Produit scalaire & Python

TI-82 Advanced
Edition Python

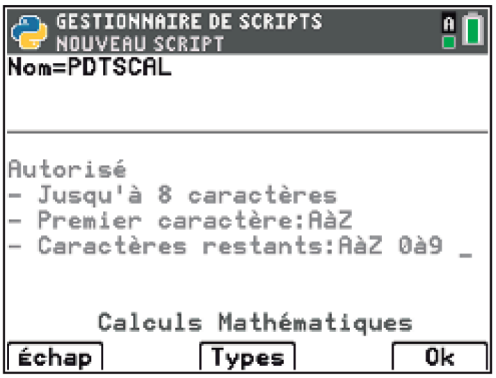
TI-83 Premium CE
Edition Python

Enoncé

1. Créer un script **Python**, de type **Calculs Mathématiques**, nommé **PDTSCAL**. Les fonctions des questions suivantes seront codées dans ce script.
2. Connaissant les coordonnées de 2 vecteurs $\vec{u}\begin{pmatrix}x_{\vec{u}}\\y_{\vec{u}}\end{pmatrix}$ et $\vec{v}\begin{pmatrix}x_{\vec{v}}\\y_{\vec{v}}\end{pmatrix}$, écrire une fonction **Python**, nommée **psvec**, prenant pour arguments les valeurs de **xu**, **yu**, **xv** et **yv**, et retournant en sortie la valeur du produit scalaire des vecteurs $\vec{u}$ et $\vec{v}$. Tester cette fonction.
3. Connaissant 2 points $A(x_A\,;y_A)$ et $B(x_B\,;y_B)$, écrire une fonction **Python**, nommée **vec**, prenant pour arguments les valeurs **xa**, **ya**, **xb** et **yb**, et retournant en sortie les coordonnées du vecteur $\overrightarrow{AB}$. Tester cette fonction.
4. Connaissant 4 points $A(x_A\,;y_A)$, $B(x_B\,;y_B)$, $C(x_C\,;y_C)$ et $D(x_D\,;y_D)$ écrire une fonction **Python**, nommée **pspts**, prenant pour arguments les valeurs **xa**, **ya**, **xb**, **yb**, **xc**, **yc**, **xd** et **yd**, et retournant en sortie la valeur du produit scalaire des vecteurs $\vec{u}$ et $\vec{v}$. Cette fonction **pspts** fera appel aux 2 fonctions précédentes. Tester cette fonction.

1. Création du script

- On commence par aller dans le module **Python** de la calculatrice, en appuyant sur [prgm] et en sélectionnant **2:Python App**.
- Une fois dans le module Python, on créé un nouveau script, à l'aide **Nouv** (touche **f3**). On saisit alors le nom du script, **PDTSCAL**.
- A l'aide de la touche **f3**, on sélectionne le type **2:Calculs Mathématiques** et on valide par **Ok** (touche **f5**).

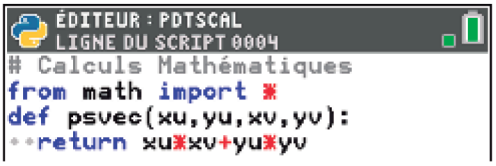


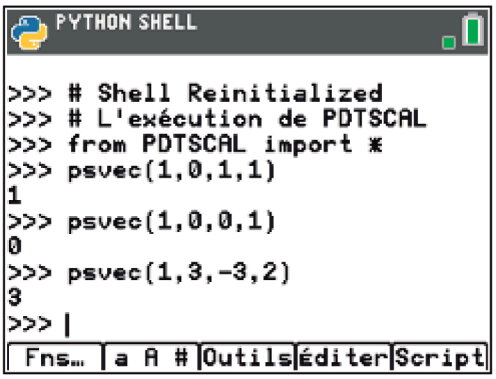
2. Fonction psvec

On rappelle l'expression analytique du produit scalaire :

$$\vec{u}\cdot\vec{v} = x_{\vec{u}}\times x_{\vec{v}} + y_{\vec{u}}\times y_{\vec{v}}$$

- Les commandes nécessaires à toute fonction **Python** se trouvent dans le menu **Fns…** (touche **f1**), dans le 1er onglet nommé **Fonc**.
- Vous trouverez ci-contre le script de cette fonction. Une fois la saisie terminée, on exécute le script via l'onglet **Exéc** (touche **f4**).
- On n'oublie évidemment pas de tester cette fonction, avec des valeurs simples qui permettent de vérifier aisément l'exactitude des résultats obtenus. Notre fonction est disponible via la touche [var] de la calculatrice.





Produit scalaire & Python

3. Fonction **vec**

On rappelle l'expression des coordonnées du vecteur $\overrightarrow{AB}$:

$$\overrightarrow{AB}\begin{pmatrix} x_B - x_A \\ y_B - y_A \end{pmatrix}$$

On procède de la même manière que pour la fonction précédente et on effectue différents tests. Attention toutefois à ne pas coder cette 2nde fonction à l'intérieur de la 1ère ; c'est l'indentation qui vous l'indique.

A noter que cette fonction retourne un couple de coordonnées. Si l'on veut accéder à une de ces 2 coordonnées (ce qui sera utile pour la question suivante), il faudra saisir (si on nomme **coord** le résultat renvoyé par notre fonction) :

- **coord[0]** pour la 1ère coordonnée ;
- **coord[1]** pour la 2nde.

```
ÉDITEUR : PDTSCAL
LIGNE DU SCRIPT 0007
# Calculs Mathématiques
from math import *
def psvec(xu,yu,xv,yv):
  return xu*xv+yu*yv

def vec(xa,ya,xb,yb):
  return xb-xa,yb-ya
```

```
PYTHON SHELL
>>> # Shell Reinitialized
>>> # L'exécution de PDTSCAL
>>> from PDTSCAL import *
>>> coord=vec(1,3,5,-2)
>>> coord
(4, -5)
>>> coord[0]
4
>>> coord[1]
-5
>>>
Fns… | a A # | Outils | éditer | Script
```

4. Fonction **pspts**

Pour cette dernière fonction, il convient de récapituler les étapes, avant de coder proprement dit :

a) Les arguments de notre fonction sont les 8 coordonnées de nos 4 points.

b) A l'aide de ces 8 coordonnées, et de la fonction **vec**, on fabrique nos 2 vecteurs, que l'on peut nommer **vec1** et **vec2**, par exemple.

c) Une fois nos 2 vecteurs créés à partir des coordonnées des points, on peut utiliser notre fonction **psvec**, en faisant attention de bien utiliser les valeurs, et non les couples de coordonnées.

d) On retourne le produit scalaire en sortie.

Vous trouverez ci-contre le script complet de l'exercice.

Comme d'habitude, il est nécessaire de tester cette dernière fonction avec différentes valeurs et de vérifier que les résultats retournés sont cohérents.

Cette façon de travailler, en divisant le travail à l'aide de « petites » fonctions très ciblées, se retrouve fréquemment en mathématiques et en informatique.

« Diviser pour régner » n'a rien de néfaste dans notre matière, bien au contraire !

```
ÉDITEUR : PDTSCAL
LIGNE DU SCRIPT 0001
# Calculs Mathématiques
from math import *
def psvec(xu,yu,xv,yv):
  return xu*xv+yu*yv

def vec(xa,ya,xb,yb):
  return xb-xa,yb-ya

def pspts(xa,ya,xb,yb,xc,yc,xd,y
    d):
  vec1=vec(xa,ya,xb,yb)
  vec2=vec(xc,yc,xd,yd)
  ps=psvec(vec1[0],vec1[1],vec2[
    0],vec2[1])
  return ps
Fns… | a A # | Outils | Exéc | Script
```

```
PYTHON SHELL
>>> # Shell Reinitialized
>>> # L'exécution de PDTSCAL
>>> from PDTSCAL import *
>>> pspts(0,0,1,0,-2,3,-1,4)
1
>>> pspts(0,0,1,3,5,1,2,3)
3
>>> pspts(1,0,5,0,0,2,-2,4)
-8
>>>
Fns… | a A # | Outils | éditer | Script
```