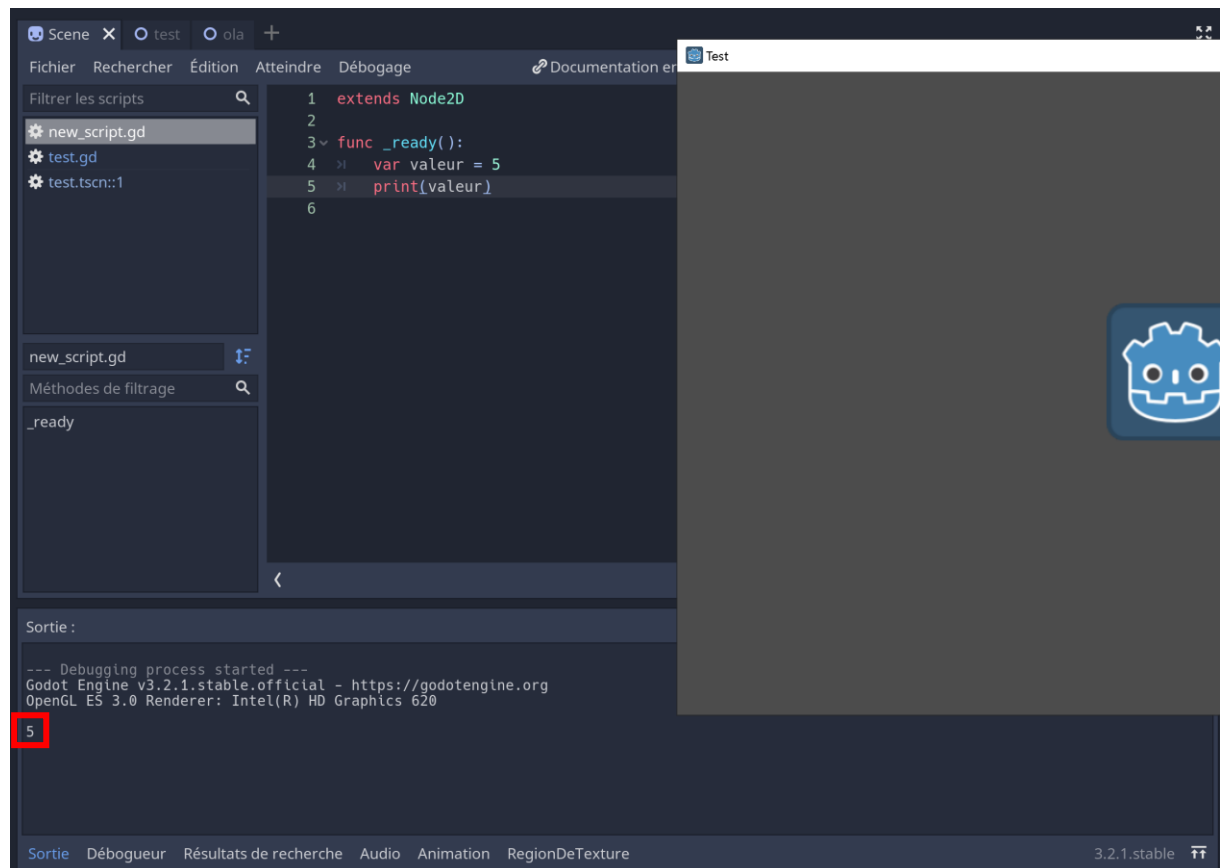


## Préambule

### Affichage d'une variable

Pour afficher une variable, il faut utiliser la fonction ***print*** donnée par Godot. L'image ci-dessous montre un exemple de script. Lorsque la scène est lancée avec la petite flèche en haut à droite (Play), le résultat de la fonction ***print*** s'affiche en bas dans la Sortie (la valeur est 5).



### Déclarer une variable accessible partout dans le script :

On peut déclarer des variables en dehors des fonctions. Si cette variable est modifiée dans une fonction, elle est modifiée pour toutes les autres fonctions qui l'utilisent.

Voici un exemple sur l'image ci-dessous :

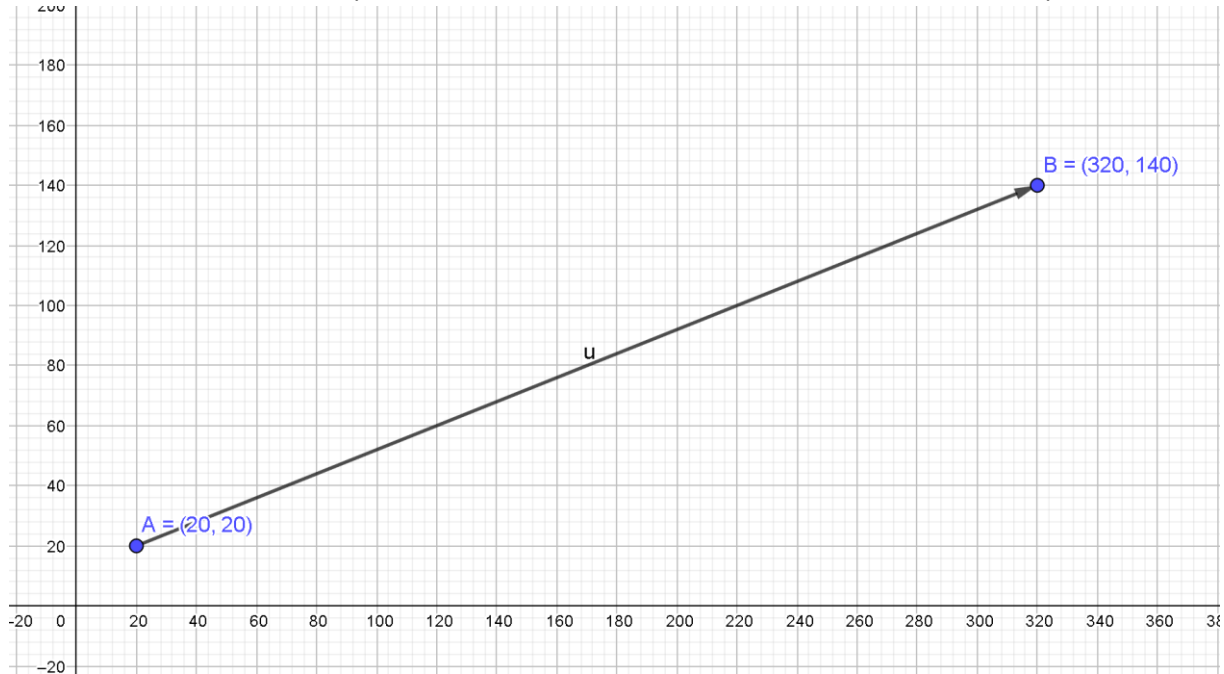
- Si j'appuie sur la touche flèche gauche : alors j'affiche la valeur de la variable ***deplacement*** qui est : `Vector2(100, 0)`
- Si j'appuie ensuite sur la touche flèche droite, ***deplacement*** est affectée de la valeur suivante `Vector2(0,0)`
- Enfin si je réappuie sur la flèche gauche, alors j'affiche la nouvelle valeur de la variable ***deplacement*** qui est : `Vector2(0,0)`

```
3 var deplacement = Vector2(100, 0)
4
5 func right():
6     deplacement = Vector2(0,0)
7
8 func left():
9     print(deplacement)
```






## Exercices

### Exercice 1

Dans la fonction `_ready()`, créer une variable pour le point A et le point B. Leur affecter les valeurs qui sont sur le graphique ci-dessous. Ensuite trouver le déplacement  $\vec{u}$  entre le point A et le point B en utilisant le calcul adapté. Afficher le résultat comme sur l'exemple suivant :



### Exercice 2

Lorsque j'appuie sur , l'objet doit se déplacer une seule fois d'une valeur 30 pixels vers la droite. Ensuite l'objet ne doit plus bouger même si l'on appuie une nouvelle fois sur . Faire l'opération pour chacune des autres touches   .

Pour rappel l'unité de mesure dans Godot 2D est le nombre de pixels.

### Exercice Bonus

Introduction d'une nouvelle instruction, la condition **if**. Elle permet de faire des opérations spécifiques dans le cas où une condition est respectée. Nous en parlerons plus en détail la prochaine fois, mais voici un court exemple d'utilisation (image page suivante).

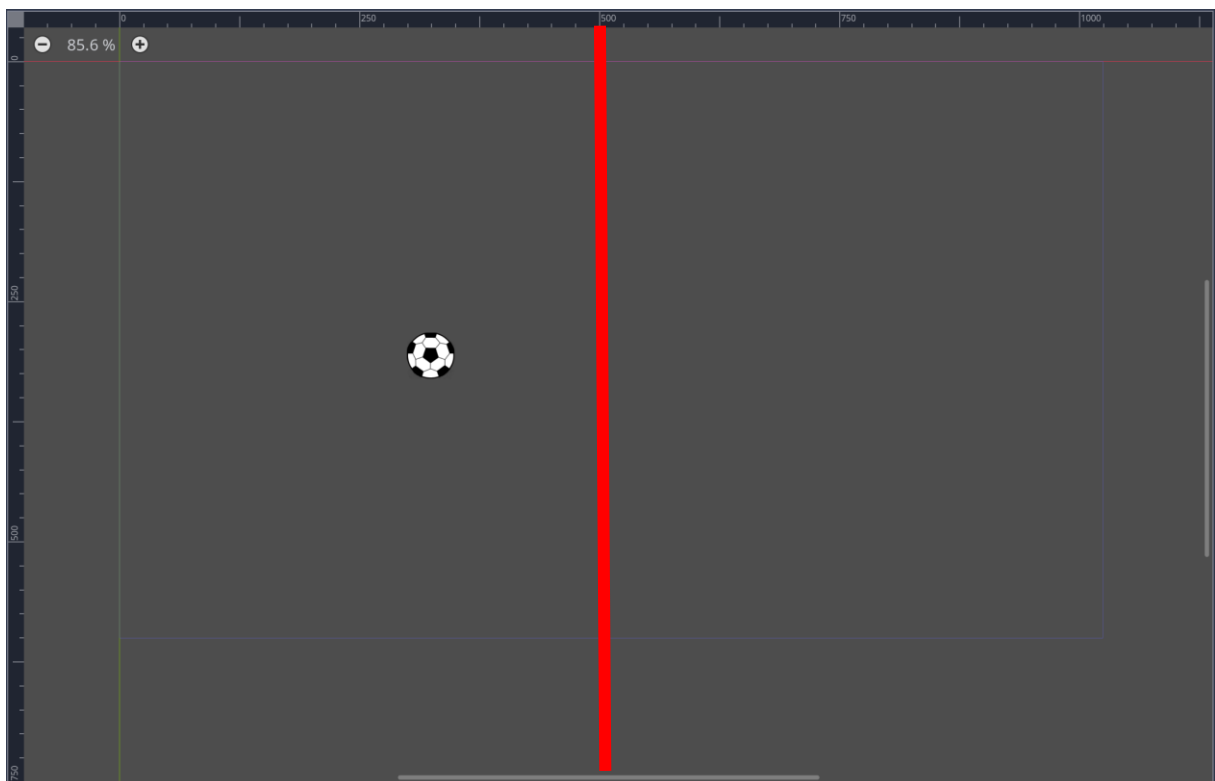
```

3 ▾ func _ready():
4   >| var nombre = 5
5   >|
6   >| # si nombre est bien égale à la valeur 5 alors
7   >| # alors j'ajoute 1 à nombre
8   >| # sinon j'enlève 1 à nombre
9 ▾ >| if nombre == 5:
10  >| >| nombre += 1
11 ▾ >| else:
12  >| >| nombre -= 1
13  >| >|
14  >| var deplacement = Vector2(10, 20)
15  >|
16  >| # si deplacement est bien égale à la valeur Vector2(10,20) alors
17  >| # alors j'affiche un message
18  >| # sinon je continue le script sans rien afficher
19 ▾ >| if deplacement == Vector2(10, 20):
20  >| >| print("il est bien égal à Vector2(10, 20)")

```

Pour l'exercice, faire la même chose que l'on a fait en cours, c'est-à-dire déplacer la balle d'un pixel en fonction de la direction. En plus de ça, dans la fonction **\_ready**, affecter à la position de la balle la position (300, 300).

Le cœur de l'exercice est d'empêcher la balle de passer la ligne verticale à  $x = 500$ . C'est-à-dire que l'on ne peut plus aller à droite lorsque la coordonnée  $x$  de la position de la balle est égale à 500. Il faut pour cela appliquer les bonnes conditions dans la fonction **go\_right**.



Bon courage 😊