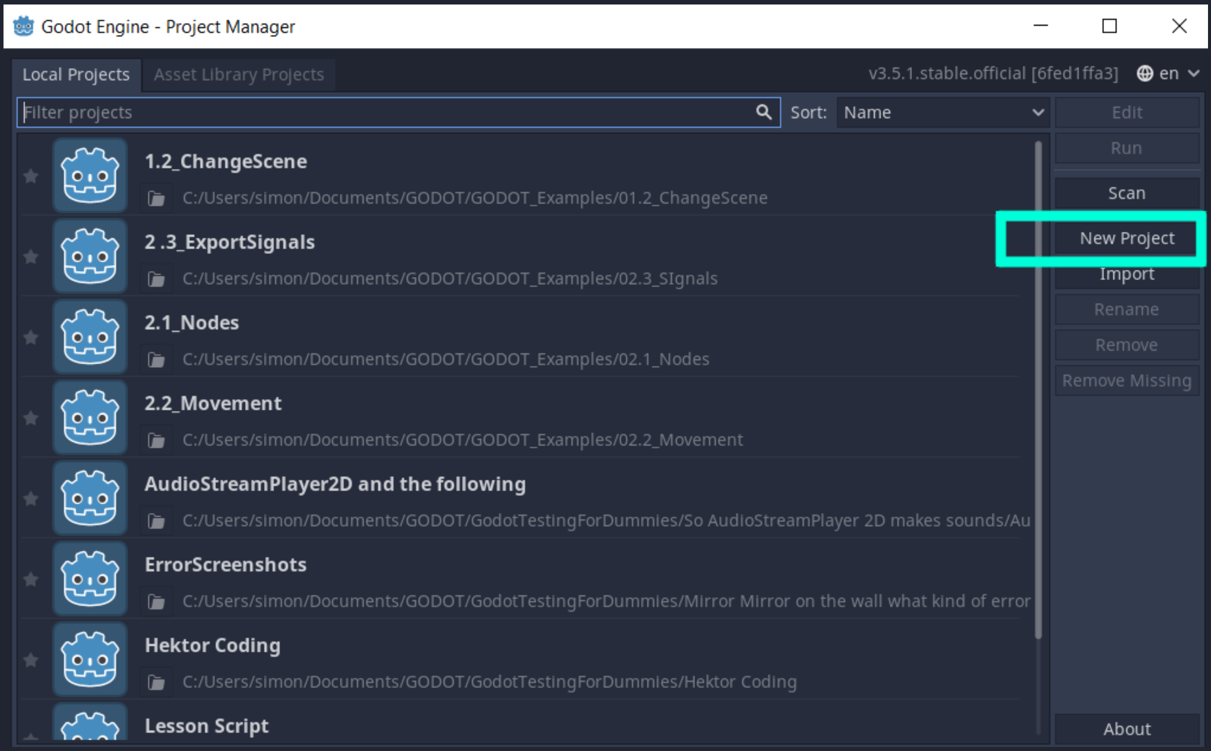
Coding in Godot: Example tasks and Setup in the Godot engine

Setup your coding environment:

1. Create a new Godot Project



Godot Engine - Project Manager

Local Projects | Asset Library Projects | v3.5.1.stable.official [6fed1ffa3] | en

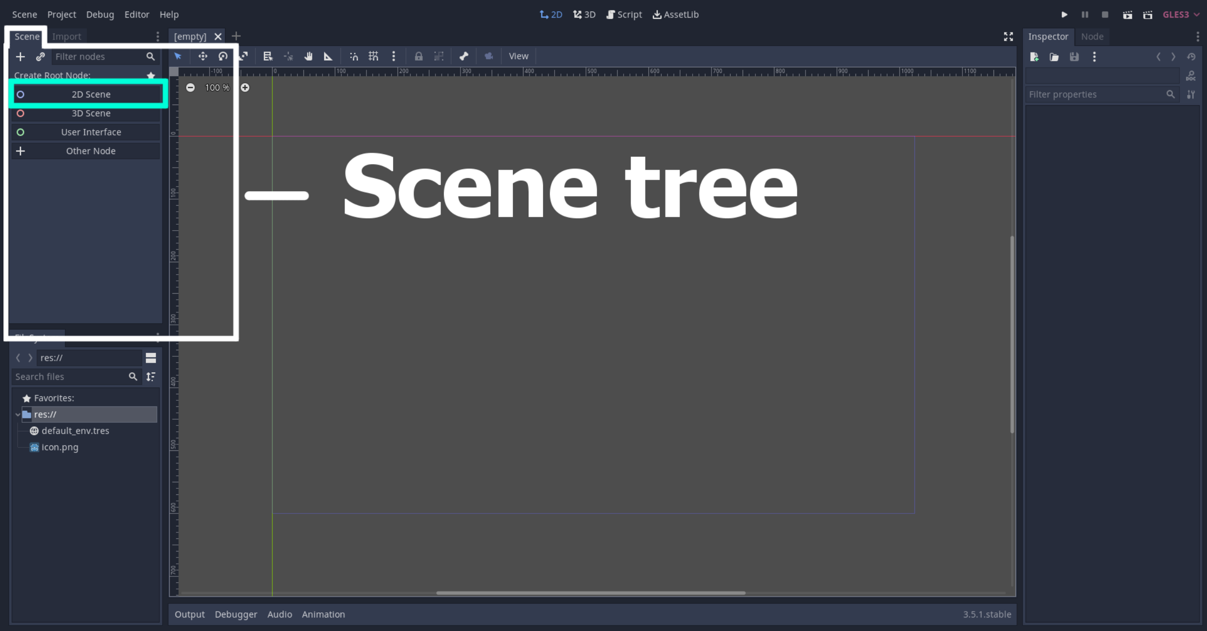
Filter projects | Sort: Name

- 1.2_ChangeScene — C:/Users/simon/Documents/GODOT/GODOT_Examples/01.2_ChangeScene
- 2.3_ExportSignals — C:/Users/simon/Documents/GODOT/GODOT_Examples/02.3_SIgnals
- 2.1_Nodes — C:/Users/simon/Documents/GODOT/GODOT_Examples/02.1_Nodes
- 2.2_Movement — C:/Users/simon/Documents/GODOT/GODOT_Examples/02.2_Movement
- AudioStreamPlayer2D and the following — C:/Users/simon/Documents/GODOT/GodotTestingForDummies/So AudioStreamPlayer 2D makes sounds/Au
- ErrorScreenshots — C:/Users/simon/Documents/GODOT/GodotTestingForDummies/Mirror Mirror on the wall what kind of error
- Hektor Coding — C:/Users/simon/Documents/GODOT/GodotTestingForDummies/Hektor Coding
- Lesson Script

Edit | Run | Scan | New Project | Import | Rename | Remove | Remove Missing | About

2. Give it a recognizable filename and save it in a folder of your choice. It's best to keep your files sorted!

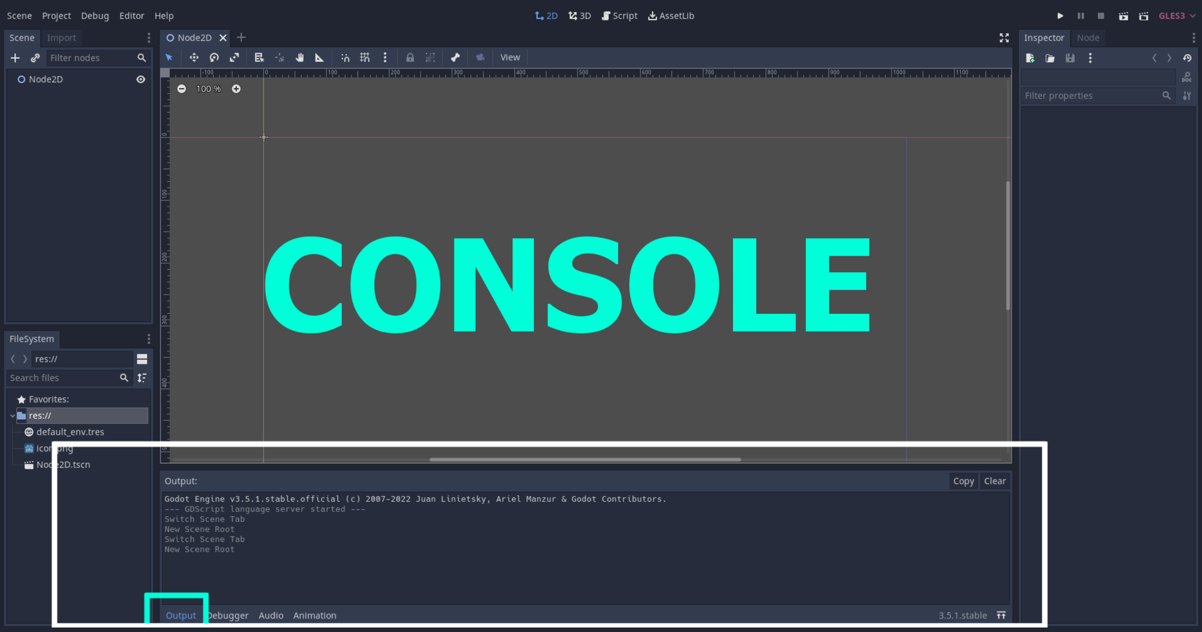
3. Create a new scene by clicking the 2D Scene button on the **SCENE TREE**. If you need to make a new scene, do so by clicking 'Scene > New scene' in the top menu.



The Scene tree will show you which NODES you have in your scene. You can choose from many different NODES, but for our purposes, we will use 2D scenes.
(Godot documentation: Using the Scene tree)
https://docs.godotengine.org/en/stable/tutorials/scripting/scene_tree.html
(Godot documentation: Scene tree as a class)
https://docs.godotengine.org/en/stable/classes/class_scenetree.html

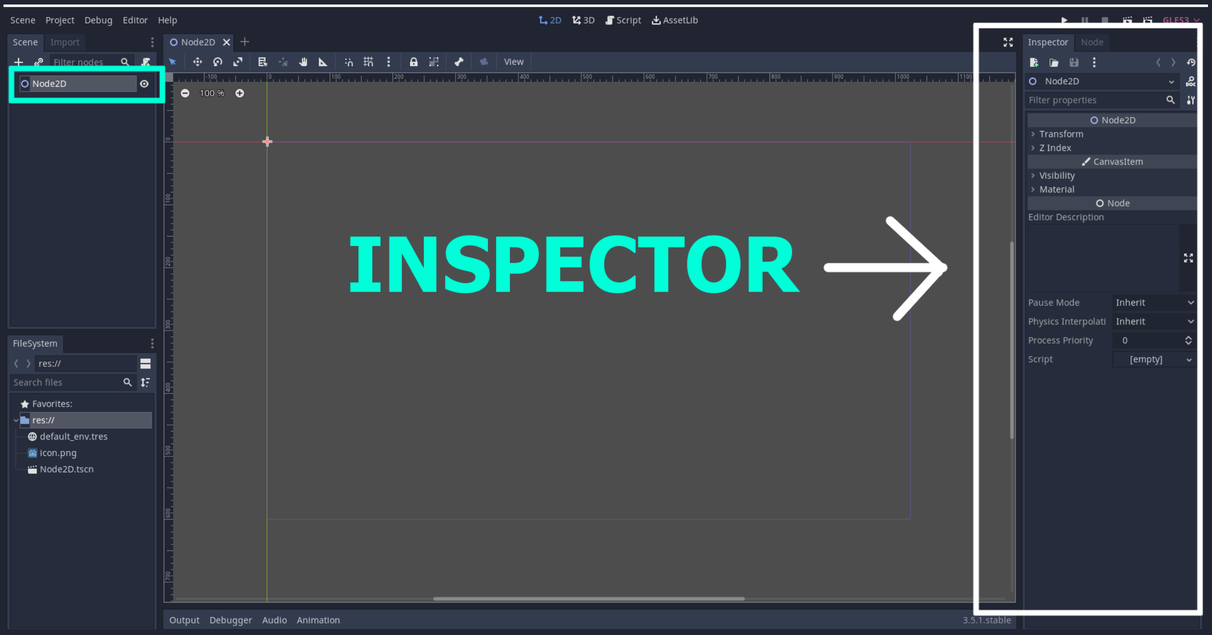
4. **!!!Save your scene!!!**
   You can save your scene by clicking
   Scene > Save scene > Save
   Or hitting CTRL+S on your keyboard (STRG+S on german keyboards).
   Make sure to save your Scene and your Scripts regularly!

5. The **CONSOLE** is where you will see the output of your code. You can open and close it by clicking 'Output' on the bottom of the screen.

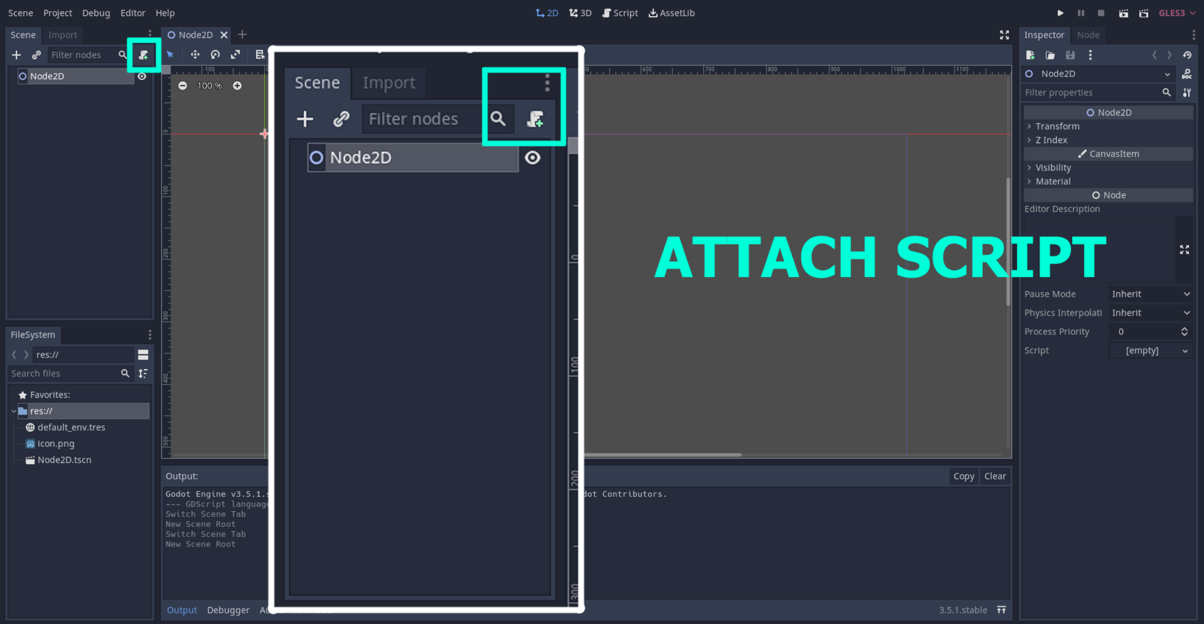


```
Godot Engine v3.5.1.stable.official (c) 2007-2022 Juan Linietsky, Ariel Manzur & Godot Contributors.
--- GDScript language server started ---
Switch Scene Tab
New Scene Root
Switch Scene Tab
New Scene Root
```

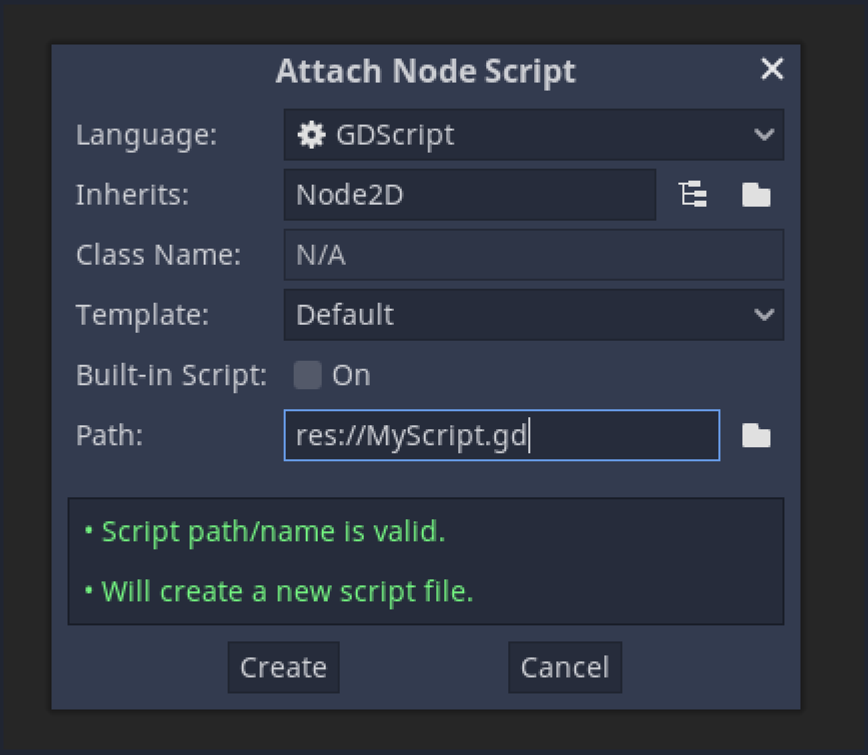
6. Click on the **root node** of your scene; The root node is always the first/highest node in the scene tree. It is the so-called **parent node** to all nodes you will add later.
   Your root node in this example is the Node2D in the scene tree.
   Click it, you should see the information in the **INSPECTOR (right hand side of the screen)** change to show the properties of the node.



7. Now attach the script. Your code/script needs to be attached to a Node to be executed once you run the program.
Do that by hitting the ATTACH SCRIPT button above the scene tree.



Rename the script and click 'Ceate'.
I named the example script MyScript.gd - Make sure it ends with .gd


Attach Node Script

Language: GDScript

Inherits: Node2D

Class Name: N/A

Template: Default

Built-in Script: On

Path: res://MyScript.gd

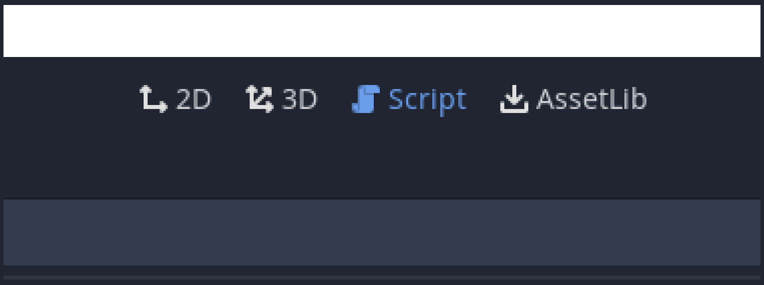
• Script path/name is valid.

• Will create a new script file.

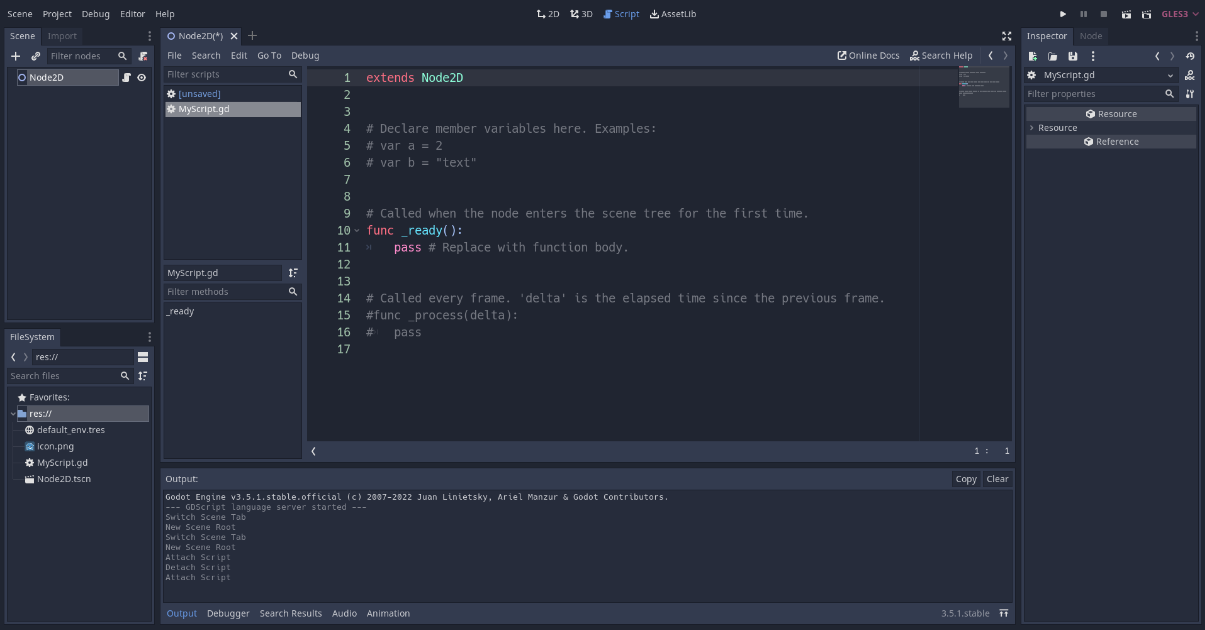
Create   Cancel

Your file names should reflect its purpose.

Now, GODOT will open the SCRIPTING ENVIRONMENT.
You can manually switch to it by clicking ‘Script’ in the top of the screen, or switch back to your 2D scene by clicking ‘2D’.



Done! Setup for coding complete. It should look like this:



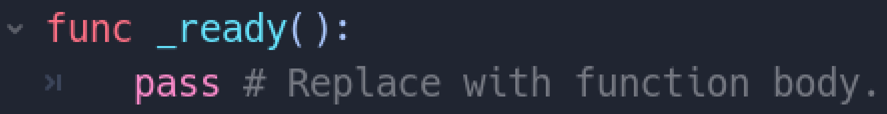
```
extends Node2D


# Declare member variables here. Examples:
# var a = 2
# var b = "text"


# Called when the node enters the scene tree for the first time.
func _ready():
    pass # Replace with function body.


# Called every frame. 'delta' is the elapsed time since the previous frame.
#func _process(delta):
#    pass

```

```
Godot Engine v3.5.1.stable.official (c) 2007-2022 Juan Linietsky, Ariel Manzur & Godot Contributors.
--- GDScript language server started ---
Switch Scene Tab
New Scene Root
Switch Scene Tab
New Scene Root
Attach Script
Detach Script
Attach Script
```

Make sure to save regularly!

Getting output onto the console

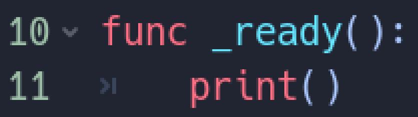
GODOT already puts a function into your code the first time you launch it.
This is the _ready(): function. It gets called once when the scene is loading.



```
func _ready():
	pass # Replace with function body.
```

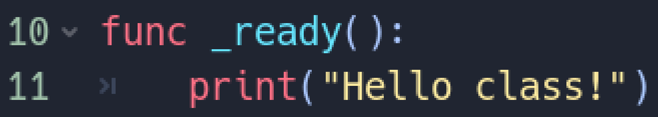
You have to put your code into the _ready(): function for this example.

1. Delete pass and everything behind it.
   Pass will let the computer ignore any code that comes after it in a function.
2. Write print() instead.

   ```
   func _ready():
   	print()
   ```

   Make sure there is a tab before the print to indicate it is inside the _ready function.
   The command: print() will let you output text, variables and other things onto the CONSOLE.
3. As long as print() has nothing written between the brackets, there won't be any output. To print a text, write your text inside the brackets surrounded by " quotation marks ".

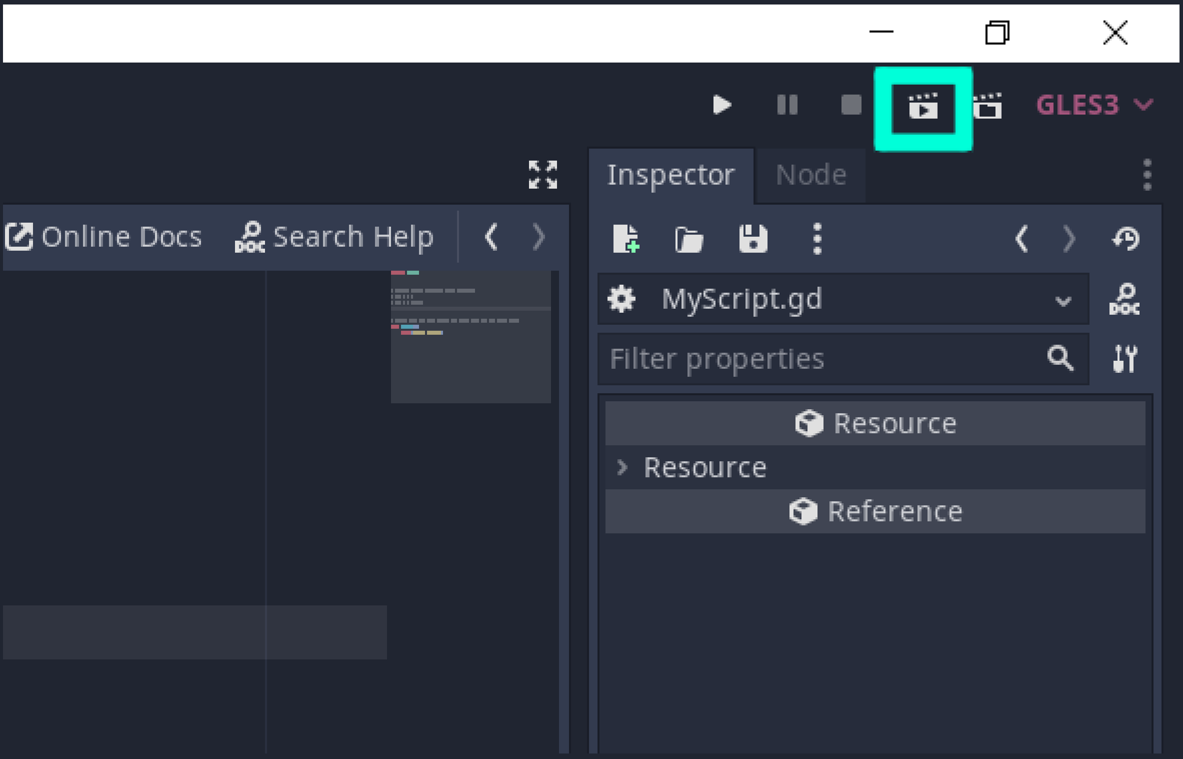
   ```
   func _ready():
   	print("Hello class!")
   ```

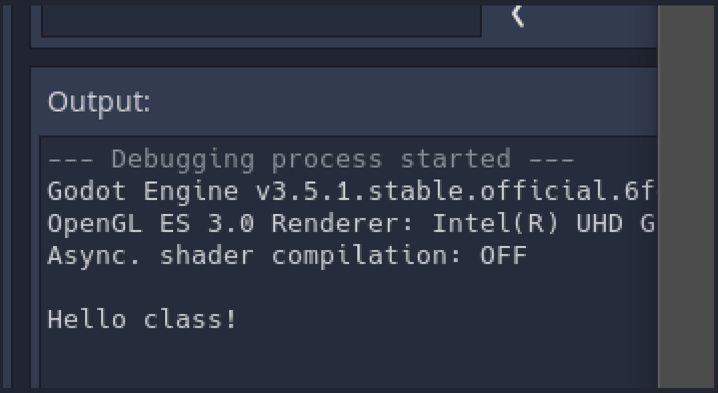
   This will indicate a string like in other programming languages.

   (Godot documentation: GD script basics; Variable types etc.)
   https://docs.godotengine.org/en/stable/tutorials/scripting/gdscript/gdscript_basics.html

4. Click the Filmreel icon in the upper right corner to run the scene. Alternatively, you can hit F6 on the keyboard.
   GODOT opens a **game window** - and you'll see your text printed on the CONSOLE output.

   Close the **game window** again before continuing coding.



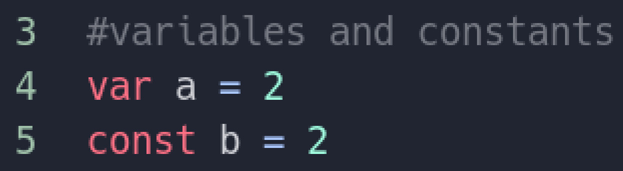

Output:

```
--- Debugging process started ---
Godot Engine v3.5.1.stable.official.6f
OpenGL ES 3.0 Renderer: Intel(R) UHD G
Async. shader compilation: OFF

Hello class!
```

Coding basics and examples

There are two data types to keep in mind - variables and constants.
Variables get changed during runtime, constants do not.
You will mostly use variables.

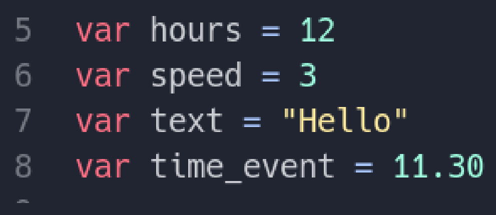
You declare variables and constants like this:



```
#variables and constants
var a = 2
const b = 2
```

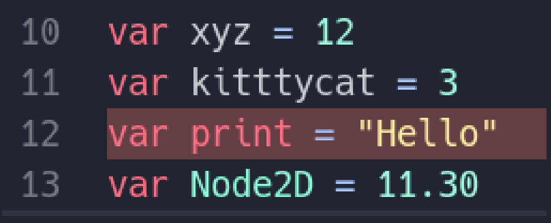
var indicates that the following is a new variable that hasn't been declared yet.
a is the name of the variable. All variables need names; They are storage containers. Variable names need to make sense, cannot contain empty spaces or special symbols (§$%&/), and cannot be named like inbuilt types (f.ex. You can't name a variable var, print or Node2D). See Lesson Slides or read the documentation for the available variable types.

Good variable names:



```
var hours = 12
var speed = 3
var text = "Hello"
var time_event = 11.30
```

Bad/not working variable names:

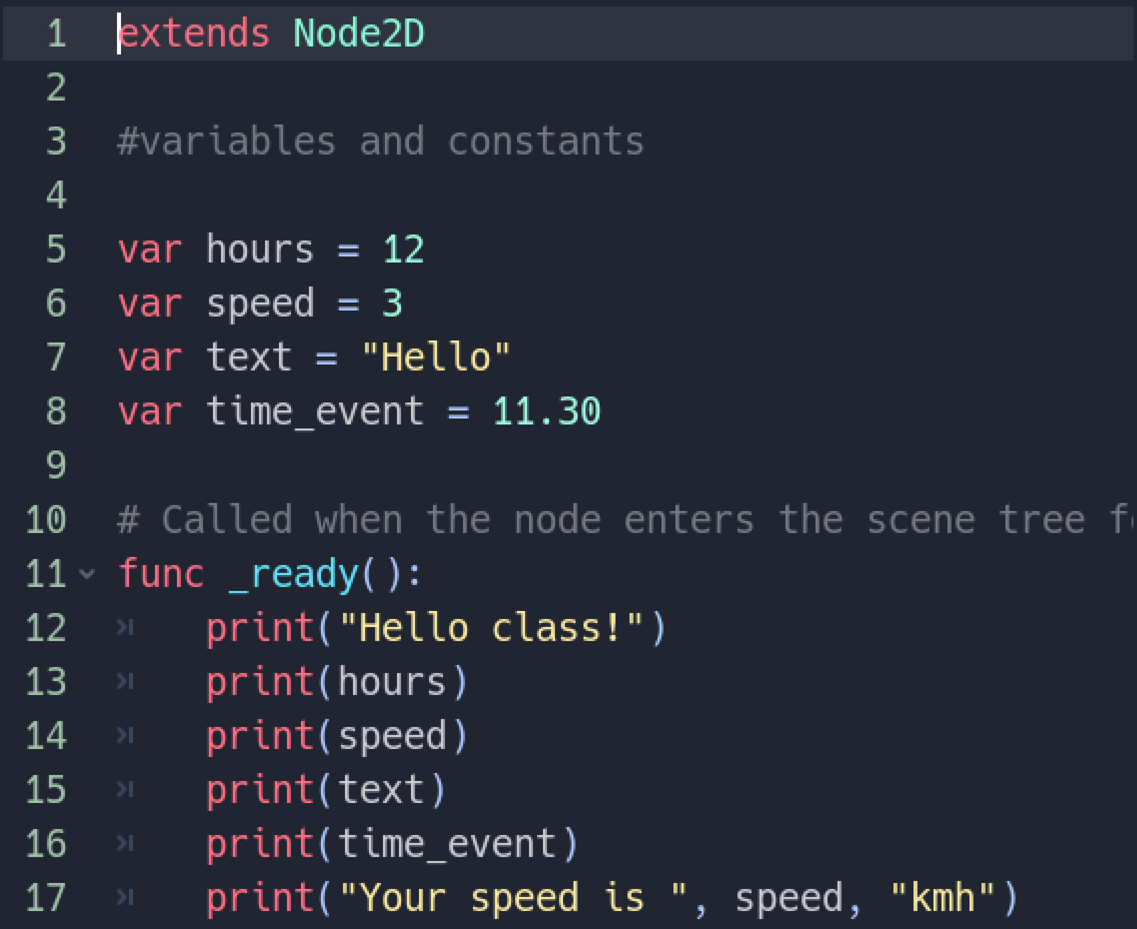


```
var xyz = 12
var kitttycat = 3
var print = "Hello"
var Node2D = 11.30
```

= is an operator. It will assign the value on its right side to the variable name on the left.

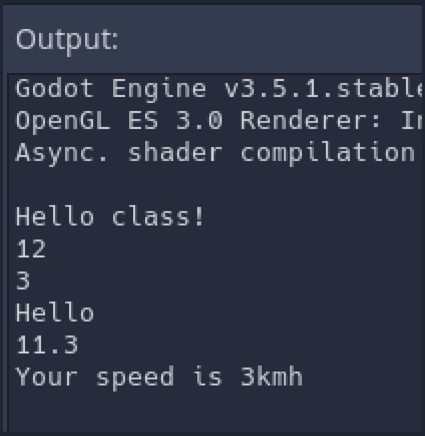
You can print variables by writing the variable name into the print brackets (See example below). This is often used for debugging/finding mistakes in the code.

If you want to print multiple variables in the same line, separate them with a comma.



```
extends Node2D

#variables and constants

var hours = 12
var speed = 3
var text = "Hello"
var time_event = 11.30

# Called when the node enters the scene tree f
func _ready():
    print("Hello class!")
    print(hours)
    print(speed)
    print(text)
    print(time_event)
    print("Your speed is ", speed, "kmh")
```

Will output:



Output:

```
Godot Engine v3.5.1.stable
OpenGL ES 3.0 Renderer: I
Async. shader compilation

Hello class!
12
3
Hello
11.3
Your speed is 3kmh
```