TP d'informatique (Maths) n°1 & 2

"Premiers pas en Python "

But du TP

- Prendre en main l'environnement de travail Python
- Découvrir Python en mode "interprété ligne à ligne" et en mode "exécution d'un fichier" via un environnement de développement (ici IDLE ou Pyzo)
- Comprendre les notions de variables et d'affectation.
- Pour les plus rapides : découverte les notions de boucle et de test.

Vous penserez **à sauvegarder régulièrement** votre travail dans un fichier dans votre espace personnel. Pour le nom de fichier, n'utilisez ni accent, ni espace, ni caractères spéciaux (autres que – et _) : en effet, certains projets nécessitent l'ouverture de fichiers que l'on aura écrits au préalable : or cet appel peut s'avérer délicat si le nom de ce fichier possède des caractères "non habituels"

1) Types numériques et type chaîne de caractères

Lancer Python (version IDLE ou Pyzo). Pour IDLE, ouvrir une fenêtre "Python Shell" qui est l'interpréteur. Pour Pyzo, l'interpréteur est la sous fenêtre du dessus.

a) Interpréteur

Il s'agit d'une fenêtre avec une invite de commandes (ou "prompt"), symbolisée par le >>>, qui vous invite à taper une commande (ou instruction). Cela fonctionne un peu comme une messagerie instantanée (ou une calculatrice) où Python vous répondra à chaque fois que vous lui écrivez quelque chose.

b) Les premières commandes Python – les opérateurs

Dans l'interpréteur, taper les instructions de la colonne de gauche et écrivez les résultats obtenus (réponses de l'interpréteur), dans la colonne de droite, en commentant éventuellement, en particulier en ce qui concerne le comportement de certaines opérations ainsi que les précédences.

instruction à taper	résultat obtenu + commentaire ou explication
`2 + 5`	
`20 / 3`	
`20 // 3`	
`20 % 3`	
`5.45*10`	
`2 ** 4`	
`3+2*5`	
`(3+2)*5`	
`"bonjour"` (avec des guillemets)	
`'bonjour'` (avec des apostrophes)	
`"il fait " + "beau"`	
`"il fait " + "beau" + ' et chaud'`	
`'bonjour ' * 5`	

Astuce : pour éviter d'avoir à retaper toute une ligne, utilisez la flèche "haut" du clavier pour vous positionner sur cette ligne, puis tapez "Entrée" : la ligne est recopiée après l'invite de commande et vous pouvez la modifier avant de taper de nouveau "Entrée"

Certaines instructions ne sont pas acceptées par l'interpréteur Python et provoquent l'affichage d'un message d'erreur. Il faut attentivement tous les messages d'erreur (même s'ils sont en anglais) car ces messages nous apprennent comment fonctionne le langage et nous aident à corriger nos instructions et nos programmes. Tapez les instructions suivantes, observez les erreurs obtenues et analysez-les :

instruction à taper	erreur observée + explication
`20 / 0`	
`20 @ 3`	
`("bonjour" * 5) / 5`	
`"bonjour" + 5`	
`(3+2))*5`	
`(3+2*5`	

Les erreurs rencontrées ici sont dues à des erreurs de syntaxe ou au fait que l'on ait voulu appliquer un opérateur Python à des données pour lequel l'opérateur ne s'applique pas. Dans Python, et dans les logiciels de programmation, chaque donnée possède un certain type. Dans les instructions que l'on a eu à tester, on a rencontré des données de type :

- entier ou int pour les données `20, 3 , 5`…
- flottant ou float pour la donnée `5.45`
- chaine de caractère ou string pour les données `'bonjour'` ou `"beau".`

Nous verrons d'autre type de données dans la suite du TP : les booléens, les listes, les tuples….

Pour connaitre le type d'une donnée, on peut utiliser la fonction `type` : taper ce qui suit :

instruction à taper	résultat obtenu + commentaire ou explication
`type(3)`	
`type(1.5)`	
`type(3.0)`	
`type("bonjour")`	
`type('beau')`	
`type("3")`	
`type(3+1.5)`	

Exercice 1 : Opérateurs et types de données

Sur quels types de données peut-on utiliser les opérateurs `+ , * , / , // , % , ** ` ?
Quel est le type du résultat ? Répondre en complétant ci-dessous. Vous pouvez faire des tests en tapant d'autres instructions dans l'interpréteur Python.

Exemple : `3 + 7.1` est accepté par l'interpréteur Python, donc l'opérateur `+` s'applique à un entier et à un flottant, et le résultat est de type flottant

opérateur	types de données utilisables	type du résultat
+		
*		
/		
//		
%		
**		

Remarque : pour les chaines de caractères, l'opérateur + s'appelle la concaténation

c) Le transtypage

On peut parfois transformer une donnée d'un certain type en une donnée d'un autre type. Cela s'appelle le transtypage (ou cast en anglais)

donnée de départ	De quel type est la donnée de départ ?	instruction à taper	type du résultat obtenu / commentaire
3.0		int(3.0)	
3.5		int(3.5)	
3		float(3)	
4		str(4)	
"3"		int("3")	
"3.5"		float("3.5")	
"3.5"		int("3.5")	
"bonjour"		int("bonjour")	

Exercice 2 : Corriger les instructions

Comment corriger les instructions suivantes pour qu'elles ne provoquent plus d'erreur ?

a) But recherché : on veut obtenir la chaîne de caractères `"Vous êtes 18 dans ce groupe de TP"`
Instruction : `"Vous êtes" + 18 + "dans ce groupe de TP"`
Instruction corrigée : ….

b) But recherché : on veut obtenir la chaîne de caractères `"blablablablablabla"` c'est-à-dire `"bla"` 6 fois de suite
Instruction : `"bla" * 6.0`
Instruction corrigée : ….

2) Les variables et l'affectation

a) L'affectation

Il peut-être pratique de ranger les données dans des contenants, aussi appelés variables, afin de les conserver dans la mémoire de l'ordinateur pour les utiliser plus tard.

Le rangement d'une donnée (un contenu) dans une variable (un contenant) s'appelle l'affectation. Elle se fait grâce à l'opérateur d'affectation = dans Python. Cette affectation se fait en deux temps :

(1) évaluation de la partie à droite de l'opérateur d'affectation. Le résultat de cette évaluation est la donnée qui va être rangée dans la variable.
(2) rangement du résultat de cette évaluation dans la partie gauche de l'opérateur d'affectation. Cette partie à gauche est la variable.

Taper les instructions suivantes dans l'ordre indiqué, et compléter le tableau

instruction à taper	Quelle est la variable ?	Quelle est la donnée (résultat de l'évaluation de la partie droite) ? De quel type est cette donnée ?
`age = 20`		
`age = 30`		
`prenom = "Perceval"`		
`taille_m = 1.75`		
`taille_cm = 100 * taille_m`		
`age = age + 3`		
`phrase1 = "bonjour " + prenom`		
`phrase1 = phrase1 + " !"`		
`phrase2 = prenom + " a " + str(age) + " ans."`		

Python accepte des affectations simultanées en séparant les noms de variables par des virgules.
Par exemple : `age, prenom, taille_m = 30, "Perceval", 1.75` a le même effet que les lignes 2, 3 et 4 du tableau précédent.
Cependant, on évitera l'utilisation systèmatique de ces affectations simultanées.
Pour afficher le contenu d'une variable dans l'interpréteur Python, il suffit de taper le nom de cette variable. Taper les instructions suivantes et compléter

instruction à taper	Quelle valeur contient la variable ?
`age`	
`prenom`	
`prenom = "Lancelot"`	
`prenom`	
`age = 40`	
`age`	
`age = 22`	
`age`	
`age = age + 1`	
`age`	
`phrase1`	
`phrase2`	
`profession`	
`profession = "étudiant"`	
`profession`	

b) Les noms de variables

Un nom de variable doit :

- débuter par une lettre
- contenir uniquement des lettres sans accents, des chiffres et le tiret de soulignement _ (appelé tiret "underscore" ou encore "tiret du 8")
- Etre aussi explicite que possible (ne pas utiliser des noms trop courts et sans signification comme a, b, x, y, n mais utiliser plutôt age, longueur, nombre, somme …)

Le langage Python fait la distinction entre les majuscules et les minuscules.

De plus, il existe des mots clés réservés du langage qui ne peuvent pas être utilisés comme nom de variables. Ce sont : (on peut y accéder à l'aide de la commande `keyword.kwlist`)

and	as	assert	break	class	continue
def	del	elif	else	except	False
finally	for	from	global	if	import
in	is	lambda	None	nonlocal	not
or	pass	raise	return	True	try
while	with	yield			

Exercice 3 : Echanger la valeur de 2 variables

Ecrire (sur papier) un algorithme pour échanger les valeurs de 2 variables que vous nommerez `var1` et `var2`

3) Le mode éditeur

Nous allons maintenant laisser de côté l'interpréteur Python (on pourra toutefois y revenir pour tester quelques commandes) et allons passer en mode éditeur, pour écrire des programmes (appelés aussi scripts ou applications) qui pourront être enregistrés et réutilisés plus tard.

Exercice 4 : Programme TP03_initial

Dans IDLE, cliquez le menu `File/Open`. Dans le répertoire TP03 qui se trouve sur le bureau, ouvrez (avec IDLE) le programme `TP03_initial.py`. Ce programme sera à rendre à la fin de ce TP.

i. Dans la partie "en-tête du module", modifiez la date et le nom de l'auteur (en indiquant le votre).

ii. Exécutez ce programme en cliquant le menu `Run / Run module` (ou en utilisant le raccourci clavier `F5`). Commentez dans votre compte-rendu de TP.

iii. (Répondez dans votre compte-rendu de TP) : pourquoi a-t-on écrit `str(age)` et pas simplement `age` dans la ligne :

```
print("bonjour " + prenom + " " + nom + ", tu as " + str(age) + " ans.")
```

iv. Dans la partie "programme principal", modifiez la valeur des variables : affectez votre nom, votre prénom et votre âge aux variables `nom, prenom, age`. Enregistrez le programme sous le nom `TP03_1.py` et exécutez-le à nouveau. Que s'affiche-t-il ? (Répondez sur votre compte-rendu de TP).

a) Le programmeur et l'utilisateur du programme

Il ne faut pas confondre le "programmeur" ou "développeur" (celui ou celle qui écrit le programme) et "l'utilisateur" (celui ou celle qui utilisera le programme). Seul le programmeur peut modifier le programme (c'est-à-dire le fichier d'extension .py). L'utilisateur est complètement extérieur au programme ; il ne peut qu'exécuter le programme en lançant la commande `Run module`.

b) Les fonctions d'entrée / sortie

Afin que l'utilisateur puisse entrer des données dans les variables du programme et visualiser les résultats calculés par le programme, le programmeur utilise des fonctions d'entrée et de sortie.

La fonction `print` est une fonction de sortie : elle affiche à l'écran (à l'attention de l'utilisateur) une donnée ou le contenu d'une variable.

La fonction `input` est une fonction d'entrée : le programme affiche une question à l'attention de l'utilisateur et attend que l'utilisateur tape sur des touches du clavier en réponse à cette question. L'utilisateur doit terminer sa saisie par la touche "Entrée" pour que le programme continue son exécution. La suite de caractères saisie par l'utilisateur est récupérée par le programme où elle peut-être utilisée. Elle est généralement affectée à une variable du programme.

Remarque : la donnée récupérée par la fonction `input` est de type chaîne de caractères. Si on souhaite récupérer un nombre (et non une chaîne de caractères), il faut transtyper dette donnée récupérée par `input`.

Exemple : dans l'interpréteur, tapez les instructions suivantes :

instruction à taper	commentaire
`numero = input("entrez un chiffre entre 1 et 5 : ")`	
`type(numero)`	
`numero = numero + 1`	
`numero = int(numero)`	
`type(numero)`	
`numero = numero + 1`	

Exercice 5 : Programme TP03_1 : saisie d'un nom, prénom, âge par l'utilisateur

Nous allons modifier le programme `TP03_1.py`. Remplacez la ligne

```
nom = "Cleese"  (ou votre nom maintenant)
```

par

```
nom = input("Quel est ton nom ? ")
```

Faites de même pour les variables prenom et age. Enregistrez et exécutez. Vérifiez que l'utilisateur peut entrer n'importe quels noms, prénoms et âges et que ceux-ci s'affichent bien ensuite dans une phrase "bonjour …."

Exercice 6 : afficher la somme de deux valeurs fournies en entrée par l'utilisateur.

Nous allons créer un nouveau programme. Pour cela, cliquez sur le menu `File / New Window` pour ouvrir un nouveau programme vide. Nommez-le `TP03_2.py` , enregistrez-le. Tapez un en-tête de programme selon le modèle suivant.

```
#!/usr/bin/python3
# -*- coding: utf-8 -*-
"""
Documentation
de ce module
"""
# fichier: TP01_02.py
# auteur: Calio François
# date : 08 septembre 2015


# ---------- imports ----------


# ---------- Programme Principal ----------
```

Tapez les lignes suivantes dans la partie "programme principal" puis exécutez :

```
var1 = input("Quelle est la première valeur ?  ")
var2 = input("Quelle est la deuxième valeur ?  ")
var_somme = var1 + var2
print("La somme des deux valeurs est ", var_somme)
```

Quel est le résultat obtenu ? On souhaite que la variable `var_somme` contienne la somme des deux nombres entrés par l'utilisateur. Corrigez ce programme pour qu'il fasse ce qui est souhaité.
Commentez dans votre compte-rendu de TP.

c) Mettre des commentaires dans les programmes

Des commentaires doivent toujours figurer dans les programmes : ils permettront aux autres programmeurs de comprendre ce que vous avez voulu faire ; ils vous seront aussi utiles lorsque vous relirez vos propres programmes plus tard.
Les commentaires ne sont pas interprétés comme des instructions du langage, mais ils sont ignorés par Python. Ils sont introduits par un caractère # (tout ce qui suit le # sur la même ligne est un commentaire). Ils peuvent aussi figurer entre des triple-guillemets (""" suivi de """)

Exercice 7 : Commenter les programmes TP03_1 et TP03_2
Ajouter des commentaires précédés du signe # dans vos deux programmes `TP03_1.py` et `TP03_2.py` pour expliquer ce que font les différentes lignes du programme.
En fin de programme, on peut aussi copier-coller les résultats de l'exécution du programme entre triple-guillemets pour visualiser différents tests de ce programme et mieux comprendre commet il agit. Evidemment pour être parlant, il faut que les tests soient réalisés sur plusieurs valeurs.

4) Le type booléen

Le type booléen est un type de donnée qui ne peut prendre que 2 valeurs : `True` ou `False`. Dans Python ces booléens peuvent être représentés sous la forme des entiers `0` pour `False` et `1` pour `True` (et en fait tout nombre autre que `0`).
Les opérateurs qui s'appliquent aux booléens sont les opérateurs logiques : `not`, `and` , `or` . D'autres opérateurs s'appliquent à des données numériques ou des chaînes de caractères et produisent un résultat booléen : ce sont les opérateurs de comparaison :

`==`	signifie	"est égal à"
`!=`	signifie	"est différent de"
`< , >`	signifient	"est strictement inférieur " et "est strictement supérieur"
`<= , >=`	signifient	"est inférieur ou égal" et " est supérieur ou égal "

Taper les instructions suivantes et compléter

instruction à taper	résultat obtenu + commentaire
`1 == 1`	
`1 == 3`	
`1 != 3`	
`1 == 1.0`	
`(3+7) == (5+2)`	
`(1 == 1) and (2<2)`	
`(1 == 1) and (2<=2)`	
`(2 == 15) or (3>2)`	
`1 < 4 < 15`	
`(1 > 6) and (1/0 > 2)`	
`(1/0 > 2) and (1 > 6)`	
`(1/0 > 2) or (1 <= 6)`	
`(1 <= 6) or (1/0 > 2)`	
`valeur = (1 == 1)`	
`type(valeur)`	

`not(valeur)`	

Que pensez-vous du comportement de Python concernant ses réponses aux lignes 10 à 13 ?

5) Quelques autres exercices

Exercice 8 : Quelles sont les valeurs de x et y après les instructions suivantes ? Prédire puis vérifier…

a) x = 42; y = 10; x = y; y = x;

b) x = 42; y = 10; z = x; x = y; y = z;

c) x = 42; y = 10; (x, y) = (y, x) ;

Exercice 9 : Exécuter les commandes suivantes dans l'interpréteur

t = [6, 12, 'a']; t[1]; t[3]; t[0]; t[-1]; len(t); t[1] = 2; t; t + t;

Exercice 10 : Exécuter les commandes suivantes dans l'interpréteur

x = "Hello"; y = "world"; x[1]; y[2]; x[9]; x[-1]; y[-2]; z = x + y;
z; t = x+" "+y; t; print(t); print(x,y); print(x+y); x[2] = "z"; x;

Exercice 11 : Tapez les lignes suivantes dans l'éditeur puis exécutez-les

```
prenom = input("Quel est votre nom ? ")
print("Bonjour ",prenom, " . Faisons un petit jeu")
print("Pensez à deux nombres")
print("Calculez la somme et la différence de ces nombres")
s = int( input("Donnez la somme  "))
d = int( input("Donnez la différence  "))
print("Je réfléchis....")
n = (s + d)/2
m = s - n
print("J'ai trouvé")
print("Vos deux nombres sont ",m," et ",n)
```

Exercice 12 : Même idée que dans l'exercice précédent mais on demande la somme et le produit

Exercice 13 : Pour lycéens ISN / Python : Idée de projets :

- Mastermind / Motus : On fait choisir à l'ordinateur une combinaison de 4 lettres parmi la liste ['A', 'B', 'C', 'D', 'E', 'F', 'G'] (à l'aide de la fonction choice du module random). Puis on demande à l'utilisateur d'entrée sa propre combinaison du même format. L'ordinateur doit répondre par une liste de 4 caractères pris parmi 'X', 'O', '-' indiquant : pour le 'X' que la lettre de la proposition est dans la combinaison secrète avec le même rang, pour le 'O' que la lettre de la proposition se trouve dans la combinaison secrète mais pas avec le même rang ; et enfin, pour le '-' que la lettre de la proposition ne se trouve pas dans la combinaison secrète.
  Par exemple, si la combinaison secrète est "ABCD" et que la proposition est "BECG", le programme doit répondre : "O-X-"

- Aide à l'arbitre de tennis : On veut réaliser un programme informatique permettant de mettre à jour le score d'un match de tennis. On écrira les différentes fonctions suivantes
  - Une fonction qui compte les points au cours du jeu. En entrée on demande répétitivement quel joueur, 1 ou 2, gagne le point ; au fur et à mesure, on calcule et on affiche le score.
  - Pour faciliter les tests, écrire une fonction qui reçoit une chaine de caractères de la forme "211221" indiquant que le premier point est marqué par "2", le suivant par "1", puis par "1" etc… et qui donne le score dans ce jeu.
  - Ecrire une fonction qui compte les jeux au cours d'un set
  - Ecrire une fonction qui compte les sets dans un match.