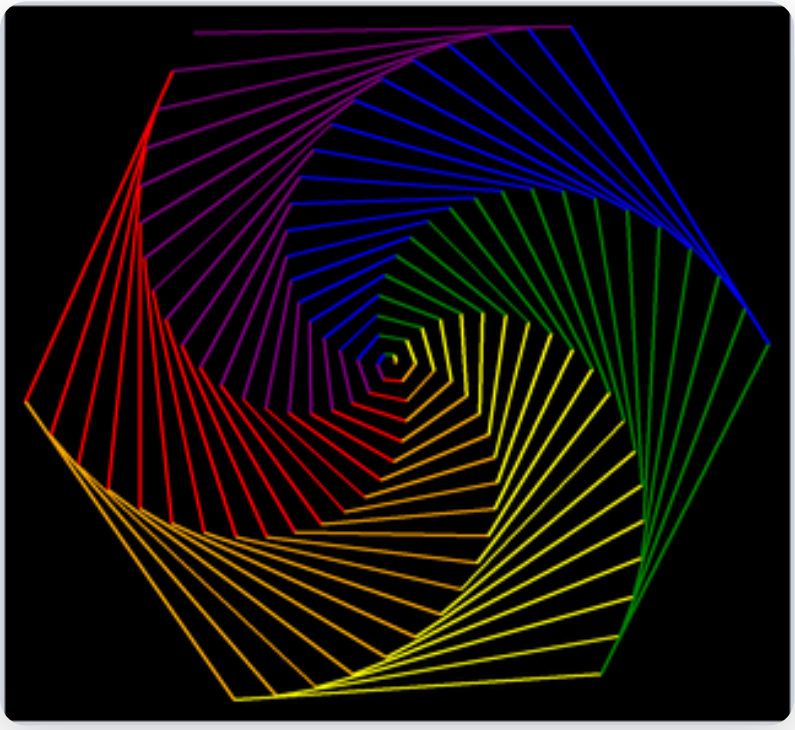
PYTHON LERNEN MIT DER SCHILDKRÖTE

Python-Turtle: Stationen mit PRIMM

7 Stationen zum selbstständigen Arbeiten in deinem eigenen Tempo. Du brauchst nur Thonny und ein bisschen Mut zum Ausprobieren.

1 Vorhersagen | 2 Ausführen | 3 Untersuchen | 4 Verändern | 5 Erfinden



Name _____ Klasse _____

So arbeitest du

In 7 Stationen bringst du einer Schildkröte (englisch: *turtle*) mit Python das Zeichnen bei. Du arbeitest allein und in deinem eigenen Tempo. Wenn du nicht weiterkommst, fragst du zuerst deine Nachbarin oder deinen Nachbarn, dann die Lehrkraft.

Jede Station hat fünf Schritte: PRIMM

Schritt	Was du tust
1 Vorhersagen (Predict)	Lies das Programm, **ohne** es zu starten. Schreib oder zeichne auf, was passieren wird.
2 Ausführen (Run)	Tippe das Programm in Thonny ab und starte es. Stimmt deine Vorhersage?
3 Untersuchen (Investigate)	Beantworte die Fragen. Probiere kleine Änderungen aus und beobachte, was passiert.
4 Verändern (Modify)	Ändere das Programm so, dass es etwas Neues zeichnet.
5 Erfinden (Make)	Schreibe ein eigenes Programm. Bei vielen Aufgaben siehst du ein Zielbild.

Tippe die Programme ab, statt sie zu kopieren. Beim Abtippen merkst du dir die Befehle viel besser.

Die Sterne zeigen, wie schwer eine Aufgabe ist

- ★ **Grundaufgabe**: Die schaffst du.
- ★★ **Weiterdenken**: Hier musst du selbst eine Idee haben.
- ★★★ **Knobelaufgabe**: Für Profis, die es genau wissen wollen.

Du machst in jeder Station **alle ★-Aufgaben und mindestens eine ★★-Aufgabe**. Dann darfst du zur nächsten Station. Die ★★★-Aufgaben sind freiwillig.

So arbeitest du mit Thonny

1. Öffne Thonny und klicke auf **Datei → Neu**.
2. Speichere die Datei mit einem Namen wie `station1.py`. **Nenne eine Datei nie `turtle.py`!** Sonst findet Python die Schildkröte nicht mehr.
3. Starte dein Programm mit dem **grünen Pfeil** oder der Taste **F5**.
4. Erscheint unten eine rote Fehlermeldung, lies sie genau. Sie sagt dir, in welcher Zeile der Fehler steckt.
5. Schließe das Turtle-Fenster, bevor du das Programm noch einmal startest.
6. Tipp: Unter **Ansicht → Variablen** zeigt dir Thonny, welche Werte deine Variablen gerade haben.

Du kennst schon Scratch?

Dann kennst du schon viel! In Python schreibst du die Blöcke als Text. Darauf musst du achten:

- **Groß- und Kleinschreibung zählt**: `forward` funktioniert, `Forward` nicht.

- Jeder Befehl braucht **runde Klammern**, auch wenn nichts darin steht: `penup()`.
- Was in Scratch *in* einem Block steckt, wird in Python **eingerückt** (4 Leerzeichen oder die Tab-Taste).
- Die Schildkröte startet in der Mitte und schaut nach **rechts**, genau wie die Scratch-Katze mit Richtung 90.
- Am Ende jedes Programms steht `done()`. Dann bleibt das Fenster offen.

Dein Laufzettel

- [ ] Station 1: Die Schildkröte bewegen
- [ ] Station 2: Farben, Stift und Füllen
- [ ] Station 3: Wiederholen mit Schleifen
- [ ] Station 4: Variablen
- [ ] Station 5: Eigene Befehle mit Funktionen
- [ ] Station 6: Zufall und Entscheidungen
- [ ] Station 7: Dein Abschlussprojekt

STATION 1

1 Die Schildkröte bewegen

Du lernst, dass ein Programm eine Liste von Befehlen ist, die der Computer **der Reihe nach** ausführt.

Python	Scratch	Was passiert
`forward(100)`	gehe 100er-Schritt	100 Schritte vorwärts
`back(100)`	gehe -100er-Schritt	100 Schritte rückwärts
`left(90)`	drehe ↺ um 90 Grad	90 Grad nach links drehen
`right(90)`	drehe ↻ um 90 Grad	90 Grad nach rechts drehen

1 Vorhersagen PREDICT

```
from turtle import *

forward(100)
left(90)
forward(100)
left(90)
forward(100)
left(90)
forward(100)
left(90)

done()
```

MEINE VORHERSAGE

Was zeichnet die Schildkröte? Zeichne deine Vermutung in das Feld. Wo hört sie auf, und in welche Richtung schaut sie am Ende?

2 Ausführen RUN

Tippe das Programm ab, speichere es als `station1.py` und starte es.

Stimmt deine Vorhersage? ☐ ja ☐ teilweise ☐ nein

3 Untersuchen INVESTIGATE

1. Setze vor die erste Zeile `from turtle import *` ein `#`. Was meldet Thonny jetzt? Wozu ist die Zeile also da?

2. Ersetze jedes `left` durch `right`. Was ist anders?

3. Schreibe `Forward(100)` mit großem F. Lies die Fehlermeldung: In welcher Zeile findet Thonny den Fehler?

4. Lösche das letzte `left(90)`. Siehst du einen Unterschied im Bild? Achte auf die Schildkröte.

MERKE

Mit `#` machst du eine Zeile zum **Kommentar**. Python überspringt sie. So kannst du Notizen in dein Programm schreiben oder Befehle kurz ausschalten.

4 Verändern MODIFY

GRUNDAUFGABE

★ Verändere das Programm so, dass ein Rechteck entsteht: 200 Schritte breit und 100 Schritte hoch.

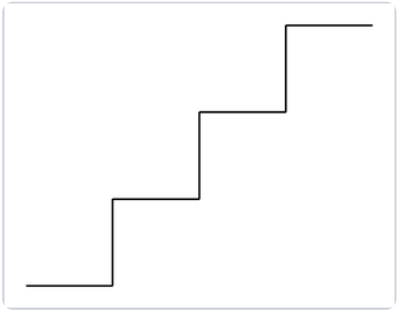
GRUNDAUFGABE

★ Zeichne ein Dreieck, bei dem alle Seiten 100 Schritte lang sind. Welchen Drehwinkel brauchst du? *Tipp: Wenn die Schildkröte wieder am Start ist, hat sie sich einmal ganz herumgedreht, also um 360 Grad.*

5 Erfinden MAKE

WEITERDENKEN

★★ Zeichne diese Treppe. Jede Stufe ist 50 Schritte breit und 50 Schritte hoch.

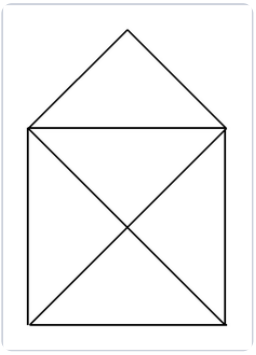


WEITERDENKEN

★★ Schreibe mit der Schildkröte den ersten Buchstaben deines Namens. Mit `back()` kannst du ein Stück zurückfahren, ohne dich umzudrehen.

KNOBELAUFGABE

★★★ Zeichne das **Haus vom Nikolaus** in einem Zug, ohne eine Linie doppelt zu fahren. *Tipp: Bei Seitenlänge 100 ist die schräge Linie quer durchs Quadrat etwa 141 Schritte lang, jede Dachseite etwa 71. Die Winkel sind 45, 90 und 135 Grad.*



2 STATION 2
Farben, Stift und Füllen

Du lernst, wie die Schildkröte ihren Stift hebt, die Farbe wechselt und Flächen ausmalt.

Python	Scratch	Was passiert
`penup()` / `pendown()`	hebe Malstift an / schalte Stift ein	Stift hoch oder runter
`pencolor("red")`	setze Stiftfarbe auf	Stiftfarbe ändern
`pensize(10)`	setze Stiftdicke auf 10	Stiftdicke ändern
`dot(50, "blue")`	–	Gefüllter Kreis, 50 breit
`hideturtle()`	verstecke dich	Schildkröte ausblenden

Farben schreibst du auf Englisch und in Anführungszeichen, zum Beispiel `"red"`, `"orange"`, `"gold"`, `"green"`, `"blue"`, `"purple"`, `"pink"`, `"brown"`, `"black"`, `"gray"`, `"skyblue"`, `"darkgreen"`.

1 Vorhersagen PREDICT

```
from turtle import *

pensize(10)
pencolor("brown")
left(90)
forward(150)

penup()
forward(40)
pendown()
dot(100, "red")

hideturtle()
done()
```

MEINE VORHERSAGE

Was für ein Bild entsteht? Zeichne es mit Farben in das Feld. Tipp: Es ist etwas Süßes.

2 Ausführen RUN

Speichere das Programm als `station2.py` und starte es.

Stimmt deine Vorhersage? ☐ ja ☐ teilweise ☐ nein

3 Untersuchen INVESTIGATE

1. Mache aus `penup()` und `pendown()` Kommentare mit `#`. Was ändert sich?

2. Warum kommt `left(90)` vor dem ersten `forward`? Was passiert ohne diese Zeile?

3. Verändere die Zahl in `dot(100, "red")`. Was gibt die Zahl an?

Neu: Flächen füllen

Alles, was die Schildkröte zwischen `begin_fill()` und `end_fill()` umrundet, wird ausgemalt:

```
from turtle import *

fillcolor("gold")
begin_fill()
forward(100)
left(90)
forward(100)
left(90)
forward(100)
left(90)
forward(100)
left(90)
end_fill()

done()
```

MERKE

`pencolor()` bestimmt die Farbe der Linie, `fillcolor()` die Farbe der Fläche. Mit `circle(50)` fährt die Schildkröte einen Kreis mit Radius 50.

4 Verändern MODIFY

GRUNDAUFGABE

★ Mach aus dem Lutscher eine Blume: grüner Stiel und gelbe Blüte.

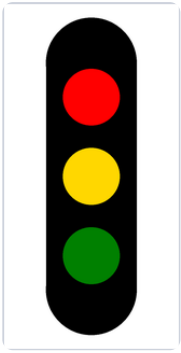
GRUNDAUFGABE

★ Ändere das Füll-Beispiel so, dass ein blaues Quadrat mit dicker roter Umrandung entsteht.

5 Erfinden MAKE

WEITERDENKEN

★★ Zeichne eine Ampel. Das schwarze Rechteck ist einfach eine sehr dicke Linie: `pensize(80)`. Die Lichter sind Punkte mit `dot(50, ...)`.



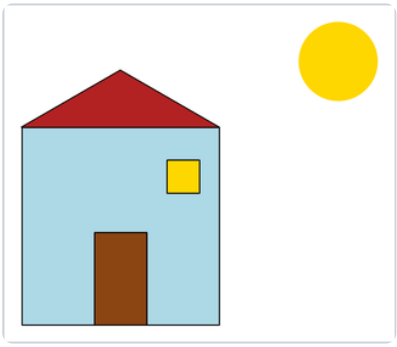
WEITERDENKEN

★★ Zeichne die deutsche Flagge mit drei gefüllten Rechtecken.



KNOBELAUFGABE

★★★ Zeichne ein Haus mit Dach, Tür, Fenster und Sonne. *Tipp: Mit `goto(x, y)` springt die Schildkröte direkt an einen Punkt, zum Beispiel `goto(0, 150)`. Die Mitte des Fensters ist der Punkt (0, 0). Hebe vorher den Stift an!*



3 STATION 3 Wiederholen mit Schleifen

Du lernst, wie du Befehle mehrmals ausführen lässt, ohne sie mehrmals hinzuschreiben. In Scratch heißt das **wiederhole 4 mal**, in Python `for i in range(4):`.

1 Vorhersagen PREDICT

```
from turtle import *

for i in range(4):
    forward(100)
    left(90)

done()
```

MEINE VORHERSAGE

Vergleiche mit deinem Programm aus Station 1. Was zeichnet dieses Programm? Zeichne es in das Feld. Wie viele Zeilen spart es?

2 Ausführen RUN

Speichere als `station3.py` und starte es.

Stimmt deine Vorhersage? ☐ ja ☐ teilweise ☐ nein

3 Untersuchen INVESTIGATE

1. Lösche den Doppelpunkt `:` am Ende der `for`-Zeile. Was meldet Thonny?

2. Rücke `left(90)` aus, sodass es ganz links steht. Was passiert jetzt und warum?

3. Ersetze `4` durch `3` und `90` durch `120`. Was entsteht?

4. Probiere 5 Ecken. Welchen Winkel brauchst du? Findest du eine Regel für jede Eckenzahl?

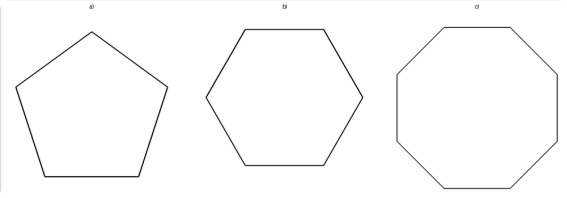
MERKE

Alles, was **eingerückt** unter der `for`-Zeile steht, wird wiederholt. Die Zeile endet immer mit einem **Doppelpunkt**. Bei einem regelmäßigen Vieleck dreht sich die Schildkröte um `360 / Ecken` Grad.

4 Verändern MODIFY

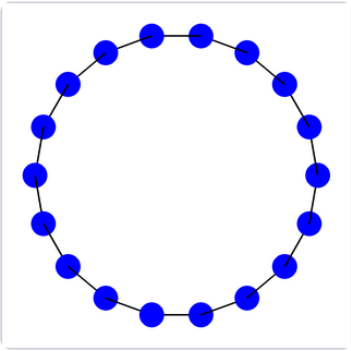
GRUNDAUFGABE

★ Zeichne ein Fünfeck (a), ein Sechseck (b) und ein Achteck (c). Ändere dazu nur die Zahlen.



GRUNDAUFGABE

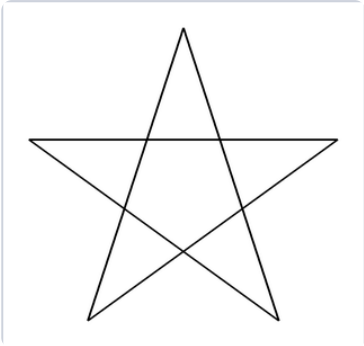
★ Zeichne die Perlenkette aus 18 Perlen. Die Schildkröte geht jeweils ein Stück vorwärts, malt einen Punkt mit `dot()` und dreht sich um 20 Grad.



5 Erfinden MAKE

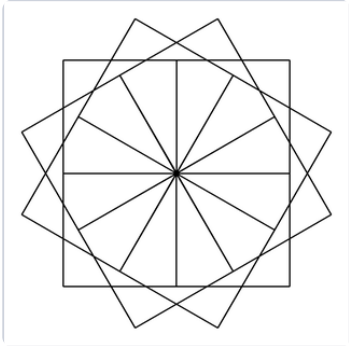
WEITERDENKEN

★★ Zeichne einen fünfzackigen Stern. Die Schildkröte dreht sich an jeder Spitze um 144 Grad. Warum gerade 144? Kannst du einen Stern mit mehr Zacken finden?



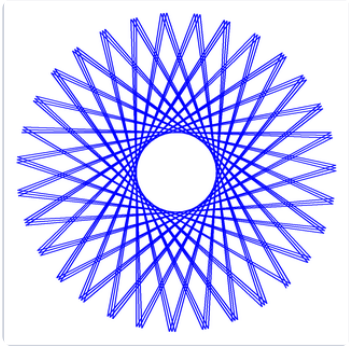
WEITERDENKEN

★★ Schleife in der Schleife: Eine Schleife zeichnet ein Quadrat. Eine zweite Schleife außen herum wiederholt das 12-mal und dreht dazwischen um 30 Grad. Achte genau auf die Einrückung!



KNOBELAUFGABE

★★★ Zeichne mit nur zwei Befehlen in der Schleife, `forward(300)` und `right(151)`, und 92 Wiederholungen dieses Kunstwerk. Verändere danach die Zahlen und erfinde eigene Muster. Mit `speed(0)` am Anfang zeichnet die Schildkröte ganz schnell.



STATION 4

4 Variablen

Du lernst, wie sich das Programm eine Zahl merkt und sie unterwegs verändert. In Scratch sind das die Blöcke **setze Variable auf** und **ändere Variable um**.

Python	Scratch
`seite = 20`	setze seite auf 20
`seite = seite + 10`	ändere seite um 10
`forward(seite)`	gehe (seite)er-Schritt

1 Vorhersagen PREDICT

```python
from turtle import *

seite = 20
for i in range(15):
    forward(seite)
    left(90)
    seite = seite + 10

done()
```

MEINE VORHERSAGE

Wie lang ist die erste Linie, wie lang die zweite und die dritte? Skizziere die ersten fünf Linien in das Feld. Was entsteht am Ende?

2 Ausführen RUN

Speichere als `station4.py` und starte es. Öffne vorher **Ansicht → Variablen**, dann siehst du, wie `seite` wächst.

Stimmt deine Vorhersage? ☐ ja ☐ teilweise ☐ nein

3 Untersuchen INVESTIGATE

1. Fülle die Tabelle aus, bevor du nachschaust:

Durchgang	Wert von `seite` beim Zeichnen
1	20
2	
3	
15	

2. Schiebe die Zeile `seite = seite + 10` ganz nach oben in die Schleife, vor `forward`. Was ändert sich?

3. Ersetze `left(90)` durch `left(91)`. Was passiert mit der Spirale?

4. `i` ist auch eine Variable! Schreibe `print(i)` in die Schleife. Welche Zahlen erscheinen unten in Thonny?

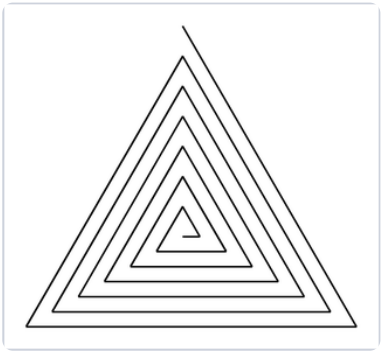
MERKE

Eine **Variable** ist ein Name für einen Wert. Bei `seite = seite + 10` rechnet Python zuerst rechts vom `=` und speichert das Ergebnis dann in `seite`. Die Variable `i` zählt in der Schleife mit: 0, 1, 2, 3 …

4 Verändern MODIFY

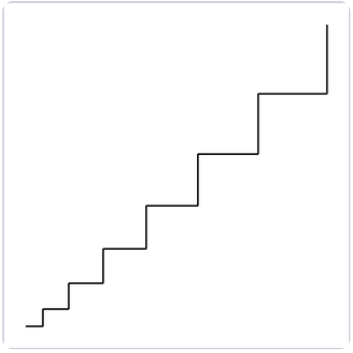
GRUNDAUFGABE

★ Ändere Startwert, Zuwachs und Winkel so, dass eine Dreiecksspirale entsteht.



GRUNDAUFGABE

★ Zeichne eine Treppe mit 7 Stufen, bei der jede Stufe 5 Schritte größer ist als die vorherige.



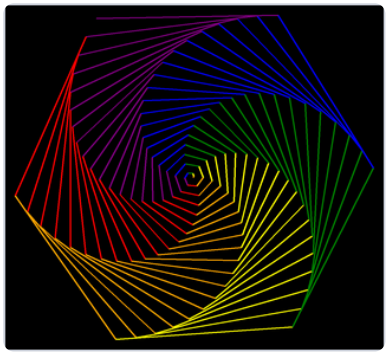
5 Erfinden MAKE

WEITERDENKEN

★★ Erfinde eine eigene Spirale. Benutze statt einer eigenen Variable die Zählvariable: `forward(i * 3)`.

KNOBELAUFGABE

★★★ Die Regenbogenspirale: Hier kommt eine **Liste** mit Farben vor. Mit `farben[0]` bekommst du die erste Farbe, mit `farben[1]` die zweite. `i % 6` ist der Rest beim Teilen durch 6 und zählt immer 0, 1, 2, 3, 4, 5, 0, 1 … Schreibe das Programm so zu Ende, dass dieses Bild entsteht. *Tipp: Drehwinkel 59 Grad, Hintergrund mit `bgcolor("black")`.*



```
farben = ["red", "orange", "yellow", "green", "blue", "purple"]
for i in range(120):
    pencolor(farben[i % 6])
    ...
```

STATION 5

5 Eigene Befehle mit Funktionen

Du lernst, wie du der Schildkröte neue Befehle beibringst. In Scratch machst du das unter **Meine Blöcke → Neuer Block**, mit einer Eingabe wie „größe“.

1 Vorhersagen PREDICT

```
from turtle import *

def quadrat(groesse):
    for i in range(4):
        forward(groesse)
        left(90)

quadrat(50)
quadrat(100)
quadrat(150)

done()
```

MEINE VORHERSAGE

Wie viele Quadrate siehst du am Ende? Wie groß sind sie, und wo liegen sie? Zeichne sie in das Feld.

2 Ausführen RUN

Speichere als `station5.py` und starte es.

Stimmt deine Vorhersage? ☐ ja ☐ teilweise ☐ nein

3 Untersuchen INVESTIGATE

1. Mache aus den drei Zeilen `quadrat(...)` Kommentare. Was zeichnet die Schildkröte jetzt? Was bedeutet das für `def`?

2. Welchen Wert hat `groesse` beim zweiten Aufruf?

3. Rufe `quadrat()` ohne Zahl auf. Was sagt die Fehlermeldung?

4. Schreibe `quadrat(30)` **über** die `def`-Zeile. Was passiert und warum?

MERKE

Mit `def` **definierst** du einen neuen Befehl, eine **Funktion**. Sie wird erst ausgeführt, wenn du sie **aufrufst**. Die Werte in der Klammer heißen **Parameter**. Eine Funktion muss oben stehen, bevor du sie benutzt.

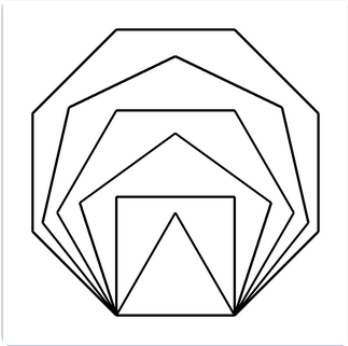
4 Verändern MODIFY

GRUNDAUFGABE

★ Schreibe eine Funktion `dreieck(groesse)` und zeichne drei verschieden große Dreiecke.

GRUNDAUFGABE

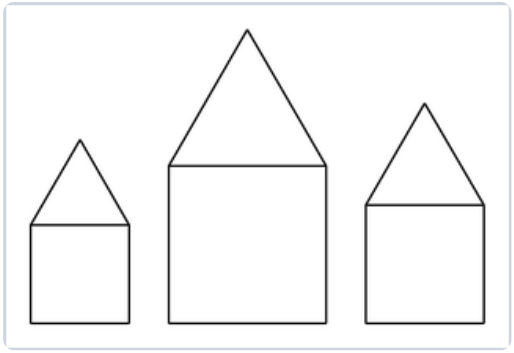
★ Erweitere die Funktion: `vieleck(ecken, groesse)` hat zwei Parameter. Zeichne damit alle Vielecke von 3 bis 8 Ecken. *Tipp: Eine Schleife* `for ecken in range(3, 9):` *zählt von 3 bis 8.*



5 Erfinden MAKE

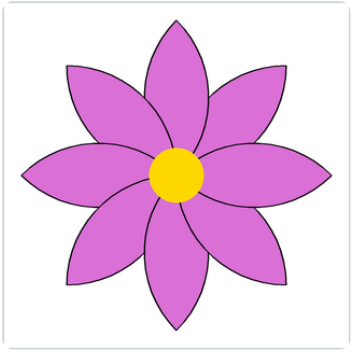
WEITERDENKEN

★★ Baue aus `quadrat` und `dreieck` eine Funktion `haus(groesse)`. Zeichne damit eine Straße mit drei verschieden großen Häusern. Schreibe dir dazu eine Funktion `lauf(strecke)`, die mit gehobenem Stift ein Stück weiterfährt.



KNOBELAUFGABE

★★★ Schreibe eine Funktion `blatt(groesse)`, die ein gefülltes Blütenblatt aus zwei Kreisbögen malt, und eine Funktion `blume(blaetter, groesse)`, die `blatt` mehrmals aufruft. *Tipp:* `circle(80, 90)` *fährt einen Viertelkreis mit Radius 80.*



STATION 6

6 Zufall und Entscheidungen

Du lernst, wie dein Programm würfelt und je nach Ergebnis etwas anderes tut. In Scratch sind das **Zufallszahl von 1 bis 10** und **falls … dann … sonst**.

Das Turtle-Fenster ist ein Koordinatensystem: Die Mitte ist (0, 0). `x` wächst nach rechts, `y` wächst nach oben. Mit `goto(x, y)` springt die Schildkröte an diesen Punkt.

1 Vorhersagen PREDICT

```
from turtle import *
from random import randint

speed(0)
penup()
for i in range(50):
    x = randint(-200, 200)
    y = randint(-200, 200)
    goto(x, y)
    if y > 0:
        dot(20, "blue")
    else:
        dot(20, "green")

hideturtle()
done()
```

MEINE VORHERSAGE

Wo landen die blauen Punkte, wo die grünen? Zeichne es in das Feld. Sieht das Bild jedes Mal gleich aus?

2 Ausführen RUN

Speichere als `station6.py` und starte das Programm zweimal hintereinander.

Stimmt deine Vorhersage? ☐ ja ☐ teilweise ☐ nein

3 Untersuchen INVESTIGATE

1. Was macht `randint(-200, 200)`? Lass dir mit `print(x, y)` die Zahlen anzeigen.

2. Ersetze `y > 0` durch `x > 0`. Was ändert sich?

3. Warum steht `penup()` vor der Schleife? Was passiert ohne?

4. Teile das Bild in vier Viertel mit vier Farben. *Tipp: Mit `and` kannst du zwei Bedingungen verbinden: `if x > 0 and y > 0:`. Zwischen `if` und `else` darfst du beliebig viele `elif` einfügen.*

MERKE

`if` prüft eine Bedingung. Ist sie wahr, wird der eingerückte Teil darunter ausgeführt, sonst der Teil unter `else`. Vergleiche schreibst du mit `>`, `<`, `==` (gleich) und `!=` (ungleich). Achtung: `=` speichert einen Wert, `==` vergleicht!

4 Verändern MODIFY

GRUNDAUFGABE

★ Die Punkte sollen zufällig groß sein, zwischen 5 und 40.

GRUNDAUFGABE

★ Erzeuge 200 Punkte statt 50 und wähle eigene Farben.

5 Erfinden MAKE

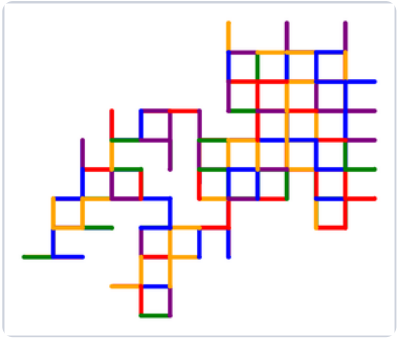
WEITERDENKEN

★★ Zeichne einen Sternenhimmel: Die meisten Sterne sind kleine weiße Punkte. Ungefähr jeder zehnte Stern soll größer und gelb sein. *Tipp: `if randint(1, 10) == 1:` ist ungefähr bei jedem zehnten Mal wahr.*



KNOBELAUFGABE

★★★ Der Zufallswanderer: Die Schildkröte geht 300-mal 20 Schritte, dreht sich davor aber zufällig um 0, 90, 180 oder 270 Grad und wählt eine zufällige Farbe. Sie darf ihr Gehege nicht verlassen: Wenn `xcor()` oder `ycor()` größer als 150 oder kleiner als -150 wird, geht sie sofort wieder zurück.



STATION 7

7 Dein Abschlussprojekt

Jetzt erfindest du ein eigenes Programm und zeigst es am Ende den anderen. Wähle eine der drei Ideen oder denk dir selbst eine aus.

So gehst du vor

1. **Planen:** Zeichne dein Bild zuerst in das Planungsfeld unten. Überlege, welche Teile sich wiederholen: Daraus werden Funktionen und Schleifen.
2. **In kleinen Schritten bauen:** Schreibe immer nur ein paar Zeilen und teste sie sofort.
3. **Testen lassen:** Lass jemand anderen dein Programm ausprobieren und frage nach einem Tipp.
4. **Vorstellen:** Erkläre den anderen eine Stelle in deinem Code, auf die du stolz bist.

Mein Plan

Meine Idee:

PLANUNGSFELD: SO SOLL MEIN BILD AUSSEHEN

Diese Teile wiederholen sich (daraus werden Schleifen und Funktionen):

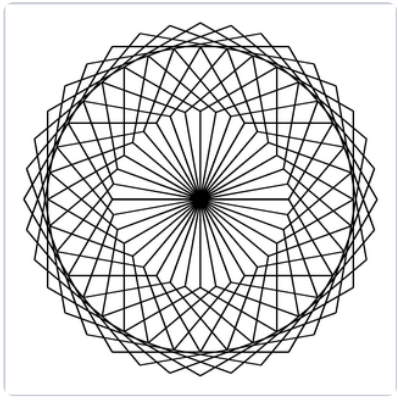
Idee A: Der Muster-Generator ★★

Dein Programm fragt nach Zahlen und zeichnet damit ein Muster. Den Anfang bekommst du hier. Frage noch nach weiteren Zahlen, zum Beispiel nach der Größe oder dem Drehwinkel.

```python
from turtle import *

ecken = int(numinput("Muster", "Wie viele Ecken?", 6))
speed(0)
for i in range(36):
    for j in range(ecken):
        forward(60)
        left(360 / ecken)
    left(10)

done()
```



Idee B: Die ferngesteuerte Schildkröte ★★★

Mit den Pfeiltasten steuerst du die Schildkröte und malst damit. Klicke nach dem Start einmal ins Turtle-Fenster, damit es die Tasten bemerkt. Baue es weiter aus: Mit der Taste `"space"` wechselt die Farbe, mit `"u"` hebst du den Stift an und mit `"d"` senkst du ihn wieder.

```python
from turtle import *

def hoch():
    setheading(90)
    forward(10)

def rechts():
    setheading(0)
    forward(10)

shape("turtle")
onkey(hoch, "Up")
onkey(rechts, "Right")
listen()
done()
```

Idee C: Mein Wimmelbild ★★

Zeichne eine ganze Szene, zum Beispiel eine Stadt bei Nacht, einen Garten oder eine Unterwasserwelt. Nutze deine Funktionen aus den anderen Stationen wieder, etwa `haus()`, `blume()` oder den Sternenhimmel.

Checkliste: Das steckt in meinem Projekt

- [ ] eine Schleife
- [ ] eine Variable
- [ ] eine eigene Funktion mit Parameter
- [ ] eine Entscheidung mit `if` oder Zufall mit `randint`
- [ ] Kommentare mit `#`, die erklären, was mein Code macht

Spickzettel: alle Befehle auf einen Blick

Jedes Programm beginnt mit `from turtle import *` und endet mit `done()`.

Befehl	Was er macht	Station
`forward(100)` / `back(100)`	100 Schritte vor / zurück	1
`left(90)` / `right(90)`	um 90 Grad nach links / rechts drehen	1
`# Text`	Kommentar, wird nicht ausgeführt	1
`penup()` / `pendown()`	Stift heben / senken	2
`pencolor("red")`	Stiftfarbe	2
`pensize(5)`	Stiftdicke	2
`fillcolor("gold")`	Füllfarbe	2
`begin_fill()` … `end_fill()`	umrundete Fläche ausmalen	2
`dot(30, "blue")`	gefüllter Punkt mit 30 Durchmesser	2
`circle(50)` / `circle(50, 90)`	Kreis / Kreisbogen mit Radius 50	2, 5
`goto(x, y)`	zum Punkt (x, y) springen	2, 6
`hideturtle()`	Schildkröte ausblenden	2
`speed(0)`	so schnell wie möglich zeichnen	3
`for i in range(4):`	eingerückten Block 4-mal wiederholen	3
`bgcolor("black")`	Hintergrundfarbe	4
`seite = 20`	Variable anlegen	4
`seite = seite + 10`	Variable um 10 erhöhen	4
`print(seite)`	Wert unten in Thonny anzeigen	4
`def name(parameter):`	eigene Funktion definieren	5
`from random import randint`	Zufall einschalten	6
`randint(1, 6)`	Zufallszahl von 1 bis 6	6
`if …:` / `elif …:` / `else:`	Entscheidung	6
`xcor()` / `ycor()`	aktuelle Position der Schildkröte	6

Häufige Fehler

Thonny meldet	Was meistens fehlt
`NameError: name 'Forward' is not defined`	Tippfehler oder falsche Groß-/Kleinschreibung
`NameError` bei jedem Turtle-Befehl	`from turtle import *` oben vergessen
`SyntaxError: expected ':'`	Doppelpunkt am Ende von `for`, `if`, `def`
`IndentationError`	Einrückung passt nicht: immer gleich weit einrücken
`SyntaxError: '(' was never closed`	eine Klammer wurde nicht geschlossen
Fenster blitzt nur kurz auf	`done()` am Ende vergessen
`AttributeError` oder „partially initialized module“	deine Datei heißt `turtle.py`: umbenennen!