1. Importação e Inicialização

```
import pygame

import random
```

- **import pygame**: Importa a biblioteca pygame, que contém todas as funções necessárias para o desenvolvimento de jogos.
- **import random**: Importa a biblioteca random para gerar números aleatórios, que usaremos para a posição dos blocos que caem.

2. Inicialização do Pygame

```
pygame.init()
```

- **pygame.init()**: Inicializa todos os módulos do Pygame, como gráficos e áudio. Isso é necessário antes de usar qualquer função da biblioteca.

3. Configurações da Tela

```
screen = pygame.display.set_mode((800, 600))

pygame.display.set_caption("Dodge the Blocks")
```

- **pygame.display.set_mode((800, 600))**: Cria uma janela de 800 pixels de largura por 600 pixels de altura.
- **pygame.display.set_caption("Dodge the Blocks")**: Define o título da janela do jogo como "Dodge the Blocks".

4. Configurações do Jogo

```
clock = pygame.time.Clock()

player_x = 400

player_y = 500

player_width = 50

player_height = 50

player_velocity = 5
```

- **clock = pygame.time.Clock()**: Cria um objeto Clock para controlar a taxa de atualização (FPS) do jogo.
- **player_x = 400, player_y = 500**: Define as coordenadas iniciais do jogador (quadrado vermelho). O jogador começará no meio inferior da tela.
- **player_width = 50, player_height = 50**: Define a largura e altura do jogador como 50 pixels cada.
- **player_velocity = 5**: Define a velocidade de movimento do jogador (5 pixels por ciclo do loop).

5. Configurações dos Blocos Inimigos

```
block_width = 50

block_height = 50

block_x = random.randint(0, 750)

block_y = -50

block_velocity = 7
```

- **block_width = 50, block_height = 50**: Define a largura e altura dos blocos que caem como 50 pixels cada.
- **block_x = random.randint(0, 750)**: Define a posição inicial horizontal do bloco como um valor aleatório entre 0 e 750 (para que o bloco apareça em qualquer lugar no eixo X da tela).
- **block_y = -50**: Define a posição inicial vertical do bloco como -50 (acima da tela, fora do campo de visão, para simular que o bloco está caindo de fora da tela).
- **block_velocity = 7**: Define a velocidade de queda do bloco (7 pixels por ciclo do loop).

6. Loop Principal do Jogo

```
running = True

while running:

  screen.fill((0, 0, 0))  # Limpa a tela
```

- **running = True**: Variável de controle para o loop principal do jogo. Quando running for False, o jogo termina.
- **while running:**: O loop principal do jogo. Ele se repete continuamente enquanto running for True.
- **screen.fill((0, 0, 0))**: Limpa a tela com a cor preta (RGB: (0, 0, 0)) a cada ciclo, para evitar que os desenhos anteriores permaneçam na tela.

7. Captura de Eventos

```
for event in pygame.event.get():

  if event.type == pygame.QUIT:

    running = False
```

- **for event in pygame.event.get()**: Percorre a lista de eventos (como clicar no botão de fechar a janela, pressionar teclas, etc.).
- **if event.type == pygame.QUIT**: Se o evento for o fechamento da janela (clicando no X da janela), define running = False, o que encerra o loop e fecha o jogo.

8. Movimentação do Jogador

```
keys = pygame.key.get_pressed()
```

```
if keys[pygame.K_LEFT] and player_x > 0:
    player_x -= player_velocity
if keys[pygame.K_RIGHT] and player_x < 750:
    player_x += player_velocity
```

- **keys = pygame.key.get_pressed()**: Captura o estado atual do teclado. Ele retorna uma lista de teclas pressionadas, que pode ser usada para verificar se uma tecla específica está sendo pressionada.
- **if keys[pygame.K_LEFT] and player_x > 0:**: Se a tecla de seta para a esquerda estiver pressionada e o jogador ainda não estiver na borda esquerda da tela, move o jogador 5 pixels para a esquerda.
- **if keys[pygame.K_RIGHT] and player_x < 750:**: Se a tecla de seta para a direita estiver pressionada e o jogador ainda não estiver na borda direita da tela, move o jogador 5 pixels para a direita.

9. Movimentação dos Blocos

```
block_y += block_velocity
if block_y > 600:
    block_y = -50
    block_x = random.randint(0, 750)
```

- **block_y += block_velocity**: Move o bloco 7 pixels para baixo a cada ciclo do loop.
- **if block_y > 600:**: Se o bloco sair da tela (ou seja, se sua coordenada Y for maior que 600), ele é reposicionado acima da tela novamente com uma nova posição horizontal aleatória.
- **block_y = -50**: Coloca o bloco acima da tela, pronto para cair novamente.
- **block_x = random.randint(0, 750)**: Reposiciona o bloco horizontalmente em uma posição aleatória entre 0 e 750 pixels.

10. Detecção de Colisão

```
if player_x < block_x + block_width and player_x + player_width > block_x and player_y < block_y + block_height and player_y + player_height > block_y:
    print("Game Over!")
    running = False
```

- **Essa linha verifica se o jogador colidiu com o bloco**:
    - **player_x < block_x + block_width**: Verifica se o lado esquerdo do jogador está à esquerda do lado direito do bloco.
    - **player_x + player_width > block_x**: Verifica se o lado direito do jogador está à direita do lado esquerdo do bloco.

    - **player_y < block_y + block_height**: Verifica se o topo do jogador está acima da parte inferior do bloco.
    - **player_y + player_height > block_y**: Verifica se a parte inferior do jogador está abaixo da parte superior do bloco.
- Se todas as condições forem verdadeiras, significa que há uma colisão entre o jogador e o bloco.
- **print("Game Over!")**: Exibe "Game Over!" no console (não aparece na tela do jogo).
- **running = False**: Termina o jogo ao interromper o loop.

11. Desenhando o Jogador e o Bloco

pygame.draw.rect(screen, (255, 0, 0), (player_x, player_y, player_width, player_height))

pygame.draw.rect(screen, (0, 0, 255), (block_x, block_y, block_width, block_height))

- **pygame.draw.rect(screen, (255, 0, 0), (player_x, player_y, player_width, player_height))**: Desenha o jogador como um retângulo vermelho (RGB: 255, 0, 0) nas coordenadas player_x, player_y com a largura e altura definidas.
- **pygame.draw.rect(screen, (0, 0, 255), (block_x, block_y, block_width, block_height))**: Desenha o bloco inimigo como um retângulo azul (RGB: 0, 0, 255) nas coordenadas block_x, block_y.

12. Atualizando a Tela

pygame.display.flip()

clock.tick(60)

- **pygame.display.flip()**: Atualiza a tela com o conteúdo recém-desenhado. O Pygame usa um método chamado "double buffering", o que significa que tudo é desenhado primeiro fora da tela e, então, o conteúdo é "flipado" (mostrado) na janela quando essa função é chamada.
- **clock.tick(60)**: Limita a taxa de quadros para 60 frames por segundo (FPS). Isso mantém a velocidade do jogo consistente em diferentes máquinas.

13. Finalizando o Pygame

pygame.quit()

- **pygame.quit()**: Encerra todos os módulos do Pygame corretamente.