Open for Innovation
KNIME

KNIME Python Integration Guide

KNIME AG, Zürich, Schweiz

Version 5.7 (letzte Aktualisierung auf)

Inhaltsverzeichnis

Einleitung

Dieser Leitfaden beschreibt die Einrichtung und Nutzung der KNIME Python Integration in KNIME Analytics Plattform mit seinen beiden Knoten: Python Script node und Python View node.

In der v4.5 Release von KNIME Analytics Platform haben wir den Python Script (Labs)-Knoten vorgestellt, die seit v4.7 Release der aktuelle Python Script-Knoten dieser Anleitung.

Die KNIME Python Integration arbeitet mit Python Versionen 3.9 bis 3.11 und kommt mit einer gebündelte Python-Umgebung, um Sie sofort starten zu lassen. Diese Bequemlichkeit ermöglicht die Verwendung der Knoten ohne Installation, Konfiguration oder sogar wissen Umgebungen. Das enthaltene gebündelt

...

.

Um sofort zu beginnen, ziehen und fallen Verlängerung KNIME Python Integration von Der KNIME Hub in den Werkbank zu installieren oder manuell über Datei → Installieren KNIME Erweiterungen... . Dann gehen Sie zu .

...

Der Abschnitt Verwendung der Pytho erklärt, wie die Konfiguration der Dialoge sein kann verwendet, sowie wie mit Daten zu arbeiten, die aus den Knoten kommen und gehen, wie zu arbeiten mit Chargen und wie man den Python Script-Knoten mit Skripten von älteren Python-Knoten verwendet. Es

...

und verweist auf weitere Beispiele.

...

...

, Sie müssen einrichten die verschiedenen Optionen um Umgebungen zu schaffen und zu verändern werden erforscht.

Die API der Python Integration finden Sie unter Lesen Sie die Docs .

Vor der Version v4.7 war diese Erweiterung in Laboren und KNIME Python Integration (Rechtheit) war die aktuelle Python Integration. Für alles, was mit die Vermächtnisknoten der ehemaligen KNIME Python Integration, bitte auf die Python Integrationsführer der KNIME Analytics Platform v4.6 . Die Vorteile von der aktuelle Python Script-Knoten und der Python View-Knoten im Vergleich zum Vermächtnis Knoten sind deutlich verbesserte Leistung und Datenübertragung zwischen Python-Prozesse und die KNIME Analytics Platform dank Apokalypse , a gebündelte Umgebung, um sofort zu starten, eine einheitliche API über die knime.scripting.io Modul, Conversion-Unterstützung und von beiden Pandas DataFrames und PyArrow Tabellen , Unterstützung beliebig großer Datensätze durch Nutzung Chargen . Wenn Sie suchen Python 2 Unterstützung, Sie werden auch die KNIME Python Integration verwenden müssen (Legalität).

Um größtmögliche Leistungsgewinne zu erzielen, empfehlen wir die Konfiguration Ihre Workflows zu verwenden Die Welt der Welt . Klicken Sie mit der rechten Maustaste auf einen Workflow in KNIME Explorer, wählen Konfigurieren... , dann wählen Sie die **Die Welt der Welt** Option unter Ausgewählte Tabelle Backend . Zusätzliche Informationen zu Tisch-Backends können gefunden Hier. .

Verwendung der Python-Knoten

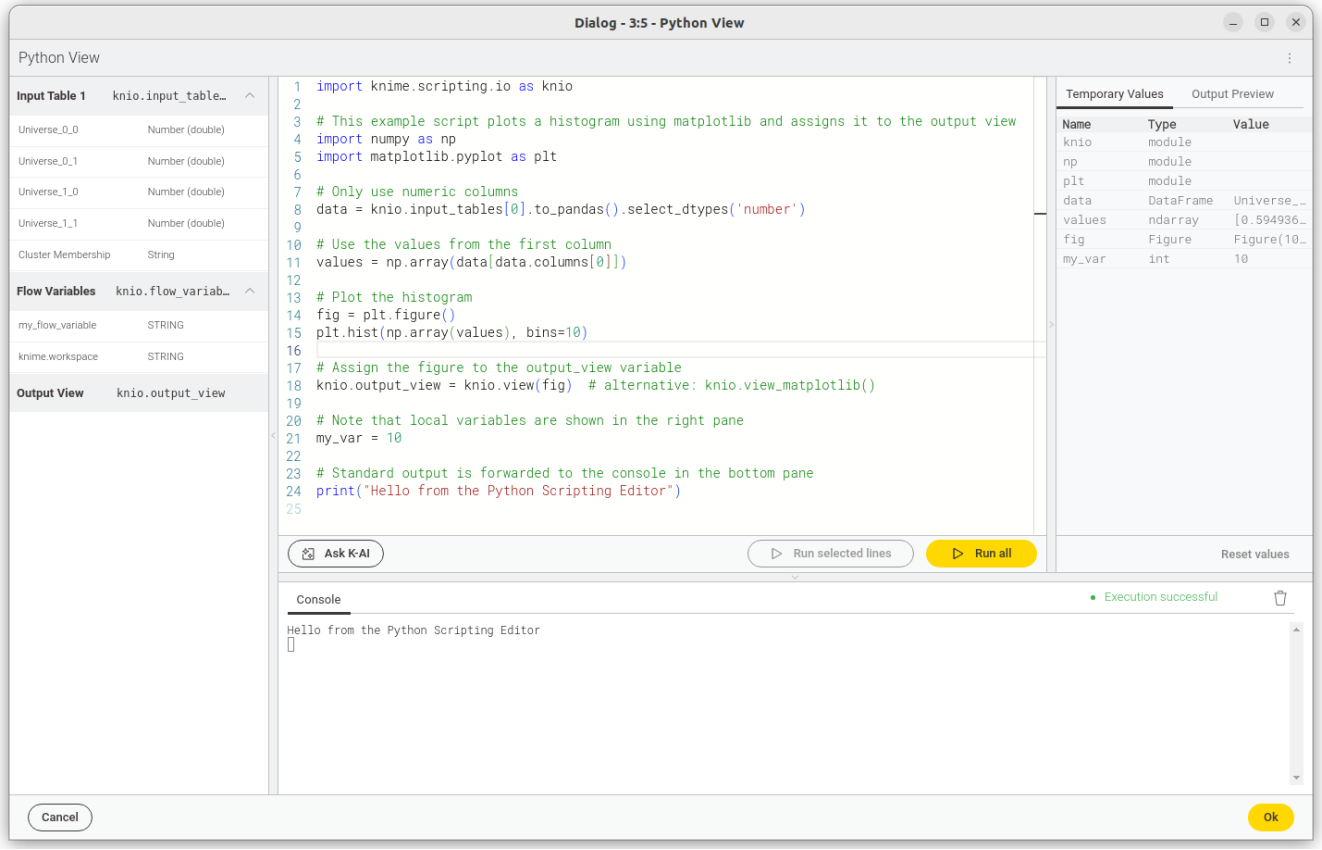
Einleitung

Dieses Kapitel führt durch die Konfiguration des Skript-Dialogs und die Anzahl der Ports, gefolgt von Nutzungsbeispielen. Diese Beispiele decken den Zugriff von Eingabedaten ab, gefolgt von Tabellenkonvertierung und Verwendung von Chargen für Daten größer als RAM. Dann wird es erklären, wie um Skripte von Python Legacy-Knoten zu dieser Erweiterung zu portieren. Danach die zusätzlichen Merkmale des Python View node werden erklärt. Das Kapitel schließt mit dem Gebrauchsfall der Beladung und auf Jupyter Notebooks zugreifen.

Siehe KNIME Hubraum für Beispiele zur Verwendung der Python-Knoten.

Konfiguration

Der Python Script-Knoten und der Python View-Knoten enthalten mehrere Panels in der Konfiguration Dialog.



- **Script Editor**

  Ihr primärer Bereich für die Codeentwicklung ist der Script Editor. Es kommt mit dem Bequemlichkeit der Auto-Vervollständigung, um Ihren Kodierungsprozess zu beschleunigen. Zusätzlich, Hovering über Funktionen oder Methoden zeigt Tooltips, die Nutzungsführung.

- **Eingaben/Ausgaben** (Left Panel)

  Hier werden die Eingabe- und Ausgabevariablen angezeigt, die Ihrem Knoten zugänglich sind. Du kannst diese einfach in Ihr Skript integrieren, indem Sie sie aus dem Panel in das Skript ziehen Editor.

- **Fragen K-AI**

  Tippen Sie auf AI für Code-Hilfe. Geben Sie eine Aufforderung in die Box "Ask K-AI" ein und unser KI-Modell wird den für Ihre Aufforderung relevanten Code vorschlagen. Überprüfen Sie den generierten Code und, wenn es trifft Ihre Anforderungen, integrieren Sie es in Ihr Skript.

- **Ausführungskontrollen** ("Run all", "Run select lines")

  Die Schaltfläche "Run all" ermöglicht die Ausführung Ihres gesamten Skripts in einem neuen Python Verfahren, das nach der Ausführung zugänglich bleibt. Um ein bestimmtes Segment Ihrer Code, wählen Sie die gewünschten Zeilen und klicken Sie auf "Ausgewählte Zeilen ausführen", indem Sie sie in der aktiver Python-Prozess.

- **Vorübergehende Werte**

  Dieses Panel listet die in Ihrem Skript definierten lokalen Variablen auf. Es ist nicht nur für zeigen; Sie können mit diesen Variablen interagieren, indem Sie auf sie klicken und ihre Werte angeben in der Konsole gedruckt werden. Diese interaktive Funktion ist besonders nützlich für schnelle variable Inspektionen und Debugging.

- **Anmerkung**

  Die Konsole zeigt den Echtzeit-Standard-Ausgang aus Ihrer Python-Sitzung, einschließlich Druckaussagen und andere Skriptausgänge. Um die Konsole zu starten oder zu decluttern, verwenden Sie die Müll-Symbol-Taste oben rechts.

- **Ausführungsstatus**

  Dieser Abschnitt liefert Feedback zum Ausführungsprozess des Skripts. Es zeigt den Status des letzten Skriptlaufs, so dass Sie bestätigen, dass das Skript wie beabsichtigt ausgeführt wurde oder zu identifizieren, ob es irgendwelche Handlungen erforderlich sind, um Script-Probleme anzusprechen.

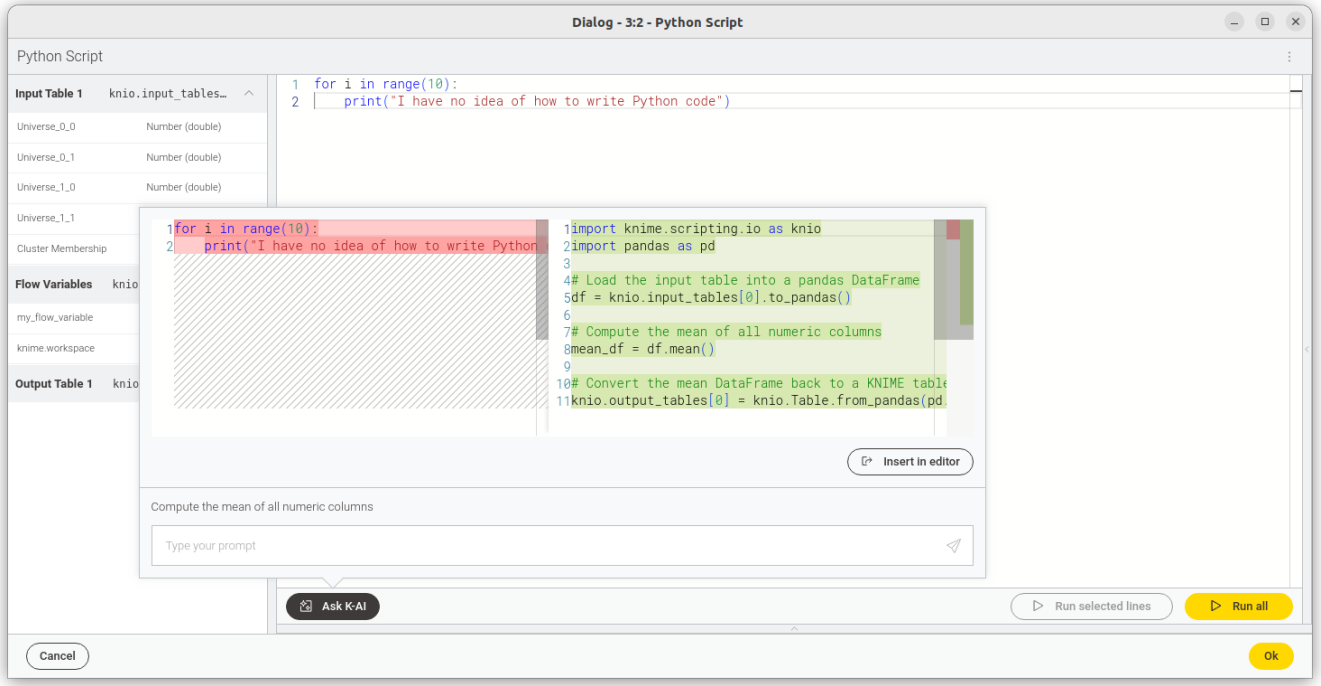
- **Vorschau**

  Das Output Preview Panel ist nur im Dialog des Knotens "Python View" sichtbar und zeigt die Ausgabeansicht nach der Skriptausführung. Diese interaktive Vorschau wird auf der Flug, wenn die Ausgabeansicht durch die interaktive Python-Sitzung aktualisiert wird.

KI-gestützte Codeerzeugung

Die "Ask K-AI"-Funktion im KNIME Python Scripting Node ist eine fortgeschrittene AI-assisted Code-Generierung Werkzeug. Bei der Aktivierung können Sie Eingabeaufforderungen eingeben, die die beabsichtigte Funktionalität des Codes. Der KI-Assistent hat das kontextuelle Bewusstsein des KNIME Python API, die Struktur der Eingabedaten und der aktuelle Skriptinhalt im Editor.

Sobald der Assistent den Code generiert, wird er Ihnen in einem diff-editor-Format präsentiert, das unterstreicht die Unterschiede zwischen Ihrem aktuellen Code und dem neuen Vorschlag. Sie haben dann die Möglichkeit, diese Vorschläge zu überprüfen und zu wählen, ob Sie sie in Ihrem Skript akzeptieren oder sie verwerfen, eine hohe Kontrolle über die Änderungen an Ihrem Code.



Wenn Sie diesen Dienst nutzen, beachten Sie, dass der aktuelle Code des Editors, die das Schema der Eingabedaten, und die Aufforderung wird über das Internet an das konfigurierte KNIME Hub und OpenAI, die eine Berücksichtigung für die Privatsphäre von Daten ist. Diese Übertragung ist für die KI notwendig, um Codevorschläge genau zu gestalten den Kontext Ihres Skripts und die Daten, mit denen Sie arbeiten.

Beispiele für die Nutzung

Wenn Sie eine neue Instanz der Python Script-Knoten erstellen, wird der Code-Editor bereits den Startercode enthalten, in dem wir `Import knime.scripting.io als knio` . Der gezeigte Inhalt in den Eingangs-, Ausgangs- und Durchflussgrößenscheiben über diese zugegriffen werden können `mime.scripting.io` Modul.

Die `mime.scripting.io` Modul ist immer verfügbar, wenn Sie das "Python" verwenden Script-Knoten. Es muss nicht manuell installiert werden, sondern wird dem "PYTHONPATH" automatisch.

Wenn das Paket `Knospen` wird über `Pip` in der Umgebung des Python Script-Knoten, Zugriff auf den `mime.scripting.io` Modul wird scheitern mit dem Fehler `Nein Modul namens 'knime.scripting'; 'knime' ist kein Paket` . In diesem Fall `Laufen pip deinstallieren Sie knime` in Ihrer Python-Umgebung.

Zugriff auf Daten

mit `Import knime.scripting.io als knio` , die Eingabe- und Ausgabetabellen und Objekte können Zugriff auf die jeweiligen Python-Listen:

- (i) und (i) ,
- knio.input_objects[i] und knio.output_objects[i] ,
- knio.output_images[i] zur Ausgabe von Bildern, die entweder eine Zeichenkette sein müssen, die eine Bild (SVG) oder ein Byte-Array, das ein Bild (PNG) kodiert,

  wenn I ist der Index der entsprechenden Tabelle/Objekt/Bild (0) für den ersten Eingangs-/Ausgangsport, 1 für den zweiten Ein-/Ausgangsport und so weiter).

Aus dem Wörterbuch können Flussgrößen aufgerufen werden:

- knio.flow_variables['name_of_flow_variable'] .

Tabellen in und aus Pandas DataFrames und PyArrow Tabellen umrechnen

Die mime.scripting.io Ein Modul bietet eine einfache Möglichkeit, die Eingabedaten als ein Pandas DataFrame oder PyArrow Tisch . Dies kann sich durchaus als nützlich erweisen, da die beiden Daten Darstellungen und entsprechende Bibliotheken bieten einen anderen Satz von Werkzeugen, die auf verschiedene Anwendungsfälle anwendbar.

- Tabellen in und aus einem Pandas DataFrame umrechnen:

```
df = knio.input_tables[0].to_pandas()
```

```
knio.output_tables[0] = knio.Table.from_pandas(df)
```

- Tabellen in und aus einer PyArrow-Tabelle umrechnen:

```
Tabelle = knio.input_tables[0].to_pyarrow()
```

```
knio.output_tables[0] = knio.Table.from_pyarrow(table)
```

Arbeiten mit Chargen

Die Python-Knoten, zusammen mit den mime.scripting.io Modul, effizient erlauben Verarbeitung größerer Datentabellen durch Batch.

ANHANG Zunächst müssen Sie eine Instanz einer Tabelle initialisieren, auf die die Chargen geschrieben werden

nach der Verarbeitung:

```
verarbeitet_table = knio.BatchOutputTable.create()
```

2. Rufen Sie die Ansätze() Verfahren auf einer Eingabetabelle gibt ein iterable zurück, dessen Elemente

Ansätze der Eingabetabelle, die über eine für Schleife:

```
verarbeitet_table = knio.BatchOutputTable.create()
für Charge in knio.input_tables[0].batches():
```

3. Im Inneren der für Schlaufe, kann die Charge in einen Pandas DataFrame oder einen PyArrow umgewandelt werden

Tabelle mit Methoden to_pandas() und to_pyarrow() oben erwähnt:

```
verarbeitet_table = knio.BatchOutputTable.create()
für Charge in knio.input_tables[0].batches():
input_batch = batch.to_pandas()
```

L 347 vom 20.12.2013, S. 1). Am Ende jeder Iteration der Schleife sollte der Ansatz an die

Verarbeitendes Gewerbe :

```
verarbeitet_table = knio.BatchOutputTable.create()
für Charge in knio.input_tables[0].batches():
input_batch = batch.to_pandas()
# process the batch
process_table.append(input_batch)
```

Porting Scripts aus dem Python Script (Legacy) Nodes

Anpassung Ihrer Python-Skripte von Python Script (Legacy) Knoten, um mit dem aktuellen

Python-Knoten sind so einfach wie das Hinzufügen der folgenden zu Ihrem Code:

```
Import knime.scripting.io als knio
input_table_1 = knio.input_tables[0].to_pandas()

# the script from the legacy nodes goes here

knio.output_tables[0] = knio.Table.from_pandas(output_table_1)
```

Beachten Sie, dass die Nummerierung von Eingängen und Ausgängen in den Python-Knoten 0-basiert ist - bedenken Sie, wenn Sie Ihre Skripte von den anderen Python-Knoten portieren, die ein 1-basiertes Numerierungssystem (z. nio.input_tables[0] im Python Knoten entspricht In den Warenkorb in den alten Python-Knoten).

Merkmale des Python View node

Der Python Der View-Knoten kann verwendet werden, um Ansichten mit Python-Skripten zu erstellen. Es hat dasselbe konfigurierbare Eingabe-Ports als Python Script-Knoten und verwendet dieselbe API, um auf die Eingabedaten. Allerdings hat der Python View-Knoten keine Ausgangsports außer einem optionalen Bildausgabeport.

Um eine Ansicht zu erstellen, muss das Skript die Variable bevölkern englisch.de mit einem Rückgabewert von einer der en.view* Funktionen. Es ist möglich, Ansichten von allen Arten der Anzeige zu erstellen Objekte über das Bequemlichkeitsverfahren en.view , die versucht, das richtige Format zu erkennen und ruft die passende Methode der folgenden Liste von en.view* Funktionen (siehe API für mehr Angaben:

- en.view ruft die entsprechenden folgenden Funktionen auf
- Pressemitteilungen erstellt eine Ansicht aus einer Reihe von HTML-Inhalte
- englisch. erstellt eine Ansicht aus einer Saite von svg Inhalt
- englisch: schafft einen Blick von Bytes, die einen Png repräsentieren
- Das ist nicht möglich. erstellt eine Ansicht von Bytes, die ein jpeg
- Pressemitteilungen erstellt einen Blick aus der aktiven oder gegebenen Matplotlib-Figur
- englisch. schafft einen Blick von der aktiven oder gegebenen Seefigur
- en.view_plotly erstellt einen Blick aus einer Figur; beachten Sie, dass die Auswahl zwischen der Ansicht und anderen KNIME-Ansichten synchronisieren, die Figurenspuren müssen auf die RowID gesetzt werden

  Beispiel:

  ```
  fig = px.scatter(df, x="my_x_col", y="my_y_col", color="my_label_col",
  benutzerdefinierte_data=[df.index]) # custom_data is set to the RowID
  node_view = view_plotly(fig)
  ```

- Pressemitteilungen erstellt eine Ansicht aus einem Objekt mit einem IPython _repr_*_ Funktion

Um ein Ausgabebild zu erstellen, muss der optionale Ausgabebildport hinzugefügt werden.

Der Ausgabebildport wird automatisch abgespeichert, wenn die Ansicht ein SVG-, PNG- oder JPEG-Bild ist oder kann in eine umgewandelt werden. Matplotlib und Seefiguren werden in ein PNG oder SVG umgewandelt Bild abhängig vom gewählten Format View_matplotlib` . Zahlen können nur in Bilder umgewandelt, wenn das Paket kaleido in der Umgebung installiert ist. Objekte, die ein IPython repressor_svg , Artikel 2 , oderDer Präsident Funktion wird durch Aufruf des ersten von diese Funktionen verfügbar. HTML-Dokumente können nicht automatisch in Bilder umgewandelt werden. Es ist jedoch möglich, eine Bilddarstellung oder eine Funktion einzustellen, die ein Bild zurückgibt Darstellung beim Aufruf Das ist der Fall.(siehe API)

Andernfalls muss das Skript die Variable bevölkern knio.output_images[0] wie im Python Script node.

Jupyter Notebooks von KNIME laden

Vorhandene Jupyter Notebooks können innerhalb von Python Scripting-Knoten aufgerufen werden, wenn wir Einfuhr knime.scripting.jupyter als knupyter . Notizbücher können über die Funktion geöffnet werden knupyter.load_notebook , die ein Standard-Python-Modul zurückgibt. Die Lastenheft Funktion benötigt den Pfad zum Ordner, der die Notebook-Datei und den Dateinamen der Notebook als Argumente. Nachdem ein Notebook geladen wurde, können Sie Funktionen anrufen, die definiert in den Codezellen des Notebooks wie jede andere Funktion eines Python-Moduls. Darüber hinaus können Sie den Textinhalt jeder Zelle eines Jupyter-Notizbuchs unter Verwendung der Funktion knupyter.print_notebook . Es nimmt die gleichen Argumente wie die Lastenheft Funktion.

Ein Beispielskript für einen Python Script-Knoten, der ein Notebook geladen hat, könnte so aussehen:

```
# Pfad zum Ordner, der das Notebook enthält, z.B. den Ordner 'Daten'
# in my workflow Ordner
notebook_directory = "knime://knime.workflow/data/"

# Dateiname des Notebooks
notebook_name = "sum_table.ipynb"

# Laden Sie das Notebook als Python-Modul
Import knime.scripting.jupyter als knupyter
my_notebook = knupyter.load_notebook(notebook_directory, notebook_name)

# Drucken Sie seine Textinhalte
knupyter.print_notebook(notebook_directory, notebook_name)

# Rufen Sie eine im Notebook definierte Funktion 'sum_each_row' an
output_table = my_notebook.sum_each_row(input_table)
```

Die Lastenheft und Print_notebook Funktionen haben zwei optionale Argumente:

- Anmerkungbook_version : Das Jupyter-Notebook-Format große Version. Manchmal die Version kann nicht aus einer Notebook-Datei gelesen werden. In diesen Fällen erlaubt diese Option die Angabe der erwartete Version, um Kompatibilität Problem zu vermeiden und sollte eine ganze Zahl sein.
- nur_include_tag : Nur Ladezellen, die mit dem angegebenen benutzerdefinierten Zelltag annotiert werden (seit Jupyter 5.0.0). Dies ist nützlich, um Zellen zu markieren, die in einem Python Modul. Alle anderen Zellen sind ausgeschlossen. Dies ist beispielsweise hilfreich, um Zellen auszuschließen, die Visualisierung oder enthalten Demo-Code und sollte ein String sein.

Die Jupyter-Notebook-Unterstützung für die KNIME Python Integration hängt von die Pakete IPython , nbformat , und Scipe , die bereits im gebündelte Umgebung und Metapaket knime-python-scripting .

Sie können Beispiel-Workflows mit der knime.scripting.jupyter Python Modul auf der KNIME Hubraum .

Konfigurieren der Python-Umgebung (Erweitert)

Die KNIME Python Integration erfordert eine konfigurierte Python-Umgebung. In diesem Abschnitt beschreiben, wie Sie die Python-Integration installieren und wie Sie ihre Python-Umgebung konfigurieren können.

Dieser Abschnitt ist nur relevant, wenn Sie etwas anderes als die <a href="#page13" style="color: #ff6600; text-decoration: underline;">vorinstallierte Umgebu .

Im folgenden Abschnitt führen wir Sie durch die Einrichtung der wesentlichen Werkzeuge, einschließlich Conda, um die KNIME Python Integration zu konfigurieren, um eine benutzerdefinierte Python-Umgebung zu verwenden.

Conda Disambiguation und Lizenzierung:

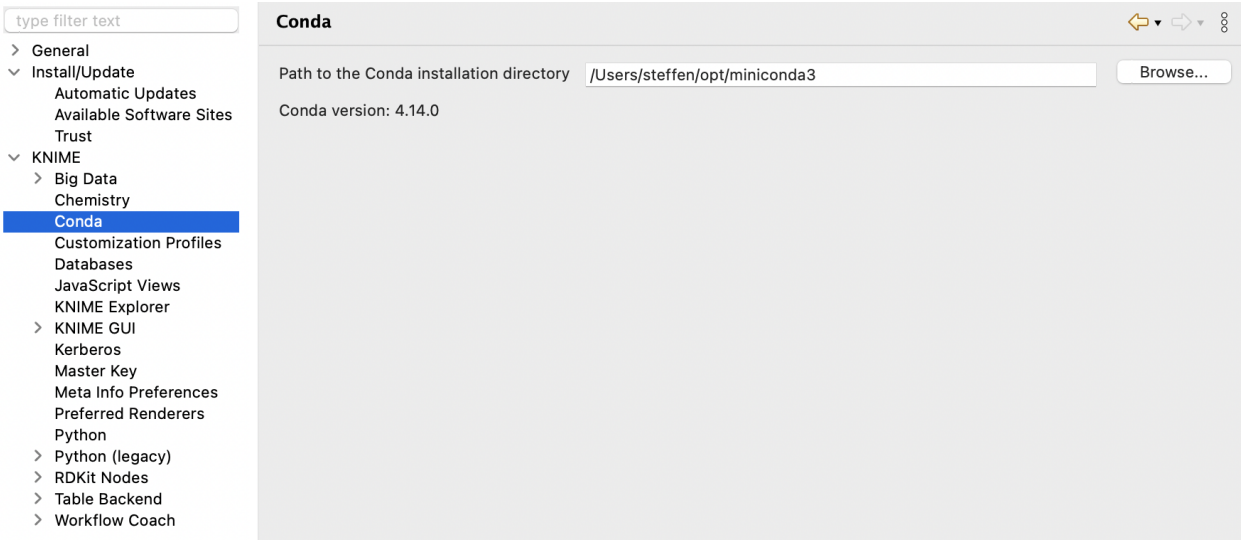
- Conda ist der Name des von Anaconda. Dieses Tool ist Open Source und kostenlos zu verwenden.
- conda-forge ist eine Open Source und kostenlos Kanal von Python und R Pakete .
- Anaconda ist ein Unternehmen, das auch die Standard Kanal und gibt die Anaconda-Verteilung frei, die Python-Pakete von die Standard Kanal. Dieser Kanal unterliegt Anacondas Bedingungen Bedingungen . Anaconda veröffentlicht eine Blog-Post zur Klärung das.
- mamba und Mikromamba sind Open-Source-Umsetzungen von conda. Conda vor kurzem auf die mamba Implementierung zur Lösung Umwelt, weil die mamba Implementierung ist viel schneller.
- Miniconda ist ein freier Installateur für conda von Anaconda zur Verfügung gestellt. Es ist konfiguriert, um die Standard Kanal (der Anaconda unterliegt) TOC) bei der Erstellung von Umgebungen standardmäßig.
- Miniforge ist ein freier Installer für conda, der nur konfiguriert ist Pakete aus der conda-forge Kanal, also, es sei denn, Sie tun das ausdrücklich. wird niemals Pakete verwenden, die Anacondas TOC unterliegen.

Neben den Voraussetzungen erklären wir Möglichkeiten für zwei verschiedene Bereiche: für die ganze KNIME Analytics Plattform und knotenspezifisch. Letzteres ist praktisch, wenn Sie Ihren Workflow teilen. Schließlich die Konfiguration für den KNIME Executor (der im KNIME Business Hub verwendet wird) wird im Konfigurationsbeispiel erläutert.

Voraussetzungen

ANHANG Installieren Sie die Python-Erweiterung. Ziehen und fallen Erweiterung vom KNIME Hub in der zu installieren. Oder muss Datei → KNIME installieren Erweiterungen in KNIME Analytics Plattform und Installation KNIME Python Integration in der Kategorie KNIME & Erweiterungen .

2. Installieren Sie Conda, ein Paket und Umweltmanager. Zum Beispiel Miniforge , die a minimale Installation von Conda, konfiguriert, um niemals Pakete zu verwenden, die Anacondas TOC. Die erste Umgebung, Basis , enthält eine Python-Installation, aber wir empfehlen neue Umgebungen für Ihre speziellen Anwendungsfälle zu schaffen. Im KNIME Analytics Plattformeinstellungen konfigurieren Sie die Pfad zum Conda-Installationsverzeichnis unter KNIME > Conda , wie in der folgenden Abbildung dargestellt.



Sie müssen den Pfad in den Ordner mit Ihrer Installation von Conda zur Verfügung stellen. Für Miniforge, der Standard-Installationspfad ist

- für Windows: C:\Benutzer\your-username>\miniforge3\
- für Mac: /Benutzer//miniforge3
- für Linux: /home//miniforge3

Sobald Sie einen gültigen Pfad eingegeben haben, wird die installierte Conda-Version angezeigt.

wie man Umgebungen ohne Conda.

Konfigurieren der AP-weiten Umgebung

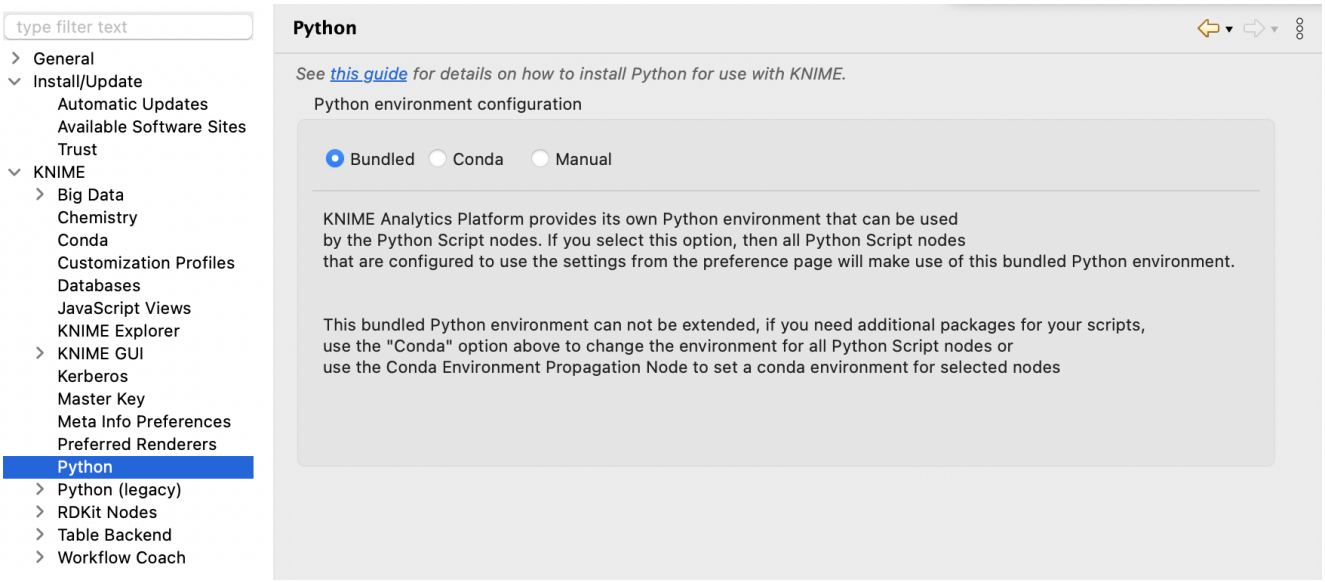
Gebündelt (verpflichtet, sofort zu starten)

Die KNIME Python Integration ist mit einer gebündelten Python-Umgebung, bestehend aus einer spezifischer Satz von Python-Paketen (d.h. Python-Bibliotheken) sofort zu starten: öffnen Sie einfach die Python Script node und start scripting.

Da nicht jeder alles braucht, ist dieses Set ziemlich begrenzt, um viele Skripte zu ermöglichen Szenarien, während die gebündelte Umgebung klein bleibt. So die Liste der enthaltenen Pakete kann gefunden werden in den Inhalten dieses Metapakets und in der folgenden Liste (mit einigen) zusätzliche Abhängigkeiten:


```
# Aktuelle Version in der gebündelten Umgebung
- Bezeichnung >=3.6,<3.7.0a0
- nomkl >=1.0,<1.1.0a0
- numpy >=1.26.4,<1.26.5.0a0
- Pandas >=2.0.3,<2.0.4.0a0
- Kissen >=11.0.0,<11.0a0
- py4j >=0.10.9,<0.10.0a0
- Pyarrow >=18.1.0,<18.1.1.0a0
- python >=3.11.10,<3.11.11.0a0
- python-dateutil >=2.9.0,<2.9.1.0a0
- python_abi 3.11.*
- schönsoup4 >=4.12.3,<4.12.4.0a0
- Cloudpickle >=3.1.0,<3.1.1.0a0
- Ipython >=8.29.0,<8.29.1.0a0
- matplotlib-base >=3.9.2,<3.9.3.0a0
- nbformat >=5.10.4,<5.10.5.0a0
- nltk >=3.9.1,<3.9.2.0a0
- openpyxl >=3.1.5,<3.1.6.0a0
- Grundstück >=5.24.1,<5.24.2.0a0
- python >=3.11,<3.12.0a0
- python_abi 3.11.*
- pytz >=2024.2,<2024.3.0a0
- pyyaml >=6.0.2,<6.0.3.0a0
- Anträge >=2.32.3,<2.32.4.0a0
- scikit-learn >=1.5.2,<1.5.3.0a0
- Empfänger >=1.14.1,<1.14,2.0a0
- Meeresfrüchte >=0.13.2,<0.13.3.0a0
- statsmodels >=0.14.4,<0.14.5.0a0
```

Die gebündelte Umgebung wird standardmäßig ausgewählt und kann hier wieder selektiert werden:



Metapakete über Terminal (empfohlen, wenn zusätzliche Pakete erforderlich sind)

Wenn Sie eine Python-Umgebung mit mehr als den Paketen von der gebündelten Umwelt, Sie können Ihre Umgebung mithilfe unserer Metapakete erstellen. Zwei Metapakete sind wichtig: knime-python-Basis enthält die Grundpakete, die immer benötigt werden. knime-python-scripting enthält knime-python-Basis und installiert zusätzlich die Pakete verwendet im Python Script-Knoten. Dies ist der Satz von Paketen, die auch im gebündelten Umwelt. Die Listen finden Hier. . Sie können zwischen verschiedenen Python-Version wählen (aktuell 3.9 bis 3.11) und wählen Sie die aktuelle KNIME Analytics Platform Version. Siehe KNIME conda Kanal für verfügbare Versionen.

Erstellen Sie eine neue Umgebung in einem Terminal durch Anpassung und Eingabe

```
conda create --name my_python_env -c knime -c conda-forge knime-python-scripting = 5.7
python = 3.11 other_package other_package_with_version_specified = 1.2.3
```

Installieren Sie zusätzliche Pakete in Ihrer bestehenden Umgebung my_python_env im Terminal durch Einstellen und Betreten

```
conda install --name my_python_env -c conda-forge
```

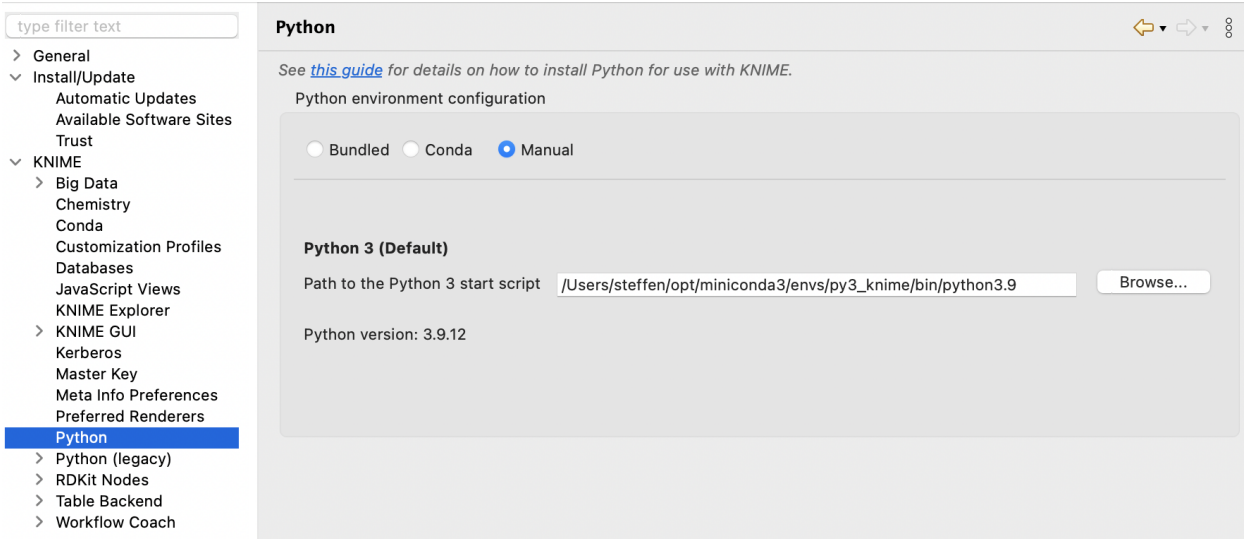
Weitere Informationen zur Verwaltung von Conda-Paketen finden Sie Hier. .

Dok. **nicht** das Paket installieren Knospen Verwendung Pip in die Umwelt, die verwendet in KNIME, da dies mit der KNIME Python Scripting API Konflikt und Importieren mime.scripting.io scheitern.

Manuelle Angabe des Python ausführbar/start-Skripts über die Präferenzseite

Die Alternative zur Verwendung des Conda-Paketmanagers ist die manuelle Einrichtung des Python Installation. Wenn Sie sich entscheiden **Handbuch** in der Auswahlseite haben Sie die folgenden Optionen:

ANHANG Punkt KNIME Analytics Platform to a Python ausführbar für Ihre Wahl



2. Punkt KNIME Analytics Platform zu einem Startskript, das die Umgebung aktiviert

   Diese Option geht davon aus, dass Sie einen geeigneten Python erstellt haben.

   Umgebung früher mit einem Python virtuellen Umweltmanager Ihrer Wahl. In Ordnung

   um die erstellte Umgebung zu nutzen, müssen Sie ein Startskript erstellen (Shell-Script auf Linux

   und Mac, Batch-Datei unter Windows). Das Skript muss die folgenden Anforderungen erfüllen:

   - Es muss Python mit den Argumenten des Skripts starten (bitte sicherstellen, dass

     dass Räume richtig entkommen sind)

   - Es muss Standard und Fehler aus der gestarteten Python Instanz ausgeben

   - Es darf nichts anderes ausgeben.

   Hier stellen wir ein Beispiel Shell-Skript für die Python-Umgebung auf Linux und Mac zur Verfügung.

   Bitte beachten Sie, dass auf Linux und Mac Sie zusätzlich die Datei ausführbar machen müssen

   (i.e. chmod gou+x py3.sh)

```
# /bin/bash
# Beginnen Sie, indem Sie sicherstellen, dass der anaconda-Ordner an der PATH
# so dass der Quellbefehl funktioniert.
# Das ist nicht nötig, wenn Sie schon wissen, dass
# the anaconda bin dir is on the PATH
Export PATH="/bin:$PATH"

In den Warenkorb < UMWELT_NAME>
python "$@" 1>&1 2>&2
```

   Unter Windows sieht das Skript so aus:


```
@REM Anpassung des Ordners in der PATH an Ihr System
@SET PATH=\Scripts;%PATH%
@CALL aktivieren || ECHO Aktivierung der Python-Umgebung gescheitert
@python %*
```

Dies sind beispielsweise Skripte für Conda. Sie müssen sie möglicherweise anpassen andere Werkzeuge durch Austausch der Conda-spezifischen Teile. Zum Beispiel wirst du müssen sie bearbeiten, um auf den Standort Ihrer Umgebung hinzuweisen Installation von Managern und Aktivieren die richtige Umgebung.

Nachdem Sie das Startskript erstellt haben, müssen Sie darauf hinweisen, indem Sie den Pfad zum Skript auf der Python Preferences Seite.

Knotenspezifische Umgebungen konfigurieren

Conda Environment Propagation Node

Neben der Einrichtung von Python für Ihren gesamten KNIME Workspace über die Optionsseite, Sie kann auch die Conda Environment Propagation Node benutzerdefinierte Python konfigurieren umgebungen und propagieren sie dann zu nachgeschalteten Python-Knoten. Dieser Knoten erlaubt auch Sie bündeln diese Umgebungen zusammen mit Ihren Workflows, so dass es für andere leicht, replizieren Sie die exakt gleiche Umgebung, in der der Workflow ausgeführt werden soll. Das macht Workflows mit Python-Knoten deutlich tragbarer und weniger fehleranfälliger.

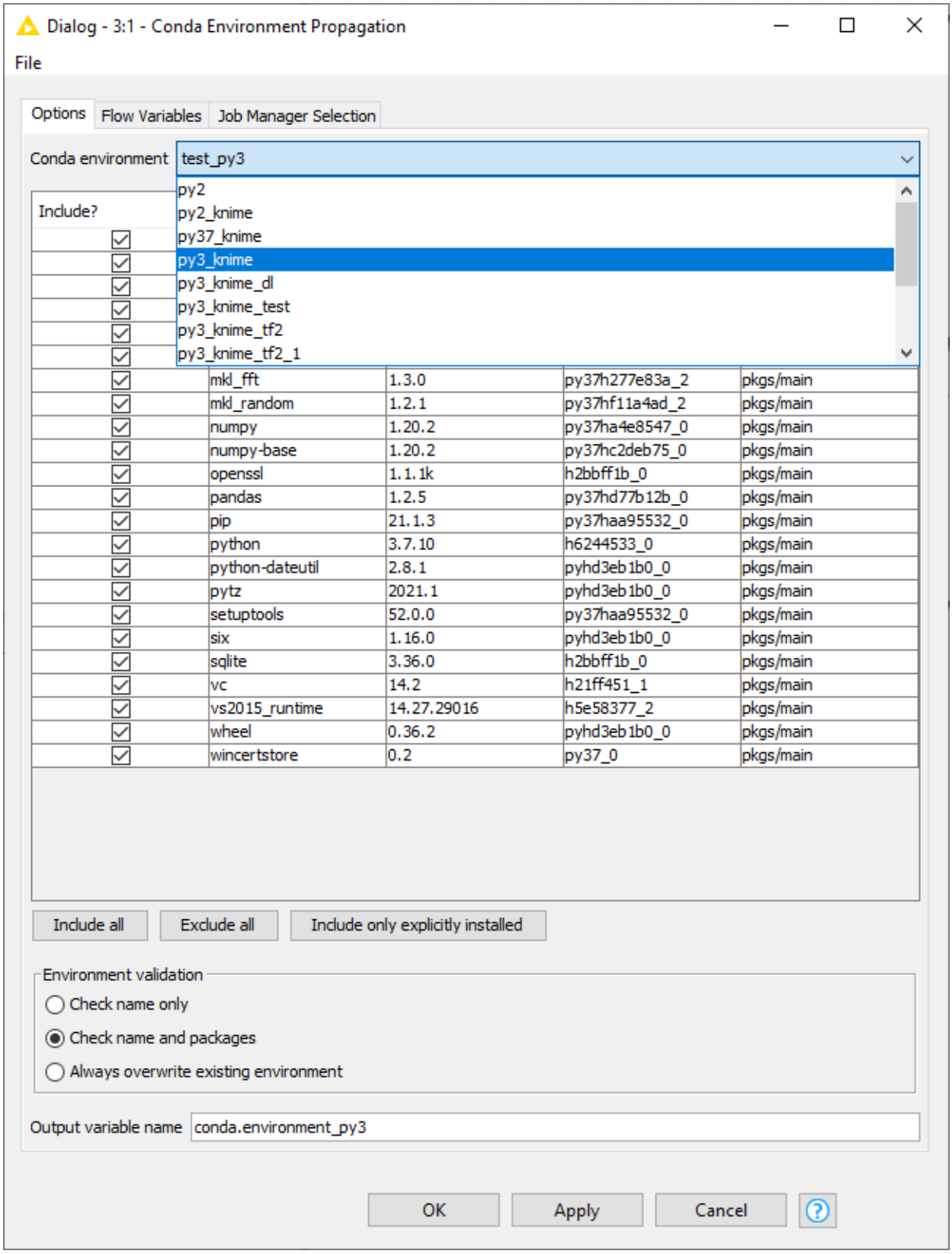
Einrichtung

Um den Conda Environment Propagation Knoten nutzen zu können, müssen Sie folgen Diese Schritte:

ANHANG Auf Ihrer lokalen Maschine sollten Sie Conda eingerichtet und in der Einstellungen der KNIME Python Integration wie in der <a href="#page12" style="color: Abschnitt

2. Öffnen Sie den Node-Konfigurationsdialog und wählen Sie die Conda-Umgebung aus, die Sie möchten propagieren und die Pakete, die in die Umwelt einzubeziehen sind, falls es wiederhergestellt wird auf einer anderen Maschine. Die Pakete können automatisch über folgende ausgewählt werden: Tasten:

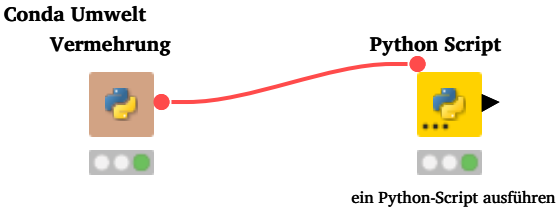
Die Nur explizit installiert enthalten Taste wählt nur die Pakete, die waren vom Benutzer explizit in die Umgebung installiert. Dies kann dazu beitragen, Konflikte zu vermeiden bei der Verwendung des Workflows auf verschiedenen Betriebssystemen, weil es Conda ermöglicht die Abhängigkeiten dieses Pakets für das Betriebssystem zu lösen, ist der Workflow weiterlaufen.



3. Der Conda Environment Propagation Knoten gibt eine Flussgröße aus, die die
   notwendige Informationen über die Python-Umgebung (d.h. den Namen der Umwelt
   und die jeweiligen installierten Pakete und Versionen). Die Strömungsgröße hat
   conda.environment als Standardname, aber Sie können einen benutzerdefinierten Namen angeben. Hier entlang
   Sie können Namenskollisionen vermeiden, die bei der Verwendung von mehreren Conda auftreten können
   Umwelt Propagationsknoten in einem einzigen Workflow.

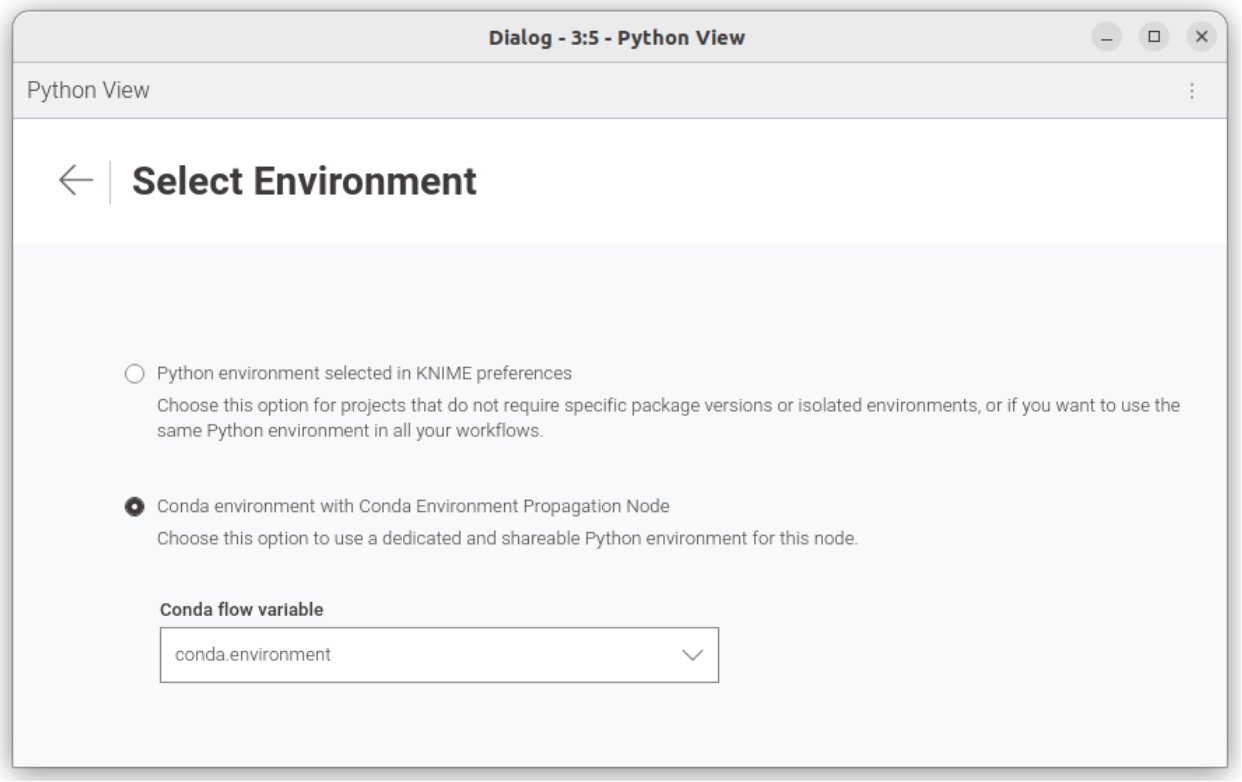
Damit jeder Python-Knoten im Workflow die gerade erstellte Umgebung nutzt, verwenden Sie
müssen:

ANHANG Verbinden Sie den strömungsvariablen Ausgangsport von Conda Environment Propagation node mit dem
Eingangsstrom variabler Port eines Python-Knotens



> Bitte beachten Sie, dass, da Flussgrößen auch durch
> Verbindungen, die keine strömungsvariablen Verbindungen sind, die strömungsvariablen
> Verbreitung der Conda-Umgebung, die Sie mit der Conda erstellt haben
> Umwelt Propagation Knoten wird auch für alle nachgeschaltet
> Knoten.

2. Öffnen Sie im Workflow erfolgreich den Konfigurationsdialog der Python-Knoten, der
   Sie wollen tragbar machen. Öffnen Sie die Einstellungsseite "Set Python" über die
   kebab Menü oben rechts und wählen Sie die gewünschte Conda-Flow-Variable aus.



Ausfuhr

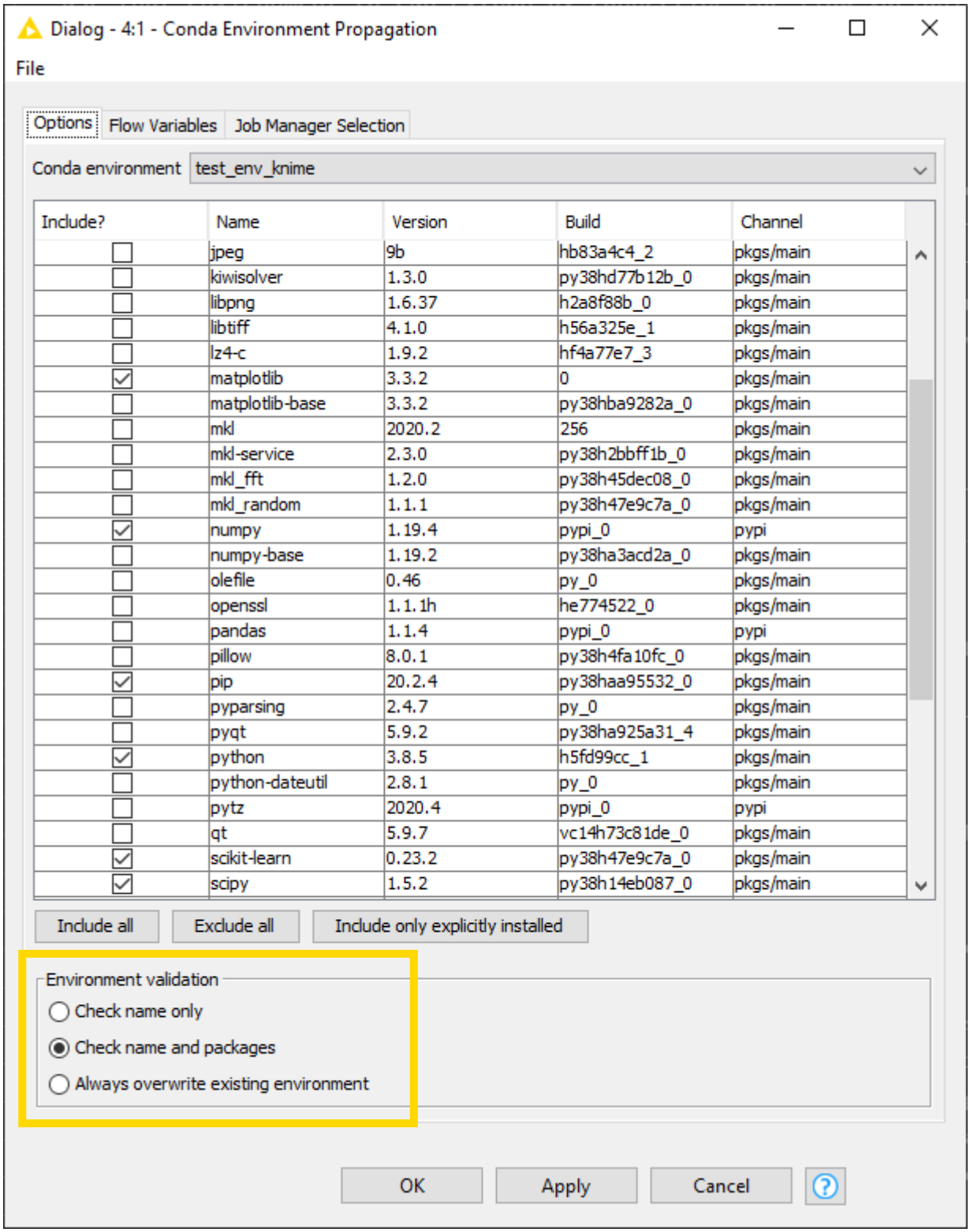
Sobald Sie den Conda Environment Propagation-Knoten konfiguriert und die gewünschte
Workflow, Sie möchten diesen Workflow auf einer Zielmaschine ausführen, zum Beispiel einem KNIME
Serverinstanz.

ANHANG Bereitstellung des Workflows durch Hochladen auf den KNIME Server, Teilen über den KNIME Hub,
oder exportieren. Stellen Sie sicher, dass der Conda Environment Propagation-Knoten vor
oder während des Bereitstellungsprozesses.

2. Auf der Zielmaschine muss Conda auch in den Einstellungen von
die KNIME Python Integration. Wenn die Zielmaschine einen KNIME Server betreibt, können Sie
um Ihren Serveradministrator zu kontaktieren oder sich auf die Serververwaltung Leitfaden in Ordnung
das zu tun.

3. Während der Ausführung (auf beiden Maschinen) wird der Knoten prüfen, ob eine lokale Conda
Umwelt existiert, die seiner konfigurierten Umgebung entspricht. Beim Konfigurieren der
node, Sie können wählen, welche Modalität für die Conda-Umgebungsvalidierung verwendet wird
auf der Zielmaschine. Nur Name prüfen nur die Existenz eines
Umgebung mit dem gleichen Namen wie das Original, Name und Pakete überprüfen wird
Überprüfen Sie den Namen und die gewünschten Pakete, während Immer überschreiben bestehende Umgebung
wird die Existenz einer gleichen Umgebung auf der Zielmaschine missachten und
Rekreieren Sie es.

Je nach obiger Konfiguration wird die Ausführungszeit des Knotens variieren. Zum Beispiel wird eine einfache Conda-Umgebungs-Namensprüfung viel sein schneller als ein Name und Paketcheck, die wiederum schneller als ein volle Umgebung Erholungsprozess.



Include?	Name	Version	Build	Channel
☐	jpeg	9b	hb83a4c4_2	pkgs/main
☐	kiwisolver	1.3.0	py38hd77b12b_0	pkgs/main
☐	libpng	1.6.37	h2a8f88b_0	pkgs/main
☐	libtiff	4.1.0	h56a325e_1	pkgs/main
☐	lz4-c	1.9.2	hf4a77e7_3	pkgs/main
☑	matplotlib	3.3.2	0	pkgs/main
☐	matplotlib-base	3.3.2	py38hba9282a_0	pkgs/main
☐	mkl	2020.2	256	pkgs/main
☐	mkl-service	2.3.0	py38h2bbff1b_0	pkgs/main
☐	mkl_fft	1.2.0	py38h45dec08_0	pkgs/main
☐	mkl_random	1.1.1	py38h47e9c7a_0	pkgs/main
☑	numpy	1.19.4	pypi_0	pypi
☐	numpy-base	1.19.2	py38ha3acd2a_0	pkgs/main
☐	olefile	0.46	py_0	pkgs/main
☐	openssl	1.1.1h	he774522_0	pkgs/main
☐	pandas	1.1.4	pypi_0	pypi
☐	pillow	8.0.1	py38h4fa10fc_0	pkgs/main
☑	pip	20.2.4	py38haa95532_0	pkgs/main
☐	pyparsing	2.4.7	py_0	pkgs/main
☐	pyqt	5.9.2	py38ha925a31_4	pkgs/main
☑	python	3.8.5	h5fd99cc_1	pkgs/main
☐	python-dateutil	2.8.1	py_0	pkgs/main
☐	pytz	2020.4	pypi_0	pypi
☐	qt	5.9.7	vc14h73c81de_0	pkgs/main
☑	scikit-learn	0.23.2	py38h47e9c7a_0	pkgs/main
☑	scipy	1.5.2	py38h14eb087_0	pkgs/main

Export von Python-Umgebungen zwischen Systemen, die verschiedene Betriebssysteme betreiben
Systeme könnten einige Bibliotheken zum Konflikt bringen. Bitte testen Sie Ihre Workflows an
anders Betriebssysteme und die Nutzung der Nur explizit enthalten
installiert Knopf.

Manuelle Angabe des ausführbaren/start-Skripts Python über die Durchflussvariable

Falls Sie die Funktionalität des Conda Environment Propagation-Knotens nicht nutzen möchten, Sie
kann auch einzelne Knoten manuell konfigurieren, um bestimmte Python-Umgebungen zu verwenden. Das ist
über die Durchflussgröße python3_command dass jeder Python-Skript-Knoten unter dem
Durchflussvariablen Tab in seinem Konfigurationsdialog. Die Variable akzeptiert den Pfad zu einem Python-Start
Skript wie in der Handtasche beschrieben.

Ausführungskonfiguration

Das KNIME Ausführen verwendet Anpassungsprofile , Sie können die folgenden Teile für Ihre
Komfort.

```
# A - KNIME Conda Integration - Pfad zu Anaconda/miniforge Installationsverzeichnis
# Diese Linie ist nur dann erforderlich, wenn conda/miniforge in einem von der
Standard,
# which is /home/knime/miniconda3/ in an executor image.
/instance/org.knime.conda/condaDirectoryPath=

# B - KNIME Python Integration - Standardoptionen für Python Integration. Standardmäßig
KNIME nutzt die gebündelte Umgebung (verschifft mit KNIME), wenn keine Conda Environment
Es wird Propagationsknoten verwendet.
# Linie unten kann entweder "gebündelt" (Standard), "conda" oder "manuell" eingestellt werden
/instance/org.knime.python3.scripting.nodes/pythonEnvironmentTyp=bundled
/instance/org.knime.python3.scripting.nodes/bundledCondaEnvPath=org_knime_pythonscriptin
g
# Folgende Zeilen werden nur benötigt, wenn "gebündelter" Wert oben durch "conda" ersetzt wird
/instance/org.knime.python3.scripting.nodes/python2CondaEnvironmentDirectoryPath=
Standard-Conda-Umgebung Dir>
/instance/org.knime.python3.scripting.nodes/python3CondaEnvironmentDirectoryPath=
Standard-Conda-Umgebung Dir>
# Folgende Zeilen werden nur benötigt, wenn "gebündelter" Wert oben durch "manuell" ersetzt wird
/instance/org.knime.python3.scripting.nodes/python2Path=
/instance/org.knime.python3.scripting.nodes/python3Path=

# C - KNIME Python Integration (Legacy) - Standardoptionen für Python Integration.
# Linie unten kann entweder auf "conda" oder "manuell" gesetzt werden
/instance/org.knime.python2/pythonEnvironmentTyp=conda
/instance/org.knime.python2/defaultPythonOption=python3
/instance/org.knime.python2/serializerId=org.knime.python2.serde.arrow
```

```
# Folgende Zeilen werden nur benötigt, wenn "conda" oben gesetzt wird
/instance/org.knime.python2/python2CondaEnvironmentDirectoryPath =
Umwelt
/instance/org.knime.python2/python3CondaEnvironmentDirectoryPath =
Umwelt
# Folgende Zeilen werden nur benötigt, wenn "conda" Wert oben durch "manuell" ersetzt wird
/instance/org.knime.python2/python2Path =
/instance/org.knime.python2/python3Path =



# D - KNIME Deep Learning Integration
# Wählen Sie entweder "python" oder "dl" (ohne Anführungszeichen) in der nächsten Zeile. Wenn "Python" ist
verwendet wird, wird die Konfiguration des obigen Abschnitts B wiederverwendet. Wenn "dl" verwendet wird, eine benutzerdefinierte config
für Deep Learning kann bereitgestellt werden.
/instance/org.knime.dl.python/pythonConfigSelection = python
# Nach Zeilen nur erforderlich, wenn Zeile oben auf "dl" gesetzt wird
/instance/org.knime.dl.python/kerasCondaEnvironmentDirectoryPath =
Umwelt
/instance/org.knime.dl.python/librarySelection = keras
/instance/org.knime.dl.python/manualConfig = python3
/instance/org.knime.dl.python/pythonEnvironmentTyp = conda
/instance/org.knime.dl.python/serializerId = org.knime.python2.serde.arrow
/instance/org.knime.dl.python/tf2CondaEnvironmentDirectoryPath =
Umwelt
/instance/org.knime.dl.python/tf2ManualConfig = python3
```

Fehlerbehebung

Falls Sie Probleme mit der Python-Integration von KNIME haben, sind hier einige nützliche Tipps, um zu helfen
Sie sammeln mehr Informationen und vielleicht sogar das Problem selbst lösen. Im Falle von Problemen
weiter bestehen und Sie um Hilfe bitten, bitte die gesammelten Informationen einschließen.

Debug Informationen finden

Ressourcenreiche Informationen helfen beim Verständnis von Problemen. Nachteilige Informationen können erhalten werden
auf folgende Weise.

Zugang zum KNIME Log

Die wohnzimmer.de enthält während der Ausführung von Knoten eingeloggte Informationen. Um es zu erhalten, dort
sind zwei Wege:

- In der KNIME Analytics Plattform: Blick → KNIME Protokoll öffnen

• In der Datei Explorer: /.metadata/knime/knime.log

Nicht alle eingeloggten Informationen sind erforderlich. Bitte beschränken Sie die Informationen, die Sie der Frage. Wenn die Log-Datei keine ausreichenden Informationen enthält, können Sie die Protokollierung ändern Verbenheit in Datei → Vorlieben → KNIME . Sie können sogar die Informationen an der Konsole anmelden im KNIME Analytics Programm: Datei → Vorlieben → KNIME → KNIME GUI .

Informationen Ã1/4ber die Python-Umgebung

wenn Conda wird verwendet, um die Informationen über die verwendete Python-Umgebung zu erhalten über:

ANH ANG conda aktivieren

2. conda env export

Informationen Ã1/4ber eine gescheiterte Installation

Wenn der Fehler **Ein Fehler beim Installieren der Artikel auftritt** erscheint bei der Installation Erweiterung mit gebündelter Python-Umgebung (die KNIME Python Integration selbst und rein Python Erweiterungen), Sie können die entsprechenden Log-Dateien wie folgt erhalten. Die Fehlermeldung enthält wie org.knime.pythonscripting.channel.v1.bin ... oder sdl.harvard.geospatial.channel.bin ...

ANHANG Windows/Linux: Gehen Sie in den Ordner der KNIME Analytics Platform Installation

macOS: Rechtsklick auf die Installation der KNIME Analytics Platform und Paket anzeigen Inhalt , öffnen Sie den Ordner Eclipse

2. Plugins → → Bin

3. Die Protokolldateien sind erstellen_env.err und erstellen_env.out

Was tun, wenn der Fehler "Kein Modul namens knime.api"

Wenn Sie den Fehler sehen

```
ModulNotFoundError: Kein Modul namens 'knime.api'; 'knime' ist kein Paket`
```

Sie haben wahrscheinlich das Paket Knospen installiert über Pip in der Umgebung des Python Script Node. Dies funktioniert derzeit nicht aufgrund eines Namensspiels. Sie können entfernen Knospen in der jeweilige Python-Umgebung durch Ausführung des Befehls pip deinstallieren Sie knime in deiner

Terminal.

Es kann mehrere Pakete wie die folgenden anzeigen. Sie können beide entfernen.

```
... \envs\py3_knime\lib\site-Pakete\knime-0.11.6.dist-info*
... \envs\py3_knime\lib\site-Pakete\knime.py
```

Windowsspezifische Probleme

- Installation scheitert - mögliches Problem: der Installationsordner der KNIME Analytics Platform hat einen langen Weg. Windows' langen Pfad Einschränkungen können durch die lange Aktivierung umgangen werden Pfadunterstützung wie hier beschrieben: https://docs.microsoft.com/en-us/windows/win32/fileio/maximum-file-path-limitation?tabs=Registry

Datentyp nicht unterstützt

Wenn Sie einen Fehler wie folgt erhalten, können Sie den Datentyp über `df["count" = pd.to_numeric(df["count"])` oder einen Blick haben in diesem Fehlerbereich .

```
ValueError: Der Datentyp 'uint32' in Spalte 'count' wird in KNIME Python nicht unterstützt.
Bitte verwenden Sie einen anderen Datentyp.
```

SSL-Fehler bei der Ausführung

Wenn Sie einen SSL-Fehler während der Ausführung eines Python-Skripting-Knotens erkennen, könnte dies sein aufgrund der Verwendung eines selbstsignierten Zertifikats. Wenn andere Knoten wie der GET Request-Knoten funktionieren, aber der Python Script-Knoten nicht, Sie können die Python Script-Knoten konfigurieren, um dem gleiche Zertifikate wie die KNIME Analytics Platform. Um dies zu tun, fügen Sie die folgende Zeile zu Ihrem `knime.ini` Datei:

```
-Dknime.python.cacerts=AP
```

Dies wird die `CA_CERTS` und `ANFORDERUNGEN` Umgebungsvariablen zu einem neu erstelltes CA-Bündnis, das die zertifizierenden Behörden der KNIME Analytics Platform enthält Vertrauen. Der Python Script-Knoten vertraut dann den gleichen Zertifikaten wie die KNIME Analytics Plattform.

Einrichtung eines Ausführers, der denselben Zertifikaten wie die KNIME Analytics vertraut Plattform

Um dies in einem Ausführung KNIME Business Hub, Sie können diese Schritte folgen:

ANHANG Holen Sie sich die Ausführungskontext-ID des Ausführenden.

```
GET api./execution-contexts
```

2. Sende eine PUT Antrag auf folgenden Endpunkt:

```
PUT api./execution-contexts/
```

mit der folgenden JSON-Körper:

```
{\r
HFF
FF}
"Betriebsinformation": {
"VmArguments": [
"-Dknime.python.cacerts=AP"
!
}
}
```

KNIME AG
Talacker 50
8001 Zürich, Schweiz
www.knime.com
Info@knime.com