

PYTHON

Fundamentals, Data Manipulation

Author: Eng. Carlos Andrés Sierra, M.Sc.
cavirguezs@udistrital.edu.co

Full-time Adjunct Professor
Computer Engineering Program
School of Engineering
Universidad Distrital Francisco José de Caldas

2026-I



Outline

- 1 Foundations of Python
- 2 Numerical Analysis with Numpy
- 3 Data Manipulation with Pandas



Outline

- 1 Foundations of Python
- 2 Numerical Analysis with Numpy
- 3 Data Manipulation with Pandas



Basic Definitions in Python

- **Python** is a **high-level**, **interpreted**, and **general-purpose** programming language.
- **High-level** means that **Python** is designed to be easy to read and write.
- **Interpreted** means that **Python** code is executed line by line, rather than being compiled into machine code.
- **Weakly typed** means that **Python** does not require you to declare the type of a variable.
- **Multiparadigm** means that **Python** supports **object-oriented**, **imperative**, and **functional** programming styles.
- **Snake-case** is the convention of writing variable names in **lowercase**, with **underscores** between words.



Basic Definitions in Python

- **Python** is a **high-level**, **interpreted**, and **general-purpose** programming language.
- **High-level** means that **Python** is designed to be easy to read and write.
- **Interpreted** means that **Python** code is executed line by line, rather than being compiled into machine code.
- **Weakly typed** means that **Python** does not require you to declare the type of a variable.
- **Multiparadigm** means that **Python** supports **object-oriented**, **imperative**, and **functional** programming styles.
- **Snake-case** is the convention of writing variable names in **lowercase**, with **underscores** between words.



Basic Definitions in Python

- **Python** is a **high-level**, **interpreted**, and **general-purpose** programming language.
- **High-level** means that **Python** is designed to be easy to read and write.
- **Interpreted** means that **Python** code is executed line by line, rather than being compiled into machine code.
- **Weakly typed** means that **Python** does not require you to declare the type of a variable.
- **Multiparadigm** means that **Python** supports **object-oriented**, **imperative**, and **functional** programming styles.
- **Snake-case** is the convention of writing variable names in **lowercase**, with **underscores** between words.




Basic Definitions in Python








- **Python** is a **high-level**, **interpreted**, and **general-purpose** programming language.
- **High-level** means that **Python** is designed to be easy to read and write.
- **Interpreted** means that **Python** code is executed line by line, rather than being compiled into machine code.
- **Weakly typed** means that **Python** does not require you to declare the type of a variable.
- **Multiparadigm** means that **Python** supports **object-oriented**, **imperative**, and **functional** programming styles.
- **Snake-case** is the convention of writing variable names in **lowercase**, with **underscores** between words.










Popular Developer Tools

Life is Short, Use Dev Tools 








Development Env

-  VS Code
-  Visual Studio
-  IntelliJ IDEA
-  Notepad++
-  Vim
-  PyCharm
-  Jupyter Notebook







Diagramming

-  draw.io
-  excalidraw
-  mindmap
-  Mermaid
-  PlantUML
-  Microsoft Visio
-  Miro







AI Tools

-  ChatGPT
-  GitHub Copilot
-  Tabnine
-  Claude
-  ollama
-  Midjourney
-  Stable Diffusion






Hosting & Deployment

-  AWS
-  Cloudflare
-  Git Hub
-  Fly.io
-  Heroku
-  Digital Ocean







Code Quality

-  Jest
-  ESLint
-  Selenium
-  sonarQube
-  FindBugs
-  Checkstyle






Security

-  1Password
-  LastPass
-  OWASP
-  snyk
-  Nmap

Note-taking

-  Notion
-  Markdown
-  Obsidian
-  Roam
-  Logseq
-  Tiddly Wiki

Design

-  Figma
-  Sketch
-  Adobe Illustrator
-  Canva
-  Adobe Photoshop



Virtual Environments

- **Virtual environments** are a way to create isolated spaces on your computer for **Python projects**.
- **Virtual environments** allow you to **install packages** and **dependencies** for a specific project without affecting other projects.
- **Virtual environments** are created using the **venv** module, which is included in the Python standard library.
- **Virtual environments** are activated using the **source** command in the terminal.
- **Virtual environments** are deactivated using the **deactivate** command in the terminal.



Virtual Environments

- **Virtual environments** are a way to create isolated spaces on your computer for **Python projects**.
- **Virtual environments** allow you to **install packages** and **dependencies** for a specific project without affecting other projects.
- **Virtual environments** are created using the **venv** module, which is included in the Python standard library.
- **Virtual environments** are activated using the **source** command in the terminal.
- **Virtual environments** are deactivated using the **deactivate** command in the terminal.



Virtual Environments

- **Virtual environments** are a way to create isolated spaces on your computer for **Python projects**.
- **Virtual environments** allow you to **install packages** and **dependencies** for a specific project without affecting other projects.
- **Virtual environments** are created using the **venv** module, which is included in the Python standard library.
- **Virtual environments** are activated using the **source** command in the terminal.
- **Virtual environments** are deactivated using the **deactivate** command in the terminal.



Modules and Packages

- **Modules** are **Python files** that contain **functions**, **classes**, and **variables**.
- **Modules** are used to **organize code** and **make it reusable**.
- **Packages** are **directories** that contain **Python files** (modules).
- **Packages** are used to **organize modules** and **make them reusable**.



Modules and Packages

- **Modules** are **Python files** that contain **functions**, **classes**, and **variables**.
- **Modules** are used to **organize code** and **make it reusable**.
- **Packages** are **directories** that contain **Python files** (modules).
- **Packages** are used to **organize modules** and **make them reusable**.



Dependencies Management

- **Dependencies** are **external libraries** or **packages** that your **Python project** relies on.
- **Dependencies** are traditionally managed using a **pyproject.toml** file, which lists the **names** and **versions** of the packages required by your project.
- **Dependencies** can be installed using the **pip** package manager, which is included with Python.
- Currently, the **poetry** package manager is recommended for managing dependencies in Python projects. It follows the **PEP 508** standard for specifying dependencies.



Dependencies Management

- **Dependencies** are **external libraries** or **packages** that your **Python project** relies on.
- **Dependencies** are traditionally managed using a **pyproject.toml** file, which lists the **names** and **versions** of the packages required by your project.
- **Dependencies** can be installed using the **pip** package manager, which is included with Python.
- Currently, the **poetry** package manager is recommended for managing dependencies in Python projects. It follows the **PEP 508** standard for specifying dependencies.



Jupyter Notebooks

- **Jupyter Notebooks** are a **web-based interactive computing environment** that allows you to create and share documents that contain **live code**, **equations**, **visualizations**, and **narrative text**.
- Jupyter Notebooks are used for data cleaning, data transformation, statistical modeling, data visualization, machine learning, and more.
- Jupyter Notebooks support multiple programming languages, including Python, R, and Julia.
- Jupyter Notebooks are used by data scientists, researchers, students, and educators to explore data, create reports, and teach programming.



Jupyter Notebooks

- **Jupyter Notebooks** are a **web-based interactive computing environment** that allows you to create and share documents that contain **live code**, **equations**, **visualizations**, and **narrative text**.
- **Jupyter Notebooks** are used for **data cleaning**, **data transformation**, **statistical modeling**, **data visualization**, **machine learning**, and **more**.
- Jupyter Notebooks support multiple programming languages, including Python, R, and Julia.
- Jupyter Notebooks are used by data scientists, researchers, students, and educators to explore data, create reports, and teach programming.



Jupyter Notebooks

- **Jupyter Notebooks** are a **web-based interactive computing environment** that allows you to create and share documents that contain **live code**, **equations**, **visualizations**, and **narrative text**.
- **Jupyter Notebooks** are used for **data cleaning**, **data transformation**, **statistical modeling**, **data visualization**, **machine learning**, and **more**.
- **Jupyter Notebooks** support **multiple programming languages**, including **Python**, **R**, and **Julia**.
- **Jupyter Notebooks** are used by **data scientists**, **researchers**, **students**, and **educators** to **explore data**, **create reports**, and **teach programming**.



Variables and Memory Management

Variables Definition

Variables are used to store data values. Python has no command for declaring a variable. A variable is created the moment you first assign a value to it.



Conditionals

Definition

Conditionals are used to execute different code blocks based on *different conditions*.

Nested Conditionals

Nested conditionals are conditionals that are *inside* other conditionals.

Elif Conditionals

Elif conditionals are used to check *multiple conditions*.



Conditionals

Definition

Conditionals are used to execute different code blocks based on *different conditions*.

Nested Conditionals

Nested conditionals are conditionals that are *inside* other conditionals.

Elif Conditionals

Elif conditionals are used to check *multiple conditions*.



Conditionals

Definition

Conditionals are used to execute different code blocks based on *different conditions*.

Nested Conditionals

Nested conditionals are conditionals that are *inside* other conditionals.

Elif Conditionals

Elif conditionals are used to check *multiple conditions*.



Loops and Range

Loops Definition

Loops are used to execute a block of code *multiple times*.

Range Definition

The **range function** is used to generate a sequence of numbers.



Loops and Range

Loops Definition

Loops are used to execute a block of code *multiple times*.

Range Definition

The **range function** is used to generate a sequence of numbers.

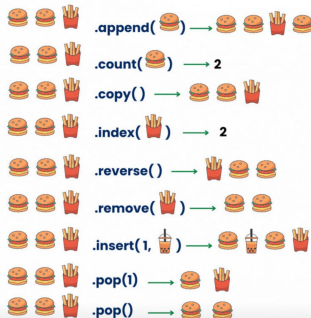


Lists

Definition

A **list** is a collection of items that are *ordered* and *changeable*. **Lists** are defined by enclosing the items in square brackets.

PYTHON LIST METHODS



Dictionaries

Definition

A **dictionary** is a collection of items that are *unordered*, *changeable*, and *indexed*. Dictionaries are defined by enclosing the items in curly braces.



Sets and Tuples

Definition Sets

A **set** is a collection of items that are *unordered* and *unindexed*. **Sets** are defined by enclosing the items in curly braces.

Definition Tuples

A **tuple** is a collection of items that are *ordered* and *unchangeable*. **Tuples** are defined by enclosing the items in parentheses.



Sets and Tuples

Definition Sets

A **set** is a collection of items that are *unordered* and *unindexed*. **Sets** are defined by enclosing the items in curly braces.

Definition Tuples

A **tuple** is a collection of items that are *ordered* and *unchangeable*. **Tuples** are defined by enclosing the items in parentheses.



Lists Comprehensions

Definition

List comprehensions provide a concise way to create lists. Common applications are to make *new lists* where each element is the result of some operation applied to each member of another sequence or iterable, or to create a subsequence of those elements that satisfy a certain condition.



Functions

Definition

A **function** is a block of code that only runs when it is called. You can pass data, known as parameters, into a function. A **function** can return data as a result.

Type of Functions

- Built-in Functions
- User-defined Functions
 - Variadic Functions
 - Recursive Functions



Functions

Definition

A **function** is a block of code that only runs when it is called. You can pass data, known as parameters, into a function. A **function** can return data as a result.

Type of Functions

- **Built-in** Functions
- **User-defined** Functions
 - **Variadic** Functions
 - **Recursive** Functions



Python Iterators

Definition

An **iterator** is an object that contains a *countable number of values*. An **iterator** is an object that can be iterated upon, meaning that you can traverse through all the values.

Maps

The **map function** is used to apply a function to *all the items* in an input list.

Filters

The **filter function** is used to *select items* from an input list that meet a certain condition.



Python Iterators

Definition

An **iterator** is an object that contains a *countable number of values*. An **iterator** is an object that can be iterated upon, meaning that you can traverse through all the values.

Maps

The **map function** is used to apply a function to *all the items* in an input list.

Filters

The **filter function** is used to *select items* from an input list that meet a certain condition.



Python Iterators

Definition

An **iterator** is an object that contains a *countable number of values*. An **iterator** is an object that can be iterated upon, meaning that you can traverse through all the values.

Maps

The **map function** is used to apply a function to *all the items* in an input list.

Filters

The **filter function** is used to *select items* from an input list that meet a certain condition.



Lambda Functions

Definition

A **lambda function** is a small anonymous function. A **lambda function** can take any number of arguments, but can only have one expression.



Classes and Objects

Definition

Python is an *object-oriented programming* language. Almost everything in Python is an **object**, with its *properties* and *methods*. A **class** is like an object constructor, or a “blueprint” for creating objects.



Outline

- 1 Foundations of Python
- 2 Numerical Analysis with Numpy
- 3 Data Manipulation with Pandas



Numerical Python Library — Numpy

- **Numpy** is the **core** library for **scientific computing** in Python. It is the **fundamental package** for scientific computing with **Python**.
- **Numpy** is a general-purpose **array-processing** package. It provides a **high-performance** multidimensional array object, and tools for working with these arrays.
- **Numpy** was created by **Travis Oliphant** in 2005, and it is an **open-source project**. Currently, **Numpy version 2.4** is available.



Numerical Python Library — Numpy

- **Numpy** is the **core** library for **scientific computing** in Python. It is the **fundamental package** for scientific computing with **Python**.
- **Numpy** is a general-purpose **array-processing** package. It provides a **high-performance** multidimensional array object, and tools for working with these arrays.
- **Numpy** was created by **Travis Oliphant** in 2005, and it is an **open-source project**. Currently, **Numpy version 2.4** is available.



Lineal Algebra with Numpy

- **Numpy** provides a **comprehensive set** of **linear algebra** functions.
- **Numpy** provides the **functionality** to **create** and **manipulate matrices**.
- **Numpy** provides the **functionality** to solve linear systems of equations.
- **Numpy** provides the **functionality** to calculate the determinant of a matrix.
- **Numpy** provides the **functionality** to calculate the inverse of a matrix.



Lineal Algebra with Numpy

- **Numpy** provides a **comprehensive set** of **linear algebra** functions.
- **Numpy** provides the **functionality** to **create** and **manipulate matrices**.
- **Numpy** provides the **functionality** to **solve linear systems** of equations.
- **Numpy** provides the **functionality** to **calculate the determinant** of a matrix.
- **Numpy** provides the **functionality** to **calculate the inverse** of a matrix.



Lineal Algebra with Numpy

- **Numpy** provides a **comprehensive set** of **linear algebra** functions.
- **Numpy** provides the **functionality** to **create** and **manipulate matrices**.
- **Numpy** provides the **functionality** to **solve linear systems** of equations.
- **Numpy** provides the **functionality** to **calculate** the **determinant** of a matrix.
- **Numpy** provides the **functionality** to **calculate** the **inverse** of a matrix.



Lineal Algebra with Numpy

- **Numpy** provides a **comprehensive set** of **linear algebra** functions.
- **Numpy** provides the **functionality** to **create** and **manipulate matrices**.
- **Numpy** provides the **functionality** to **solve linear systems** of equations.
- **Numpy** provides the **functionality** to **calculate** the **determinant** of a matrix.
- **Numpy** provides the **functionality** to **calculate** the **inverse** of a matrix.



Vectorization with Numpy

- **Vectorization** is the **process** of **converting** an **algorithm** from **operating** on a single value at a time to **operating** on a set of values at one time.
- **Vectorization** is the **process** of **replacing** explicit **loops** with **array expressions** or **matrix operations**.
- The advantages of vectorization are speed and clarity. The disadvantages are memory and complexity.
- **Numpy** provides the functionality to vectorize operations on arrays.



Vectorization with Numpy

- **Vectorization** is the **process** of **converting** an **algorithm** from **operating** on a single value at a time to **operating** on a set of values at one time.
- **Vectorization** is the **process** of **replacing** explicit **loops** with **array expressions** or **matrix operations**.
- The **advantages** of **vectorization** are **speed** and **clarity**. The **disadvantages** are **memory** and **complexity**.
- **Numpy** provides the **functionality** to **vectorize operations** on **arrays**.



Typical Operations with Numpy

- **Numpy** provides the **functionality** to **create** and **manipulate** arrays.
- **Numpy** provides the **functionality** to **perform** element-wise operations on **arrays**. **matrix operations** on **arrays**.
- **Numpy** provides the **functionality** to **perform** linear algebra operations on **arrays**.
- **Numpy** provides the **functionality** to **perform** statistical operations on **arrays**.



Outline

- 1 Foundations of Python
- 2 Numerical Analysis with Numpy
- 3 Data Manipulation with Pandas



Introduction to Pandas

- **Pandas** is a fast, powerful, flexible, and easy-to-use open-source data manipulation and data analysis library built on top of the Python programming language.
- **Pandas** is a high-level data manipulation tool developed by Wes McKinney in 2008.
- Pandas is a fast and efficient data manipulation tool that is built on top of NumPy.
- Pandas is one of the most popular and widely-used data manipulation libraries in the world.



Introduction to Pandas

- **Pandas** is a fast, powerful, flexible, and easy-to-use open-source data manipulation and data analysis library built on top of the Python programming language.
- **Pandas** is a high-level data manipulation tool developed by Wes McKinney in 2008.
- **Pandas** is a fast and efficient data manipulation tool that is built on top of NumPy.
- **Pandas** is one of the most popular and widely-used data manipulation libraries in the world.



The “Series” Data Structure

- A **Series** is a **one-dimensional array-like object** that contains a **sequence of values** and an **associated array of data labels**, called the **index**.
- The **index** of a **Series** is an **array of labels** that **correspond** to the **values** in the **Series**. The **index** of a **Series** is an **optional parameter** that **defaults** to a **sequence of integers** starting at **zero**.
- The **Series object** is a **core data structure** in **Pandas**.



Querying a Series

- You could **query** a **Series** using **indexing** (**boolean** or **fancy**).
- You could **query** a **Series** using **loc** and **iloc** indexers.



Querying a Series

- You could **query** a **Series** using **indexing** (**boolean** or **fancy**).
- You could **query** a **Series** using **loc** and **iloc** indexers.



The “DataFrame” Data Structure

- A **DataFrame** is a two-dimensional labeled data structure with columns of potentially different types.
- A **DataFrame** is a tabular data structure that is similar to a spreadsheet or a SQL table.
- A **DataFrame** is a core data structure in **Pandas**. It is a two-dimensional size-mutable data structure with labeled axes (rows and columns).
- A **DataFrame** is a container for Series objects.



The “DataFrame” Data Structure

- A **DataFrame** is a two-dimensional labeled data structure with columns of potentially different types.
- A **DataFrame** is a tabular data structure that is similar to a spreadsheet or a SQL table.
- A **DataFrame** is a core data structure in **Pandas**. It is a two-dimensional size-mutable data structure with labeled axes (rows and columns).
- A **DataFrame** is a container for **Series** objects.



DataFrame Indexing and Loading

- You could **index** a **DataFrame** using **column names**.
- You could **load** a **DataFrame** from a **CSV** file.
- You could **load** a **DataFrame** from a **JSON** file.
- You could **load** a **DataFrame** from a **SQL** database.



DataFrame Indexing and Loading

- You could **index** a **DataFrame** using **column names**.
- You could **load** a **DataFrame** from a **CSV file**.
- You could **load** a **DataFrame** from a **JSON file**.
- You could **load** a **DataFrame** from a **SQL database**.



DataFrame Indexing and Loading

- You could **index** a **DataFrame** using **column names**.
- You could **load** a **DataFrame** from a **CSV file**.
- You could **load** a **DataFrame** from a **JSON file**.
- You could **load** a **DataFrame** from a **SQL database**.



DataFrame Indexing and Loading

- You could **index** a **DataFrame** using **column names**.
- You could **load** a **DataFrame** from a **CSV file**.
- You could **load** a **DataFrame** from a **JSON file**.
- You could **load** a **DataFrame** from a **SQL database**.



DateTime Handling in Pandas

- You could **convert** a **string** to a **datetime** object using the **to_datetime()** method.
- You could **convert** a **datetime** object to a **string** using the **strftime()** method.
- You could **convert** a **datetime** object to a **timestamp** using the **timestamp()** method.



DateTime Handling in Pandas

- You could **convert** a **string** to a **datetime** object using the **to_datetime()** method.
- You could **convert** a **datetime** object to a **string** using the **strftime()** method.
- You could **convert** a **datetime** object to a **timestamp** using the **timestamp()** method.



DateTime Handling in Pandas

- You could **convert** a **string** to a **datetime** object using the **to_datetime()** method.
- You could **convert** a **datetime** object to a **string** using the **strftime()** method.
- You could **convert** a **datetime** object to a **timestamp** using the **timestamp()** method.



Querying a DataFrame

- You could **query** a **DataFrame** using **indexing** (**boolean** or **fancy**).
- You could **query** a **DataFrame** using **loc** and **iloc** indexers.
- You could **query** a **DataFrame** using **query** method.



Querying a DataFrame

- You could **query** a **DataFrame** using **indexing** (**boolean** or **fancy**).
- You could **query** a **DataFrame** using **loc** and **iloc** indexers.
- You could **query** a **DataFrame** using **query** method.



Querying a DataFrame

- You could **query** a **DataFrame** using **indexing** (**boolean** or **fancy**).
- You could **query** a **DataFrame** using **loc** and **iloc** indexers.
- You could **query** a **DataFrame** using **query method**.



Missing Values in a DataFrame

- You could **detect** missing values in a **DataFrame**. The **isnull()** method returns a **Boolean DataFrame** indicating the **presence** of missing values.
- You could **fill** missing values in a **DataFrame**. The **fillna()** method returns a **DataFrame** with missing values filled.
- You could **drop** missing values in a **DataFrame**. The **dropna()** method returns a **DataFrame** with missing values dropped.



Missing Values in a DataFrame

- You could **detect** missing values in a **DataFrame**. The **isnull()** method returns a **Boolean DataFrame** indicating the **presence** of missing values.
- You could **fill** missing values in a **DataFrame**. The **fillna()** method returns a **DataFrame** with missing values filled.
- You could **drop** missing values in a **DataFrame**. The **dropna()** method returns a **DataFrame** with missing values dropped.



Missing Values in a DataFrame

- You could **detect** missing values in a **DataFrame**. The **isnull()** method returns a **Boolean DataFrame** indicating the **presence** of missing values.
- You could **fill** missing values in a **DataFrame**. The **fillna()** method returns a **DataFrame** with missing values filled.
- You could **drop** missing values in a **DataFrame**. The **dropna()** method returns a **DataFrame** with missing values dropped.



Merging DataFrames

- You could **merge** two **DataFrames** using the **merge()** method.
- You could **concatenate** two **DataFrames** using the **concat()** method.
- You could **join** two **DataFrames** using the **join()** method.



Merging DataFrames

- You could **merge** two **DataFrames** using the **merge()** method.
- You could **concatenate** two **DataFrames** using the **concat()** method.
- You could **join** two **DataFrames** using the **join()** method.



Merging DataFrames

- You could **merge** two **DataFrames** using the **merge()** method.
- You could **concatenate** two **DataFrames** using the **concat()** method.
- You could **join** two **DataFrames** using the **join()** method.



GroupBy in Pandas

- You could **group** a **DataFrame** using the **groupby()** method.
- You could **aggregate** a **DataFrame** using the **agg()** method.
- You could **transform** a **DataFrame** using the **transform()** method.
- You could **filter** a **DataFrame** using the **filter()** method.



GroupBy in Pandas

- You could **group** a **DataFrame** using the **groupby()** method.
- You could **aggregate** a **DataFrame** using the **agg()** method.
- You could **transform** a **DataFrame** using the **transform()** method.
- You could **filter** a **DataFrame** using the **filter()** method.



GroupBy in Pandas

- You could **group** a **DataFrame** using the **groupby()** method.
- You could **aggregate** a **DataFrame** using the **agg()** method.
- You could **transform** a **DataFrame** using the **transform()** method.
- You could **filter** a **DataFrame** using the **filter()** method.



GroupBy in Pandas

- You could **group** a **DataFrame** using the **groupby()** method.
- You could **aggregate** a **DataFrame** using the **agg()** method.
- You could **transform** a **DataFrame** using the **transform()** method.
- You could **filter** a **DataFrame** using the **filter()** method.



Pandas Idioms

- You could **apply functions** to **DataFrames**.
- You could **chain methods** in **Pandas**.



Pandas Idioms

- You could **apply functions** to **DataFrames**.
- You could **chain methods** in **Pandas**.



Outline

- 1 Foundations of Python
- 2 Numerical Analysis with Numpy
- 3 Data Manipulation with Pandas



Thanks!

Questions?



Repo: <https://github.com/EngAndres/ud-public/tree/main/courses/data-analysis-programming>

