GyroStraight (Imersão)

1. Booleano

```
if cancel:
    return

global run_generator, runSmall
global pRegler, iRegler, dRegler
```

A condicional “if cancel:” no código é usada para verificar se a variável chamada cancel é uma condição verdadeira. Se for, a função retornará imediatamente e interromperá a execução. Isso pode ser usado como uma forma de interromper a execução da função, por exemplo, se a função estiver sendo executada em um loop e você quiser interrompê-la prematuramente. A variável (cancel) precisaria ser definida como True em algum lugar fora da função para que a função parasse mais cedo.

Exemplo:

cancel = True

return

Run_genereto, runSmall = 0

pRegler, iRegler, dRegler = 0

print (GYROSTRAIGHT INTERROMPIDO)

2. Booleano

```
if generator == None:
    run_generator = False
```

A linha if generator == None: no código fornecido é usada para verificar se a variável generator é None. Se for, a variável run_generator é definida como False.

Isso é usado posteriormente no código quando a função verifica o valor de run_generator para determinar se o código deve ser executado dentro do bloco if run_generator:. Se run_generator for False, o código dentro do bloco não será executado.

Portanto, o propósito dessa linha é desativar a execução do código dentro do bloco if run_generator: quando a variável generator é None. Isso pode ser útil se a variável generator não for passada como argumento ao chamar a função ou se for definida explicitamente como None dentro da função.

3. Váriaveis principais

```
#Set starting speed of robot
speed = startspeed
#Sets PID values

change = 0
old_change = 0
integral = 0
steeringSum = 0

invert = -1

#Sets values based on user inputs
loop = True


gyroStartValue = getGyroValue()
```

A maior parte desse código é responsável por inicializar variáveis e configurações importantes para o funcionamento do robô.

Inicialização de variáveis: Variáveis como "speed", "change", "old_change", "integral", "steeringSum" e "invert" são inicializadas.

Leitura do valor inicial do giroscópio: A função "getGyroValue()" é chamada para obter o valor inicial do giroscópio, que é armazenado na variável "gyroStartValue".

Configuração do loop principal: A variável "loop" é definida como "True", o que significa que o loop principal do código continuará executando até que seja interrompido de alguma forma (1 e 2).

Em resumo, este trecho de código é usado para inicializar e configurar o ambiente em que o robô será executado. Isso inclui configurar variáveis, ler o valor inicial do giroscópio e definir a configuração do loop principal.

4. Distância

```python
#Error check for distance
if distance < 0:
    print('ERR: distance < 0')
    distance = abs(distance)

#Calulation of degrees the motors should turn to
#17.6 is wheel circumference in cm. You might need to adapt it
rotateDistance = (distance / 17.6) * 360
accelerateDistance = rotateDistance * addspeed
deccelerateDistance = rotateDistance * (1 - brakeStart)
```

O trecho mostrado é um exemplo de código Python que verifica se a distância é válida e calcula os ângulos que os motores devem girar para cobrir essa distância. O código realiza os seguintes passos:

1. Verifica se a distância é negativa. Se for, exibe uma mensagem de erro e converte a distância para positiva.
2. Calcula a rotação do eixo que é necessária para cobrir a distância. Para isso, divide a distância pela circunferência da roda e multiplica o resultado por 360, já que 360 graus equivalem a uma rotação completa.
3. Calcula os ângulos adicionais que os motores devem girar para acelerar e desacelerar. Estes ângulos são baseados na fração da distância que os motores devem percorrer para acelerar e desacelerar.

O trecho de código que você forneceu parece ser responsável por calcular o ponto de frenagem para um robô com dois motores. O código realiza os seguintes passos:

1. Obtém o valor atual dos contadores de graus dos motores esquerdo e direito.
2. Calcula o ponto de frenagem como uma fração da distância total a ser percorrida.
3. Calcula a distância percorrida até o momento usando uma função getDrivenDistance.

5. Execução

```
while loop:
    if cancel:
        break
    if run_generator: #run parallel code execution
        next(generator)

    #Calculation of driven distance and PID values
    oldDrivenDistance = drivenDistance
    drivenDistance = getDrivenDistance(self)

    pidCalculation(speed)
    change = getGyroValue() - gyroStartValue #yaw angle used due to orientation of the self.hub


    currenSteering = (change * pRegler + integral * iRegler + dRegler * (change - old_change)) + offset + steeringSum*0.02

    currenSteering = max(-100, min(currenSteering, 100))
    #print("steering: " + str(currenSteering) + " gyro: " + str(change) + " integral: " + str(integral))

    steeringSum += change
    integral += change - old_change
    old_change = change

    #Calculation of speed based on acceleration and braking, calculation of steering value for robot to drive perfectly straight
    speed = speedCalculation(speed, startspeed,maxspeed, endspeed, accelerateDistance, deccelerateDistance, brakeStartValue, drivenDistance, o
    self.movement_motors.start_at_power(int(speed), invert * int(currenSteering))
```

6. Método de finalizar a execução

```
if stopMethod != None:
    if stopMethod.loop():
        loop = False
elif rotateDistance < drivenDistance:
        loop = False
```

```
if stop:
    self.movement_motors.stop()
```

7. Difunde para um próximo código

```
run_generator = True
runSmall = True
```