TP1 - Reprise en main de python - Corrigé

Dans les premiers TP de l'année , nous allons reprendre les bases du langage python. Nous commençons aujourd'hui avec les notions suivantes : - réaliser un calcul - définir une variable - afficher une valeur - définir une fonction

1 Réaliser un calcul, définir une variable

Python permet de réaliser simplement les calculs usuels sur les nombres

Opération	Syntaxe Python	Exemple
Addition	+	2 + 5
Soustraction	-	2 - 5
Multiplication	*	2 * 5
Division	/	2 / 5
Puissance	**	2 ** 5

La virgule des nombres décimaux s'écrira avec un point (par exemple, 2.3).

Il est toujours utile de pouvoir donner un nom à une valeur pour pouvoir la réutiliser plus tard. On dit que l'on **affecte une valeur** à une variable avec la commande : **nom = valeur**

1.1 Exercice 1

1. Réaliser le calcul $\frac{3,1+10}{8\times 10^{-6}}$.

[1]:
```
(3.1 + 10)/(8 * 10**(-6))
```

[1]:
```
1637500.0
```

2. Créer trois variables A, B et C et affecter à la variable A la valeur 10, à la variable B la valeur 12 et à C la valeur 3.

[2]:
```
A = 10
B = 12
C = 3
```

Vérifier demandant les valeurs de ces trois variables :

[3]:
```
A
```

[3]: 10

[4]:
```
B
```

[4]: 12

[5]:
```
C
```

[5]: 3

3. Effectuer les opérations suivantes :

- $2A - 3B$,
- $C(A + 2B)$,
- $2A^3 - (-1)^C + A^{B+C^2}$.

[6]:
```
2*A - 3*B
```

[6]: -16

[7]:
```
C*(A+2*B)
```

[7]: 102

[8]:
```
2*A**3 - (-1)**C + A**(B+C**2)
```

[8]: 1000000000000000002001

1.2 Exercice 2

Dans chaque cas, écrire le résultat attendu (de tête) puis, dans un second temps, vérifier.

[9]:
```
s = 0
s = s+1
s = s+2
```

[10]:
```
s
```

[10]: 3

[11]:
```
a = 1
b = 2
a = a-b
b = a+b
```

[12]:
```
a,b
```

[12]: (-1, 1)

[13]:
```
a,b,c = 1,2,1
a = 2*c + 3*b
c = b
b = a
```

[14]:
```
a,b,c
```

[14]: (8, 8, 2)

[15]:
```
a,b,c = 1,2,1
a = 2*c + 3*b
b = c
a = b
```

[16]:
```
a,b,c
```

[16]: (1, 1, 1)

1.3 Exercice 3 - division euclidienne

La **division euclidienne** entre deux entiers est la division avec reste.

Exemple : La division euclidienne de 17 par 3 est

$$17 = \underbrace{5}_{\text{quotient}} \times 3 + \underbrace{2}_{\text{reste}}.$$

Le **quotient** s'obtient avec //

Le **reste** s'obtient avec %

[17]:
```
17 // 3
```

[17]: 5

[18]:
```
17 % 3
```

[18]: 2

Calculer avec Python le reste et le quotient de la division euclidienne de 123456789 par 1234.

[20]:
```
123456789 % 1234 # reste
```

[20]: 25

[21]:
```
123456789 // 1234 # quotient
```

[21]: 100046

2 Afficher une valeur

Pour afficher la valeur d'une variable, on peut la demander en écrivant son nom.

[22]:
```
A = 10
A
```

[22]:
```
10
```

Cette méthode fonctionne si on a une seule valeur à afficher et qu'elle est demandée en dernière ligne d'une cellule. Dans l'exemple suivant, ça me marche pas :

[23]:
```
A = 10
A
B = 2
```

Pour plus de modularité, on utilisera la fonction **print**.

[24]:
```
A = 10
print('A =', A)  # on affiche un message (entre guillemets) puis A
B = 2
print('B =', B)
```

```
A = 10
B = 2
```

2.1 Exercice 4 - échange de deux variables

On considère deux variables a et b et on souhaite échanger leurs valeurs.

Si a = 2 et b = 5 au départ, on doit avoir a = 5 et b = 2 à la fin.

Votre programme doit fonctionner pour toutes valeurs de a et b.

[25]:
```
a = 2 # exemple, tester aussi d'autres valeurs
b = 5 # exemple, tester aussi d'autres valeurs

temp = a # on stocke la valeur de a
a = b    # on modifie a
b = temp # b = ancienne valeur de a

print("a =", a)
print("b =", b)
```

```
a = 5
b = 2
```

3 Définir et utiliser une fonction

Une **fonction** est définie de cette façon :

```
def nom_fonction(argument1, argument2, etc.):
    bloc_instructions
    return valeur_renvoyée_1, valeur_renvoyée_2, etc.
```

- Attention à l'**indentation** et aux **deux points** à la fin de la première ligne.
- **nom_fonction** est le nom de la fonction qui sera utilisé ensuite. Pas d'espace, pas de caractère spécial. On utilisera des noms de fonctions explicites.
- **argument1, argument2,...** sont les paramètres d'entrées. Ce sont les variables de la fonctions. Ne pas mettre de valeur ici, on garde des noms génériques.
- Le mot clé **return** est essentiel. Il permet de définir le résultat final. L'exécution d'une fonction s'arrête dès que l'on tombe sur return et la fonction renvoie ce résultat.

[26]:
```
def fonction_f(x) :
    y = 3*x**2 - 2*x + 1
    return y
```

Pour **faire appel** à cette fonction, on donnera son nom et la valeur des paramètres d'entrée (ici x).

[27]:
```
fonction_f(2)
```

[27]:
```
9
```

3.1 Exercice 5

Définir une fonction renvoyant le discriminant de l'équation $ax^2 + bx + x = 0$.

Les paramètres d'entrée seront a,b et c.

[28]:
```
def discriminant(a,b,c):
    return b**2 - 4*a*c
```

Tester votre fonction en calculant les discriminants de :

- $2x^2 - 4x + 5 = 0$
- $x^2 + 6x + 9 = 0$

[29]:
```
discriminant(2,-4,5)
```

[29]:
```
-24
```

[30]:
```
discriminant(1,6,9)
```

[30]:
```
0
```