TP ALGO 1 : NOTIONS DE BASE ET LANGAGE PYTHON

I. NOTION D'ALGORITHME ET DE PROGRAMME

Un **algorithme** est une suite d'instructions : une recette de cuisine est un algorithme.

Il est composé :

- Des entrées (les ingrédients)
- Du traitement (les différentes étapes de la recette)
- De la sortie (un bon gâteau ou tout autre plat de votre choix …)

Un algorithme peut s'écrire en langage naturel, ou en pseudo-code.

Exemples

1.

Langage naturel :	*pseudo-code*
a prend la valeur 3	$a \leftarrow 3$
b prend la valeur 5	$b \leftarrow 5$
c prend la valeur $a \times b + 2 \times a$	$c \leftarrow a*b+2*a$
Ecrire c	*Afficher c*

Dans cet exemple, identifier les entrées, le traitement et la sortie.

Compléter le tableau suivant, chaque ligne du tableau correspondant à une ligne de l'algorithme. Préciser la valeur de la sortie.

étape	*Valeur de a*	*Valeur de b*	*Valeur de c*
1	*3*		
2			
3			

2. Reprendre les questions précédentes avec l'algorithme écrit en langage naturel suivant :

a prend la valeur -1

b prend la valeur 2

a prend la valeur a^2

a prend la valeur $a \times 5$

b prend la valeur $a \times b$

Afficher a

Afficher b

étape	*Valeur de a*	*Valeur de b*
1	*-1*	
2		
3		
4		
5		

Un **programme** est la traduction d'un algorithme dans un langage compréhensible par un ordinateur. Il est composé d'**instructions**, une instruction correspondant à une action.

II. ENVIRONEMMENT PYTHON

Nous utiliserons le langage de programmation Python, et l'environnement de travail anaconda (Spyder).

Dans un premier temps, nous utiliserons uniquement la console (fenêtre en bas à droite).

Exercice : Compléter par le résultat du calcul écrit dans la console Python :

*3+4*7*	*2**3*	*2*(3**2)*	*21/2*	*5/2*3*	*212*10*-2*

III. LES VARIABLES

Pour écrire un programme, il faut enregistrer les données nécessaires au traitement dans des espaces mémoires.

Les **variables** désignent des emplacements de stockage. Elles sont repérées par des noms, et leur valeur peut évoluer au cours du temps.

Les variables peuvent être de différents types : entier (int), décimal (on parle de flottant : float), texte (on parle de chaîne de caractère : str), booléen (bool) …

En Python, on donne une valeur à une variable à l'aide de l'instruction =. Par exemple $a=2$ signifie que la variable a contient le nombre entier 2. On parle **d'affectation de variable**.

Exemples

```
>>> a=2
>>> b=5
>>> c=a*b
>>> c
10
```

```
>>> article="le"
>>> nom="lycée"
>>> verbe="est"
>>> adjectif="chouette"
>>> phrase=article + nom + verbe + adjectif
>>> phrase
'lelycéeestchouette'
```

Exercice 2 : Quelles sont les variables utilisées dans l'exemple ci-contre ? Préciser la valeur de chacune d'elles. Change-t-elle lors de l'exécution du programme ?

```
>>> a=3
>>> b=5
>>> c=a*b+2*a
>>> print(c)
21
```

Même question avec le programme ci-contre.

Quelles sont les différences entre ces deux programmes ?

```
>>> a=3
>>> b=5
>>> a=a*b+2*a
>>> print(a)
21
```

Exercice 3 : Compléter le tableau en indiquant les valeurs successives de a et b au fur et à mesure de l'exécution de la séquence d'instructions donnée :

Instructions
`>>> a=4`
`>>> b=3`
`>>> a=b`
`>>> a=a+b`

a	b
4	

IV. LES INSTRUCTIONS ELEMENTAIRES

Il existe deux types d'instructions élémentaires :

- Les opérations sur les variables : opérations mathématiques sur les nombres, longueur d'une chaîne de caractère ...
- Les instructions d'entrée et de sortie :
  - ✓ Une instruction d'entrée permet à un programme de lire des valeurs saisies au clavier par l'utilisateur
  - ✓ Une instruction de sortie affiche la (ou les) valeur(s) de variables à l'écran.

En Python, l'instruction d'entrée *input* permet d'affecter la valeur saisie dans une variable et l'instruction de sortie *print* permet d'afficher à l'écran la valeur de variables.

Attention : l'instruction *input* permet d'obtenir des chaînes de caractères. Si on veut saisir un entier, il faut écrire *int(input(...))*, si on veut saisir un décimal, il faut écrire *float(input(...))*. On peut mettre dans la parenthèse après le *input* un texte qui s'affichera lors de l'exécution.

Exercice 4 :

*1. Recopier le programme suivant **dans la zone d'écriture du code**, puis l'exécuter. Que fait ce programme ?*

```
a=input("entrez un mot")
b=2*a
print (b)
```

2. Mêmes questions avec le programme :

```
a=float(input("entrez un nombre"))
b=2*a
print (b)
```

Exercice 5 : Vous avez vu en seconde comment évaluer l'intensité de la pesanteur g selon l'altitude à laquelle on se situe, notée h. Pour cela, on rappelle que $g=\frac{G\times m_{Terre}}{\left(R_{Terre}+h\right)^2}$.

Avec : $T_{Terre}=6371$ *km,* $G=6{,}67\times10^{-11}$ *USI, et* $m_{Terre}=6{,}0\times10^{24}$ *kg.*

1. Ecrire un script Python dans l'éditeur qui permet d'évaluer la constante g à une altitude de 100 m. L'exécuter et afficher le résultat de manière conviviale (c'est-à-dire avec du texte). Indiquer le type du résultat obtenu.

2. Modifier le programme pour que l'utilisateur puisse rentrer lui-même la hauteur h.

Exercice 6 : On souhaite écrire un programme qui demande à l'utilisateur un nombre d'œufs et affiche le nombre de boîtes de 6 œufs nécessaires à leur transport. On considère ce programme, qui utilise la division euclidienne :

```
n = int(input("combien d'oeufs ?"))
print (n//6)
```

Tester ce programme sur différentes entrées.

1. Sur quelles valeurs de n ce programme est-il correct ?

2. Pourquoi n'est-il pas correct de remplacer n//6 par n//6+1 ?

3. Proposer une solution correcte.

V. Les bibliothèques

Python propose beaucoup de bibliothèques contenant des fonctions spécifiques. On doit importer ces bibliothèques pour pouvoir les utiliser.

Par exemple, si on a besoin de calculer une racine carrée, on soit importer la bibliothèque math. Il y deux façons d'importer une bibliothèque :

1ère façon :

```
from math import *

print(sqrt(9))
```

2ème façon :

```
import math

print(math.sqrt(9))
```

La première façon, qui semble a priori plus simple, peut parfois poser des problèmes lors de l'import de plusieurs bibliothèques, car on peut trouver le même nom de fonction dans deux bibliothèques différentes. Dans ce cas, seule la dernière importée sera opérationnelle. Cette méthode est donc déconseillée lorsque l'on importe plusieurs bibliothèques.

La bibliothèque TURTLE

Python permet, grâce à la bibliothèque TURTLE, de créer des images et animations.

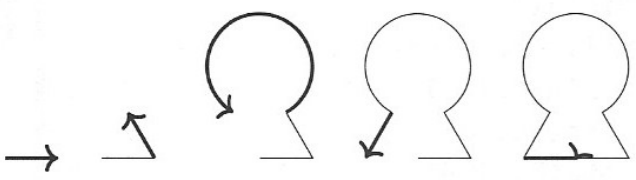
Vous trouverez toute la documentation utile ICI.

Les instructions de base sont les suivantes :

instruction	description
`goto(x, y)`	aller au point de coordonnées (x, y)
`forward(d)`	avancer de la distance d
`backward(d)`	reculer de la distance d
`left(a)`	pivoter à gauche de l'angle a
`right(a)`	pivoter à droite de l'angle a
`circle(r, a)`	tracer un arc de cercle d'angle a et de rayon r
`dot(r)`	tracer un point de rayon r

Exemple

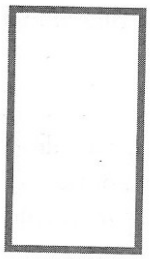
```
from turtle import *
forward(60)
left(120)
forward(60)
right(90)
circle(60, 300)
right(90)
forward(60)
goto(0, 0)
```



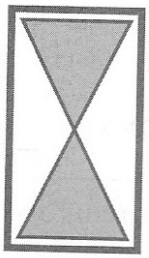
instruction	description
`up()`	relever le crayon (et interrompre le dessin)
`down()`	redescendre le crayon (et reprendre le dessin)
`width(`e`)`	fixer à e l'épaisseur du trait
`color(`c`)`	sélectionner la couleur c pour les traits
`begin_fill()`	activer le mode remplissage
`end_fill()`	désactiver le mode remplissage
`fillcolor(`c`)`	sélectionner la couleur c pour le remplissage

Exemple

```
from turtle import *
# Rectangle épais
width(6)
color(0.2, 0.2, 0.2)
goto(60, 0)
goto(60, 110)
goto(0, 110)
goto(0, 0)
# Déplacement
up()
goto(5, 5)
down()
```



```
# Sablier gris clair
width(1)
fillcolor("grey")
begin_fill()
goto(55, 5)
goto(5, 105)
goto(55, 105)
goto(5, 5)
end_fill()
```



Applications :

Reproduire les dessins suivants avec Turtle :