

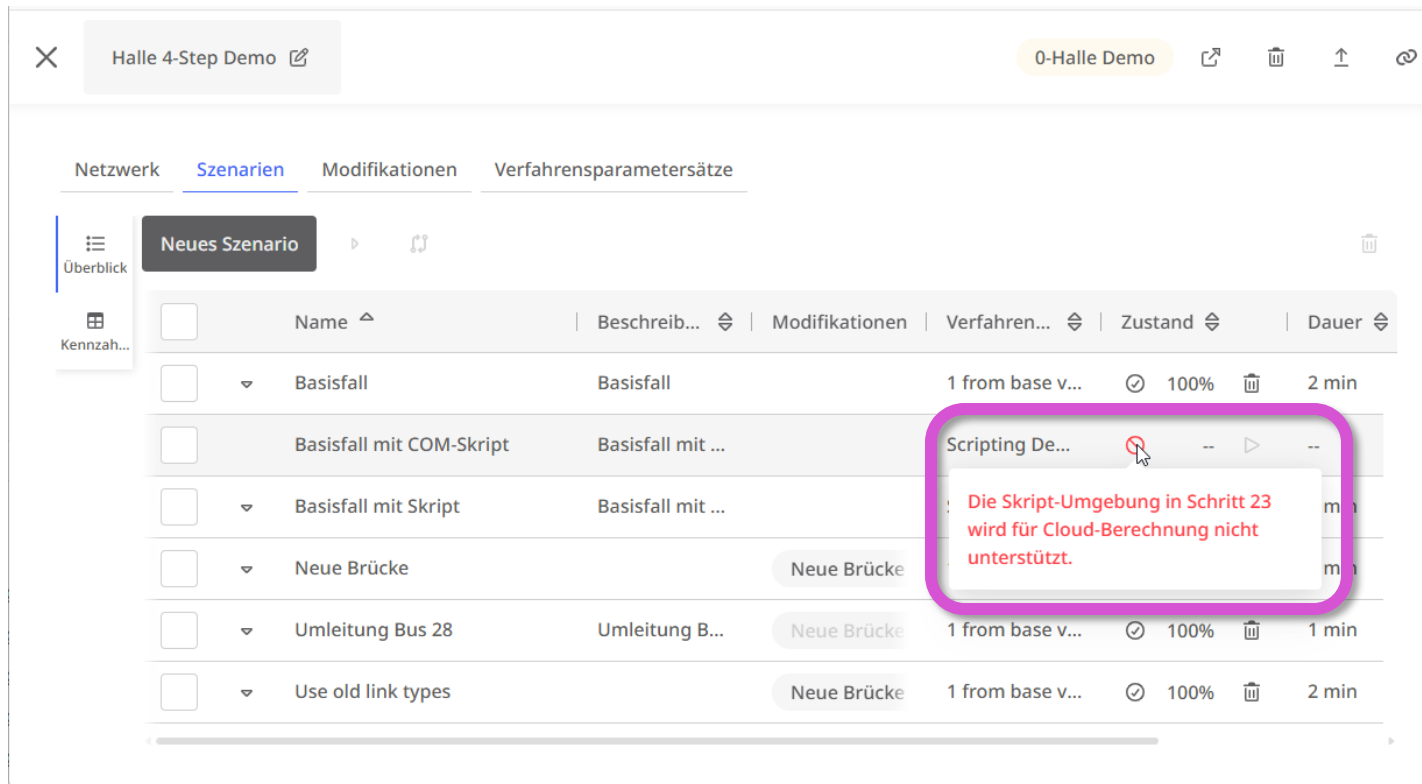
PTV GROUP

part of Umovity

Scripting in PTV Visum & PTV Hub

```
for (unsigned char radio_id=0; radio_id<2; radio_id++)  
{  
    nt_rm_data_valid_proc (radio_id, display);  
    nt_rm_data_valid_proc (radio_id, blind);  
    (void) nt_proc_rm_tuning_data (radio_id, display);  
    (void) nt_proc_rm_tuning_data (radio_id, blind);  
    // =====  
    if (nt_as_i_radio_owner(radio_id))  
    {  
        if (radio_id == 0)  
            rnNtApiMgr.xmitNavRadioDisplChnlsInfo(radio_id, &nt_nt_rad_1_displ_chnls_info); // send onside radio data to  
        else  
            if (radio_id == 1)  
                rnNtApiMgr.xmitNavRadioDisplChnlsInfo(radio_id, &nt_nt_rad_2_displ_chnls_info); // send onside radio data to  
    }  
  
    if (nt_as_i_radio_owner(radio_id,blind))  
    {  
        if (radio_id == 0)  
            rnNtApiMgr.xmitNavRadioBlindChnlsInfo(radio_id, &nt_nt_rad_1_blind_chnls_info); // send onside radio data to  
        else  
            if (radio_id == 1)  
                rnNtApiMgr.xmitNavRadioBlindChnlsInfo(radio_id, &nt_nt_rad_2_blind_chnls_info); // send onside radio data to  
    }  
}  
  
prePrimaryPair = pRnMgr->getPrimaryPair();  
preSysActiveMembers = pRnMgr->cduGetSysActiveMembers();  
preSysActiveState = pRnMgr->cduGetSysActiveState();
```

Motivation: PTV Hub



- › Cloud computing in PTV Hub basiert auf Linux
- › COM-Technologie wird unter Linux / in PTV Hub nicht unterstützt

→ Bereitstellung einer plattformunabhängigen Python-API

→ Ermöglicht es Skripte zu erstellen, die in Hub lauffähig sind

2 Motivation: COM vs Python

Table of Contents

- ILinkList
- ILinkRoute
- ILinkRouteAggregationPara
- ILinkRouteDisaggregationPara
- ILinkRouteItem
- ILinkRouteItemList
- ILinkRouteItemPuTDetail
- ILinkRouteItemPuTDetailList
- ILinkRouteItemPuTDetails
- ILinkRouteItems
- ILinkRouteList
- ILinkRoutes
- ILines
- ILineSelectionAndStopSequence
- ILineSelectionAndStopSequenceTool
- ILineSignatureStyle
- ILink
 - Overview
 - Members
 - Attributes
 - Relations
 - ReferencedBy
 - Methods
 - Properties
 - ILinkBarGeneralGPar
 - ILinkBarGPar
 - ILinkBarItem

Index

Search

Visum - COM API reference

ILink Attributes

Overview ▸ Collapse All

Visum - COM API reference : ILink

Summary

Identifier	Short name	Long name
AccNegHeight	AccNegHeight	Accumulted negative height
AccPosHeight	AccPosHeight	Accumulated positive height
AddVal_TSys	AddVal_TSys	AddValue-TSys
AddVal1	AddVal1	AddValue 1
AddVal2	AddVal2	AddValue 2
AddVal3	AddVal3	AddValue 3
AllowLeadLagOptimization	AllowLeadLagOptimization	Allow Lead/Lag Optimization
AssignDev	AssignDev	Assignment deviation
AssignICAApproachCap	AssignICAApproachCap	Approach capacity for assignment with ICA
AssignICAEffCap	AssignICAEffCap	Effective capacity for assignment with ICA
AssignICASupprUpstrVol	AssignICASupprUpstrVol	Suppressed upstream volume for assignment with ICA
AssignNumConvergedIterations	AssignNumConvergedIterations	Number of successive

Visual Basic

```
Public Function AddDemandSegment( _  
    ByVal Code As String, _  
    ByVal Mode As Variant _  
) As IDemandSegment
```

› COM-API sehr alt und nicht gut für modernes Programmieren geeignet

› Nicht durchgängig typisiert (ENUMs, VARIANT, ...)

› Zugriff auf Attribute über 'Attribut-IDs'

→ Bereitstellung einer modernen Python-API

→ Intuitivere Erstellung von Skripten, Code-Vervollständigung etc.

Cloud-kompatible Skripting-Umgebung



```
31 def __init__(self, settings):
32     self.file = None
33     self.fingerprints = set()
34     self.logdupes = True
35     self.debug = debug
36     self.logger = logging.getLogger(__name__)
37     if path:
38         self.file = open(os.path.join(path, 'requests.log'),
39                         'a')
40         self.fingerprints.update([os.path.join(path, 'requests.log')])
41
42 @classmethod
43 def from_settings(cls, settings):
44     debug = settings.getbool('SUPERFILTER_DEBUG')
45     return cls(job_dir(settings), debug)
46
47 def request_seen(self, request):
48     fp = self.request_fingerprint(request)
49     if fp in self.fingerprints:
50         return True
51     self.fingerprints.add(fp)
52     if self.file:
53         self.file.write(fp + os.linesep)
54
55 def request_fingerprint(self, request):
56     return request_fingerprint(request)
```

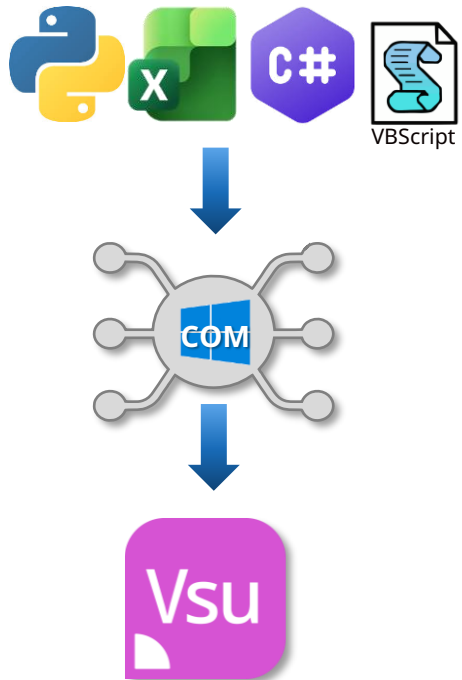


- › Neue Python-API in Visum
- › Ermöglicht die Nutzung von Skripten in Modellen für Cloud-Computing in PTV Hub
- › Nutzung im Verfahren "Skript ausführen" d.h. nur im Verfahrensablauf
- › Zugriff auf Datenmodell und Matrizen
- › Kein Zugriff auf Fenster, Dateien etc.

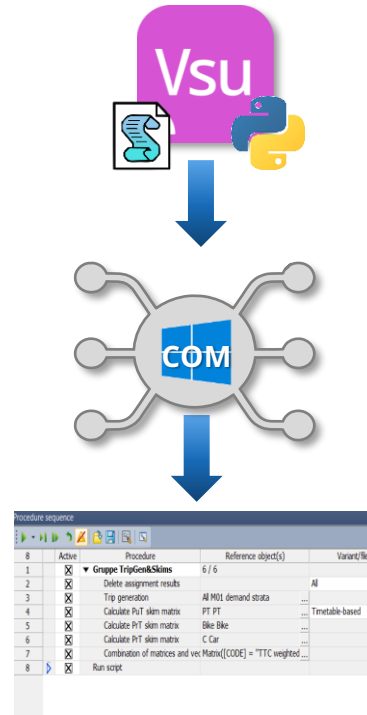
Alternativ: COM-basierte API ist weiterhin verfügbar und wird gepflegt!

Skripting-Anwendungen und APIs

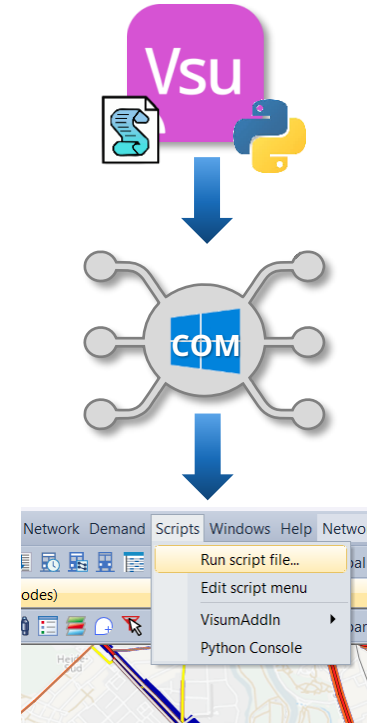
Automation



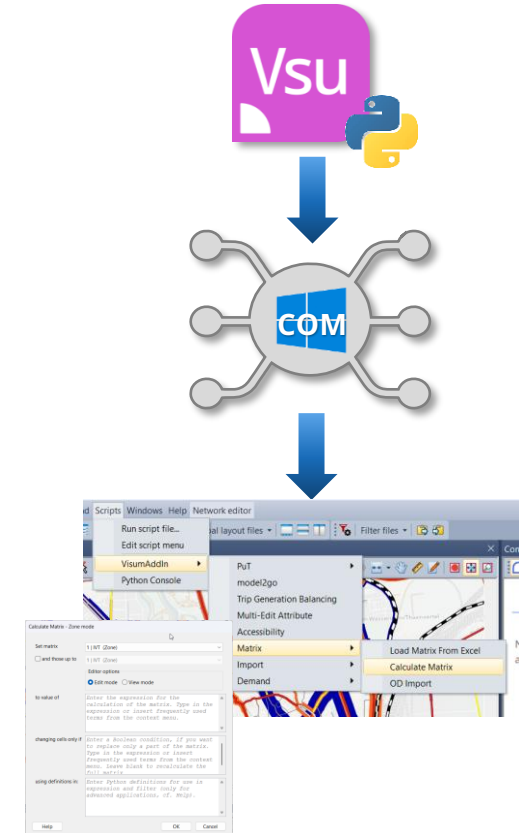
Verfahrensablauf



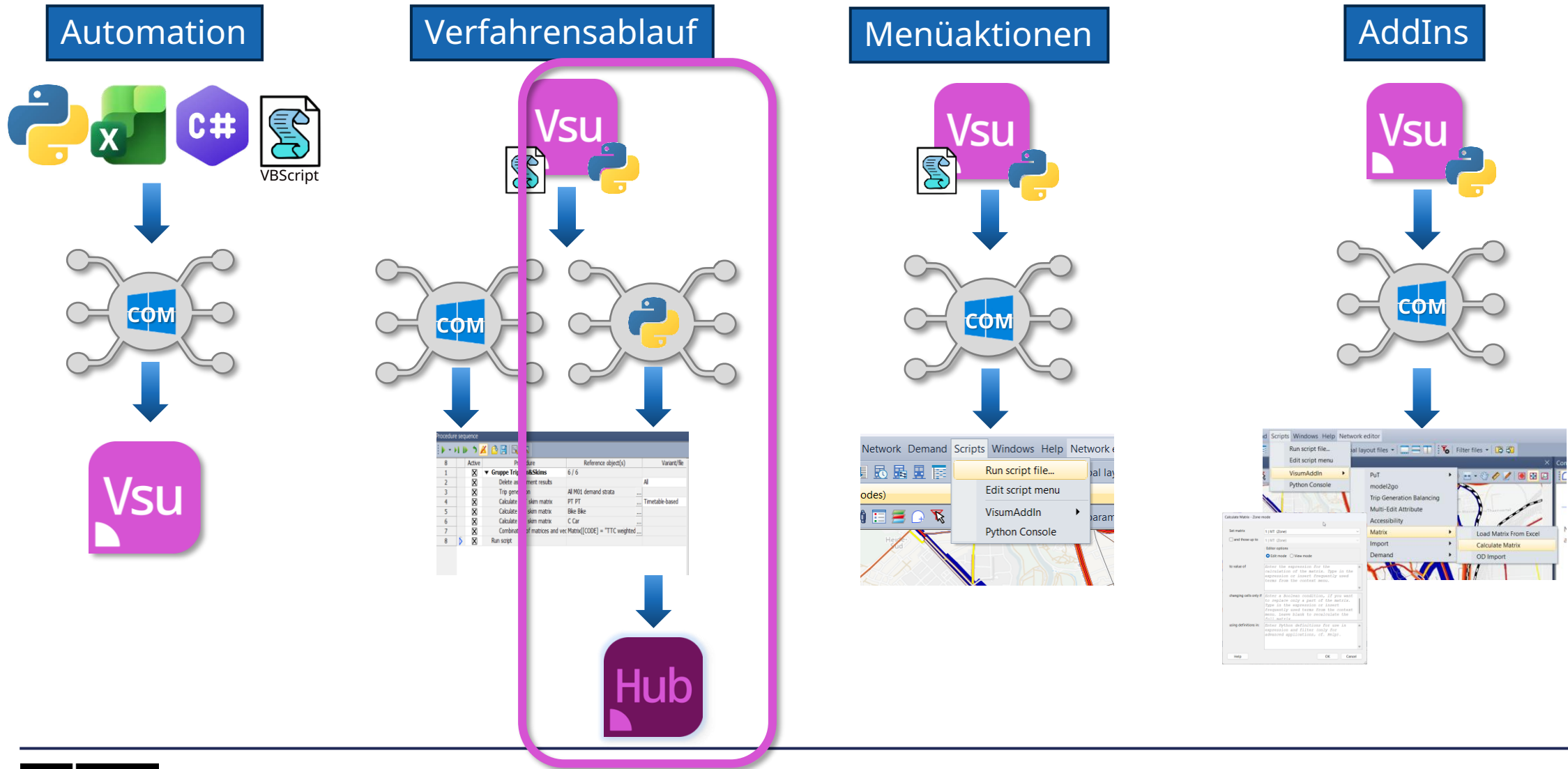
Menüaktionen



AddIns



Skripting-Anwendungen und APIs



Cloud-kompatible Skripting-Umgebung

› Verfahren "Skript ausführen" - mehrere Python 3.13 Umgebungen:

- **Standard: Python 3.13**: COM + Python-API (→ schneller)
- **Legacy**: COM-Skripte mit Benutzeroberfläche (wxPython etc.)
- **PTV Hub Cloud Computing**: Nur Python-API, kein COM, nur 'numpy', keine beliebigen 3rd-Party-Bibliotheken

The screenshot displays the PTV software interface. At the top, a yellow header bar contains the title 'Verfahrensablauf' and a set of icons for workflow management. Below this is a table with columns: 'Verfahren', 'Bezugsobjekt(e)', 'Variante/Datei', 'Kommentar', and 'Meldungen'. The table lists several steps, with step 8 highlighted in orange and labeled 'Skript ausführen'. A dialog box titled 'Skript ausführen' is open over the table. It features a dropdown menu for 'Skript-Umgebung' with the following options: 'Python 3.13 (PTV Hub cloud computing)', 'Python 3.13', 'Python 3.13 (PTV Hub cloud computing)', 'Python 3.13 (Legacy)', and 'VBScript'. The 'Python 3.13 (PTV Hub cloud computing)' option is selected. Below the dropdown, there are radio buttons for 'Skript-Datei angeben' and 'Skript-Code direkt eingeben', with the latter being selected. The dialog also shows a preview of the Python code being executed, which includes imports for 'visum' and 'numpy', and logic for creating a numeric data attribute for 'DmdCarAvg'.

Verfahren	Bezugsobjekt(e)	Variante/Datei	Kommentar	Meldungen
8	Gruppe TripGen&Skims	0 / 6	TripGen&Skims	
1	Umlegungsergebnisse löschen	Alle	Init assignment	
2	Verkehrserzeugung	Alle M01-NSchichten	Trip Generation all demand strata	
3	ÖV-Kenngrößenmatrix			
4	IV-Kenngrößenmatrix			
5	IV-Kenngrößenmatrix			
6	Kombination von			
7				
8	Skript ausführen			

```
1 import visum
2 import numpy
3
4 allZones = list(visum.net.zones)
5
6 mtxCar = visum.net.matrices[39]
7 npMtxCar = numpy.array(mtxCar)
8
9 if visum.net.attribute_definitions.zone_udas.get("DmdCarAvg") == None:
10     visum.net.attribute_definitions.zone_udas.create_numeric_data_attribute("DmdCarAvg")
11     print("Created UDA DmdCarAvg for zones")
12 else:
```

Python-Umgebung allgemein

- › Alle Python-Umgebungen basieren auf Python 3.13
- › Bibliotheken auf aktuellem Stand
- › Weniger AddIns (→ siehe 'Neues In Visum 2026')
- › Diverse nicht mehr benutzte 3rd-Party-Bibliotheken entfernt (Softwaresicherheit)

The screenshot displays the PTV Visum software interface. The main window, titled 'Verfahrensablauf', shows a process flow table with columns for 'Verfahren' (Method), 'Bezugsobjekt(e)' (Reference Object(s)), 'Variante/Datei' (Variant/File), 'Kommentar' (Comment), and 'Meldungen' (Messages). The table lists various steps, including 'Gruppe TripGen&Skims' and 'Skript ausführen'. A dialog box titled 'Skript ausführen' is open, showing a dropdown menu for 'Skript-Umgebung' (Script Environment) with options: 'Python 3.13 (PTV Hub cloud computing)', 'Python 3.13', 'Python 3.13 (PTV Hub cloud computing)', 'Python 3.13 (Legacy)', and 'VBScript'. The 'Skript-Code direkt' (Run Script Code Directly) option is selected. The script code is visible in the dialog box, starting with 'import visum' and 'import numpy'. On the right side of the interface, there is a panel titled 'Aktionen' (Actions) with various icons for actions like 'Einfügen' (Insert), 'Gruppe einfügen' (Insert Group), 'Wechseln' (Switch), 'Bearbeiten' (Edit), 'Nach oben' (Move Up), 'Nach unten' (Move Down), 'Duplizieren' (Duplicate), 'Löschen' (Delete), 'In Zwischenablage ausschneiden' (Cut to Clipboard), 'In Zwischenablage kopieren' (Copy to Clipboard), 'Aus Zwischenablage einfügen' (Paste from Clipboard), 'Alle aktiv setzen' (Set All Active), and 'Alle inaktiv setzen' (Set All Inactive).

	Aktiv	Verfahren	Bezugsobjekt(e)	Variante/Datei	Kommentar	Meldungen
8						
1	☒	Gruppe TripGen&Skims	0 / 6		TripGen&Skims	
2	☐	Umlegungsergebnisse löschen		Alle	Init assignment	
3	☐	Verkehrserzeugung	Alle M01-NSchichten		Trip Generation all demand strata	
4	☐	ÖV-Kenngrößenmatrix				
5	☐	IV-Kenngrößenmatrix				
6	☐	IV-Kenngrößenmatrix				
7	☐	Kombination von Mat				
8	☒	Skript ausführen				

```
1 import visum
2 import numpy
3
4 allZones = list(visum.net.zones)
5
6 mtxCar = visum.net.matrices[39]
7 npMtxCar = numpy.array(mtxCar)
8
9 if visum.net.attribute_definitions.zone_udas.get("DmdCarAvg") == None:
10     visum.net.attribute_definitions.zone_udas.create_numeric_data_attribute("DmdCa
11     print("Created UDA DmdCarAvg for zones")
12 else:
```

Python-Umgebungen

Standard

Run script

Script environment: Python 3.13

☐ Specify script file

☒ Directly enter script code:

```
1 # Access Visum as Visum object via the variable 'COM'.
```

Hub

Run script

Script environment: Python 3.13 (PTV Hub cloud computir

☐ Specify script file

☒ Directly enter script code:

```
1 import visum
2 print("Nodes in network:", visum.net.nodes)
```

,Legacy'

Run script

Script environment: Python 3.13 (Legacy)

☐ Specify script file

☒ Directly enter script code:

```
1 # Access Visum as Visum object via the variable 'COM'.
```

COM	✓	✗	✓
Python-API	✓	✓	✓
GUI (wx, TK)	✗	✗	✓
Verfahrensablauf	✓	✓	✓
PTV Hub	✗	✓	✗
Skriptmenu	✗	✗	✓
AddIns	✗	✗	✓
Vorinstallierte Bibliotheken	numpy, pandas, matplotlib, osgeo, , ...	numpy	numpy, pandas, matplotlib, osgeo, ... + wx, VisumPy
Eigene Bibliotheken	✓	✓	✓

Python-Umgebungen

Standard

Run script

Script environment: Python 3.13

☐ Specify script file

☒ Directly enter script code:

```
1 # Access Visum as Visum object via the variable 'COM'.
```

Hub

Run script

Script environment: Python 3.13 (PTV Hub cloud computir

☐ Specify script file

☒ Directly enter script code:

```
1 import visum
2 print("Nodes in network:", visum.net.nodes)
```

,Legacy'

Run script

Script environment: Python 3.13 (Legacy)

☐ Specify script file

☒ Directly enter script code:

```
1 # Access Visum as Visum object via the variable 'COM'.
```

COM	✓	✗	✓
Python-API	✓	✓	✓
GUI (wx, TK)	✗	✗	✓
Verfahrensablauf	✓	✓	✓
PTV Hub	✗	✓	✗
Skriptmenu	✗	✗	✓
AddIns	✗	✗	✓
Vorinstallierte Bibliotheken	numpy, pandas, matplotlib, osgeo, , ...	numpy	numpy, pandas, matplotlib, osgeo, ... + wx, VisumPy
Eigene Bibliotheken	✓	✓	✓

COM vs Python-API

Skript-Code direkt eingeben:



```
1 import visum
2
3 print("Example 01, using Python-API and direct attribute access")
4 fromNode = visum.net.nodes[984]
5 toNode = visum.net.nodes[10000675]
6
7 link1 = visum.net.links.find_by_nodes(fromNode,toNode)
8 reverse_link1 = link1.reverse_link
9 print("Link {} connects nodes {} and {}".format(link1.no, link1.from_node.no, link1.to_node_no))
10 print("Reverse link {} is of type {} with volume {:.2f}". \
11       format(reverse_link1.no, reverse_link1.type_no, reverse_link1.vol_veh_prt[visum.network.AHP.AP]))
```

Skript-Code direkt eingeben:



```
1 # Greifen Sie über die Variable 'Visum' auf Visum als COM-Objekt zu.
2 print("Example 01, using COM-API")
3 fromNode = Visum.Net.Nodes.ItemByKey(984)
4 toNode = Visum.Net.Nodes.ItemByKey(10000675)
5
6 link1 = Visum.Net.Links.ItemByKey(fromNode, toNode)
7 print("Link {:.0f} connects nodes {:.0f} and {:.0f}").format(link1.AttValue("No"),link1.AttValue("FromNodeNo"), link1.AttValue("ToNodeNo"))
8 link2 = Visum.Net.Links.ItemByKey(10000675, 984)
9 print("Reverse Link is of type {:.0f} with volume {:.2f}").format(link2.AttValue("TypeNo"), link2.AttValue("VolVehPrt(AP)"))
```

COM vs Python-API

Skript-Code direkt eingeben:

```
1 import visum
2
3 print("Example 01, using Python-API and direct attribute access")
4 fromNode = visum.net.nodes[984]
5 toNode = visum.net.nodes[10000675]
6
7 link1 = visum.net.links.find_by_nodes(fromNode,toNode)
8 reverse_link1 = link1.reverse_link
9 print("Link {} connects nodes {} and {}".format(link1.no, fromNode.no, toNode.no))
10 print("Reverse link {} is of type {} with volume {}".format(reverse_link1.no, reverse_link1.type, reverse_link1.vol))
11 print("Reverse link {} is of type {} with volume {}".format(reverse_link1.no, reverse_link1.type, reverse_link1.vol))
```



Skript-Code direkt eingeben:

```
1 # Greifen Sie über die Variable 'Visum' auf das COM-Objekt zu.
2 print("Example 01, using COM-API")
3 fromNode = Visum.Net.Nodes.ItemByKey(984)
4 toNode = Visum.Net.Nodes.ItemByKey(10000675)
5
6 link1 = Visum.Net.Links.ItemByKey(fromNode, toNode)
7 print("Link {:.0f} connects nodes {:.0f} and {:.0f}".format(link1.AttValue("No"), link1.AttValue("FromNodeNo"), link1.AttValue("ToNodeNo")))
8 link2 = Visum.Net.Links.ItemByKey(10000675, 984)
9 print("Reverse Link is of type {:.0f} with volume {:.2f}".format(link2.AttValue("TypeNo"), link2.AttValue("VolVehPrt(AP)")))
```



Demo

COM vs Python-API

Skript-Code direkt eingeben:



```
1 import visum
2
3 print("Example 01, using Python-API and direct attribute access")
4 fromNode = visum.net.nodes[984]
5 toNode = visum.net.nodes[10000675]
6
7 link1 = visum.net.links.find_by_nodes(fromNode,toNode)
8 reverse_link1 = link1.reverse_link
9 print("Link {} connects nodes {} and {}".format(link1.no, link1.from_node.no, link1.to_node_no))
10 print("Reverse link {} is of type {} with volume {:.2f}". \
11       format(reverse_link1.no, reverse_link1.type_no, reverse_link1.vol_veh_prt[visum.network.AHP.AP]))
```

Skript-Code direkt eingeben:



```
1 # Greifen Sie über die Variable 'Visum' auf Visum als COM-Objekt zu.
2 print("Example 01, using COM-API")
3 fromNode = Visum.Net.Nodes.ItemByKey(984)
4 toNode = Visum.Net.Nodes.ItemByKey(10000675)
5
6 link1 = Visum.Net.Links.ItemByKey(fromNode, toNode)
7 print("Link {:.0f} connects nodes {:.0f} and {:.0f}").format(link1.AttValue("No"),link1.AttValue("FromNodeNo"), link1.AttValue("ToNodeNo"))
8 link2 = Visum.Net.Links.ItemByKey(10000675, 984)
9 print("Reverse Link is of type {:.0f} with volume {:.2f}").format(link2.AttValue("TypeNo"), link2.AttValue("VolVehPrt(AP)"))
```

COM vs Python-API

Skript-Code direkt eingeben:



```
1  import visum
2
3  print("Example 01, using Python-API and direct attribute access")
4  fromNode = visum.net.nodes[984]
5  toNode = visum.net.nodes[10000675]
6
7  link1 = visum.net.links.find_by_nodes(fromNode,toNode)
8  reverse_link1 = link1.reverse_link
9  print("Link {} connects nodes {} and {}".format(link1.no, link1.from_node.no, link1.to_node.no))
10 print("Reverse link {} is of type {} with volume {:.2f}". \
11       format(reverse_link1.no, reverse_link1.type_no, reverse_link1.vol_veh_prt[visum.VehPrtType.AP]))
9  print("Reverse Link is of type {:.0f} with volume {:.2f}".format(link2.AttValue("TypeNo"), link2.AttValue("VolVehPrt(AP)")))
```

COM vs Python-API

Skript-Code direkt eingeben:



```
1 import visum
2
3 print("Example 01, using Python-API and direct attribute access")
4 fromNode = visum.net.nodes[984]
5 toNode = visum.net.nodes[10000675]
6
7 link1 = visum.net.links.find_by_nodes(fromNode,toNode)
8 reverse_link1 = link1.reverse_link
9 print("Link {} connects nodes {} and {}".format(link1.no, link1.from_node.no, link1.to_node_no))
10 print("Reverse link {} is of type {} with volume {:.2f}". \
11       format(reverse_link1.no, reverse_link1.type_no, reverse_link1.vol_veh_prt[visum.network.AHP.AP]))
```

Skript-Code direkt eingeben:



```
1 # Greifen Sie über die Variable 'Visum' auf Visum als COM-Objekt zu.
2 print("Example 01, using COM-API")
3 fromNode = Visum.Net.Nodes.ItemByKey(984)
4 toNode = Visum.Net.Nodes.ItemByKey(10000675)
5
6 link1 = Visum.Net.Links.ItemByKey(fromNode, toNode)
7 print("Link {:.0f} connects nodes {:.0f} and {:.0f}").format(link1.AttValue("No"),link1.AttValue("FromNodeNo"), link1.AttValue("ToNodeNo"))
8 link2 = Visum.Net.Links.ItemByKey(10000675, 984)
9 print("Reverse Link is of type {:.0f} with volume {:.2f}").format(link2.AttValue("TypeNo"), link2.AttValue("VolVehPrt(AP)"))
```

COM vs Python-API

Skript-Code direkt eingeben:



```
1 import visum
2
3 print("Example 01. using Python-API and direct attribute access")
4 fromNode = visum.net.nodes[984]
5 toNode = visum.net.nodes[10000675]
6
7 link1 = visum.net.links.find_by_nodes(fromNode,toNode)
8 reverse_link1 = link1.reverse_link
9 print("Link {} connects nodes {} and {}".format(link1.no, fromNode.id, toNode.id))
10 print("Reverse link {} is of type {} with volume {}".format(reverse_link1.no, reverse_link1.type_no, reverse_link1.volume))
11 format(reverse_link1.no, reverse_link1.type_no, reverse_link1.volume)
```

visum **statt** Visum

[id] **statt** ItemByKey(id)

Skript-Code direkt eingeben:

```
1 # Greifen Sie über die Variable 'visum' auf visum als COM-Objekt zu.
2 print("Example 01. using COM-API")
3 fromNode = Visum.Net.Nodes.ItemByKey(984)
4 toNode = Visum.Net.Nodes.ItemByKey(10000675)
5
6 link1 = Visum.Net.Links.ItemByKey(fromNode, toNode)
7 print("Link {:.0f} connects nodes {:.0f} and {:.0f}").format(link1.AttValue("No"),link1.AttValue("FromNode"),link1.AttValue("ToNode"))
```

COM vs Python-API

Skript-Code direkt eingeben:



```
1 import visum
2
3 print("Example 01, using Python-API and direct attribute access")
4 fromNode = visum.net.nodes[984]
5 toNode = visum.net.nodes[10000675]
6
7 link1 = visum.net.links.find_by_nodes(fromNode,toNode)
8 reverse_link1 = link1.reverse_link
9 print("Link {} connects nodes {} and {}".format(link1.no, link1.from_node.no, link1.to_node_no))
10 print("Reverse link {} is of type {} with volume {:.2f}". \
11       format(reverse_link1.no, reverse_link1.type_no, reverse_link1.vol_veh_prt[visum.network.AHP.AP]))
```

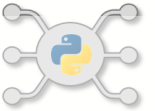
Skript-Code direkt eingeben:



```
1 # Greifen Sie über die Variable 'Visum' auf Visum als COM-Objekt zu.
2 print("Example 01, using COM-API")
3 fromNode = Visum.Net.Nodes.ItemByKey(984)
4 toNode = Visum.Net.Nodes.ItemByKey(10000675)
5
6 link1 = Visum.Net.Links.ItemByKey(fromNode, toNode)
7 print("Link {:.0f} connects nodes {:.0f} and {:.0f}").format(link1.AttValue("No"),link1.AttValue("FromNodeNo"), link1.AttValue("ToNodeNo"))
8 link2 = Visum.Net.Links.ItemByKey(10000675, 984)
9 print("Reverse Link is of type {:.0f} with volume {:.2f}").format(link2.AttValue("TypeNo"), link2.AttValue("VolVehPrt(AP)"))
```

COM vs Python-API

```
6
7 link1 = visum.net.links.find_by_nodes(fromNode,toNode)
8 reverse_link1 = link1.reverse_link
9 print("Link {} connects nodes {} and {}".format(link1.no, link1.from_node.no, link1.to_node_no))
10 print("Reverse link {} is of type {} with volume {:.2f}". \
11 format(reverse_link1.no, reverse_link1.type_no, reverse_link1.vol_veh_prt[visum.network.AHP.AP]))
```



link2.type_no **statt** link2.AttValue („TypeNo“)

Direkter Zugriff **statt** ‚Attribut-ID‘

[visum.network.AHP.AP] **statt** „(AP)“

```
ByKey(fromNode, toNode)
nodes {:.0f} and {:.0f})).format(link1.AttValue("No"),link1.AttValue("FromNodeNo"), link1.AttValue
ByKey(10000675, 984)
type {:.0f} with volume {:.2f})).format(link2.AttValue("TypeNo"), link2.AttValue("VolVehPrt(AP)")))
```

COM vs Python-API

Skript-Code direkt eingeben:



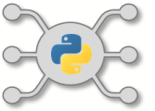
```
1 import visum
2
3 print("Example 01, using Python-API and direct attribute access")
4 fromNode = visum.net.nodes[984]
5 toNode = visum.net.nodes[10000675]
6
7 link1 = visum.net.links.find_by_nodes(fromNode,toNode)
8 reverse_link1 = link1.reverse_link
9 print("Link {} connects nodes {} and {}".format(link1.no, link1.from_node.no, link1.to_node_no))
10 print("Reverse link {} is of type {} with volume {:.2f}". \
11       format(reverse_link1.no, reverse_link1.type_no, reverse_link1.vol_veh_prt[visum.network.AHP.AP]))
```

Skript-Code direkt eingeben:



```
1 # Greifen Sie über die Variable 'Visum' auf Visum als COM-Objekt zu.
2 print("Example 01, using COM-API")
3 fromNode = Visum.Net.Nodes.ItemByKey(984)
4 toNode = Visum.Net.Nodes.ItemByKey(10000675)
5
6 link1 = Visum.Net.Links.ItemByKey(fromNode, toNode)
7 print("Link {:.0f} connects nodes {:.0f} and {:.0f}").format(link1.AttValue("No"),link1.AttValue("FromNodeNo"), link1.AttValue("ToNodeNo"))
8 link2 = Visum.Net.Links.ItemByKey(10000675, 984)
9 print("Reverse Link is of type {:.0f} with volume {:.2f}").format(link2.AttValue("TypeNo"), link2.AttValue("VolVehPrt(AP)"))
```

COM vs Python-API



```
ing Python-API and accessing attributes through Attribute-ID")  
.nodes[984]  
nodes[10000675]
```

```
visum.net.links.get(5115)
```

```
= visum.net.links[5110]
```

```
links.find_by_nodes(fromNode,toNode)
```

```
link.reverse_link
```

```
ts node {} and node {}".format(link1.no, link1.from_node_no, link1.to_node_no))
```

```
s of type {} with volume {}".format(reverse_link1.attributes["TypeNo"], reverse_link1.attributes["VolVehPrt(AP)"])]
```

```
2 print("Example 01, using COM-API")
```

```
3 fromNode
```

```
4 toNode =
```

```
5
```

```
6 link1 =
```

```
7 print("l
```

```
8 link2 =
```

```
9 print("R
```

Alternativ:

Link2.attributes („TypeNo“) **statt** link2.AttValue („TypeNo“) je(„ToNodeNo“))

visum.links.get(#) **statt** find_by_nodes()

COM vs Python-API - Funktionsumfang (Stand 2026)

		
Netzobjekte	✓	✓
Attribute, Subattribute	✓	✓
BDAs, Indirekte Attribute	✓	✓
Matrizen	✓	✓
Filter (laden / speichern)	✓	✓
Analyse (Spinne, Isochronen)	✓	✗
Verfahrensparameter	✓	✗ / (✓)
Import / Export	✓	✗
Listen, Fenster	✓	✗
Screenshots, Drucken	✓	✗
...	✓	✗

Dokumentation

- › API-Referenz
 - › in Arbeit 😊
- › Beispiele
 - › In Arbeit 😊
- › Code-Bervolksständigung im Skript-Editor
 - › Visum 2027

Visum - Internal Scripting API

MODULES

visum.network

visum.procedures

visum.network

Public reference for `visum.network` in the Visum internal scripting API.

```
class visum.network.AHP(*values)
    AH = 0
        analysis horizon

    AP = 1
        analysis period

class visum.network.AP(*values)
    Represents the static part of API sub attributes.

    AP = 1
        analysis period

class visum.network.ActChain
    act_pairs: network__containers__ActChainActPairsContainer
        Activity pairs

    activity_codes: str
        Comma-separated list of the codes of the activities.

    attributes

    code: str
        Short name of the activity chain.

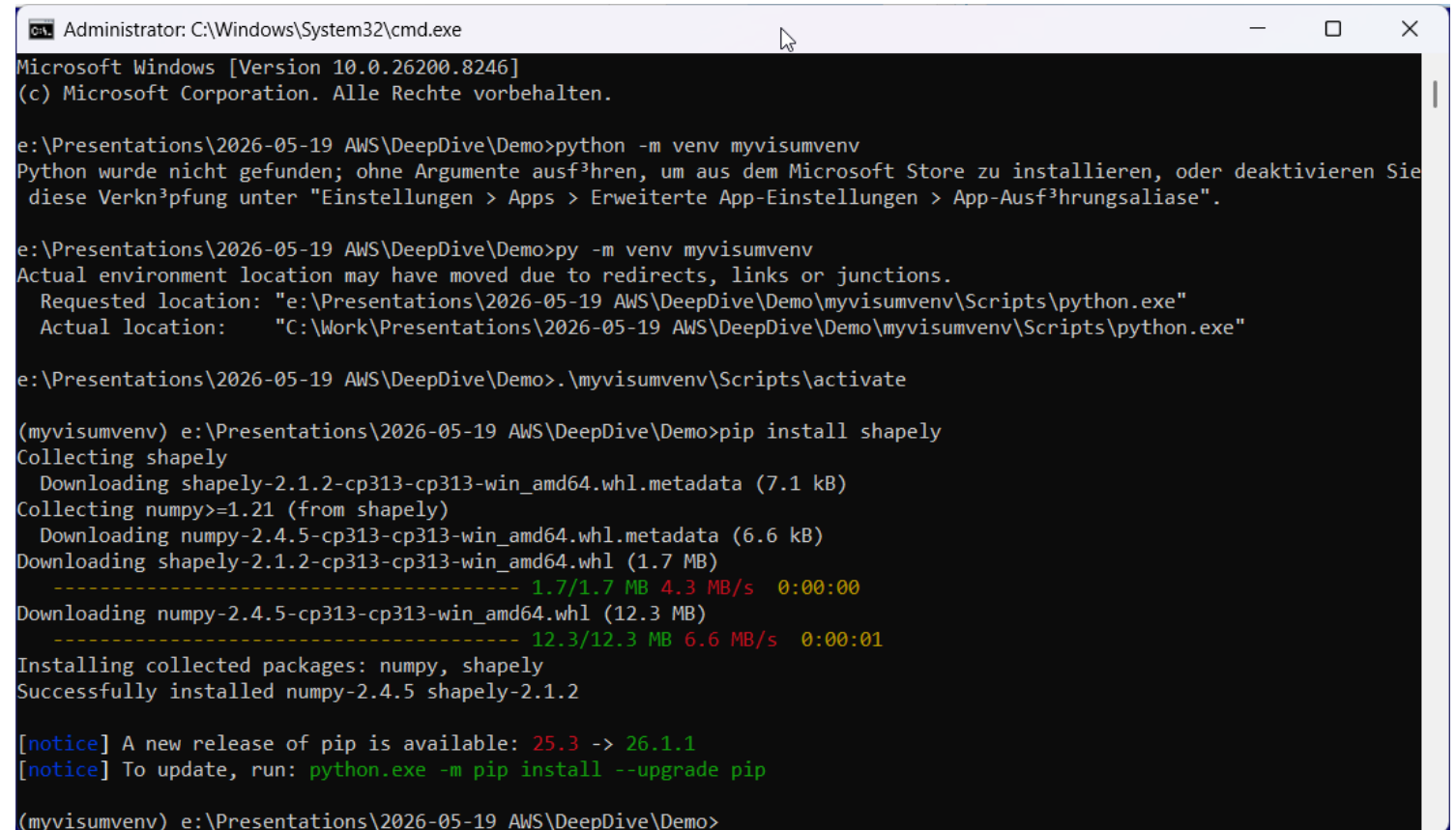
    demand_model: DemandModel
        Demand model
```

ON THIS PAGE

- AHP
- AP
- **ActChain**
 - ActChain.act_pairs
 - ActChain.activity_codes
 - ActChain.attributes
 - ActChain.code
 - ActChain.demand_model
 - ActChain.demand_model_code
 - ActChain.demand_strata
 - ActChain.destroy()
 - ActChain.name
- ActPair
- Activity
- ActivityExecution
- ActivityLocation
- ActuationType
- AutomaticConnectorsSettings
- BBackLinkCapModel
- Block
- BlockItem
- BlockItemType
- BlockVersion
- BypassControl
- COPERTCategory
- COPERTCountry
- COPERTFuel
- COPERTPeriod
- COPERTSlope
- COPERTSpeed
- COPERTVehicleStratum

Zusätzliche Bibliotheken nutzen (in Standard-Umgebung, nicht für PTV Hub)

- › Virtuelle Python-Umgebung (venv) erstellen
- › Virtuelle Umgebung aktivieren
- › Bibliotheken installieren mit *pip*



```
Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.26200.8246]
(c) Microsoft Corporation. Alle Rechte vorbehalten.

e:\Presentations\2026-05-19 AWS\DeepDive\Demo>python -m venv myvisumenv
Python wurde nicht gefunden; ohne Argumente ausführen, um aus dem Microsoft Store zu installieren, oder deaktivieren Sie
diese Verknüpfung unter "Einstellungen > Apps > Erweiterte App-Einstellungen > App-Ausführungsalias".

e:\Presentations\2026-05-19 AWS\DeepDive\Demo>py -m venv myvisumenv
Actual environment location may have moved due to redirects, links or junctions.
  Requested location: "e:\Presentations\2026-05-19 AWS\DeepDive\Demo\myvisumenv\Scripts\python.exe"
  Actual location:    "C:\Work\Presentations\2026-05-19 AWS\DeepDive\Demo\myvisumenv\Scripts\python.exe"

e:\Presentations\2026-05-19 AWS\DeepDive\Demo>.\myvisumenv\Scripts\activate

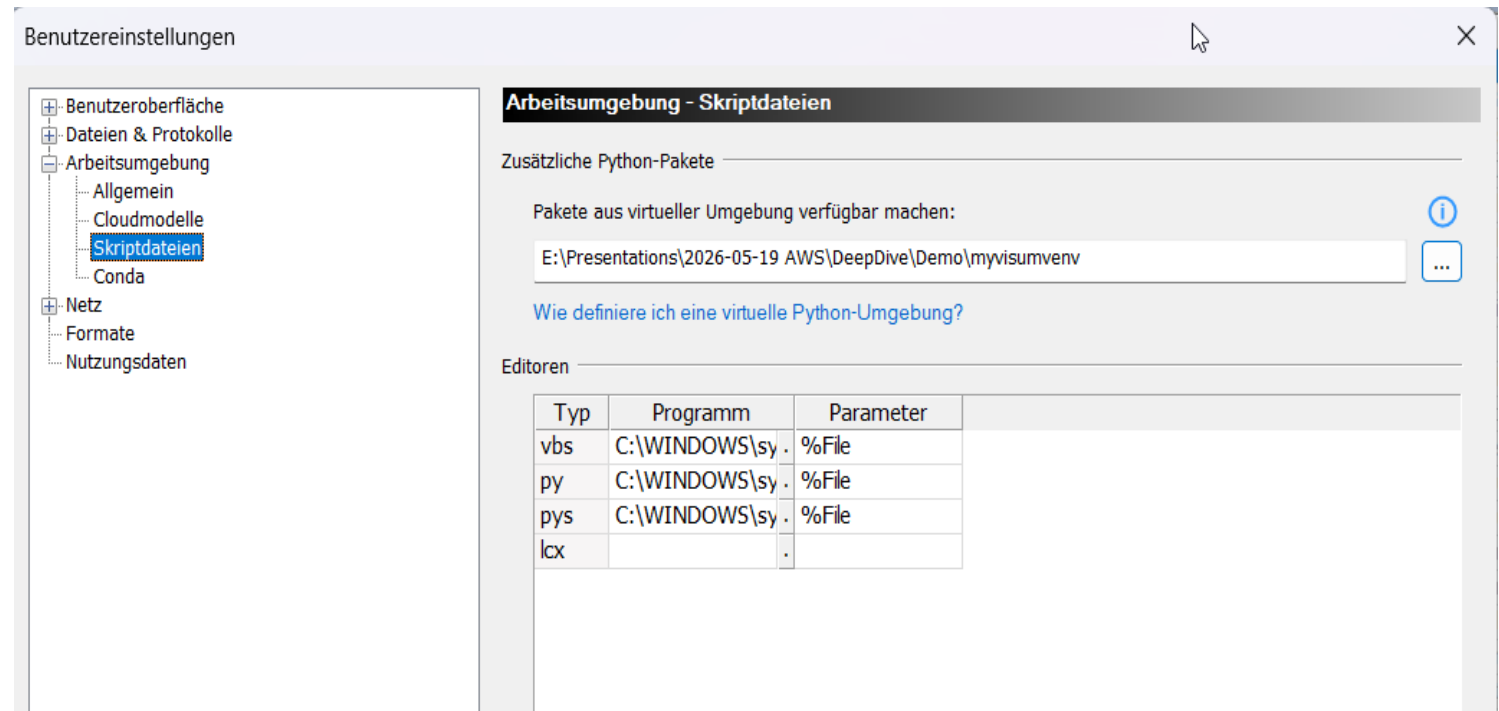
(myvisumenv) e:\Presentations\2026-05-19 AWS\DeepDive\Demo>pip install shapely
Collecting shapely
  Downloading shapely-2.1.2-cp313-cp313-win_amd64.whl.metadata (7.1 kB)
Collecting numpy>=1.21 (from shapely)
  Downloading numpy-2.4.5-cp313-cp313-win_amd64.whl.metadata (6.6 kB)
Downloaded shapely-2.1.2-cp313-cp313-win_amd64.whl (1.7 MB)
----- 1.7/1.7 MB 4.3 MB/s 0:00:00
Downloaded numpy-2.4.5-cp313-cp313-win_amd64.whl (12.3 MB)
----- 12.3/12.3 MB 6.6 MB/s 0:00:01
Installing collected packages: numpy, shapely
Successfully installed numpy-2.4.5 shapely-2.1.2

[notice] A new release of pip is available: 25.3 -> 26.1.1
[notice] To update, run: python.exe -m pip install --upgrade pip

(myvisumenv) e:\Presentations\2026-05-19 AWS\DeepDive\Demo>
```

Zusätzliche Bibliotheken nutzen (in Standard-Umgebung, nicht für PTV Hub)

- › Virtuelle Umgebung
in Visum-Benutzereinstellungen
auswählen
- › Ggf. neu starten





PTV GROUP

part of Umovity