TD : Bases de programmation

1) Environnement de travail

1.a) Qu'est-ce qu'un IDE ?

Lorsqu'on fait de la programmation, il est agréable d'avoir à sa disposition différents outils qui permettent de travailler plus simplement et plus efficacement. Les fonctionnalités les plus répandues sont :

- La coloration syntaxique : les mots clefs et les différents types d'objets d'un langage prennent automatiquement des couleurs différents lors de la saisie
- La compilation : un raccourci clavier et/ou un bouton permettent d'exécuter tout ou partie du code
- Le débuggeur : un mode d'exécution qui permet d'exécuter le code ligne par ligne en observant les valeurs des variables (très pratique pour identifier les erreurs)
- L'auto-complétion : lors de la saisie d'un mot, plusieurs suggestions sont proposées au développeur, comme le nom des objets existants commençant par les lettres déjà tapées, ou encore le nombre et les arguments de la fonction utilisée.

Pour répondre à ces besoins, on a créé des logiciels regroupant ces différents outils. Ces logiciels sont appelés des IDE, de l'anglais *Integrated Development Environment* (la traduction française, "environnement de développement", est peu utilisée).

1.b) Notepad++

Il existe de nombreux IDE pour programmer en Python. Pour cette année, nous utiliserons *Notepad++*, qui a l'avantage d'être léger, gratuit et assez performant.

Il constitue en outre une alternative très intéressante au bloc-notes de Windows, et permet de programmer dans un grand nombre de langages.

C'est cet IDE qui est mis à votre disposition sur les nouvelles machines du laboratoire de physique.

Si vous travaillez sur votre machine personnelle, vous trouverez un guide d'installation complet sur le support en ligne du cours :

http://andre.lovichi.free.fr/teaching/ea/2015-2016/installation_windows.html

1.c) Komodo Edit

Pour les utilisateurs de Mac, l'IDE *Komodo Edit* possède des fonctionnalités proches de celles de Notepad++.

Un guide d'installation est également disponible pour cet IDE :

http://andre.lovichi.free.fr/teaching/ea/2015-2016/installation_mac.html

1.d) Programmer en Python avec Notepad++

Question 1. Ouvrez Notepad++ et créez un nouveau fichier contenant le texte suivant :

```
x = 2
carre = x * x
print(carre)
```

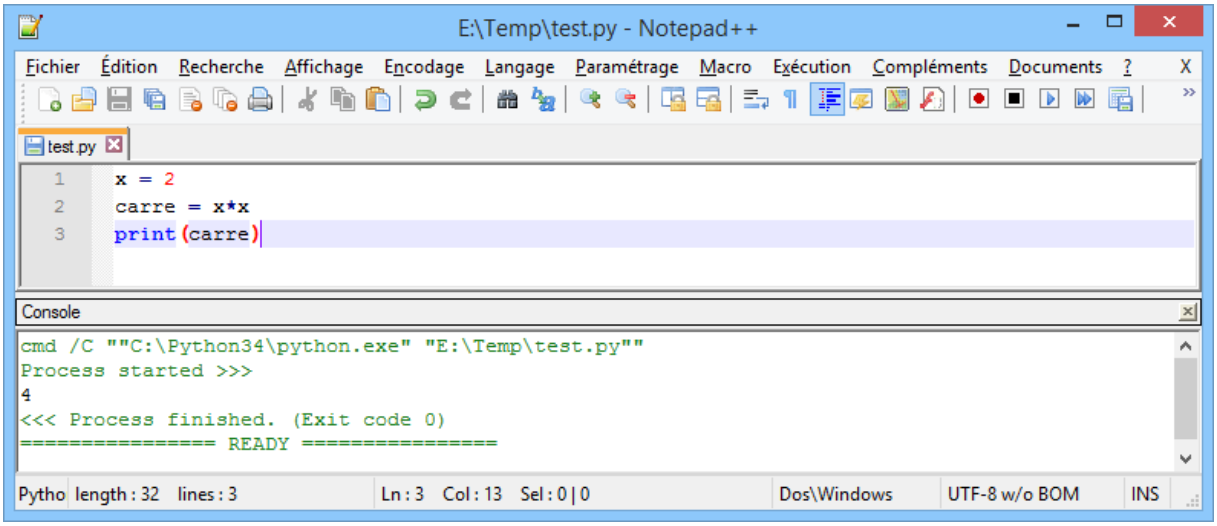
Comme vous pouvez le constater, ce nouveau fichier est nommé `new X`, et il n'est associé à aucun langage. Aucune coloration syntaxique ne s'applique : le texte est intégralement noir.

Question 2. Enregistrez ce fichier sous le nom `tp1.py` dans votre répertoire de travail.

Notepad++ déduit automatiquement de l'extension utilisée (`.py`) qu'il s'agit d'un fichier Python, et applique la coloration syntaxique correspondante.

Question 3. Utilisez la touche `F6` et validez avec `Entrée` pour exécuter votre code.

Vous devriez obtenir le résultat suivant :



Vous pouvez désormais appuyer sur `F9` pour obtenir le même résultat (ce raccourci a pour effet de ré-exécuter la dernière commande utilisée).

1.e) Avant de vous lancer

Cette question est à traiter uniquement depuis chez vous, afin de vérifier que votre environnement de travail est correctement configuré.

Question 4. Exécutez les commandes ci-dessous, et envoyez les résultats par email à l'adresse suivante : andre-lovichi@ecole-alsacienne.org

```
import sys
print(sys.version)

import os
print(os.getcwd())
print(os.getlogin())
```

Remarque : Vous pouvez traiter cette question avant d'envoyer un TD complet (dès que vous avez terminé l'installation de votre environnement de travail à domicile).

2) Manipulations des types de base

a) Afficher une valeur

Comme vous avez pu le constater dans l'exemple précédent, pour afficher une valeur dans la console, on utilise la fonction `print`.

Contrairement à Caml où vous utilisiez une fonction d'affichage pour chaque type d'objet (`print_int`, `print_string`, etc.), en Python, vous pouvez utiliser cette fonction `print` quel que soit le type d'entrée.

Question 5. Utilisez la fonction `print` pour afficher des valeurs des différents types de base : entiers, flottants, chaînes de caractère et booléens.

Remarque : En Python, les booléens `True` et `False` prennent une majuscule.

b) Commentaires

Les commentaires vous seront très utiles pour structurer votre code. Vous pouvez également vous en servir pour noter vos réponses à l'intérieur de votre code.

Pour écrire un commentaire, ajoutez simplement un symbole dièse (`#`) en début de ligne : toute la ligne (jusqu'au prochain saut de ligne) sera considérée comme un commentaire.

Question 6. Ecrivez votre premier commentaire en Python.

c) Opérateurs arithmétiques

Question 7. Pour chacun des types évoqués ci-dessus, testez chacun des opérateurs suivants : `+`, `-`, `*`, `/`, `//` et `%`. Pour chaque opérateur, indiquez si l'opération est possible, et le cas échéant à quoi elle correspond.

Question 7.b. Comment expliquez-vous les résultats obtenus avec les booléens ?

Question 8. Essayez d'utiliser ces mêmes opérateurs en mélangeant les types (par exemple en ajoutant un nombre entier et un nombre réel). Pour chacun des cas de figures autorisés par Python, indiquez le type obtenu.

Remarque : Vous pouvez vérifier vos réponses en utilisant la fonction `type`. Cette fonction permet en effet d'obtenir le type d'un objet. Vous pouvez ainsi afficher le type d'un objet en la combinant avec la fonction `print`. Par exemple, `print(type(3))` affiche `<class 'int'>`

d) Opérateurs de comparaison et d'égalité

Question 9. Testez les opérateurs suivants sur des nombres entiers pour deviner leur signification : `==`, `!=`, `<`, `>`, `<=`, `>=`

Question 9.b Testez ensuite ces opérateurs sur des chaînes de caractères. Comment expliquez-vous les résultats obtenus ?

e) Nombres réels

Question 10. Affichez les valeurs des expressions suivantes : qu'en déduisez-vous ?

- `0.1 + 0.2 == 0.3`
- `1000000000000000001 / 1 - 1000000000000000000`

3) Constructions classiques

a) Variables

Question 11 : Utilisez la syntaxe vue en cours pour effectuer les manipulations suivantes :

- Créer une variable x initialisée à 42
- Afficher la valeur de x
- Créer une variable y initialisée à x/2
- Augmenter la valeur de x de 5
- Afficher la valeur de y
- Afficher la valeur de x

b) Génération aléatoire

Comme on a pu le voir l'an dernier, il est très pratique de pouvoir générer des nombres aléatoires. En Python, il existe un module dédié à cette génération aléatoire.

On commence donc par faire appel à ce module, en écrivant la ligne suivante :
```
import random
```

Il faut ensuite initialiser le générateur aléatoire (pour éviter d'avoir toujours la même séquence), ce qui se fait avec l'instruction:
```
random.seed()
```

Il suffit ensuite de faire appel à la fonction `random.randint`, qui prend renvoie un entier tiré au hasard entre les deux valeurs entrées en argument. Par exemple, `random.randint(3, 6)` renvoie un entier tiré au hasard parmi 3, 4, 5 et 6.

Question 12 : Utilisez les fonctions ci-dessus pour générer et afficher des entiers aléatoires.

c) Conditions

Question 13 : Imaginez une séquence d'instructions effectuant les opérations suivantes :

- Choisir un nombre aléatoire entre 1 et 20 (inclus)
- Afficher ce nombre
- Afficher
  - "pair" si le nombre est pair
  - "impair" si le nombre est impair

Question 13.b. Compléter le code de la question précédente pour afficher "Au-dessus de 10" si le nombre généré est strictement supérieur à 10.

Question 14 : Imaginez une séquence d'instructions effectuant les opérations suivantes :

- Choisir un nombre aléatoire entre 0 et 3000
- Afficher ce nombre
- Afficher `True` si ce nombre correspond à une année bissextile, et `False` sinon

Pour rappel, une année bissextile est

- soit divisible par 4 et non divisible par 100
- soit divisible par 400.

d) Boucles `for`

Question 15 : Utilisez une boucle `for` pour afficher 50 fois le texte "Les boucles for, c'est pratique"

Question 16 : Utilisez des boucles `for` pour afficher les séquences de nombre suivantes :

- Entiers compris entre 0 et 10 (inclus)
- Entiers compris entre 10 et 20 (inclus)
- Entiers pairs compris entre 0 et 20 (inclus)
- Multiples de 13 compris entre 0 et 100 (inclus)
- Entiers positifs impairs inférieurs à 16, par ordre décroissant

Question 17 : Utilisez deux boucles `for` pour afficher une table de multiplication, sur le modèle ci-dessous, mais avec 10 lignes et 10 colonnes.

```
1       2       3       4       5
2       4       6       8       10
3       6       9       12      15
4       8       12      16      20
5       10      15      20      25
```

Remarque : Pour obtenir cet affichage, vous devrez faire appel à la fonction `print` de différentes manières :

- Pour afficher un nombre `x` suivi d'une tabulation, utilisez `print(x, end="\t")`
- Pour passer simplement à la ligne, utilisez `print()` (sans argument)

e) Boucles `while`

Question 18 : Utilisez une boucle `while` pour exécuter les opérations suivantes

- Choisir un entier `nombre_mystere` au hasard entre 0 et 20 (inclus)
- Afficher cet entier `nombre_mystere`
- Choisir un entier `proposition` au hasard entre 0 et 20 (inclus)
- Tant que `proposition` est différent de `nombre_mystere`,
    - Afficher `proposition`
    - Affecter une nouvelle valeur à `proposition`, choisie au hasard entre 0 et 20

Question 19 : Modifiez le code de la question précédente pour ajouter un compteur calculant le nombre de tentatives nécessaires pour trouver le nombre mystère, et afficher la valeur de ce compteur lors que le nombre est trouvé.

Question 20 : Modifiez votre code pour que le nombre mystère soit toujours strictement supérieur à 20. Quel est le problème d'un tel programme ? Comment se comporte alors Notepad++ ?