```
#%% md
# Komplexe Zahlen

## Vektorrotation
Komplexe Zahlen können genutzt werden, um Vektoren schnell und einfach zu rotieren. Dies wird in vielen Softwarebereichen benutzt, vor allem zum Beispiel in der Spieleindustrie.
#%% md
### Grundlagen
Um zu verstehen, wie ein Vektor mittels komplexer Zahlen rotiert werden kann, muss erst einmal verstanden werden, wieso Vektoren und komplexe Zahlen sich so ähnlich sind.
Die komplexe Zahl $x = a + b \times i$ kann so in einem kartesischen Koordinatensystem als Punkt $P$ in der Ebene dargestellt werden. Dieser Punkt $P$ hat den realen Anteil $a$ als x-Koordinate und den imaginären Anteil $b$ als y-Koordinate. $i^2$ ist in dem Fall -1.
Dieser Punkt $P$ kann jedoch auch als Vektor vom Ursprung zu diesem Punkt dargestellt werden. Der sogenannte Ortsvektor zum Punkt $P$.
#%% md
### kartesische vs. Polarkoordinaten
Wie bereits im vorhergehenden Abschnitt erwähnt, können komplexe Zahlen als ein Punkt im kartesischen Koordinatensystem dargestellt werden, mit sogenannten kartesischen Koordinaten.
Es gibt jedoch noch eine andere Schreibweise Vektoren, bzw. komplexe Zahlen darzustellen, welche unter anderem auch für die Vektorrotation benötigt wird. Die sogenannten Polarkoordinaten.
Der Gedanke hinter dieser Darstellung ist, dass man eine Zahl $r$ hat, welche den Betrag des Ortsvektors zum Punkt $P$ bzw. er komplexen Zahl $c$ angibt, und einen Winkel $\phi$, welcher den Winkel des Vektors zur Realachse, also der x-Achse angibt. Damit kann man ebenfalls den Ortsvektor bestimmen und damit auch die komplexe Zahl $c$.
#%% md
```

```
17 ### Rotation
18 Wir können dieses Wissen nun nutzen, um einen
   gegebenen Vektor im Realraum mittels einer
   imaginären Zahl $c$ um einen gegebenen Winkel $\
   omega$ zu rotieren.
19 #%%
20 from complex_jupyter import ComplexNumber
21 from vector_rotation_jupyter import Vector
22 from math import cos, sin, pi
23
24 def rotate(self, angle):
25     c = ComplexNumber(self.x, self.y)
26     rad = angle * (pi / 180)
27     # real * cos = real number.
28     # imag * sin = real number, because imag and
   sin each contain i (i^2 = -1).
29     # real * sin = imag number, because sin
   contains i.
30     # imag * cos = imag number, because imag
   contains i.
31     rotated_real = c.real * cos(rad) - c.imag * sin
   (rad)
32     rotated_imag = c.real * sin(rad) + c.imag * cos
   (rad)
33     rotated_c = ComplexNumber(rotated_real,
   rotated_imag)
34     return Vector(rotated_c.real, rotated_c.imag)
35 #%% md
36 # Footer
37 Projektbeteiligte:
38 * Gregor Gottschewski
39 * Leon Heiner
40 * Marvin Igrec
41 * Julian Schumacher
42 * Daniel Ziegler
```