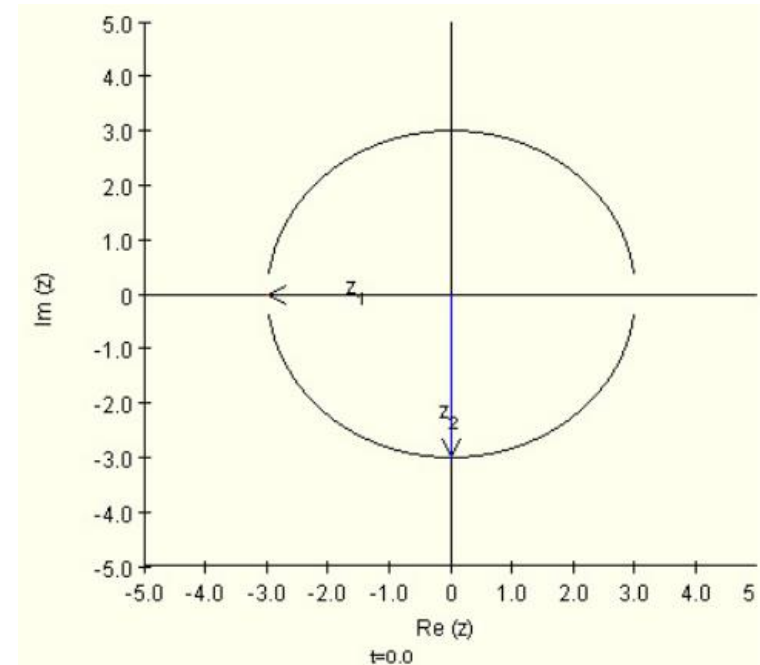
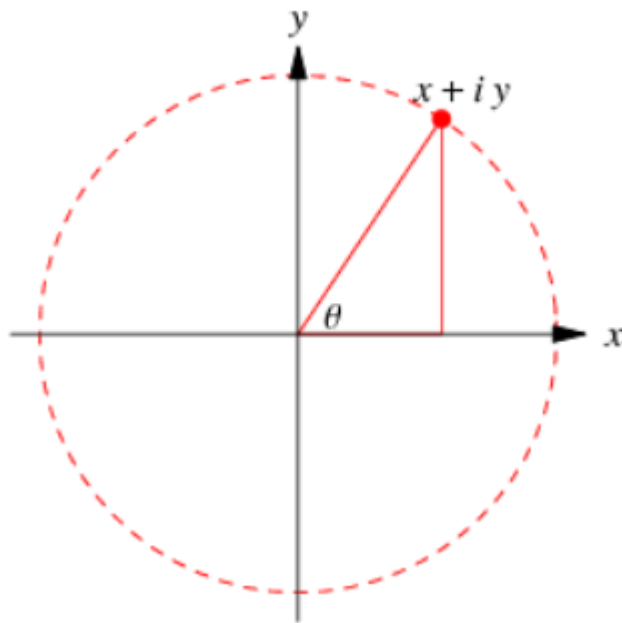


# Zu Beginn: Komplexe Zahlen

Eine Besonderheit der Programmiersprache Python ist, dass sie ohne weitere Anpassungen mit komplexen Zahlen arbeiten kann.

Komplexe Zahlen spielen in der Naturwissenschaft eine wichtige Rolle. Mit ihrer Hilfe können Punkte in einer Zahlenebene leicht dargestellt werden.

Diese Zahlenebene besteht aus einer  $x$ - und einer  $iy$ -Achse. Jede komplexe Zahl besitzt dadurch zwei Anteile: Einen Realteil und einen Imaginärteil.



In unserem Beispiel nennen wir die komplexen Zahlen  $c$  und schreiben sie z.B. als  $c = 0.33 + 0.33j$ . In der Mathematik wird als Einheit für den imaginären Teil als „ $i$ “ benutzt, Python nutzt dafür das „ $j$ “, wie es in den Ingenieurwissenschaften üblich ist.

Somit kann eine komplexe Zahl als Summe eines Real- und eines Imaginärteils zusammengefasst werden.

## Wozu dient das Programm?

- Mit Hilfe des Programms soll grafisch dargestellt werden, ob eine komplexe Zahl  $c$  zur Mandelbrot-Menge gehört oder nicht.
- Annahme: Unsere komplexe Zahl  $c$  gehört dann zur Mandelbrotmenge, wenn die Funktion des Typs von  $f(z) := z * z + c$  bei einem Startwert von  $z$  mit  $z_0 = 0$  nicht divergiert.
- Divergiert bedeutet: Der Betrag eines Folgenglieds wird während der Iteration größer als  $r = 2$ .
- Bleibt der Betrag eines Folgenglieds während der Iteration kleiner als  $r=2$ , gehört die Komplexe Zahl  $c$  zur Mandelbrotmenge. Der Betrag konvergiert gegen einen Grenzwert.
- Die Mandelbrotmenge und die dazugehörige Julia-Menge werden mit Funktionen des Typs  $f(z) := z * z + c$  erzeugt:

```
2 def f(z):  
3     return z * z + c
```

- Wir gehen davon aus, dass  $c = 0.33 + 0.33j$  zu der Mandelbrotmenge gehört.  $c = 0.36 + 0.36j$  gehört nicht zur Mandelbrotmenge.
- Die grafische Ausgabe für  $c = 0.33 + 0.33j$  erfolgt in blauer Farbe, für  $c = 0.36 + 0.36j$  in roter Farbe.

```
11 if isMandelbrot:  
12     c = 0.33 + 0.33j  
13     setColor("blue")  
14  
15 else:  
16     c = 0.36 + 0.36j  
17     setColor("red")
```

- Für  $c = 0.33 + 0.33j$  ist zu erwarten, dass  $z$  nicht größer als 2 und divergieren wird.
  - Folge: Bei der Iteration der Gleichung bleiben die Werte von  $z$  in einem beschränktem Gebiet liegen („Gefangene“).
- Für  $c = 0.36 + 0.36j$  ist zu erwarten, dass  $z$  größer als 2 wird und konvergieren wird.
  - Folge: Bei der Iteration dieser Gleichung streben einige Werte von  $z$  unter grafischer Iteration ins Unendliche.
- Der Startpunkt ist  $z = 0j$

```

18
19 title("Mandelbrot-Iteration mit: c = " + str(c))
20
21 move(c)
22 fillCircle(0.001)
23
24 z = 0j
25 while True:
26     if z == 0:
27         move(z)
28     else:
29         draw(z)
30     z = f(z)
31     print(z)
32     print(z.real)
33     print(z.imag)
34     delay(500)

```

- Für jede Stufe  $n$  der Iteration wird in der Konsole der Wert von  $z$  ausgegeben.
- Es wird solange iteriert, bis das Programm abgebrochen wird.

## Konkretes Beispiel: Gehört $c = 0.33 + 0.33j$ zur Mandelbrotmenge?

- Annahme: Die komplexe Zahl  $c = 0.33 + 0.33j$  gehört zur Mandelbrotmenge.
- Bedingung: Ein Folgeglied wird während der Iteration *nicht größer* als 2.
- Die eingezeichneten Punkte verlassen die Gefangenenmenge nicht, sie gehören zur Julia-Menge.
- Der Radius  $r = 0.513768$  hat damit den Wert von  $r = 2$  nicht überschritten: Das System konvergiert gegen einen Grenzwert.
- Es wird bei der Durchführung nach einer Iterationsdauer von 150 s folgende Grafik und Werte ausgegeben:

**<module>**  
Julia\_Mandelbrot\_T3.py [39]

**isMandelbrot** **True**

**z** **0.15085696797+0.491120975647j**

**delay** **function**

**draw** **function**

**drawGrid** **function**

**fillCircle** **function**

**makeGPanel** **function**

**move** **function**

**setColor** **function**

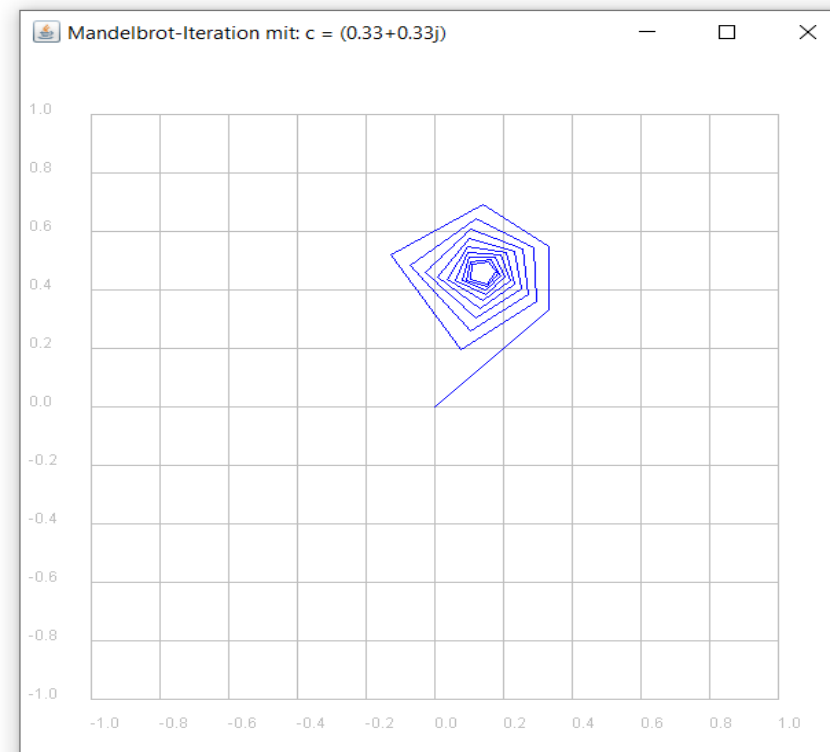
**title** **function**

**setColor** [759]

**args** **blue**

**\_toColor** [1066]

**args** **blue**



**z** **0.15085696797+0.491120975647j**

**delay** **complex**

**draw** **Re** **0.15085696797040424**

**draw** **Im** **0.49112097564699303**

**rd** **r** **0.5137680775463555**

**fillCircle** **function**

Ausgabe der Werte von z nach 25 Durchgängen für  $c = 0.33 + 0.33j$ :

#	Werte	#	Werte
1	(0.33+0.33j)	14	(-0.0275526829731+0.458656616768j)
	0.33		-0.0275526829731
	0.33		0.458656616768
2	(0.33+0.5478j)	15	(0.120393258234+0.304725559289j)
	0.33		0.120393258234
	0.5478		0.304725559289
3	(0.13881516+0.691548j)	16	(0.251636870144+0.4033738059j)
	0.13881516		0.251636870144
	0.691548		0.4033738059
4	(-0.128968987658+0.521994692535j)	17	(0.23061068713+0.53300744403j)
	-0.128968987658		0.23061068713
	0.521994692535		0.53300744403
5	(0.0741545407425+0.195357745882j)	18	(0.0990843536274+0.575834425826j)
	0.0741545407425		0.0990843536274
	0.195357745882		0.575834425826
6	(0.297334247037+0.358973327853j)	18	(0.00823242316781+0.444112363759j)
	0.297334247037		0.00823242316781
	0.358973327853		0.444112363759
7	(0.289545804351+0.543470128287j)	19	(0.132831981148+0.337312241825j)
	0.289545804351		0.132831981148
	0.543470128287		0.337312241825
8	(0.118476992477+0.644718990871j)	20	(0.233864786731+0.419611706694j)
	0.118476992477		0.233864786731
	0.644718990871		0.419611706694
9	(-0.0716257794437+0.482768734063j)	21	(0.208618754078+0.526264804591j)
	-0.0716257794437		0.208618754078
	0.482768734063		0.526264804591
10	(0.102064601692+0.260842626263j)	22	(0.0965671400014+0.549577415698j)
	0.102064601692		0.0965671400014
	0.260842626263		0.549577415698
11	(0.272378307243+0.383245597508j)	23	(0.037289876683+0.436142238487j)
	0.272378307243		0.037289876683
	0.383245597508		0.436142238487
12	(0.257312754247+0.538775574215j)	24	(0.141170482711+0.362527380579j)
	0.257312754247		0.141170482711
	0.538775574215		0.362527380579
13	(0.105930734128+0.607267653845j)	25	(0.218503003519+0.432356330625j)
	0.105930734128		0.218503003519
	0.607267653845		0.432356330625

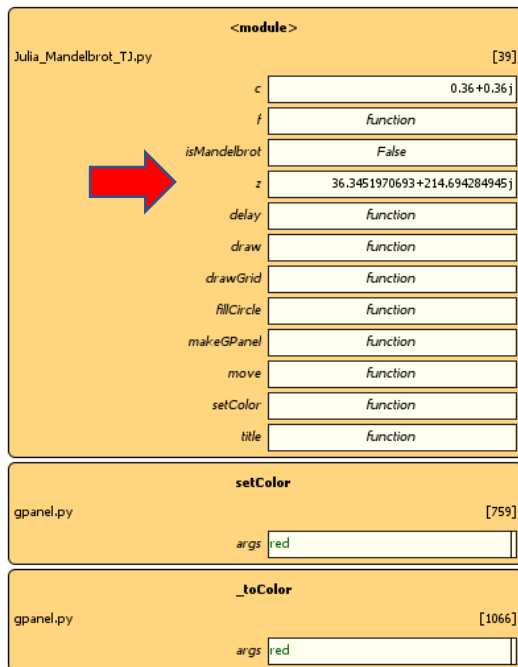
Ergebnis:

c ist Element der Mandelbrotmenge!

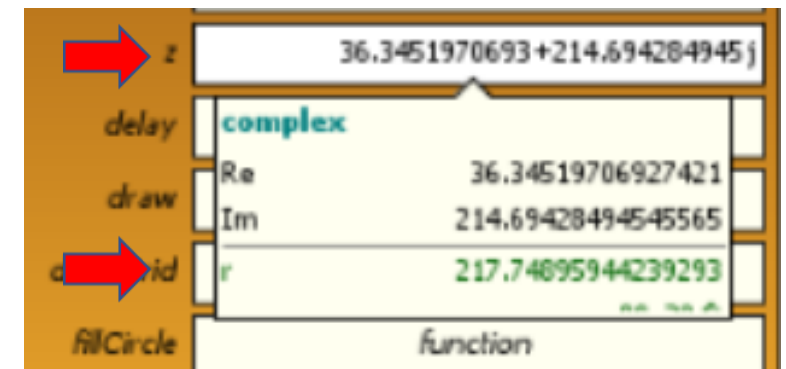
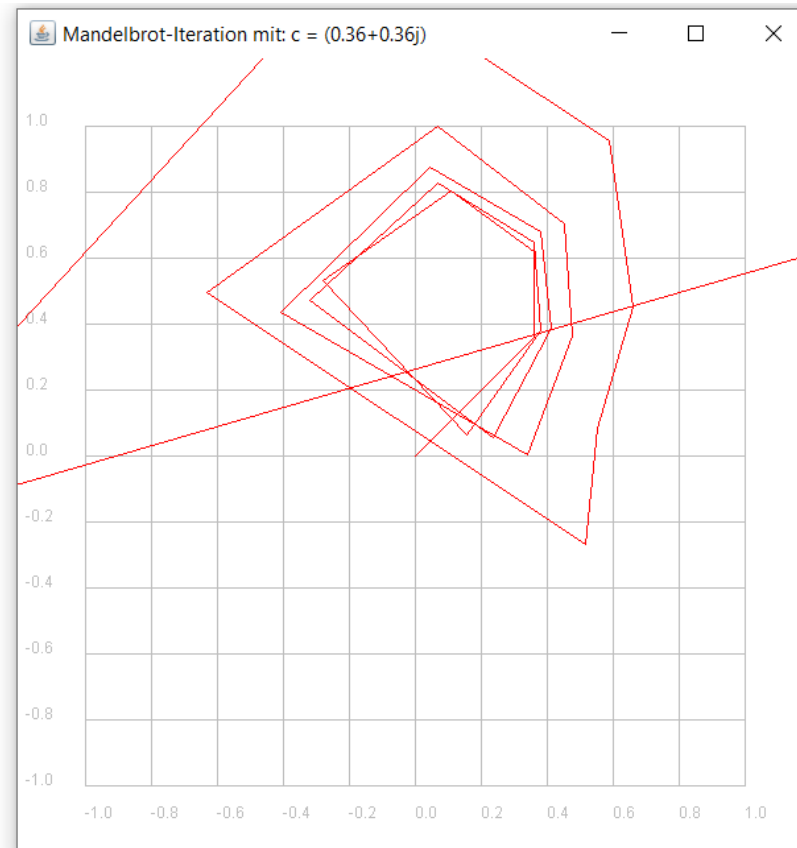
Ein Anfangspunkt z liegt innerhalb der Mandelbrotmenge („Gefangenenmenge“), wenn seine Folge von Iterierten unter Iteration der Funktion  $f(z) := z^2 + c$  nicht ins Unendliche entkommt.

## Konkretes Beispiel: Gehört $c = 0.36 + 0.36j$ zur Mandelbrot-Menge?

- Annahme: Die komplexe Zahl  $c = 0.36 + 0.36j$  gehört nicht zur Mandelbrotmenge.
- Ein Folgeglied wird während der Iteration *größer* als 2.
- Die eingezeichneten Werte fliehen nach Unendlich. Die Julia-Mengen, die zu diesen Werten gehören, sind unzusammenhängend.
- Der Radius beträgt  $r = 217.789594$  und hat damit den Wert von  $r = 2$  überschritten. Das System divergiert.
- Es wird bei der Durchführung nach einer Iterationsdauer von 150 s folgende Grafik und Werte ausgegeben:



NetLogo patch showing the 'isMandelbrot' function being called. The patch includes a 'Julia\_Mandelbrot\_TJ.py' module and a 'gpanel.py' patch. The 'isMandelbrot' function is called with the argument 'z' (36.3451970693+214.694284945j). The 'setColor' and '\_toColor' functions are also visible.



NetLogo patch showing the 'z' variable being updated. The 'z' variable is set to the complex number '36.3451970693+214.694284945j'. The 'delay' and 'draw' functions are also visible.

Ausgabe der Werte von z nach 25 Durchgängen für  $c = 0.36 + 0.36j$

#	Werte	#	Werte
1	(0.36+0.36j)	14	(-0.408630205031+0.434135484998j)
	0.36		-0.408630205031
	0.36		0.434135484998
2	(0.36+0.6192j)	15	(0.33850502513+0.00519825550766j)
	0.36		0.33850502513
	0.6192		0.00519825550766
3	(0.10619136+0.805824j)	16	(0.474558630178+0.363519271223j)
	0.10619136		0.474558630178
	0.805824		0.363519271223
4	(-0.278075714037+0.531143092961j)	17	(0.453059632926+0.705022414789j)
	-0.278075714037		0.453059632926
	0.531143092961		0.705022414789
5	(0.155213117537+0.0646040103376j)	18	(0.068206425632+0.998834392898j)
	0.155213117537		0.068206425632
	0.0646040103376		0.998834392898
6	(0.379917433704+0.3800547797j)	18	(-0.633018027938+0.496253847476j)
	0.379917433704		-0.633018027938
	0.3800547797		0.496253847476
7	(0.359895620859+0.648778873141j)	19	(0.51444394256-0.268275263771j)
	0.359895620859		0.51444394256
	0.648778873141		-0.268275263771
8	(0.06861083168+0.826985350699j)	20	(0.552680952885+0.0839748312282j)
	0.06861083168		0.552680952885
	0.826985350699		0.0839748312282
9	(-0.319197324047+0.473480305397j)	21	(0.658404463402+0.452822579483j)
	-0.319197324047		0.658404463402
	0.473480305397		0.452822579483
10	(0.23770333208+0.0577327070565j)	22	(0.588448148938+0.956280814922j)
	0.23770333208		0.588448148938
	0.0577327070565		0.956280814922
11	(0.413169808618+0.387446513675j)	23	(-0.208201772999+1.48544335081j)
	0.413169808618		-0.208201772999
	0.387446513675		148.544.335.081
12	(0.380594489795+0.680162403809j)	24	(-1.80319397019-0.258543878657j)
	0.380594489795		-180.319.397.019
	0.680162403809		-0.258543878657
13	(0.0422312701066+0.87773212611j)	25	(3.54466355694+1.29240952605j)
	0.0422312701066		354.466.355.694
	0.87773212611		129.240.952.605

Ergebnis: c ist nicht Element der Mandelbrotmenge!

Ein Anfangspunkt z liegt außerhalb der Mandelbrotmenge („Gefangenenmenge“), wenn seine Folge von Iterierten unter Iteration der Funktion  $f(z) := z^2 + c$  ins Unendliche entkommt.

# Programmcode

```
1 from gpanel import *
2 def f(z):
3     return z * z + c
4
5 makeGPanel(-1.2, 1.2, -1.2, 1.2)
6 title("Mandelbrot-Iteration: Gehört eine komplexe Zahl c zur Mandelbrot-Menge?")
7
8 drawGrid(-1, 1.0, -1, 1.0, 10, 10, "gray")
9
10 isMandelbrot = askYesNo("Soll der Wert c ein Element der Mandelbrotmenge sein?")
11 if isMandelbrot:
12     c = 0.33 + 0.33j
13     setColor("blue")
14
15 else:
16     c = 0.36+ 0.36j
17     setColor("red")
18
19 title("Mandelbrot-Iteration mit: c = " + str(c))
20
21 move(c)
22 fillCircle(0.001)
23
24 z = 0j
25 while True:
26     if z == 0:
27         move(z)
28     else:
29         draw(z)
30     z = f(z)
31     print(z)
32     print(z.real)
33     print(z.imag)
34     delay(500)
```