Godot Cheatsheet

Variáveis

```
var x     # Declara uma variável
var x = 1 # Declara uma variável e atribui um valor a ela

x = 2          # Muda o valor de x para 2
x = "Graviola" # Muda o valor de x para "Graviola"
```

Variáveis tipadas

```
var x: int   # Declara uma variável com tipo fixo
var x: int = # Declara uma variável com tipo fixo (explícito) e atribui um valor a ela
var x := 1   # Declara uma variável com tipo fixo (inferido pelo valor atribuido)
```

Nomes de variáveis

```
# O nome de uma variável deve:
# 1. Ser composto apenas de caracteres alfanuméricos e underlines (a-z, A-Z, 0-9, _)
# 2. Começar com uma letra ou underline
# 3. Não ser >apenas< uma underline

# Exemplos válidos:
var graviola
var graviola2
var x
var x_pos
var _private

# Exemplos inválidos:
var 2x
var 123
var _
```

Condições

```
# Condicionais só executam se uma condição específica for atendida.
if x == 1:  # se
    print("x é igual a 1!")
elif x > 1: # senão, se
    print("x é maior que 1!")
else:
    print("x é menor que 1!")
```

Loops

```
# Executa 10 vezes, alterando a variável i de 0 até 9 (o fim do "range" não é incluído)
for i in range(10): # range(fim) (começa em 0)
    print(i)
# i vai de 2 até 4 (o fim do "range" não é incluído)
for i in range(2, 5): # range(início, fim)
    print(i)
# i vai de 5 até -2
for i in range(5, -3, -1): # range(início, fim, passo)
    print(i)
# itera por todos os elementos da lista
var lista: Array[float] = [5, 3, 10.4, 13, 21.2]
for i in lista:
    print(i)

# executa enquanto uma certa condição for verdadeira.
var x = -10
while x < 2:
    print(x)
    x += 1
```

Tipos

```
# Texto. Pode ser usado tanto aspas ("") quanto apóstrofes ('')
var texto:      String  = "Hello, world!"
# Número inteiro, sem vírgula.
var inteiro:    int     = -125
# Números reais. São doubles de 64 bits, mas a maior parte da engine trabalha em 32 bits.
var real:       float   = 3.1415926535
# Variável booleana. Ou é true, ou é false (verdadeiro ou falso).
var verdade:    bool = true
# Vetores 2D. usados para posição direção, velocidade, etc.
var vetores_2d: Vector2 = Vector2(1.5, -3.4)
# Também temos tipos para vetores 3D e 4D.
var vetores_3d: Vector3 = Vector3(1, 2, 3)
var vetores_4d: Vector4 = Vector4(1, 2, 3, 4)
```

Arrays e Dicionários

```
# Arrays são listas de itens
var array1: Array = [1, 2, 3]
# Eles podem conter itens de tipos diferentes
var array2: Array = [1, "Graviola", Vector2(0, 1)]
# A não ser que você especifique o tipo do array!
var array_int:    Array[int] = [1, 2, 3]
var array_float:  Array[float] = [1.0, 0.24, 1, -3]
var array_string: Array[String] = ["Olá,", "Mundo!"]
# Eles podem ser acessados da seguinte maneira, com índices começando em 0:
var elemento_1 = array1[0]
var elemento_2 = array1[1]
var elemento_3 = array1[2]
```

```gdscript
# Dicionários são como "tabelas" chave - elemento
var dict1: Dictionary = {"idade": 21, "nome": "Júlia", 0: "chave [0]"}
# Você também pode especificar o tipo das chaves e dos elementos do dicionário
var dict_idades: Dictionary[String, int] = {"Júlia": 21, "Gerson": 42, "Enzo": 12}
var dict_id: Dictionary[int, String] = {48238: "Júlia", 293: "Gerson", 381803: "Enzo"}

# Dicionários são acessados que nem arrays.
var idade_ju = dict1["idade"]
var chave_0_ju = dict1[0]
var idade_enzo = dict_idades["Enzo"]
```

Funções

```gdscript
# Funções são como procedimentos que fazem alguma coisa.
# Sintaxe geral:
func nome_da_funcao(entrada: tipo_e1, entrada2: tipo_e2) -> tipo_saida:
    # [...] um monte de código
    return algo

# exemplos:
func add(n1: int, n2: int) -> int:
    return n1 + n2

func adulto(idade: int) -> bool:
    if idade >= 18:
        return true
    else:
        return false

# Funções que não retornam nada tem tipo de retorno "void"
var position: Vector2 = Vector2(0, 0)
func set_position(new_pos: Vector2) -> void:
    position = new_pos

#...Você pode omitir os tipos também, mas nn faça isso nn plmdds
func faz_algo(coisa):
    # [...?]
```

Funções e coisinhas úteis

```gdscript
# Operadores matemáticos:
# +, -, *, / (Adição, Subtração, Multiplicação e Divisão)
# %, ** (Módulo (resto de divisão) e Potência

# Você também pode fazer coisas assim:
var x = 10
x *= 10 # O mesmo que x = x * 10
x += 10 # O mesmo que x = x + 10
x **= 10 # O mesmo que x = x ** 10
# E por aí vai...
```

```
# Funções matemáticas em geral
# TODAS as funções matemáticas trabalham com RADIANOS, e NÃO GRAUS.
sin(x) cos(x) tan(x) # seno, cosseno e tangente.
asin(x) acos(x) atan(x) # arco seno, arco cosseno, arco tangente.
floor(x) ceil(x) # arredondam floats pra baixo e pra cima, respectivamente
clamp(x, min, max) # retorna min se x < min, max se x > max, ou x se min < x < max
PI # A constante pi (3.1415926535) já é definida como uma variável dentro da engine.

# Números aleatórios
randf() # Gera um número float aleatório entre 0 e 1.
randi() # Gera um número int aleatório entre 0 e (2^32 - 1).
randf_range(inicio, fim) # Gera um número real aleatório entre início e fim.
randi_range(inicio, fim) # Gera um número inteiro aleatório entre início e fim.
```

Métodos padrão de todos os nodes

```
# Essa função é chamada toda vez que o nó estiver pronto pra ser usado.
# Geralmente só depois de ele já ter entrado na árvore da cena atual, etc.
func _ready() -> void:
    pass

# Essa função é chamada todo frame. "delta" é o tempo (em milissegundos)
# do frame anterior pro atual.
func _process(delta: float) -> void:
    pass

# Essa função é chamada a cada update da física do jogo.
# Ela é chamada MUITO mais frequentemente do que _process().
# Assim como em _process(), delta é o tempo do update anterior pro atual.
func _physics_process(delta: float) -> void:
    pass
```