LESSON 2:
CONDITIONAL LOGIC

REVIEW

int

Whole numbers

var x = 7

float

Any Number

var x = 3.4

bool

True or False

var x = true

String

Text

var x = “hello”

enum

Defined set of things

var x = States.Idle

Vector2

Point in space

var x = Vector2(1.3, 2.4)

print()

Prints a thing: print(“Hello, this is a message!”)

Can separate multiple things with commas: print(“My number is: “, myInt)

CONDITIONAL LOGIC

Actually do things with your variables.

CONDITIONAL LOGIC?

Conditional logic is the simplest form of system building in a programming language.

Put simply, it allows you to check the value of your variables and only execute lines of code if that check comes up true.

There are two structures in GDScript that enable conditional logic:

1. If/Else Statement
2. Match Statement

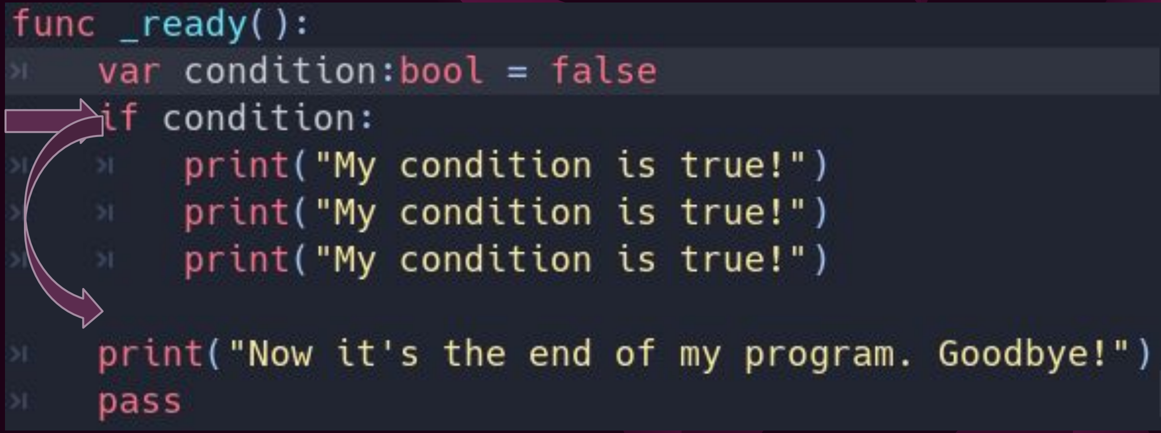
If/Else is used much MUCH more frequently than Match, but Match is cool too so we'll learn it.

IF STATEMENT

Reads a bool.

If the bool is true, jumps into the indentation of the if statement and executes that code.

If the bool is false, jumps over the indentation block of the if statement, skipping over that code.

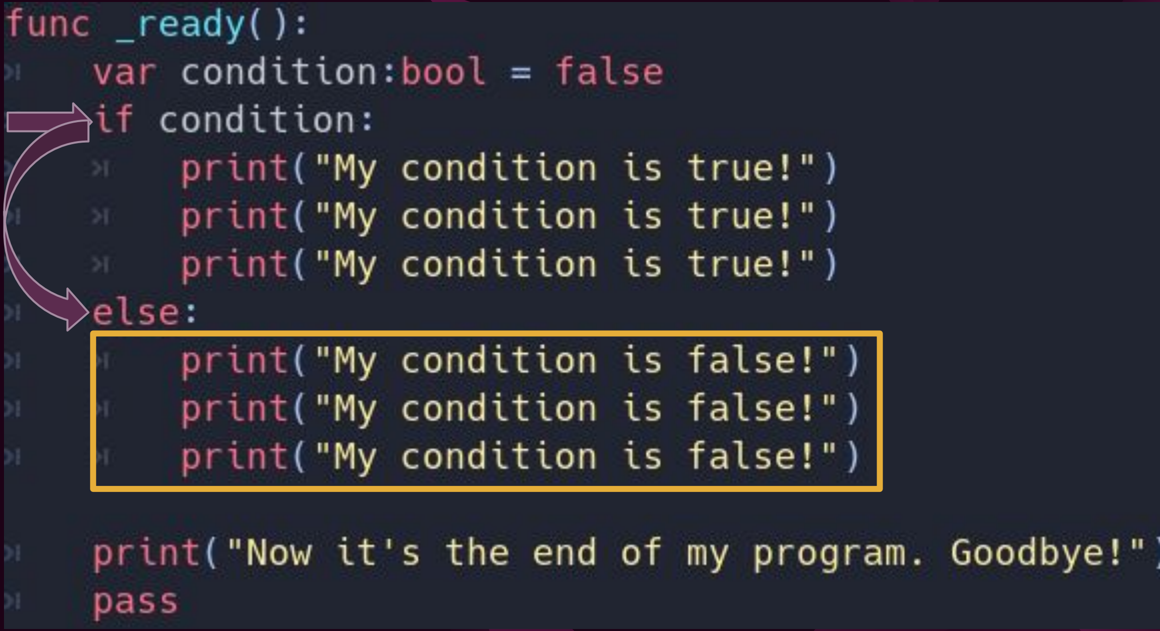


```
func _ready():
	var condition:bool = false
	if condition:
		print("My condition is true!")
		print("My condition is true!")
		print("My condition is true!")

	print("Now it's the end of my program. Goodbye!")
	pass
```

ELSE STATEMENT

Must be placed directly after an if statement.
When the if statement gets skipped, the else statement runs instead.

But, if the if statement is successfully executed, the else statement is skipped.



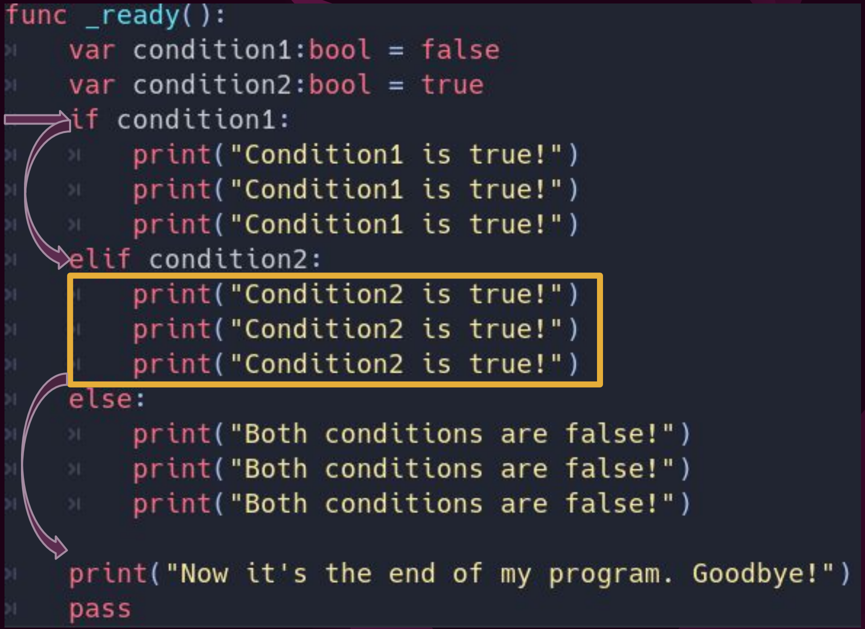
```
func _ready():
	var condition:bool = false
	if condition:
		print("My condition is true!")
		print("My condition is true!")
		print("My condition is true!")
	else:
		print("My condition is false!")
		print("My condition is false!")
		print("My condition is false!")

	print("Now it's the end of my program. Goodbye!"
	pass
```

ELIF STATEMENT

ELIF stands for “Else If”; It combines the functionality of the “else” statement with the “if statement”

It will only execute when the if statement above it fails (else) BUT it also lets you check another condition (if)

Hense… else if. Elif.

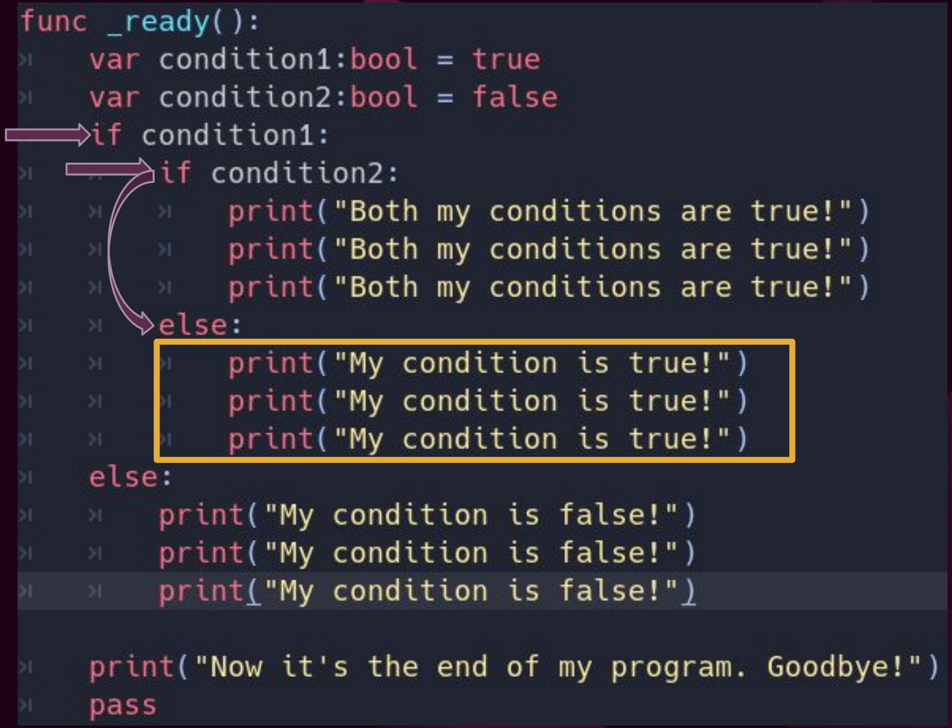


```
func _ready():
    var condition1:bool = false
    var condition2:bool = true
    if condition1:
        print("Condition1 is true!")
        print("Condition1 is true!")
        print("Condition1 is true!")
    elif condition2:
        print("Condition2 is true!")
        print("Condition2 is true!")
        print("Condition2 is true!")
    else:
        print("Both conditions are false!")
        print("Both conditions are false!")
        print("Both conditions are false!")

    print("Now it's the end of my program. Goodbye!")
    pass
```

NESTING IF STATEMENTS

You can write whatever code you want inside of an if statement… including another if statement.

This can get complicated very quickly.



```
func _ready():
	var condition1:bool = true
	var condition2:bool = false
	if condition1:
		if condition2:
			print("Both my conditions are true!")
			print("Both my conditions are true!")
			print("Both my conditions are true!")
		else:
			print("My condition is true!")
			print("My condition is true!")
			print("My condition is true!")
	else:
		print("My condition is false!")
		print("My condition is false!")
		print("My condition is false!")

	print("Now it's the end of my program. Goodbye!")
	pass
```

SO WHAT'S THE BIG DEAL?

- If I wanted to just run some code, couldn't I just write the code I want to run instead of making a bool variable and setting it to true?

Yes! But…

- Other systems may use bools to indicate state. For example, animationPlayer.playing is a bool that's true if an animation is playing, false if not
- We're about to learn a bunch of cool stuff on the next slide that expands our world of possibility.

Boolean Operators

Allow us to convert each of our other variable types into booleans. All boolean operators take this form:

data OP data

E.g…

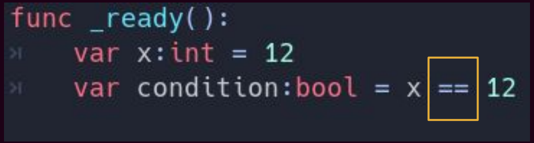
```
myVar == 7
myVar1 > myVar2
```

BOOLEAN OPERATORS

Equality

==

Returns true if left and right side data are equal; false otherwise.

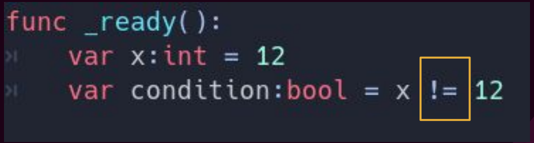


```
func _ready():
    var x:int = 12
    var condition:bool = x == 12
```

Inequality

!=

Returns true if left and right side are not equal; false otherwise.

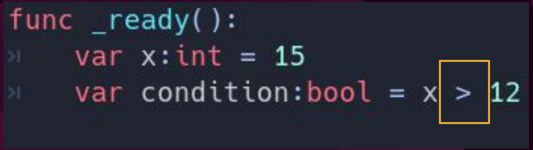


```
func _ready():
    var x:int = 12
    var condition:bool = x != 12
```

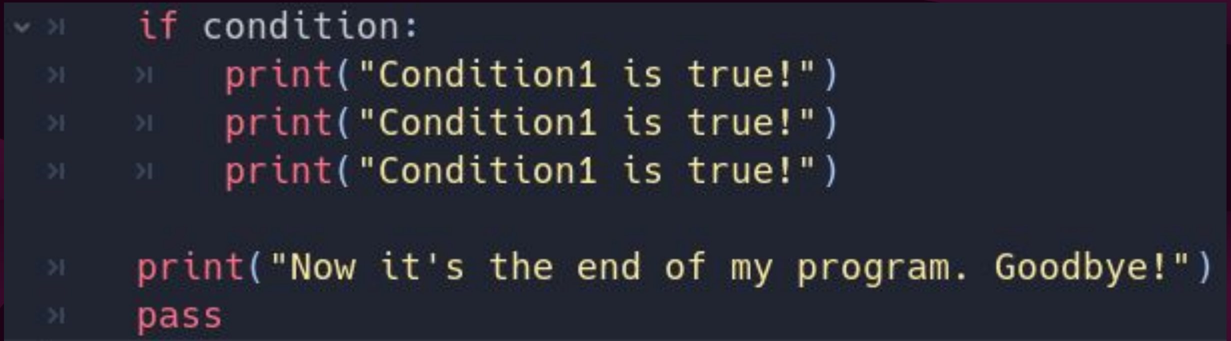
Greater Than

>

Returns true if left side is greater than right side, false otherwise.



```
func _ready():
    var x:int = 15
    var condition:bool = x > 12
```



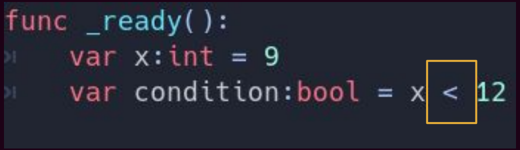
```
    if condition:
        print("Condition1 is true!")
        print("Condition1 is true!")
        print("Condition1 is true!")

    print("Now it's the end of my program. Goodbye!")
    pass
```

BOOLEAN OPERATORS

Less Than

<

Returns true if left side is less than right side; false otherwise.

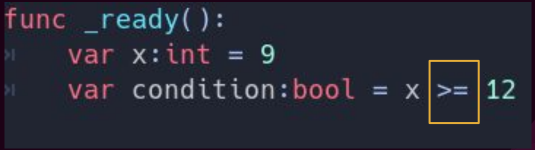


```
func _ready():
    var x:int = 9
    var condition:bool = x < 12
```

GT or Equal To

>=

Returns true if left side is greater than or equal to right side; false otherwise.

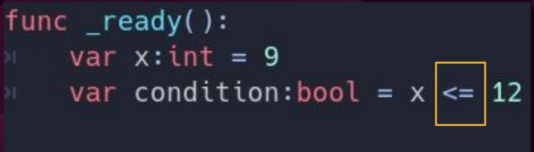


```
func _ready():
    var x:int = 9
    var condition:bool = x >= 12
```

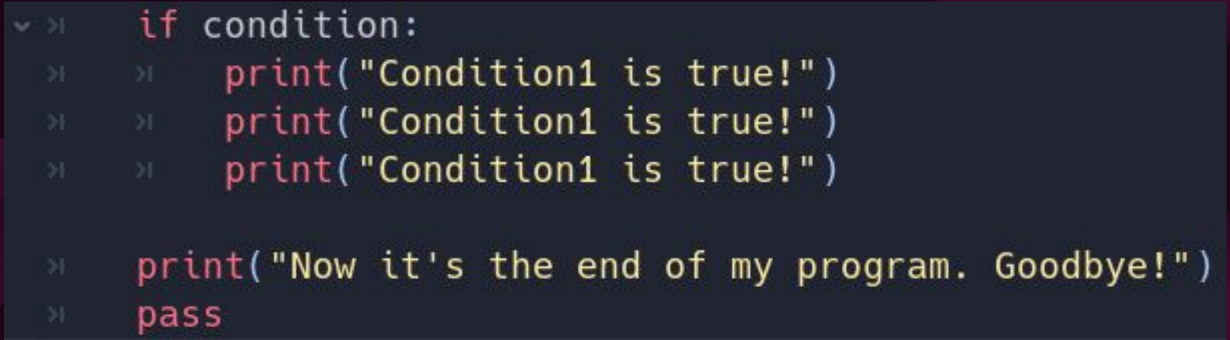
LT or Equal To

<=

Returns true if left side is less than or equal to right side; false otherwise.



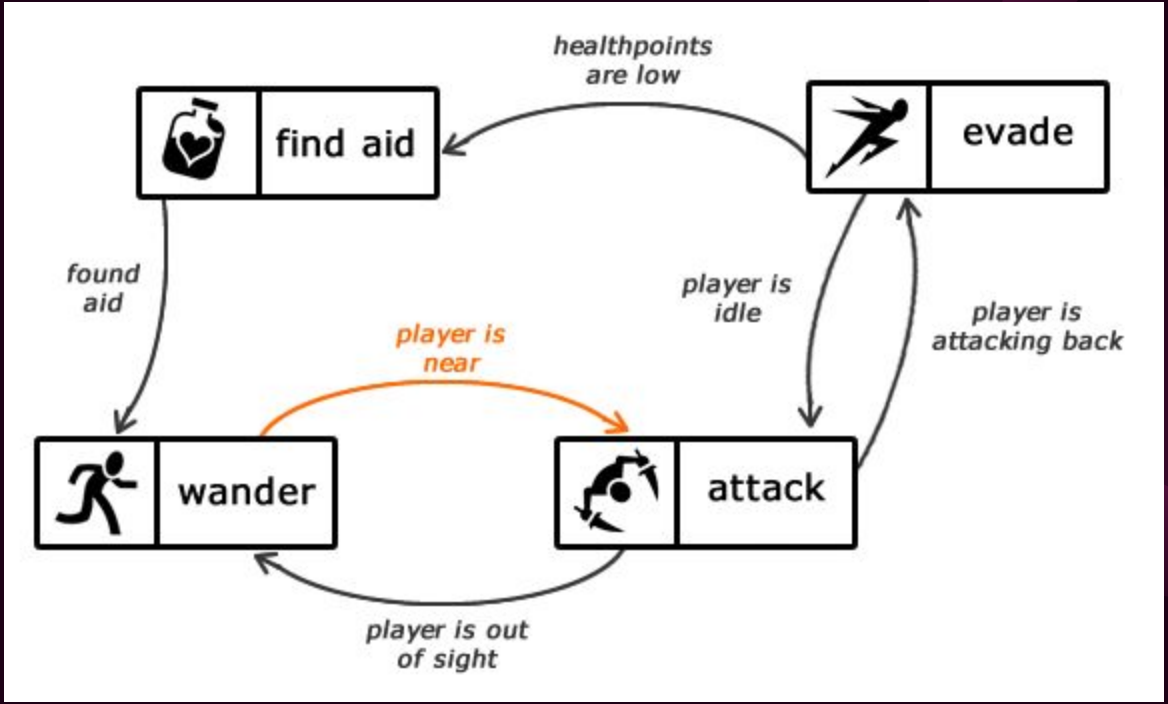
```
func _ready():
    var x:int = 9
    var condition:bool = x <= 12
```



```
    if condition:
        print("Condition1 is true!")
        print("Condition1 is true!")
        print("Condition1 is true!")

    print("Now it's the end of my program. Goodbye!")
    pass
```

Hp < maxHP*.25

inventory.HasHpPot



playerState == States.Idle
playerState == States.Attack

Near: playerPos <= sightRange
Far: playerPos > sightRange

CONDITIONAL OPERATORS

Allows us to combine multiple bools into a single bool. Increases the power of a single if statement.

E.g …

True and False

True or False

CONDITIONAL OPERATORS

AND

and / &&

Returns true if both left AND right side are both true.

OR

or / ||

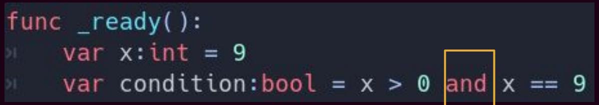
Returns true if either left OR right side are true.

NOT

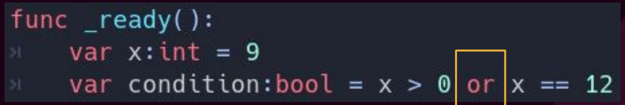
not / !

Only affects 1 bool.

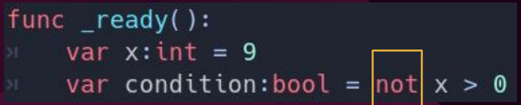
Flips the state of the bool; true -> false; false -> true.



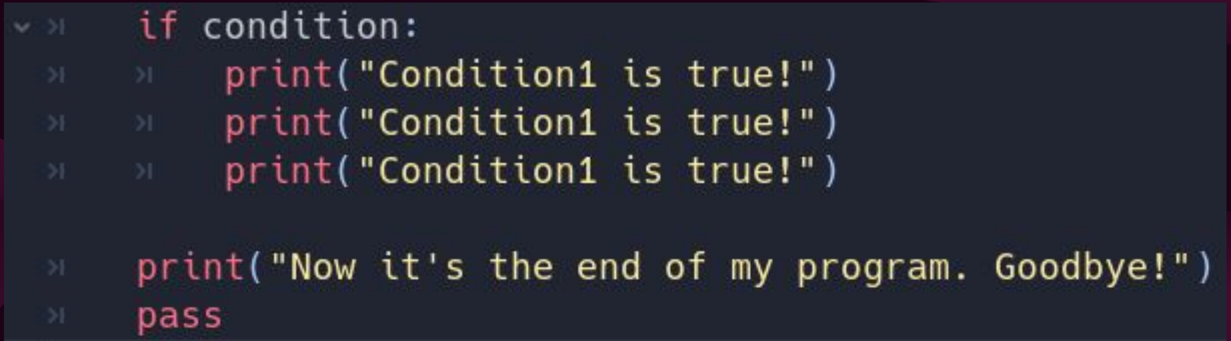
```
func _ready():
	var x:int = 9
	var condition:bool = x > 0 and x == 9
```



```
func _ready():
	var x:int = 9
	var condition:bool = x > 0 or x == 12
```



```
func _ready():
	var x:int = 9
	var condition:bool = not x > 0
```



```
	if condition:
		print("Condition1 is true!")
		print("Condition1 is true!")
		print("Condition1 is true!")

	print("Now it's the end of my program. Goodbye!")
	pass
```

PRACTICE PROBLEM

1. Attach a script to a node2D.
2. Create and export an int variable called “HP”
3. Uncomment the _process(): function in the script
4. Write an if statement that prints a warning message if HP drops below 25.
5. Write an else-if statement that prints a good message if HP is exactly 100.
6. Write an if statement that prints the message “You Are Dead” if your HP is less than OR equal to zero.