Solarbotic_Wheels_two

Get

get_mm_per_sec

Methode

```
get_mm_per_sec(only_mm: bool = False, only_sec: bool = False) -> List[int, float] | int | float
```

Parameter-Erklärung

only_mm

- `True`: Es werden nur die Millimeter zurückgegeben
- `False`: Die Millimeter werden nicht zurückgegeben

only_sec

- `True`: Es werden nur die Sekunden zurückgegeben
- `False`: Die Sekunden werden nicht zurückgegeben

Rückgabe

- `List[int, float]`: Wie lange er in Sekunden für gewisse Millimeter braucht
- `int`: Die Millimeter, die er gebraucht hat
- `float`:
    - Die Sekunden, die er gebraucht hat
    - Der Berechnete Wert der Millimeter pro Sekunde

Allgemeine Erklärung

Diese Methode dient dazu, um die einst kalibrierte Geschwindigkeit zu erhalten.

---

get_distances

Methode

```
get_distances(calibrated: bool = False) -> tuple[list[int], list[int]] | None
```

Parameter-Erklärung

calibrated

- `True`: Schreibt neue Werte in die Datei `distances_arr.txt` und verwendet dabei den zuletzt gemessenen Bias, um die Kalibrierung zu aktualisieren.
- `False`: Liest nur den letzten durchschnittlichen Bias (Standardverhalten).

Rückgabe

- `tuple[list[int], list[int]]`: Enthält zwei Listen - einmal die gemessenen Werte und einmal die entsprechenden Millimeter-Werte. Wird zurückgegeben, wenn `calibrated=False`.
- `None`: Wird zurückgegeben, wenn `calibrated=True`, da in diesem Fall nur die Datei aktualisiert wird.

Allgemeine Erklärung

Diese Methode liest die gespeicherten Distanzen aus der Datei `distances_arr.txt`. Wenn `calibrated=True` gesetzt ist, werden die neuen Werte in die Datei geschrieben. Andernfalls werden die zuletzt gespeicherten Werte (bzw. der letzte Durchschnittsbias) aus der Datei ausgelesen.

---

get_current_standard_gyro

Methode

```
get_current_standard_gyro() -> int
```

Parameter-Erklärung

(Keine Parameter) Diese Methode benötigt keine Eingabeparameter.

Rückgabe

- `int`: Gibt den aktuellen Bias-Wert des Gyrosensors zurück. Je nach Lage des Controllers (stehend oder liegend) wird entweder der `gyro_z`- oder der `gyro_y`-Wert verwendet.

Allgemeine Erklärung

Diese Methode ermittelt den aktuellen Standardwert (Bias) des Gyrosensors. Abhängig davon, ob der Controller steht oder liegt, wird der passende Achsenwert (`gyro_z` oder `gyro_y`) als Referenzwert zurückgegeben.

---

get_degrees

Methode

```
get_degrees(calibrated: bool = False) -> float
```

Parameter-Erklärung

calibrated

- `True`: Aktualisiert die Datei `bias_degrees.txt` mit dem neuesten gemessenen Bias und berechnet anschließend den neuen Durchschnitt.
- `False`: Liest nur den zuletzt gespeicherten durchschnittlichen Bias aus der Datei (Standardverhalten).

Rückgabe

- `float`: Gibt den durchschnittlichen Bias-Wert in Grad zurück. Wenn `calibrated=True`, wird der neue kalibrierte Durchschnitt berechnet und zurückgegeben. Wenn `calibrated=False`, wird der zuletzt gespeicherte Durchschnittswert zurückgegeben.

Allgemeine Erklärung

Diese Methode dient dazu, die durchschnittliche Zeit für eine Drehung aus der Datei `bias_degrees.txt` zu ermitteln.

---

get_bias_gyro_z

Methode

```
get_bias_gyro_z(calibrated: bool = False) -> float
```

Parameter-Erklärung

calibrated

- `True`: Aktualisiert die Datei `bias_gyro_z.txt` mit dem neuesten gemessenen Bias und berechnet anschließend den neuen Durchschnitt.
- `False`: Liest nur den zuletzt gespeicherten durchschnittlichen Bias aus der Datei (Standardverhalten).

Rückgabe

- `float`: Gibt den durchschnittlichen Bias-Wert der **Z-Achse** des Gyrosensors zurück. Wenn `calibrated=True`, wird der neue kalibrierte Durchschnittswert berechnet und zurückgegeben. Wenn `calibrated=False`, wird der zuletzt gespeicherte Durchschnittswert aus der Datei zurückgegeben.

Allgemeine Erklärung

Diese Methode liest den Bias (Nullpunktabweichung) des Gyrosensors auf der **Z-Achse** aus der Datei `bias_gyro_z.txt`.

---

get_bias_gyro_y

Methode

```
get_bias_gyro_y(calibrated: bool = False) -> float
```

Parameter-Erklärung

calibrated

- `True`: Aktualisiert die Datei `bias_gyro_y.txt` mit dem neuesten gemessenen Bias und berechnet anschließend den neuen Durchschnitt.
- `False`: Liest nur den zuletzt gespeicherten durchschnittlichen Bias aus der Datei (Standardverhalten).

Rückgabe

- `float`: Gibt den durchschnittlichen Bias-Wert der **Y-Achse** des Gyrosensors zurück. Wenn `calibrated=True`, wird der neue kalibrierte Durchschnittswert berechnet und zurückgegeben. Wenn `calibrated=False`, wird der zuletzt gespeicherte Durchschnittswert aus der Datei zurückgegeben.

Allgemeine Erklärung

Diese Methode liest den Bias (Nullpunktabweichung) des Gyrosensors auf der **Y-Achse** aus der Datei `bias_gyro_y.txt`.

---

get_bias_accel_y

Methode

```
get_bias_accel_z(calibrated: bool = False) -> float
```

Parameter-Erklärung

calibrated

- `True`: Aktualisiert die Datei `bias_accel_z.txt` mit dem neuesten gemessenen Bias und berechnet anschließend den neuen Durchschnitt.
- `False`: Liest nur den zuletzt gespeicherten durchschnittlichen Bias aus der Datei (Standardverhalten).

Rückgabe

- `float`: Gibt den durchschnittlichen Bias-Wert des **Beschleunigungssensors (Accelerometer)** auf der **Z-Achse** zurück. Wenn `calibrated=True`, wird der neue kalibrierte Durchschnitt berechnet und zurückgegeben. Wenn `calibrated=False`, wird der zuletzt gespeicherte Durchschnittswert aus der Datei zurückgegeben.

Allgemeine Erklärung

Diese Methode liest den Bias (Nullpunktabweichung) des Beschleunigungssensors auf der **Z-Achse** aus der Datei `bias_accel_z.txt`.

---

get_bias_accel_y

Methode

```
get_bias_accel_y(calibrated: bool = False) -> float
```

Parameter-Erklärung

calibrated

- `True`: Aktualisiert die Datei `bias_accel_y.txt` mit dem neuesten gemessenen Bias und berechnet anschließend den neuen Durchschnitt.
- `False`: Liest nur den zuletzt gespeicherten durchschnittlichen Bias aus der Datei (Standardverhalten).

Rückgabe

- `float`: Gibt den durchschnittlichen Bias-Wert des **Beschleunigungssensors (Accelerometer)** auf der **Y-Achse** zurück. Wenn `calibrated=True`, wird der neue kalibrierte Durchschnitt berechnet und zurückgegeben. Wenn `calibrated=False`, wird der zuletzt gespeicherte Durchschnittswert aus der Datei zurückgegeben.

Allgemeine Erklärung

Diese Methode liest den Bias (Nullpunktabweichung) des Beschleunigungssensors auf der **Y-Achse** aus der Datei `bias_accel_y.txt`.

---

get_standard_speed

Methode

```
get_standard_speed() -> int
```

Parameter-Erklärung

(Keine Parameter) Diese Methode benötigt keine Eingabeparameter.

Rückgabe

- `int`: Gibt die **Standardgeschwindigkeit** zurück, mit der sich der Roboter bewegt. Der Wert entspricht der voreingestellten Geschwindigkeitseinheit, die für Bewegungsabläufe verwendet wird.

Allgemeine Erklärung

Diese Methode gibt die aktuell gespeicherte **Standardgeschwindigkeit** des Roboters zurück. Sie dient als Referenzwert für Bewegungsfunktionen, die keine eigene Geschwindigkeitsangabe erhalten. So kann der Roboter einheitlich und reproduzierbar gesteuert werden, ohne dass die Geschwindigkeit manuell angepasst werden muss.

Fahren

break_motor

Methode

```
break_motor(*args: int) -> None
```

Parameter-Erklärung

***args**

- `int`: Beliebig viele Ports, an denen Motoren angeschlossen sind. Alle angegebenen Motoren werden sofort gestoppt.

Rückgabe

- `None`: Diese Methode gibt keinen Wert zurück.

Allgemeine Erklärung

Diese Methode stoppt die Motoren an den angegebenen Ports **sofort**. Sie kann für einzelne oder mehrere Motoren gleichzeitig verwendet werden. `*args` erlaubt es, flexibel mehrere Ports anzugeben, ohne dass eine feste Anzahl von Parametern erforderlich ist. Dies ist nützlich, um schnelle Stopps bei Notfällen oder präzisen Bewegungsanpassungen durchzuführen.

---

break_all_motors

Methode

```
break_all_motors(stop: bool = False) -> None
```

Parameter-Erklärung

stop

- `True`: Schaltet **alle Motoren komplett aus**.
- `False` (Standard): Stoppt nur die aktuelle Bewegung der Motoren, ohne die Stromversorgung vollständig zu unterbrechen.

Rückgabe

- `None`: Diese Methode gibt keinen Wert zurück.

Allgemeine Erklärung

Diese Methode stoppt **alle Motoren sofort**. Je nach Parameterwahl kann sie die Motoren entweder **nur anhalten** oder **komplett ausschalten**. Sie eignet sich für Notfälle, schnelle Bewegungsunterbrechungen oder für Situationen, in denen alle Motoren gleichzeitig deaktiviert werden sollen.

---

align_drive_side

Methode

```
align_drive_side(speed: int, drive_dir: bool = True, millis: int = 5000) -> bool
```

Parameter-Erklärung

speed

- `int`: Geschwindigkeit, mit der der Roboter fahren soll.
    - Positive Werte -> vorwärts
    - Negative Werte -> rückwärts

drive_dir

- `bool`, optional (Standard: `True`): Wenn `True`, fährt der Roboter nach dem Aufprall kurz in die entgegengesetzte Richtung zurück, um erneut ausrichten zu können, ohne sofort wieder zu stoßen. Wenn `False`, bleibt der Roboter so nah wie möglich am Hindernis stehen.

millis

- `int`, optional (Standard: 5000): Maximale Zeit in Millisekunden, die der Roboter für den Ausrichtvorgang versuchen darf.

Rückgabe

- `bool`: `True`, wenn alle angegebenen Motoren den Ausrichtvorgang abgeschlossen haben.

Allgemeine Erklärung

Diese Methode fährt den Roboter vorwärts oder rückwärts, bis er auf ein Hindernis trifft, **ohne dass sich die andere Radseite automatisch anpasst**. Je nach Einstellung kann der Roboter danach kurz zurücksetzen, um eine erneute Ausrichtung zu ermöglichen. Die maximale Ausrichtzeit kann über `millis` begrenzt werden. So kann der Roboter eine Seite präzise an einer Wand oder einem Objekt ausrichten.

---

align_drive_front

Methode

```
align_drive_front(drive_bw: bool = True, millis: int = 2000) -> None
```

Parameter-Erklärung

drive_bw

- `bool`, optional (Standard: `True`): Wenn `True`, fährt der Roboter nach dem Aufprall kurz rückwärts, um sich anschließend leichter drehen zu können. Wenn `False`, bleibt der Roboter direkt am Hindernis ausgerichtet stehen.

millis

- `int`, optional (Standard: 2000): Maximale Zeit in Millisekunden, die der Roboter für den Ausrichtvorgang nach vorne verwenden darf.

Rückgabe

- `None`: Diese Methode gibt keinen Wert zurück.

Allgemeine Erklärung

Diese Methode richtet die Vorderseite des Roboters aus, indem sie vorsichtig gegen ein Hindernis fährt, bis **beide Fronttaster ausgelöst** werden. Wenn ein Taster nicht korrekt gedrückt wird, greift ein **Failsafe-Mechanismus**, um Schäden oder Fehlfunktionen zu vermeiden. Optional kann der Roboter danach ein kleines Stück rückwärts fahren, um eine erneute Drehung zu ermöglichen oder die Ausrichtung zu optimieren. Die maximale Ausrichtzeit wird über `millis` begrenzt.

---

align_drive_back

Methode

```
align_drive_back(drive_fw: bool = True, millis: int = 2000) -> None
```

Parameter-Erklärung

drive_fw

- `bool`, optional (Standard: `True`): Wenn `True`, fährt der Roboter nach dem Aufprall kurz vorwärts, um sich anschließend leichter drehen zu können. Wenn `False`, bleibt der Roboter direkt am Hindernis ausgerichtet stehen.

millis

- `int`, optional (Standard: 2000): Maximale Zeit in Millisekunden, die der Roboter für den Ausrichtvorgang nach hinten verwenden darf.

Rückgabe

- `None`: Diese Methode gibt keinen Wert zurück.

Allgemeine Erklärung

Diese Methode richtet die Rückseite des Roboters aus, indem sie vorsichtig gegen ein Hindernis fährt, bis **beide Rücktaster ausgelöst** werden. Wenn ein Taster nicht korrekt gedrückt wird, greift ein **Failsafe-Mechanismus**, um Schäden oder Fehlfunktionen zu vermeiden. Optional kann der Roboter danach ein kleines Stück vorwärts fahren, um eine erneute Drehung zu ermöglichen oder die Ausrichtung zu optimieren. Die maximale Ausrichtzeit wird über `millis` begrenzt.

---

turn_wheel

Methode

```
turn_wheel(direction: str, speed: int, millis: int) -> None
```

Parameter-Erklärung

direction

- `str`: Richtung, in die der Roboter drehen soll. Mögliche Werte: `"left"` -> nach links drehen, `"right"` -> nach rechts drehen.

speed

- `int`: Geschwindigkeit, mit der das Rad sich drehen soll. Positive oder negative Werte bestimmen auch die Drehrichtung.

millis

- `int`: Dauer in Millisekunden, wie lange der Roboter diese Drehbewegung ausführen soll.

Rückgabe

- `None`: Diese Methode gibt keinen Wert zurück.

Allgemeine Erklärung

Diese Methode lässt den Roboter mit **nur einem Rad** drehen, um Richtungsänderungen vorzunehmen. Durch Variation von `speed` und `direction` kann die Drehbewegung fein gesteuert werden, während `millis` die Dauer der Drehung begrenzt. Die Methode ist nützlich für präzise Kurvenfahrten oder kleine Ausrichtungsanpassungen.

---

drive_straight_condition_analog

Methode

```
drive_straight_condition_analog(
    Instance: Analog,
    condition: str,
    value: int,
    millis: int = 9999999,
    speed: int = None
) -> None
```

Parameter-Erklärung

Instance

- `Analog`: Ein analoges Sensorobjekt, z. B. ein **Licht- oder Abstandssensor**. Es wird der Sensor überwacht, um die Fahrbewegung zu steuern.

condition

- `str`: Vergleichsbedingung für den Sensorwert. Gültige Werte:
  - `"<"` oder `"lt"` -> kleiner als
  - `">"` oder `"ht"` -> größer als
  - `"<="` oder `"let"` -> kleiner oder gleich
  - `">="` oder `"get"` -> größer oder gleich

value

- `int`: Der Zielwert, mit dem der aktuelle Sensorwert verglichen wird. Die Fahrt wird gestoppt, sobald die Bedingung erfüllt ist.

millis

- `int`, optional (Standard: 9999999): Maximale Zeit in Millisekunden, die der Roboter für die Fahrt verwenden darf.

speed

- `int`, optional (Standard: `ds_speed`): Geschwindigkeit, mit der der Roboter geradeaus fährt (Vorwärts oder Rückwärts).

Rückgabe

- `None`: Diese Methode gibt keinen Wert zurück.

Allgemeine Erklärung

Diese Methode fährt den Roboter **geradeaus**, bis ein bestimmter Wert eines **analogen Sensors** erreicht wird. Die Fahrt kann nach Zeit (`millis`) begrenzt werden, um endlose Bewegungen zu verhindern. Sie eignet sich für präzise Bewegungen entlang einer Linie, Annäherung an ein Hindernis oder Lichtquellen und andere Sensor-gesteuerte Fahrmanöver.

turn_to_black_line

Methode

```
turn_to_black_line(direction: str, millis: int = 80, speed: int = None) -> None
```

Parameter-Erklärung

direction

- `str`: Richtung, in die der Roboter drehen soll. Mögliche Werte: `"left"` -> nach links drehen, `"right"` -> nach rechts drehen.

millis

- `int`, optional (Standard: 80): Zeit in Millisekunden, die der Roboter maximal dreht, bevor der Sensorwert erneut geprüft wird. Hinweis: Es wird kein Multithreading verwendet, daher erfolgt die Prüfung sequenziell.

speed

- `int`, optional (Standard: `ds_speed`): Geschwindigkeit der Drehung. Negative Werte bewirken, dass der **Hintergrundsensor** (Back-Light-Sensor) für die Linienerkennung verwendet wird.

Rückgabe

- `None`: Diese Methode gibt keinen Wert zurück.

Allgemeine Erklärung

Diese Methode dreht den Roboter so lange, bis der **Lichtsensor eine schwarze Linie** erkennt. Je nach Vorzeichen der Geschwindigkeit wird entweder der **Front-** oder der **Back-Light-Sensor** verwendet. Die Methode eignet sich für präzise Linienausrichtung oder Navigation entlang von schwarzen Markierungen auf dem Boden.

---

drive_align_line

Methode

```
drive_align_line(direction: str, speed: int = None) -> None
```

Parameter-Erklärung

direction

- `str`: Richtung, in die der Roboter sich auf der Linie ausrichten soll. Mögliche Werte: `"left"` -> nach links ausrichten, `"right"` -> nach rechts ausrichten.

speed

- `int`, optional (Standard: `ds_speed`): Geschwindigkeit, mit der der Roboter fährt, um die Linie zu erreichen und sich auszurichten.

Rückgabe

- `None`: Diese Methode gibt keinen Wert zurück.

Allgemeine Erklärung

Diese Methode richtet den Roboter entlang einer **schwarzen Linie** aus:

- Wenn der Roboter die Linie noch nicht erreicht hat, fährt er **vorwärts oder rückwärts** (abhängig vom Vorzeichen der Geschwindigkeit), bis die Linie gefunden wird, und richtet sich dann aus.
- Optional kann er **rückwärts ausrichten**, sodass ein 180°-Drehmanöver vermieden wird, was Zeit spart.

---

on_line_align

Methode

```
on_line_align(millis: int = None, speed: int = None, leaning_side: str = None) -> None
```

Parameter-Erklärung

millis

- `int`, optional: Maximale Zeit in Millisekunden, die der Roboter nach der Linie suchen darf. Standardwert: `NINETY_DEGREES_SECS`.

speed

- `int`, optional (Standard: `ds_speed`): Geschwindigkeit, mit der der Roboter sich dreht, während er nach der Linie sucht.

leaning_side

- `str`, optional: Seite, auf die der Roboter sich ausrichten soll. Mögliche Werte: `"right"` -> nach rechts ausrichten, `"left"`-> nach links ausrichten. Standardwert: `None`.

Rückgabe

- `None`: Diese Methode gibt keinen Wert zurück.

Allgemeine Erklärung

Diese Methode richtet den Roboter aus, **wenn er bereits auf einer Linie steht**:

- Der Roboter dreht sich in die gewünschte Richtung, um die Linie zu finden.
- Wird die Linie innerhalb der angegebenen Zeit (`millis`) nicht gefunden, dreht er automatisch in die **andere Richtung**.
- Optional kann die Ausrichtung auf eine bestimmte Seite (`leaning_side`) erfolgen, um die Position

präzise anzupassen.

---

black_line

Methode

```
black_line(millis: int, speed: int = None) -> None
```

Parameter-Erklärung

millis

- `int`: Gibt an, wie lange der Roboter der schwarzen Linie folgen soll (in Millisekunden).

speed

- `int`, optional (Standard: `ds_speed`): Geschwindigkeit, mit der der Roboter der Linie folgt. Positive Werte -> vorwärts fahren, negative Werte -> rückwärts fahren.

Rückgabe

- `None`: Diese Methode gibt keinen Wert zurück.

Allgemeine Erklärung

Diese Methode lässt den Roboter **für eine festgelegte Zeitspanne einer schwarzen Linie folgen**. Der Roboter nutzt dabei seine Lichtsensoren, um die Linie zu erkennen und seine Fahrtrichtung kontinuierlich anzupassen. Die Fahrdauer wird über `millis` begrenzt, und die Geschwindigkeit kann über `speed` angepasst werden. Sie eignet sich insbesondere für einfache Linienverfolgungsaufgaben auf fest markierten Wegen.

---

drive_straight

Methode

```
drive_straight(millis: int, speed: int = None) -> None
```

Parameter-Erklärung

millis

- `int`: Zeit in Millisekunden, wie lange der Roboter geradeaus fahren soll.

speed

- `int`, optional (Standard: `ds_speed`): Geschwindigkeit, mit der der Roboter fährt. Positive Werte -> vorwärts fahren, negative Werte -> rückwärts fahren.

Rückgabe

- `None`: Diese Methode gibt keinen Wert zurück.

Allgemeine Erklärung

Diese Methode lässt den Roboter **für eine festgelegte Zeitspanne geradeaus fahren**. Die Bewegung erfolgt mit konstanter Geschwindigkeit über die angegebene Dauer (`millis`). Der Parameter `speed` steuert die Fahrgeschwindigkeit und -richtung. Sie eignet sich besonders für einfache Bewegungsabläufe oder zeitgesteuerte Fahrsequenzen, bei denen keine Sensorrückmeldung benötigt wird.

---

next_to_onto_line

Methode

```
next_to_onto_line(leaning_side: str = None) -> None
```

Parameter-Erklärung

leaning_side

- `str`, optional: Gibt an, auf welcher Seite sich die schwarze Linie befindet, zu der sich der Roboter ausrichten soll. Mögliche Werte:
  - `"right"` -> Linie befindet sich rechts vom Roboter
  - `"left"` -> Linie befindet sich links vom Roboter Standardwert: `None`.

Rückgabe

- `None`: Diese Methode gibt keinen Wert zurück.

Allgemeine Erklärung

Diese Methode bewegt den Roboter **von einer Position neben einer schwarzen Linie direkt auf die Linie** und richtet ihn anschließend korrekt darauf aus. Sie wird verwendet, wenn sich der Roboter nahe einer Linie befindet, diese aber noch nicht exakt unter den Sensoren liegt. Der Parameter `leaning_side` bestimmt, **von welcher Seite** sich der Roboter der Linie nähert. Nach dem Erreichen der Linie richtet sich der Roboter automatisch entlang der Linie aus.

---

drive_til_distance

Methode

```
drive_til_distance(mm_to_object: int, speed: int = None) -> None
```

Parameter-Erklärung

mm_to_object

- `int`: Zielentfernung in **Millimetern**, die zwischen dem **Distanzsensor** und dem **Objekt vor dem Roboter** verbleiben soll. Der Roboter fährt so lange geradeaus, bis diese Distanz erreicht ist.

speed

- `int`, optional (Standard: `ds_speed`): Geschwindigkeit, mit der der Roboter fährt. Positive Werte -> vorwärts fahren, negative Werte -> rückwärts fahren.

Rückgabe

- `None`: Diese Methode gibt keinen Wert zurück.

Allgemeine Erklärung

Diese Methode lässt den Roboter **geradeaus fahren**, bis ein Objekt in der gewünschten Entfernung (`mm_to_object`) vor dem Distanzsensor erkannt wird. Sie nutzt die Messwerte des Distanzsensors, um die Bewegung automatisch zu stoppen, sobald der Roboter die eingestellte Distanz erreicht.

> **Hinweis:** Diese Methode eignet sich besonders zum **präzisen Anfahren von Objekten oder Wänden**, ohne dabei eine Kollision zu riskieren. Stelle sicher, dass der Distanzsensor korrekt ausgerichtet ist und keine Hindernisse zwischen Sensor und Zielobjekt liegen.

---

turn_degrees_far

Methode

```
turn_degrees_far(direction: str, degree: int, straight: bool = True) -> None
```

Parameter-Erklärung

direction

- `str`: Gibt an, **in welche Richtung** der Roboter drehen soll. Mögliche Werte:
  - `"left"` → Der Roboter dreht nach links
  - `"right"` → Der Roboter dreht nach rechts

degree

- `int`: Gibt den **Drehwinkel in Grad (°)** an, um den sich der Roboter drehen soll. Nur Werte zwischen **0 und 180 Grad** werden empfohlen, um eine **effiziente Bewegung** zu gewährleisten.

straight

- `bool`, optional (Standard: `True`): Legt fest, **in welche Fahrtrichtung** die Drehung erfolgen soll:
  - `True`: Der Roboter fährt **vorwärts** in einem Bogen.
  - `False`: Der Roboter fährt **rückwärts** in einem Bogen.

Rückgabe

- `None`: Diese Methode gibt keinen Wert zurück.

Allgemeine Erklärung

Diese Methode lässt den Roboter **eine Drehung über eine bestimmte Anzahl an Grad (°)** ausführen, wobei **nur ein Rad aktiv bewegt** wird. Dadurch entsteht eine **weite, bogenförmige Drehung**, die sich von einer „Drehung auf der Stelle“ unterscheidet. Mit dem Parameter `straight` kann gesteuert werden, ob die Drehung **vorwärts oder rückwärts** ausgeführt wird.

---

turn_degrees

Methode

```
turn_degrees(direction: str, degree: int) -> None
```

Parameter-Erklärung

direction

- `str`: Gibt an, **in welche Richtung** der Roboter sich drehen soll. Mögliche Werte:
  - `"left"` -> Der Roboter dreht sich nach links.
  - `"right"` -> Der Roboter dreht sich nach rechts.

degree

- `int`: Der **Drehwinkel in Grad (°)**, um den sich der Roboter drehen soll. Werte zwischen **0 und 180 Grad** werden empfohlen, um eine **stabile und effiziente Drehbewegung** zu gewährleisten.

Rückgabe

- `None`: Diese Methode gibt keinen Wert zurück.

Allgemeine Erklärung

Diese Methode lässt den Roboter **auf der Stelle um eine bestimmte Anzahl an Grad drehen**, indem **beide Räder gegensätzlich bewegt** werden (ein Rad vorwärts, das andere rückwärts). Dadurch führt der Roboter eine **präzise Rotationsbewegung** um seine Mittelachse aus, ohne seine Position zu verändern.