

Welcome to the Software Carpentry

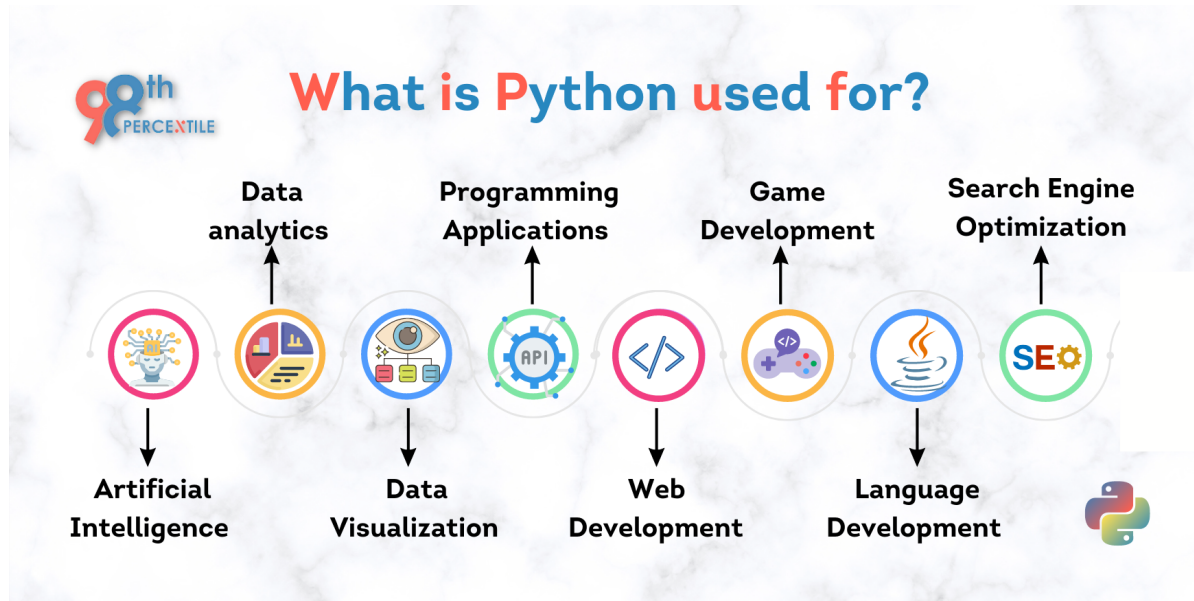


Agenda

- **Python lesson 1 (day 1)**
 - Break 10:45
- Python lesson 2 (day 2)
- Unix Shell (day 3)
- Git (day 4)

Python lesson 1 (day 1)

- Created by Guido van Rossum
- First release in 1991
- Ranks as one of the most popular programming languages

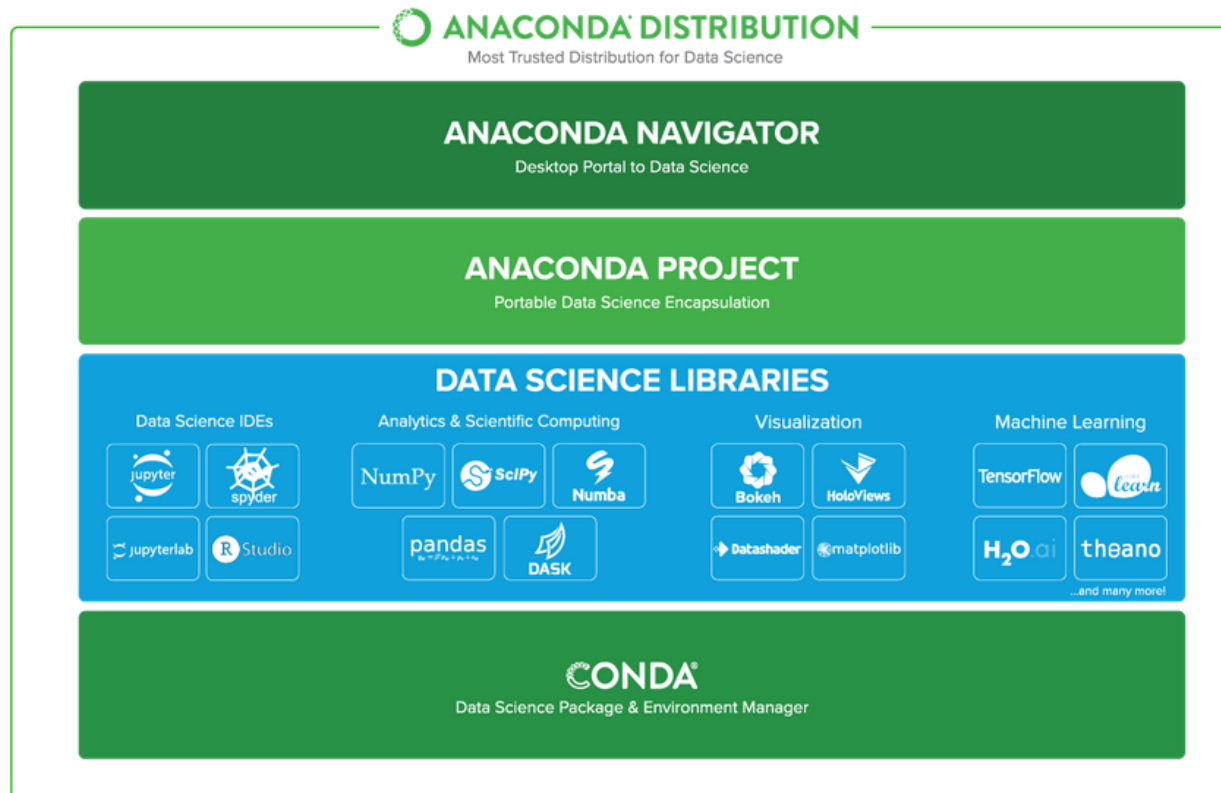


Tools and Setup

- software: Anaconda
 - Jupyter Notebooks

Anaconda

Anaconda Navigator is a graphical user interface to the conda package and environment manager and a free scientific-computing environment.



Installation comes with:

- Latest version of Python
- JupyterLab, Jupyter notebooks and several other IDE options
- An easy-to-install collection of high performance Python libraries
- Conda, tool for managing packages and environments
- Collection of open source packages and tools to install over 1.5k additional packages

Jupyter Notebooks

- Web application that combines computer code, plain language descriptions, data, rich visualizations like 3D models, charts, graphs and figures, and interactive controls.

Advantadges

- Interactive data analysis
- Prototyping
- Visualizations
- Reporting /show casing a tool

Disadvantadges

- Limited collaboration capabilities
- They can become quite large if working with large datasets
- Difficulties with debugging
- Not advisable if working in large software projects

Tools and Setup

- data : <https://swcarpentry.github.io/python-novice-inflammation/>
 - swc-python
 - python-novice-inflammation-code.zip
 - python-novice-inflammation-data.zip

Expectations for the lesson

By the end of the lesson , you should be able to know what are:

- Variables and their types
- Use other python libraries
- Visualize tabular data
- Perform repetitive tasks in a simpler way

Start of the lesson



Key points: Python variables

- Basic data types in Python include integers, strings, and floating-point numbers.
- Use `variable = value` to assign a value to a variable in order to record it in memory.
- Use `print(something)` to display the value of something.
- Use `#` some kind of explanation to add comments to programs.
- Built-in functions are always available to use.

Exercise 1 Python lists

Use slicing to access only the last four characters of a string or entries of a list.

```
string_for_slicing = 'Observation date: 02-Feb-2013'  
list_for_slicing = [['fluorine', 'F'],  
                    ['chlorine', 'Cl'],  
                    ['bromine', 'Br'],  
                    ['iodine', 'I'],  
                    ['astatine', 'At']]
```

so you get this output :

```
'2013'  
[['chlorine', 'Cl'], ['bromine', 'Br'], ['iodine', 'I'], ['astatine', 'At']]
```

Solution

```
string_for_slicing[-4:]  
list_for_slicing[-4:]
```

Keypoints of Lists in Python

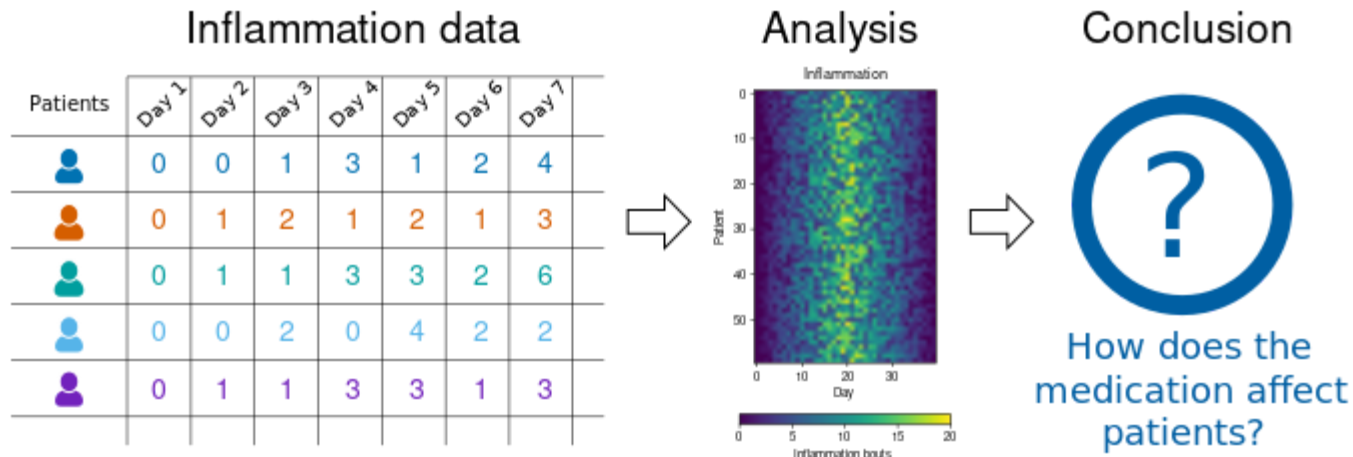
- `[value1, value2, value3, ...]` creates a list.
- Lists can contain any Python object, including lists (i.e., list of lists).
- Lists are indexed and sliced with square brackets (e.g., `list[0]` and `list[2:9]`), in the same way as strings and arrays.
- Lists are mutable (i.e., their values can be changed in place).
- Strings are immutable (i.e., the characters in them cannot be changed).

Our case study

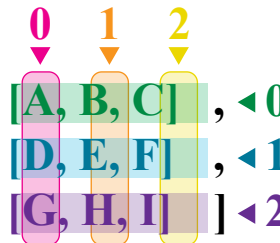
Arthritis Inflammation

We are studying inflammation in patients who have been given a new treatment for arthritis.

There are 60 patients, who had their inflammation levels recorded for 40 days. We want to analyze this data to study the effect of the new arthritis treatment.

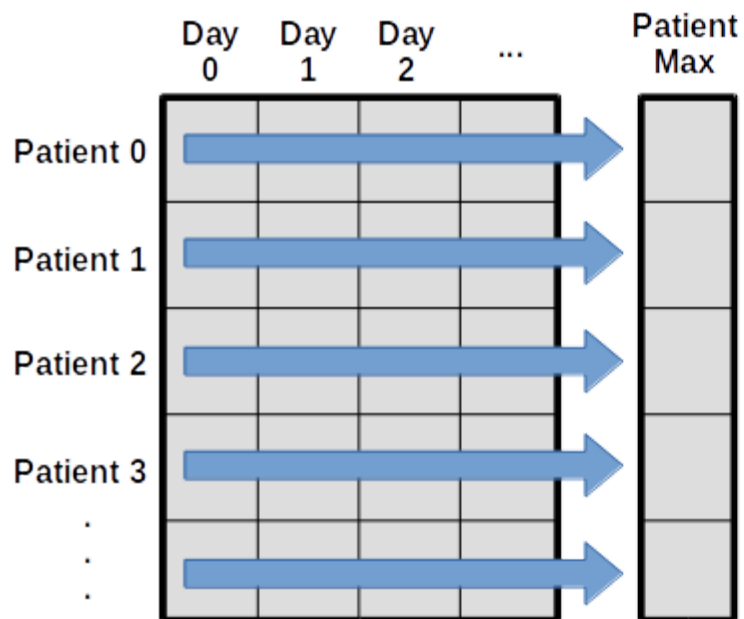


Indexes in Python arrays

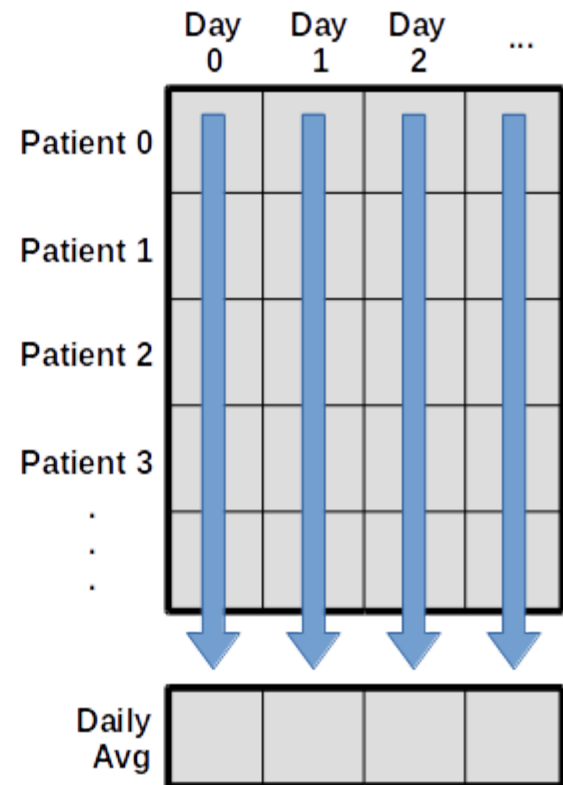
The diagram shows a 3x3 array structure. The first row contains elements A, B, and C, highlighted with a green box. The second row contains D, E, and F, highlighted with a blue box. The third row contains G, H, and I, highlighted with a purple box. Above the first row, three arrows point down to the columns: a pink arrow labeled '0' for the first column, an orange arrow labeled '1' for the second column, and a yellow arrow labeled '2' for the third column. To the right of each row, there is a label: '◀0' for the first row, '◀1' for the second row, and '◀2' for the third row.

```
data = [ [A, B, C], [D, E, F], [G, H, I] ]
```

data[0, 0] =	A	data[0, 1] =	B	data[0, 2] =	C
data[1, 0] =	D	data[1, 1] =	E	data[1, 2] =	F
data[2, 0] =	G	data[2, 1] =	H	data[2, 2] =	I



Max for each patient
`numpy.max(data, axis=1)`



Average for each day
`numpy.mean(data, axis=0)`

Exercise 2 Change in inflammation

The patient data is longitudinal in the sense that each row represents a series of observations relating to one individual. This means that the change in inflammation over time is a meaningful concept. The `numpy.diff()` function takes an array and returns the differences between two successive values. Let's use it to examine the changes each day across the first week of patient 3 from our inflammation dataset.

```
patient3_week1 = data[3, :7]
patient3_week1 = [0. 0. 2. 0. 4. 2. 2.] #Note that the array of differences is shorter by one element
```

Q1: When calling `numpy.diff` with a multi-dimensional array, an axis argument may be passed to the function to specify which axis to process. When applying `numpy.diff` to our 2D inflammation array data, which axis would we specify?

Q2: If the shape of an individual data file is (60, 40) (60 rows and 40 columns), what would the shape of the array be after you run the `diff()` function and why?

Solution

Q1: Since the row axis (0) is patients, it does not make sense to get the difference between two arbitrary patients. The column axis (1) is in days, so the difference is the change in inflammation – a meaningful concept.

```
numpy.diff(data, axis=1)
```

Q2: The shape will be (60, 39) because there is one fewer difference between columns than there are columns in the data.

Key points of Analysing tabular data with Python

- Import a library into a program using `import libraryname`.
- Use the numpy library to work with arrays in Python.
- The expression `array.shape` gives the shape of an array.
- Use `array[x, y]` to select a single element from a 2D array.
- Array indices start at 0, not 1.
- Use `low:high` to specify a slice that includes the indices from low to high-1.
- Use `numpy.mean(array)`, `numpy.amax(array)`, and `numpy.amin(array)` to calculate simple statistics.
- Use `numpy.mean(array, axis=0)` or `numpy.mean(array, axis=1)` to calculate statistics across the specified axis.

Break



