



COS10022 – INTRODUCTION TO DATA SCIENCE

Dr Pei-Wei Tsai (Lecturer, Unit Convenor)
ptsai@swin.edu.au, EN508d

SWIN
BUR
NE

SWINBURNE
UNIVERSITY OF
TECHNOLOGY



Lab 01



Topic:

- Basic Python Programming



Topic:

- KNIME Analytics Platform



Tools:

- Google Colab
- KNIME Software



BASIC OF PYTHON PROGRAMMING

Useful Online Resource

- CS Dojo
 - https://www.youtube.com/watch?v=Z1Yd7upQsXY&list=PLBZBJbE_rGRWeh5mIBhD-hhDwSEDxogDg
 - CS Dojo YouTube channel has a series of Python programming course for absolute beginners.
 - The key points of Python is introduced in the series in detail.





colab.research.google.com ▼

Google Colab

Colab notebooks allow you to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more. When you create your ...

You've visited this page many times. Last visit: 24/02/21

Introduction to Colab and Python Colaboratory

Welcome to this Colab where you will get a quick introduction to ...

Colaboratory, or "Colab" for short, is a product from Google ...

Google Drive

Uploading files from your local file system · Downloading files to ...

Colab Pro

Colab Pro · Get more from Colab · Faster GPUs · Longer runtimes ...

Tensorflow with GPU

In this notebook you will connect to a GPU, and then run some ...

Using Google Colab with GitHub

Colab can load public github notebooks directly, with no ...

[More results from google.com »](#)

Login to Google Colab

- Register a Google account if you do not have one.
- Download the .ipynb file from Canvas and upload the file to Google Colab.

 + Code + Text

Let's create the first comment in your program.



```
[ ] print(5)
```



```
[ ] print("I'm learning Python Programming.")
```

```
[ ] a = 1
    b = 6
    print(a)
    print(b)
    print(a,b)
```

```
[ ] a = b
    b = "It's new!"
    print(a,b)
```

Edit and Execute

- Add cells by clicking the “+ Code”.
- Click on the play icon to execute the cell.

Write Comments in Your Code

- Try the comment symbol to leave comments in your code.
 - The comment is commonly used in the programs for helping programmers to remember what is going on in the code.
 - A more import fact is that the comments help other programmers in the team to understand your code to boost up the efficiency of cooperation and team works.
- Tip: Try to write a sentence with the symbol # as the start and press “Shift” + “Enter” to execute the codes in the cell.

```
In [39]: # Let's create the first comment in your program.
```

Displaying the Result from the Code

- The basic function to use is *print()*
- It prints whatever in a given contain between the brackets.

- **Practice**

- Try to print a number 5.

```
In [7]: print(5)  
5
```

- **Practice**

- Try to print the sentence: I'm learning Python programming.

```
In [6]: print("I'm learning Python Programming.")  
I'm learning Python Programming.
```


Variable

A variable is a container, which can be used to temporary store data.

In Python, you don't need to declare the type of variable before using it.

A variable can be used to represent a single value, a vector, or a dictionary (array).

The value inside a variable can be changed anytime.

Set Variables and Print the Values

- Let's create 2 variables called *a* and *b* and assign values 1 and 6 to them, accordingly.
- Try to print *a* and *b* independently and then try to print them in the same line.

```
In [13]: a = 1
         b = 6
         print(a)
         print(b)
         print(a,b)
```

```
1
6
1 6
```

- We can reassign value to any of the variable.
 - Let's try to assign the value stored in *b* to *a* and change *b* to store a string "It's new!"
 - Print the final results of *a* and *b*.

```
In [14]: a = b
         b = "It's new!"
         print(a,b)
```

```
6 It's new!
```

Brainstorm Time!

Assume we have two variables, a and b , holding two strings “Alpha” and “Beta,” respectively.

What are you going to do if we want to make a swap to let variable a hold “Beta” and variable b hold “Alpha?”



Brainstorm Time!

```
In [22]: # Initial State
a = "Alpha"
b = 'Beta'
print("Before:")
print("a =", a, "\nb =", b)

# This is how you're going to do it.
temp = a
a = b
b = temp
print("\nAfter:")
print("a =", a, "\nb =", b)
```

Before:
a = Alpha
b = Beta

After:
a = Beta
b = Alpha



Mathematic Operators

- **Relationships:**
 - **>** Greater
 - **<** Smaller
 - **==** Equal to
 - **>=** Greater and equal to
 - **<=** Smaller and equal to
 - **!=** Not equal to

Flow Control: If-Else

In many situations, you'll need to control your program to do one thing if a certain criterion appears, otherwise, your program needs to achieve another operation when that criterion doesn't exist.

If-Else is a flow control description to help you change to execution structure of your code.

You need to “indent” the description under the criterion description to let Python know that this part of the code belongs to the criterion corresponding situation.

If-Else

- Expression:
 - *if criterion₁ relation criterion₂:*
descriptions₁
else:
descriptions₂
- Example:

```
In [24]: a = 2
         b = 3
         if a < b:
             print("a is less than b.")
         else:
             print("a is greater than or equal to b.")

a is less than b.
```

```
In [25]: a = 5
         b = 3
         if a < b:
             print("a is less than b.")
         else:
             print("a is greater than or equal to b.")

a is greater than or equal to b.
```

Brainstorm Time!

Assume you have 10 dollars on your hand.

A set of hamburger causes 12 dollars.

Write a program using If-Else to check whether you can afford to have a hamburger set.



Brainstorm Time!

```
In [27]: # Brainstorm 2

# initial values
x = 10
burger = 12

# main program
if x >= burger:
    print('You can afford to have a burger meal.')
else:
    print("The burger set is too expensive. You can't afford it.")

The burger set is too expensive. You can't afford it.
```



Brainstorm Time!

Assume you have 10 dollars on your hand.

A set of hamburger causes 12 dollars while a set of hotdog causes 8 dollars.

Write a program using If-Else to check what meal you can get.



Brainstorm Time!

```
In [32]: # Brainstorm 3

# initial values
x = 10
burger = 12
hotdog = 8

# main program
if x >= burger:
    if x >= hotdog:
        print("You can afford choose either a burger or a hotdog set as the meal.")
    else:
        print("You can only afford to have a burger set.")
else:
    if x >= hotdog:
        print("You can only afford to have a hotdog set.")
    else:
        print("You cannot afford either a burger or a hotdog set as the meal.")
```

You can only afford to have a hotdog set.



If-Elif-Else

- Expression:

- *if* *criteria*₁ *relation* *criteria*₂:
 descriptions₁
 - elif* *criteria*₃ *relation* *criteria*₄:
 descriptions₂
 - else*:
 descriptions₃

- Example:

```
In [38]: # The three cases example

# Set input variables
a = 2
b = 3

# Main program
if a < b:
    print("a is less than b.")
elif a == b:
    print("a is equal to b.")
else:
    print("a is greater than b.")

a is less than b.
```

Functions

Function can be treated as a collection of instructions.

Function can also be treated as a collection of codes.

The purpose of using function is to put the repeatable part of codes in an independent place and reuse in the future without rewrite them again.

Using functions can also help you to make your code more tidy.

Function

- `def function_name(input_vars):`
descriptions_in_the_function
- To use the function, simply call the function in the code.
- Example:

In [46]: # Functions

```
def my_first_function():  
    print("This line is inside the function")  
    print("This line is also inside the function")  
  
print("This line is outside of the function")
```

This line is outside of the function

In [48]: # Functions

```
def my_first_function():  
    print("This line is inside the function")  
    print("This line is also inside the function")  
  
print("This line is outside of the function")  
my_first_function()
```

This line is outside of the function
This line is inside the function
This line is also inside the function

Function

- Passing a variable into the function and use it.
- Example:

```
In [49]: # Passing variable(s) into functions

def function2(var1):
    print("The variable sent to the function is", var1)

print("This line is outside of the function")
myvar = 5
function2(myvar)
```

```
This line is outside of the function
The variable sent to the function is 5
```

Return Value from a Function

- *def function_name(input_vars):*
descriptions_in_the_function
return a_variable
- To use the function, simply call the function in the code and assign the function to a variable.

```
In [52]: # Return value from a function

def function3(var1):
    print("The variable sent to the function is", var1)
    return 3 * var1 + 2

print("This line is outside of the function")
myvar = 5
result = function3(myvar)
print(result)
```

```
This line is outside of the function
The variable sent to the function is 5
17
```

Passing and Returning Multiple Values from a Function

```
In [56]: # Passing and returning multiple values from a function
```

```
def function3(var1, var2):  
    return (3 * var1 + 2), (-1 * var2)  
  
print("This line is outside of the function")  
myvar = 5  
[result, res2] = function3(myvar, (myvar - 2))  
print(result, res2)
```

```
This line is outside of the function  
17 -3
```



Exercise 1 – Choosing Available Rooms

- Assume we have three class rooms called RoomA, Room B, and RoomC with capacities of 20, 50, and 120, respectively.
- Only 1 room can be used to contain all students.
- The student enrolment number is 48.
- Use *If-Else* and/or *If-Elif-Else* structure to recommend the available classroom.
- Use a function to handle the output of available recommendations.

List

List is one of the data structure in Python that stores a series of data in one variable.

The list can contain items in the same or different data types.

Some built-in functions can help you to manage the items on the list.

List

- Try the following command to create a list:

```
In [31]: # Create a List called "a"  
a = [3,1,4,-2]  
print(a)  
  
[3, 1, 4, -2]
```

- To insert an item at the end of a list:

```
In [52]: # Insert an item at the end of a list  
a.append('new item')  
print(a)  
  
[3, 1, 4, -2, 'new item']
```

- To insert an item at a specific location:

```
In [42]: a.insert(2,5)  
print(a)  
  
[3, 1, 5, 4, -2, 'new element']
```

- A list inside another list:

```
In [43]: # Create a list inside a list  
b = [1,2,3]  
b.append([5,6])  
print(b)  
  
[1, 2, 3, [5, 6]]
```

List

- Pop (delete) an element from the list:

```
In [54]: # Pop (delete) an item from the list
c = [3,7,2,8,5]
print(c)

c.pop() # pop the last item and remove it from the list
print(c)

c.pop(1) # pop the item situated on the specific point
print(c)

[3, 7, 2, 8, 5]
[3, 7, 2, 8]
[3, 2, 8]
```

List

- Retrieve a specific item from the list:

```
In [60]: # Retrieve the specific item on the list  
print(b)  
print(b[1])  
print(b[3])  
print(b[3][1])
```

```
[1, 2, 3, [5, 6]]
```

```
2
```

```
[5, 6]
```

```
6
```

Brainstorm Time!

Assume we have a list, which holds two variables called “Alpha” and “Beta.”

What are you going to do if we want to make a swap?



Brainstorm Time!

```
In [61]: # Swap items in a list  
myList = ["Alpha", "Beta"]  
print("Before:", myList)  
myList[0], myList[1] = myList[1], myList[0]  
print("After:", myList)
```

Before: ['Alpha', 'Beta']
After: ['Beta', 'Alpha']



The range() Function

The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

range(x): Defines a range of value starting from 0 and ending at (x-1) with an interval of 1.

range(x, y): Defines a range of value starting from x and ending at (y-1) with an interval of 1.

range(x, y, z): Defines a range of value starting from x and ending at (z-1) with an interval of y.

The list() Function

The list() function creates a list for you.

It converts range(x) and range(x,y) outputs into list.

```
In [78]: # Example of using list() and range()
b = [1,2,3,4,5]
print(b)
c = list(range(1,6))
print(c)
```

```
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]
```

For-Loop Control

For-loop is quite handy when you have a list of items to visit.

It helps to make the same operation on different items much more efficient.

For-Loop Example

```
In [87]: # for-loop example
a = [1,2,3]
print("This is printed without a for-loop:")
print(a[0])
print(a[1])
print(a[2])

print("\nThis is printed with a for-loop:")
for x in [0,1,2]:
    print(a[x])
```

This is printed without a for-loop:

1
2
3

This is printed with a for-loop:

1
2
3

For-Loop Example (2)

```
In [98]: # for-loop example
a = [1,2,3]

print("This is printed with the elements only (x represents the item/element inside the list, directly):")
for x in a:
    print(x)

print("The result is equal to print with a for-loop and a range function (mind the count of the index):")
for x in a:
    print(a[(x-1)])
```

This is printed with the elements only (x represents the item/element inside the list, directly):

1
2
3

The result is equal to print with a for-loop and a range function (mind the count of the index):

1
2
3

For-Loop Example (3)

```
In [92]: # for-loop example
a = [1,2,3]

print("This is printed with a for-loop and a range function:")
print("range(3) is equal to range(0,3) and range(0,1,3)")
for x in range(3):
    print(a[x])

print("\nThis is printed with a for-loop and a range function with range() and len()")
for x in range(len(a)):
    print(a[x])
```

This is printed with a for-loop and a range function:

range(3) is equal to range(0,3) and range(0,1,3)

1
2
3

This is printed with a for-loop and a range function with range() and len()

1
2
3

For-Loop Example (4)

```
In [105]: # for-loop example: add sum
a = list(range(1,5))
print(a)

total = 0
for i in a:
    total += i # This is equal to total = total + i
print(total)
```

```
[1, 2, 3, 4]
```

```
10
```

Modulo Operator (%)

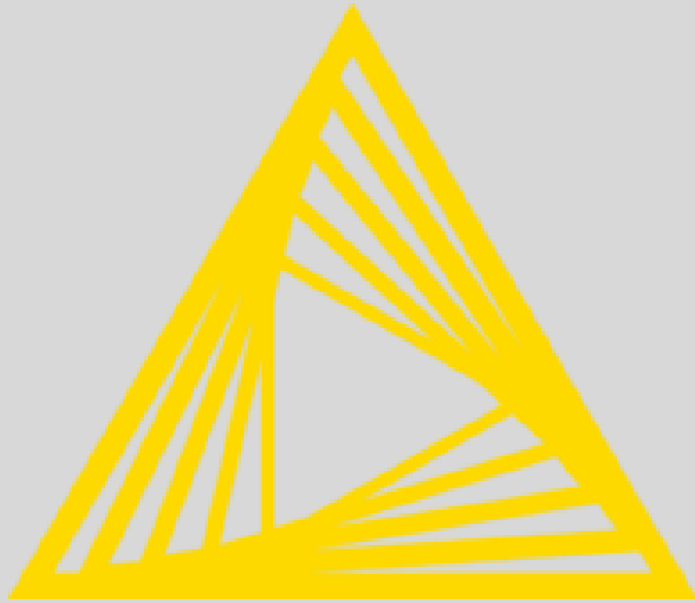
A modulo operator (noted by the symbol %) gives the remainder of the calculation result.

For example: $7\%3$ will return 1 because the remainder of this calculation is 1.

1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50

Exercise 1 – Find the sum of only the multiples of a specific number.

- Assume we have a sequence $[1, 2, \dots, 50]$.
- Try to use for-loop, if, and the modulo operator to find the sum of only the multiples of 3.



Open for Innovation

KNIME

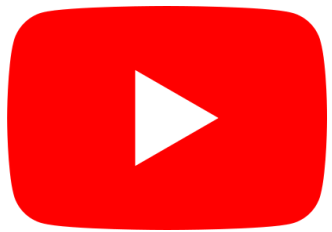
KNIME ANALYTICS PLATFORM

- Go to the online document ([KNIME Quick Start](#)) provided in Canvas to continue on the second part of the tutorial.



Open for Innovation

KNIME



YouTube

Useful Online Resource

CS Dojo

https://www.youtube.com/watch?v=Z1Yd7upQsXY&list=PLBZBJbE_rGRWeh5mlBhD-hhDwSEDxogDg

CS Dojo YouTube channel has a series of Python programming course for absolute beginners.

The key points of Python is introduced in the series in detail.

KNIME Official Website Documents

<https://docs.knime.com/>

Basics of KNIME

My First Workflow

By Rosaria Silipo

Adapted from:

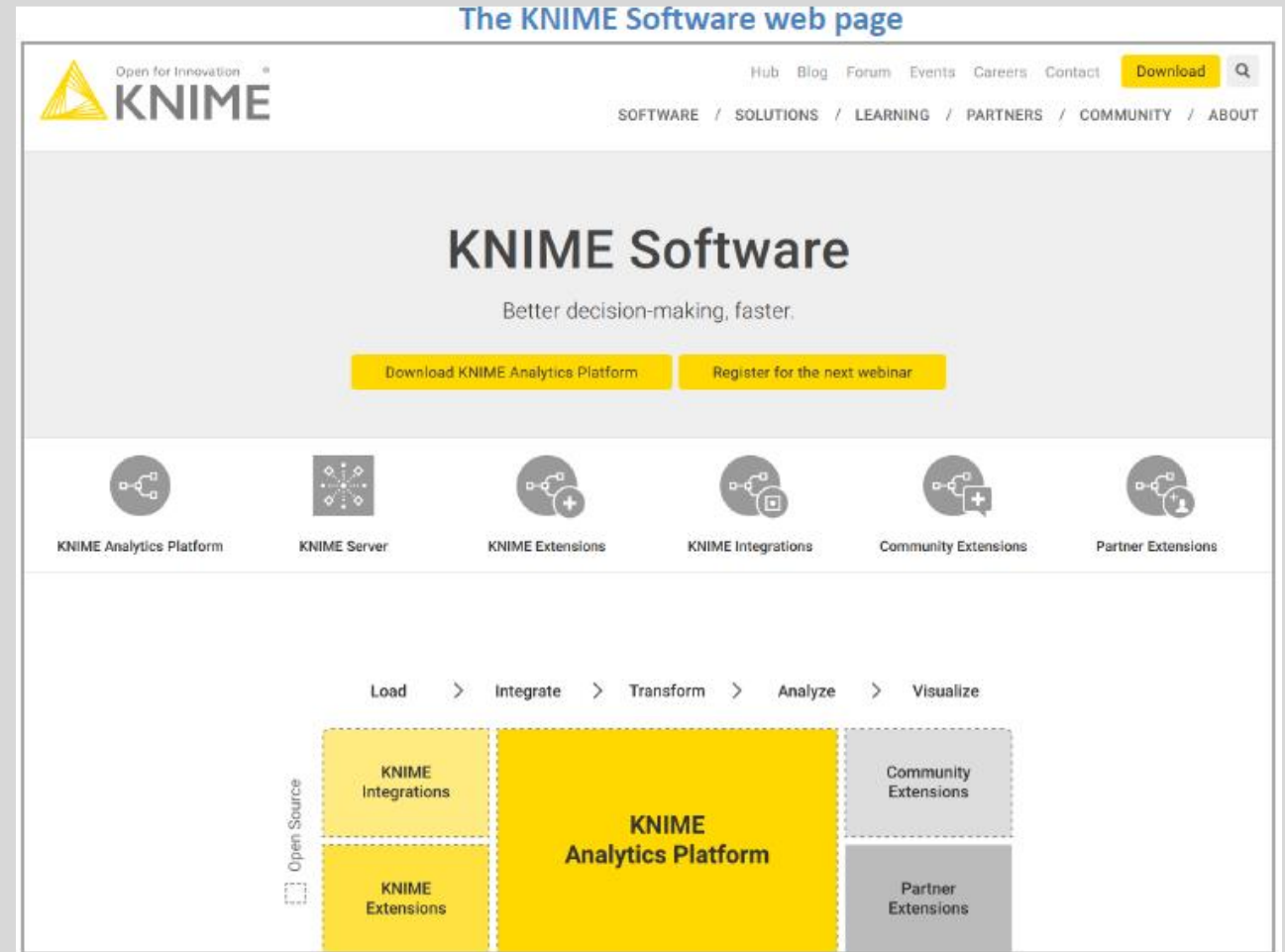
KNIME® Beginner's Luck: A Guide to KNIME analytics Platform for Beginners

Download and install KNIME Analytics Platform

To start playing with KNIME Analytics Platform, first, you need to download it to your machine.

Download KNIME Analytics Platform

- Go to www.knime.org
- In the lower part of the first screen of the main page, click "KNIME Software"
- In the "KNIME Software" page, click the button "Download KNIME Analytics Platform".
- Provide a little information about yourself (that is appreciated), then proceed to step 2 "Download KNIME"
- Choose the version that suits your environment (Windows/Mac/Linux, 32 bit/64 bit, with or without Installer for Windows) optionally including all free extensions
- Accept the terms and conditions
- Start downloading
- You will end up with a zipped (*.zip), a self-extracting archive file (*.exe), or an Installer application
- For .zip and .exe files, just unpack it in the destination folder
- If you selected the installer version, just run it and follow the installer instructions



Workspace

To start KNIME Analytics Platform, open the folder where KNIME has been installed and run knime.exe (or knime on a Linux/Mac machine). If you have installed KNIME using the Installer, then you can just click the icon on your desktop or on your Windows main menu.

After the splash screen, the "Workspace Launcher" window requires you to enter the path of the workspace.

The "Workspace Launcher"

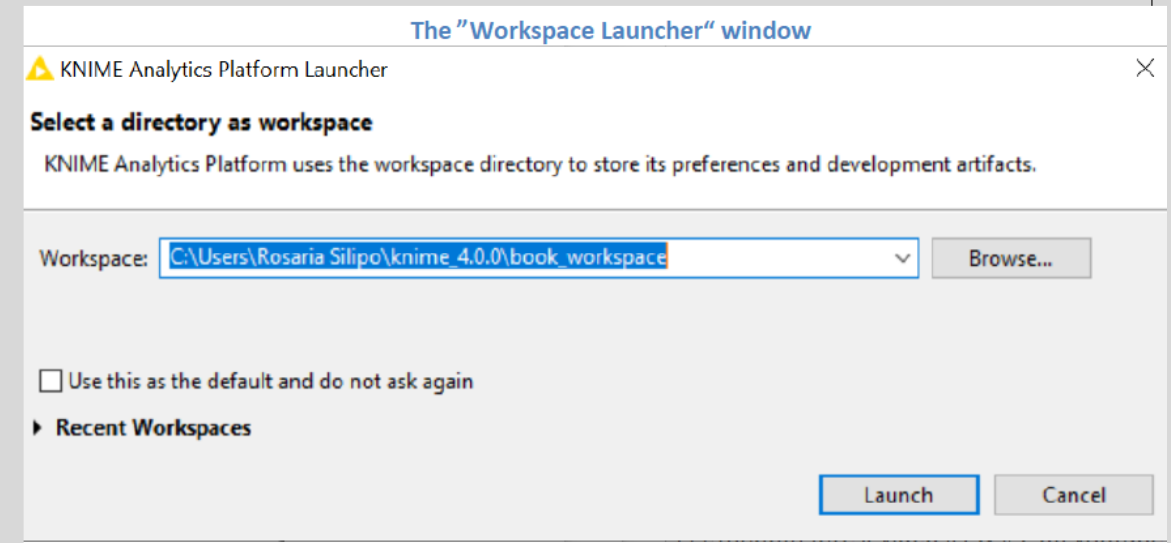
The **workspace** is a folder where all preferences and applications (workflows), both developed and currently under development, are saved for the next KNIME session.

The workspace folder can be located anywhere on the hard-disk.

By default, the workspace folder is "..\knime-workspace". However, you can easily change that, by changing the path proposed in the "Workspace Launcher" window, before starting the KNIME working session.

Once KNIME Analytics Platform has been opened, from within the KNIME workbench you can switch to another workspace folder, by selecting "File" in the top menu and then "Switch Workspace". After selecting the new workspace, KNIME Analytics Platform restarts, showing the workflow list from the newly selected workspace. Notice that if the workspace folder does not exist, it will be automatically created.

If I have a large number of customers for example, I can use a different workspace for each one of them. This keeps my work space clean and tidy and protects me from mixing up information by mistake. For this project I used the workspace "KNIME_4.x.y\book_workspace".



KNIME Workflow

KNIME Analytics Platform does not work with scripts, it works with graphical workflows.

Small little boxes, called nodes, are dedicated each to implement and execute a given task. A sequence of nodes makes a workflow to process the data to reach the desired result.

What is a workflow?

A workflow is an **analysis flow**, i.e. the **sequence of analysis steps** necessary to reach a given result. It is the pipeline of the analysis process, something like:

KNIME Analytics Platform implements its workflows **graphically**. Each step of the data analysis is implemented and executed through a little box, called **node**. A sequence of nodes makes a workflow.

Step 1. Read data

Step 2. Clean data

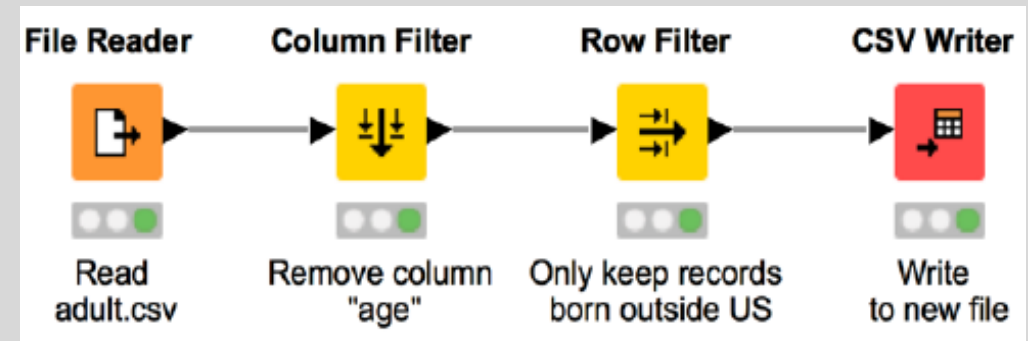
Step 3. Filter data

Step 4. Train a model

Below is an example of a KNIME workflow, with:

- a node to read data from a file
- a node to exclude some data columns
- a node to filter out some data rows
- a node to write the processed data into a file

Example of a KNIME workflow



Note. A workflow is a data analysis sequence, which in a traditional programming language would be implemented by a series of instructions and calls to functions. KNIME Analytics Platform implements it graphically. This graphical representation is more intuitive to use, lets you keep an overview of the analysis process, and makes for the documentation as well.

KNIME Workflow

What is a node?

A node is the **single processing unit** of a workflow.

A node takes a data set as input, processes it, and makes it available at its output port. The "processing" action of a node ranges from modeling - like an Artificial Neural Network Learner node - to data manipulation - like transposing the input data matrix- from graphical tools - like a scatter plot, to reading/writing operations.

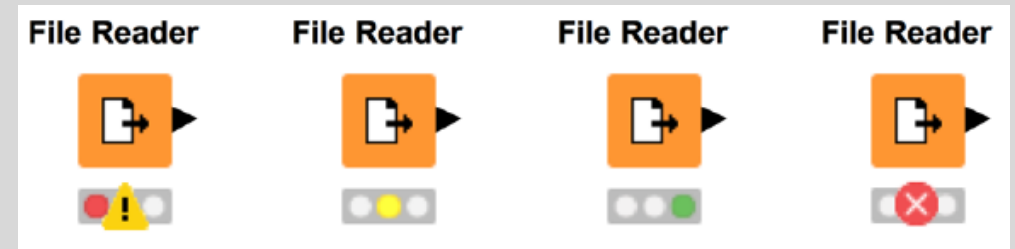
Every node in **KNIME** has 4 states:

Inactive and not yet configured
Configured but not yet executed
Executed successfully
Executed with errors

- ✓ **red** light
- ✓ **yellow** light
- ✓ **green** light
- ✓ **red with cross** light

Below are four examples of the same node (a File Reader node) in each one of the four states.

File Reader node with



Nodes containing other nodes are called **metanodes** or **components**.

.knwf and .knar file extensions

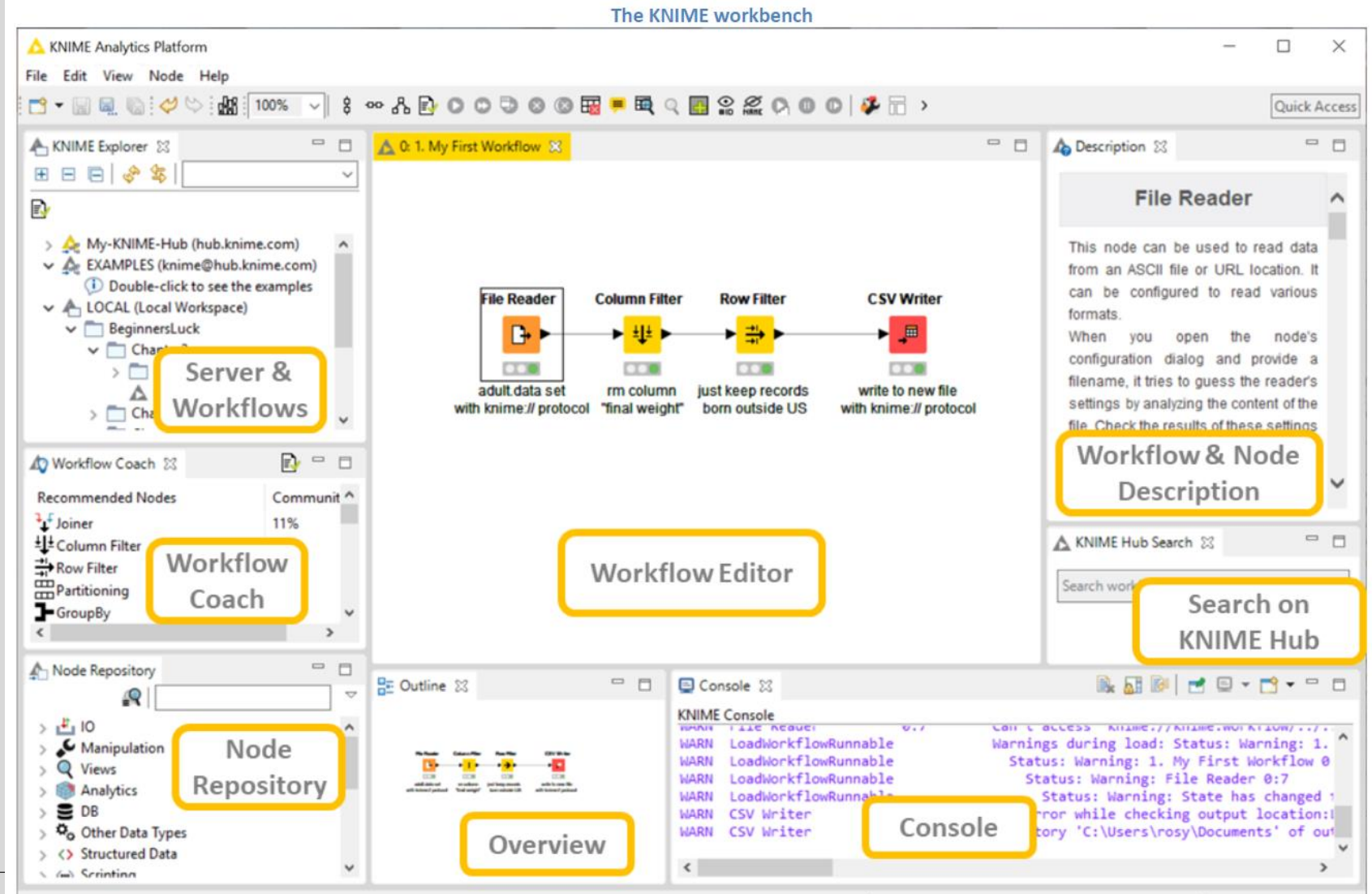
KNIME workflows can be packaged and exported in *.knwf* or *.knar* files. A *.knwf* file contains only one workflow, while a *.knar* file contains a group of workflows. Such extensions are associated with KNIME Analytics Platform. A double-click opens the workflow inside KNIME Analytics Platform.

.knwf and *.knar* files are associated with KNIME Analytics Platform. A double-click opens the workflow(s) directly inside the platform.

01_From_String,_to_Documents.knwf	10/4/2017 9:45 AM	KNIMEWorkflow ...	18,619KB
04_Interaction_Graph.knwf	9/29/2017 8:20 AM	KNIME Warkflow ...	9,465 KB
06_RE5T_Examples_Google_Geocode.knwf	7/29/2017 7:09 PM	KNIME Workflow...	62KB
06_Semantic_Web_updated.knar	11/3/2016 2:24 PM	KNIME Archive File	178KB
AzureDemoWorkflowArchive.knar	5/5/2017 11:24 AM	KNIME Archive File	24,104 KB
Building a Simple Classifier.knwf	2/18/2017 5:46 PM	KNIME Workflow ...	43 KB
Cookbook_Ch5.knar	11/24/2017 10:03 ...	KNIME Archive File	477KB
Cookbook_Ch6.knar	11/24/2017 10:26 ...	KNIME Archive File	1.1 KB
Corsai.knwf	7/10/2017 7:20 PM	KNIME Workflow...	106KB

KNIME Workbench




























After accepting the workspace path, the KNIME workbench opens on a "Welcome to KNIME" page. This page provides a few links to get started, such as for example to the KNIME Hub, to some basic documentation, to the current courses and events, to available updates, and so on. The "KNIME Workbench" consists of a top menu, a tool bar, and a few panels. Panels can be closed, re-opened, and moved around.






























KNIME Workbench

The KNIME Workbench		
Top Menu: File, Edit, View, Node, Help		
Tool Bar: New, Save (Save As, Save All), Undo/Redo, Open Report (if reporting was installed), zoom (in %), Align selected nodes vertically/horizontally, Auto layout, Configure, Execute options, Cancel execution options, Reset, Edit node name and description, Open node's first out port table, Open node's first view, Open the "Add Meta node" Wizard, , Append IDs to node names, Hide all node names, Loop execution options, Change Workflow Editor Settings, Edit Layout in Components, configure job manager.		
KNIME Explorer This panel shows the list of workflow projects available in the selected workspace (LOCAL), on the EXAMPLES server, on the My-KNIME-Hub (your own space on the KNIME Hub), or on other connected KNIME servers.	Workflow Editor The central area consists of the "Workflow Editor" itself. A node can be selected from the "Node Repository" panel and dragged and dropped here, in the "Workflow Editor" panel. Nodes can be connected by clicking the output port of one node and releasing the mouse either at the input port of the next node or at the next node itself.	Node Description If a node or a workflow is selected, this panel displays a summary description of the node's functionalities or the workflow's meta information.
Workflow Coach This is a node recommendation engine. It will provide the list of the top most likely nodes to follow the currently selected node.		Search box for KNIME Hub To search for material on the KNIME Hub
Node Repository This panel contains all the nodes that are available in your KNIME installation. It is something similar to a palette of tools when working in a report or with a web designer software. There we use graphical tools, while in KNIME we use data analytics tools.	Outline The "Outline" panel contains a small overview of the contents of the "Workflow Editor". The "Outline" panel might not be of so much interest for small workflows. However, as soon as the workflows reach a considerable size, all the workflow's nodes may no longer be visible in the "Workflow Editor" without scrolling. The "Outline" panel, for example, can help you locate newly created nodes.	Console The "Console" panel displays error and warning messages to the user. This panel also shows the location of the log file, which might be of interest when the console does not show all messages. There is a button in the tool bar as well to show the log file associated with this KNIME instance.

KNIME Workbench

Top menu		
File	Edit	View
 New... Ctrl+N  Save Ctrl+S  Save As...  Save All Ctrl+Shift+S  Close All Ctrl+Shift+W  Print... Ctrl+P  Import KNIME Workflow...  Export KNIME Workflow...  Export to SVG... Switch Workspace > Preferences  Export Preferences...  Import Preferences... Install KNIME Extensions... Update KNIME... Restart Exit	 Undo Ctrl+Z  Redo Ctrl+Y  Cut Ctrl+X  Copy Ctrl+C  Paste Ctrl+V  Delete Delete Select All Ctrl+A	 Console Alt+Shift+Q, C  Description  Error Log Alt+Shift+Q, L  KNIME Explorer  KNIME Hub Search  Node Repository  Outline Alt+Shift+Q, O  Workflow Coach Other... Alt+Shift+Q, Q Reset Perspective...  Quick Node Insertion... Ctrl+Space  Open KNIME log
<p>File includes the traditional File commands, like “New” and “Save”, in addition to some KNIME specific commands, like:</p> <ul style="list-style-type: none"> - Import/Export KNIME workflow... - Export to SVG - Switch Workspace - Preferences with Export/Import Preferences - Install KNIME Extensions - Update KNIME 	<p>Edit contains edit commands.</p> <p>Undo and Redo refer to the last performed actions.</p> <p>Cut, Copy, Paste, and Delete refer to selected nodes in the workflow.</p> <p>Select All selects all the nodes of the workflow in the workflow editor.</p>	<p>View contains the list of all panels that can be opened in the KNIME workbench.</p> <p>A closed panel can be re-opened here.</p> <p>Also, when the panel disposition is messed up, the option “Reset Perspective” re-creates the original panel layout when the workbench was started for the first time.</p> <p>Option “Other” opens additional views useful to customize the workbench.</p>

KNIME Workbench

Node	Help
 Configure... F6  Execute F7  Execute All Shift+F7  Execute and Open Views Shift+F10  Cancel F9  Cancel All Shift+F9  Reset F8  Edit Node Name and Description... Alt+F2  Open First Out-Port View Shift+F6  Open First View F10  Update Metanode Links Ctrl+Alt+U  Open Metanode Wizard...  Show Node IDs Ctrl+Alt+W  Hide Node Names Ctrl+Alt+Q  Link selected nodes Ctrl+L  Unlink selected nodes Ctrl+Shift+L  Step Loop Execution Ctrl+Alt+F6  Pause Execution Ctrl+Alt+F7  Resume Loop Execution Ctrl+Alt+F8  Select Loop Ctrl+Alt+F10  Workflow Editor Settings...  Component Usage And Layout  Configure JobMgr...	 Help Contents  Search Welcome Page  About KNIME Analytics Platform  Install New Software... Show Active Keybindings... Cheat Sheets...
<p>Node refers to all possible operations that can be performed on a node. A node can be:</p> <ul style="list-style-type: none">- Configured- Executed- Cancelled (stopped during execution)- Reset (resets the results of the last “Execute” operation)- Given a name and description- Set to show its View (if any) <p>Options are only active if they are possible. For example, an already successfully executed node cannot be re-executed unless it is first reset or its configuration has been changed. The “Cancel” and “Execute” options are then inactive.</p> <p>Option “Open Meta Node Wizard” starts the wizard to create a new meta node in the workflow editor.</p>	<p>Help Contents provides general Help about the Workbench, BIRT, and KNIME.</p> <p>Search opens a panel on the right of the “Node Description” panel to search for specific Help topics or nodes.</p> <p>Welcome Page (re-)opens the Welcome Page</p> <p>Install New Software is the door to install KNIME Extensions from the KNIME Update sites.</p> <p>Show Active Keybindings summarizes all keyboard commands for the workflow editor.</p> <p>Cheat Sheets offer tutorials on specific topics: the reporting tool, cvs, Plug-ins.</p>