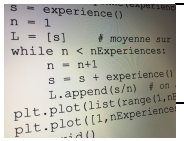

FICHE n°2 : PROGRAMMER L'*AFFECTATION*

Syntaxe des instructions utiles dans cette fiche :

Langage naturel	Python
Affecter à A la valeur 5	A=5
Saisir x	def nom_fonction(x) *Dans la console, on appellera :* nom_fonction(…)
Afficher A	return A *Si une fonction a été définie*
Afficher A	print(A)
Quotient de la division euclidienne de A par B	A//B
Reste de la division euclidienne de A par B	A%B
Racine carrée de A	sqrt(A) *Dans la console, on commencera par importer la fonction* sqrt
Reste de la division euclidienne de A par B	A%B
A ← 2 B ← 2 x A C ← B ^{2} Afficher C	

Exercice 1 :

Voici un algorithme écrit en langage naturel :

Ce même algorithme peut se traduire en langage de programmation Python :

```
A=2
B=2*A
C=B**2
print(C)
```

1) Quelle valeur obtient-on en sortie ? Vérifier en saisissant le programme.

2) Modifier le programme en affectant à A la valeur 4 et en affichant également la valeur de B en sortie. Tester le programme et noter la valeur obtenue en sortie.

3) Modifier la première ligne du programme pour obtenir C = 25 en sortie.

Exercice 2 :

1) Programmer avec Python chacun des algorithmes suivants. On recopiera les programmes saisis sur la copie.

2) Quelle(s) valeur(s) obtient-on en sortie pour chaque programme ?

Algorithme 1	Algorithme 2	Algorithme 3
A ← 7 B ← 6 x A C ← A + B D ← B – C Afficher D	M ←2 N ← 4 A ← M x N B ← M + N C ← A/B Afficher C	A ← -1 B ← 6 P ← B^A Q ← P^A Afficher P Afficher Q

Exercice 3 :

1) a) Saisir le programme Python ci-contre.
   b) Depuis la console, saisir **equation(0)**. Qu'obtient-on en sortie ?
   c) Donner une interprétation des résultats obtenus en sortie.

```
def equation(x):
    y1=x**2-5
    y2=-3*x**2+8*x+7
    return y1,y2
```

2) a) À l'aide du programme, calculer les images de et pour toutes les valeurs entières de x de 1 à 10.
   b) Existe-t-il une valeur de x pour laquelle ?

Exercice 4 :

Pour chacune des équations suivantes, écrire et tester un programme permettant d'en trouver au moins une solution.

1) $2x^2 - 3x - 20 = x + 28$
2) $-2x^3 + 111x = x^3 + 252$
3) $-2x^3 + 27x^2 = 16x + 240$

Exercice 5 :

Ecrire un programme où l'on saisit deux nombres entiers naturels au départ et où l'on obtient le quotient et le reste de la division euclidienne de ces deux nombres en sortie.

Exercice 6 :

Ecrire un programme qui affiche la longueur d'un segment AB connaissant les coordonnées de A et de B.

Exercice 7 :

Inventer et tester un programme mettant en œuvre de nombreuses instructions vues dans cette fiche (saisie, affectation, affichage, quotient, reste, …).

L'*INSTRUCTION CONDITIONNELLE*

Langage naturel	Python
Si *Condition* **Alors** *Instructions1* **Sinon** *Instructions2* **Fin Si**	**if** *condition*: *Instruction1* **else:** *Instruction2*
Quotient de la division euclidienne de A par B	A//B
Reste de la division euclidienne de A par B	A%B

Exercice 1 :

1) Expliquer le principe de l'algorithme ci-contre. Que permet-il de faire ?

```
Saisir a
b ← a/13
c ← le quotient de la division
                euclidienne de a par 13
Si b = c
   Alors afficher "True"
Sinon
   Afficher "False"
Fin Si
```

2) Ce même algorithme peut se traduire par le programme ci-dessous.

Quelles valeurs obtient-on pour b et c lorsqu'on saisit a = 182 au départ ? Qu'affiche l'algorithme en sortie dans ce cas.

```
def div(a):
    b=a/13
    c=a//13
    if b==c:
        return True
    else:
        return False
```

Commentaires :
"==" est le symbole d'égalité ; "=" celui d'affectation.

3) a) Modifier le programme dans le but de vérifier si un nombre est divisible par 29.
b) Les nombres suivants sont-ils divisibles par 29 ?
565 – 6785 – 646 195 034 – 1 970 659 794

Exercice 2 :

Ecrire un programme permettant de vérifier si un nombre donné est divisible par 13 en effectuant un test sur le reste de la division de ce nombre par 13.

```
Saisir x
Saisir y
Si x < 5y
   Alors  x ← 10x
Sinon
    y ← 10y
Fin Si
Afficher xy
```

Exercice 3 :

Dans le programme ci-dessous traduisant l'algorithme ci-contre, les instructions conditionnelles ont été supprimées.

```
def fonction(x,y):
    x<5*y:
    x=10*x
    y=10*y
    return x*y
```

1) Corriger en complétant le programme par les instructions conditionnelles manquantes.

2) Tester ce programme pour x = 5 et y = 9. Même question pour x = 12 et y = 2.

Exercice 4 :

1) Ecrire un programme traduisant l'algorithme ci-contre.

2) Tester ce programme pour trouver quelques triplets de Pythagore.

Exercice 5 :

Dans le programme ci-contre, les affichages en sortie de l'algorithme ont été supprimés.

1) Quel problème permet de résoudre cet algorithme ?

2) Compléter le programme par les affichages en sortie manquants.

3) Tester ce programme pour différentes valeurs de a et b.

```
def signe(a,b):
    if a>0:
        if b>0:
            print("le produit axb est ...")
        else:
            print("le produit axb est ...")
    else:
        if b>0:
            print("le produit axb est ...")
        else:
            print("le produit axb est ...")
```

Exercice 6 :

Écrire et tester un programme qui demande en entrée à un client le montant total de ses achats.
En fonction de la somme dépensée, le programme affiche en sortie le prix à payer :
- Si la somme dépensée est strictement inférieure à 75 €, il obtient 5 % de remise.
- Si la somme dépensée est supérieure à 75 €, il obtient 8 % de remise.

DES *BOUCLES*

Syntaxe :

Langage naturel	Python
Tant que *Condition est vraie* *Instructions* **Fin Tant que**	**while** *Condition*: *Instructions*
Pour *i* **allant de** *3* **à** *7* *Instructions* **Fin Pour**	**for** *i* **in range(***3,8***):** *Instructions*

En Python, **range**(*3,8*) désigne la séquence des entiers n vérifiant .
range(*5*) désigne la séquence des entiers 0, 1, …, 4.

Sortir de la boucle	break
Générer un nombre entier aléatoire de l'intervalle [1;5]	randint(1,5)

Rappels :

Si *Condition* **Alors** *Instructions1* **Sinon** *Instructions2* **Fin Si**	**if** *condition*: *Instruction1* **else:** *Instruction2*

Exercice 1 :

1) Tester le programme ci-contre. Qu'affiche-t-il en sortie ?
2) Ecrire et tester un programme qui affiche tous les entiers inférieurs à 16.
3) Ecrire et tester un programme qui affiche tous les entiers compris entre 18 et 45.

```
for i in range(10):
    print(i)
```

Exercice 2 :

1) Tester le programme ci-contre. Qu'affiche-t-il en sortie ?
2) Ecrire et tester un programme qui affiche tous les entiers pairs compris entre 18 et 45.
3) Ecrire et tester un programme qui affiche tous les entiers impairs
compris entre 50 et 150.

```
n=0
while n<10:
    print(n)
    n=n+2
```

Exercice 3 :

1) On donne le programme ci-contre.
Recopier et compléter le tableau suivant par les premières valeurs prises par les variables S et i.

```
s=0
for i in range(101):
    s=s+i
print(s)
```

i		1	2	3									
S	0	1	3										

2) Quel problème permet de résoudre cet algorithme.
3) a) En s'inspirant des programmes précédents, écrire et tester un programme permettant de calculer la somme des entiers de 34 à 145.
b) Même question pour la somme des entiers de 67 à 456.

Exercice 4 :

On place un capital de 500€ sur un compte rémunéré à 3% par an.
L'algorithme ci-contre, écrit en langage naturel, permet de calculer le nombre d'années au bout desquelles le capital sera doublé.

```
S ← 500
A ← 0
Tant que S<1000
    S ← 1,03xS
    A ← A+1
Fin Tant que
Afficher A
```

1) Le programme ci-contre traduisant l'algorithme précédent comprend une erreur. Corriger le programme et le tester.
2) Modifier le programme précédent de telle sorte que le capital et le taux de rémunération soient saisis en entrée. Le tester dans un nouveau contexte à décrire.

```
s=500
a=0
while s<1000:
    s=1.03*s
a=a+1
print(a)
```

Exercice 5 :

On dépose 25€ dans une tirelire.
L'algorithme suivant, écrit en langage naturel, permet de calculer le nombre de pièces de 1€ ou 2€ ajoutés de façon aléatoire dans la tirelire avant de dépasser 50€.

1) Compléter l'algorithme.
2) Pourquoi le programme affiche en sortie « D–1 » ?

```
Affecter à S la valeur 25
Affecter à D la valeur 0
Tant que S<…
    Affecter à A la valeur aléatoire 1 ou 2
    S ← …
    D ← D+1
    Afficher A
Fin Tant que
Afficher D–1
```

3) Ecrire et tester un programme traduisant cet algorithme.

Exercice 6 :

D'après "Document ressource pour la classe de seconde" – juin 2009

On demande à l'utilisateur de deviner en moins de six essais un nombre tiré au hasard entre 10 et 100.
On lui indique à chaque fois si le nombre proposé est supérieur ou inférieur au nombre cherché.

1) L'algorithme qui suit, écrit en langage naturel, permet d'effectuer le jeu.
   a) Que représentent les variables E, S et N ?
   b) Ecrire et tester un programme traduisant cet algorithme.

2) Sans stratégie, il est difficile de gagner. En effet, selon le choix des valeurs, il sera ou non possible de déterminer à coup sûr la solution.
La méthode consiste, en choisissant à chaque fois la valeur située au milieu de l'intervalle en cours, à réduire de moitié l'amplitude de l'intervalle dans lequel se trouve le nombre.
Tester cette stratégie pour gagner à tous les coups à ce jeu et détailler un coup gagnant.

```
Affecter à S la valeur d'un nombre
  aléatoire entier compris entre 10 et 100
Affecter à E la valeur 1
Tant que E<7
      Saisir N
      Si N>S
         Alors afficher "C'est moins"
      Sinon
         Si N<S
            Alors afficher "C'est plus"
         Sinon
            Afficher "C'est gagné"
            Sortir de la boucle
         Fin Si
      Fin Si
      Affecter à E la valeur E+1
Fin Tant que
Si E=7
   Alors afficher "C'est perdu"
Fin Si
```

Exercice 7 :

Ecrire et tester un programme permettant de calculer la somme des entiers naturels pairs inférieurs ou égaux à 1000.

Exercice 8 :

Ecrire et tester un programme permettant de calculer la puissance d'un nombre.

Exercice 9 :

1) Ecrire un algorithme qui permet de déterminer la moyenne d'une série de notes, où N est le nombre de notes. On pourra utiliser une boucle **Pour**.

2) Ecrire et tester un programme traduisant cet algorithme.