

TYPE

Python est un langage intuitif et très facile de prise en main. Il est idéal pour des débutants comme pour les utilisateurs plus avancés. Nous ne pouvons dans le cadre de cet atelier faire « un cours de python » car cela prendrait un module entier en soi. Cependant, nos besoins sont faibles et en utilisant 5% de ce qui existe en python, on peut faire déjà énormément de chose.

En python, vous n'avez pas à préciser le type des variables lors des déclarations :

```
a = 5    # a un entier
b = 4.3  # b est un double
```

Attention aux résultats des opérations :

```
a / 2 => 2.5    # par défaut ; python convertit dans un type plus fort si besoin
a // 2 => 2      # résultat imposé en entier
b = int(a/2)    # résultat en double et conversion en entier
```

Testez votre premier programme (aucun espace en début de lignes, svp) :

```
a = 4
b = 5
c = a + b
print(c)
```

Vous remarquez qu'il n'y a pas non plus de ";" à la fin des lignes. Appuyez sur F5 pour lancer le programme.

```
>>> (executing file "<tmp 1>")
9
```

INDENTATION

ATTENTION il n'y a pas de système d'accolades en python { ... }

Python a choisi de mettre des espaces en début de ligne pour déterminer les blocs. S'il n'y a pas d'espaces sur la gauche de vos lignes, nous sommes dans le bloc principal.

Ajoutez un espace sur la gauche du print(c) et appuyez sur F5 pour lancer le programme. Vous obtenez le message d'erreur suivant :

```
File "<tmp 1>", line 4
  print(c)
  ^
IndentationError: unexpected indent
```

Python vous signale qu'un espace/tabulation (=indent en anglais) a été trouvé et que ce décalage ne correspond pas à ce qu'il attendait.

BOUCLE FOR

Le grand classique de l'informatique !! La syntaxe est la suivante :

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

Remarquez la présence d'un symbole ":" à la fin de la ligne for. Il indique le début du bloc for et correspond à une accolade ouvrante. L'instruction print fait partie du bloc du for, elle est donc ici décalée de **TROIS ESPACES**. Vous pouvez en mettre 1 ou 2 seulement, mais il faudra rester cohérent dans l'ensemble de votre programme. Les trois espaces sont un usage.

Appuyez sur F5 pour lancer le programme suivant.

```
>>> (executing file "<tmp 1>")
0
1
2
3
4
5
6
7
8
9
```

L'instruction range(10) indique à la variable de boucle i qu'elle va varier entre 0 et 9 (10 non compris). Testez ce nouveau code :

```
for i in range(2,10,3):
    print(i)
```

```
>>> (executing file "<tmp 1>")
2
5
8
```

Ici, la variable i commence à 2, elle ne devra pas dépasser 10 et avance de 3 en 3.

CONDITION IF

Veuillez tester le code suivant :

```
x = 5
if ( x % 2 == 1 ):
    print("impair")
```

Comme pour l'instruction FOR, vous remarquez la présence d'un symbole ":" à la fin de la ligne du if. Pour indiquer les instructions faisant partie du bloc IF, on indente l'instruction print. Appuyez sur F5 pour lancer le code :

```
>>> (executing file "<tmp 1>")
impair
```

On trouve la structure habituelle if/else:

```
x = 4
if ( x % 2 == 1 ):
    print("impair")
else:
    print("pair")
```

Appuyez sur F5 pour lancer le code :

```
>>> (executing file "<tmp 1>")
pair
```

On trouve aussi les conditions multiples :

```
x = 2
if ( (x ==1) or (x==2) ):
    print("1 ou 2")
```

Appuyez sur F5 pour lancer le code :

```
>>> (executing file "<tmp 1>")
1 ou 2
```

FONCTION

Le bloc de code associé à une fonction est identifié en utilisant l'indentation. Si une instruction for ou if se trouve dans le bloc, alors les instructions propres au bloc if/for doivent être indentées une deuxième fois.

```
def somme(a,b):
    res = a+b
    if (a == 5):
```

```
    res += 5
    return res


print(somme(4,6))
print(somme(5,10))
```

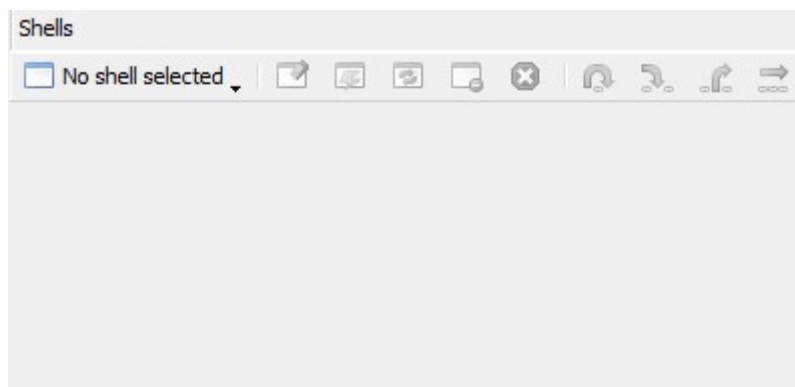
Le mot clef « def » permet de définir une fonction nommée ici « somme ». Remarquez que les paramètres x et y transmis sont non typés ! La ligne « res += 5 » est une instruction propre au if, elle est donc indentée une deuxième fois.

Appuyez sur F5 pour tester le code :

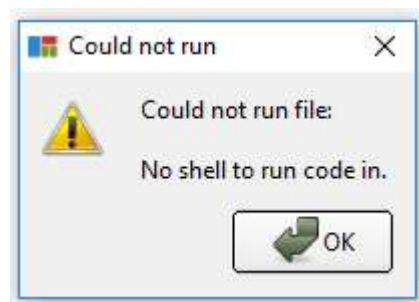
```
>>> (executing file "aa.py")
10
20
```

SHELL

Parfois, cela arrive, plus rien ne marche et l'appui sur F5 ne fournit aucun résultat. L'environnement vient de lâcher, il faut pouvoir relancer. Pour simuler ce problème, nous allons arrêter l'environnement courant en cliquant sur l'icône:  Voici le résultat, le SHELL python est désactivé :



Si vous cliquez sur F5, le message d'erreur suivant apparaît :

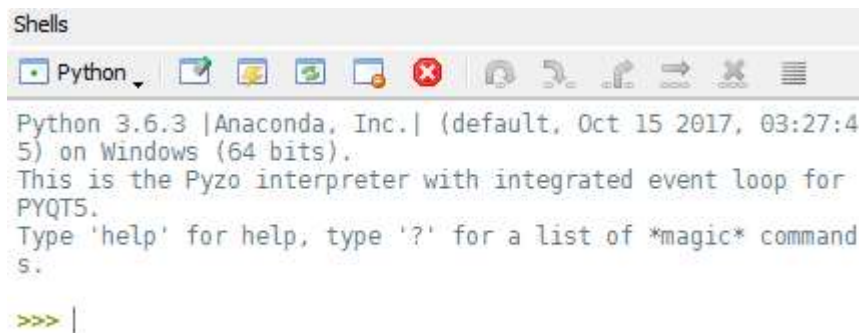


Pour pouvoir travailler, PyZO lance en tâche de fond un shell Python qui va exécuter votre programme ligne à ligne. Un shell est une interface où l'on peut rentrer des instructions depuis le clavier. Si le shell

python plante, PyZO ne peut plus rien exécuter. Cliquez sur le bouton shell → New shell → Create shell :



Une fois activée, la fenêtre de SHELL apparaît :



Vous pouvez vous aussi vous servir du shell pour tester des commandes. Effectuez les tests suivants :

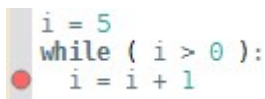


DEBUG

Lorsque vous aurez un problème, il faudra utiliser cet outil pour déterminer la source de vos erreurs.

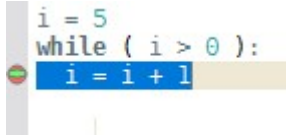
```
i = 5
while ( i > 0 ):
    i = i + 1
```

Oui, il s'agit d'une boucle infinie ! Cliquez sur le bord gauche de la ligne 3 pour positionner un point d'arrêt appelé breakpoint. Il apparaît sous la forme d'un rond rouge.



```
i = 5
while ( i > 0 ):
    i = i + 1
```

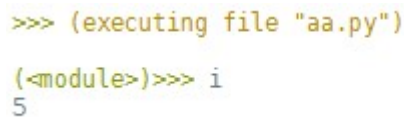
Lancez le programme en appuyant sur F5. PyZO va stopper l'exécution dès qu'il arrive sur la ligne 3 :



```
i = 5
while ( i > 0 ):
    i = i + 1
```

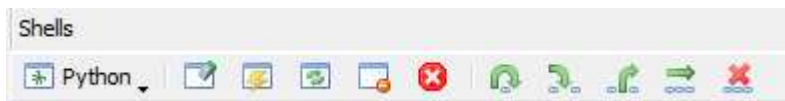
PyZO vous informe que le programme vient d'être stoppé. Il indique là où il a stoppé en positionnant une ligne bleue et en montrant un trait vert sur la gauche. Remarquez que le Shell est à nouveau actif. Profitons-en pour regarder la valeur de la variable i.


Dans le shell, tapez i + ENTER :

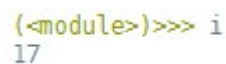


```
>>> (executing file "aa.py")
(<module>)>>> i
5
```


C'est bien la valeur que doit avoir i lorsque nous rentrons dans la boucle while. L'outil shell est puissant car vous pouvez inspecter toutes vos valeurs de programme à ce niveau. Remarquez que de nouveaux icônes sont apparus sur la droite :





Cliquez plusieurs fois sur l'icône  pour déclencher l'exécution pas à pas du programme. Affichez la valeur de la variable i dans le shell :



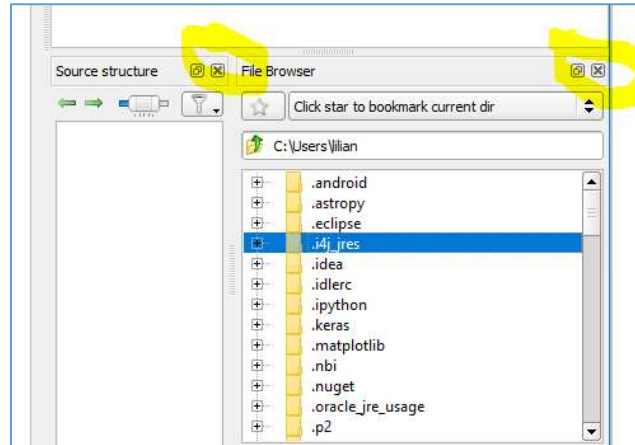
```
(<module>)>>> i
17
```

Supprimez le breakpoint en cliquant dessus. On relance le programme en mode exécution normale en cliquant sur l'icône . L'exécution reprend et le shell n'est plus interactif.

A ce niveau nous sommes coincés dans la boucle infinie, il n'y a rien à faire. Pour forcer l'arrêt du programme, appuyez sur l'icône  puis sur l'icône  pour relancer le shell python.

FENETRES

Pour gagner de la place à l'écran, vous pouvez fermer les fenêtres « Source Structure » et « File Browser » en cliquant sur la croix :



L'écran de travail est un peu plus confortable :

