

Data Analysis with Python

Cristo Daniel Alvarado

January 21, 2026

Contents

| | | |
|----------|----------------------------|----------|
| 1 | Python Basics | 3 |
| 1.1 | Basics | 3 |
| 1.1.1 | Strings | 3 |
| 1.1.2 | f-Strings | 3 |
| 1.1.3 | RegEx | 5 |
| 2 | Data Structures | 8 |
| 2.1 | Lists and Tuples | 8 |

Code List

| | | |
|-----|-------------------|---|
| 1.1 | caption | 4 |
| 1.2 | caption | 4 |
| 1.3 | caption | 4 |
| 1.4 | caption | 5 |
| 1.5 | caption | 5 |

CHAPTER 1

PYTHON BASICS

This chapter contains the basics of Python syntax for data analysis.

1.1 BASICS

Definition 1.1.1 (Types)

Type is how Python represents different types of data.

Observation 1.1.1

To get the type of a variable use the `type()` method.

Also, we can convert some type to other using the respective method for converting, for example `float(2)` converts the integer 2 into the float 2.0.

Also, in Python we have `int`, `float`, `string` and `boolean`.

Observation 1.1.2

The operation `//` is the integer division.

1.1.1 STRINGS

For a `string`, we can obtain his length with the method `len`, also, we can multiply a string by an integer `n` and the string is `n`-times multiplied.

Also, we can do `.upper()` and `.lower()` after the variable to change to upper or lower case, respectively. Also, we can replace strings inside a string using the `.replace(,)` method (which requires the string to replace and the one to be replaced with).

Additionally, we can find substrings in a string using the `.find(,)` method (which returns the position where the string to find is first located).

1.1.2 F-STRINGS

Introduced in Python 3.6, f-strings are a new way to format strings in Python. They are prefixed with 'f' and use curly braces to enclose the variables that will be formatted. For example:

```
1 name = "John"
2 age = 30
3 print(f"My name is {name} and I am {age} years old.")
```

This will output:

```
1 My name is John and I am 30 years old.
```

Another way to format strings in Python is using `str.format()`. It uses curly braces `{}` as placeholders for variables which are passed as arguments in the `format()` method. For example:

```
1 name = "John"
2 age = 50
3 print("My name is {} and I am {} years old.".format(name, age))
```

This will output:

```
1 My name is John and I am 50 years old.
```

Code 1.1: caption

Observation 1.1.3

Each of these methods has its own advantages and use cases. However, f-strings are generally considered the most modern and preferred way to format strings in Python due to their readability and performance.

Idea 1.1.1

F-strings are also able to evaluate expressions inside the curly braces, which can be very handy. For example:

```
1 x = 10
2 y = 20
3 print(f"The sum of x and y is {x+y}.")
```

This will output:

```
1 The sum of x and y is 30.
```

In Python, raw strings are a powerful tool for handling textual data, especially when dealing with escape characters. By prefixing a string literal with the letter `'r'`, Python treats the string as raw, meaning it interprets backslashes as literal characters rather than escape sequences.

Consider the following examples of regular string and raw string:

Regular string:

```
1 regular_string = "C:\new_folder\file.txt"
2 print("Regular String:", regular_string)
```

Code 1.2: caption

```
1 Regular String: C:
2 ew_folderile.txt
```

Code 1.3: caption

In the regular string `regular_string` variable, the backslashes (`\n`) are interpreted as escape sequences. Therefore, `\n` represents a newline character, which would lead to an incorrect file path representation.

Raw string:

```
1 raw_string = r"C:\new_folder\file.txt"
2 print("Raw String:", raw_string)
```

Code 1.4: caption

This will output:

```
1 Raw String: C:\new_folder\file.txt
```

Code 1.5: caption

However, in the raw string `raw_string`, the backslashes are treated as literal characters. This means that `\n` is not interpreted as a newline character, but rather as two separate characters, `\` and `n`. Consequently, the file path is represented exactly as it appears.

Observation 1.1.4

Indirectly, we will be working with arrays, so that we can select certain pieces of information of a string using `[n:m]`, where we indicate the interval from n to $m - 1$ of the string which we want to use.

This is called a slice.

Also, we can take every 2 numbers or 3, using `[::2]` and `[::3]`, respectively.

1.1.3 REGEX

In Python, RegEx (short for Regular Expression) is a tool for matching and handling strings.

This RegEx module provides several functions for working with regular expressions, including `search`, `split`, `findall`, and `sub`.

Python provides a built-in module called `re`, which allows you to work with regular expressions. First, import the `re` module.

Observation 1.1.5

Regular expressions (RegEx) are patterns used to match and manipulate strings of text. There are several special sequences in RegEx that can be used to match specific characters or patterns.

| Term | Definition |
|-------------------------|--|
| AI | AI (artificial intelligence) is the ability of a digital computer or computer-controlled robot to perform tasks commonly associated with intelligent beings. |
| Application development | Application development, or app development, is the process of planning, designing, creating, testing, and deploying a software application to perform various business operations. |
| Arithmetic Operations | Arithmetic operations are the basic calculations we make in everyday life like addition, subtraction, multiplication and division. It is also called as algebraic operations or mathematical operations. |
| Array of numbers | Set of numbers or objects that follow a pattern presented as an arrangement of rows and columns to explain multiplication. |

| Term | Definition |
|-------------------------------|--|
| Assignment operator in Python | Assignment operator is a type of Binary operator that helps in modifying the variable to its left with the use of its value to the right. The symbol used for assignment operator is " = ". |
| Asterisk | Symbol "*" used to perform various operations in Python. |
| Backslash | A backslash is an escape character used in Python strings to indicate that the character immediately following it should be treated in a special way, such as being treated as escaped character or raw string. |
| Boolean | Denoting a system of algebraic notation used to represent logical propositions by means of the binary digits 0 (false) and 1 (true). |
| Colon | A colon is used to represent an indented block. It is also used to fetch data and index ranges or arrays. |
| Concatenate | Link (things) together in a chain or series. |
| Data engineering | Data engineers are responsible for turning raw data into information that an organization can understand and use. Their work involves blending, testing, and optimizing data from numerous sources. |
| Data science | Data Science is an interdisciplinary field that focuses on extracting knowledge from data sets which are typically huge in amount. The field encompasses analysis, preparing data for analysis, and presenting findings to inform high-level decisions in an organization. |
| Data type | Data type refers to the type of value a variable has and what type of mathematical, relational or logical operations can be applied without causing an error. |
| Double quote | Symbol " " used to represent strings in Python. |
| Escape sequence | An escape sequence is two or more characters that often begin with an escape character that tell the computer to perform a function or command. |
| Expression | An expression is a combination of operators and operands that is interpreted to produce some other value. |
| Float | Python float () function is used to return a floating-point number from a number or a string representation of a numeric value. |
| Forward slash | Symbol "/" used to perform various operations in Python |
| Foundational | Denoting an underlying basis or principle; fundamental. |
| Immutable | Immutable Objects are of in-built datatypes like int, float, bool, string, Unicode, and tuple. In simple words, an immutable object can't be changed after it is created. |
| Integer | An integer is the number zero (0), a positive natural number (1, 2, 3, and so on) or a negative integer with a minus sign (-1, -2, -3, and so on.) |
| Manipulate | Is the process of modifying a string or creating a new string by making changes to existing strings. |
| Mathematical conventions | A mathematical convention is a fact, name, notation, or usage which is generally agreed upon by mathematicians. |
| Mathematical expressions | Expressions in math are mathematical statements that have a minimum of two terms containing numbers or variables, or both, connected by an operator in between. |
| Mathematical operations | The mathematical "operation" refers to calculating a value using operands and a math operator. |
| Negative indexing | Allows you to access elements of a sequence (such as a list, a string, or a tuple) from the end, using negative numbers as indexes. |

| Term | | Definition |
|--------------------|-----------|--|
| Operands | | The quantity on which an operation is to be done. |
| Operators | in Python | Operators are used to perform operations on variables and values. |
| Parentheses | | Parentheses is used to call an object. |
| Replicate | | To make an exact copy of. |
| Sequence | | A sequence is formally defined as a function whose domain is an interval of integers. |
| Single quote | | Symbol ' ' used to represent strings in python. |
| Slicing | in Python | Slicing is used to return a portion from defined list. |
| Special characters | | A special character is one that is not considered a number or letter. Symbols, accent marks, and punctuation marks are considered special characters. |
| Stride value | | Stride is the number of bytes from one row of pixels in memory to the next row of pixels in memory. |
| Strings | | In Python, Strings are arrays of bytes representing Unicode characters. |
| Substring | | A substring is a sequence of characters that are part of an original string. |
| Type casting | | The process of converting one data type to another data type is called Typecasting or Type Coercion or Type Conversion. |
| Types in Python | | Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data. |
| Variables | | Variables are containers for storing data values. |

Table 1.1: Glossary of Technical Terms

CHAPTER 2

DATA STRUCTURES

2.1 LISTS AND TUPLES

Tuples and lists are called compound data types and *are one of the fundamental structures in Python*.

Definition 2.1.1 (Tuples)

A **Tuple** is an ordered sequence of Python data types.

Example 2.1.1

Example of a tuple is `tuple=('disco',20,1.2)`

We index the same tuple values the same way as an array. We can slice them.

Observation 2.1.1

Tuples are immutable, therefore you cannot change its values. We can assign a different value to a variable.

Therefore, if we want to manipulate a tuple, we must create another variable to save the value of the original variable modified.

Definition 2.1.2 (Lists)

List are ordered sequences and are represented with `[]`.

List are the same as tuples, but Lists are mutable, we can change its values.

Observation 2.1.2

We can extend a list using the `.extend(a)` method (where `a` is a list) or add elements using `.append(b)` (where `b` is a variable).

Also, we can delete elements from a list using `del(A[n])`, where `n` is the position in `A` of the element which we want to delete.

Observation 2.1.3

Also, we can convert a string into a list using `.split()`.

| Package/Method | Description | Code Example |
|---------------------|---|--|
| List Methods | | |
| append() | Adds an element to the end of a list. | <pre> 1 fruits = ["apple", "banana", "orange"] 2 fruits.append("mango") 3 print(fruits) </pre> |
| copy() | Creates a shallow copy of a list. | <pre> 1 my_list = [1, 2, 3, 4, 5] 2 new_list = my_list.copy() 3 print(new_list) </pre> |
| count() | Counts occurrences of a specific element. | <pre> 1 my_list = [1, 2, 2, 3, 4, 2, 5, 2] 2 count = my_list.count(2) 3 print(count) </pre> |
| Creating a list | Creates an ordered, mutable collection. | <pre> 1 fruits = ["apple", "banana", "orange", "mango"] </pre> |
| del | Removes element at specified index. | <pre> 1 my_list = [10, 20, 30, 40, 50] 2 del my_list[2] 3 print(my_list) </pre> |

| Package/Method | Description | Code Example |
|------------------|--|---|
| extend() | Adds multiple elements from an iterable. | <pre> 1 fruits = ["apple", "banana", "orange"] 2 more_fruits = ["mango", "grape"] 3 fruits.extend(more_fruits) 4 print(fruits) </pre> |
| Indexing | Accesses elements by position. | <pre> 1 my_list = [10, 20, 30, 40, 50] 2 print(my_list[0]) 3 print(my_list[-1]) </pre> |
| insert() | Inserts element at specified position. | <pre> 1 my_list = [1, 2, 3, 4, 5] 2 my_list.insert(2, 6) 3 print(my_list) </pre> |
| Modifying a list | Changes values using indexing. | <pre> 1 my_list = [10, 20, 30, 40, 50] 2 my_list[1] = 25 3 print(my_list) </pre> |

| Package/Method | Description | Code Example |
|----------------------|---------------------------------------|---|
| pop() | Removes and returns element at index. | <pre> 1 my_list = [10, 20, 30, 40, 50] 2 removed_element = my_list.pop (2) 3 print(removed_element) 4 print(my_list) </pre> |
| remove() | Removes first occurrence of value. | <pre> 1 my_list = [10, 20, 30, 40, 50] 2 my_list.remove (30) 3 print(my_list) </pre> |
| reverse() | Reverses the order of elements. | <pre> 1 my_list = [1, 2, 3, 4, 5] 2 my_list.reverse () 3 print(my_list) </pre> |
| Slicing | Accesses a range of elements. | <pre> 1 my_list = [1, 2, 3, 4, 5] 2 print(my_list [1:4]) 3 print(my_list [:3]) 4 print(my_list [::2]) </pre> |
| sort() | Sorts elements in ascending order. | <pre> 1 my_list = [5, 2, 8, 1, 9] 2 my_list.sort() 3 print(my_list) </pre> |
| Tuple Methods | | |

| Package/Method | Description | Code Example |
|-----------------|-------------------------------------|---|
| count() | Counts occurrences of a value. | <pre> 1 fruits = ("apple", "banana", "apple", "orange") 2 print(fruits.count("apple")) </pre> |
| index() | Finds first occurrence of value. | <pre> 1 fruits = ("apple", "banana", "orange", "apple") 2 print(fruits.index("apple")) </pre> |
| sum() | Calculates sum of numeric elements. | <pre> 1 numbers = (10, 20, 5, 30) 2 print(sum(numbers)) </pre> |
| min() and max() | Finds smallest/largest element. | <pre> 1 numbers = (10, 20, 5, 30) 2 print(min(numbers)) 3 print(max(numbers)) </pre> |
| len() | Gets number of elements. | <pre> 1 fruits = ("apple", "banana", "orange") 2 print(len(fruits)) </pre> |

Table 2.1: List and Tuple Methods in Python