blue_potion.gd:
```
extends Node

var print_one = 1

func _ready() -> void:
	drop_potion()

func drop_potion():
	$AnimationPlayer.play("potion_drop")
	$AnimationPlayer.play("fade")
	if print_one == 1:
		print("Obtained Blue Potion")
		#queue_free()
		print_one += 1
		queue_free()
```

green_potion.gd:
```
extends Node

var print_one = 1

func _ready() -> void:
	drop_potion()

func drop_potion():
	$AnimationPlayer.play("potion_drop")
	$AnimationPlayer.play("fade")
	if print_one == 1:
		print("Obtained Green Potion")
		#queue_free()
		print_one += 1
		queue_free()
```

inventory.gd:
```
extends Resource

class_name Inventory

@export var items: Array[InventoryItem]
```

inventory_item.gd:
```
extends Resource

class_name InventoryItem

@export var name : String = ""
@export var texture : Texture2D
```

inventory_ui.gd:
```
extends Control

@onready var inventory: Inventory = preload("res://inventory/playerinventory.tres")

@onready var slots: Array = $NinePatchRect/GridContainer.get_children()

var is_open = false

func _ready():
	update_slots()
	close()

func update_slots():
	for i in range(min(inventory.items.size(), slots.size())):
		slots[i].update(inventory.items[i])

func _process(delta):
	if Input.is_action_just_pressed("Inventory"):
		if is_open:
			close()
		else:
			open()

func open():
	visible = true
	is_open = true

func close():
	visible = false
	is_open = false
```

inventory_ui_slot.gd:

```
extends Panel

@onready var item_visual: Sprite2D = $CenterContainer/Panel/item_display

func update(item: InventoryItem):
	if !item:
		item_visual.visible = false
	else:
		item_visual.visible = true
		item_visual.texture = item.texture
```

red_potion.gd:

```
extends StaticBody2D


var print_one = 1

func _ready() -> void:
	drop_potion()

func drop_potion():
	$AnimationPlayer.play("potion_drop")
	$AnimationPlayer.play("fade")
	if print_one == 1:
		print("Obtained Red Potion")
		#queue_free()
		print_one += 1
		queue_free()
```

coin.gd:

```
extends Area2D

func _on_body_entered(body):
	queue_free()
```

deadzone.gd:

```
extends Area2D


@onready var timer: Timer = $Timer


func _on_body_entered(_body):
        print("You died")
        timer.start()


func _on_timer_timeout() -> void:
        get_tree().reload_current_scene()
```

<u>item_block.gd:</u> (item_block_2 and item_block_3 are the same but with ‘blue’ and ‘green’ substituted for ‘red’ respectively)

```
extends Node2D

var state = ">= 20"
var player_in_area = false

var red_potion = preload("res://inventory/red_potion.tscn")


func _ready():
        if player_in_area == true:
                $drop_item_timer.start()
                print("item")

func _process(delta):
        if state == ">= 20":
                if player_in_area:
                        drop_red_potion()
        else:
                print("need more coins")


func _on_body_entered(body):
        if body.has_method("player"):
                player_in_area = true

func _on_body_exited(body):
        if body.has_method("player"):
                player_in_area = false

#drops the red potion from the item block with an animation
func drop_red_potion():
        var red_potion_instance = red_potion.instantiate()
        red_potion_instance.global_position = $Marker2D.global_position
```

```
		get_parent().add_child(red_potion_instance)
		$drop_item_timer.start()


#Check the count of coins and change the state
var coin_counter = 0
func _on_area_2d_area_entered(area: Area2D):
	if area.is_in_group("coin"):
		set_coin(coin_counter + 1)
		return coin_counter
func set_coin(new_coin_count: int):
	coin_counter = new_coin_count

func state_change(player):
	if player.has_method("set_coin"):
		if coin_counter == 30:
			state == ">= 30"
		elif coin_counter > 30:
			state == ">= 30"
		else:
			state == "< 30"


monolith1.gd:
extends Area2D

const FILE_BEGIN = "res://levels/sudoku"


#transports the player to the next level upon the player entering the collision
#zone of the monolith
func _on_body_entered(body):
	if body.is_in_group("Player"):
		print("Starting sudoku puzzle")

		var current_scene_file = get_tree().current_scene.scene_file_path
		var next_level_number = current_scene_file.to_int() + 1

		var next_level_path = FILE_BEGIN + str(next_level_number) + ".tscn"
		get_tree().change_scene_to_file(next_level_path)
```

```
player.gd:
extends CharacterBody2D

const SPEED = 140.0
const JUMP_VELOCITY = -330.0

@onready var animated_sprite: AnimatedSprite2D = $AnimatedSprite2D

@export var inv: Inventory

var coin_counter = 0

func player():
	pass

#collects coin if player is in the collision area of the coin
func _on_area_2d_area_entered(area: Area2D):
	if area.is_in_group("coin"):
		set_coin(coin_counter + 1)
		print(coin_counter)

#coin counter
func set_coin(new_coin_count: int) -> void:
	coin_counter = new_coin_count

#gravity when player jumps
func _physics_process(delta: float) -> void:
	if not is_on_floor():
		velocity += get_gravity() * delta


	# jumping
	if Input.is_action_just_pressed("Jump") and is_on_floor():
		velocity.y = JUMP_VELOCITY

	#Move left and right
	var direction := Input.get_axis("MoveLeft", "MoveRight")

	#change sprite direction
	if direction > 0:
		animated_sprite.flip_h = false
	elif direction < 0:
		animated_sprite.flip_h = true
	#Animations based on movement
	if is_on_floor():
		if direction == 0:
			animated_sprite.play("idle")
		else:
			animated_sprite.play("running")
	else:
		animated_sprite.play("jumping")

	#movement
	if direction:
		velocity.x = direction * SPEED
	else:
		velocity.x = move_toward(velocity.x, 0, SPEED)

	move_and_slide()
```

game.gd: (game2, game3, and game4 are the same code with and extra number appended for the different difficulties of the puzzles)

```
extends Node2D

@onready var grid : GridContainer = $GridContainer

var game_grid = []
var puzzle = []
var solution_grid = []

var solution_count = 0

var selected_button: Vector2i = Vector2(-1, -1)

var select_button_answer = 0

const GRID_SIZE = 9

const FILE_BEGIN = "res://levels/level"

func _ready():
	bind_selectgrid_button_actions()
	init_game()
```

```
#if the game is done, move to next platforming level
func _process(delta):
    if is_sudoku_completed():
        transition_to_next_level()

#calls all the functions to create the game, giving the grid its width and depth,
#removing tiles based on the difficulty selected, and populating the grid accordingly
func init_game():
    _create_empty_grid()
    _fill_grid(solution_grid)
    _create_puzzle(Settings.DIFFICULTY)
    _populate_grid()


func _populate_grid():
    game_grid = []
    for i in range(GRID_SIZE):
        var row = []
        for j in range(GRID_SIZE):
            row.append(create_button(Vector2(i, j)))
        game_grid.append(row)

#buttons for the player to click and input the number they want in that slot
func create_button(pos: Vector2i):
    var row = pos[0]
    var col = pos[1]
    var ans = solution_grid[row][col]

    var button = Button.new()
    if puzzle[row][col] != 0:
        button.text = str(puzzle[row][col])
    button.set("theme_override_font_sizes/font_size", 32)
    button.custom_minimum_size = Vector2(52, 52)

    button.pressed.connect(_on_grid_button_pressed.bind(pos, ans))

    grid.add_child(button)
    return button


func _on_grid_button_pressed(pos: Vector2i, ans):
    selected_button = pos
    select_button_answer = ans
```

```
func bind_selectgrid_button_actions():
	for button in $SelectGrid.get_children():
		var b = button as Button
		b.pressed.connect(_on_selectgrid_button_pressed.bind(int(b.text)))

#if the number the player selects for that slot is correct, it turns green
#if the number the player selects for that slot is incorrect, it turns red
func _on_selectgrid_button_pressed(number_pressed):
	if selected_button != Vector2i(-1, -1):
		var grid_selected_button = game_grid[selected_button[0]][selected_button[1]]
		grid_selected_button.text = str(number_pressed)

		if Settings.SHOW_HINTS:
			var result_match = (number_pressed == select_button_answer)

			var btn = game_grid[selected_button[0]][selected_button[1]] as Button

			var stylebox:StyleBoxFlat =
btn.get_theme_stylebox("normal").duplicate(true)
			if result_match == true:
				stylebox.bg_color = Color.SEA_GREEN
			else:
				stylebox.bg_color = Color.DARK_RED
			btn.add_theme_stylebox_override("normal", stylebox)

#generates the solved sudoku puzzle
func _generate_sudoku_soln():
	for i in range(GRID_SIZE):
		var row = []
		for j in range(GRID_SIZE):
			row.append(j + i)
		randomize()
		row.shuffle()
		solution_grid.append(row)

	print(solution_grid)

#fills the grid with the numbers 1-9
func _fill_grid(grid_obj):
	for i in range(GRID_SIZE):
		for j in range(GRID_SIZE):
			if grid_obj[i][j] == 0:
				var numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

```
                numbers.shuffle()
                for num in numbers:
                    if is_valid(grid_obj, i, j, num):
                        grid_obj[i][j] = num
                        if _fill_grid(grid_obj):
                            return true
                        grid_obj[i][j] = 0
                return false
    return true

#creates the 81x81 tiles based on the 'GridContainer' node in the .tscn file

func _create_empty_grid():
    solution_grid = []
    for i in range(GRID_SIZE):
        var row = []
        for j in range(GRID_SIZE):
            row.append(0)
        solution_grid.append(row)



func is_valid(grd, row, col, num):
    return(
        num not in grd[row] and
        num not in get_column(grd, col) and
        num not in get_subgrid(grd, row, col)
    )



func get_column(grd, col):
    var col_list = []
    for i in range(GRID_SIZE):
        col_list.append(grd[i][col])
    return col_list



func get_subgrid(grd, row, col):
    var subgrid = []
    var start_row = (row / 3) * 3
    var start_col = (col / 3) * 3
    for r in range(start_row, start_row + 3):
        for c in range(start_col, start_col + 3):
            subgrid.append(grd[r][c])
    return subgrid
```

```
#generates the puzzle based on the difficulty that it is set to in the settings
func _create_puzzle(difficulty):
    puzzle = solution_grid.duplicate(true)
    var removals = difficulty * 10
    while removals > 0:
        var row = randi_range(0, 8)
        var col = randi_range(0, 8)
        if puzzle[row][col] != 0:
            var temp = puzzle[row][col]
            puzzle[row][col] = 0
            if not has_unique_solution(puzzle):
                puzzle[row][col] = temp
            else:
                removals -= 1


func has_unique_solution(puzzle_grid):
    solution_count = 0
    try_to_solve_grid(puzzle_grid)
    return solution_count == 1


func try_to_solve_grid(puzzle_grid):
    for row in range(GRID_SIZE):
        for col in range(GRID_SIZE):
            if puzzle_grid[row][col] == 0:
                for num in range(1, 10):
                    if is_valid(puzzle_grid, row, col, num):
                        puzzle_grid[row][col] = num
                        try_to_solve_grid(puzzle_grid)
                        puzzle_grid[row][col] = 0
                return
    solution_count += 1
    if solution_count > 1:
        return

#checks if the game is completed after the user has filled in all
#slots with the missing numbers
func is_sudoku_completed():
    for row in range(GRID_SIZE):
        for col in range(GRID_SIZE):
            var btn = game_grid[row][col] as Button
            if btn.text == "" or int(btn.text) != solution_grid[row][col]:
                return false
    return true

#switches to next platforming section
func transition_to_next_level():
    get_tree().change_scene_to_file("res://levels/level2.tscn")
    #var current_scene_file = get_tree().current_scene.scene_file_path
    #var next_level_number = current_scene_file.to_int() + 1
        #
    #var next_level_path = FILE_BEGIN + str(next_level_number) + ".tscn"
    #get_tree().change_scene_to_file(next_level_path)
```

settings.gd: (settings2, settings3, and settings4 are the same with the Difficulty number being different)

```
extends Node

#the sudoku is generated solved

#difficulty number is multiplied by 10 to remove x amount of tiles from the
#(already solved) sudoku

#The maximum on my laptop is 5 before it starts lagging and won't switch
#scenes to the sudoku
var DIFFICULTY = 3
var SHOW_HINTS = true
```