Coanim 1- Bases pour débuter avec Python

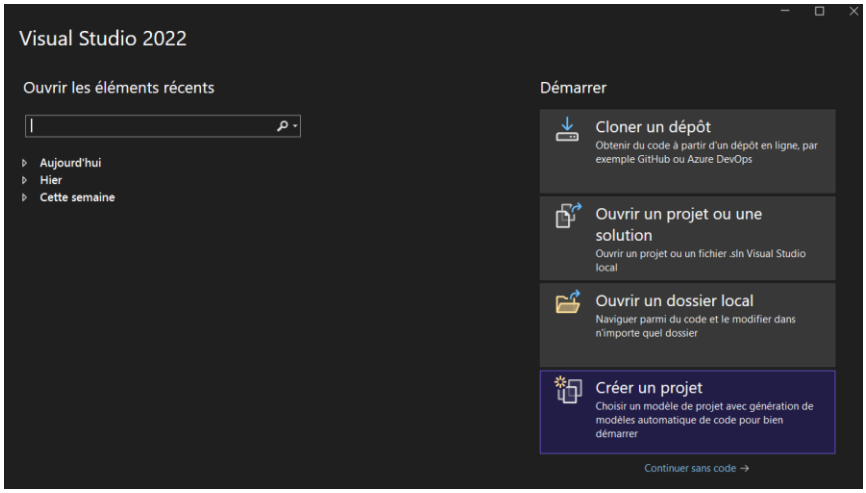
On voit dans ce chapitre comment créer des premiers scripts Python. L'objectif étant de créer une application qui permette de saisir des valeurs, de les stocker dans une liste python et d'en calculer la moyenne et l'écart-type.

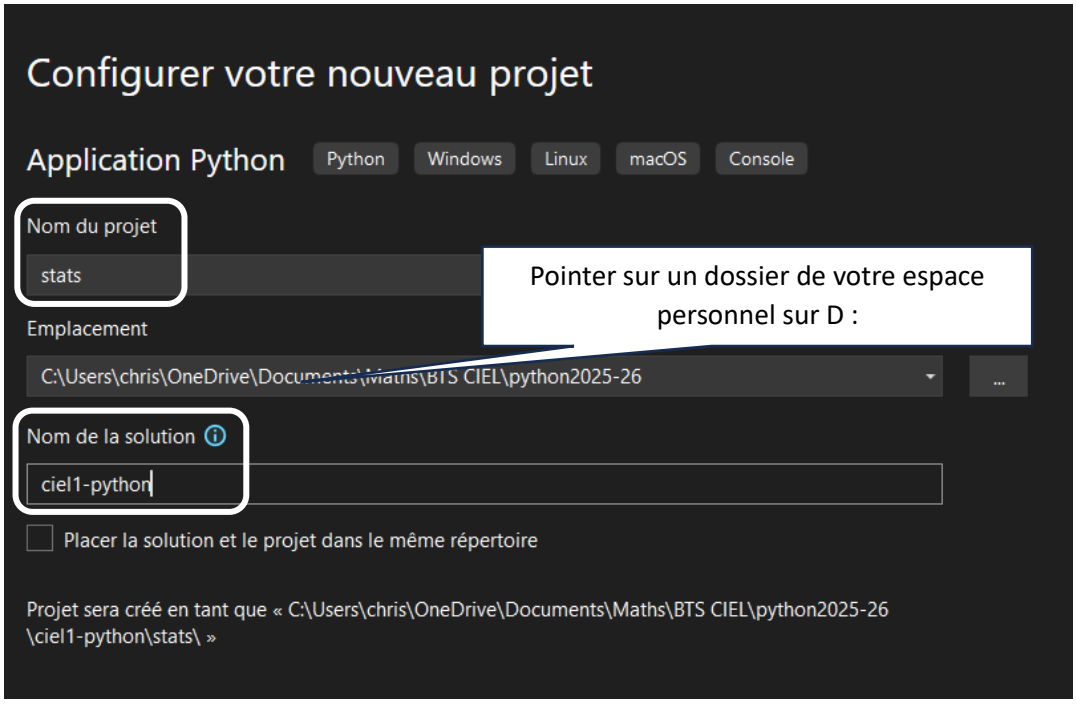
1- MISE EN PLACE DE L'ENVIRONNEMENT :

⇨ Ouvrir Visual Studio

⇨ Choisir l'option « Créer un projet »

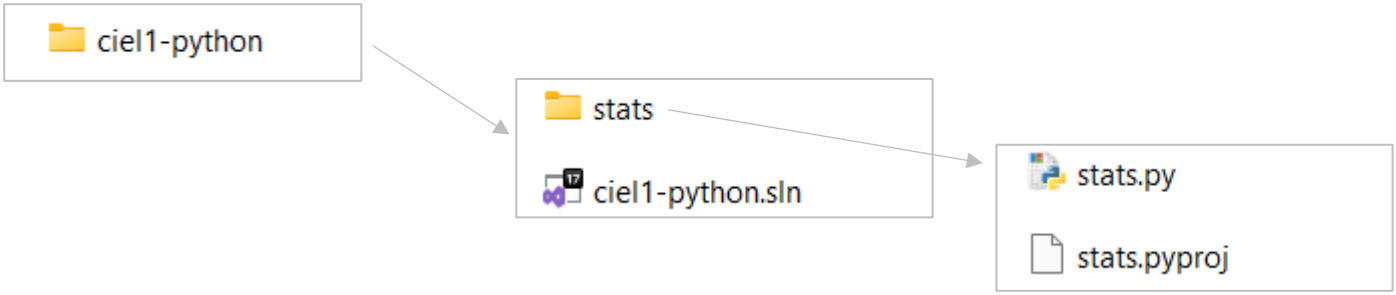
⇨ Choisir l'option :
« Application Python »





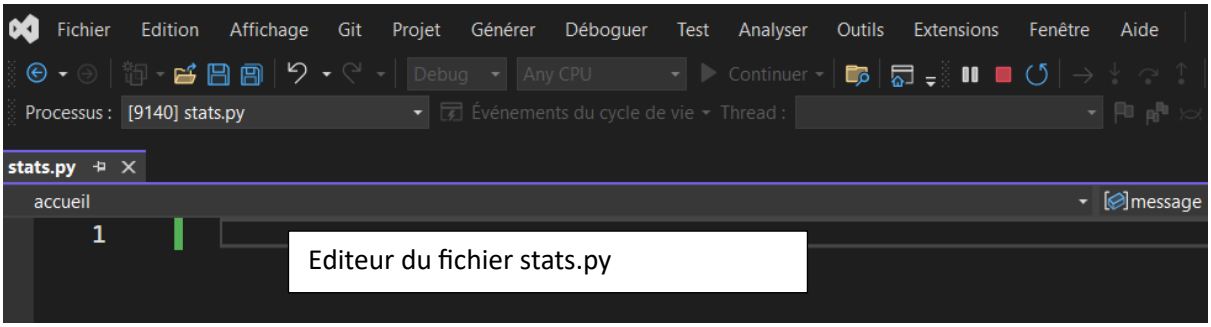
⇨ Dans la fenêtre « Configurer Projet », compléter les champs *Nom du projet*, *Emplacement et Nom de la solution* comme indiqué ci-contre.

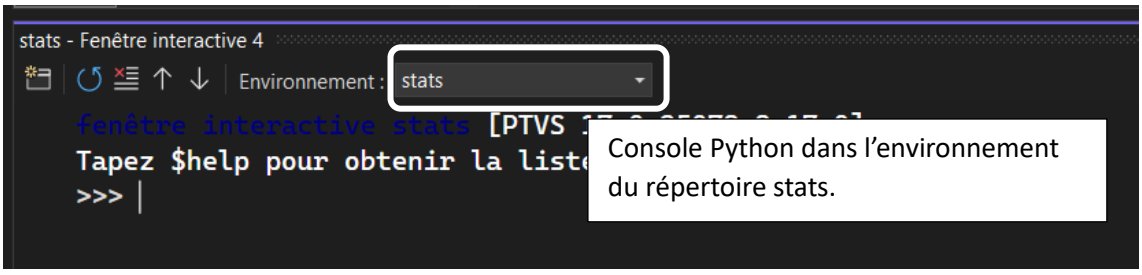
⇨ Suite à cette opération, le dossier repéré dans votre espace personnel, comprend à présent les répertoires suivants (vérifier) :



Sur Visual Studio :

- Dans la fenêtre supérieure, on retrouve l'éditeur du fichier *stats.py* qui a été créé.
- Dans la fenêtre inférieure, on retrouve :
  - le terminal après un « Ctrl ù »,
  - la console python après un « Alt i »



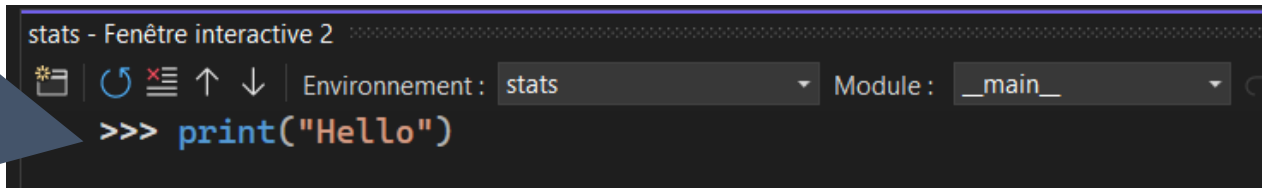


2- FONCTIONS PRINT() ET INPUT() :

On commence par travailler dans la console python afin de découvrir par l'exemple, les concepts de base.

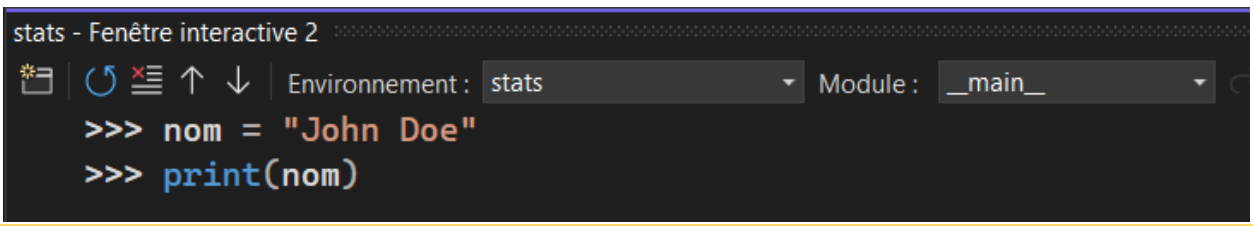
⇨ Exécuter dans la console :

Les 3 chevrons >>> préfixent toutes les lignes de la console



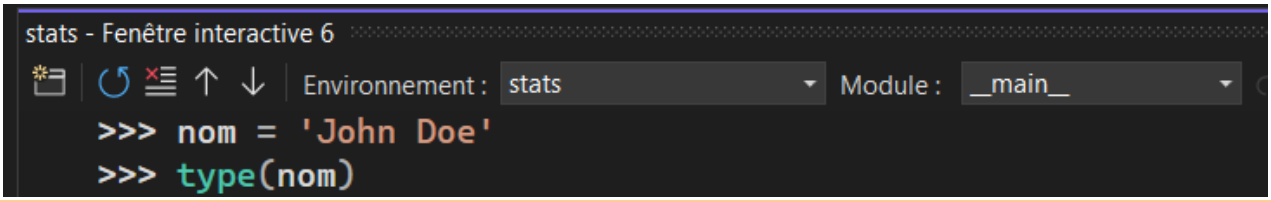
print() *est une* ***fonction*** *native de python qui permet d'afficher la valeur mise en* ***argument****. Ici cet argument est une chaine de caractère aussi appelé* ***string****.*

⇨ Exécuter dans la console :



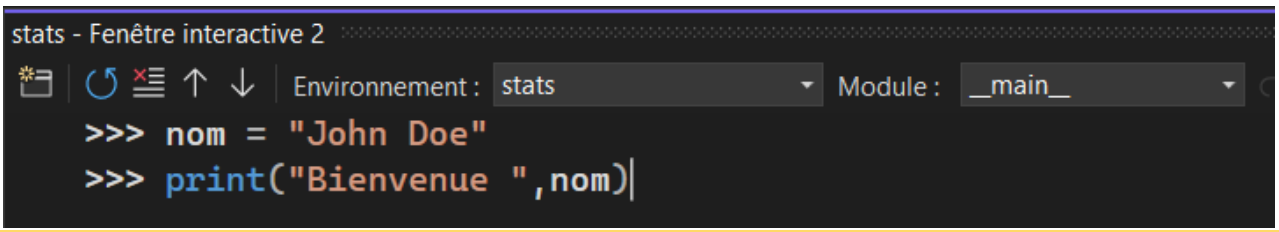
Une ***variable*** *nom* ***est créée*** *avec comme valeur initiale 'John Doe'. Cette variable est mise en argument de la fonction print().*

⇨ Exécuter dans la console :



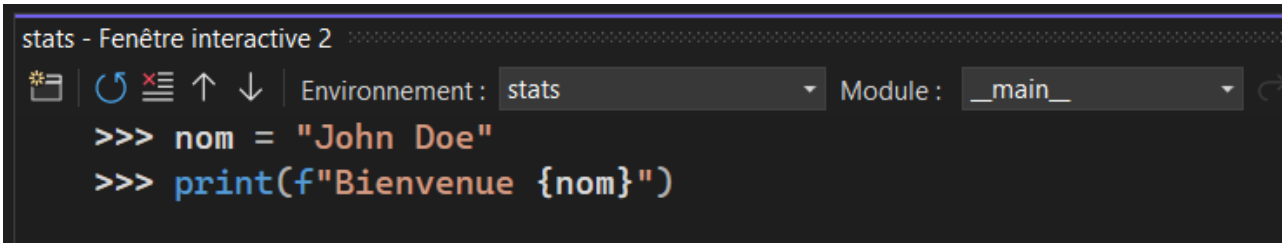
type() *est une fonction native de python qui retourne le type de la variable mise en argument.*

⇨ Exécuter dans la console :



La fonction ***print()*** *accepte plusieurs arguments. Le séparateur pour les séparer est la virgule.*

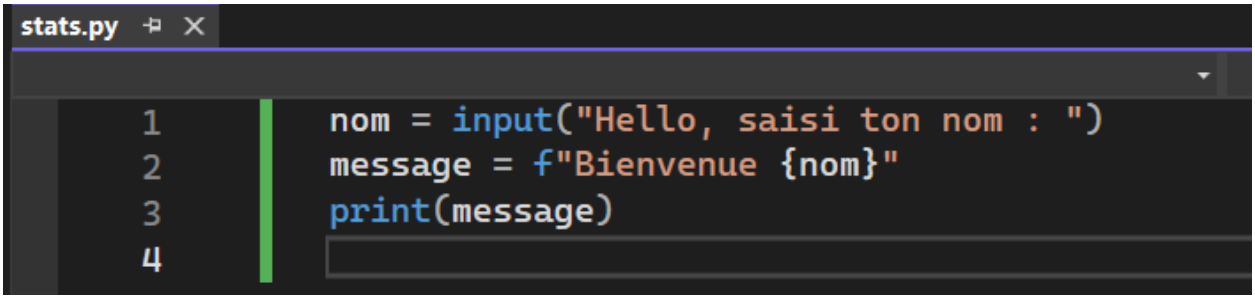
⇨ Exécuter dans la console :

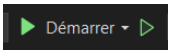


En ***préfixant*** *un string avec la* ***lettre f****, il est possible d'y insérer des variables. La valeur de la variable sera affichée à condition que le nom de la variable soit encadré par des* ***accolades****.*

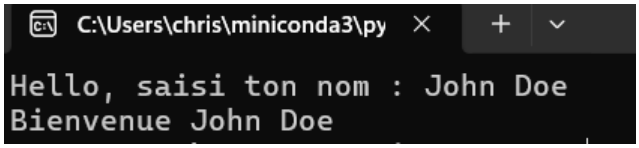
On continue en écrivant à présent les lignes pythons dans l'éditeur du fichier *stats.py* :

⇨ Ecrire dans le fichier stats.py :



⇨ Cliquer sur 

. Une fenêtre d'exécution s'ouvre :



Ici les 3 lignes du fichier stats.py sont exécutées ***dans l'ordre*** *descendant,* ***l'une après l'autre****.*

Un des principes de base de l'informatique est de compartimenter son code. Cela permet de pouvoir concevoir et tester séparément les différents modules qui le composent. C'est une condition indispensable pour pouvoir travailler en équipe afin de pouvoir créer et maintenir une application.

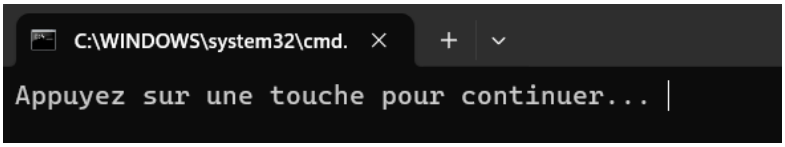
On peut compartimenter son code, en cloisonnant dans une **fonction**, les lignes pythons qui concernent une action précise.

Dans l'exemple ci-dessous, les 3 lignes de code déjà écrites, sont à présent compartimentées dans la fonction que l'on nomme ici *accueil()*.

⇨ Ecrire les lignes suivantes dans le fichier *stats.py* :

⇨ Cliquer sur 

. La fenêtre d'exécution s'ouvre, mais rien ne se passe …



*Ici les 4 lignes du fichier stats.py sont lues. La fonction est mémorisée, mais **elle n'est pas exécutée.***

On retourne dans la console python :

⇨ Dans la console, avec l'environnement défini comme étant l'espace de travail *Stats*, exécuter la ligne suivante afin de lire le fichier stats.py pour le mettre en mémoire :

*La fonction **accueil()** est mémorisée*

Pour preuve, en exécutant dans la console `>>> accueil`, la valeur de la variable accueil est retournée : `<function accueil at 0x0000022B4E4B0C10>`.

Adresse mémoire de la variable *accueil* dont la valeur est une fonction.

⇨ Appeler à présent cette fonction dans la console, afin d'exécuter les lignes de code qui la composent :

```
>>> accueil()
Hello, saisi ton nom :
John Doe

'Bienvenue John Doe'
>>>
```

La valeur retournée par cette fonction est affichée.

*La fonction **accueil()** est exécutée*

⇨ Appeler à présent la fonction dans la console et mémoriser la valeur retournée dans une nouvelle variable que l'on nomme ici *msg* :

```
>>> msg = accueil()
Hello, saisi ton nom :
John Doe

>>>
```

La valeur retournée par cette fonction est ici mémorisée dans une variable nommée *msg*

*La fonction **accueil()** est exécutée et la valeur retournée est mémorisée*

⇨ Afficher la valeur de la variable *msg* :

```
>>> msg
'Bienvenue John Doe'
```

Cette façon d'appeler les fonctions en les exécutant à partir de la console, est utile lorsqu'on souhaite les exécuter en les isolant du reste du code déjà écrit.
On intègre à présent l'appel de la fonction dans le code du fichier stats.py.

⇨ Compléter le fichier stats.py en y ajoutant la partie « programme principal » ci-dessous :

stats.py

accueil — message

```
1  def accueil() ->str :
2      nom = input("Hello, saisi ton nom : ")
3      message = f"Bienvenue {nom}"
4      return message
5
6  # programme principal
7  msg = accueil()
8  print(msg)
```

La ligne préfixée d'un # ne sera pas exécutée et sera considérée comme étant un commentaire.

Une variable nommée ici **msg** est créée. Elle aura comme valeur celle retournée par l'appel de la fonction accueil()

L'écriture de accueil() permet d'appeler la fonction et donc de l'exécuter. Si la fonction n'est pas appelée, elle sera seulement lue, mais pas exécutée.

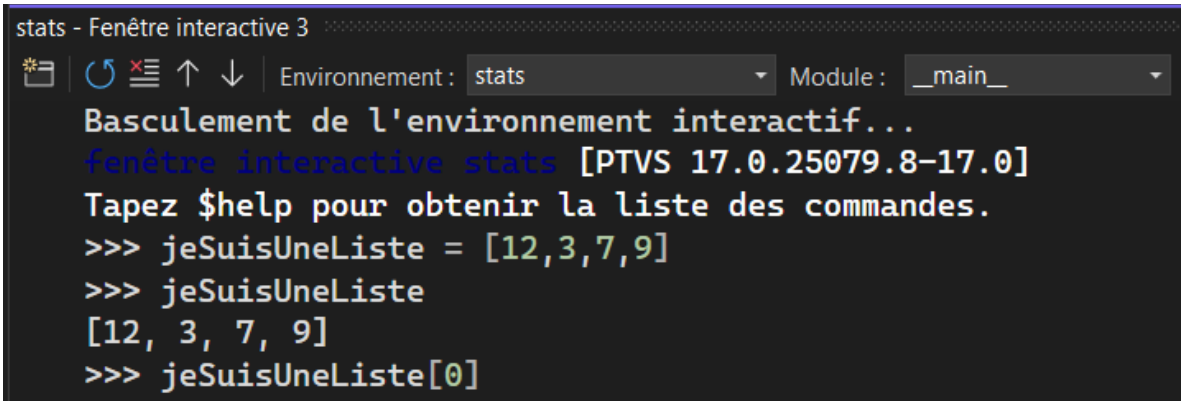
⇨ Cliquer sur 

. La fenêtre d'exécution s'ouvre, la fonction *accueil()* est lue et ensuite exécutée ….

3- LISTES PYTHONS ET BOUCLES WHILE :

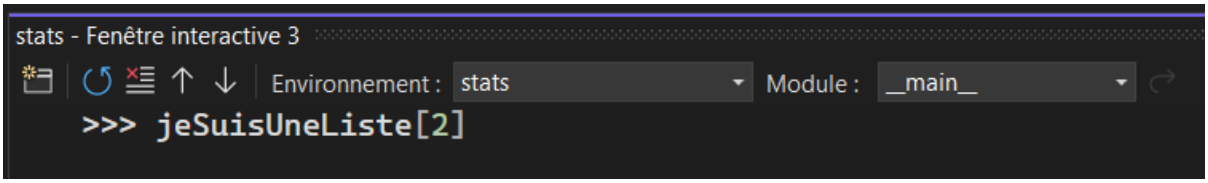
On se fixe à présent comme objectif de créer une autre fonction qui permette se saisir rapidement des valeurs numériques et de les stocker dans une liste. Comme dans la partie précédente, on commence par travailler dans la console python :

⇨ Exécuter dans la console :

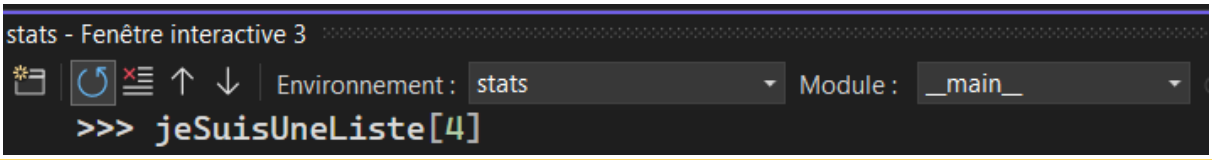


*Une variable nommée « jeSuisUneListe » est créée et sa valeur initiale est une **liste** contenant 4 nombres. Les listes en pythons correspondent aux tableaux dans les autres langages. Elles sont repérées par des **crochets**. Pour lire ou modifier les différents éléments de cette liste, on met l'index de l'élément entre crochet.*

⇨ Exécuter dans la console :

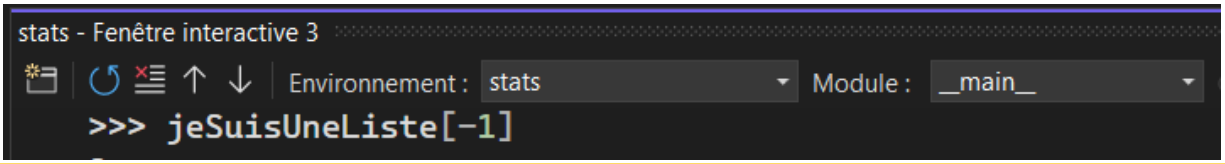


⇨ Exécuter dans la console :

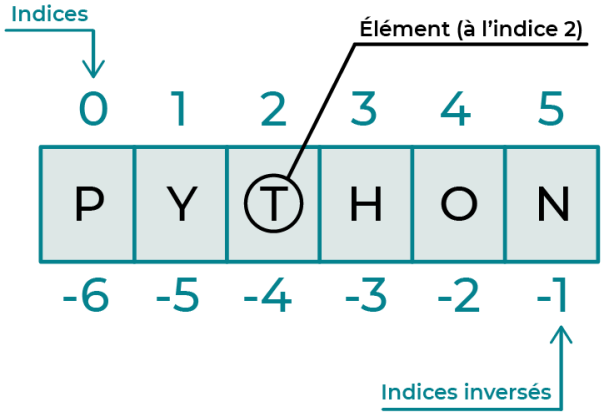


Si la liste contient 4 éléments, l'index maximal sera 3 …

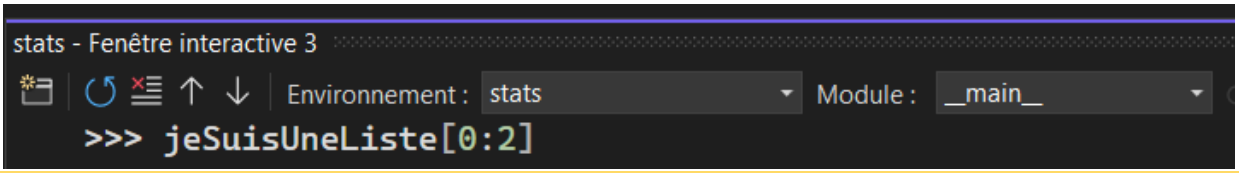
⇨ Exécuter dans la console :



Avec les index négatifs, on part de la fin …

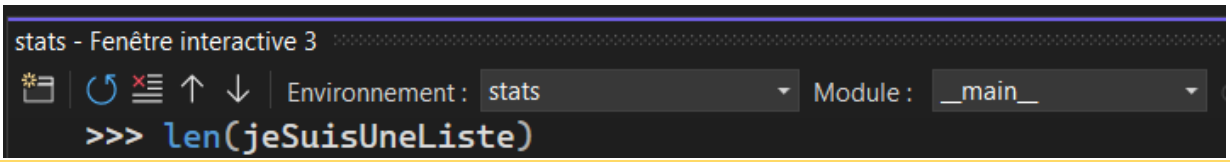


⇨ Exécuter dans la console :



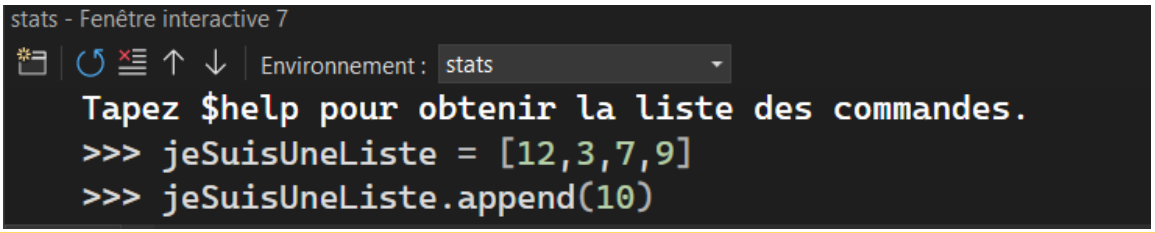
En utilisant dans les crochets des double-points : , on peut s'intéresser qu'à une tranche de cette liste (slice)

⇨ Exécuter dans la console :



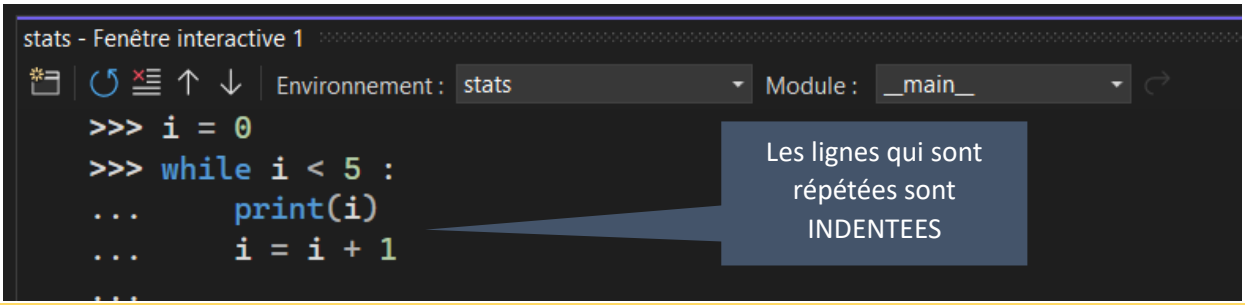
La fonction len() est native de python. Elle prend en argument une liste et retourne le nombre d'éléments qu'elle contient.

⇨ Exécuter dans la console :



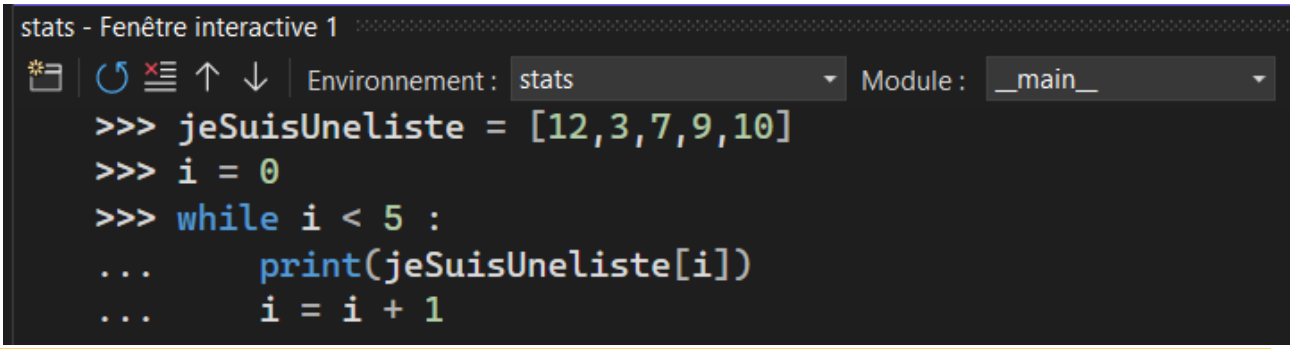
*La méthode **append()** est une méthode de la classe list. Elle prend en argument une valeur qui sera ajoutée en fin de liste.*

⇨ Exécuter dans la console :



*La structure **while** permet de répéter ici 2 lignes tant que la valeur de la variable i sera strictement inférieure à 5.*

⇨ Exécuter dans la console :



*La structure **while** permet aussi ici, de répéter 2 lignes tant que la valeur de la variable i sera strictement inférieure à 5.*

4- CODE A COMPLETER :

On donne ci-dessous le code incomplet de la fonction *saisieNotes()*.

```
1   > def accueil() ->str :...
5
6   v def saisieNotes() ->list :
7         mesNotes = []
8         i = 1
9         note = input(f"Entre la note {i} (ou rien si fin) : ")
10  v      while note != "" :
11            note = int(note)
12            mesNotes.append(note)
13
14            Partie à compléter
15
16
17
18        # programme principal
19        msg = accueil()
20        print(msg)
21        notes = saisieNotes()
22        print(f"{msg}, tu as saisi comme notes {notes}")
```

En cliquant sur >, on réduit l'affichage de la fonction

⇨ Compléter le code du fichier stats.py afin qu'à l'exécution, ▶ Démarrer ▷ on puisse par exemple obtenir le résultat suivant :

```
Hello, saisi ton nom : John Doe
Bienvenue John Doe
Entre la note 1 (ou rien si fin) : 14
Entre la note 2 (ou rien si fin) : 7
Entre la note 3 (ou rien si fin) : 9
Entre la note 4 (ou rien si fin) : 1
Entre la note 5 (ou rien si fin) : 20
Entre la note 6 (ou rien si fin) : 4
Entre la note 7 (ou rien si fin) : 6
Entre la note 8 (ou rien si fin) :
Bienvenue John Doe, tu as saisi comme notes [14, 7, 9, 1, 20, 4, 6]
Press any key to continue . . .
```

5- CALCUL DE LA MOYENNE DES NOTES SAISIES :

⇨ Ecrire le script de la fonction *calculMoyenne()* suivante dans le fichier *stats.py* :

```
 1      def accueil() ->str :...
 5
 6      def saisieNotes() ->list :...
16
17      def calculMoyenne(notes: list) -> float:
18          s = 0
19          for i in range(len(notes)) :
20              s = s + notes[i]
21          return s / len(notes)
```

Cette fonction a comme paramètre une liste nommée ici « *notes* » dans ce script.

Pour réaliser une boucle, on utilise ici non pas une structure while … mais une structure for ….

Cette fonction retourne une valeur de type *float*

⇨ On teste dans un premier temps le bouclage avec la structure for …. Dans la console, exécuter :

```
>>> for i in range(3) :
...     print(i)
...
```

L'affichage de i est exécuté 3 fois. La variable i prend les valeurs 0, 1 et 2

⇨ On continue à s'approprier ce type de bouclage. Dans la console, exécuter :

```
>>> a = [10,11,12]
>>> for i in range(len(a)):
...     print(f"l'index vaut {i} et la valeur associee est {a[i]}")
```

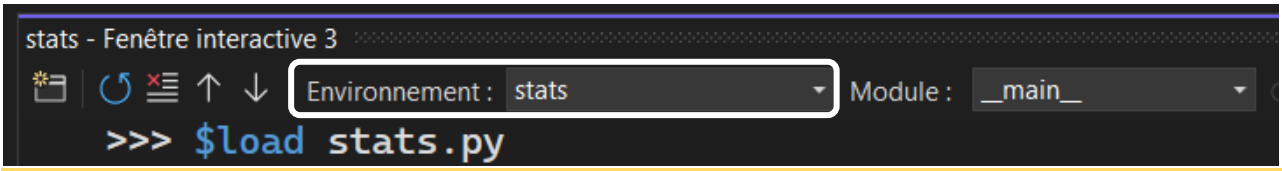
L'affichage est exécuté 3 fois. Ici len(a) = 3

⇨ On continue encore à s'approprier ce type de bouclage. Dans la console, exécuter :

```
>>> a = [10,11,12]
>>> for i in range(1,len(a)):
...     print(f"l'index vaut {i} et la valeur associee est {a[i]}")
```

L'affichage est exécuté à partir de i = 1 ici …

⇨ Tester ensuite la nouvelle fonction écrite dans la console python :



La fonction est mémorisée

Puis par exemple :

```
>>> calculMoyenne([0,5,10,15,20])
10.0
```

La fonction est exécutée avec comme liste mise en argument : [0,5,10,15,20]. La valeur moyenne est retournée.

Puis par exemple :

```
>>> moy = calculMoyenne([0,5,10,15,20])
>>> moy
10.0
```

La fonction est exécutée avec comme liste mise en argument : [0,5,10,15,20]. La valeur moyenne est retournée et mémorisée dans la variable nommée moy.

⇨ Complète la partie programme principal du fichier *stats.py* afin qu'à l'exécution on ait : ▶ Démarrer

```
Hello, saisi ton nom : John Doe
Bienvenue John Doe
Entre la note 1 (ou rien si fin) : 0
Entre la note 2 (ou rien si fin) : 5
Entre la note 3 (ou rien si fin) : 10
Entre la note 4 (ou rien si fin) : 15
Entre la note 5 (ou rien si fin) : 20
Entre la note 6 (ou rien si fin) :
La moyenne de ces notes est : 10.0
Appuyez sur une touche pour continuer...
```

6- CALCUL DE L'ECART-TYPE DES NOTES SAISIES :

⇨ Dans le fichier *stats.py,* compléter le script de la fonction *calculEcartType()* ci-dessous et le programme principal …

```
from math import sqrt

def accueil() ->str :...

def saisieNotes() ->list :...

def calculMoyenne(notes: list) -> float:...

def calculEcartType(notes: list , moyenne : float) ->float :
    s = 0
    Partie à compléter

# programme principal
msg = accueil()
print(msg)
notes = saisieNotes()
moy = calculMoyenne(notes)
sigma = calculEcartType(notes,moy)
print(f"La moyenne de ces notes est : {moy} et l'ecart-type est : {sigma}")
```

On importe en mémoire la fonction $\sqrt{\ }$ de la bibliothèque ***math*** de python.

La fonction nommée *calculEcartType()* est définie avec 2 paramètres.

Afin que l'on puisse retrouver l'exécution suivante : ▶ Démarrer

```
Hello, saisi ton nom : John Doe
Bienvenue John Doe
Entre la note 1 (ou rien si fin) : 0
Entre la note 2 (ou rien si fin) : 5
Entre la note 3 (ou rien si fin) : 10
Entre la note 4 (ou rien si fin) : 15
Entre la note 5 (ou rien si fin) : 20
Entre la note 6 (ou rien si fin) :
La moyenne de ces notes est : 10.0 et l'ecart-type est : 7.0710678118654755
Appuyez sur une touche pour continuer...
```

7- RENDU DU TRAVAIL :

⇨ Uploader votre fichier stats.py à partir de la page https://mathsapp-front.vercel.app/ à ouvrir dans un navigateur :