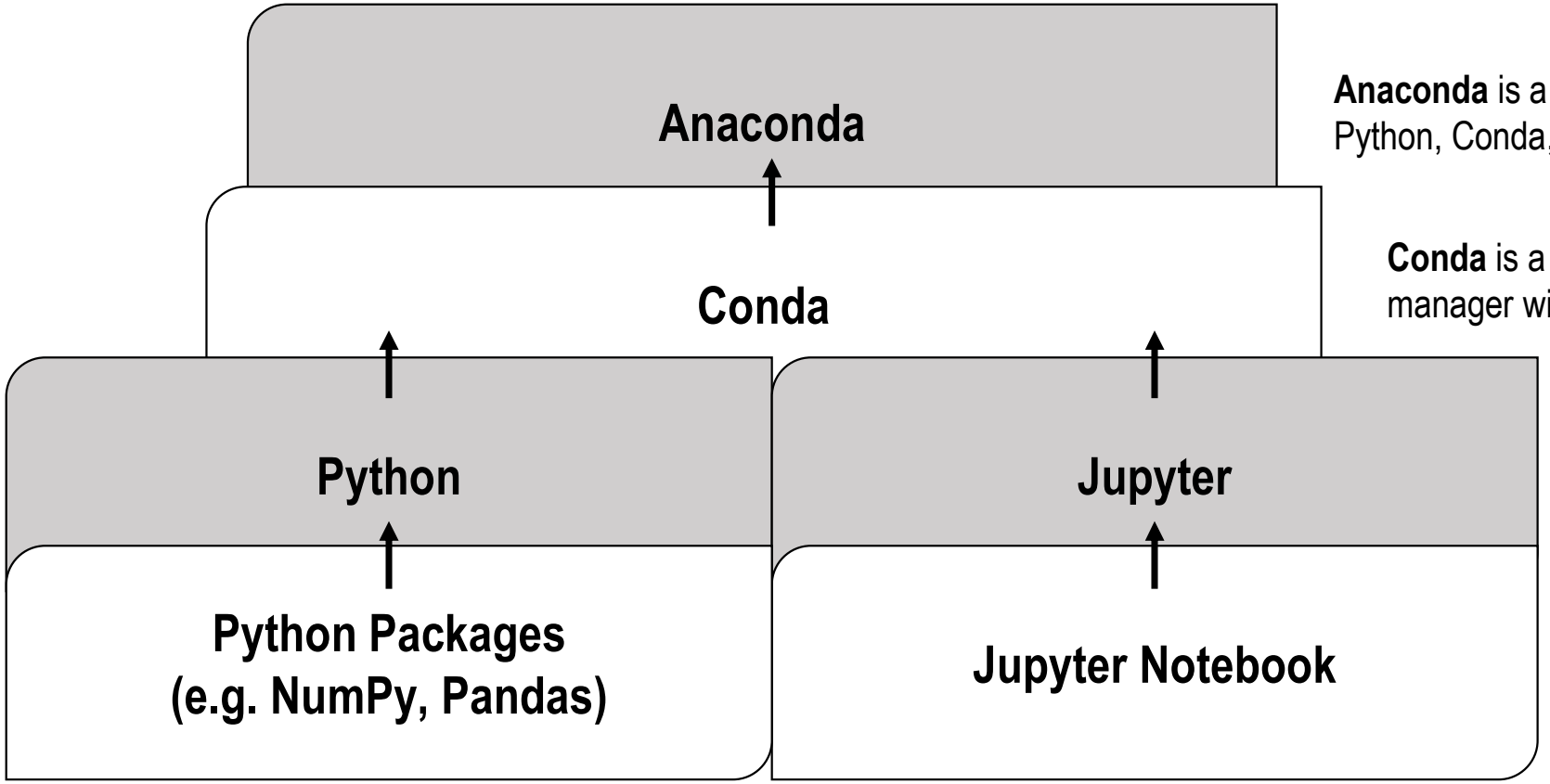
How Python, Conda, and Jupyter fit together?



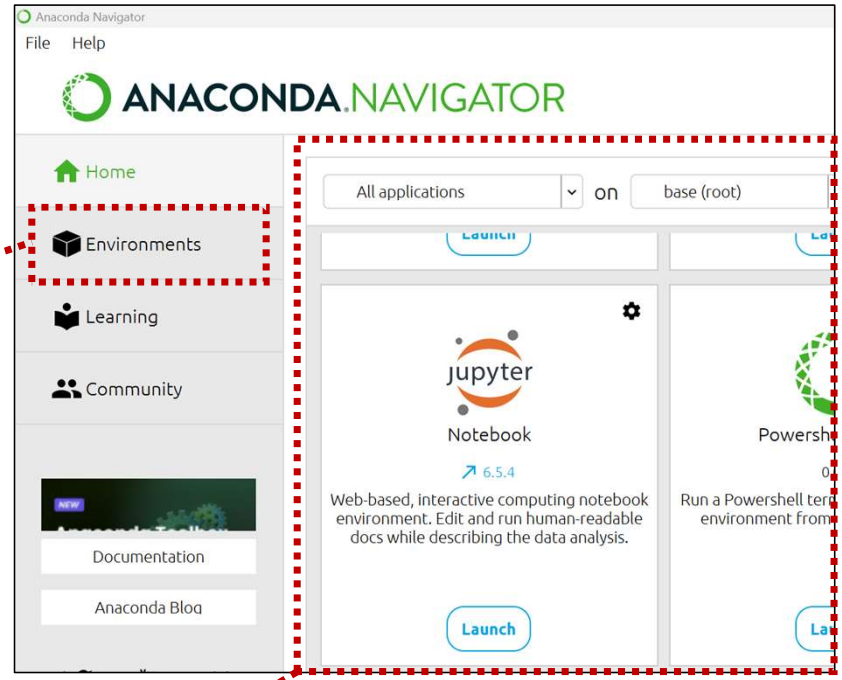
Anaconda is a distribution that includes Python, Conda, and Jupyter tools.

Conda is a package and environment manager within Anaconda.

Python is the programming language used. **Python packages** add functionality and are managed by Conda within environments.

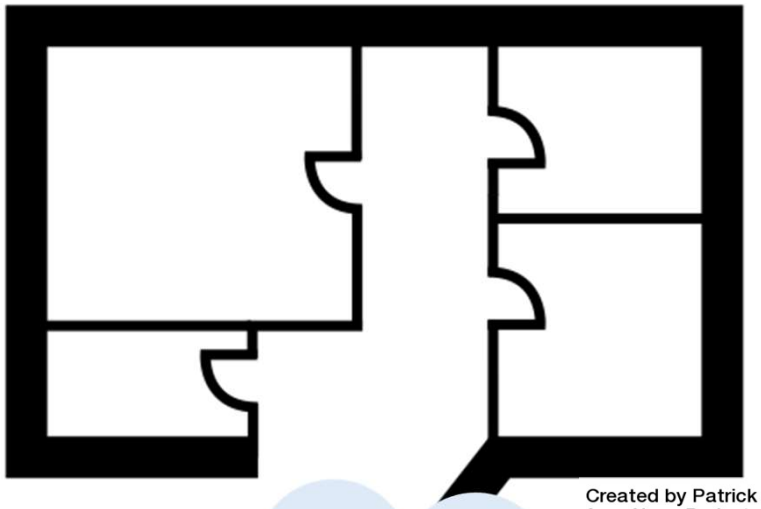
Jupyter is a tool for creating and sharing interactive notebooks. **Jupyter Notebook** is a web-based interface to Jupyter.

Explore Anaconda



Installed environments

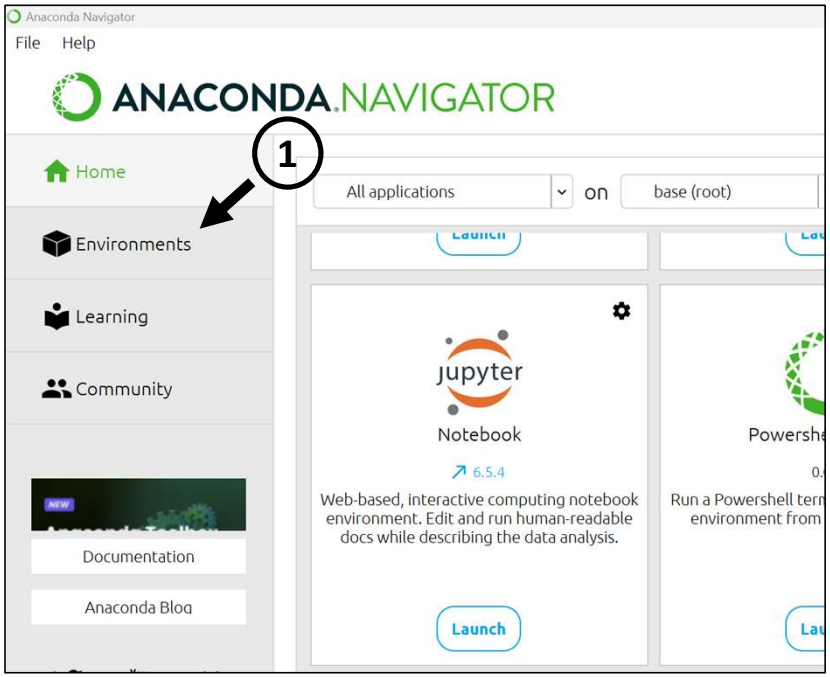
Applications available through Anaconda

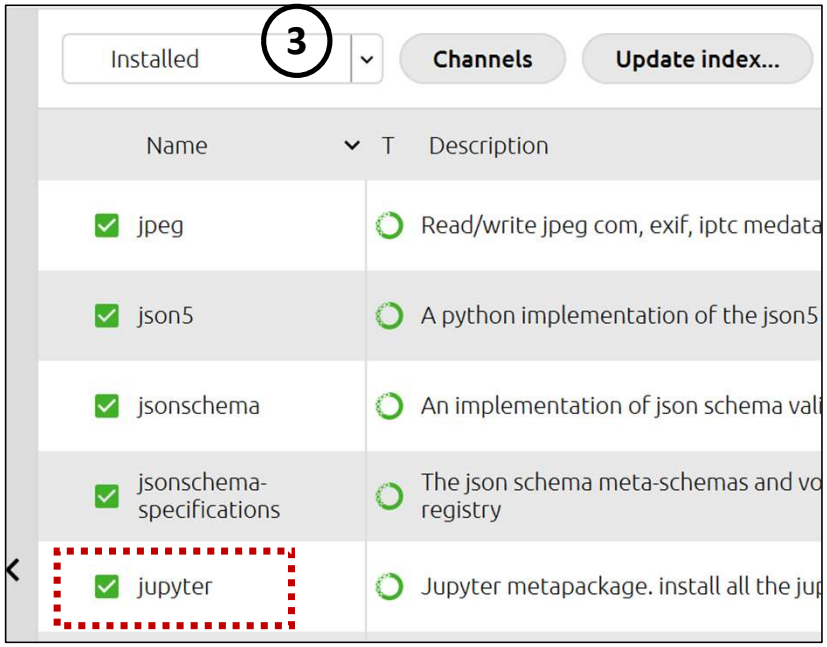


Created by Patrick Multani from Noun Project

Environments are like seperate rooms in a house

Python packages



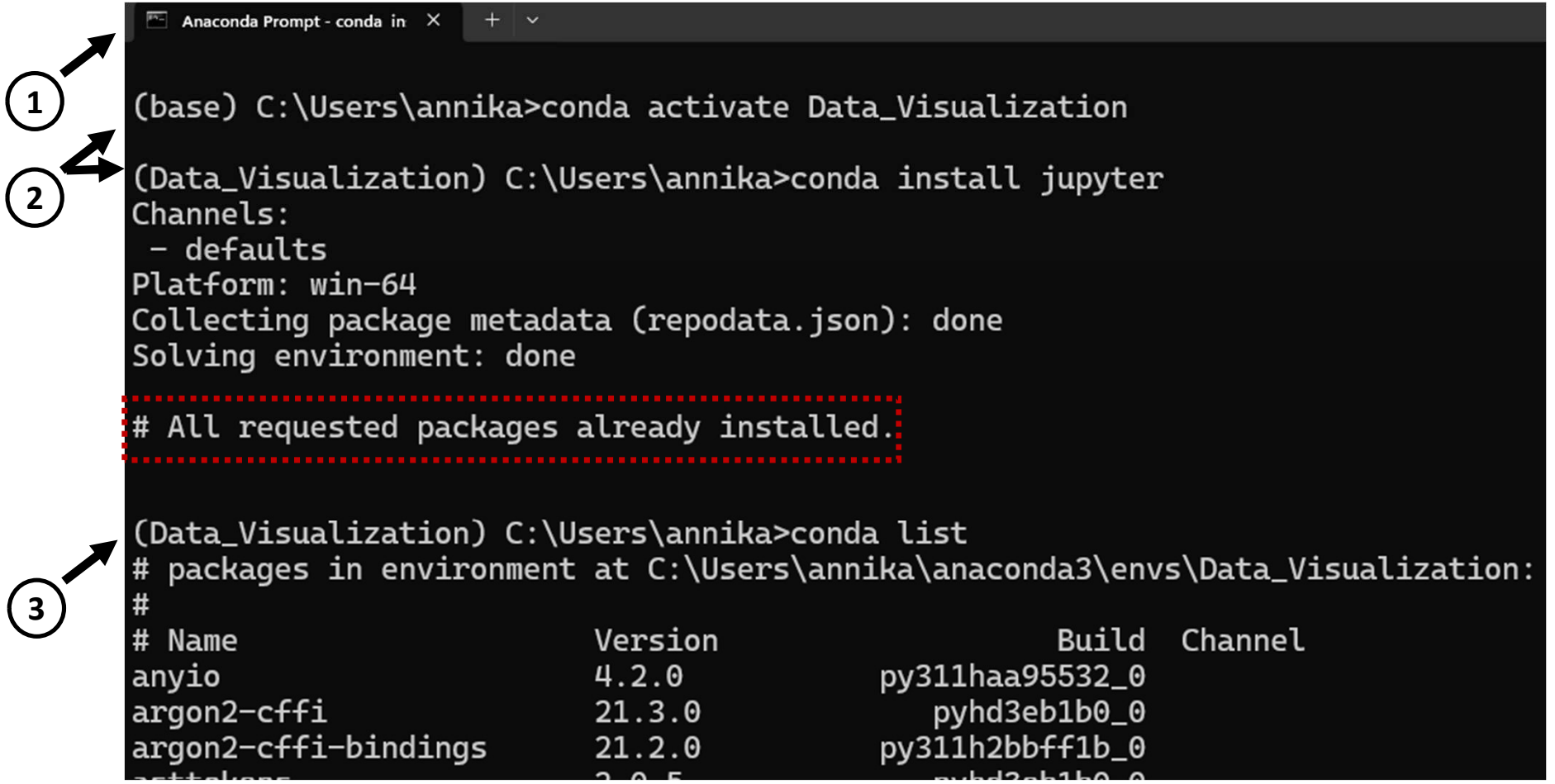


Jupyter is installed!

or

...

Python packages

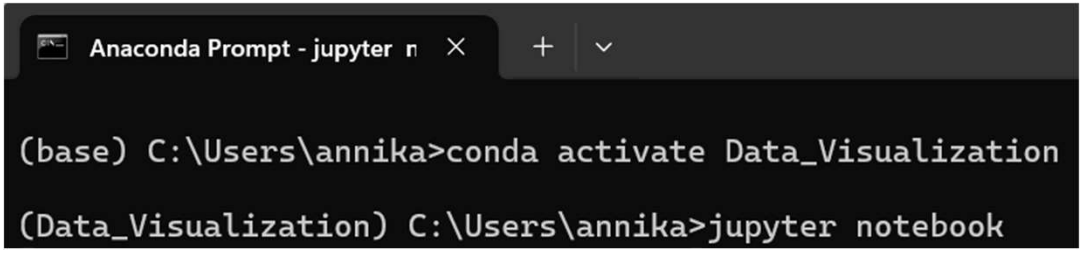


Launch Jupyter Notebook

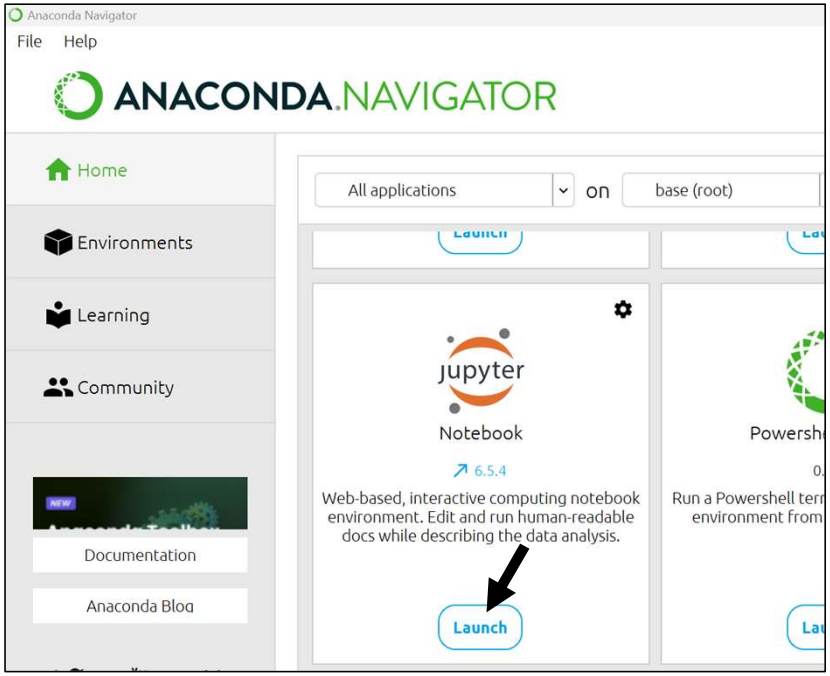
University of Bremen

What is Jupyter Notebook?

- A web-based interactive computing platform
- Write and execute code, visualize data, and add narrative text all in one place
- One out of multiple options to execute Python code

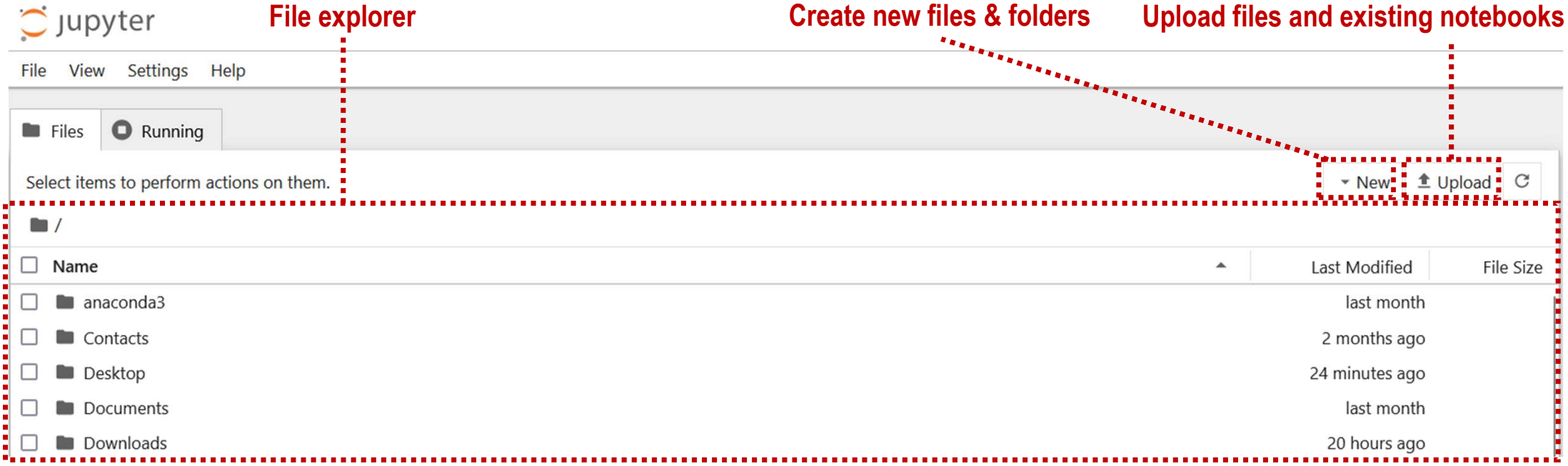


or

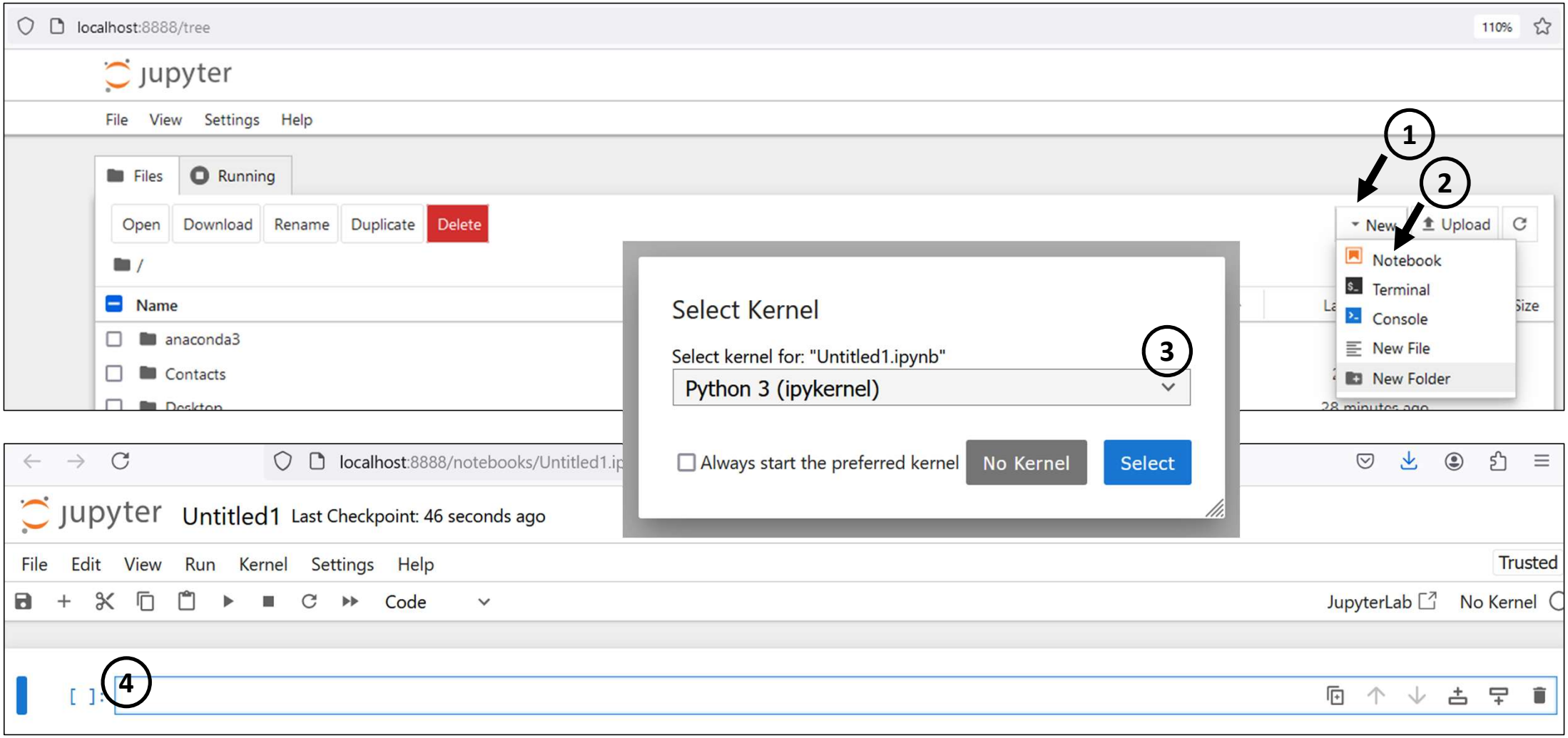


Jupyter home screen

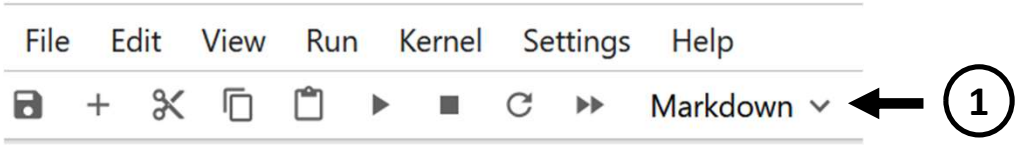
University of Bremen

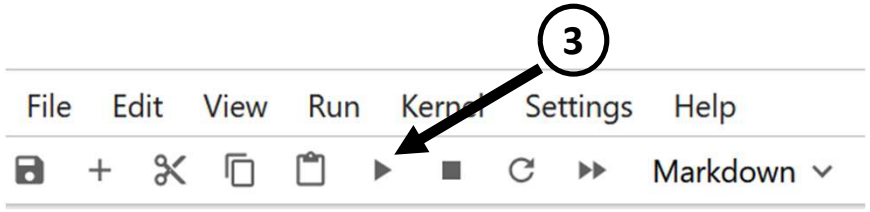


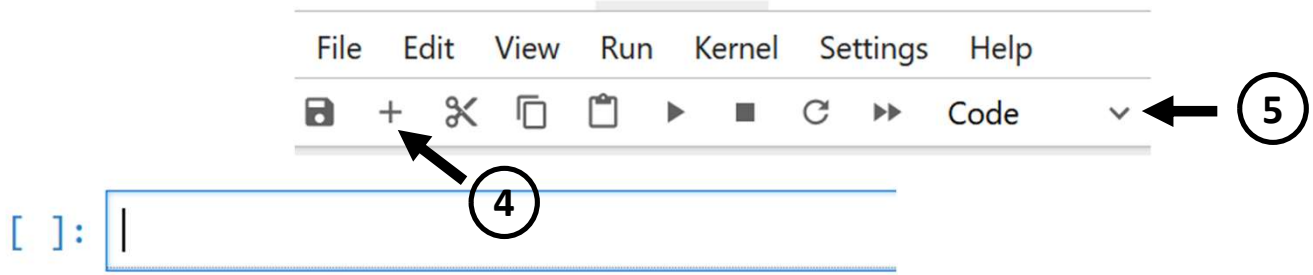
Open a new Jupyter Notebook



Jupyter cell executions







What Python code looks like

University of Bremen

1. Variables and data types

```
my_integer = 10
my_float = 3.14
my_string = "Hello, Python!"
print(my_integer)
print(my_float)
print(my_string)
```

2. Basic operations

```
sum_result = my_integer + 5
product_result = my_float * 2
string_result = my_string + " Hello, world!"
print("Sum:", sum_result, "Product:", product_result, "Concatenation:", string_result)
```

3. Conditionals

```
if my_integer > 5:
    print("my_integer is greater than 5")
else:
    print("my_integer is less than or equal to 5")
```

4. Loops

```
print("Counting from 1 to 3:")
for i in range(1, 4):
    print(i)
```

5. Functions

```
def greet(name):
    return f"Hello, {name}!"

print(greet("Ana"))
print(greet("Alex"))
```

```
# This is a comment!
1 + 1
```

```
2
```

6. Lists and list comprehensions

```
squares = [x * x for x in range(1, 6)]
print("Squares of numbers from 1 to 5:", squares)
```

What Python code looks like

University of Bremen

1. Variables and data types

```
my_integer = 10
my_float = 3.14
my_string = "Hello, Python!"
print(my_integer)
print(my_float)
print(my_string)
```

```
10
3.14
Hello, Python!
```

4. Loops

```
print("Co
for i in
    print
```

```
Counting from 1 to 3:
1
2
3
```

5. Functions

```
def greet(name):
    return f"Hello, {name}!"

print(greet("Ana"))
print(greet("Alex"))
```

```
Hello, Ana!
Hello, Alex!
```

2. Basic operations

```
Sum: 15 Product: 6.28 Concatenation: Hello, Python! Hello, world!
```

```
sum_result = my_integer + 5
product_result = my_float * 2
string_result = my_string + " Hello, world!"
print("Sum:", sum_result, "Product:", product_result, "Concatenation:", string_result)
```

```
# This is a comment!
1 + 1

2
```

3. Conditionals

```
my_integer is greater than 5
```

```
if my_integer > 5:
    print("my_integer is greater than 5")
else:
    print("my_integer is less than or equal to 5")
```

6. Lists and list comprehensions

```
squares = [x * x for x in range(1, 6)]
print("Squares of numbers from 1 to 5:", squares)
```

```
Squares of numbers from 1 to 5: [1, 4, 9, 16, 25]
```

Pandas and NumPy

Use-case: Iris dataset

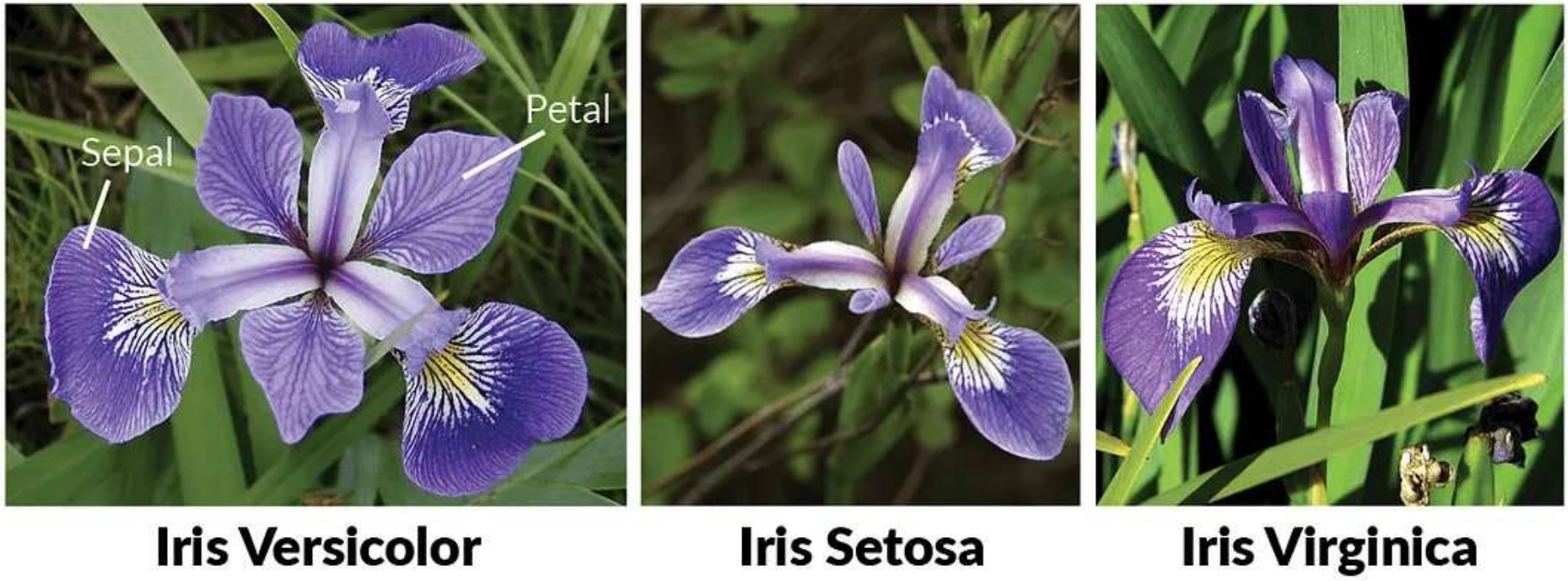


Figure from: https://www.datacamp.com/tutorial/machine-learning-in-r

Introduction to Pandas and DataFrames

University of Bremen

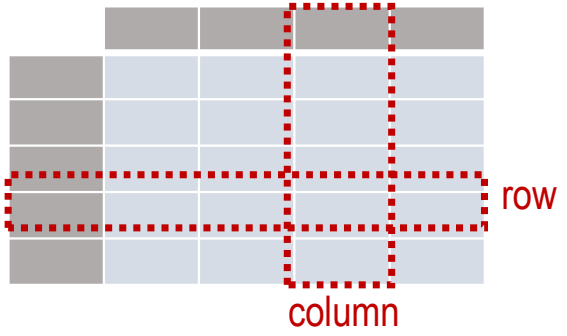
What is Pandas?

- The go-to Python library for handling and analyzing data in almost all data science tasks.
- Works with tabular data using **DataFrames**, similar to Excel spreadsheets or database tables.
- Simplifies data cleaning, transformation, and complex data analysis.
- Reading and writing data from CSV, Excel, and other commonly used formats in research

What is a DataFrame?

- 2-dimensional data structure
- Has labeled axes (rows and columns)
- Rows and columns are indexed for efficient data retrieval

DataFrame:



Introduction to Pandas and DataFrames

University of Bremen

1. Import packages

```
import numpy as np
import pandas as pd
```

2. Import the Iris dataset into a Pandas DataFrame

```
url = 'https://github.com/anolte-DSC/Python_for_Earth_Sciences/blob/main/Quickstart_Python_Jupyter/Datasets/Iris.csv'
data = pd.read_csv(url)
```

3. Explore the data

Plot the first 5 rows of the dataset.

```
data.head(5) # note: indexing starts at 0!
```

How many samples per species?

```
data['Species'].value_counts()
```

Plot basic statistics of the dataset.

```
data.describe()
```

Introduction to Pandas and DataFrames

University of Bremen

4. Data manipulation

Check for missing values.

```
data.isnull().sum()
```

Create a new feature as a function of other features.

```
data['PetalAreaCm2'] = data['PetalLengthCm'] * data['PetalWidthCm']
data.head(5)
```

Introduction to NumPy and arrays

University of Bremen

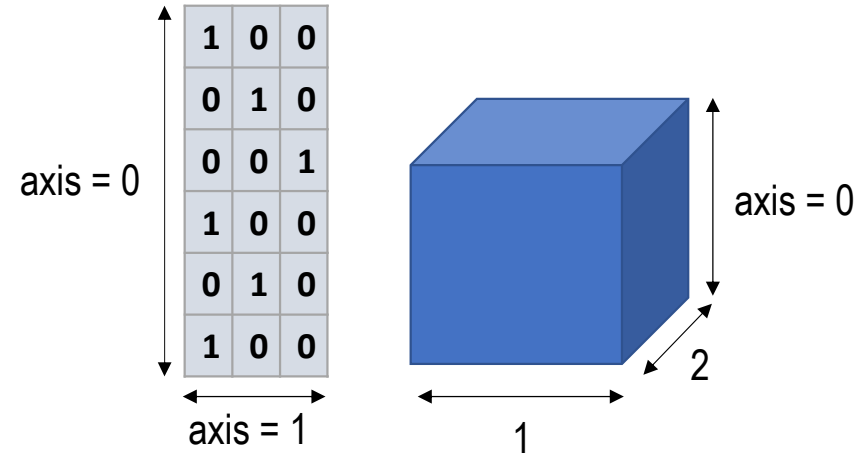
What is NumPy?

- The fundamental library for numerical computing.
- Support for large, multi-dimensional arrays and matrices.
- Offers a vast collection of mathematical functions to operate on these arrays.
- Enables efficient and fast computations, often used as a base for other scientific libraries like Pandas.

What is an array?

- A data structure for storing elements of the same type.
- Can be one-dimensional (like a list) or multi-dimensional (like a matrix).
- Supports vectorized operations, meaning you can perform operations on entire arrays without using loops.

Array with shape=(6,3):



Introduction to NumPy and arrays

University of Bremen

5. Data analysis with NumPy

Convert DataFrame to NumPy array.

```
# Factorize the species column to convert it into integers:
data_int = data.copy()
data_int['Species'], unique_species = pd.factorize(data['Species'])
species_mapping = dict(enumerate(unique_species)) # mapping of species to integers
print("Species to integer mapping:", species_mapping)
# Receive NumPy aray from DataFrame
data_array = data_int.iloc[:, 1:].values
data_array
```

Similar calculations possible with NumPy & Pandas:

```
species = data_array[:, -2]  # select the column with the species
unique_species, counts = np.unique(species, return_counts=True)
print("Species and their counts:", dict(zip(unique_species, counts)))
```

Introduction to NumPy and arrays

University of Bremen

Compute mean and standard deviation for each feature.

```
mean_values = np.mean(data_array, axis=0)
std_deviation = np.std(data_array, axis=0)
mean_values = mean_values.round(2)
std_deviation = std_deviation.round(2)
print("Mean of each feature:", mean_values)
print("Standard deviation of each feature:", std_deviation)
```

Calculate Pearson correlation with continuous numerical variables.

```
correlation_matrix = np.corrcoef(data_array.T)  # Transpose due to expectation for the input format: variables (features) as rows, observations as columns
print("Correlation matrix:\n", correlation_matrix)
```

Select data.

```
filtered_data = data_array[data_array[:, 2] > 2.0] # Example: Filter data where petal length (third column) is greater than 2.0 cm
print('Full data sample size:', data_array.size)
print('Filtered data sample size:', filtered_data.size)
```

Pandas DataFrames vs. NumPy arrays

University of Bremen

Feature	Pandas DataFrames	Numpy Arrays
Purpose	Data manipulation and analysis	Numerical computations
Data Structure	Heterogeneous (different data types)	Homogeneous (same data types)
Operations	Filtering, grouping, merging, reshaping	Mathematical, linear algebra, simulations
Use cases	Data cleaning, exploratory analysis	Scientific computing, machine learning