

2 Introduction to Python

2.1 Introduction

The quality of Python packages can vary considerably and depends, among other things, on the size of the developer community and the level of maintenance a project receives.

Another popular package for data analysis and manipulation is *pandas* (<https://pandas.pydata.org>), which we use only to a limited extent in this book, since the examples focus primarily on *NumPy*-based workflows.

2.2 Getting Started

Help with a particular function can also be found by writing the function name in parentheses after the `help` command in the Console. Graphs are displayed in the *Plots* tab.

2.3 Python Syntax

The built-in Python function `sum()` sums a two-dimensional *NumPy* array row by row and therefore returns a vector containing the column sums. The four results of `sum(A)` are obviously the sums of the elements in each of the four columns of `A`.

in Python together with `print()` in order to actually display the array element located in the third row and second column, which is 9. The indices are therefore specified using the `[row, column]` format and start at 0 in each case.

Both indexing conventions have advantages and disadvantages, but as this book progresses, it will become clear that the indexing used by Python can be an additional source of error for users of other programming languages, for example, when selecting spectral bands from satellite images, such as those from the Hyperion hyperspectral sensor on the EO-1 satellite (Chap. 8.6).

This fact may be familiar to Python users but may lead to errors among users who have switched to Python from other languages. Accordingly, if we want to replace the value 9 in the 3rd row and 2nd column of array `A` with the value 30, we type

and to display the entire array, as shown below:

```
A[2,0:4] = [1,3,3,5]
```

2.4 Array Manipulation

2.5 Data Types in Python

The commutativity of addition [$a+b=b+a$] and multiplication [$a\cdot b=b\cdot a$] still holds, whereas associativity [$(a+(b+c))=(a+b)+c$] does not, as we can see in the following example.

However, the result would be the same (i.e., zero) with real numbers. Distributivity [$a\cdot(b+c)=a\cdot b+a\cdot c$] is also violated, as the following example demonstrates.

We use `random.rand()`, which generates uniformly distributed pseudorandom numbers within the interval (0,1).

The function `random.rand()` and its sister function `random.randn()` (which are used to generate normally distributed pseudorandom numbers)—together with `random.seed(0)` (which is used to define the seed)—are **convenient** functions for users who port code from MATLAB.

Since 8 bits can be used to store 256 different values, this data type can be used to store integer values between 0 and 255, whereas using `int8` to create signed 8-bit integers generates values between -128 and +127, **including 0**.

2.6 Data Storage and Handling

The classic example of a data format that can be used with different computer platforms and software is the *American Standard Code for Information Interchange (ASCII) format*, which was first published in 1963 by the *American Standards Association (ASA)*. As a 7-bit code, **ASCII-1963** consists of $2^7=128$ characters (codes 0 to 127). Since **1963**, a number of variants have appeared in order to facilitate the exchange of text written in languages other than English. Such variants include the **extended (8-bit) ASCII, which was released in 1981, contains $2^8=256$ codes, and includes special symbols and foreign-language characters, and the most recent ASCII-1986, which includes a new set of control characters. The 8-bit *Universal Coded Character Set (UCS) Transformation Format (UTF-8)* is now the most widely used text format for computer code as well as for most web pages on the Internet.**

The simplest way to exchange data between a certain piece of software and Python is by using the **ASCII or UTF-8** format. Although some packages (e.g., **pandas**) provide various functions for reading binary files, such as Microsoft Excel files, most data arrive in the form of ASCII files.

We use `loadtxt()` to perform this task, as shown below:

The character string `'i4,f8,f8'` that is used as input for `dtype()` defines the conversion specifiers, which are enclosed in single quotation marks, where `i4` stands for the 32 bit (or 4 byte) **signed** integers and `f8` stands for 64 bit (or 8 byte) double-precision floating point numbers. The function `loadtxt()` uses `dtype` as an input argument together with `unpack=True` to transpose the returned arrays, and it uses `skiprows=1` to ignore a single header line while reading the file.

We again use `loadtxt()` to read the file:

We skip the header, read the first column (i.e., the sample ID) as a 32 bit **signed** integer **number** (`int32`) with specifier `i4`, read the next three columns (i.e., `X`, `Y`, and `Z`) as 64 bit double-precision floating point numbers (`double`) with specifier `f8`, and then read the date and time as character strings with specifier `U12`.

The parentheses used in `loadtxt()` are also the reason that we do not need the explicit line breaks inside the function in order to write the input arguments on separate lines.

We then write data to the file using the formatting operators `%i` for **signed** integers and `%6.4f` for fixed point numbers with a field width of six characters and four digits after the decimal point.

2.7 Control Flow

2.8 Scripts and Functions

These files contain text and can be edited using a standard text editor. However, the built-in Spyder Editor color highlights various syntax elements: Comments are highlighted in gray; keywords such as *if* and *for* are highlighted in blue; and character strings are highlighted in green.

By Martin Trauth, 24 May 2023

2.9 Basic Visualization Tools

Recommended Reading

Figure captions