Erste Schritte

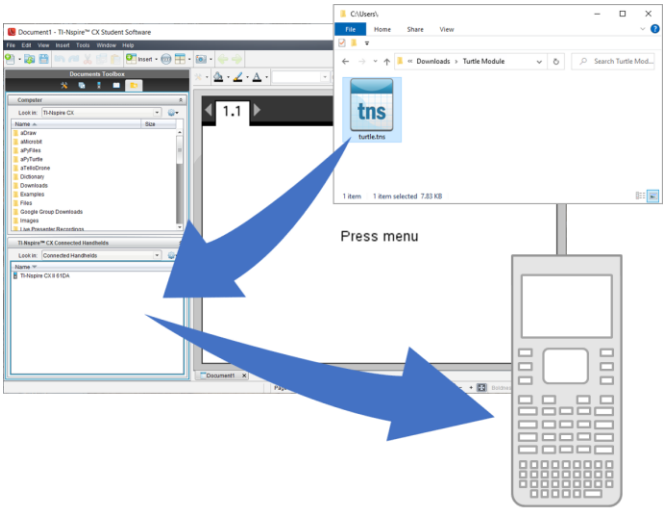
Installation des Turtle-Moduls

1) Übertragen Sie die Turtle-Moduldatei "tns" auf Ihre TI-:Rechner der Nspire CX II-Familie

Nach dem Herunterladen der Zip-Datei und dem Extrahieren der Dateien:

- Öffnen Sie Ihre TI-Nspire CX Desktop Software.
- Schließen Sie Ihre TI-Nspire CX II-Taschenrechner über das Computer-zu-Taschenrechner USB-Kabel, das mit dem Taschenrechner geliefert wird, an Ihren Computer an.
- Stellen Sie sicher, dass der verbundene Rechner im Fenster „Verbundener Handheld“ auf der Registerkarte Inhalts-Explorer der Desktop-Software angezeigt wird.
- Übertragen Sie die Turtle "tns" -Datei auf den angeschlossenen Taschenrechner, indem Sie die Datei in das Fenster Connected Handheld ziehen.
- Nachdem Sie die Datei auf Ihrem Taschenrechner geöffnet haben, fahren Sie mit Schritt 3) **Installieren des Turtle-Moduls** fort**.**

Dieser Schritt benötigt zwingend die TI-Nspire CX Desktop Software für PC oder Mac

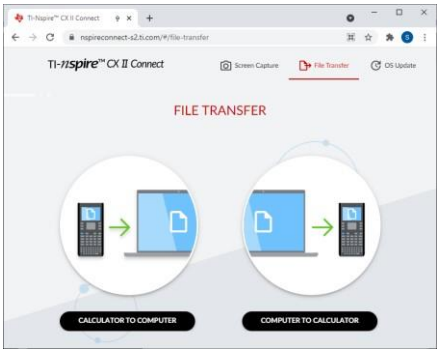


Alternativ (funktioniert **nur mit Chrome-Books**):

- Verwenden Sie das kostenlose TI-Nspire CX II Connect Programm (https://nspireconnect.ti.com/), um die Turtle "tns"-Datei auf einen TI-Nspire CX II zu übertragen, der mit dem Computer-zu-Handheld-USB-Kabel an den Computer angeschlossen ist.

Hinweis: Die Turtle "tns" -Datei kann vor der Installation auch über die Verbindung von Einheit zu Einheit mittels USB-Kabel von Handheld zu Handheld übertragen werden.

- Gehen Sie nun zu Schritt 3) und **Installieren Sie das Turtle-Modul.**

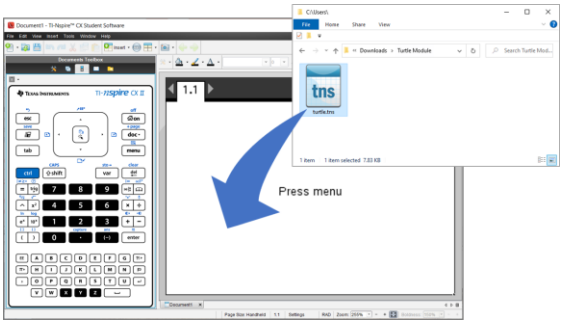


Erste Schritte

2) Übertragen Sie die Turtle-Moduldatei "tns" auf Ihre Desktop-Software (PC oder Mac).

Nach dem Herunterladen der Zip-Datei und dem Extrahieren der Dateien:

- Ziehen Sie die Turtle "tns"-Datei per Drag & Drop auf Ihre TI-Nspire CX II Desktop-Software.
- Nachdem die Datei geöffnet wurde, fahren Sie mit Schritt 3) **Installieren des Turtle-Moduls** fort.

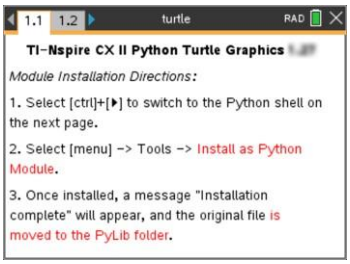


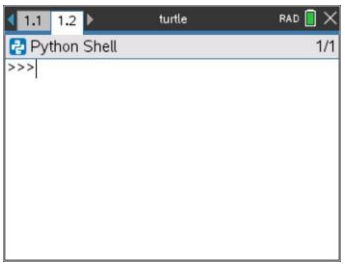
3) Installieren des Turtle-Moduls

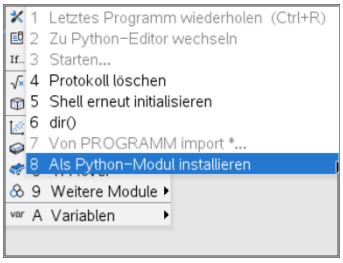
Mit diesen Schritten wird das Modul auf dem Taschenrechner installiert. Dieselben Schritte werden auch verwendet, um das Modul auf der Desktop-Software zu installieren.

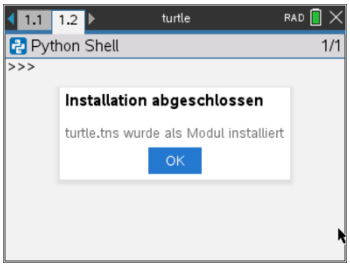
1. Öffnen Sie die Turtle-Datei und lesen Sie die Anweisungen auf der Seite 1.1 .
2. Drücken Sie **[Ctrl] + [~],** um zur Python-Shell auf Seite 1.2 zu wechseln.
3. Drücken Sie **[Menü]** und wählen Sie **Extras** > **Als Python-Modul installieren.**
4. Nach der Installation erscheint die Meldung "Installation abgeschlossen", die die Installation bestätigt.

Hinweis: Nach der Installation wird die Turtle-Moduldatei in den Pylib-Ordner verschoben.









Erste Schritte

4) Zugriff auf die Menüauswahl des Turtle-Moduls

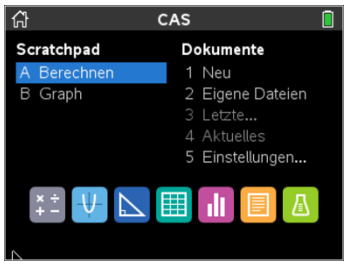
Nachdem das Turtle-Modul die Installation abgeschlossen hat, müssen Sie ein neues Dokument erstellen, um ein Turtle-Programm zu schreiben.

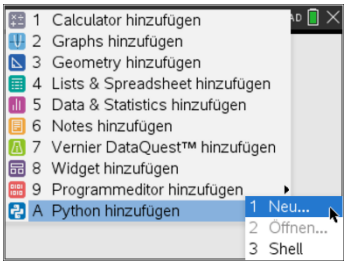
1. Wählen Sie auf dem Startbildschirm **Neu** aus.
2. Wählen Sie **Python hinzufügen**, **Neu...**
3. Geben Sie dem Python-Programm einen Namen (Beispiel: **quadrat**) und drücken Sie **OK**.

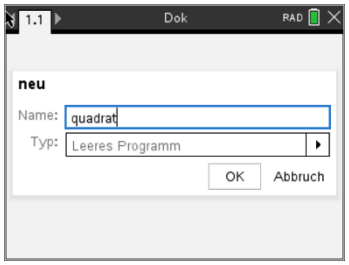
   Jetzt befinden Sie sich auf einer Python-Programm-Editor-Seite.

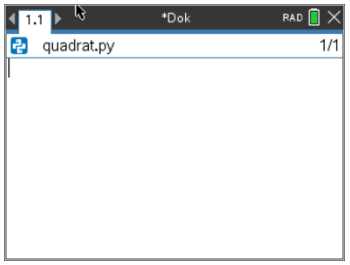
4. Drücken Sie die **[Menü]**-Taste und wählen Sie **Weitere Module** > **Turtle Graphics**

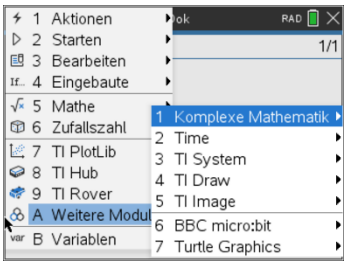
Nun können Sie mit dem Schreiben eines Python-Programms beginnen.
Hinweis: Die abgebildeten Menüs können bei Ihnen etwas anders aussehen.

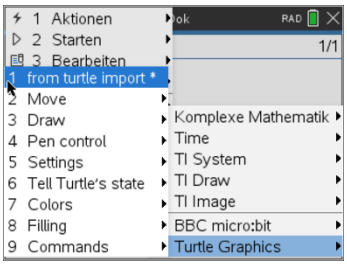












Erste Schritte

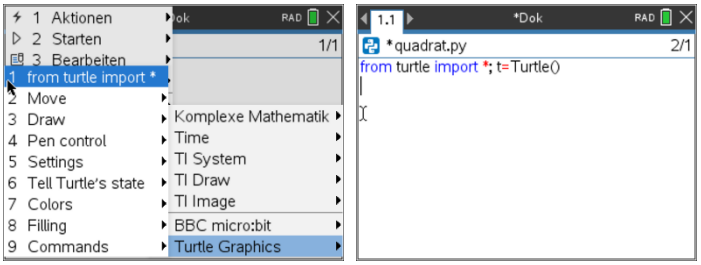
Erstellen Ihres ersten Turtle-Programms: ein Quadrat zeichnen

Fortsetzung der vorherigen Seite (immer noch im Editor-Modus):

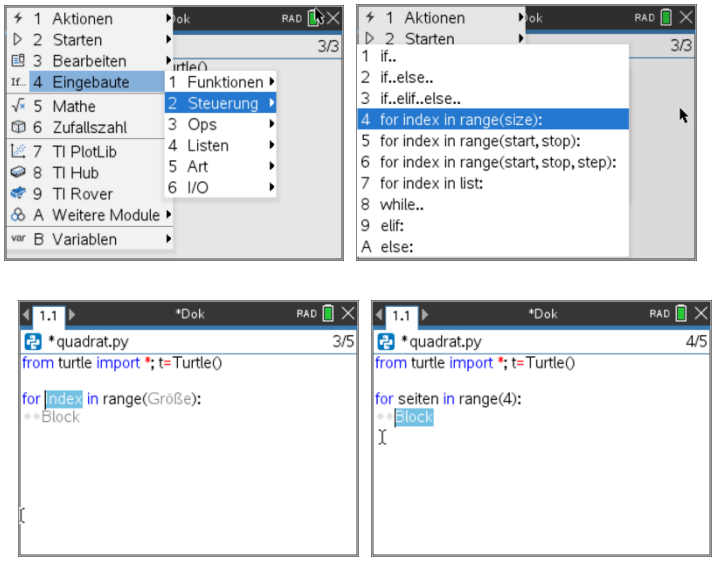
1. Wählen Sie als erstes aus **from turtle import ***

2. Durch Drücken von **[Enter]** werden diese Importanweisung und der Objekt-Bezeichner **t=Turtle()** als erste Zeile in Ihr Programm eingefügt. Dadurch werden die Funktionen und Methoden des Turtle-Moduls in Ihr Programm importiert und Ihre Turtle als Objekt mit dem Namen "t" definiert.

3. *Hinweis*: Bei allen folgenden Menüauswahlen wird von einem Turtle-Objekt mit dem Namen "t" ausgegangen. Ändert man diesen Objektnamen, muss diese Änderung auch in allen folgenden Befehlen berücksichtigt werden.



4. Drücken Sie ein- oder zweimal die Eingabetaste, um zur Zeile 3 zu gelangen.
5. Hinweis: Zeilennummern werden in der oberen rechten Ecke der Seite angezeigt.
6. Drücken Sie [**Menü**] und wählen Sie **Built-ins**, dann **Control** und dann **for index in range(size):**
7. Beachten Sie die Eingabeaufforderung, die das einzugebende Wort hervorhebt
   Durch Drücken der Tabulatortaste werden die Eingabeaufforderungen durchlaufen.
8. Geben Sie "**Seiten**" für den Index und "**4**" für die Größe ein

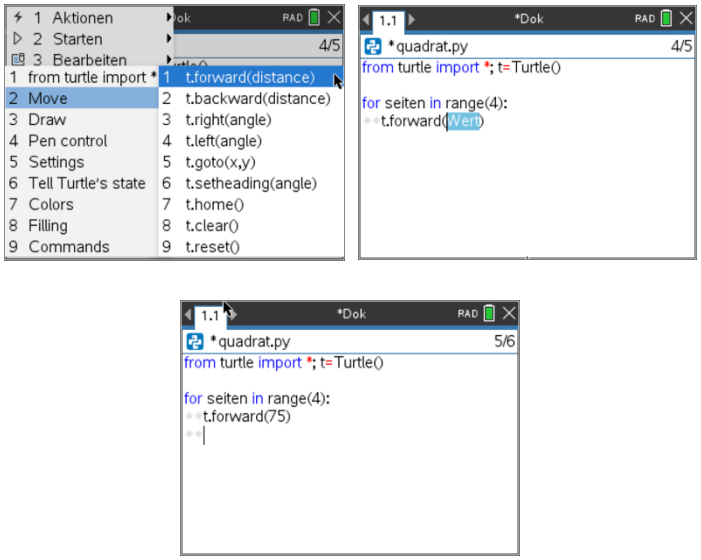


Erste Schritte

9. Drücken Sie mit dem Cursor auf die Eingabeaufforderung "block" und wählen Sie aus dem **[Menü] Weitere Module**, dann **Turtle Graphics**, dann **Move** und dann die Auswahl **t.forward(distance).**
10. Geben Sie die Entfernung ein, die die Turtle vorwärts fahren soll.
    In diesem Beispiel sind es 75 Einheiten.

    Hinweis: Die Entfernungseinheit wird in Pixeln gemessen. Standardmäßig ist die Rasterskala auf 25 Pixel breite Rasterquadrate festgelegt.

11. Springen Sie zum Ende der Zeile, indem Sie **[Tab]** und dann **[Enter]** drücken**,** um eine neue Zeile zu erzeugen.
    Befinden Sie sich schon am Zeilenende, reicht ein einfaches [**Enter**]. Achten Sie auf die richtige Anzahl der Einrückungen!

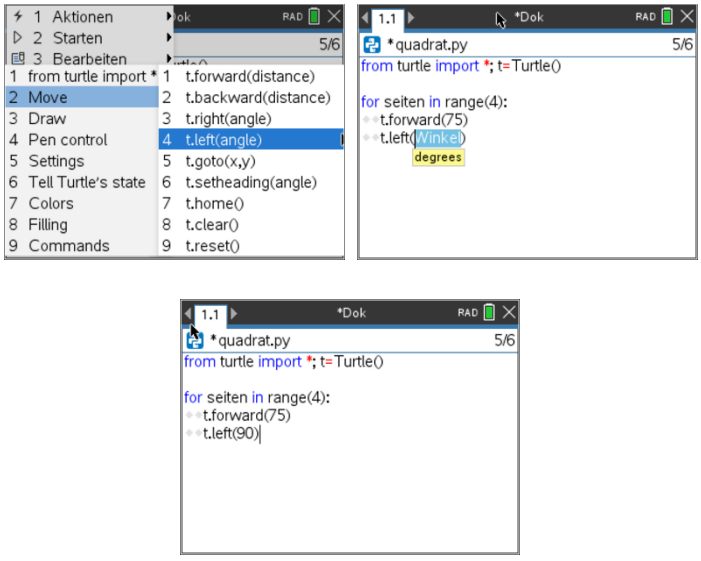


12. Drücken Sie wieder [**Menü**] und wählen Sie **Weitere Module,** dann **Turtle Grafik**, dann **Move** und dann die Auswahl **t.left (Winkel)**
    Beachten Sie den "Tooltip", der den Hinweis "degrees" (Grad) enthält.

13. Geben Sie den Winkel in Grad ein, in dem die Schildkröte nach links abbiegen soll. Für ein Quadrat geben Sie 90 Grad ein.

14. Springen Sie zum Ende der Zeile, indem Sie [Tab] drücken.

Nun sind Sie bereit, Ihr erstes Turtle-Programm auszuführen!



Erste Schritte

15. Um das Programm auszuführen, drücken Sie **[Menü]** und wählen Sie dann **Starten** und nochmals **Starten (**alternativ **(Ctrl+R)).**

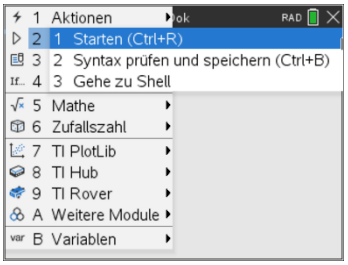
Beachten Sie den Shortcut-Hinweis Ctrl+R in der Menüauswahl. In Zukunft können Sie einfach die Tastenkombination Ctrl + R verwenden, um Ihr Programm auszuführen.

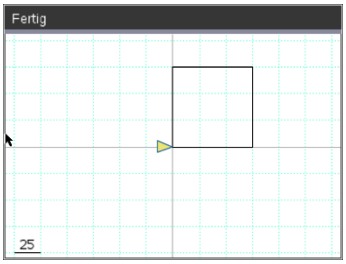
16. Standardmäßig ist die Schildkröte sichtbar und Sie sehen, wie sie zeichnet. Wenn Sie mit der Bewunderung Ihrer Arbeit fertig sind, drücken Sie **[Esc],** um den Bildschirm zu löschen und in die Python-Shell (Seite 1.2) zu wechseln. Drücken Sie **[Ctrl]+[◄],** um zur Editor-Seite 1.1 zurückzukehren. Von dort aus können Sie Ihr Programm ändern und erneut ausführen.

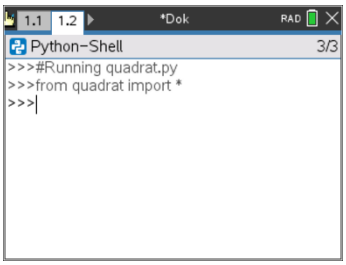
Beachten Sie die Skala in der unteren linken Ecke.

Erweiterungen:

1. Ändern der Stiftdicket
2. Ändern der Stiftfarbe
3. Turtle-Symbol ausblenden
4. Ändern Sie die Geschwindigkeit, mit der sich die Schildkröte bewegt
5. Ausblenden des Raster- und Skalierungsindikators
6. Füllen Sie das Quadrat





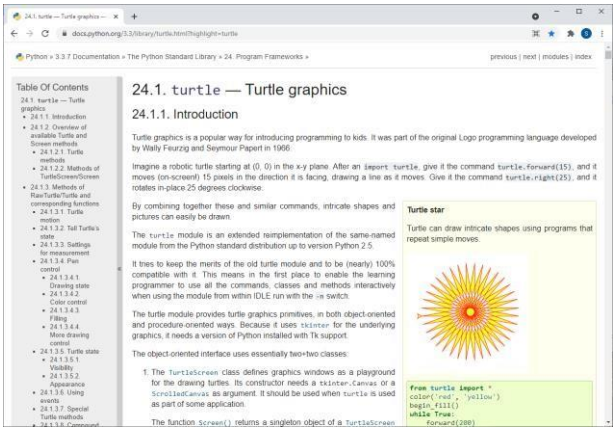


Turtle Modul - Methoden

Wir haben uns bemüht, uns an die Syntax und das zugehörige Verhalten des Python-API (Application Programming Interface) für Turtle Graphics auf der Python Dokumentations-Website zu halten. Obwohl es leichte Verhaltens- und Syntaxunterschiede bei der Implementierung geben kann, besuchen Sie bitte diese Website, um sich mit Syntaxdefinitionen und Turtle-Verhaltensweisen vertraut zu machen.

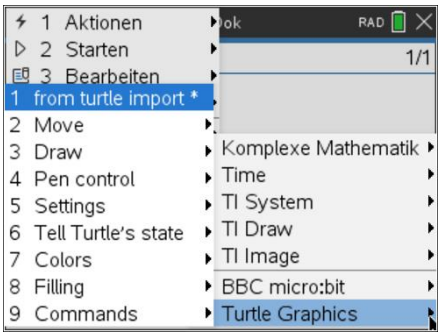
https://docs.python.org/3.3/library/turtle.html?highlight=turtle

Eine Übersicht über Hinweise und bekannte Ausnahmen finden Sie im Abschnitt **Standardeinstellungen** und **bekannte Ausnahmen**.



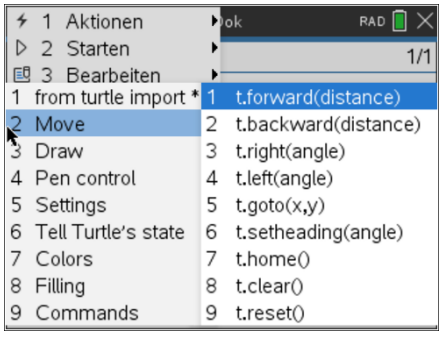
Das Modul-Menü

- from turtle import *
- Move
- Draw
- Pen Control
- Settings
- Tell Turtle's state
- Colors
- Filling



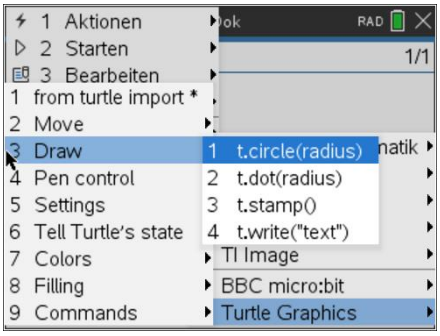
Move

- t.forward(distance)
- t.backward(distance)
- t.right(angle)
- t.left(angle)
- t.goto(x,y)
- t.setheading(angle)
- t.home()
- t.clear()
- t.reset()

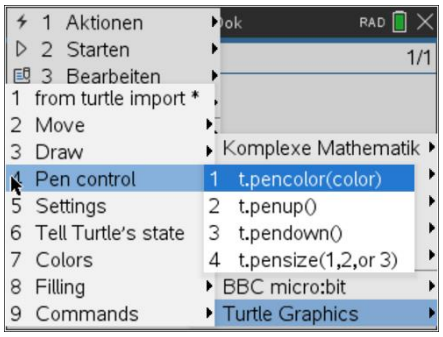


Erste Schritte

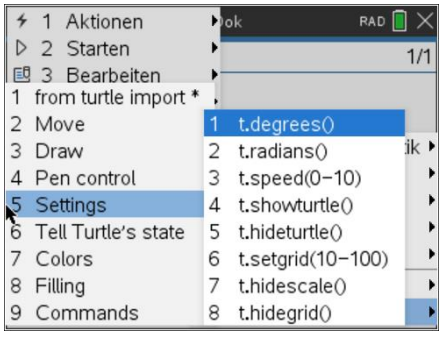
Draw
- t.circle(radius)
- t.dot(radius)
- t.stamp()
- t.write(“text”)



Pen control
- t.pencolor(color)
- t.penup()
- t.pendown()
- t.pensize(1,2,or3)



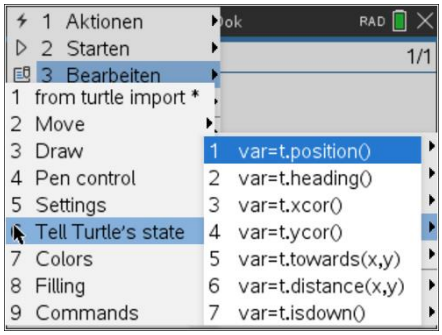
Settings
- t.degrees()
- t.radians()
- t.speed(0-10)
- t.showturtle()
- t.hideturtle()
- t.setgrid(10-100)
- t.hidescale()
- t.hidegrid()
- version()



Erste Schritte

Tell Turtle's state

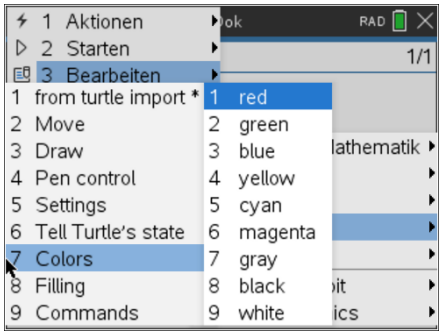
- var=t.position()
- var=t.heading()
- var=t.xcor()
- var=t.ycor()
- var=t.towards(x,y)
- var=t.distance(x,y)
- var=t.isdown()



Colors

- red
- green
- blue
- yellow
- cyan
- magenta
- gray
- black
- white

Beispiele:
t.pencolor("red")
t.fillcolor("green")

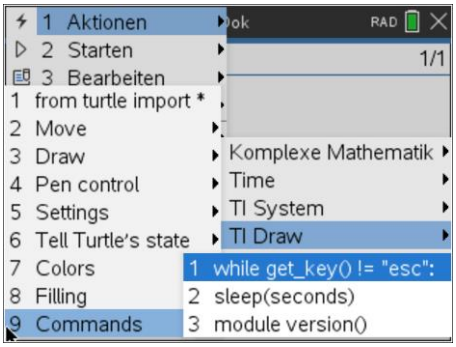


Filling

- t.fillcolor(color)
- t.begin_fill()
- t.end_fill()

Commands

- while get_key()!="esc":
- sleep(seconds)
- module version()



Standardeinstellungen und Ausnahmen	
Wenn diese Methoden nicht programmgesteuert angegeben werden, werden die folgenden Standardwerte angewendet.	
Turtle	**Standardwert: ein**. Verwenden Sie t.hideturtle(), um das Turtle-Symbol auszublenden. Wenn es sichtbar ist, gilt: 1. Die Stiftgröße ist auf 1 festgelegt und kann nicht geändert werden, auch wenn im Programm eine andere Stiftgröße angegeben ist. 2. Die Geschwindigkeit kann mit den Argumenten 1 bis 10 angepasst werden. 3. Bei Verwendung von speed(0) wird das Turtle-Symbol automatisch ausgeblendet. 4. Das Turtle-Symbol kann sich bei der Ausführung langer oder komplexer Programme zunehmend verlangsamen. In diesen Fällen ist es am besten, t.hideturtle() anzuwenden.
Grid	**Standardwert: ein**. Verwenden Sie t.hidegrid(), um das Raster auszublenden.
Scale	**Standardwert: 25 Pixel**. Verwenden Sie t.setgrid(), um die Skalierung des Rasters zu ändern. Gültige Argumente sind 10 bis 100. Verwenden Sie t.hidescale(), um den Skalierungsindikator auszublenden.
Speed	**Standardwert: 5** Verwenden Sie t.speed(), um die Geschwindigkeit zu ändern. Gültige Argumente sind 0 bis 10. Während die Werte 1 bis 10 von langsam bis schnell reichen, ist speed(0) die schnellste (gemäß der Turtle Graphics API).
Pen	**Standarddicke des Stifts: 1**. Um die Stiftgröße anzupassen, verwenden Sie t.pensize() mit den gültigen Argumenten 1, 2 und 3. Hinweis: Die Stiftgröße ist immer 1, wenn die Turtle sichtbar ist. **Standardfarbe des Stifts: schwarz**. Verwenden Sie t.pencolor(color), um die Stiftfarbe zu ändern. RGB-Werte (Rot, Grün, Blau) sind gültige Argumente für die Stiftfarbe mit Werten im Bereich von 0 bis 255. Alternativ können Sie das Argument t.pencolor(color) mit Auswahlen aus dem Menü Farbe füllen. **Standardstellung des Stifts: unten.** Um die Turtle an einen Ort zu bringen, ohne eine Linie zu zeichnen, verwenden Sie die penup()-Methode. Die pendow()-Methode muss anschließend angewendet werden, um mit dem Zeichnen fortzufahren.
Fill	**Standardfüllfarbe: Schwarz.** Verwenden Sie t.fillcolor(color), um andere Farben anzugeben. RGB-Werte (Rot, Grün, Blau) sind gültige Argumente für Füllfarben mit Werten im Bereich von 0 bis 255. Alternativ können Sie das Argument t.fillcolor(color) mit einer Auswahl aus dem Menü Farbe füllen.

Beispielprogramme:

Beispiel 1: Mein erster Stern

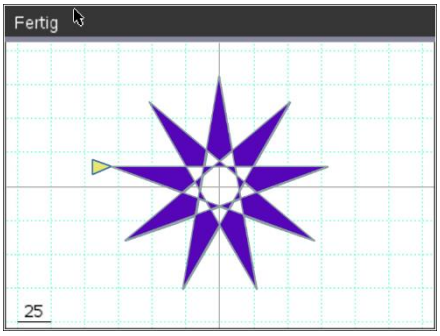
Zeichnen Sie einen einzelnen Stern. Der Stern wird mit einer zufälligen Linienfarbe gezeichnet und mit einer zufälligen Farbe gefüllt.

Beachten Sie, dass t.pencolor() mit R,G,B-Werten (Rot, Grün, Blau) im Bereich von 0 bis 255 definiert werden kann.

sterne.py 1/17

```
from turtle import *; t=Turtle()
from random import *
t.speed(10)
t.penup()
t.goto(-80,14)
t.pendown()
t.pencolor(randint(0,255),randint(0,255),randint(0,255))
t.fillcolor(randint(0,255),randint(0,255),randint(0,255))
t.begin_fill()
while True:
    t.forward(160)
    t.right(160)
    if t.heading()<1:
        break
t.end_fill()
```

Hinweis: Die "break"-Funktion befindet sich derzeit nicht in einer Menüauswahl und muss von Hand eingegeben werden. "True" finden Sie unter [menü], Built-ins, Ops.



Ergänzungen:

1. Ändern Sie den t.right()-Winkel, um Sterne unterschiedlicher Form zu erzeugen. Was passiert, wenn der Winkel klein ist oder ergrößer?
2. Ändern Sie den Abstand t.forward() für immer kleinere Sterne.

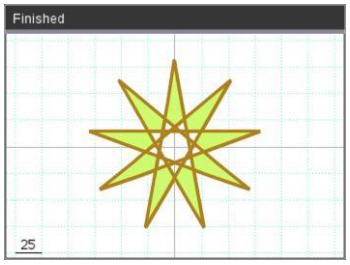
Programm: **sterne.tns**

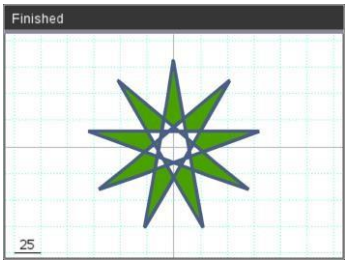
Beispiel 2: Super Nova

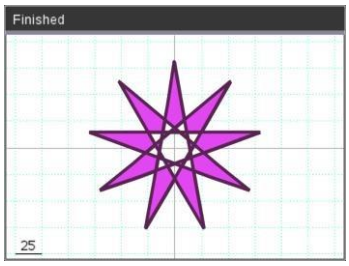
Baut auf dem vorherigen Programm auf, nur dass jetzt jede Sekunde ein neuer Stern erzeugt wird, bis [esc] gedrückt wird.

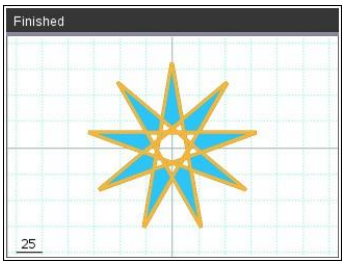
sterne.py 1/22

```
from turtle import *; t=Turtle()
from random import *
from time import *
t.speed(10)
t.hideturtle()
t.pensize(2)
while get_key() != "esc":
    t.penup()
    t.goto(-80,14)
    t.pendown()
    t.pencolor(randint(0,255),randint(0,255),randint(0,255))
    t.fillcolor(randint(0,255),randint(0,255),randint(0,255))
    t.begin_fill()
    while True:
        t.forward(160)
        t.right(160)
        if t.heading()<1:
            break
    t.end_fill()
    sleep(1)
```









Programm: **nova.tns**

Beispiel 3: Sommerblumen

Baut auf den vorherigen Programmen auf, nur dass jetzt die Position der „Blumen“ zufällig ist. Mit [esc] wird das Programm beendet.

```
from turtle import *; t=Turtle()
from random import *
from time import *
t.speed(0)
t.hideturtle()
t.pensize(1)
while get_key() != "esc":
  t.penup()
  t.goto(randint(-200,120),randint(-100,100))
  t.pendown()
  t.pencolor(randint(0,255),randint(0,255),randint(0,255))
  t.fillcolor(randint(0,255),randint(0,255),randint(0,255))
  t.begin_fill()
  while True:
    t.forward(80)
    t.left(162)
    if t.heading()<1:
      break
t.end_fill()
sleep(uniform(.1,.4))
```

Ergänzungen:

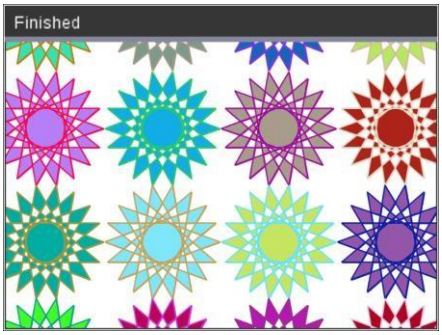
- Verwenden Sie anstelle von Sternen Kreise, Quadrate mit zufällig erzeugten Radien oder Seitenlängen
- Verwenden Sie für den Stift und die Füllung Farben, d Grundfarbe sind.

Programm: **blumen.tns**

Beispiel 4: Großmutters Quilt

Verwendet die vorherigen Programme, nur dass die Sterne (Quilts) ordentlich nebeneinander liegen.

```
from turtle import *; t=Turtle()
from random import randint
t.speed(10)
t.hidegrid()
t.hideturtle()
t.pensize(1)
for j in range(112,-150,-84):
  for i in range(-169,170,86):
    t.penup()
    t.goto(i, j)
    t.pendown()
    t.pencolor(randint(0,255),randint(0,255),randint(0,255))
    t.fillcolor(randint(0,255),randint(0,255),randint(0,255))
    t.begin_fill()
    while True:
      t.forward(80)
      t.left(140)
      if t.heading() < 1:
        break
t.end_fill()
```



Ergänzungen:

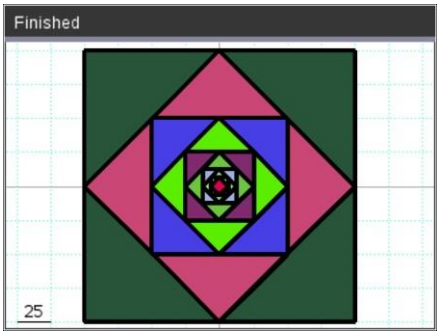
- Ordnen Sie die Sterne so an, dass sie sich sowohl horizontal als auch vertikal überlappen.
- Ändern Sie die Anzahl der Zacken der Sterne.
- Verwenden Sie Kreise oder Quadrate anstelle von Sternen.

Programm: **quilt.tns**

Beispiel 5: Buntglas

Erzeugen Sie mehrere Quadrate innerhalb von Quadraten.

```
from turtle import *; t=Turtle()
from random import randint
from math import sqrt
from ti_system import *
from time import sleep
t.pensize(2)
t.hideturtle()
t.speed(10)
while get_key() != "esc":
  n=200
  t.penup()
  t.goto(-n/2,-n/2)
  t.pendown()
  t.setheading(0)
  for i in range(10):
    t.fillcolor(randint(0,255),randint(0,255),randint(0,255))
    t.begin_fill()
    for j in range(4):
      t.forward(n)
      t.left(90)
    t.end_fill()
    t.forward(n/2)
    t.left(45)
    n=sqrt((n**2)+(n**2))/2
sleep(.5)
```



Ergänzungen:

- Verändern Sie die Anzahl der Quadrate.
- Ändern Sie den Drehwinkel.
- Erzeugen Sie ein Kachelmuster aus Quadraten, die aussehen wie das Original, aber kleiner sind.

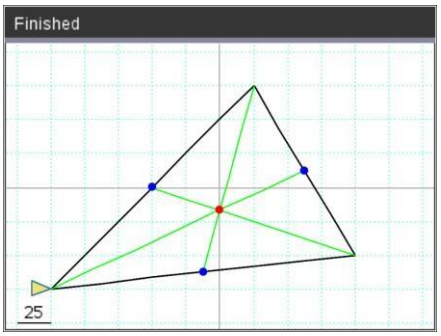
Programm: **buntglas.tns**

Beispiel 6: Schwerpunkt eines Dreiecks

Erzeugen Sie ein Dreieck mit den Seitenmitten, den Seienhalbierenden und dem Schwerpunkt.

```
from turtle import *; t=Turtle()

# Mittelpunkt einer Dreiecksseite
def midpoint(pt1,pt2):
  return ((pt1[0] + pt2[0])/2, (pt1[1] +
pt2[1])/2)

# Einen Punkt zeichnen
def plot_point(pt):
  t.penup()
  t.goto(pt)
  t.pendown()
  t.dot(3)

# Eckpunkte des Dreiecks
v1 = (25,75)
v2 = (-125,-75)
v3 = (100,-50)

# Berechnung des Schwerpunktes sp
sp=((v1[0]+v2[0]+v3[0])/3,(v1[1]+v2[1]
+v3[1])/3)

# Berechnung der Seitenmitten
mid_1_2 = midpoint(v1,v2)
mid_2_3 = midpoint(v2,v3)
mid_1_3 = midpoint(v1,v3)
```

```
# Dreieck in schwarz zeichnen
t.penup()
t.goto(v1)
t.pendown()
t.goto(v2)
t.goto(v3)
t.goto(v1)

# Die drei Seitenhalbierenden in grün
zeichnen
t.pencolor("green")

t.penup()
t.goto(mid_1_2)
t.pendown()
t.goto(v3)

t.penup()
t.goto(mid_2_3)
t.pendown()
t.goto(v1)

t.penup()
t.goto(mid_1_3)
t.pendown()
t.goto(v2)

# Schwerpunkt in rot zeichnen
t.pencolor("red")
plot_point(sp)

# Die Seitenmitten in blau zeichnen
t.pencolor("blue")
plot_point(mid_1_2)
plot_point(mid_2_3)
plot_point(mid_1_3)
```



Ergänzungen:

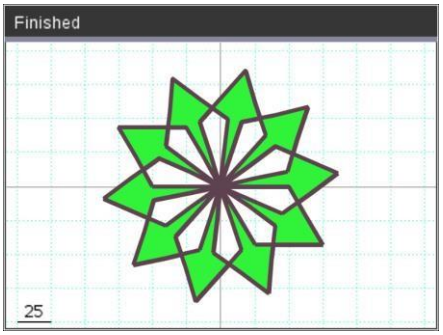
- Ändern Sie die Koordinaten eines Eckpunktes und beobachten Sie das Ergebnis.

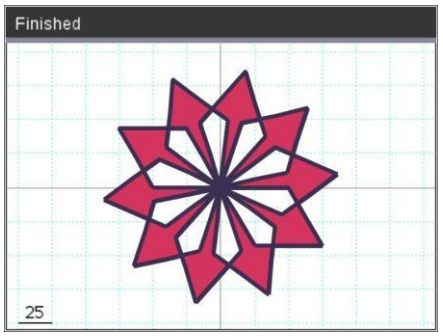
Programm: **schwer.tns**

Beispiel 7: Rautenstern

Eine Raute wird um den Ursprung rotiert. Dabei wird jede Sekunde ein neues Bild erzeugt, bis [esc] gedrückt wird.

```
from turtle import *; t=Turtle() from time import sleep
from random import randint t.speed(10)
t.hideturtle()
t.pensize(2)
while get_key() != “esc”:
  t.pencolor(randint(0,255),randint(0,255),randint(0,255))
  t.fillcolor(randint(0,255),randint(0,255),randint(0,255))
  t.begin_fill()
  for I in range(10):
    for j in range (2):
      t.forward(50)
      t.right(60)
      t.forward(50)
      t.right(120)
    t.right(36)
  t.end_fill()
  sleep(1)
```





Ergänzungen:

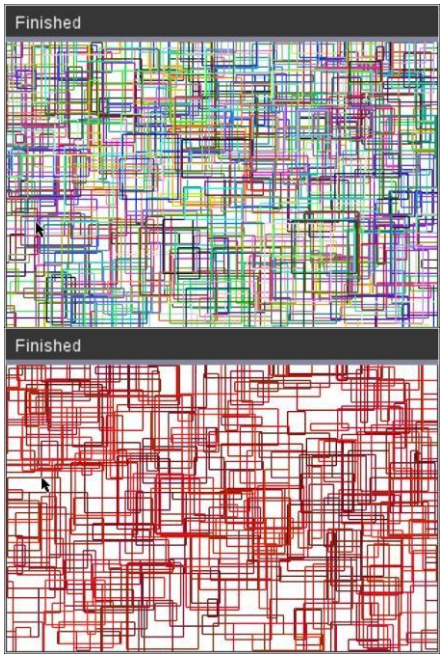
- Verwenden Sie ein Quadrat anstelle der Raute.
- Lassen Sie einen Kreis rotieren.
- Ändern Sie die Anzahl der Objekte, die um den Ursprung rotieren.

Programm: **raute.tns**

Beispiel 8: Rechteck-Muster

Erzeugen Sie zufällig verteilte unterschiedliche große Rechtecke, die solange gezeichnet werden, bis [esc] gedrückt wird.

```
from turtle import*;t=Turtle()
from ti_system import *
from random import randint
t.hidegrid()
t.speed(10)
t.hideturtle()
t.pensize(1)
while get_key() !="esc":
  t.penup()
  t.goto(randint(-200,150), randint(-120,100))
  t.pendown()
 t.pencolor(randint(0,255),randint(0,255),randint(0,255))
 b=randint(5,60)
 h=randint(5,60)
 for i in range(2):
    t.forward(b)
    t.left(90)
    t.forward(h)
    t.left(90)
```



Ergänzungen:

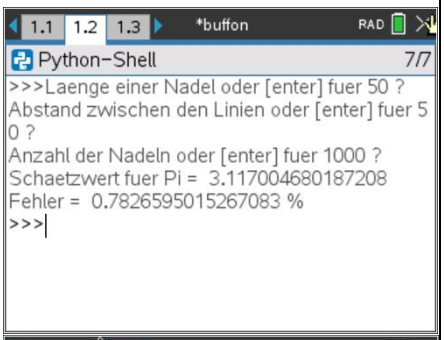
- Füllen Sie die Rechtecke.
- Verwenden Sie Quadrate anstelle von Rechtecken.

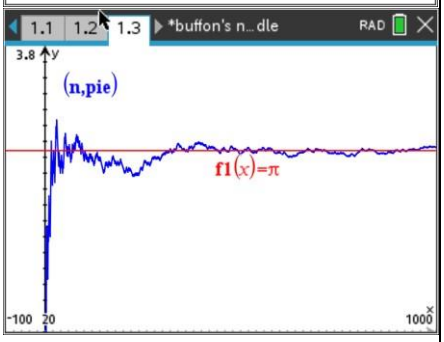
Programm: **recht.tns**

Beispiel 9: Buffons Nadeln

Simulieren Sie Buffons Nadel-Experiment.

```
from ti_system import *
from random import *

#Sie starten zunächst in der Shell,
#bevor Sie in die Turtle-Umgebung wechseln.

clear_history()
length=int(input("Laenge einer Nadel oder
[enter] fuer 50 ?")or '50')
spacing=int(input("Abstand zwischen den
Linien oder [enter] fuer 50 ?") or '50')
needles=int(input("Anzahl der Nadeln oder
[enter] fuer 1000 ?") or '1000')
the_pies=[]
the_count=[]

# Einstellungen der Turtle-Umgebung

from turtle import *; t=Turtle()
t.hidegrid()
t.speed(0)
w,h=get_screen_dim()
# Liste der x-Koordinaten aller Linien
the_lines=[]
#Anzahl der Linien
n_lines=int((w/spacing)/2)+2
# Anzahl der Nadeln, die eine Linie schneiden
crossing = 0
# berechneter Wert von Pi
estimate = 0

# Zeichnen aller Linien
for x in range(-
n_lines*spacing,(n_lines+1)*spacing,spacing):
  the_lines.append(x)
  t.penup()
  t.goto(x,-h/2)
  t.pendown()
  t.goto(x,h/2)

 # Zeichnen der Nadeln
for i in range(needles):
  x1=randint(0,w)-w//2
  y1=randint(0,h)-h//2
  angle=random()*360
  t.penup()
  t.goto(x1,y1)
  t.setheading(angle)
  t.forward(length)
  x2= t.xcor()
  y2= t.ycor()
  t.pencolor("red")

# Beruehrt oder schneidet
eine Nadel eine Linie?
  for n in
range(len(the_lines)):
    if((x1<=the_lines[n] and
x2>=the_lines[n]) or
(x2<=the_lines[n] and
x1>=the_lines[n])):
      t.pencolor("green")
      crossing+=1
  t.pendown()
  t.goto(x1,y1)

# Buffons Formel
  try:
    estimate =(
2*length*i)/(crossing*spacing)
  except:

    pass
  the_pies.append(estimate)
  the_count.append(i)
store_list("n",the_count)
store_list("pie",the_pies)
error=(pi-estimate)*100/pi
print("Schaetzwert fuer Pi = ",
estimate,"\nFehler =
",error,"%")
```





Ergänzungen:

- Ändern Sie die Nadellängen und den Abstand zwischen den Linien.

Programm: **buffon.tns**