

ProfLycee (4.00b), Piton et Pyluatex

NB : il est possible, suivant la version de piton installée, que le rendu ne soit pas *strictement identique* (au niveau des marges par exemple).

Ce très léger dysfonctionnement n'affecte pas le *bon fonctionnement* des différents codes.

1 Code « Piton », indépendant de Pyluatex

1.1 Préambule basique

```
\documentclass[french,a4paper,10pt]{article}
\usepackage{ProfLycee}
\useproflyclub{piton} % lua
```

1.2 Exemples

```
%Sortie par défaut
\begin{CodePiton}{}
def valeur_absolue(x) :
    "Renvoie la valeur absolue de x"
    #le petit test qui va bien
    if x > 0 :
        return x
    else:
        return -x
\end{CodePiton}
```

</> Code Python

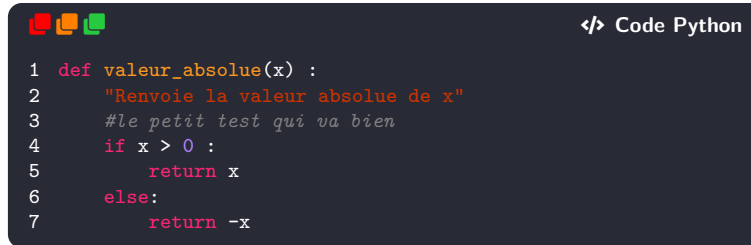
```
1 def valeur_absolue(x) :
2     "Renvoie la valeur absolue de x"
3     #le petit test qui va bien
4     if x > 0 :
5         return x
6     else:
7         return -x
```

```
%Sortie avec style Classique, Largeur=10cm
\begin{CodePiton}[Largeur=10cm,CouleurNombres=olive]{}
def valeur_absolue(x) :
    "Renvoie la valeur absolue de x"
    #le petit test qui va bien
    if x > 0 :
        return x
    else:
        return -x
\end{CodePiton}
```

</> Code Python

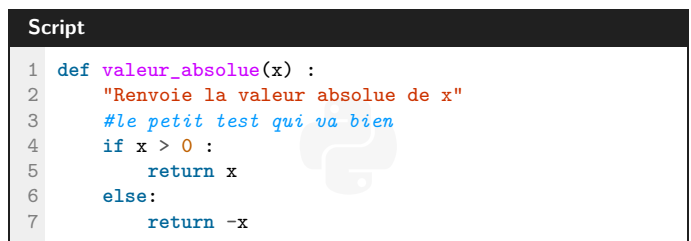
```
1 def valeur_absolue(x) :
2     "Renvoie la valeur absolue de x"
3     #le petit test qui va bien
4     if x > 0 :
5         return x
6     else:
7         return -x
```

```
%Sortie avec Style=Carbon,Largeur=10cm, centré
\begin{CodePiton}[Style=Carbon,Largeur=10cm,Alignement=center]{}
def valeur_absolue(x) :
    "Renvoie la valeur absolue de x"
    #le petit test qui va bien
    if x > 0 :
        return x
    else:
        return -x
\end{CodePiton}
```



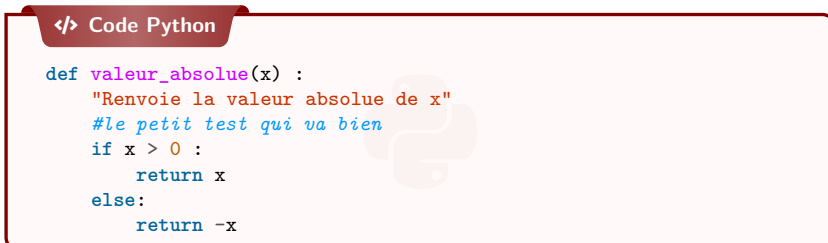
```
Code Python
1 def valeur_absolue(x) :
2     "Renvoie la valeur absolue de x"
3     #le petit test qui va bien
4     if x > 0 :
5         return x
6     else:
7         return -x
```

```
%Sortie avec Style=Classique, Largeur=0.5\linewidth, aligné à droite, sans Cadre, avec Filigrane
\begin{CodePiton}%
    [Largeur=0.5\linewidth,Cadre=false,Alignement=flush right,Filigrane,Titre={Script}]{}
#environnement piton avec numéros de ligne, pleine largeur, style moderne
def valeur_absolue(x) :
    "Renvoie la valeur absolue de x"
    #le petit test qui va bien
    if x > 0 :
        return x
    else:
        return -x
\end{CodePiton}
```



```
Script
1 def valeur_absolue(x) :
2     "Renvoie la valeur absolue de x"
3     #le petit test qui va bien
4     if x > 0 :
5         return x
6     else:
7         return -x
```

```
%Sortie Onglet, Couleur rouge, Largeur=11cm, avec Filigrane, aligné à gauche, sans ligne
\begin{CodePiton}[Style=Onglet,Couleur=red,Largeur=11cm,Filigrane,Alignement=flush left,Lignes=false]{}
def valeur_absolue(x) :
    "Renvoie la valeur absolue de x"
    #le petit test qui va bien
    if x > 0 :
        return x
    else:
        return -x
\end{CodePiton}
```



```
Code Python
def valeur_absolue(x) :
    "Renvoie la valeur absolue de x"
    #le petit test qui va bien
    if x > 0 :
        return x
    else:
        return -x
```

2 Console « Piton », dépendant de Pyluatex

2.1 Préambule, avec le package pyluatex

```
\documentclass[french,a4paper,10pt]{article}
\usepackage{ProfLycee}
\usepackage{ProfLycee}
\usepackage[executable=python]{pyluatex} % lua + shell-escape
```

2.2 Commande

```
\begin{ConsolePiton}[Options piton]<Clés>{Options tcbox}
...
...
\end{ConsolePiton}
```

Les clés, à placer entre <...>, sont :

- **<AltStyle>** (booléen) pour modifier le style de la console
- **<Logo>** pour afficher un petit logo dans les *titres* de la console REPL; défaut : **<true>**
- **<Largeur>** pour spécifier la largeur de la console REPL; défaut : **<\linewidth>**
- **<Alignement>** pour spécifier l'alignement de la console REPL. défaut : **<flush left>**

2.3 Exemples

```
%Déclaration d'une fonction python + librairie random pour utilisation ultérieure
\begin{python}
from random import randint

def valeur_absolue(x) :
    "Renvoie la valeur absolue de x"
    #le petit test qui va bien
    if x > 0 :
        return x
    else:
        return -x
\end{python}
```

```
\begin{ConsolePiton}{}
1+1
2**10
valeur_absolue(-3)
valeur_absolue(0)
valeur_absolue(5)
print(f"La valeur absolue de 5 est {valeur_absolue(5)}")
print(f"La valeur absolue de -4 est {valeur_absolue(-4)}")
\end{ConsolePiton}
```

◆ Début de la console Python ◆

```
>>> 1+1
2
>>> 2**10
1024
>>> valeur_absolue(-3)
3
>>> valeur_absolue(0)
0
>>> valeur_absolue(5)
5
>>> print(f"La valeur absolue de 5 est {valeur_absolue(5)}")
La valeur absolue de 5 est 5
>>> print(f"La valeur absolue de -4 est {valeur_absolue(-4)}")
La valeur absolue de -4 est 4
```

◆ Fin de la console Python ◆

```

\begin{ConsolePiton}<Largeur=11cm,Alignement=center,Logo=false>{}
1+1
2**10
valeur_absolue(-3)
valeur_absolue(0)
valeur_absolue(5)
print(f"La valeur absolue de 5 est {valeur_absolue(5)}")
print(f"La valeur absolue de -4 est {valeur_absolue(-4)}")
liste = [randint(1,20) for i in range(10)]
print(liste)
print(max(liste), min(liste), sum(liste))
\end{ConsolePiton}

```

----- Début de la console Python -----

```

>>> 1+1
2
>>> 2**10
1024
>>> valeur_absolue(-3)
3
>>> valeur_absolue(0)
0
>>> valeur_absolue(5)
5
>>> print(f"La valeur absolue de 5 est {valeur_absolue(5)}")
La valeur absolue de 5 est 5
>>> print(f"La valeur absolue de -4 est {valeur_absolue(-4)}")
La valeur absolue de -4 est 4
>>> liste = [randint(1,20) for i in range(10)]
>>> print(liste)
[12, 18, 8, 16, 12, 10, 20, 17, 9, 19]
>>> print(max(liste), min(liste), sum(liste))
20 8 141

```

----- Fin de la console Python -----

```

\begin{ConsolePiton}<AltStyle,Largeur=10cm,Alignement=center>{}
[i**2 for i in range(50)]
\end{ConsolePiton}

```

----- Début de la console Python -----

```

>>> [i**2 for i in range(50)]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144, 169, 196,
225, 256, 289, 324, 361, 400, 441, 484, 529, 576, 625, 676,
729, 784, 841, 900, 961, 1024, 1089, 1156, 1225, 1296, 1369,
1444, 1521, 1600, 1681, 1764, 1849, 1936, 2025, 2116, 2209,
2304, 2401]

```

----- Fin de la console Python -----

3 Présentation, et exécution, comme avec Thonny

3.1 Préambule, avec le package pyluatex

```
\documentclass[french,a4paper,10pt]{article}
\usepackage{ProfLycee}
\useproflyclib{piton}
\usepackage[executable=python]{pyluatex} % lua + shell-escape
```

3.2 Commandes

```
\begin{PitonThonnyEditor}[clés]<options tcbox><options linie numebrs piton>
...
\end{PitonThonnyEditor}
```

```
\begin{PitonThonnyConsole}[clés]<options tcbox><options piton>
...
\end{PitonThonnyConsole}
```

3.3 Exemples

```
\begin{python}
from math import gcd

def est_duffy(n) :
    nb_div, somme_div = 0, 0
    for i in range(1, n+1) :
        if n % i == 0 :
            nb_div += 1
            somme_div += i
    if gcd(somme_div, n) == 1 :
        return True
    else :
        return False

\end{python}
```

```
\begin{PitonThonnyEditor}[NomFichier=tpcapytale.py,Largeur=12cm]{ }
#PROJET CAPYTALE
from math import gcd

def est_duffy(n) :
    nb_div = 0
    somme_div = 0
    for i in range(1, n+1) :
        if n % i == 0 :
            nb_div += 1
            somme_div += i
    if gcd(somme_div, n) == 1 :
        return True
    else :
        return False
\end{PitonThonnyEditor}
```

tpcapytale.py ×

```
1 #PROJET CAPYTALE
2 from math import gcd
3
4 def est_duffy(n) :
5     nb_div = 0
6     somme_div = 0
7     for i in range(1, n+1) :
8         if n % i == 0 :
9             nb_div += 1
10            somme_div += i
11     if gcd(somme_div, n) == 1 :
12         return True
13     else :
14         return False
```

```
\begin{PitonThonnyConsole}[IntroConsole={python 3.8.10},Largeur=12cm]{ }
#Run tpcapytale.py
est_duffy(6)
est_duffy(13)
est_duffy(265)

from random import randint
nb = randint(1,100000)
nb, est_duffy(nb)
\end{PitonThonnyConsole}
```

console ×

```
python 3.8.10
>>> #Run tpcapytale.py
>>> est_duffy(6)
False
>>> est_duffy(13)
True
>>> est_duffy(265)
True
>>>
>>> from random import randint
>>> nb = randint(1,100000)
>>> nb, est_duffy(nb)
(6906, False)
```

```
\begin{PitonThonnyConsole}[Largeur=8cm]{ }
[i**2 for i in range(50)]
\end{PitonThonnyConsole}
```

console ×

```
Python 3.11.6 /usr/bin/python
>>> [i**2 for i in range(50)]
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121,
144, 169, 196, 225, 256, 289, 324, 361, 400,
441, 484, 529, 576, 625, 676, 729, 784, 841,
900, 961, 1024, 1089, 1156, 1225, 1296, 1369,
1444, 1521, 1600, 1681, 1764, 1849, 1936, 2025,
2116, 2209, 2304, 2401]
```

4 Via lecture de fichiers

4.1 Création des fichiers, directement dans le document

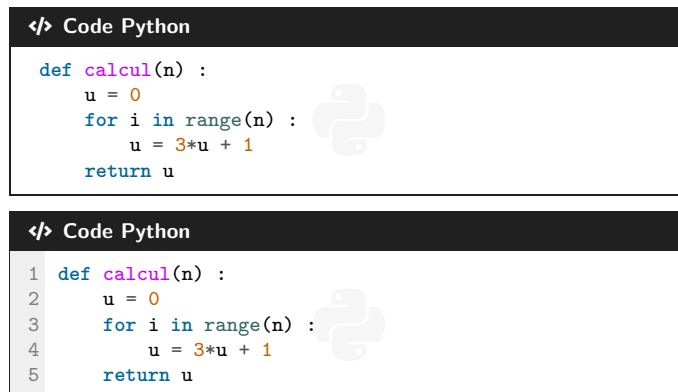
```
\begin{scontents}[overwrite,write-out=testscript.py]
def calcul(n) :
    u = 0
    for i in range(n) :
        u = 3*u + 1
    return u
\end{scontents}

\begin{scontents}[overwrite,write-out=testpcode.txt]
Algorithme : calcul mental
Variables : nb1, nb2, resultat_saisi

Début
    nb1 ← randint (1, 10)
    nb2 ← randint (1, 10)
    Afficher(nb1 , " * ", nb2 , " = ? ")
...
Fin
\end{scontents}
```

4.2 Code python

```
\CodePitonFichier[Largeur=9cm,Filigrane,Lignes=false]{testscript.py}
\CodePitonFichier[Largeur=9cm,Filigrane]{testscript.py}
```

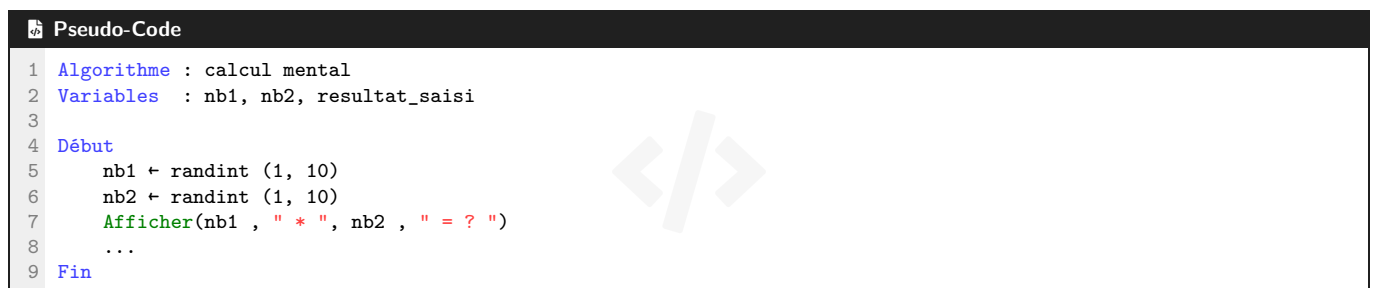


```
</> Code Python
def calcul(n) :
    u = 0
    for i in range(n) :
        u = 3*u + 1
    return u

</> Code Python
1 def calcul(n) :
2     u = 0
3     for i in range(n) :
4         u = 3*u + 1
5     return u
```

4.3 Pseudo-code

```
\PseudoCodePitonFichier[Largeur=\linewidth,Filigrane]{testpcode.txt}
\PseudoCodePitonFichier[Largeur=\linewidth,Filigrane,Lignes=false]{testpcode.txt}
```



```
Pseudo-Code
1 Algorithme : calcul mental
2 Variables : nb1, nb2, resultat_saisi
3
4 Début
5     nb1 ← randint (1, 10)
6     nb2 ← randint (1, 10)
7     Afficher(nb1 , " * ", nb2 , " = ? ")
8     ...
9 Fin
```

Pseudo-Code

```
Algorithme : calcul mental
Variables : nb1, nb2, resultat_saisi

Début
    nb1 ← randint (1, 10)
    nb2 ← randint (1, 10)
    Afficher(nb1 , " * ", nb2 , " = ? ")
    ...
Fin
```

4.4 Version Thonny

\PitonThonnyEditorFichier[Largeur=9cm]{testscript.py}

script.py ×

```
1 def calcul(n) :
2     u = 0
3     for i in range(n) :
4         u = 3*u + 1
5     return u
```