

LITTLE GAMES

GAMES EM PYTHON COM A PYGAME



PARTICIPANTES

Camila Fontes Santos

Laila Esterfane dos Santos Valença

Marlysson Silva Dantas

Miguel Ferreira França

Vinícius Lima Santos

PYGAME

COMO UTILIZAR A BIBLIOTECA?

DOCUMENTAÇÃO

<https://www.pygame.org/docs/>

INSTALAÇÃO

```
pip install pygame
```



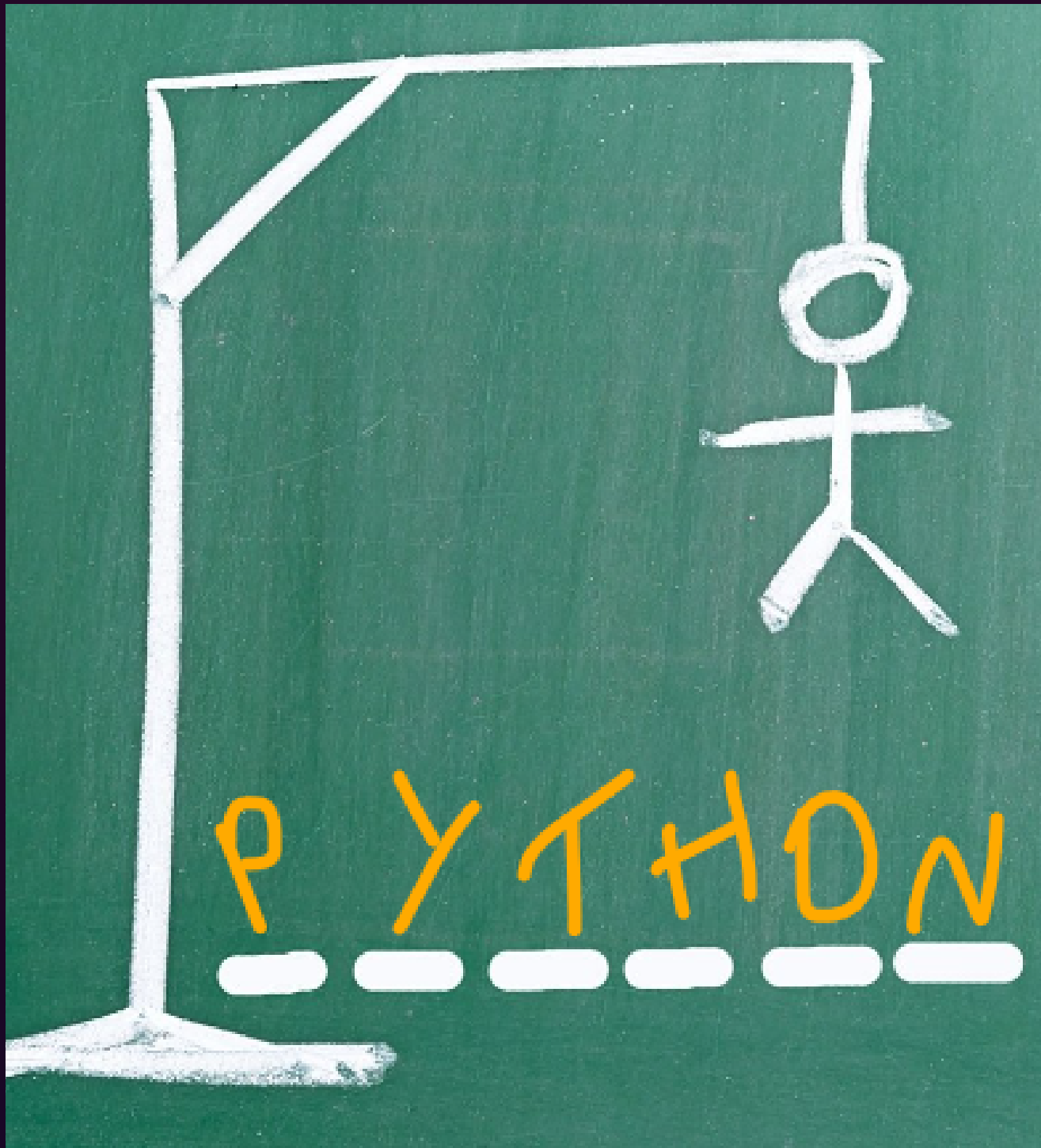
JOGO DA FORCA

DICIONÁRIO


Chaves enumeradas com valores igual
ao nome dos pokemons

PALAVRA REVELADA


- Palavra escondida
- Revela se a letra existir nela
- 10 tentativas com retorno visual




JOGO DA FORCA



```
def verificaLetra(letra, palavra):  
    if letra in palavra:  
        return True  
    else:  
        return False
```



```
def gerarPalavra(palavras):  
    sorteado = randint(1, len(palavras))  
    palavra_escolhida = palavras.get(sorteado)  
    return palavra_escolhida.upper()
```



```
def revelaPalavra(letra, p_completa, p_incompleta):  
    for x in range(len(p_completa)):  
        if letra == p_completa[x]:  
            p_incompleta[x] = letra
```

JOGO DA VELHA

PLAYER X PLAYER

VERIFICAR_VITÓRIA

Verifica a matriz com a “peça” do jogador (X ou O)

VERIFICAR_EMPATE

Verifica espaços em branco na matriz
Empate se não tiver espaços nem vencedor

X	X	O
	O	

JOGO DA VELHA

PLAYER X PLAYER



```
def verificar_vitoria(tabuleiro, jogador):  
    for i in range(3): #Verificação da horizontal e vertical  
        if all(tabuleiro[i][j] == jogador for j in range(3)) or all(  
            tabuleiro[j][i] == jogador for j in range(3)):  
            return True  
    #Verificação das diagonais  
    if all(tabuleiro[i][i] == jogador for i in range(3)) or all(  
        tabuleiro[i][2 - i] == jogador for i in range(3)  
    ):  
        return True  
  
    return False
```

JOGO DA VELHA

PLAYER X PLAYER



```
def verificar_empate(tabuleiro):  
    for linha in tabuleiro:  
        for cell in linha:  
            if cell == " "  
                return False # Tem espaço vazio na matriz  
    return True # Não há espaços
```


SUDOKU

BACKTRACKIN

Algoritmo de backtracking resolve a tabela

VERIFICAR NUMEROS

- Verificar linhas, colunas e quadrados 3x3

```
Numero: 1          linha: 1          coluna: 1
```

SUDOKU



```
# Função para verificar se uma jogada é válida
def verificaJogada(tabuleiro, linha, coluna, numero):
    for x in range(9):
        if tabuleiro[linha][x] == numero and x != coluna: # Verifica na coluna
            return False
        if tabuleiro[x][coluna] == numero and x != linha: # Verifica na linha
            return False
    lin_inicio = (linha // 3) * 3
    col_inicio = (coluna // 3) * 3
    for i in range(3):
        for j in range(3):
            if tabuleiro[lin_inicio + i][col_inicio + j] == numero and (
                lin_inicio + i != linha or col_inicio + j != coluna
            ):
                return False
    return True
```

Snake Game

ORIENTADO A OBJETOS



CLASSE COMIDA

Responsável por gerenciar a comida do jogo.

Construtor.

desenhar()

gerar_celula_aleatoria()

gerar_posicao_aleatoria()

SNAKE GAME

Gerar uma célula aleatória para a comida

```
def gerar_celula_aleatoria(self):  
    x = random.randint(0, numero_celulas - 2)  
    y = random.randint(0, numero_celulas - 2)  
    return Vector2(x, y)
```

Gerar uma posição aleatória para a comida, garantindo que não esteja na cobra

```
def gerar_posicao_aleatoria(self, corpo_cobra):  
    posicao = self.gerar_celula_aleatoria()  
    while posicao in corpo_cobra:  
        posicao = self.gerar_celula_aleatoria()  
    return posicao
```

SNAKE GAME

ORIENTADO A OBJETOS



CLASSE COBRA

Responsável pela cobra e suas funcionalidades.

Construtor.

desenhar()

atualizar()

reiniciar()

Snake Game

ORIENTADO A OBJETOS

CLASSE JOGO

Responsável pelo gerenciamento do Jogo.

Construtor.

desenhar()

atualizar()

verificar_colisao_com_comidas()

verificar_colisao_com_bordas()

verificar_colisao_com_cauda()

fim_jogo()



SNAKE GAME

```
def verificar_colisao_com_comida(self):  
    if self.cobra.corpo[0] == self.comida.posicao:  
        self.comida.posicao = self.comida.gerar_posicao_aleatoria(self.cobra.corpo)  
        self.cobra.adicionar_segmento = True  
        self.pontuacao += 1  
        self.cobra.som_moeda.play()
```

```
def verificar_colisao_com_bordas(self):  
    if self.cobra.corpo[0].x == numero_celulas or self.cobra.corpo[0].x == -1:  
        self.fim_jogo()  
    if self.cobra.corpo[0].y == numero_celulas or self.cobra.corpo[0].y == -1:  
        self.fim_jogo()
```

```
def verificar_colisao_com_cauda(self):  
    corpo_sem_cabeca = self.cobra.corpo[1:]  
    if self.cobra.corpo[0] in corpo_sem_cabeca:  
        self.fim_jogo()
```



THANK YOU

OBRIGADO PELA ATENÇÃO!