

Maximizing the Potential of Data Science with KNIME and Python

Paolo Tamagnini

paolo.tamagnini@knime.com

Mahantesh Pattadkal

mahantesh.pattadkal@knime.com

Carsten Haubold

carsten.haubold@knime.com

April 19, 2023



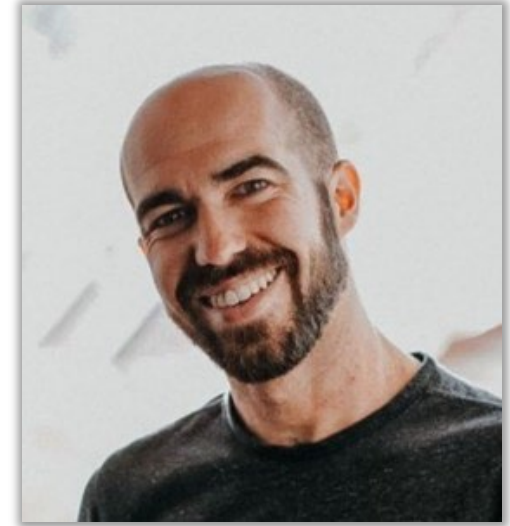
Presenters



Paolo Tamagnini
Data Scientist

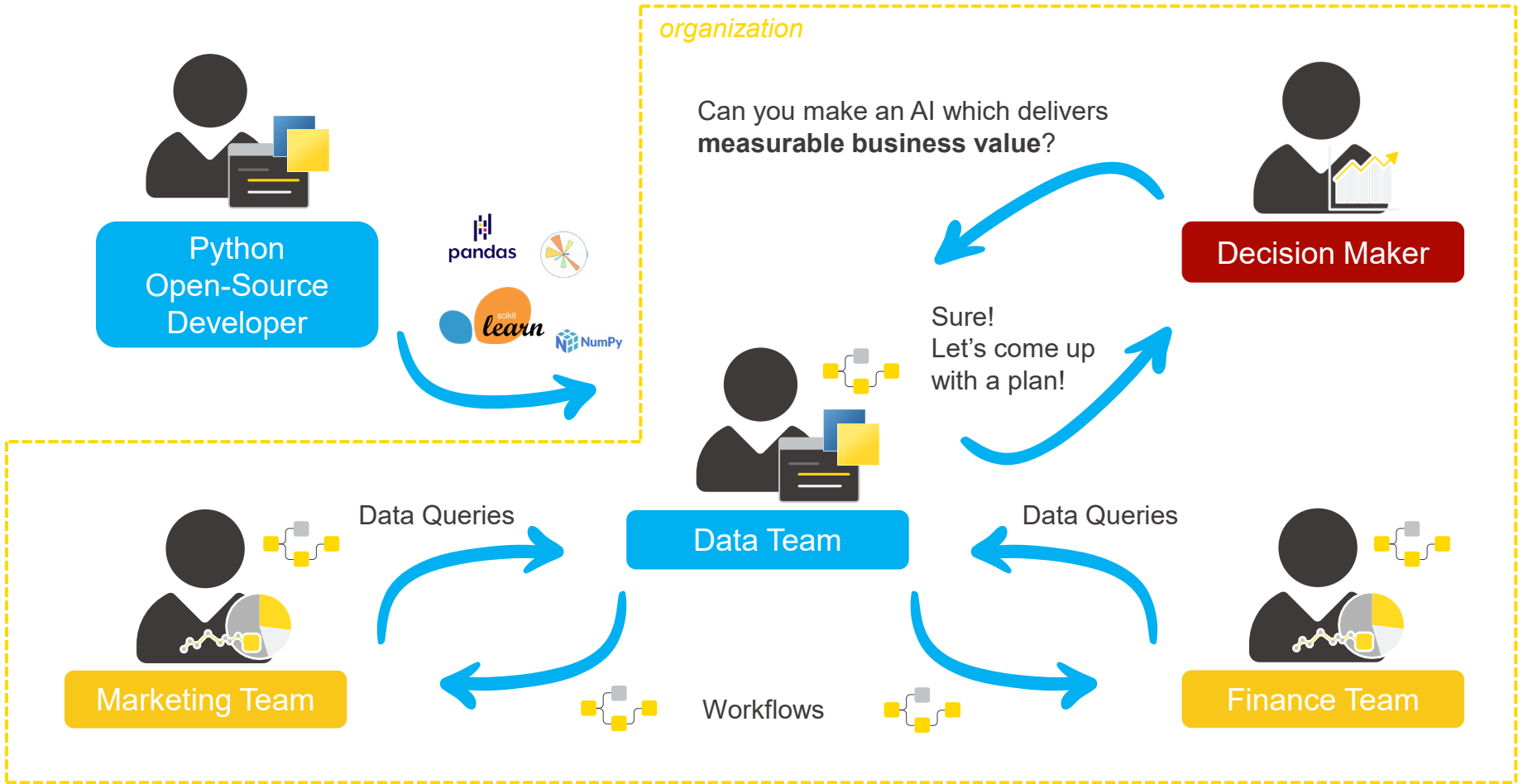


Mahantesh Pattadkal
Data Scientist



Carsten Haubold
Python Integration
Product Owner

How Low Code can be Adopted in an Organization



Main Agenda

1. Introduction

2. Example 1: Build Low Code Workflows to Compute Geo Distances

3. Example 2: Pure-Python KNIME nodes for Geo Distances

4. Wrap up



Example 1:

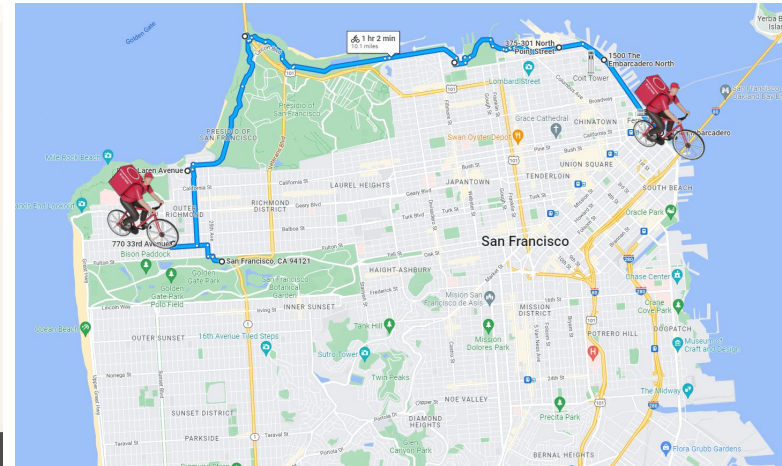
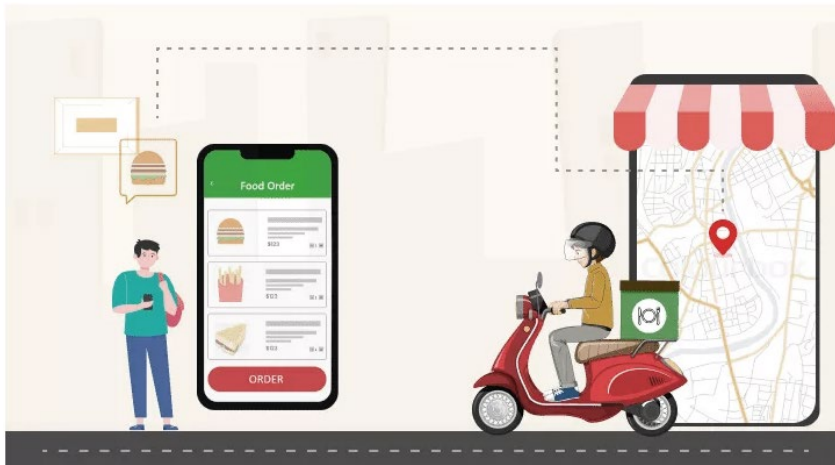
Build Low-Code Workflows to Compute Distances

- The Python Script node
- Adopting Bundled Packages
- Wrapping Low Code in Components
- Adopting Custom Packages



Today's Use Case

- A Pizza Company operating in San Francisco have several delivery agents who are responsible for delivering Pizza 🍕 across the city.



- The Company wants to compute the total trip distance of the delivery agents based on their trip coordinates

Let's Dive into the Data

- The data is stored in 2 tables, “Location” table contains the GPS coordinates of major locations in San Francisco and “Trip” table consist of the Trip details of the Delivery Agent

Row ID	S Agent ...	S Trip	S Trip Point
Row0	007	Golden Gate Park	Start
Row1	007	Fulton Playgorund	Drop
Row2	007	World War II W...	Drop
Row3	007	Alessandro DeS...	Drop
Row4	007	Golden Gate Po...	Drop
Row5	007	Presidio Tunnel ...	Drop
Row6	007	Twister Trees	Drop
Row7	007	Madam Tussads	Drop
Row8	007	Waterfront Plaza	Drop
Row9	007	Rincon Park	Finish

Trip Details

Row ID	S Location	S Latitude	S Longitude
Row0	Golden Gate Park	37.771712	-122.4824576
Row1	Fulton Playgorund	37.774591	-122.4924962
Row2	Alessandro DeS...	37.786556	-122.4895785
Row3	Golden Gate Po...	37.808685	-122.4776187
Row4	Twister Trees	37.804342	-122.4341242
Row5	Madam Tussads	37.806798	-122.4123781
Row6	Waterfront Plaza	37.804758	-122.4030593
Row7	Rincon Park	37.792771	-122.3914055
Row8	Mission Dolores...	37.781329	-122.4130082
Row9	Coit Tower	37.781329	-122.4130082

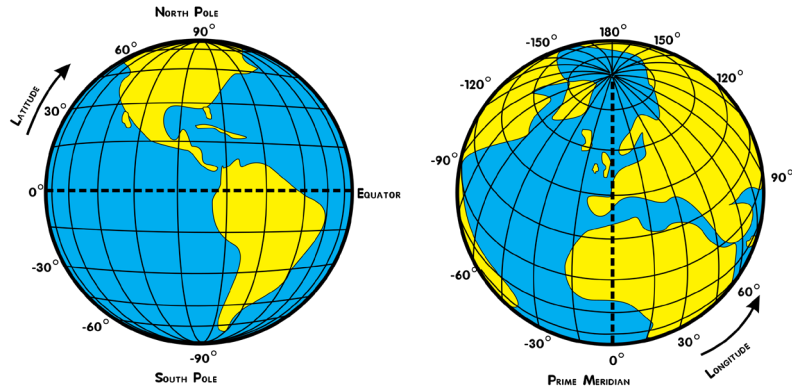
Location Coordinates

Row ID	S Agent ...	S Trip	S Trip Point	D Latitude	D Longitude
Row0	007	Golden Gate Park	Start	37.772	-122.482
Row1	007	Fulton Playgorund	Drop	37.775	-122.492
Row2	007	World War II W...	Drop	37.796	-122.481
Row3	007	Alessandro DeS...	Drop	37.787	-122.49
Row4	007	Golden Gate Po...	Drop	37.809	-122.478
Row5	007	Presidio Tunnel ...	Drop	37.801	-122.458
Row6	007	Twister Trees	Drop	37.804	-122.434
Row7	007	Madam Tussads	Drop	37.807	-122.412
Row8	007	Waterfront Plaza	Drop	37.805	-122.403
Row9	007	Rincon Park	Finish	37.793	-122.391

Trip Coordinates

Haversine Formula for Distance Computation

- The Distance (approx.) between two points can be computed using Haversine Formula



THE HAVERSINE FORMULA
FINDING THE SHORTEST DISTANCE BETWEEN 2 POINTS ON A SPHERE

GIVES THE GREAT-CIRCLE DISTANCE FROM 2 POINTS' LATITUDE + LONGITUDE

$$\text{hav}(\theta) = \sin^2\left(\frac{\theta}{2}\right)$$

HANDY FOR NAVIGATION

sketchplanations

The diagram shows a sphere with a green great circle path connecting two points. A red arrow points from the text 'GIVES THE GREAT-CIRCLE DISTANCE...' to the path. A circular inset shows a right-angled triangle with a green arc and a blue line segment.

Source : <https://sketchplanations.com/the-haversine-formula>

Geo Distance Computation with Python Script

- Use Scikit-learn package to calculate the distance using Haversine Formula

```
from sklearn.metrics.pairwise import haversine_distances
from math import radians

def geodistance(coord1, coord2):
    coord1_radians = [radians(_) for _ in coord1]
    coord2_radians = [radians(_) for _ in coord2]
    result = haversine_distances([coord1_radians, coord2_radians])
    result = result*6371000/1000
    distance = result[0][1]
    return distance
```

Add an Output Column “Trip Distance (km)”

	Agent Code	Trip	Trip Point	Location	Latitude	Longitude	Trip Distance (km)
0	007		Golden Gate Park	Start	Golden Gate Park	37.771712 -122.482458	0.000000
1	007		Fulton Playgorund	Drop	Fulton Playgorund	37.774591 -122.492496	0.938605
2	007		World War II West Coast Memorial	Drop	World War II West Coast Memorial	37.795650 -122.481000	3.488898
3	007		Alessandro DeSogos Portrait Photography	Drop	Alessandro DeSogos Portrait Photography	37.786556 -122.489578	4.750154
4	007		Golden Gate Postcard Viewpoint	Drop	Golden Gate Postcard Viewpoint	37.808685 -122.477619	7.425779
5	007		Presidio Tunnel Tops	Drop	Presidio Tunnel Tops	37.801140 -122.458000	9.342727
6	007		Twister Trees	Drop	Twister Trees	37.804342 -122.434124	11.470408
7	007		Madam Tussads	Drop	Madam Tussads	37.806798 -122.412378	13.400323
8	007		Waterfront Plaza	Drop	Waterfront Plaza	37.804758 -122.403059	14.249865
9	007		Rincon Park	Finish	Rincon Park	37.792771 -122.391406	15.930652



Python Script Node

- The node allows executing a Python script in a local Python 3 environment
- Supports conversion to both Pandas DataFrame and PyArrow Tables



The screenshot shows the "Python Script (Distance Calculation)" dialog box. The main area contains a Python script with two lines highlighted in yellow:

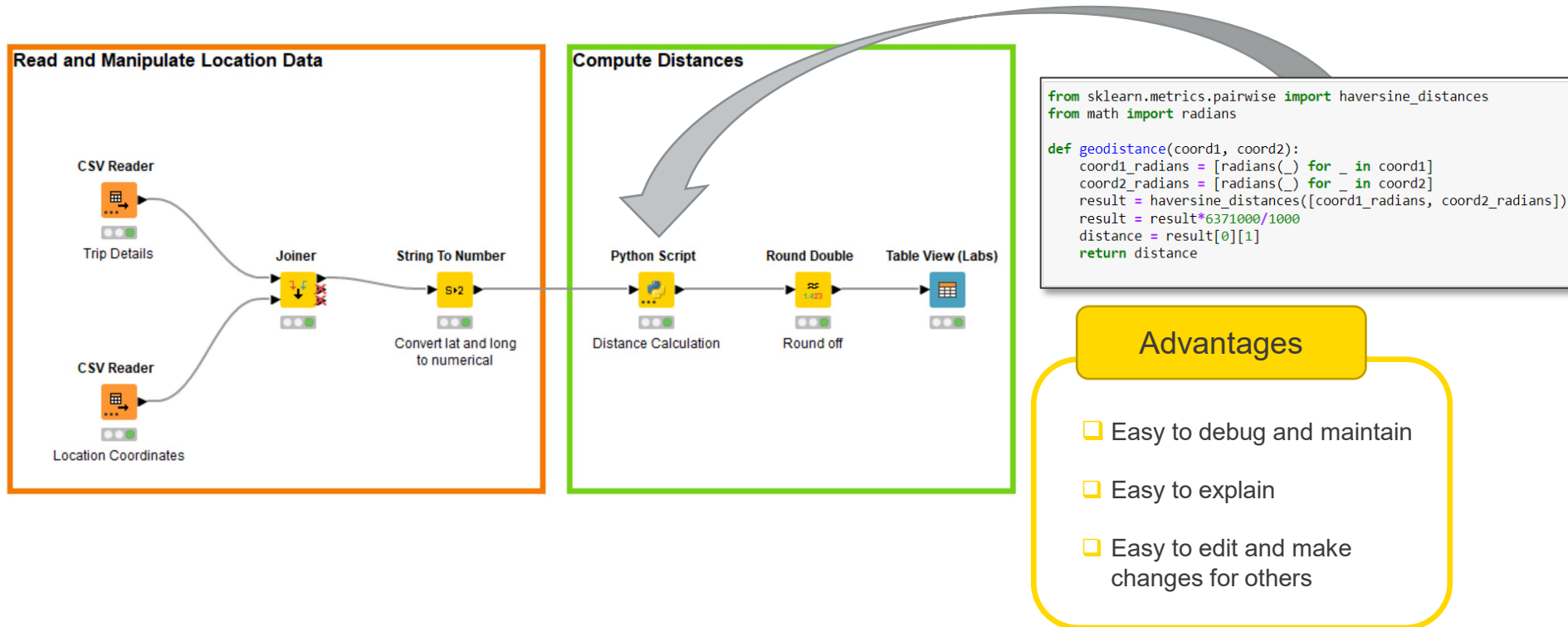
```
1 import knime_io as knio
2 #Read data as pandas dataframe
   df = knio.input_tables[0].to_pandas()
3 #Read data as Pyarrow Table
   table = knio.input_tables[0].to_pyarrow()
4
```

The dialog also shows input variables (Agent Code, Trip, Trip Point, Latitude, Longitude) and output variables (knime.output_tables[0]). At the bottom, there are buttons for "Execute script", "Execute selected lines", and "Reset workspace". A status bar at the bottom indicates "Successfully loaded input data into Python".

Blog : [Lightning Fast Data Transfer between KNIME and Python with the KNIME Python Integration](#)

Low-code Approach: Using Python code inside KNIME

- Use the Python code for distance computation inside Python Script node



Live Demo

Live Demo

Example 1:

Build Low-Code Workflows to Compute Distances

- Python Script node
- Adopting Bundled Packages
- Wrapping Low Code in Components
- Adopting Custom Packages



What are Bundled Packages?

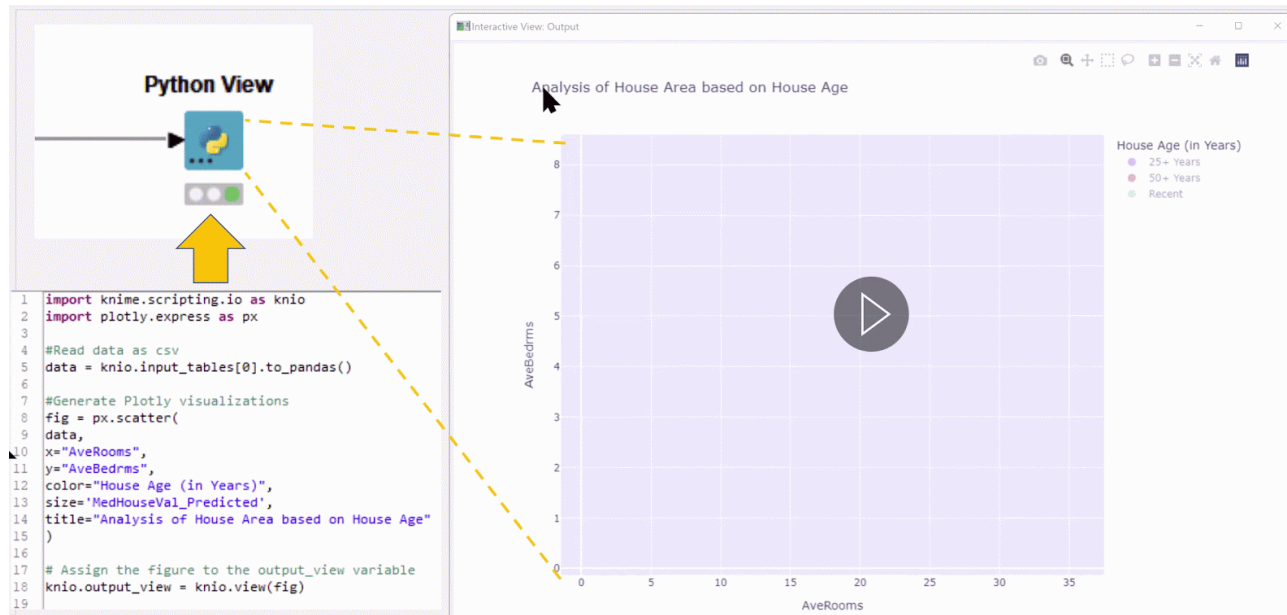
- The KNIME Python extensions installs a Python Environment that comes inbuilt with the KNIME Analytics Platform
- Provided with a selection of Python packages to get you started
- Allows for using the Python Script node without installing, configuring or even knowing environments

beautifulsoup4	Scikit-learn	pandas	numpy
scipy	Seaborn	statsmodel	cloudpickle
Matplotlib-base	ipython	nltk	plotly
requests	pyarrow	nbformat	packaging
py4j	pytz	pyaml	Jedi
openpyxl	nbformat	nomkl	pillow

[13] KNIME Python Integration – KNIME Docs

Python View Node for All Python based Visualizations

- The node allows executing a Python script that creates visualizations
- Supports static as well as interactive visualizations by plotly and share the same “interactivity” properties as native KNIME View nodes



Blog : [All Python-based Visualization Libraries Easily Accessible through KNIME](#)

Example 1:

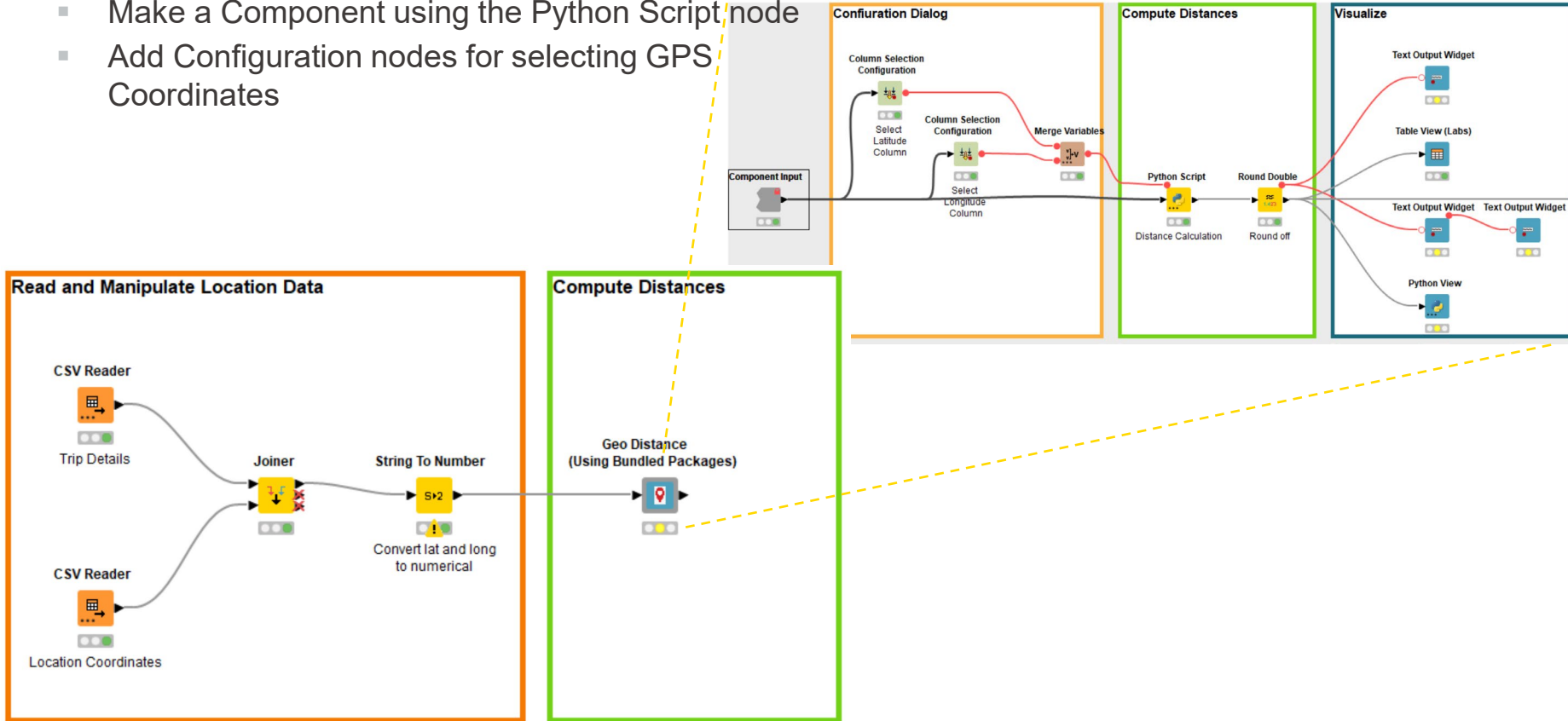
Build Low-Code Workflows to Compute Distances

- Python Script node
- Adopting Bundled Packages
- Wrapping Low Code in Components
- Adopting Custom Packages



Reliable and Reusable Solution with Components

- Build Python Scripted Components
 - Make a Component using the Python Script node
 - Add Configuration nodes for selecting GPS Coordinates



Live Demo

Live Demo

Example 1:

Build Low-Code Workflows to Compute Distances

- Python Script node
- Adopting Bundled Packages
- Wrapping Low Code in Components
- Adopting Custom Packages



Low code Solution – Using GeoPy Package

- Now you are made aware of a new package in Python called “GeoPy” that can compute the distance much accurately and provide distance in kms and miles



Link : <https://geopy.readthedocs.io/en/stable/>

- The bundled environment lacks this package, but you want to use this package for distance calculation

Let's understand how to implement this solution

What is Bundled Environment?

- The KNIME Python extensions installs a Python Environment that comes inbuilt with the KNIME Analytics Platform
- Provided with a selection of Python packages to get you started
- Allows for using the Python Script node without installing, configuring or even knowing environments

beautifulsoup4	Scikit-learn	pandas	numpy
scipy	Seaborn	statsmodel	cloudpickle
Matplotlib-base	ipython	nlTK	plotly
requests	pyarrow	nbformat	packaging
py4j	pytz	pyaml	Jedi
openpyxl	nbformat	nomkl	pillow

Not included

GeoPy??

Documentation : [Bundled Environment with KNIME](#)

Steps for Implementing Low-Code Solution

- Step 1 : Create New Python Environment via Conda and install GeoPy package

Python Environment



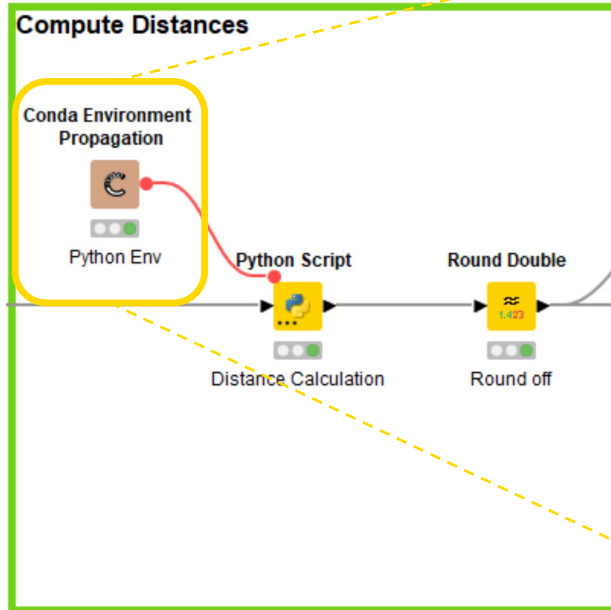
Install GeoPy to the Environment

GeoPy

Note : Please make sure Anaconda or minconda is installed on your system

Steps for Implementing Low-Code Solution

- Step 2 : Add Conda Environment Propagation node with Python Script node
 - Choose “Include only explicitly installed” option for propagating necessary packages
 - Specify the Output variable name



The screenshot shows the 'Conda environment' configuration window for 'py3_knime_geodistance'. The window displays a table of packages with columns for 'Include?', 'Name', 'Version', 'Build', and 'Channel'. The 'Include only explicitly installed' option is selected. The 'Output variable name' is set to 'py3_knime_geodistance'.

Include?	Name	Version	Build	Channel
<input checked="" type="checkbox"/>	geopy	2.2.0	pypi_0	pypi
<input type="checkbox"/>	gflags	2.2.2	ha925a31_0	pkgs/main
<input type="checkbox"/>	glib	2.69.1	h5dc1a3c_1	pkgs/main
<input type="checkbox"/>	glog	0.6.0	h4797de2_0	conda-forge
<input type="checkbox"/>	grpc-cpp	1.45.2	hb22af2e_3	conda-forge
<input type="checkbox"/>	h5py	2.10.0	py39hc0c1f7e_0	pkgs/main
<input type="checkbox"/>	hdfs	1.10.6	h1756f20_1	pkgs/main
<input type="checkbox"/>	icc_rt	2019.0.0	h0cc432a_1	pkgs/main
<input type="checkbox"/>	icu	58.2	vc14hc45fdbb_0	anaconda
<input checked="" type="checkbox"/>	python	8.4.0	py39haa95532_0	pkgs/main
<input type="checkbox"/>	jedi	0.17.2	py39haa95532_1	pkgs/main
<input type="checkbox"/>	jpeg	9b	vc14h4d7706e_1	anaconda
<input type="checkbox"/>	jpeg1	1.4.0	pypi_0	pypi
<input type="checkbox"/>	jsonschema	4.4.0	py39haa95532_0	pkgs/main
<input type="checkbox"/>	jupyter_core	4.10.0	py39haa95532_0	pkgs/main
<input type="checkbox"/>	kiwisolver	1.4.2	py39hd77b12b_0	pkgs/main
<input type="checkbox"/>	libblas	3.9.0	5_hd5c7e75_netlib	conda-forge
<input type="checkbox"/>	libbrotlicommon	1.0.9	h2bbff1b_7	pkgs/main
<input type="checkbox"/>	libbrotlidec	1.0.9	h2bbff1b_7	pkgs/main
<input type="checkbox"/>	libbrotlienc	1.0.9	h2bbff1b_7	pkgs/main
<input type="checkbox"/>	libblas	3.9.0	5_hd5c7e75_netlib	conda-forge
<input type="checkbox"/>	libcurl	7.84.0	h86230a5_0	pkgs/main
<input type="checkbox"/>	libffi	3.4.2	hd77b12b_4	pkgs/main
<input checked="" type="checkbox"/>	libiconv	1.16	h2bbff1b_2	pkgs/main

Include all Exclude all **Include only explicitly installed**

Environment validation:
 Check name only
 Check name and packages
 Always overwrite existing environment

Output variable name: py3_knime_geodistance

Steps for Implementing Low-Code Solution

- Step 2 : Add Conda Propagation node with Python Script node
 - Choose “Include only explicitly installed” option for propagating necessary packages
 - Specify the Output variable name
 - Set the Conda flow variable in “Executable Selection” tab of Python Script node

The image shows a KNIME workflow diagram on the left and a detailed view of the Python Script node configuration on the right.

Workflow Diagram (Left): A workflow titled "Compute Distances" is shown. It includes a "Conda Environment Propagation" node, a "Python Env" node, a "Python Script" node (highlighted with a yellow box), a "Distance Calculation" node, a "Round Double" node, and a "Round off" node. A red arrow points from the "Conda Environment Propagation" node to the "Python Script" node. A dashed yellow line connects the "Python Script" node in the workflow to its configuration window on the right.

Python Script Node Configuration (Right): The configuration window for the "Python Script" node is shown, with the "Executable Selection" tab selected. The main area is titled "Conda environment propagation (Python 3)". There are two radio button options:

- Use KNIME Preferences: Conda (ignore Conda flow variables)
- Use Conda flow variable

The "Use Conda flow variable" option is selected, and a dropdown menu is open, showing two options: "py3_knime_geodistance" (selected) and "py3_knime_geodistance". At the bottom of the configuration window, there is a warning icon and the text: "The 'python3_command' parameter is controlled by a variable."

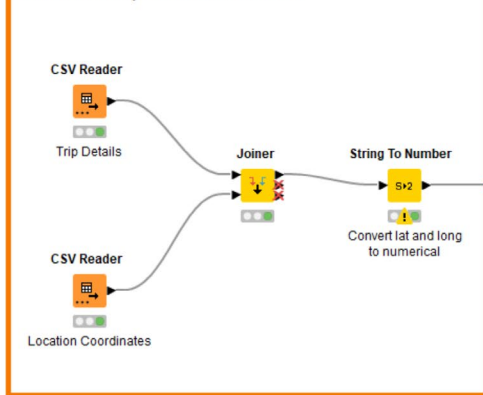
Steps for Implementing Low-Code Solution

- Step 3 : Build KNIME Workflow with Python Script node
 - Use the Reader nodes to read data into KNIME
 - Insert the code with GeoPy package inside the Python Script node

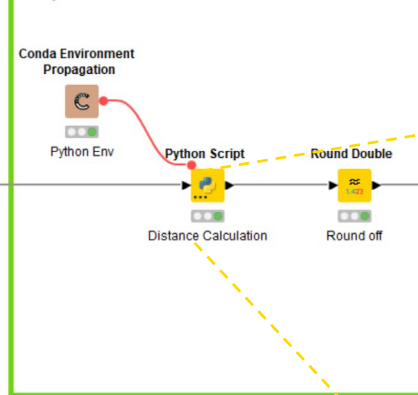
Geo Distance Calculation using Low-Code Component

This Workflow is used to compute distance in kilometer (km) between two locations given their GPS Coordinates

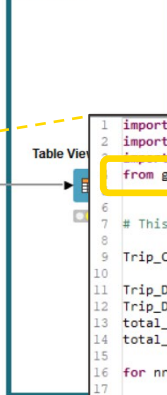
Read and Manipulate Location Data



Compute Distances



Visualize



```
from sklearn.metrics.pairwise import haversine_distances
from math import radians

def geodistance(coord1, coord2):
    coord1_radians = [radians(c) for c in coord1]
    coord2_radians = [radians(c) for c in coord2]
    result = haversine_distances([coord1_radians, coord2_radians])
    result = result[0][1] * 6371 * 1000
    distance = result[0][1]
    return distance
```

```
1 import knime_io as knio
2 import math
3
4 # This example script simply outputs the node's input table.
5
6
7 from geopy import distance
8
9 # This example script simply outputs the node's input table.
10
11 Trip_Coordinates = knio.input_tables[0].to_pandas()
12
13 Trip_Distance_km = []
14 Trip_Distance_miles = []
15 total_dist_km = 0
16 total_dist_miles = 0
17
18 for nrow in range(len(Trip_Coordinates)):
19     if nrow==0:
20         Trip_Distance_km.append(0)
21         Trip_Distance_miles.append(0)
22     else:
23         start_cordordinates = (Trip_Coordinates["Latitude"][nrow-1], Trip_Coordinates["Longitude"][nrow-1])
24         end_cordordinates = (Trip_Coordinates["Latitude"][nrow], Trip_Coordinates["Longitude"][nrow])
25
26         #use the distance function from Geopy
27         dist_km = distance.distance(start_cordordinates, end_cordordinates).km
28         dist_miles = distance.distance(start_cordordinates, end_cordordinates).miles
29
```

Where to find these workflows?

Monty's workflows available at:
tinyurl.com/Trip-KNIME-Python-World

Public space

Trip to KNIME-Python World

Last edited: 6 Sep 2022

Home



Geo Distance Geopy workflow



Geo Distance Low Code Component



Geo Distance Using GeoPy

This Space contains the workflows used in the presentation "Trip to code solutions can be implemented using KNIME and Python.

More Low Code Examples at:
tinyurl.com/Python-Script-Space

Public space

Python Script Space

Last edited: Sep 27, 2022

Home



01_Getting_Started



02_Using_Bundled_Python_Packages



03_Using_Custom_Python_Packages



04_Sharing_Python_Scripts_in_Components



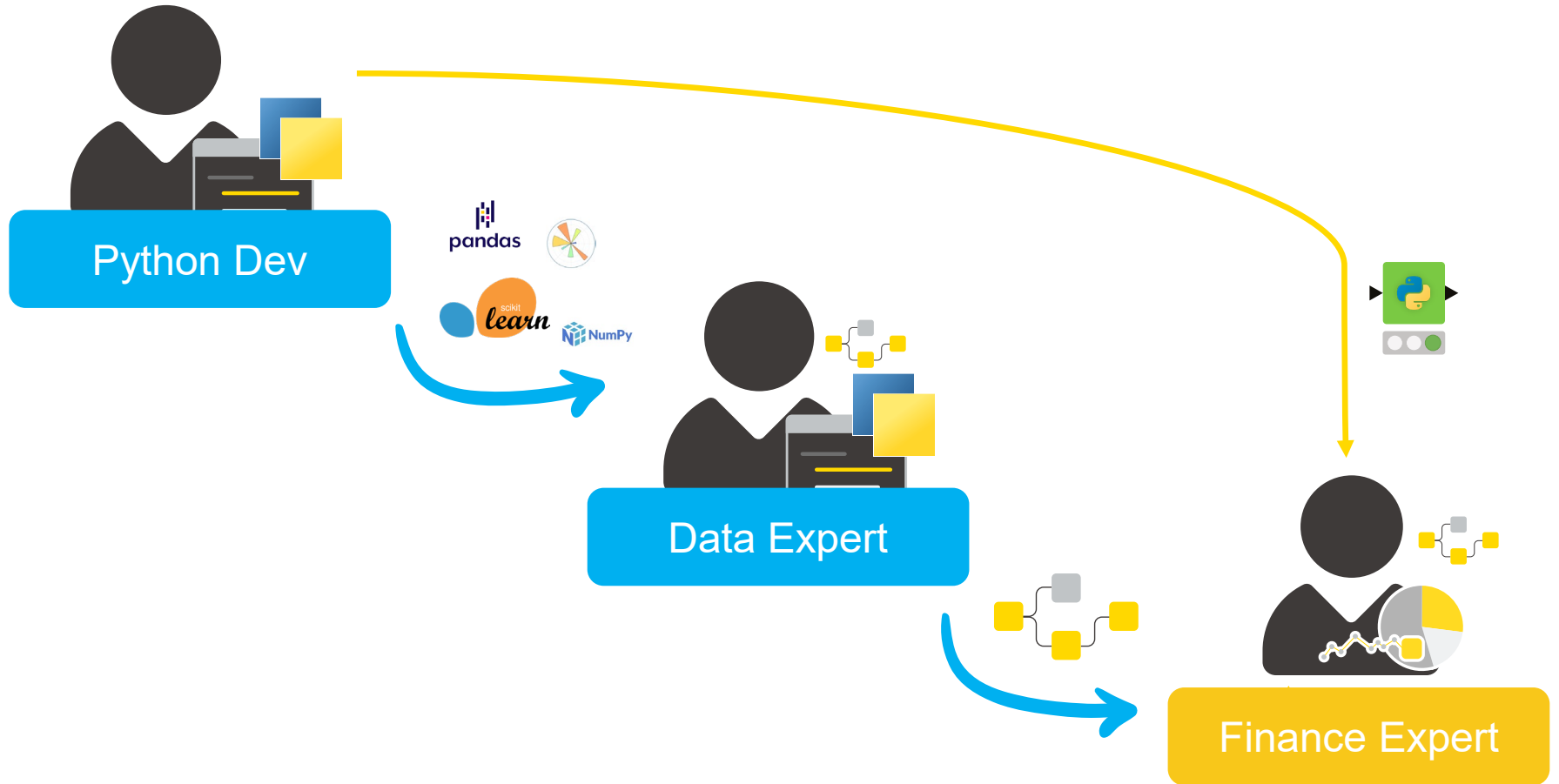
05_Jupyter_Notebook

Main Agenda

1. Introduction
2. Example 1: Build Low Code Workflows to Compute Geo Distances
3. Example 2: Pure-Python KNIME nodes for Geo Distances
4. Wrap up



From Low Code to Pure Code KNIME Solutions



Example 2:

Pure-Python KNIME nodes for
Geo Distances



A KNIME node implemented in Python

Trip Details

Row ID	from_city	to_city
Row0	Bangalore	Kolkatta
Row1	Mumbai	Chennai
Row2	Delhi	Bhopal

Location Coordinates

Row ID	city	lat	lng
Row0	Bangalore	12.972	77.595
Row1	Mumbai	19.076	72.877
Row2	Delhi	28.704	77.103
Row3	Kolkatta	22.573	88.639
Row4	Chennai	13.083	80.271
Row5	Bhopal	23.26	77.413

Geo Distances



Trip Coordinates

Row ID	from_city	to_city	Distance (Km)
Row0	Bangalore	Kolkatta	1,582.525
Row1	Mumbai	Chennai	1,033.48
Row2	Delhi	Bhopal	606.35

Developing a pure-Python KNIME node

```
@knext.node(name="Geo Distances", node_type=knext.NodeType.MANIPULATOR, icon_path="...", category="/")
@knext.input_table(name="Distances Table", description="Location pairs")
@knext.input_table(name="Coordinates Table", description="Lat/Long...")
@knext.output_table(name="Output Data", description="Distances...")
@knext.output_view("Scatter View", "Distances in a scatter plot")

class GeoDistances:
    """
    This node is able to compute distances between locations given latitude and longitude coordinates
    """
    location_column = knext.ColumnParameter("Location Column", "...", port_index=1)
    lat_column = knext.ColumnParameter("Latitude Column", "...", port_index=1)
    long_column = knext.ColumnParameter("Longitude Column", "...", port_index=1)
    output_column_name = knext.StringParameter("Title of Output Column", "...", "Distance (Km)")

    def configure(self, configure_context, input_schema_1, input_schema_2):
        """
        # Return output table schema
        return input_schema_1.append(knext.Column(knext.double(), name=self.output_column_name))

    def execute(self, exec_context, input_1, input_2):
        """
        # TODO: do something useful here
        trips_df = pd.DataFrame()

        # Return results
        return knext.Table.from_pandas(trips_df), knext.view_matplotlib()
```

Title of node, category/color, PNG name, location

Input ports with description

Output ports with description

Node class and node description

Parameters for node dialog

“configure” defines input and output table spec

“execute” defines your data analysis

Live Demo

Live Demo

Pure-Python KNIME node setup

What do you need?

1. **Python code** describing the node
2. **A conda environment** recipe containing all used Python libraries
3. **A knime.yml** file describing your extension

Sharing a Python Extension

Python Team



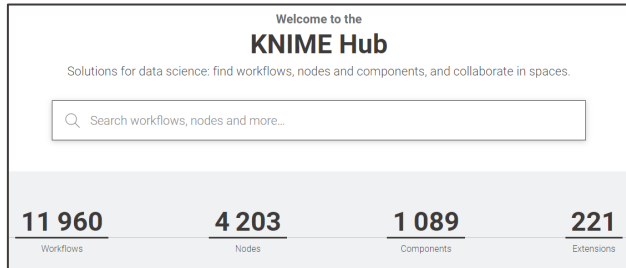
Pure-Python
Nodes



KNIME

Finance Team

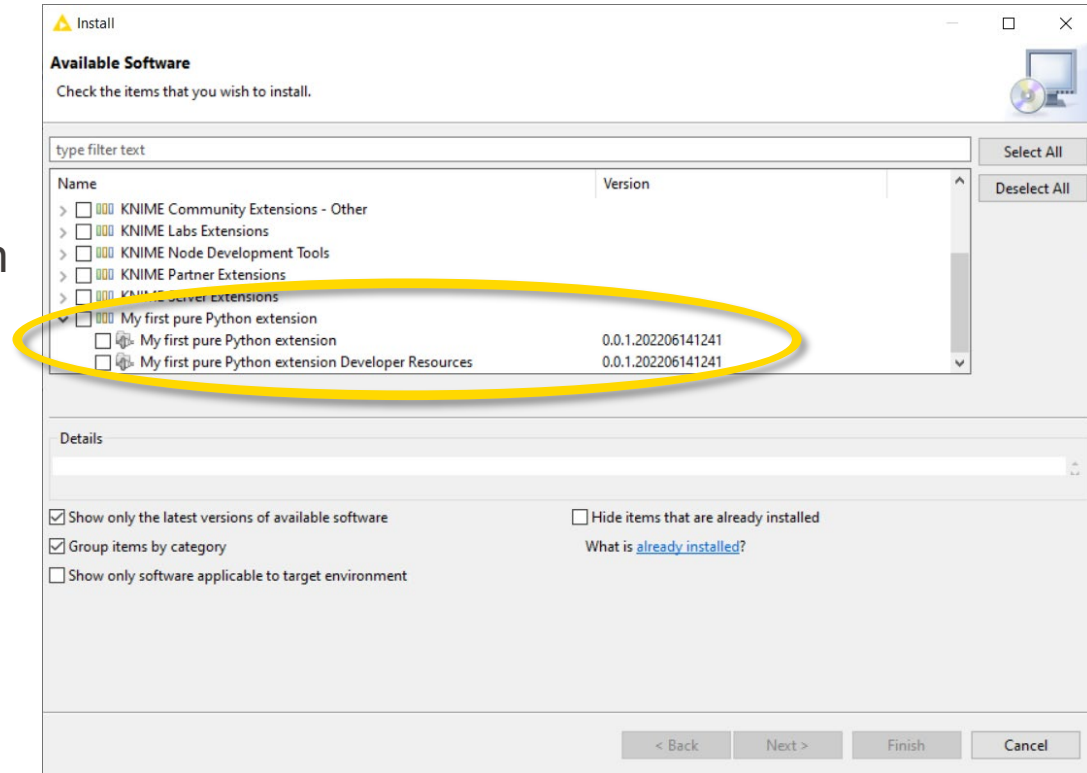
- Extensions can be shared
- Publicly on the KNIME Hub
 - Within an organization



Bundling and Sharing Within An Organization

We provide a command line tool and instructions to turn your Python code into a KNIME extension that can be installed in the KNIME Analytics Platform

1. Set up a conda environment for bundling
2. Build a local update site that can be used by KNIME
3. Share this update site with colleagues
4. Let them install your Python-based KNIME extension



Sharing to the Entire Open-Source Community

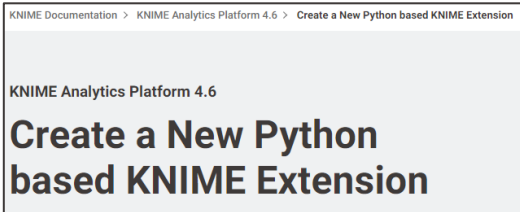
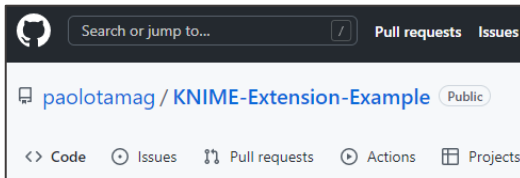
- Share your code in a publicly available git repository (e.g. BitBucket, GitLab, GitHub)
- Send a request to community-contributions@knime.org
- We will build the extension for you
- Your nodes will go live on: hub.knime.com

The screenshot shows a search results page on the KNIME Hub. At the top, it displays '1 292 results'. Below this, there are navigation tabs for 'All', 'Workflows', 'Nodes', 'Components', 'Extensions', and 'Collections'. A 'Filter by tag:' section is visible, with 'Community Nodes' selected. Other tags include 'OpenMS', 'Vernalis', 'KNIME Image Processing', 'Geospatial Analytics', 'SeqAn', 'Geometric Distance', 'Miscellaneous', 'Map Alignment', and 'Spatial Manipulation'. The main content area lists five nodes, each with a yellow icon, a title, tags, a heart icon with a count of 0, and the creator's name:

Icon	Node / Manipulator	Tags	Heart Count	Creator
Distance Map	Distance Map	Community Nodes, KNIME Image Processing, Image, +2	0	bioml-konstanz
Haversine Distance	Haversine Distance	Community Nodes, Geospatial Analytics, Spatial Manipulation	0	Center for Geograp...
Euclidean Distance	Euclidean Distance	Community Nodes, Geospatial Analytics, Spatial Manipulation	0	Center for Geograp...
Distance (1-D)	Distance (1-D)	Community Nodes, Vernalis, Miscellaneous, +2	0	vernalis
Distance (2-D)	Distance (2-D)	Community Nodes, Vernalis, Miscellaneous, +2	0	vernalis

Develop KNIME nodes using Python

- Since KNIME Analytics Platform 4.6
- Pure-Code approach: adopting an API define the node from input to output



```
@knext.node(name="Geo Distances", node_type=knext.NodeType.LEARNER, icon_path="icons/my_node_icon.png", category=
knext.input_table(name="Distances Table", description="Location pairs")
@knext.input_table(name="Coordinates Table", description="Lat/Long Coordinates of Cities")
@knext.output_table(name="Output Data", description="Distances in KM Attached to Second Input")

@knext.output_view("Scatter View", "Distances in a scatter plot")

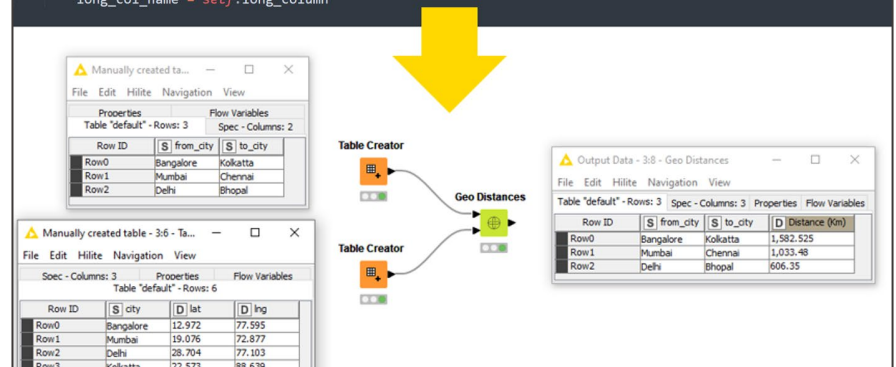
class GeoDistances:
    """
    This node is able to compute distances between locations given latitude and longitude coordinates
    """
    another_param = knext.StringParameter("Title of Distance Column", "Name for the new column reporting distance")
    location_column = knext.ColumnParameter("Location Column", "Name of locations as reported in second input")
    lat_column = knext.ColumnParameter("Latitude Column", "Latitude column in decimal format", port_index=1)
    long_column = knext.ColumnParameter("Longitude Column", "Longitude column in decimal format", port_index=1)

    def configure(self, configure_context, input_schema_1, input_schema_2):
        return input_schema_1.append(knext.Column(knext.double(), name=self.another_param))

    def execute(self, exec_context, input_1, input_2):
        exec_context.set_warning("This node was developed on Friday evening.")

        input_1_pandas = input_1.to_pandas()
        input_2_pandas = input_2.to_pandas()

        loc_col_name = self.location_column
        lat_col_name = self.lat_column
        long_col_name = self.long_column
```



Main Agenda

1. Introduction
2. Example 1: Build Low Code Workflows to Compute Geo Distances
3. Example 2: Pure-Python KNIME nodes for Geo Distances
4. Wrap up



KNIME Python Integrations: From Low-Code to Pure-Code

What?

Adopting the Python Script node with the Bundled Environment



Adding the Conda node for a Custom Environment



Implementing a new KNIME Extension in Python



Why?

Code Snippet with commonly used packages

Code Snippet with a special package

Make your Python code available as KNIME nodes

What your users will need:

Install KNIME Python Integration

Install KNIME Python Integration
+
Conda installed and configured in KNIME Preferences

Install your new KNIME extension

Thank You
Questions?



#KNIMESpringSummit
#KNIME