LESSON 1: DATA & VARIABLES

PRIMITIVE

- Simple to use and understand
- Dependable, always there for you
- Is the foundation for every system, no matter how complex.

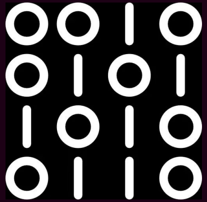
4

Restricts what you can store in this block to an accepted, interpretable range.

TYPE

programming language

Given to you by OS 32 or 64 bit blocks stored in RAM.



Raw Data

COMPLEX

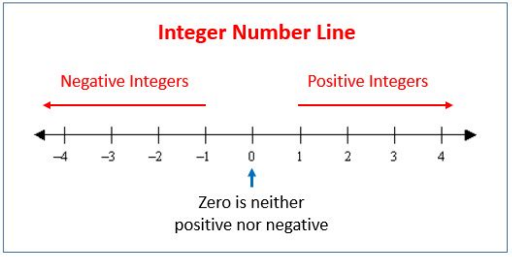
- Powerful ability to systemize ideas.
- Sparkly, shiny, alluring.
- Not very nice to newbies, comes at a cost.

2

DATA TYPES 1-3

int

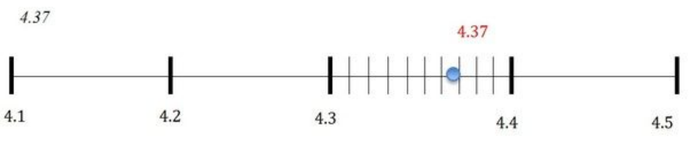
Whole numbers



float

Whole and decimal numbers

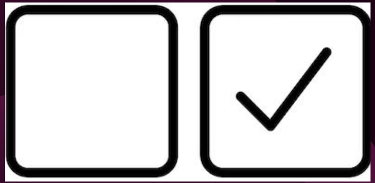
(but has drawbacks)



bool

true or false

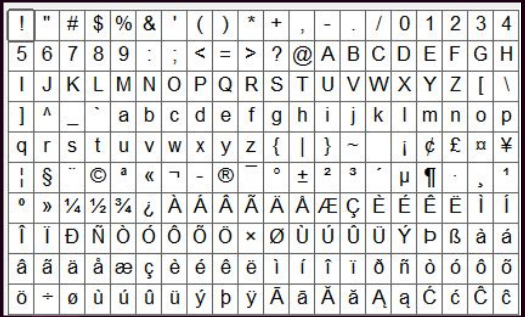
(and that’s it)



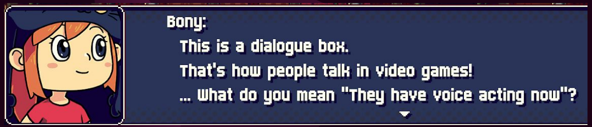
DATA TYPES 4-6

String

Text



!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	0	1	2	3	4
5	6	7	8	9	:	;	<	=	>	?	@	A	B	C	D	E	F	G	H
I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	[	\
]	^	_	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
q	r	s	t	u	v	w	x	y	z	{	|	}	~		¡	¢	£	¤	¥
¦	§	¨	©	ª	«	¬	-	®	¯	°	±	²	³	´	µ	¶	·	¸	¹
º	»	¼	½	¾	¿	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í
Î	Ï	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß	à	á
â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï	ð	ñ	ò	ó	ô	õ
ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ	Ā	ā	Ă	ă	Ą	ą	Ć	ć	Ĉ	ĉ

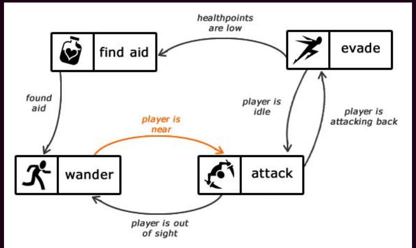


Bony:
This is a dialogue box.
That's how people talk in video games!
... What do you mean "They have voice acting now"?

enum

A category of options defined by the programmer

Days of the Week
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday



find aid

evade

wander

attack

healthpoints are low

found aid

player is near

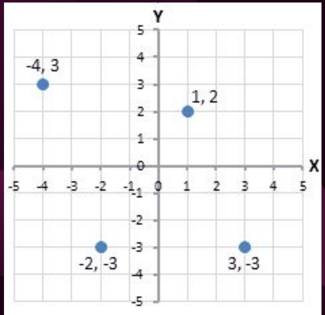
player is idle

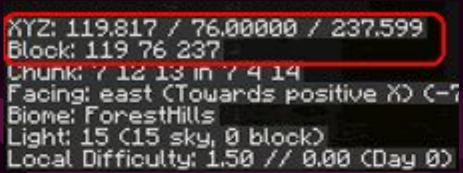
player is attacking back

player is out of sight

Vector2

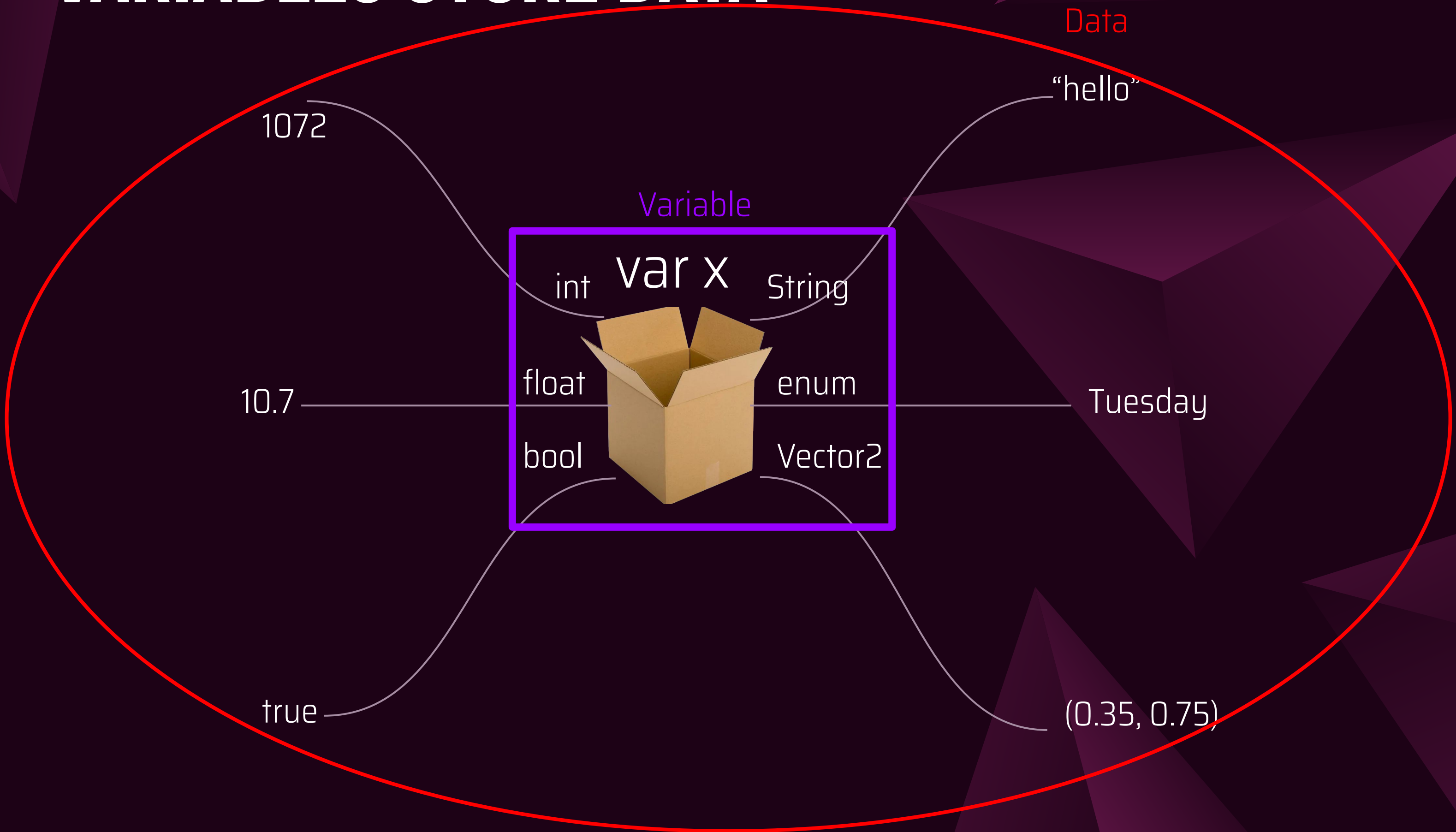
2 floats, called “X” and “Y”





```
XYZ: 119.817 / 76.00000 / 237.599
Block: 119 76 237
Chunk: 7 12 13 in 7 4 14
Facing: east (Towards positive X) (-7
Biome: ForestHills
Light: 15 (15 sky, 0 block)
Local Difficulty: 1.50 // 0.00 (Day 0)
```

VARIABLES STORE DATA



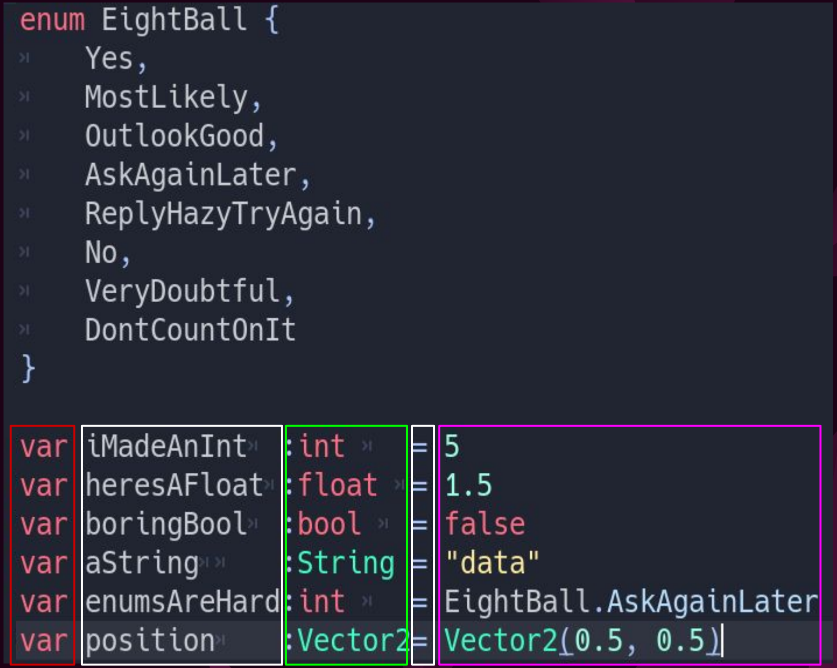
SYNTAX IN GDSCRIPT

How do I create a variable to store some data?

Normally you see it like this:

One of each variable type:

General form distilled into…



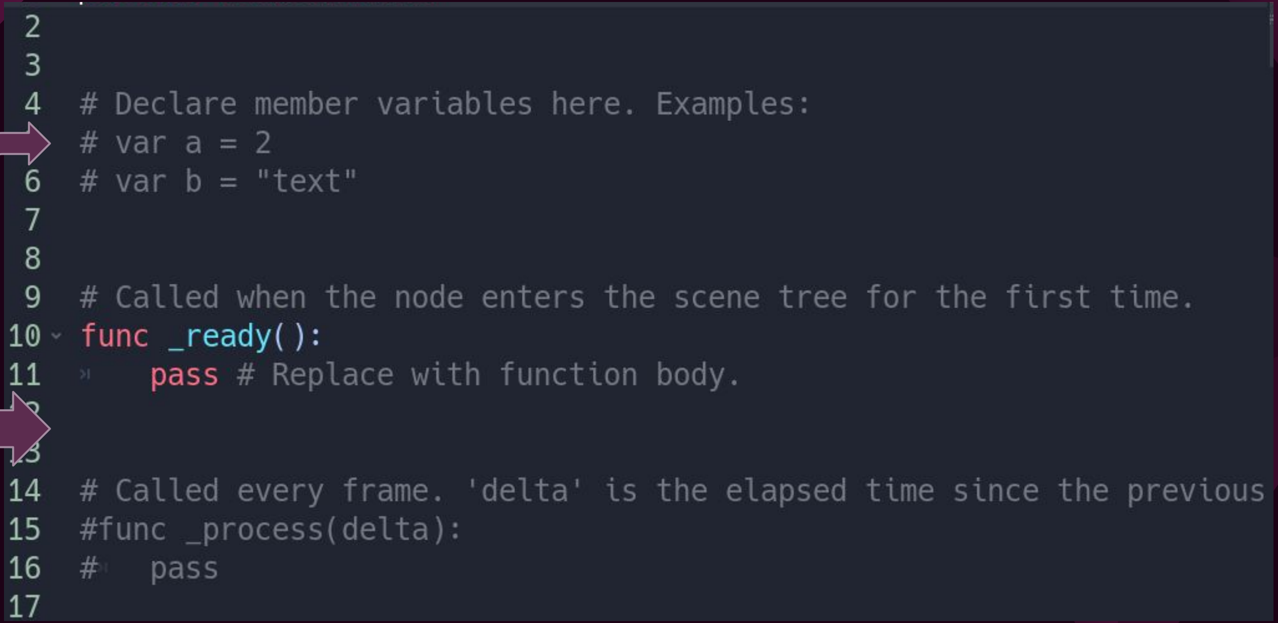
```
enum EightBall {
	Yes,
	MostLikely,
	OutlookGood,
	AskAgainLater,
	ReplyHazyTryAgain,
	No,
	VeryDoubtful,
	DontCountOnIt
}

var iMadeAnInt	:int	= 5
var heresAFloat	:float	= 1.5
var boringBool	:bool	= false
var aString		:String	= "data"
var enumsAreHard:int	= EightBall.AskAgainLater
var position	:Vector2= Vector2(0.5, 0.5)
```

Form to declare a variable var name :type = data

^and this is optional

GODOT STUFF



```


# 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
#func _process(delta):
#	pass

```

Commented section where Godot’s developers tell you it’s best to declare “member variables” here.

Section for two functions that come by default in Godot: _ready() and _process(delta):

Implies the questions: What’s a “member variable”? What happens if I declare a variable in ready or process?

MEMBER VS FUNCTION VARIABLES

Member Variables

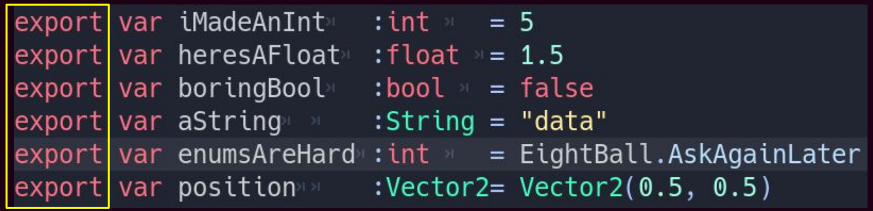
- Shared through every function in a script.
- Can be exposed to be accessible by other scripts.
- Can be exported to be settable through the Godot editor.

Function Variables

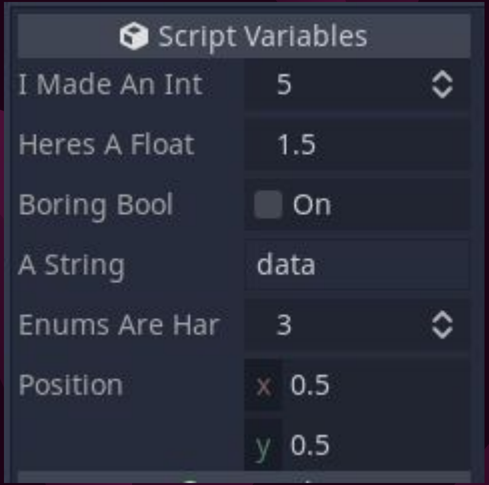
- Only usable in the function that created it.
- Is destroyed when the function is over
- Generally regarded as temporary holding zones for calculations and processes.

Open Godot and show instantiating a member vs function variable.

EXPORTING MEMBER VARIABLES



```
export var iMadeAnInt   :int     = 5
export var heresAFloat  :float   = 1.5
export var boringBool   :bool    = false
export var aString      :String  = "data"
export var enumsAreHard :int     = EightBall.AskAgainLater
export var position     :Vector2= Vector2(0.5, 0.5)
```



Script Variables

I Made An Int	5
Heres A Float	1.5
Boring Bool	□ On
A String	data
Enums Are Har	3
Position	x 0.5 y 0.5

Exporting member variables puts them in the Godot inspector. There, you can use GUI fields to set the initial state of that variable. Useful for…

- Variables you need to change frequently to tune game design
- Entities you're going to clone often and want different starting parameters on some instances