Nom :	Prénom :	Classe :

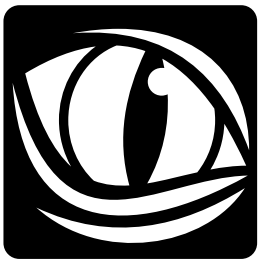
NSI 1re — Séquence d'exécution de code (calculs, commentaires)

Élément du programme : Constructions élémentaires.

Pourquoi faire des calculs avec Python ?

Pourquoi utiliser Python pour effectuer des calculs, alors qu'une simple calculette (non scientifique) peut faire le boulot ? Et bien parce que Python peut réaliser des calculs bien plus compliqués. Et puis, très prochainement nous pourrions demander à Python de faire des calculs dans le cadre d'un petit jeu, par exemple !

La calculette, c'est pratique pour les choses simples : additionner, soustraire, multiplier et diviser. Mais avec Python nous pouvons aller plus loin : traiter des milliers de nombres en même temps, calculer la moyenne de toutes les notes d'une classe.



Nous verrons également que Python permet de créer des « boucles », permettant de répéter des calculs plusieurs fois, sans avoir à les réécrire encore et encore. Python deviendra alors comme un assistant, qui fait le même travail plusieurs fois pour nous.

Opérateurs arithmétiques de base, en Python

En Python, nous pouvons faire toutes les opérations classiques des Mathématiques.
Le tableau ci-dessous les résume :

Opération	Symbole en Python	Commentaire
Addition	+	Symbole « plus »
Soustraction	-	Symbole « tiret du six »
Multiplication	*	Symbole « astérisque »
Division	/	Symbole « slash »

Voici comment nous demandons à Python de réaliser une addition :

```
>>> 1 + 1
2
```

La première ligne de l'exemple ci-dessus est précédée des symboles `>>>`
dans le monde de Python, cela signifie que **cette ligne a été envoyée à Python** (appelée aussi le « prompt »).

Ici nous avons envoyé la commande `1 + 1` à Python.
Et Python nous a répondu : `2`

Dans le monde de Python, la ligne qui n'est pas précédée des symboles `>>>` correspond à ce que Python répond.

➤ Exercice 1 (à vous de jouer)

Dans les exemples ci-dessous, devinez ce qui a été envoyé à Python, et ce que Python a répondu.

Exemples Python	Votre analyse (à compléter)
`>>> 1 – 1` `0`	Ligne envoyée à Python : Réponse de Python :
`>>> 3 * 3` `9`	Ligne envoyée à Python : Réponse de Python :
`>>> 1 + 2 * 3` `7`	Ligne envoyée à Python : Réponse de Python :

➤ Exercice 2

Observez le code ci-dessous :

```
>>> 10 / 2
5.0
```

Quel est le nom de l’opération arithmétique qui vient d’être réalisée ?

Quelle observation pouvez-vous formuler concernant le résultat retourné par Python ?

➤ Exercice 3

Observez les deux commandes ci-dessous :

Commande n°1

```
>>> 1 + 2 * 3
7
```

Commande n°2

```
>>> (1 + 2) * 3
9
```

À partir de ces deux commandes, quelle observation pouvez-vous formuler concernant le résultat retourné par Python ? Quelle différence réside entre ces deux commandes ? Cette règle est-elle également présente en mathématiques ?

➤ Exercice 4

Avant de passer à d'autres opérateurs arithmétiques, notons qu'il est également possible *d'imbriquer* des parenthèses, comme dans l'exemple qui suit :

```
>>> 5 * (2 + 3) - (4 - (1 + 2))
```

À votre avis, dans l'exemple ci-dessus, quel résultat retourne Python ?

Autres opérateurs arithmétiques de Python

En plus de l'addition, la soustraction, la multiplication et la division, Python propose des opérateurs permettant de calculer :

- Le résultat **entier** d'une division (son quotient).
- Le **reste** d'une division.
- Une puissance.

Le tableau si-dessous les résume :

Opération	Symbole en Python	Commentaire
Résultat entier d'une division	//	Deux symboles « slash » à la suite
Reste d'une division	%	Symbole « pourcentage »
Puissance	**	Deux symboles « astérisque »

Testons ces nouveaux opérateurs arithmétiques avec Python :

Prenons un exemple : c'est l'heure de manger, nous sommes 4 personnes, mais nous avons 14 sushis devant nous. Combien faut-il donner de sushis (entiers) par personne ?

Demandons à Python, grâce à l'opérateur `//` qui permet d'obtenir le **résultat entier d'une division** :

```
>>> 14 // 4
3
```

Python nous dit qu'il faut donner 3 sushis entiers par personne. Parfait !

Combien reste-t'il de sushis ?

Demandons à Python, grâce à l'opérateur `%` qui permet d'obtenir le **reste d'une division** :

```
>>> 14 % 4
2
```

Un autre exemple, pour tester les puissances :
Combien de résultats différents pouvons-nous obtenir, lorsque nous lançons 3 fois de suite un dé à six faces ?

Lorsque je lance un dé à six faces trois fois, chaque lancer peut donner l'un des six résultats possibles (1, 2, 3, 4, 5 ou 6). Pour déterminer le nombre total de résultats différents, je peux utiliser la formule suivante :

Nombre total de résultats = Nombre de résultats par lancer $^{\text{Nombre de lancers}}$

Ce qui se note : $6^3 = 6 \times 6 \times 6 = 216$

Demandons à Python la confirmation de ce résultat, grâce à l'opérateur `**` qui permet d'obtenir une puissance :

```
>>> 6 ** 3
216
```

Tant que ça ? Nous pouvons obtenir 216 résultats différents en seulement 3 jets de dé. Impressionnant.

➤ Exercice 5

Liam a recopié rapidement des lignes de commandes Python.
Le problème, c'est qu'il était en même temps en train de regarder des vidéos sur un réseau social bien connu !

Pour chacune des lignes Python ci-dessous, pouvez-vous dire si Liam s'est trompé ou non ? Merci pour lui.

Python (recopié par Liam)	Correct ? (à compléter et justifier)
`>>> 12 + 3 * 2` `18`	

Python (recopié par Liam)	Correct ? (à compléter et justifier)
`>>> 12 + 3 * 2 / 6` `13`	
`>>> 3 * (7 + 3)` `30`	
`>>> 12 – 2**3` `4`	
`>>> 50 // 8` `6`	
`>>> 50 % 8` `1`	
`>>> 3 * (1 + 2) - (3 - (2 + 2))` `0`	
`>>> 15 % 3` `0`	
`>>> 20 // (2 + 1)` `6`	
`>>> 20 // 5` `4.0`	
`>>> 20 / 5` `4`	
`>>> 10 ** 3` `1000`	
`>>> 20 % (2 + 1)` `2`	
`>>> 10 / 2` `5`	
`>>> 81 / 9` `9.0`	

Séquence d’exécution de code

Il est possible de créer un programme constitué de plusieurs lignes de codes.
Dans ce cas, Python va exécuter chaque ligne du programme, l’une après l’autre.

Voici un exemple de programme Python :

```
>>> 10 + 10
20
>>> 30 – 20
10
>>> 50 / 10
5.0
```

Pour y voir un peu plus clair, nous allons ajouter des numéros de lignes :

```
1  >>> 10 + 10
2  20
3  >>> 30 – 20
4  10
5  >>> 50 / 10
6  5.0
```

Dans ce programme constitué de plusieurs lignes de codes, chaque ligne est exécutée l’une après l’autre :

- Ligne 1 : on demande à Python le résultat de `10 + 10`
- Ligne 2 : Python répond `20`
- Ligne 3 : on demande à Python le résultat de `30 – 20`
- Ligne 4 : Python répond `10`
- Ligne 5 : on demande à Python le résultat de `50 / 10`
- Ligne 6 : Python répond `5.0`

On parle de « **séquence d'exécution** de code » car un programme est constitué d’une suite d’instructions **exécutées** les unes après les autres **dans l’ordre** où elles apparaissent dans le code.

Une séquence d’exécution c’est le fait que le programme suive un parcours linéaire, instruction après instruction, ce qui permet de définir précisément le comportement du programme et de prédire ce qu’il va faire, étape par étape.

➤ Exercice 6

Quels sont les trois résultats affichés par la séquence ci-dessous ?

```
1  12 + 4
2  30 / 10
3  30 // 10
```

Résultats (à compléter)

Commentaires

En Python, comme dans d'autres langages de programmation, il est possible d'ajouter des **lignes qui ne seront pas exécutées**. Ces lignes sont appelées des lignes de **commentaires**.

Les commentaires accompagnent des lignes de codes et servent à expliquer brièvement ce que l'on cherche à faire.

Prenons par exemple le programme suivant :

```
1  import random
2  h = random.randint(1, 6)
3  print(h)
```

Le code ci-dessus est loin d'être clair !

Voici maintenant le **même programme**, mais **agrémenté de lignes de commentaires** :

```
1  # charge la bibliothèque appelée « random »
2  import random
3  # affecte une valeur aléatoire à une variable « h »
4  h = random.randint(1, 6)
5  # affiche la valeur stockée dans la variable « h »
6  print(h)
```

Dans le code ci-dessus, les lignes de commentaires sont précédées du symbole `#`. Ces lignes de commentaires ne sont pas exécutées par Python, mais elles permettent d'expliquer ce que l'on cherche à faire.

En l'occurrence, grâce à ces commentaires, il nous est possible de comprendre que ce programme *charge une bibliothèque appelée « random »* ligne 2, qu'il *affecte une valeur aléatoire à une variable « h »* ligne 4, puis qu'il *affiche la valeur stockée dans la variable « h »* ligne 6.

Commentaires sur une même ligne

Notons qu'il est possible d'ajouter un commentaire (non exécuté) juste à côté d'une commande (exécutée).

Voici un exemple de programme où des commentaires ont été ajoutés sur la même ligne que le code :

```
10 + 10 # addition
234 – 4 # soustraction
10 * 10 # multiplication
100 / 3 # division
10 // 3 # résultat entier d'une division
100 % 3 # reste d'une division (en math, on dit aussi « modulo »)
10 ** 4 # puissance
```

Dans le programme ci-dessus, Python va exécuter chaque instruction l'une après l'autre, mais **sans exécuter les parties commençant par le symbole** `#`.

Ce programme est de fait lisible pour les êtres humains (grâce aux commentaires), tout en restant fonctionnel (grâce au code écrit avant les commentaires).

➤ Exercice 7

Essayez de deviner ce que Python calcule et affiche dans les programmes suivants :

Programmes	Résultats (à compléter)
`100 // 10` `3 * 2`	10 6
`60 // 10` `60 / 10`	
`# 3 + 3` `6 + 6`	
`10 + 10` `# 20 + 20`	
`50 / 5 # 10 + 10` `20 + 30`	
`60 // 10` `#140 + 2`	

`18 + 2 # addition 20 + 5`
`20 – 10 # soustraction`
`#10 + 5 # addition`	

➤ Exercice 8

Ajoutez les lignes de commentaires appropriées au code fourni.

Exemple :

Avant	Après

`#`
`64 + 16`
`#`
`120 / 10`
	`# addition`
	`64 + 16`
	`# division`
	`120 / 10`

À vous de jouer :

Avant	Après (à compléter)

`#`
`10 ** 4`
`#`
`123 * 210`
`#`
`123 + 17`
`#`
`10 + (2 + 8)`
`#`
`140 // 12`
`#`
`18 / 4`
`#`
`20 % 3`
	`#`
	`10 ** 4`
	`#`
	`123 * 210`
	`#`
	`123 + 17`
	`#`
	`10 + (2 + 8)`
	`#`
	`140 // 12`
	`#`
	`18 / 4`
	`#`
	`20 % 3`

Vous avez terminé cet exo ? Alors vous pouvez prendre une *petite* pause de 5 minutes en passant voir vos camarades pour leur proposer votre aide *dans le calme*.

Vous poursuivrez sur votre copie juste après.

Pour aller plus loin

🌶️ Supplément pimenté – Quels sont les résultats retournés par Python dans les programmes ci-dessous ?

Python	Résultat (à compléter)	Python	Résultat (à compléter)
`5 + 3` `#10 + 10`		`#5 + 10` `(5 + 3) * 2`	

`10 - 4`		`(10 - 2) / 2`	
`7 * 6 # 3 * 3`		`(7 * 3) % 5`	
`20 / 4`		`2 ** 4 + 3`	
`15 // 4`		`20 // 3 - 1`	
`2 ** 3`		`10 / 3 + 1`	
`10 % 3`		`3 + 2 ** 2`	
`10 - 2 * 3`		`10 % 3 // 1`	
`21 // 2`		`(2 ** 5) % 3`	
`10 % 4 + 2`		`(10 + 5 - 3) / 2`	
`3 * (2 ** 3)`		`(6 * 4) / 3 % 2`	
`10 - (5 % 3)`		`(3 ** 2) + (5 - 2)`	
`(5 + 3) * (10 - 2)`		`(20 // 4) * 3 + 1`	
`2 ** (3 // 2)`		`(10 / 2) - (2 ** 2)`	
`10 // 3 + 1`		`(10 + 5) % 4 // 1`	
`20 + 5 / 2`		`27 / 3`	
`60 // 10 + 2`		`27 // 3`	
`60 // (10 + 2)`		`10 ** 6`	
`30 % 9`		`2 * 3 ** 2`	