TP1 - Introduction à Python

Ce premier TP a pour objet de vous familiariser avec l'environnement de développement Pyzo ainsi que de manipuler les premiers concepts de base de tout langage de programmation.

1 Environnement de travail : les bonnes habitudes

Les éléments de cette partie sont essentiels à la bonne organisation de vos deux années de CPGE.

1.1 Répertoire réseau

Il vous a été communiqué vos identifiants pour vous connecter à une machine de l'établissement. **Vous devez les avoir à portée de main systématiquement pour chaque séance de TP.** Vous pouvez par exemple les recopier sur un papier qui sera toujours en votre possession ou les enregistrer sur votre téléphone.

Une fois connecté à une machine, vous avez accès aux applications mais aussi à votre répertoire réseau. Pour cela, utilisez l'**explorateur de fichiers** puis ouvrez votre **espace personnel** portant votre nom et prénom.

Chaque TP fera l'objet d'un fichier de code Python qu'il vous faudra enregistrer. Le plus simple est de créer un répertoire Informatique dans votre espace personnel afin d'y enregistrer tous les codes Python afin que vous puissiez reprendre vos travaux facilement lors des séances suivantes de TP et lors de vos révisions. **Pensez à sauvegarder régulièrement votre travail dans le menu Fichier (File) avec la commande Enregistrer (Save) ou avec le raccourci clavier Ctrl+S.**

1.2 Environnement Pyzo

Ce n'est pas obligatoire mais programmer en informatique demande d'utiliser un environnement de développement. C'est avant tout un logiciel qui permet de faciliter la tâche. Il en existe plein pour tous les langages. Celui que vous allez utiliser s'appelle Pyzo. Pour le lancer, allez dans la barre de recherche du menu démarrer et cherchez Pyzo.

L'ouverture de Pyzo entraîne la création d'un fichier qu'il vous faudra nommer et enregistrer en allant dans le menu **Fichier** sur **Enregistrer sous** (*Save As*).

Si vous souhaitez ouvrir un fichier déjà enregistré dans votre espace personnel, allez dans le menu **Fichier** puis cliquez sur **Ouvrir** (**Open**) et cherchez votre fichier.

L'environnement Pyzo est divisé en deux sous-fenêtres principales :

- la sous-fenêtre totalement blanche s'appelle **éditeur de code** : c'est un simple éditeur de texte avec des fonctionnalités dédiées à l'écriture de code Python ;
- la sous-fenêtre où les caractères `>>>` (ou parfois `[1]:`) sont présents s'appelle **console** : c'est une interface en mode texte qui permet d'interagir directement avec Python, le résultat est calculé et affiché immédiatement comme une calculatrice, pour peu que l'on ait donné une expression syntaxiquement valide.

Attention seul le texte se trouvant dans l'éditeur de texte est enregistré.

2 Utilisation de la console

Les questions ci-dessous sont à traiter **dans la console**, n'hésitez pas à garder une trace papier ou numérique de ce que vous allez (re-)découvrir.

Question 1. A l'aide de la console, essayez les opérateurs arithmétiques en calculant $54 + 23$, $24 - 7$, $123 * 87$ et $34/5$.

Question 2. Définir le rôle des opérateurs `//`, `%` et `**` en utilisant des opérandes entières, par exemple `7//3`.

Question 3. Peut-on indiquer plusieurs calculs différents sur une même ligne ? Si oui, quel(s) séparateur(s) mettre ? Qu'observez-vous ?

3 Variables et types simples

Les questions ci-dessous peuvent se traiter dans la console ou dans l'éditeur. Pour pouvoir garder la trace de vos travaux, placez vous dans l'éditeur. N'hésitez pas à utiliser le symbole dièse `#` pour commenter ce que vous écrivez, observez, comprenez. Les commentaires peuvent également s'écrire entre triples guillemets (`""" commentaires"""`).

Attention, l'éditeur de code n'est rien qu'un éditeur de texte. Autrement dit, si vous écrivez le texte dans l'éditeur, vous n'aurez fait qu'écrire et Python ne vous donnera pas de résultat.

Pour que Python effectue les calculs demandés, il vous faut compiler votre fichier. Pour ce faire, allez dans le menu **Compiler** (**Run**) puis sélectionnez **Compiler Fichier** (**Execute File**), ou bien utilisez le raccourci clavier **Ctrl+E** ou encore **F5**. Cette commande provoque la lecture complète du fichier en cours, ligne à ligne et du haut vers le bas. Ce détail a son importance car une variable utilisée dans une portion de code doit avoir été au préalable déclarée et initialisée, sinon il y a erreur. De même, si une portion de code n'est pas complète, il y a erreur.

Il est également possible de compiler une partie du fichier seulement en la sélectionnant puis en allant dans le menu **Compiler** (**Run**) et en sélectionnant **Compiler Sélection** (**Execute Selection**), ou bien en utilisant le raccourci clavier **Alt+Entrée**.

Pour que Python affiche les résultats calculés, il faut le lui avoir demandé. Pour ce faire, utilisez la fonction `print`.

Question 4. Créer une variable `x` et y affecter la valeur 3. Afficher le contenu de la variable.
Peut-on écrire l'égalité d'affectation dans le sens que l'on veut ?

Question 5. Créer une variable `y` égale à $7\times$`x` et afficher sa valeur. Modifier la valeur de `x`.
Quel est l'effet de cette modification sur la valeur de `y` ?

Question 6. Tester le code suivant : `type(x)`. Interpréter la réponse.
Attention, dans l'éditeur il faut écrire `print(type(x))`.

Question 7. Rajouter 1 à `x`. Quel est l'effet sur le type de `x` ?
Même question en rajoutant $0,5$.

Question 8. Echanger le contenu des deux variables `x` et `y` sans utiliser de troisième variable (trouvez la méthode vous-même et ne la donnez pas aux autres sinon il n'y a aucun intérêt).

Question 9. Tester le code suivant : `valeur = "3"` puis `type(valeur)`. Interpréter le résultat.

Question 10. Tester le code suivant : `valeur*2`. Interpréter le résultat.

Question 11. Tester le code suivant : `int(valeur)`. Quel est le type du résultat ? Que s'est-il passé ?
Afficher le contenu de la variable `valeur`. Interpréter le résultat.

Question 12. Tester le code suivant : `int(valeur*2)`. Quel est le type du résultat ? Que s'est-il passé ?

Question 13. Tester le code suivant : `valeur = 3` puis `type(valeur)`. Interpréter le résultat.

Question 14. Tester le code suivant : `valeur*2`. Afficher le contenu de la variable `valeur`. Interpréter le résultat.

Question 15. Tester le code suivant : `str(valeur)`. Quel est le type du résultat ? Que s'est-il passé ?
Afficher le contenu de la variable `valeur`. Interpréter le résultat.

Question 16. Tester le code suivant : `str(valeur*2)`. Quel est le type du résultat ? Que s'est-il passé ?

Question 17. Tester le code suivant : `str(valeur)*2`. Quel est le type du résultat ? Que s'est-il passé ?

Le langage Python possède aussi le type simple booléen. Une variable booléenne ne prend que deux valeurs : `True` (Vrai) ou `False` (Faux).

Question 18. Définir le rôle des opérateurs suivants : `<`, `>`, `<=`, `>=`, `==` et `!=`. Par exemple, `2 < 3`.

On dispose aussi des opérateurs logiques "ET", "OU" et "NON" avec respectivement les opérateurs `and`, `or` et `not`.

Question 19. Compléter les tables de vérité de ces opérateurs en utilisant la console.

bool	**NON** bool
True	
False	

bool1	bool2	bool1 **OU** bool2
True	True	
True	False	
False	True	
False	False	

bool1	bool2	bool1 **ET** bool2
True	True	
True	False	
False	True	
False	False	

4 Modules et fonctions

Question 20. Calculer le sinus de 1 radian avec la commande `sin(1)`.

Vous devez obtenir une erreur comme ceci :

```
Traceback (most recent call last):
  File "<console>", line 1, in <module>
NameError: name 'sin' is not defined
```

Le message est assez clair : Python ne connaît pas le mot `sin`. Même s'il possède beaucoup de fonctions de base, toutes ne sont pas chargées au démarrage car toutes ne seront pas utilisées à chaque fois. On a ainsi séparé les fonctions par thème dans des modules.

Question 21. Importer le module `math` en utilisant le code suivant : `import math`. Reprendre la question précédente.

Vous obtenez le même message d'erreur bien que l'importation a dû se faire sans problème. Le souci vient du fait que pour accéder tout le contenu de ce module, il faut précéder le nom de la fonction par le nom du module.

Question 22. Valider le code suivant : `math.sin(1)`.

Python est un langage très utilisé aujourd'hui, il existe bien évidemment beaucoup de modules (des milliers !). Cette année, nous en rencontrerons d'autres qui deviendront familiers.

Question 23. Un module peut contenir aussi des constantes pré-définies. Evaluer `math.cos(math.pi)`.

Question 24. Evaluer $1 + \ln(\pi)^2$.

Indication : Pour connaître les fonctions contenues dans un module, on peut utiliser la commande `help(math)`.

Si on souhaite éviter d'avoir à précéder le nom de chaque fonction du nom du module, on peut utiliser la forme suivante pour l'importation : `from math import *`. Ceci donne la possibilité d'utiliser TOUTES les fonctions disponibles directement, sans avoir à utiliser le nom du module d'où elles viennent. Attention cependant quand il y a des fonctions qui portent le même nom dans des modules différents, mais nous ne rencontrerons pratiquement jamais ce cas.

Question 25. Evaluer $\sin(1)+\sin(2)+\sin(3)+\sin(4)+\sin(5)+\sin(6)$. *Réponse :* $-0,10325384847654717$.

Question 26. Evaluer $\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{2}}}}}}$. *Réponse :* $1,617443$.

5 Introduction à l'écriture de fonctions

Pour pouvoir réutiliser un algorithme identique dans une variété de problèmes, il faut en faire une fonction. C'est un mécanisme extrêmement classique qu'on retrouve dans énormément de langages, Python n'y fait pas exception.

Question 27. Dans l'éditeur de texte, saisir le code ci-contre représentant une définition de fonction :

```
def carre(x) :
    return x*x
```

On rappelle que l'éditeur de code n'est rien qu'un éditeur de texte. Pour que Python mémorise la fonction et puisse l'utiliser, il vous faut compiler votre fichier.

Question 28. Compiler le code de la fonction contenu dans votre fichier. Que se passe-t-il ?

Normalement rien ne doit s'afficher dans la console, si vous n'avez pas fait d'erreur. En revanche, à partir de maintenant, l'interpréteur Python a une version exploitable en mémoire de la fonction `carre`.

Question 29. Dans la console, saisir `carre(5)`, valider et observer.

De même, saisir `carre(-2)`, observer et conclure.

Vous pouvez aussi saisir ces commandes dans l'éditeur mais pour que Python vous affiche les résultats, il faut ajouter la fonction `print`.

Question 30. Dans l'éditeur de texte, recopier et compléter le code ci-contre pour obtenir un programme qui calcule la somme des carrés de deux nombres.
Il est possible d'utiliser une fonction dans une fonction.

```
def sommeCarres(x,y) :
    # A compléter
    return # A compléter
```

Question 31. Exécuter le programme ainsi enrichi, puis dans la console, évaluer `sommeCarres(-2,3)`.

Question 32. Quel est le rôle de `return` dans une fonction ?

Dans la suite, vous devez écrire des fonctions par vous mêmes. Attention, il est **interdit** d'utiliser des structures algorithmiques (if, for, while) ou des structures de données (listes, tableaux, ...).

Question 33. Ecrire une fonction `est_pair` prenant en entrée un entier `n` et renvoyant un booléen indiquant si l'entier `n` est pair ou non.

Question 34.

1. Ecrire une fonction `est_multiple3o5` prenant en entrée un entier `n` qui renvoie `true` si `n` est un multiple de 3 ou 5.
2. Ecrire une fonction `est_multiple3p5` prenant en entrée un entier `n` qui renvoie `true` si `n` est un multiple de 3 mais pas de 5.
3. Ecrire une fonction `est_multiple35` prenant en entrée un entier `n` qui renvoie `true` si `n` est un multiple de 3 et 5. Ecrire une version de cette fonction ne faisant appel à aucun connecteur logique.

Question 35. Ecrire une fonction `signe` prenant en entrée deux réels `x` et `y` qui renvoie `True` si `x` et `y` ont le même signe et `False` sinon.

Question 36.

1. A votre avis, qu'est-ce que Python va afficher si on exécute la fonction `fonction` ?
2. Faites le test sur pyzo.
3. Que calcule la fonction ?

```
def fonction() :
    a=1; print(a)
    b=1; print(b)
    c=a+b; print(c)
    d=b+c; print(d)
    e=c+d; print(e)
    f=d+e; print(f)
```

Question 37. On suppose ici que le booléen `True` est représenté par l'entier `1` et le booléen `False` par l'entier `0`. On travaille donc sur les entiers.

1. Ecrire une fonction `non` prenant en entrée un entier `a` valant 0 ou 1 et renvoyant la valeur de non a.
2. Ecrire une fonction `et` prenant en entrée deux entiers `a` et `b` valant 0 ou 1 et renvoyant la valeur de "a et b" (c'est-à-dire 1 si c'est vrai et 0 sinon).
3. Ecrire une fonction `ou` prenant en entrée deux entiers `a` et `b` valant 0 ou 1 et renvoyant la valeur de "a ou b".

Question 38.

1. Ecrire une fonction `implique` prenant en entrée deux booléens `a` et `b` et renvoyant la valeur de $a \Rightarrow b$.
2. Ecrire une fonction `equivalence` prenant en entrée deux booléens `a` et `b` et renvoyant la valeur de $a \Leftrightarrow b$.
   (a) En utilisant la fonction précédente.
   (b) Sans utiliser la fonction précédente.

6 Extra

Dans cette partie, il est possible d'utiliser des structures algorithmiques et des structures de données.

Question 39. Ecrire une fonction `Fibonacci` qui calcule les `n` premiers termes de la suite de Fibonacci.

Question 40. Ecrire une fonction `puissance2` qui prend en entrée un entier `n` et qui renvoie l'entier `k` tel que $2^k \leq n < 2^{k+1}$.

Question 41. Ecrire une fonction `pgpremier` qui prend en entrée un entier `n` et qui renvoie le plus grand nombre premier plus petit que `n`. On essaiera de limiter les tests inutiles.

Question 42. Somme de 3 carrés Beaucoup de nombres peuvent s'écrire comme la somme de trois carrés, par exemple $62 = 2^2 + 3^2 + 7^2$ et $8 = 2^2 + 0^2 + 2^2$. Certains ne peuvent pas, comme 7.

Combien de nombres inférieurs ou égaux à 10 000 ne peuvent pas s'écrire comme somme de trois carrés ?

Question 43. Le jardin des Hesperides Les Hespérides, filles d'Atlas, habitaient un merveilleux jardin dont les pommiers donnaient des pommes en or. Pour son onzième travail, Eurysthée, demande à Hercule de ramener ces pommes.

Une fois atteint le jardin merveilleux, l'oracle Nérée apprit à Hercule qu'il pourrait repartir avec une partie des pommes ... à condition qu'il montre ses facultés en calcul mental. Nérée lui tint ce propos :

"J'ai empilé les pommes d'or pour toi, sous la forme d'une pyramide. L'étage le plus haut ne contient qu'une pomme. l'étage juste en dessous forme un carré 2×2 (contenant 4 pommes), l'étage juste en dessous forme un carré 3×3 (contenant 9 pommes). La pyramide que tu vois contient 50 étages. L'étage de base contient donc 2 500 pommes ... Je suis d'accord pour te laisser partir avec les pommes contenues dans certains étages. Précisément,si un étage contient un nombre de pommes multiple de 3, tu peux l'emporter. Si tu m'annonces combien de pommes tu emporteras au total, je te laisserai partir avec ces pommes ... "

Pour relever ce défi, vous devez aider Hercule en lui indiquant le nombre de pommes qu'il pourra emporter. Par exemple, si la pyramide n'avait compté que six étages, chaque étage aurait été composé de 1, 4, 9, 16, 25 et 36 pommes. Hercule aurait pu emporter les 9 pommes de l'étage 3 car 9 est multiple de 3 et les 36 pommes de l'étage 6 car 36 est multiple de 3. Au total il aurait donc emporté 45 pommes.

Mais combien peut-il en emporter pour une pyramide de 50 étages ?