

TP 1

Découverte de Python

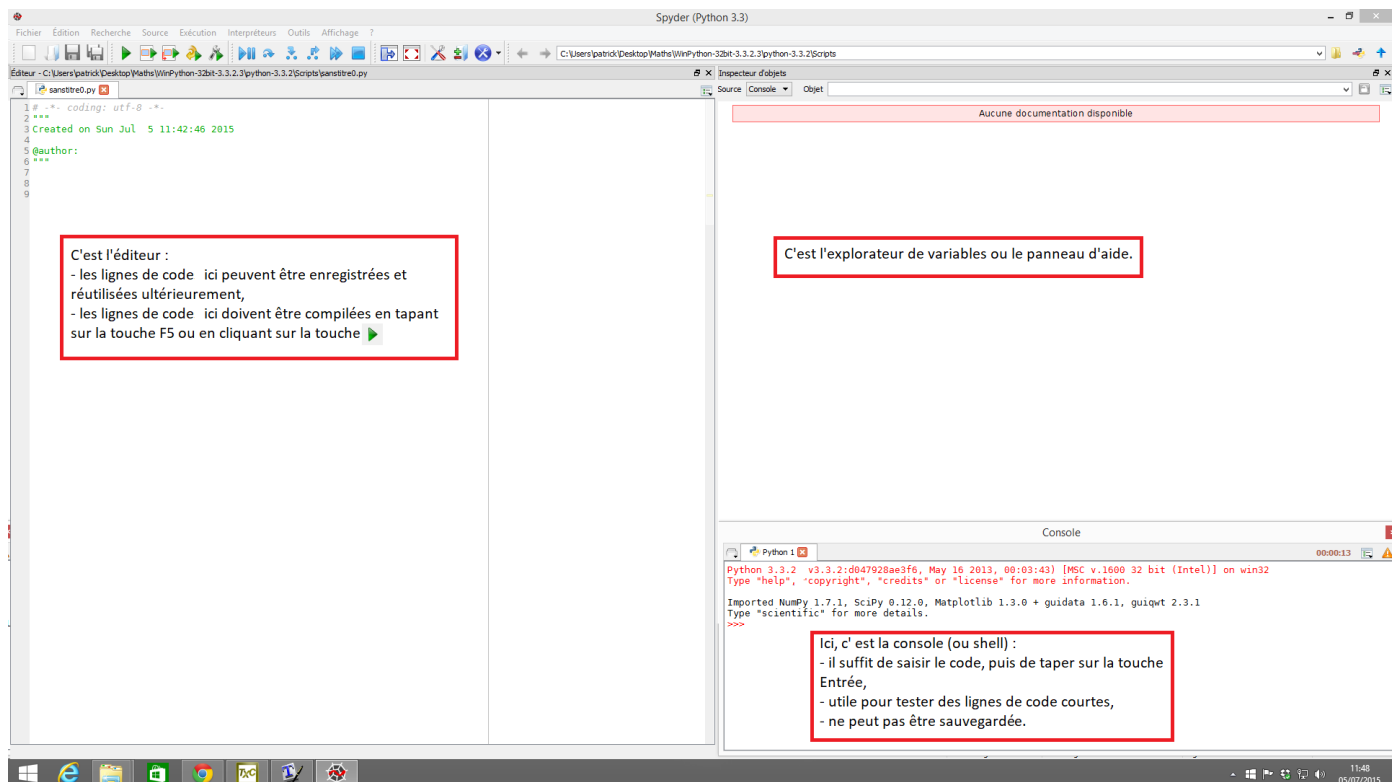
I - Savoir ouvrir une session et démarrer l'application Spyder

🐍 **Explication** 🐍 L'application Spyder est une interface vous permettant de saisir vos lignes de code et de les exécuter. Pour démarrer cette application, il vous suffit de suivre les étapes suivantes :

- étape 1 : démarrer une session en utilisant les codes IACA fournies le jour de la rentrée,
- étape 2 : aller sur *Ordinateur*, puis *diaxens*, ensuite *Applications*, pour enfin lancer *Spyder*.

Une fois l'application lancée, vous obtiendrez une fenêtre de ce type constituée de trois zones bien délimitées :

l'éditeur, la console et l'explorateur de variables.



🐍 Explication 🐍 (Console versus éditeur)

- La console est comme une feuille de brouillon : elle vous permet de tester rapidement des commandes courtes (en les saisissant puis en tapant sur *Entrée*), mais ce qui y est saisi ne peut pas être sauvegardé pour être ré-utilisé ultérieurement.
- L'éditeur est plus adapté pour des lignes de code longues, et ce qui y est saisi peut être sauvegardé. En contrepartie, il faut compiler en appuyant sur F5 ou la touche ▶ pour exécuter les lignes de code. Sous réserve d'existence et d'affichage, le résultat de la compilation apparaît dans la console.

II - Savoir calculer avec Python, mais pas que ...

🐡 **Explication** 🐡 (À connaître par ❤️!) Vous pouvez effectuer les calculs de bases avec Python sous réserve de respecter les syntaxes suivantes :

Syntaxe Python	À quoi ça sert ?	Exemple à saisir dans la console, puis taper sur la touche Entrée	Exemple à saisir dans l'éditeur, puis compiler (F5 ou ►)
+	additionner	14+31	print(14+31)
-	soustraire	52-23	print(52-23)
*	multiplier	7*11	print(7*11)
**	élever à la puissance	3**2	print(3**2)
/	diviser	1e2/4 (1e2 = 10 ²)	print(1e2/4)
//	obtenir le quotient de la div. euclidienne	17//3	print(17//3)
%	obtenir le reste de la div. euclidienne	17%3	print(17%3)

Les règles de priorité des opérations sont exactement les mêmes que les règles en mathématiques, autrement dit elles suivent la règle « **PEMDAS** » :

Parenthèses >> **E**xponentiation (i.e. puissances) >> **M**ultipli./**D**ivision >> **A**ddition/**S**oustrac.

où le symbole >> signifie « est prioritaire sur ».

Lorsqu'un calcul est saisi dans l'éditeur puis compilé, le calcul est bien effectué par Python mais n'est pas affiché :

la fonction `print()` permet d'afficher le résultat d'un calcul saisi dans l'éditeur !

🐡 **Explication** 🐡 (Astuce à connaître dans la console)

Après avoir validé des expressions avec la touche *Entrée*, vous pouvez revenir à l'une d'elles en appuyant autant de fois que nécessaire sur la touche ↑.

🐡 **Explication** 🐡 (**Savoir importer un module**) Pour utiliser certaines fonctions mathématiques classiques (cosinus, sinus, tangente, exponentielle, logarithme, $\sqrt{\quad}$, ...) ou certaines constantes célèbres (π , e , ...), il faut au préalable importer le module « *math* » en saisissant la commande suivante :

```
1 from math import *
```

dans la console (suivi de *Entrée*) ou dans l'éditeur (suivi d'une compilation).

Exemple Les syntaxes Python sont alors très intuitives. Voici quelques exemples saisis dans la console qui vous permettent de mémoriser les syntaxes en question :

Les explications en vert n'apparaissent pas dans la console évidemment !

```
1 # pensez à importer le module math !
2 >>> from math import *
3
4 >>> cos(3*pi)
5 -1.0
6
7 >>> sin(5*pi/6)
8 0.49999999999999994 # Que remarquer ?
9
10 >>> tan(5*pi/4)
11 0.9999999999999999 # Que remarquer ?
```

```
1 # pour exponentielle de 2
2 >>> e**2
3 7.3890560989306495
4
5 # log pour le logarithme népérien (ln)
6 >>> log(e)
7 1.0
8
9 # sqrt pour square root (racine carrée)
10 >>> sqrt(16+9)
11 5.0
```

🔗 **Explication** 🔗 Pour les exercices suivants, vous aurez à déterminer sans ordinateur les syntaxes Python à saisir, et vous vérifierez ensuite à l'aide de l'ordinateur.

Mémorisez donc les syntaxes précédentes avant de passer aux exercices !

Pour les exercices 1 et 2 suivants, la notation $R(a, b)$ (resp. $Q(a, b)$) désignera le reste (resp. quotient) de la division euclidienne d'un entier a par un entier b .

1

Exercice

🕒 Pour chacun des calculs du tableau ci-bas :

- Calculer le résultat à la main,
- Déterminer l'expression en langage Python à saisir dans la **CONSOLE**,
- Vérifier le tout à l'aide de la **CONSOLE**.

Calcul à la main	Expression Python
$\frac{(1+1)^3}{12^{-1}} = \dots\dots$	
$\frac{-3^2 + 7 \times \frac{1}{3}}{12 - 2^2} = \dots\dots$	
$\frac{R(7, 4) - \frac{1}{9}}{Q(7, 5)^2} = \dots\dots$	
$\cos\left(\frac{51\pi}{3}\right) = \dots\dots$	
$\sqrt{\ln(e^{36})} = \dots\dots$	

2


Exercice

🕒 Pour chacun des calculs du tableau ci-bas :

- Calculer le résultat à la main,
- Déterminer l'expression en langage Python à saisir dans l' **ÉDITEUR** pour que le résultat soit affiché,
- Vérifier le tout à l'aide de l' **ÉDITEUR**.

Calcul à la main	Expression Python
$\frac{(-2+1)^4}{12^2} = \dots\dots$	
$\frac{(-3)^2 + 4^2 \times \frac{1}{3}}{7^2} = \dots\dots$	
$\frac{\sqrt{Q(9, 4)}}{\frac{Q(7, 5)}{2}} = \dots\dots$	
$\tan\left(\frac{14\pi}{4}\right) = \dots\dots$	
$2^{2+\sqrt{9}} - R(7, 4) = \dots\dots$	



3**Exercice**

 Les expressions suivantes sont saisies dans la **CONSOLE** :

```
1 >>> (7+3**2)//2
2 ???
3
4 >>> (35%4)+3**2*3+1
5 ???
6
7 >>> (3**(2*5)-4**2)%2
8 ???
9
10 >>> cos(5**2*pi/3)
11 ???
12
13 >>> e
14 ???
```



Pour chacune des expressions ci-dessus :

- Calculer le résultat qui sera affiché dans la console après exécution (avec la touche *Entrée*).
- Vérifier à l'aide de la console.

 **Explication**  (**Mettre des remarques/commentaires**) Il est souvent utile de commenter ses lignes de code (notamment dans l'éditeur). Pour ce faire, il suffit de précéder vos commentaires de la commande `#` pour que ces lignes ne soient pas prises en compte lors de la compilation. Les commentaires permettent d'expliquer des lignes de code et de rendre un travail plus intelligible.


Exemple

```
1 # ne pas oublier d'importer le module math !!
2 from math import *
3
4 # Exercice 4
5 print(15+16)
6 print(3**2) # ** permet de calculer des puissances
```

 **Explication**  Python n'est pas qu'une simple calculatrice géante permettant de faire du calcul. Vous pouvez aussi évaluer d'autres types d'expressions. Exécuter les exemples suivants dans la console, puis compléter les pointillés par le résultat obtenu :


```
1 # une liste
2 >>> [1,2,3]
3 ..... # qu'obtenez-vous ?
4
5 # == pour vérifier l'égalité
6 >>> 1==3
7 ..... # qu'obtenez-vous ?
8
9 # > strictement supérieur
10 >>> 1>3
11 ..... # qu'obtenez-vous ?
```

4**Exercice**

 Les expressions suivantes sont saisies dans l'**ÉDITEUR** :

```
1 (1+5**2)%3
2 ???
3
4 print((32//5)%5-3**2/3)
5 ???
6
7 print(sin(17)**2+cos(17)**2)
8 ???
9
10 print(-3*e**(log(15)))
11 ???
12
13 print(tan(pi))
14 ???
```

Pour chacune des expressions ci-dessus :

- Calculer le résultat qui sera affiché dans la console après compilation (avec la touche **F5** ou ).
- Vérifier à l'aide de l'éditeur.

```
1 # <= inférieur ou égal
2 >>> 3<=4
3 ..... # qu'obtenez-vous ?
4
5 >>> (1>2)==True
6 ..... # qu'obtenez-vous ?
7
8 >>> (1,2,3)
9 ..... # qu'obtenez-vous ?
10
11 >>> "hello world"
12 ..... # qu'obtenez-vous ?
```

✗ ATTENTION ! ✗ L'opérateur d'égalité n'est pas le symbole « = » mais le symbole « == » ! Nous verrons un peu plus loin la signification du symbole « = ».

III - Connaître les types en Python et savoir « transtyper »

Dans la suite du TP, nous travaillerons uniquement dans la console

🐞 **Explication** 🐞 (À connaître par ❤️!) Comme en mathématiques, les objets manipulés en Python ont une « nature », qu'on appelle **TYPE** et pour chacun de ces types on peut effectuer des opérations bien précises.

Voici un tableau qui vous présente différents types de Python :

Type	Explication	Exemple à saisir dans la console
int	les entiers (integer)	1 , 17%5 , 15//3
float	les flottants	1.0 , pi , e , 15/3 , cos(pi)
str	les chaînes de caractères (string)	"bonjour" , "la PTSI 2" , "111"
list	les listes	[1,2,3] , ["hello",14,-17.0]
tuple	les tuples	(1,2,3) , ("hello",14,-17.0)
bool	les booléens (True, False)	1>2 , True , 1==3 , 5>=3
NoneType	les « sans » types	print(3) , print("hello")

🐞 **Explication** 🐞 Pour connaître le type d'un objet, on peut utiliser la fonction `type()`.

Exemple Compléter à l'aide de la console :

```
1 >>> type(1>3)
2 ..... # qu'obtenez-vous ?
3
4 >>> type(1.5)
5 ..... # qu'obtenez-vous ?
```

```
1 >>> type("hello world")
2 ..... # qu'obtenez-vous ?
3
4 >>> type(17)
5 ..... # qu'obtenez-vous ?
6
7 >>> type(["hello",1.0,17])
8 ..... # qu'obtenez-vous ?
```

5

Exercice

a) Déterminer le résultat et son type (sans ordinateur!).

b) Vérifier les réponses données à l'aide de la console.

Expression	Résultat et type	Expression	Résultat et type
5		15.0//3	
5.0		15%3	
15//3		17==3	
15/3		(4<3)==False	
cos(pi)		["PTSI 2"]	
"PTSI 2"		cos(pi/3)==-1/2	

👉 **Explication** 👉 (Comprendre les messages d'erreur) En étant capable de comprendre les messages d'erreurs, vous serez plus à même de les identifier et corriger. L'exercice suivant recense une liste non exhaustive de messages que vous pouvez rencontrer avec Python.

6**Exercice**

Compléter le tableau suivant :

Expression à taper	Erreur observée et explications
20/0	
"bonjour"/3	
"15"+15	
(3+2))*5	
(3+2*5	

👉 **Explication** 👉 Le transtypage consiste à changer le type d'une donnée en un autre type.

7**Exercice**

Compléter le tableau suivant :

Donnée de départ	Quel est son type ?	Expression à taper	Type du résultat et commentaires
3.0		int(3.0)	
3.5		int(3.5)	
3		float(3)	
4		str(4)	
"3"		int("3")	
"3.5"		float("3.5")	

Donnée de départ	Quel est son type ?	Expression à taper	Type du résultat et commentaires
"3.5"		int("3.5")	
"bonjour"		int("bonjour")	

IV - Savoir manipuler des variables

👉 Explication 👉

Le symbole « = » permet de donner une valeur à une variable.

Exemple Voici quelques exemples à lire et à tester dans la console :

```

1 # on affecte à la variable x la valeur 42.
2 >>> x=42 # Python enregistre la valeur mais ne répond rien
3
4 # pour afficher le contenu d'une variable dans la console, il suffit de taper son nom
5 >>> x
6 42
7
8 >>> type(x)
9 <class 'int'>
10
11 # on affecte à la variable x la chaîne de caractère "hello".
12 >>> y="hello" # Python enregistre la chaîne de caractère mais ne répond rien
13
14 >>> type(y)
15 <class 'str'>
16
17 # on peut effectuer plusieurs affectations en une seule ligne comme suit :
18 >>> a,b=1,3
19
20 >>> a
21 1
22
23 >>> b
24 3
25
26 # échange des valeurs de a et b
27 >>> a,b=b,a

```

Il est important de différencier :

- les expressions qui sont évaluées par le logiciel,
- les instructions comme l'affectation qui sont des ordres que le logiciel va exécuter.

👉 **Explication** 👉 **(Les 3 critères pour nommer une variable)** Un « bon » nom de variable vérifie les trois critères suivants :

- 1) il est sans majuscule et sans accent,
- 2) il est sans espace (on utilise des underscores « _ » à la place),
- 3) il est de préférence significatif :

↪ age pour stocker un âge, taille_cm pour stocker une taille en cm, etc...

8**Exercice**

Compléter le tableau suivant :

<i>Instructions à taper dans la console</i>	<i>Variable</i>	<i>Valeur et type de la donnée</i>
>>> age=20		
>>> prenom="Thomas"		
>>> taille_m=1.75		
>>> taille_m=100*taille_m		
>>> age=age+3		

9**Exercice**

Compléter le tableau suivant :

<i>Instructions à taper dans la console</i>	<i>Valeur obtenue et type de la donnée</i>
>>> age	
>>> age=40 >>> age	
>>> age=22 >>> age=age+1 >>> age	
>>> phrase1="Bonjour "+prenom >>> phrase1	
>>> phrase1=phrase1+"!" >>> phrase1	
>>> phrase2=prenom+" a "+str(age)+" ans" >>> phrase2	

👉 **Explication** 👉 (La fonction `input()`) La fonction `input()` suspend l'exécution des instructions jusqu'à ce que l'utilisateur entre une réponse au clavier. Elle renvoie alors cette réponse sous forme d'une **CHAÎNE DE CARACTÈRES**, que l'on affecte fréquemment à une variable.

Exemple Voici un exemple illustré et commenté du fonctionnement de la fonction `input()` qu'il faut parfaitement comprendre :

```

1 # on stocke dans la variable age ce que l'utilisateur va rentrer :
2 >>> age=input("Quel est ton âge ? ")
3 Quel est ton âge ? # c'est ce qui apparait après avoir exécuté la ligne précédente.\\
4 # Un curseur clignote et vous permet de rentrer votre âge, disons 18.
5
6 # la donnée est stockée dans la variable age sous la forme de chaîne de caractère !
7 >>> age
8 "18"
9
10 # pour la transformer en entier, vous pouvez la transtyper !
11 >>> age=int(age)
12
13 # vous récupérer l'entier 18
14 >>> age
15 18

```

👉 **Explication** 👉 Pour l'exercice suivant, il est fortement conseillé d'intuiter la valeur à obtenir et surtout le type de la donnée, puis de vérifier votre intuition avec la console.

10

Exercice

Compléter le tableau suivant :

Instructions à taper dans la console	Valeur obtenue et type de la donnée
<pre>>>> prenom=input("Entrez votre prénom : ") Thomas >>> prenom</pre>	
<pre>>>> numero=input("Entrez un numéro entre 1 et 5 : ") 3 >>> numero >>> numero=numero+1</pre>	
<pre>>>> numero=int(numero) >>> numero=numero+1</pre>	
<pre>>>> numero=int(input("Entrez un numéro entre 1 et 5 : ")) 3 >>> numero=numero+1 >>> numero</pre>	

👉 **Explication** 👉 (La fonction `print()`) On rappelle que la fonction `print()` permet d'afficher du texte à l'écran et que cet affichage est un `NoneType`.

11**Exercice**

Compléter le tableau suivant :

Instructions à taper dans la console	Valeur obtenue et type de la donnée
<code>>>> print(17+3)</code>	
<code>>>> type(print(3))</code>	
<code>>>> h,m,s=15,27,34</code> <code>>>> print("nb de secondes = ",h*3600+m*60+s)</code>	
<code>>>> x,y,z=3,100,45</code> <code>>>> print(x,y,z)</code>	
<code>>>> print("x = ", x,"y = ", y,"z = ", z)</code>	
<code>>>> print(x,y,z,sep=" ")</code>	
<code>>>> print(x,y,z,sep="; ")</code>	
<code>>>> print("x = ", x,"\ny = ", y,"\nz = ", z)</code>	