

TP 2 : découverte de Python (suite)

0.1 Mise en route

1. Allumez l'ordinateur et ouvrez votre session.
2. Ouvrez l'application Edupython.
3. Au fur et à mesure du TP, vous ferez un compte rendu de vos observations et de vos réponses sur une feuille. Cette feuille sera **ramassée** par l'enseignant à l'issu du TP.
4. Il peut être utile d'avoir sous les yeux le compte rendu et le sujet du TP $n^{\circ}1$.

0.2 Rappels sur la notion d'affectation et l'utilisation de variables

Objectif $n^{\circ}1$: *Vérifier que vous maîtrisez bien les notions abordées dans le TP $n^{\circ}1$.*

Exercice 1. Voici un programme de calculs : étant donnés deux nombres entiers a et b ,

Étape 1 : multiplier le premier par le nombre 2;

Étape 2 : multiplier le second par le nombre 3;

Étape 3 : ajouter les deux résultats obtenus;

Étape 4 : afficher le dernier résultat.

1. Que donne ce programme de calculs si $a = 4$ et $b = 7$? Inscrivez les résultats intermédiaires puis le résultat final sur votre compte rendu.
2. L'objectif de cette question est de **traduire le programme de calculs** précédent sous la forme d'un **algorithme**. Dans un premier temps, les notations employées seront celles de *scratch* (le langage étudié au collège). Cette traduction a été entamé ci-dessous :

```
a ← 4
b ← ...
x ← 2 * a
y ← ...
z ← x + ...
```

Renvoyer ...

- (a) Quelle valeur faut-il affecter à la variable b ?
- (b) A quelle étape du programme de calculs correspond la 3ème ligne de l'algorithme ?
- (c) Compléter la ligne 4.

- (d) A quelle étape du programme de calculs correspond la 4ème ligne de l’algorithme ? Compléter cette ligne.
- (e) Quelle variable doit renvoyer l’algorithme pour que l’utilisateur connaisse le résultat du programme de calculs ?
3. A présent nous allons écrire l’algorithme de la question précédente **en langage python**
- (a) Compléter les lignes ci-dessous :

```
def      programme_de_calculs() :
        a = 4
        b ...
        ...
        ...
        ...
        ...
        return(z)
```

Remarque : sous python, la commande renvoyer devient return ; les instructions du programme sont alignées les unes au dessous des autres.

- (b) Ouvrez un nouveau fichier et enregistrez-le dans votre dossier personnel sous le nom **TP2 exo1 NOM** ; écrire le programme de question précédente dans l’éditeur et l’enregistrer.
- (c) Tester votre programme et comparer avec les réponses obtenues à la question 1.

0.3 Les fonctions

En informatique, il est possible de créer des **fonctions**. Celles-ci s’apparentent à des petits programmes et sont souvent utilisées à l’intérieur d’un programme plus complexe et plus vaste. Ces fonctions sont désignées par un nom choisi par le programmeur et utilisent **zéro, une ou plusieurs variables**.

Lors de l’élaboration de fonctions, certains points sont essentiels :

- le mot **def**, le **nom** de la fonction (sans espace) et la présence des deux points « : »,
- l’**alignement** des instructions (sans cela, l’ordinateur ne comprendra pas).

Objectif n°2 : Découvrir comment créer des programmes utilisant plusieurs variables.

- Exercice 2.*
1. Ouvrez un nouveau fichier et enregistrez-le dans votre dossier personnel sous le nom **TP2 exo2 NOM**.
 2. Dans l’éditeur, taper les commandes ci-dessous. **Attention à l’alignement**. Appuyer sur la flèche verte pour exécuter le programme.

```
def somme_carres(a,b):  
    somme=a**2+b**2  
    return somme
```

3. Voyons comment utiliser ce programme.

- Taper l'instruction `somme_carres(10,2)` dans la **console** Python. Quelle valeur est renvoyée par l'ordinateur ?
- Tester la fonction avec les valeurs 3 et 4. Quel résultat obtenez-vous ?
- Combien de paramètres possède cette fonction ? (i.e. de combien de valeurs le programme a-t-il besoin pour renvoyer un résultat ?).
- (bonus) Modifier la fonction `somme_carres` afin qu'elle puisse calculer la longueur de l'hypoténuse d'un triangle rectangle à partir des longueurs a et b des deux autres côtés.

Indications :

- La nouvelle fonction pourra s'appeler `hypotenuse(a,b)` et elle doit renvoyer la valeur 5 si $a = 3$ et $b = 4$.
- Il faudra sûrement utiliser la commande `from math import sqrt` afin d'utiliser la fonction racine carrée.

Comme en cours de mathématiques, il est possible de définir des fonctions avec python.

Exercice 3. Considérons la fonction

```
def exercice4(x) :  
    y=x**2-x+1  
    return y
```

- Combien de paramètres cette fonction possède-t-elle ? Le(s)quel(s) ?
- Ouvrez un nouveau fichier et enregistrez-le dans votre dossier personnel sous le nom **TP2 exo3 NOM**. Recopiez les commandes de la fonction `exercice4` dans ce fichier et enregistrez-le.
- Quel résultat fournit la fonction `exercice4` lorsque l'on tape les instructions suivantes dans la console
 - `exercice4(2)` ?
 - `exercice4(0)` ?
 - Le programme `exercice4` est associée à une fonction f . Compléter la formule $f(x) = \dots$

Voyons comment construire un programme en complétant quelques instructions.

Exercice 4. Le but de cet exercice est de créer une fonction *vitesse* qui retourne la vitesse (en km/h) lorsque l'utilisateur donne une distance en **kilomètre** et une durée en **heure**.

1. Ouvrez un nouveau fichier et enregistrez-le dans votre dossier personnel sous le nom **TP2 exo4 NOM**.
2. Recopier et compléter dans **l'éditeur** le programme ci-dessous :

```
def vitesse( , ):  
    return
```

3. Tester votre programme en tapant dans la **console** *vitesse(140,2)*. Quel résultat est renvoyé ? Est-ce cohérent ?

Travailler l'algorithmie chez soi.

Si vous souhaitez vous exercer chez vous et terminez vos TP, plusieurs solutions sont envisageables.

1. Il est possible de **télécharger le logiciel** utilisé en classe sur le site *edupython*.
2. Il est possible d'utiliser **un éditeur de programme en ligne** pour vous entraîner :
 - il suffit de s'inscrire (« sign in ») sur le site **<https://trinket.io>**;
 - puis de cliquer sur l'onglet bleu « *new trinket* » pour ouvrir la console de programmation ;
 - une fois les instructions écrites, cliquer sur le bouton « *run* » pour exécuter votre programme.