Bases en algorithmique

1 Les notions de base en algorithmique et en programmation 2

2 Type d'une variable 2

3 Affectation d'une valeur à une variable 2

4 La structure conditionnelle Si alors Sinon 3

5 La boucle bornée Pour 3

6 La boucle non bornée Tant que 4

7 Les fonctions 5

8 La programmation sur un environnement Python 6

8.1 Les variables 6

8.2 Les fonctions 8

8.3 La structure if else 13

8.4 La boucle for 14

8.5 La boucle while 14

Bases en algorithmique

1 Les notions de base en algorithmique et en programmation

- Un **algorithme** est constitué de commandes nommées **instructions**.
- Une **instruction** peut contenir :
    - des **mots clés** par exemple `si`
    - des **variables** qui contiennent des valeurs par exemple `i`, `nombre_boutons`.
    - des **connecteurs de** la **logique** par exemple `et`, ou, non.
- Un **algorithme** peut s'écrire :
    - en **langage naturel** par exemple `si nombre_boutons > 4 alors`
    - en **langage informatique** par exemple `if nombre_boutons > 4:`
- La **programmation** est la traduction dans un langage informatique (par exemple Python, Java) d'un algorithme pour qu'il soit compris par une machine (ordinateur, téléphone portable …).

2 Type d'une variable

- Le type d'une variable renseigne sur ce qu'elle contient.

Exemples :

Nom de la variable	Contenu	Type de variable
`article`	`'Ramette de papier A4'`	*Chaîne de caractères*
`nombre_invites`	`24`	*Entier*
`poids`	`27.812`	*A virgule flottante* (on dit parfois *flottant*)

3 Affectation d'une valeur à une variable

Pour qu'une variable puisse contenir une valeur, on utilise l'affection de valeur. Le symbole est une flèche vers la gauche et se lit "prend la valeur".

Exemple :

`nombre_boutons` ← `4`

signifie "nombre_boutons **prend la valeur** 4".

4 La structure conditionnelle Si alors Sinon

Pour aiguiller dans différentes directions la suite du déroulement du programme selon la réalisation ou non d'une condition, on utilise des instructions conditionnelles.

Exemple en langage naturel :

```
si nombre_boutons > 10 alors
      prix ← nombre_boutons × 0,40
sinon si nombre_boutons > 5 alors
      prix ← nombre_boutons × 0,50
sinon
      prix ← nombre_boutons × 0,60
```

5 La boucle bornée Pour

Faire une boucle bornée signifie faire faire un nombre de tours de boucle déterminé à l'avance.

Exemple en langage naturel avec un compteur `i` :

```
somme ← 0
pour i allant de 3 à 6
      somme ← somme + 50 + i
```

Méthode pour dresser un tableau d'évolution des variables

1. On dresse un tableau ayant, en en-tête des lignes, le nom des variables utilisées *autres que le compteur*.
2. La première colonne contient l'initialisation c'est-à-dire la valeur donnée aux variables au départ
3. Les colonnes suivantes sont appelées `i = ...` pour chaque tour de boucle Pour.

Dans cet exemple le tableau d'évolution des variables est :

	initialisation	$i = 3$	$i = 4$	$i = 5$	$i = 6$
somme	**0**	$0 + 50 + 3 = \mathbf{53}$	$53 + 50 + 4 = \mathbf{107}$	$107 + 50 + 5 = \mathbf{162}$	$162 + 50 + 6 = \mathbf{218}$

A la sortie de la boucle Pour la variable somme contient la valeur 218.

Il y a 4 tours de boucle.

Ce nombre est déterminé par `i` allant de 3 à 6 puisque `i` prend successivement les valeurs 3, 4, 5, 6.

6 La boucle non bornée Tant que

Programmer une boucle non bornée signifie programmer de faire des tours de boucle tant qu'une condition est vraie. Dès que la condition devient fausse, il n'y a plus de nouveau tour et on sort de la boucle. On n'a donc pas besoin prévoir un compteur de tours de boucle.

Exemple en langage naturel :

```
somme ← 10
tant que somme > 0
      somme ← somme - 3
```

Méthode pour dresser un tableau d'évolution des variables

1. On dresse un tableau ayant, en en-tête des lignes, le nom des variables utilisées.
2. La première colonne contient l'initialisation c'est-à-dire la valeur donnée aux variables au départ
3. La dernière ligne contient la condition de maintien dans la boule Tant que.

Quand la condition est fausse, on sort de la boucle.

Dans cet exemple le tableau d'évolution des variables est :

	initialisation				
somme	**10**	$10-3=\mathbf{7}$	$7-3=4$	$4-3=\mathbf{1}$	$1-3=\mathbf{-2}$
Somme > 0	Vrai	Vrai	Vrai	Vrai	Faux

Après le moment où somme prend la valeur -2, la condition somme > 0 de maintien dans la boucle devient fausse. Donc il n'y a pas de nouveau tour de boucle.

A la sortie de la boucle Tant que la variable somme contient donc la valeur -2.

Au total, il y a 4 tours de boucle. Mais ce nombre n'est pas déterminé à l'avance.

On exprime le fait que le nombre de tours n'est pas déterminé à l'avance en disant que la boucle *Tant que* est une "boucle non bornée".

Au contraire la boucle *Pour* est une "boucle bornée".

7 Les fonctions

Programmer une fonction signifie regrouper un bloc d'instructions en lui donnant un nom de fonction. Dès lors, chaque fois qu'on aura besoin d'exécuter ce bloc d'instructions, il suffira de faire appel à la fonction par son nom et en indiquant pour quelles valeurs des paramètres on veut l'exécuter.

Exemple en langage naturel d'une fonction ayant comme paramètres x et y :

```
fonction calcul_produit(x, y)
      produit ← x * y
      renvoyer produit
```

Remarque :

Les variables x et y dont les valeurs sont communiquées à la fonction pour qu'elle effectue son travail sont les **paramètres** de la fonction. On dit aussi que x et y sont les **arguments** de la fonction.

Méthode pour écrire une fonction

1. On identifie les paramètres de la fonction c'est-à-dire les variables qu'elle a besoin de connaitre pour effectuer son travail.
2. On identifie la valeur de retour c'est-à-dire quelle variable la fonction doit renvoyer.
3. On écrit la fonction.

Utilisation : Une fois la fonction écrite, on peut lui faire appel.

Exemple :

```
fonction calcul_produit(x, y)
      produit ← x * y
      renvoyer produit

a ← 5
b ← 8
c ← 3 + calcul_produit(a, b)
```

Quelle est la valeur de c après l'exécution de ce programme ?

Réponse :

- `calcul_produit(a, b)` fait appel à la fonction `calcul_produit`.

Plus précisément, l'appel `calcul_produit(a, b)` est équivalent à `calcul_produit(5, 8)` car au moment de l'appel, `a` vaut 5 et `b` vaut 8.

- `calcul_produit` calcule la variable `produit` qui vaut `x * y = 40`.
- `calcul_produit` renvoie la valeur `40`

La variable c prend alors la valeur `3 + 40` c'est à dire 43.

8 La programmation sur un environnement Python

8.1 Les variables

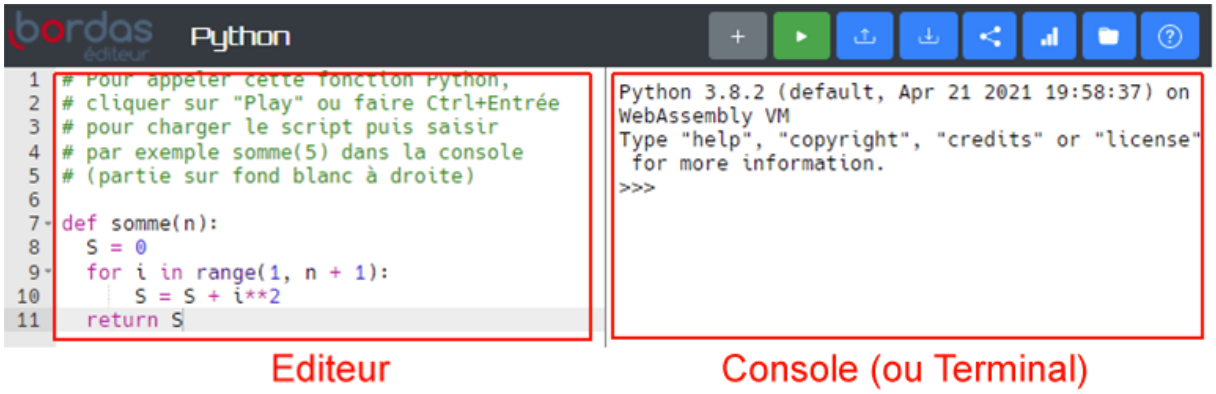
Syntaxe : Affectation d'une nouvelle valeur à une variable

En langage naturel : `nombre_boutons ← 4`
 	
On peut affecter l'ancienne valeur additionnée de 1 :
`a ← a + 1`
 	
ou en lui soustrayant 1 :
`b ← b - 1`
	En Python : `nombre_boutons = 4`
	
	`a = a + 1` ou avec le **raccourci** `+=` : `a += 1`
	
	`b = b - 1` ou avec le **raccourci** `-=` : `b -= 1`

TRAVAIL A FAIRE

1. Ouvrez le navigateur Firefox et saisissez dans la barre d'adresses cahier-nsi.fr/webpython

A l'ouverture, l'environnement webpython affiche ceci :



- **A gauche** se trouve la fenêtre "**éditeu**r" c'est-à-dire une fenêtre où nous pouvons écrire des programmes et les enregistrer sur notre ordinateur.
- **A droite** se trouve une fenêtre "**console**" c'est-à-dire une fenêtre où nous pouvons exécuter un programme que nous venons juste d'écrire ou que nous venons juste de charger dans l'éditeur depuis notre ordinateur.

Remarques :

L'éditeur contient un programme de démonstration. On ne s'en occupe pas ici.

La console contient une dernière ligne qui commence par trois chevrons >>>

Les trois chevrons indiquent que la console est prête à recevoir des commandes en Python et à les exécuter.

2. Dans la console juste après les trois chevrons >>> saisissez `a = 22` puis appuyez sur la touche Entrée de votre clavier.

Une nouvelle ligne avec trois chevrons apparait.

3. Sur la nouvelle ligne commençant par >>> saisissez `b = 6` puis appuyez sur la touche Entrée .

4. Sur la nouvelle ligne commençant par >>> saisissez `a + b` puis appuyez sur la touche Entrée.

Une ligne avec le résultat de l'addition des variables a et b apparait.

Le résultat est bien celui de l'addition 22 + 6

Le signe = est le symbole d'affectation d'une valeur à une variable. `a = 22` affecte la valeur 22 à la variable a et `b = 6` affecte la valeur 6 à la variable b.

5. Sur la nouvelle ligne commençant par >>> saisissez `a - b` puis appuyez sur la touche Entrée.

6. Sur la nouvelle ligne commençant par >>> saisissez `a * b` puis appuyez sur la touche Entrée.

L'étoile * est le symbole de la multiplication.

7. Sur la nouvelle ligne commençant par >>> saisissez `a / b` puis appuyez sur la touche Entrée.

La barre de fraction / est le symbole de la division ordinaire c'est-à-dire qu'elle donne un quotient généralement sous la forme d'un nombre à virgule flottante.

8. Sur la nouvelle ligne commençant par >>> saisissez `a**3` puis appuyez sur la touche Entrée.

Les deux étoiles ** est le symbole de la mise à la puissance. `a**3` signifie `a * a * a`. En mathématiques cela s'écrit a^3.

9. Saisissez les deux lignes

```
>>> from math import sqrt
>>> sqrt(a)
```

La fonction `sqrt` (*square root* c'est-à-dire *racine carrée*) n'existe pas dans Python. C'est pourquoi on doit écrire d'abord la ligne :
`from math import sqrt` qui importe de la bibliothèque `math` la fonction `sqrt`.
Une bibliothèque est un fichier qui contient des fonctions préfabriquées. La bibliothèque `math` contient la fonction `sqrt`

10. Saisissez la ligne

```
>>> a // b
```

La double barre de fraction // est le symbole de la division euclidienne (appelée aussi division entière). Le quotient est toujours un entier. Ici le quotient de 22 divisé par 6 est 3.

11. Saisissez la ligne

```
>>> a % b
```

Le pourcent % est le symbole du reste de la division euclidienne. Le reste est toujours un entier positif. Ici le reste de 22 divisé par 6 est 4. En effet $22 = 6 \times 3 + 4$.

Remarques

- Le **chiffre des unités** d'un nombre N est **le reste** de la division de N par 10. ***Ex. :*** `64 % 10` donne **4**.
- Pour obtenir le nombre de dizaines, on soustrait à N le nombre d'unités et on divise par 10.
  Exemple : `(64 - 64%10) // 10` donne **6** qui est le nombre de dizaines.

12. Saisissez la ligne

```
>>> longueur, largeur = 4, 3
>>> longueur * largeur
```

On voit s'afficher 12 qui est le produit 4 * 3.
La virgule permet d'affecter des valeurs à plusieurs variables en une seule ligne.

13. Dans la console juste après les trois chevrons >>> saisissez `longueur` puis appuyez sur la touche Entrée de votre clavier.

La valeur 4 de la variable `longueur` s'affiche.

14. Saisissez les lignes

```
>>> longueur, largeur = largeur, longueur
>>> longueur
```

La valeur 3 de la variable `longueur` s'affiche.
La virgule permet d'échanger plusieurs variables en une seule ligne.

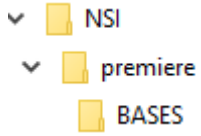
8.2 Les fonctions

En langage naturel	*En Python*
fonction calcul_produit(x, y) produit ← x * y **renvoyer** produit	def calcul_produit(x, y): produit = x * y return produit

TRAVAIL A FAIRE : Partie 1 Créer le script BASES et envoyer le lien par messagerie

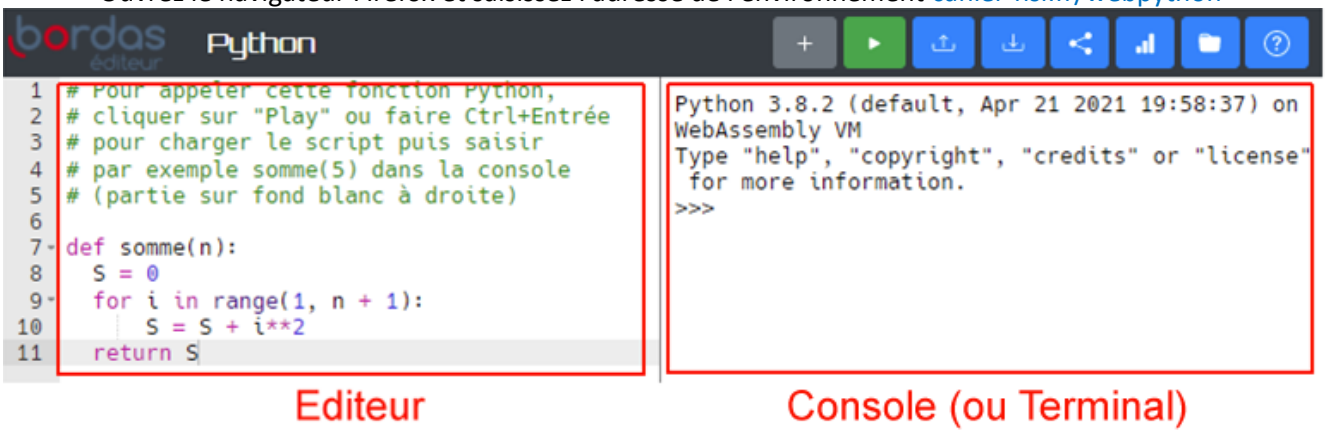
0. Créer le dossier BASES pour y enregistrer toutes les fonctions de ce chapitre

- Sur le réseau du lycée, sur votre lecteur personnel P: qui porte votre nom, créez l'arborescence :

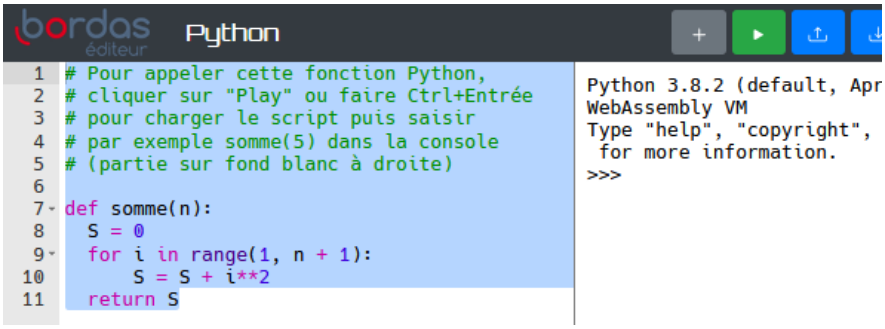


1. Préparer l'environnement

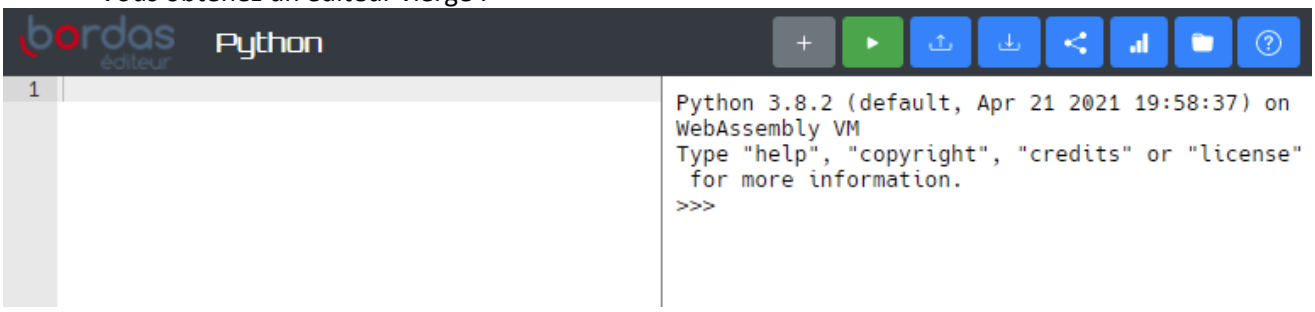
- Ouvrez le navigateur Firefox et saisissez l'adresse de l'environnement cahier-nsi.fr/webpython



- **Sélectionnez** tout le programme déjà présent dans l'éditeur et appuyez sur la touche **suppr** pour le supprimer.



- Vous obtenez un éditeur vierge :



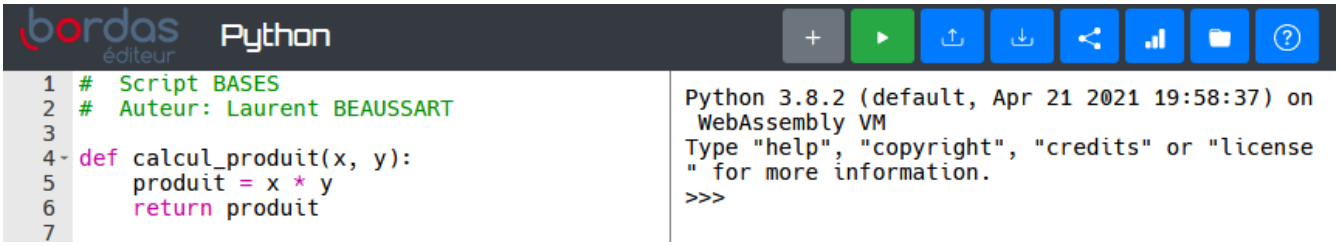
2. Écrire le programme en Python

- Commencez à écrire ceci :

```
#  Script BASES
#  Auteur: <ICI VOTRE Prénom Nom>
```

- Passez **deux lignes** puis saisissez :

```
def calcul_produit(x, y):
    produit = x * y
    return produit
```



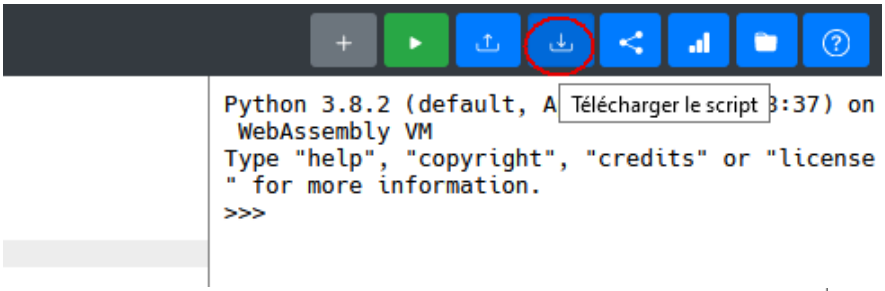
Remarques :

- Un **commentaire** en Python commence toujours par le symbole **#** (dièse). Ce dièse indique à l'interpréteur Python qu'il ne faut pas prendre ce commentaire pour une instruction.
- Lorsqu'un nom de fonction ou un nom de variable contient plusieurs mots, on les sépare avec _ (trait de soulignement dit "underscore"). De plus on ne met pas d'accent.

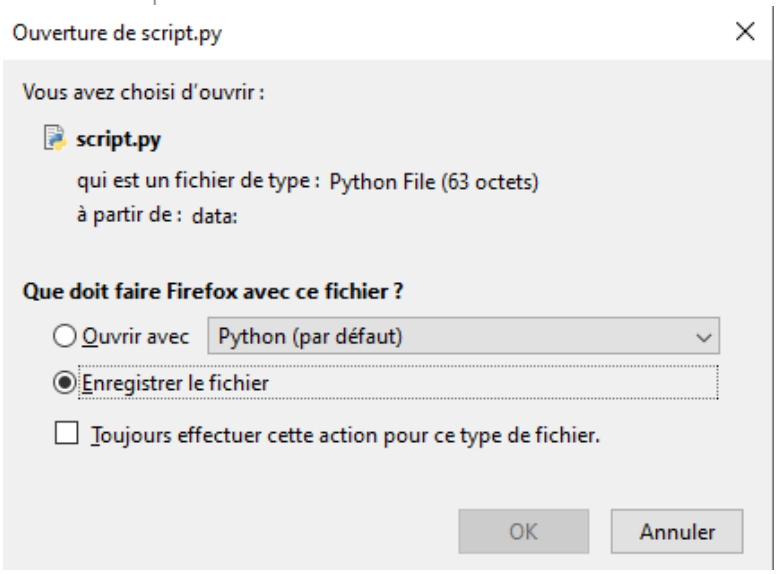
3. Enregistrer sur le disque dur le programme

- Cliquez sur le bouton bleu avec une flèche vers le bas "Télécharger le script".

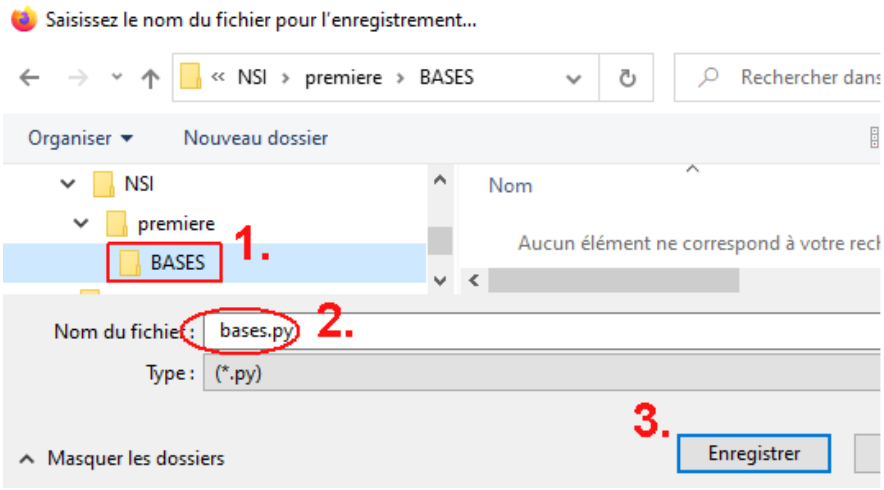
Télécharger signifie **Récupérer sur le site de l'éditeur Python en ligne**.



- Dans la boite de dialogue qui s'ouvre, cochez "Enregistrer le fichier" et cliquez sur le bouton "OK"

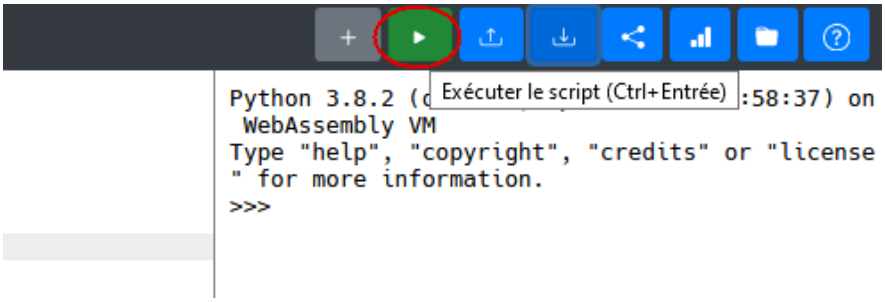


- 1. Sélectionnez le dossier préalablement créé NSI/premiere/BASES
- 2. Renommez script.py en **bases.py**
- 3. Cliquez sur le bouton "Enregistrer"



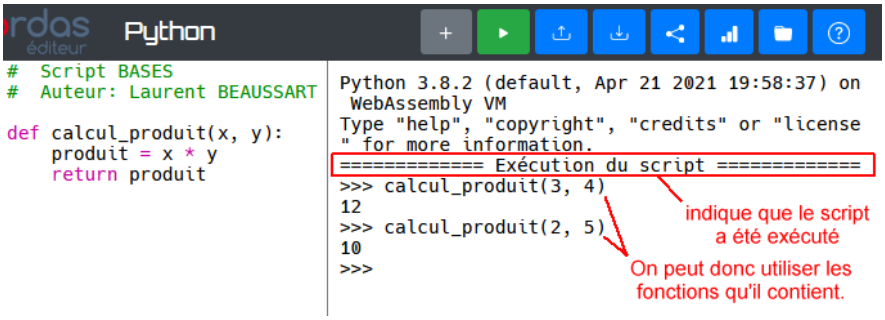
4. Exécuter le programme

Exécutez ce script **bases.py** en cliquant sur la flèche verte "Exécuter".



- La **mention**
  === Exécution du script ===
  dans la console signale que le script *a été exécuté*.

- Les **trois chevrons** qui suivent indiquent que la console est *prête à prendre des commandes* Python utilisant le script BASES et les fonctions qu'il contient.



- Par exemple, essayez >>>**`calcul_produit(3, 4)`** et appuyez sur Entrée.

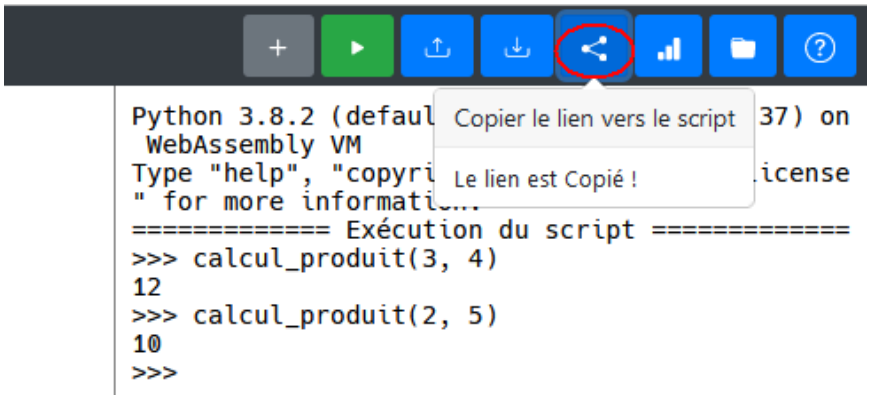
Remarque :

Si le programme ne fonctionne pas, modifiez-le puis cliquez sur la flèche verte "Exécuter".
Lorsque le programme est correct, réenregistrez-le sur NSI/premiere/BASES en tant que **bases.py**

5. Envoyer le lien sur Ecole Directe

- Cliquez sur le bouton "copie du lien".

Ceci a pour effet de copier dans le presse-papier un lien vers votre programme pour pouvoir le partager.



- Allez dans la messagerie d'Ecole Directe.
- Cliquez sur Nouveau message
- Remplissez les champs :
  A M. BEAUSSART
  Sujet Lien vers le script BASES

- Dans le corps du message écrivez *Voici le lien* et Collez avec un clic droit le contenu du presse papier. A ce moment un lien assez long apparait dans le message.

Exemple :

Voici le lien

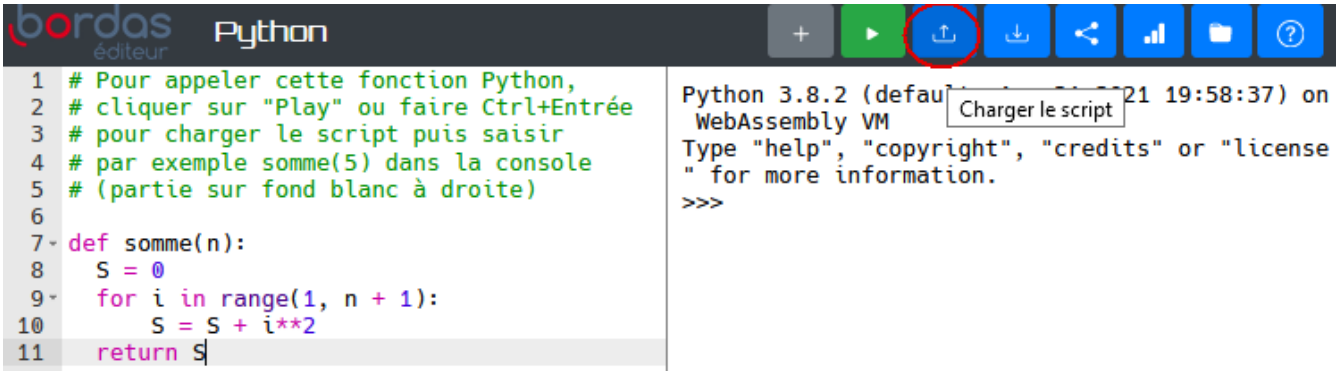
https://www.cahier-nsi.fr/webpython/#eJxTVlAITi7KLChRcPRx9zfkUlZQcCwtSS0tslLwSSwtSs0rUXBKT

- Cliquez sur le bouton bleu en bas de la page "Envoyer".
- Fermez le navigateur Firefox.

TRAVAIL A FAIRE : Partie 2 Rouvrir le script BASES, le compléter et envoyer le lien

1. Charger le script bases.py

- Ouvrez le navigateur Firefox et saisissez l'adresse de l'environnement cahier-nsi.fr/webpython
- Cliquez sur le bouton "Charger le script":

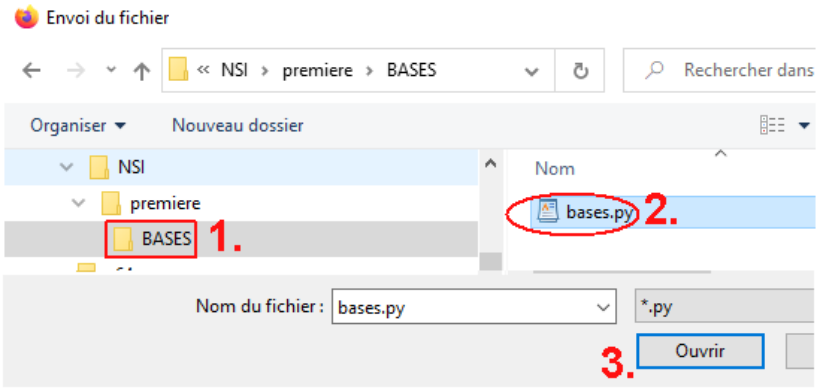


Dans la fenêtre qui s'ouvre :

1. Cherchez votre dossier personnel.

2. Cliquez sur le script bases.py

3. Cliquez sur le bouton "Ouvrir".

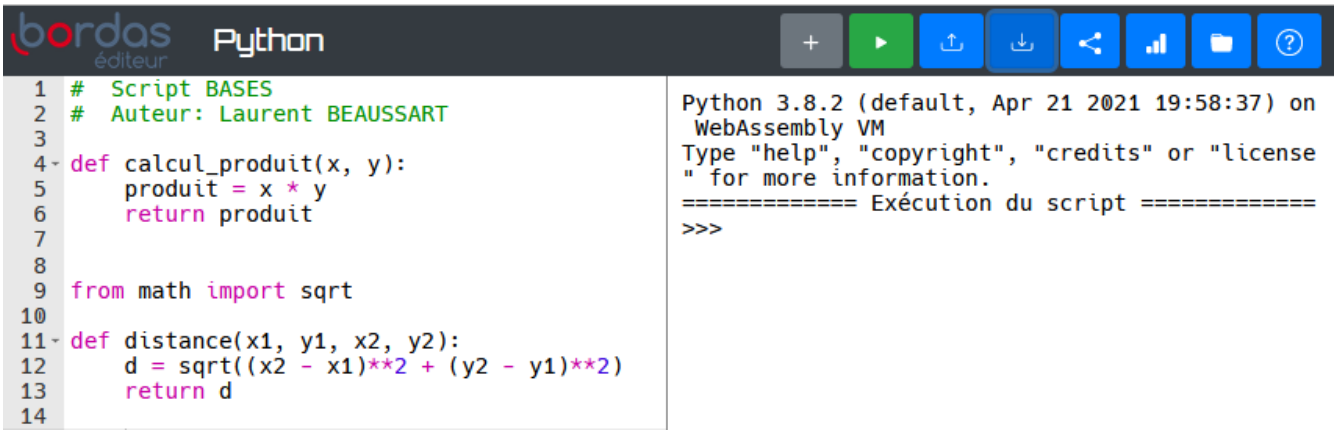


2. Compléter le programme en Python

- Vous retrouvez la fonction `calcul_produit(x, y)` déjà écrite dans le script BASES.
- Passez **deux lignes** et écrivez la nouvelle fonction
- Commencez à écrire le mot clé `from` **contre de la marge**.

```
from math import sqrt
def distance(x1, y1, x2, y2):
    d = sqrt((x2 - x1)**2 + (y2 - y1)**2)
    return d
```

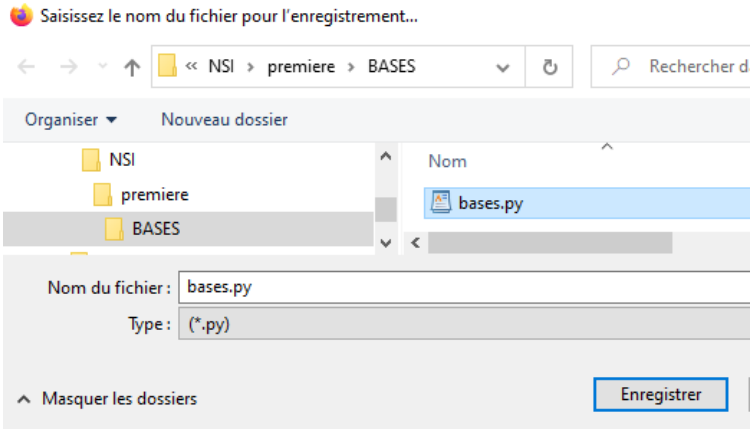
Vous obtenez :



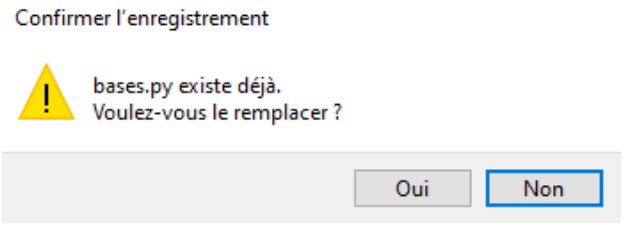
3. Enregistrer sur le disque dur le programme

Cliquez sur le bouton bleu avec une flèche vers le bas "Télécharger le script".

- 1. Sélectionnez le dossier préalablement créé NSI/premiere/BASES
- 2. Cliquez sur le nom du fichier en **bases.py déjà présent.**
- 3. Cliquez sur le bouton "Enregistrer"



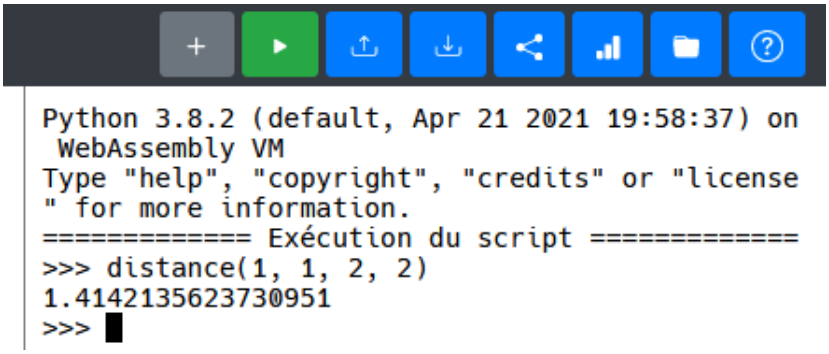
- Puisque bases.py existe déjà, il faut confirmer l'enregistrement



4. Exécuter le programme

Exécutez ce script **bases.py** en cliquant sur la flèche verte "Exécuter".

- La **mention**
  === Exécution du script ===
  dans la console signale que le script *a été exécuté*.
- Testez la fonction `distance` pour quelques valeurs de coordonnées.
- Par exemple pour les points $A(1\,;1)$ et $B(2\,;2)$ $AB \approx 1{,}414$.



Remarque :
Si le programme ne fonctionne pas, modifiez-le puis cliquez sur la flèche verte "Exécuter".
Lorsque le programme est correct, réenregistrez-le sur NSI/premiere/BASES en tant que **bases.py**

5. Envoyer le lien sur Ecole Directe

- Cliquez sur le bouton "copie du lien".
- Allez dans la messagerie d'Ecole Directe.
- Cliquez sur Nouveau message
- Remplissez les champs :
  A M. BEAUSSART
  Sujet Lien vers le script BASES
- Dans le corps du message écrivez *Voici le lien* et Collez avec un clic droit le contenu du presse papier. A ce moment un lien assez long apparait dans le message.
- Cliquez sur le bouton bleu en bas de la page "Envoyer".

8.3 La structure if else

Syntaxe

En langage naturel	***En Python***

fonction prix_boutons(nombre_boutons)
  si nombre_boutons < 5 **alors**
  prix ← 0.60 * nombre_boutons
  sinon si 5 <= nombre_boutons < 10 **alors**
  prix ← 0.50 * nombre_boutons
  sinon
  prix ← 0.40 * nombre_boutons
  renvoyer prix
	def prix_boutons(nombre_boutons):
	  if nombre_boutons < 5:
	  prix = 0.60 * nombre_boutons
	  elif 5 <= nombre_boutons < 10:
	  prix = 0.50 * nombre_boutons
	  else:
	  prix = 0.40 * nombre_boutons
	  return prix

8.4 La boucle for

Syntaxe

En langage naturel	***En Python***

`somme ← 0`
`pour i allant de 3 à 6`
`    somme ← somme + 50 + i`
	`Somme = 0`
	`for i in range(3, 7):`
	`    somme = somme + 50 + i`

Remarques :

La fonction `range()` permet d'énumérer le nombre de tours de boucles for. Elle peut être appelée de plusieurs façons :

- `range(n)` où n est un entier fait prendre à la variable les valeurs entières de 0 à $n-1$, donc n valeurs.
- `range(n, m)` où n et m sont des entiers fait prendre à la variable les valeurs entières de n à $m-1$.
- `range(n, m, k)` où n, m et k sont des entiers fait prendre à la variable les valeurs entières de n à $m-1$ avec un pas de k

Exemple :

`for i in range(9):` `    print(i)`	affiche 0, 1, 2, 3, 4, 5, 6, 7, 8 (9 valeurs donc 9 tours de boucle).
`for i in range(3, 9):` `    print(i)`	affiche 3, 4, 5, 6, 7, 8.
`for i in range(3, 9, 2):` `    print(i)`	affiche 3, 5, 7.

8.5 La boucle while

Syntaxe

En langage naturel	***En Python***

`somme ← 10`
`tant que somme > 0`
`    somme ← somme - 3`
	`somme = 10`
	`while somme > 0:`
	`    somme = somme - 3`

Remarque :

La condition de maintien dans la boucle (dans l'exemple c'est **`somme > 0`**) doit être modifiée dans la boucle jusqu'à devenir fausse à un moment donné. Sinon la boucle est infinie et l'exécution du programme reste "bloquée dans la boucle while".