 | 1 | 4	Fonction aire qui prend en argument la variable x def aire(x): return x**2 print(aire(2)) La console Python indique 4