

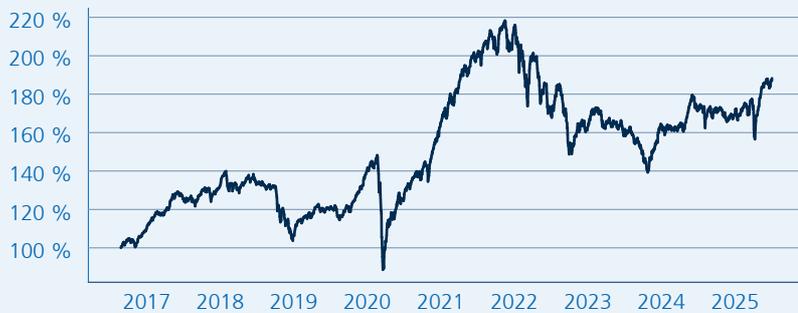
Mandarine Funds - Mandarine Europe Microcap R

Aktienfonds - ISIN: LU1303940784



WERTENTWICKLUNG IN DER VERGANGENHEIT*

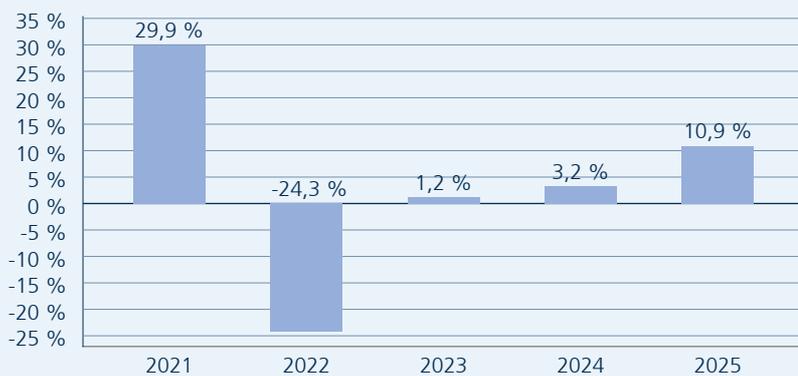
Indexierte Wertentwicklung



WERTENTWICKLUNG ÜBER VERSCHIEDENE ZEITRÄUME

1 Monat	3 Monate	6 Monate	lfd. Jahr	1 Jahr
1,41 %	10,75 %	9,43 %	10,93 %	8,81 %
3 Jahre p.a.	5 Jahre p.a.	10 Jahre p.a.	Seit Auflage p.a.	
3,70 %	7,99 %	-	7,40 %	
3 Jahre	5 Jahre	10 Jahre	Seit Auflage	
11,53 %	46,89 %	-	88,29 %	

WERTENTWICKLUNG PRO KALENDERJAHR



ROLLIERENDE 12-MONATS ENTWICKLUNG

03.07.21-03.07.22	03.07.22-03.07.23	03.07.23-03.07.24	03.07.24-03.07.25
-16,06 %	-4,60 %	7,45 %	8,81 %

*Historische Wertentwicklungen sind keine Garantie für eine ähnliche Entwicklung in der Zukunft. Diese ist nicht prognostizierbar. Die Berechnung der Wertentwicklung erfolgt gemäß BVI-Methode.

03. JULI 2025

Anlageziel

Der Mandarine Europe Microcap ist ein europäischer Aktienfonds, der sein Vermögen hauptsächlich in Aktien von Micro und Small Caps mit hohem Wachstumspotenzial anlegt. Der diskretionär verwaltete Fonds strebt über einen empfohlenen Anlagehorizont von 5 Jahren ein langfristiges Wachstum an, das über dem seines Referenzindex, des MSCI® Europe Micro Cap (mit Wiederanlage der Dividenden) liegt. Der Fonds verfolgt jedoch nicht das Ziel, die Performance dieses Index auf die eine oder andere Weise nachzubilden. Der Fondsmanager ist bestrebt, die Ziele des Fonds durch Anlage des Vermögens in Mikrounternehmen oder kleine und mittelständische Unternehmen, die ihren Sitz im Europäischen Wirtschaftsraum haben oder dort notiert sind, zu erreichen. Im Rahmen seiner Liquiditätsplanung kann der Fonds zudem bis zu 25 % seines Vermögens in Forderungstiteln, die von einem Unternehmen oder einem Staat begeben werden, sowie in Geldmarktinstrumenten oder Schultiteln, handelbaren Schultiteln, Euro-Anleihen mit mittlerer Laufzeit und sonstigen Anleihe- oder Geldmarktpapieren anlegen. Zur Absicherung oder für ein Engagement können Finanztermingeschäfte abgeschlossen werden. Der Fonds kann bis zu 10 % seines Nettovermögens in Anteilen von Organismen für gemeinsame Anlagen und bis zu 10 % seines Vermögens an anderen internationalen Märkten als im Europäischen Wirtschaftsraum anlegen.

```

class MirrorX(object):
    """This adds an
    """
    def __init__(self, mirror_mod, mirror_obj):
        self.mirror_mod = mirror_mod
        self.mirror_obj = mirror_obj

    def __getattr__(self, attr):
        if attr == "mirror_mod":
            return self.mirror_mod
        elif attr == "mirror_obj":
            return self.mirror_obj
        elif attr == "mirror_mod.use_x":
            return self.mirror_mod.use_x
        elif attr == "mirror_mod.use_y":
            return self.mirror_mod.use_y
        elif attr == "mirror_mod.use_z":
            return self.mirror_mod.use_z
        else:
            return getattr(self.mirror_obj, attr)

    def __setattr__(self, attr, value):
        if attr == "mirror_mod":
            self.mirror_mod = value
        elif attr == "mirror_obj":
            self.mirror_obj = value
        elif attr == "mirror_mod.use_x":
            self.mirror_mod.use_x = value
        elif attr == "mirror_mod.use_y":
            self.mirror_mod.use_y = value
        elif attr == "mirror_mod.use_z":
            self.mirror_mod.use_z = value
        else:
            setattr(self.mirror_obj, attr, value)

    def __delattr__(self, attr):
        if attr == "mirror_mod":
            del self.mirror_mod
        elif attr == "mirror_obj":
            del self.mirror_obj
        elif attr == "mirror_mod.use_x":
            del self.mirror_mod.use_x
        elif attr == "mirror_mod.use_y":
            del self.mirror_mod.use_y
        elif attr == "mirror_mod.use_z":
            del self.mirror_mod.use_z
        else:
            delattr(self.mirror_obj, attr)

    def __str__(self):
        return f"MirrorX(mirror_mod={self.mirror_mod}, mirror_obj={self.mirror_obj})"

    def __repr__(self):
        return f"MirrorX(mirror_mod={self.mirror_mod}, mirror_obj={self.mirror_obj})"

    def __eq__(self, other):
        return self.mirror_mod == other.mirror_mod and self.mirror_obj == other.mirror_obj

    def __ne__(self, other):
        return not self.__eq__(other)

    def __lt__(self, other):
        return self.mirror_mod < other.mirror_mod and self.mirror_obj < other.mirror_obj

    def __gt__(self, other):
        return self.mirror_mod > other.mirror_mod and self.mirror_obj > other.mirror_obj

    def __le__(self, other):
        return self.mirror_mod <= other.mirror_mod and self.mirror_obj <= other.mirror_obj

    def __ge__(self, other):
        return self.mirror_mod >= other.mirror_mod and self.mirror_obj >= other.mirror_obj

    def __add__(self, other):
        return self.mirror_mod + other.mirror_mod and self.mirror_obj + other.mirror_obj

    def __sub__(self, other):
        return self.mirror_mod - other.mirror_mod and self.mirror_obj - other.mirror_obj

    def __mul__(self, other):
        return self.mirror_mod * other.mirror_mod and self.mirror_obj * other.mirror_obj

    def __div__(self, other):
        return self.mirror_mod / other.mirror_mod and self.mirror_obj / other.mirror_obj

    def __mod__(self, other):
        return self.mirror_mod % other.mirror_mod and self.mirror_obj % other.mirror_obj

    def __pow__(self, other):
        return self.mirror_mod ** other.mirror_mod and self.mirror_obj ** other.mirror_obj

    def __getitem__(self, item):
        return self.mirror_mod[item] and self.mirror_obj[item]

    def __setitem__(self, key, value):
        self.mirror_mod[key] = value and self.mirror_obj[key] = value

    def __delitem__(self, key):
        del self.mirror_mod[key] and del self.mirror_obj[key]

    def __iter__(self):
        return iter(self.mirror_mod) and iter(self.mirror_obj)

    def __len__(self):
        return len(self.mirror_mod) and len(self.mirror_obj)

    def __contains__(self, item):
        return item in self.mirror_mod and item in self.mirror_obj

    def __call__(self):
        return self.mirror_mod() and self.mirror_obj()

    def __getitem__(self, item):
        return self.mirror_mod[item] and self.mirror_obj[item]

    def __setitem__(self, key, value):
        self.mirror_mod[key] = value and self.mirror_obj[key] = value

    def __delitem__(self, key):
        del self.mirror_mod[key] and del self.mirror_obj[key]

    def __iter__(self):
        return iter(self.mirror_mod) and iter(self.mirror_obj)

    def __len__(self):
        return len(self.mirror_mod) and len(self.mirror_obj)

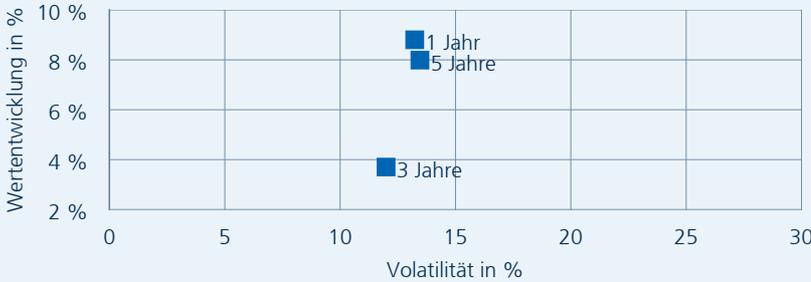
    def __contains__(self, item):
        return item in self.mirror_mod and item in self.mirror_obj

    def __call__(self):
        return self.mirror_mod() and self.mirror_obj()
    
```

RISIKOINDIKATOR*



Risiko-Rendite-Diagramm



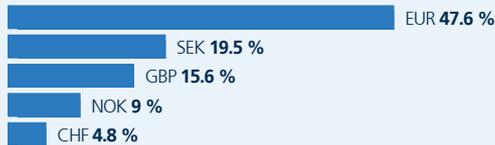
*SRI gemäß Basisinformationsblatt

PORTFOLIOSTRUKTUR

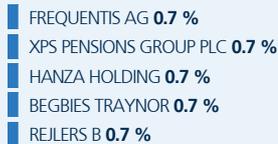
Top 5 Länderverteilung



Top 5 Währungsverteilung



Top 5 Holdings



Rechtliche Hinweise

Datenquelle für Fondsinformationen: cleversoft GmbH. Die von der cleversoft GmbH bereitgestellten Informationen stellen keine Empfehlungen dar und dienen nicht der Anlageberatung. Insbesondere stellen diese Informationen keine Finanzanalyse im Sinne von § 34 b WpHG dar. Die cleversoft GmbH trifft keinerlei Aussage zur bisherigen und zukünftigen Wertentwicklung der angezeigten Wertpapiere. Wegen der Dynamik der Finanzmärkte muss jegliche Haftung im Zusammenhang mit der Nutzung der Informationen oder dem (evtl. auch irrtümlichen) Vertrauen auf deren Richtigkeit, Vollständigkeit oder Aktualität ausgeschlossen werden. Informieren Sie sich daher vor dem Fondserwerb auf der Internetseite der jeweiligen Fondsgesellschaft.

Fonds-Fakten

Fondskategorie	Aktienfonds
WKN	A2AQ7L
ISIN	LU1303940784
Auflagedatum	31. Dezember 2013
Fondsvolumen	95,34 Mio. EUR (03.07.2025)
Kapitalverwaltungs-gesellschaft	Mandarine Gestion
Fondsmanagement	Augustin Lecoq, Théo Colombani
Depotbank	BNP Paribas Securities Services (L)
Sitzland	Luxemburg
Fondswährung	EUR
Rücknahmepreis	26,69 EUR
Ertragsverwendung	Thesaurierend
Geschäftsjahr	01. Januar - 31. Dezember
Transparenzverord-nung (EU) 2019/2088	Art. 8 TVO ESG Merkmale

Fonds-Konditionen

Ausgabeauschlag	Ein Ausgabeaufschlag fällt im Rahmen der fondsge-bundenen Rentenversiche-rungen von R+V nicht an.
Max. Verwaltungs-vergütung p. a.	1,95 %
Max. Fondsmanage-ment Gebühr p.a.	-
Max. Depotbankver-gütung p.a.	-
Laufende Kosten p.a.	1,95 %
Erfolgsabhängige Vergütung	20,00 %

Kennzahlen

Volatilität	Maximum Drawdown
1 Jahr +13,23 %	1 Jahr -11,84 %
3 Jahre +11,99 %	3 Jahre -24,76 %
5 Jahre +13,46 %	5 Jahre -36,17 %

Sharpe Ratio	Verlustdauer in Monaten
1 Jahr 0,50	1 Jahr 4
3 Jahre 0,08	3 Jahre 4
5 Jahre 0,49	5 Jahre 4

Informationen erhalten Sie in den Volksbanken und Raiffeisenbanken, R+V-Agenturen sowie bei der Direktion der Gesellschaften der R+V Versicherungsgruppe, Raiffeisenplatz 1, 65189 Wiesbaden.

Telefon: 0800 533-1171

Kostenfrei aus allen deutschen Fest- und Mobilfunknetzen.

www.ruv.de

R+V Lebensversicherung AG