

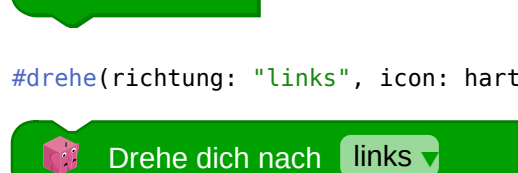
Typst Hathi Blockly Package

Einfache Hathi-Blöcke

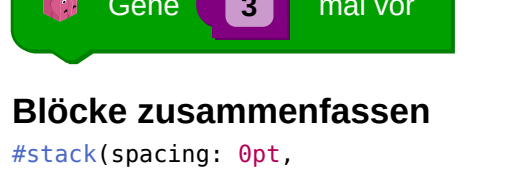
#gehe-vor()



#drehe(richtung: "links")

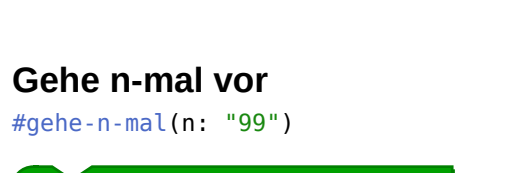


#drehe(richtung: "rechts")

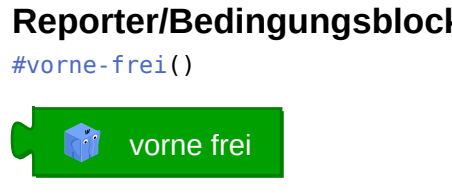


Blöcke mit benutzerdefiniertem Icon

#gehe-vor(icon: hartmut)

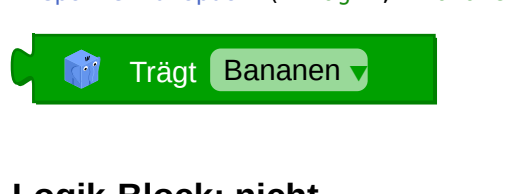


#gehe-n-mal(n: "3", icon: hartmut)



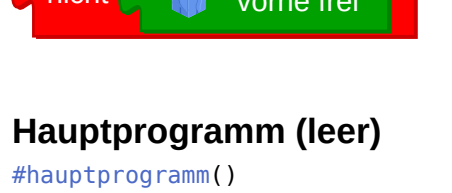
Blöcke zusammenfassen

```
#stack(spacing: 0pt,  
gehe-vor(),  
drehe(richtung: "links"))
```



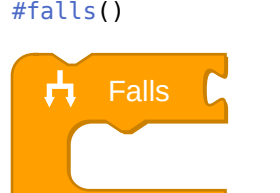
Gehe n-mal vor

#gehe-n-mal(n: "99")

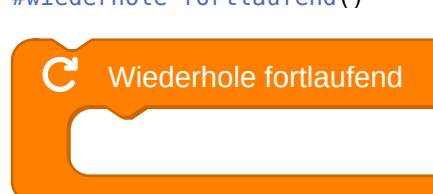


Reporter/Bedingungsblock

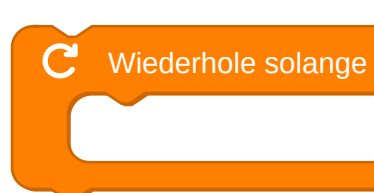
#vorne-frei()



#steht-vor(Objekt: "Bananen")



#steht-vor(Objekt: "Kiste")



#reporter-dropdown("Trägt", "Bananen")

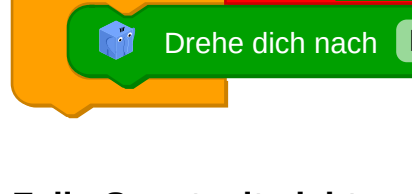


Logik-Block: nicht

#nicht()

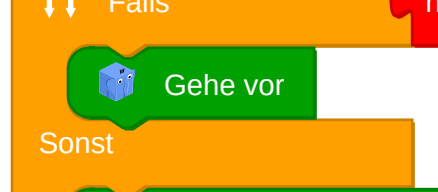


#nicht(condition: vorne-frei())



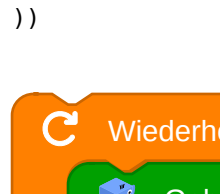
Hauptprogramm (leer)

#hauptprogramm()



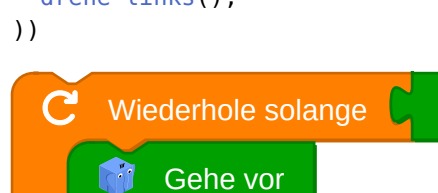
Falls-Block (leer)

#falls()



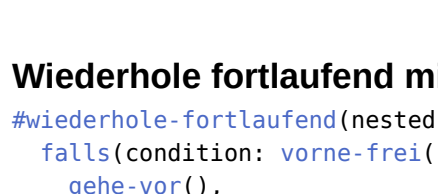
Wiederhole-Block

#wiederhole-fortlaufend()



Wiederhole solange (kopfgesteuert)

#wiederhole-solange()



#wiederhole-solange(condition: vorne-frei())



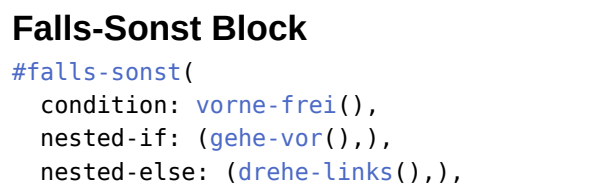
Wiederhole solange mit nicht

```
#wiederhole-solange(condition: nicht(condition: vorne-frei()), nested: (  
  gehe-vor(),  
)
```



Falls mit nicht

```
#falls(condition: nicht(condition: vorne-frei()), nested: (  
  drehe-links(),  
)
```



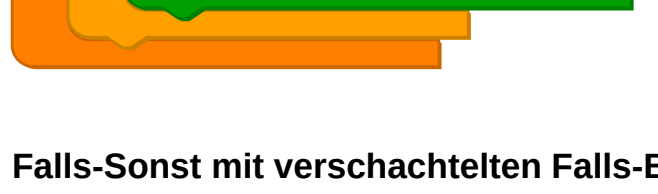
Falls-Sonst mit nicht

```
#falls-sonst(  
  condition: nicht(condition: vorne-frei()),  
  nested-if: (gehe-vor()),,  
  nested-else: (drehe-links()),,  
)
```

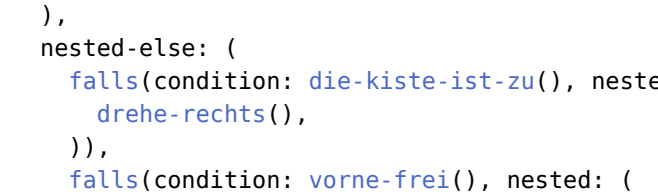


Wiederhole solange mit nested Content

```
#wiederhole-solange(condition: vorne-frei(), nested: (  
  gehe-vor(),  
)
```

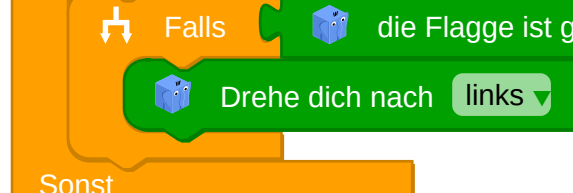


```
#wiederhole-solange(condition: vorne-frei(), nested: (  
  gehe-vor(),  
  falls(condition: vorne-frei(), nested: (  
    gehe-vor(),  
  )),  
  drehe-links(),  
)
```



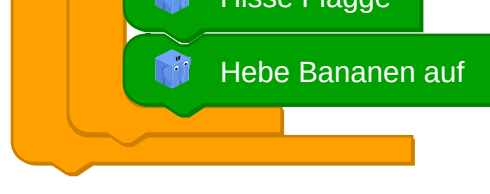
Wiederhole fortlaufend mit nested Content

```
#wiederhole-fortlaufend(nested: (  
  falls(condition: vorne-frei(), nested: (  
    gehe-vor(),  
  )),  
  drehe-links(),  
)
```



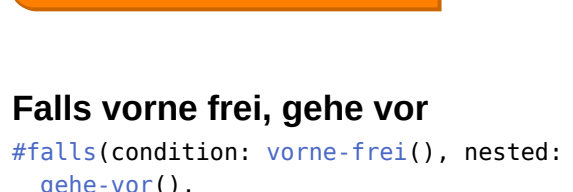
Wiederhole n-mal Block (mit lila Zahlenblock)

#wiederhole-n-mal(n: "4")



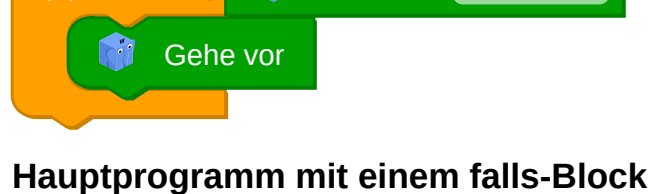
Wiederhole n-mal mit nested Content

```
#wiederhole-n-mal(n: "10", nested: (  
  gehe-vor(),  
  drehe-links(),  
)
```



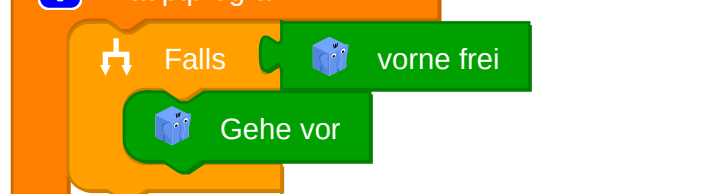
Falls-Sonst Block

```
#falls-sonst(  
  condition: vorne-frei(),  
  nested-if: (gehe-vor()),,  
  nested-else: (drehe-links()),,  
)
```



Falls-Sonst im Hauptprogramm

```
#hauptprogramm(nested: (  
  falls-sonst(  
    condition: vorne-frei(),  
    nested-if: (gehe-vor()),,  
    nested-else: (drehe-links()),,  
  ),  
)
```



Falls-Sonst mit verschachtelten Falls-Blöcken

```
#falls-sonst(  
  condition: vorne-frei(),  
  nested-if: (  
    falls(condition: vorne-frei(), nested: (  
      gehe-vor(),  
    )),  
    falls(condition: die-flagge-ist-gehisst(), nested: (  
      drehe-links(),  
    )),  
  ),  
  nested-else: (  
    falls(condition: die-kiste-ist-zu(), nested: (  
      drehe-rechts(),  
    )),  
    falls(condition: vorne-frei(), nested: (  
      hisse-flagge(),  
      hebe-bananen-auf(),  
    )),  
  ),  
)
```



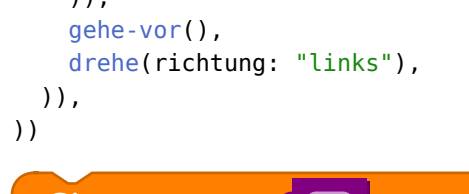
Einfaches Programm mit Hauptprogramm und Bewegungsblöcken

```
#hauptprogramm(nested: (  
  gehe-vor(),  
  gehe-vor(),  
)
```



Falls vorne frei, gehe vor

```
#falls(condition: vorne-frei(), nested: (  
  gehe-vor(),  
)
```

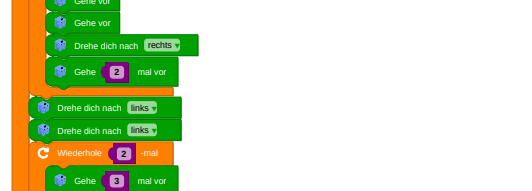


```
#falls(condition: steht-vor(Objekt: "Bananen"), nested: (  
  gehe-vor(),  
)
```



Hauptprogramm mit einem falls-Block

```
#hauptprogramm(nested: (  
  falls(condition: vorne-frei(), nested: (  
    gehe-vor(),  
  )),  
  falls(condition: vorne-frei(), nested: (  
    gehe-vor(),  
  )),  
  falls(condition: vorne-frei(), nested: (  
    gehe-vor(),  
  )),  
)
```

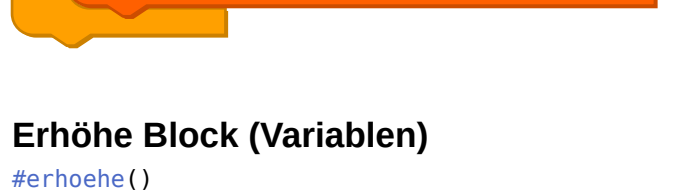


Verschachtelte Falls-Blöcke

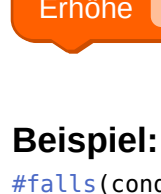
```
#hauptprogramm(nested: (  
  falls(condition: vorne-frei(), nested: (  
    falls(condition: vorne-frei(), nested: (  
      gehe-vor(),  
      drehe-links(),  
    )),  
    gehe-vor(),  
    drehe(richtung: "rechts"),  
  )),  
)
```



```
#wiederhole-n-mal(n: "11", nested: (  
  gehe-vor(),  
  drehe-links(),  
  falls(condition: vorne-frei(), nested: (  
    falls(condition: vorne-frei(), nested: (  
      gehe-vor(),  
      drehe-links(),  
    )),  
    gehe-vor(),  
    drehe(richtung: "links"),  
  )),  
)
```



Beispiel 1

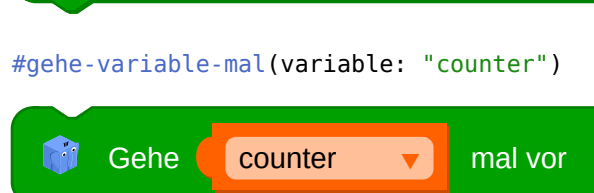


Setze-Auf Block (Variablen)

#setze-auf()

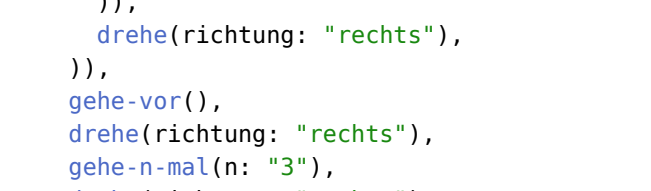


#setze-auf(name: "counter", wert: "42")



Beispiel: Falls vorne frei, setze auf

```
#falls(condition: vorne-frei(), nested: (  
  setze-auf(name: "<Name>", wert: "0"),  
  erhöhe(name: "<Name>"),  
)
```



Erhöhe Block (Variablen)

#erhoehe()



#erhoehe(name: "counter")



Beispiel: Falls vorne frei mit setze-auf und erhöhe

```
#falls(condition: vorne-frei(), nested: (  
  setze-auf(name: "<Name>", wert: "0"),  
  erhöhe(name: "<Name>"),  
)
```


Beispiel: Gehe Variable mal vor

#gehe-variable-mal()

#gehe-variable-mal(variable: "counter")

Verkleinertes Beispielprogramm

```
#scale(30%, reflow: true)[  
  #hauptprogramm(nested: (  
    setze-auf(name: "Bäume", wert: "0"),  
    wiederhole-n-mal(n: "10", nested: (  
      gehe-vor(),  
      drehe(richtung: "links"),  
      falls(condition: steht-vor(Objekt: "Baum"), nested: (  
        erhöhe(name: "Bäume"),  
        drehe(richtung: "rechts"),  
      )),  
      drehe(richtung: "rechts"),  
    )),  
    gehe-vor(),  
    drehe(richtung: "rechts"),  
    drehe(n-mal: "3"),  
    drehe(richtung: "rechts"),  
    gehe-variable-mal(variable: "Bäume"),  
    drehe(richtung: "rechts"),  
    gehe-vor(),  
  ))  
)
```

