LIBRE HUB
Latam Hub for Bioimaging through Open Hardware

PoL
Physics of Life
TU Dresden

CENTER FOR
SYSTEMS BIOLOGY
DRESDEN

Chan
Zuckerberg
Initiative

Python
Data structures

Marcelo Leomil Zoccoler

With materials from Robert Haase

August 2023

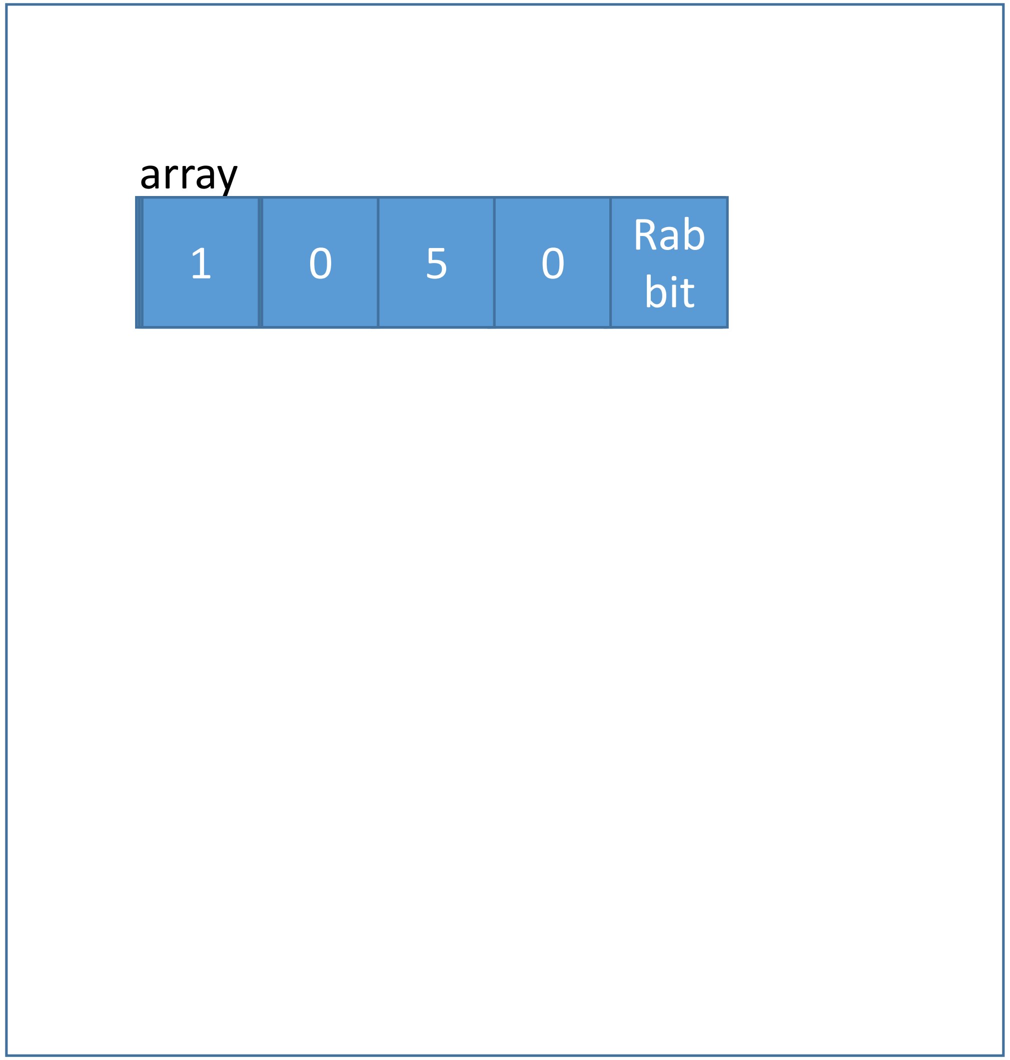
Lists

- Lists are variables, where you can store multiple values

Give me a “0”, five times!

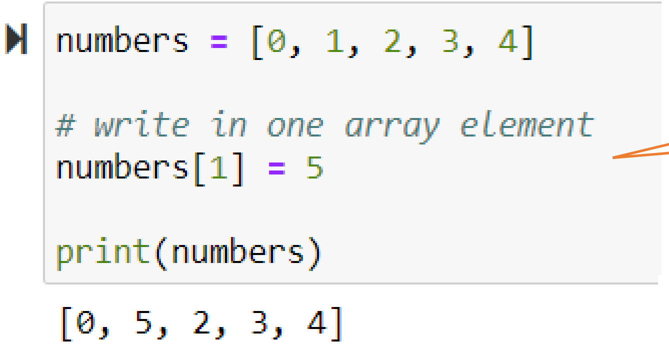
```
array = [0] * 5
```

Computer memory



Arrays in Python

- Modifying lists entries



```
numbers = [0, 1, 2, 3, 4]

# write in one array element
numbers[1] = 5

print(numbers)
```

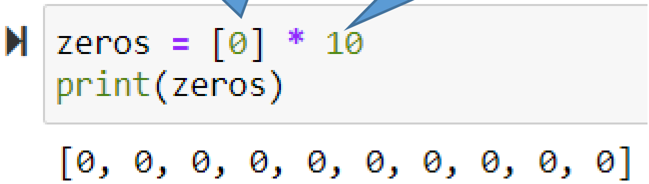
```
[0, 5, 2, 3, 4]
```

Note: The first element has index 0!

- Creating lists of defined size

What?

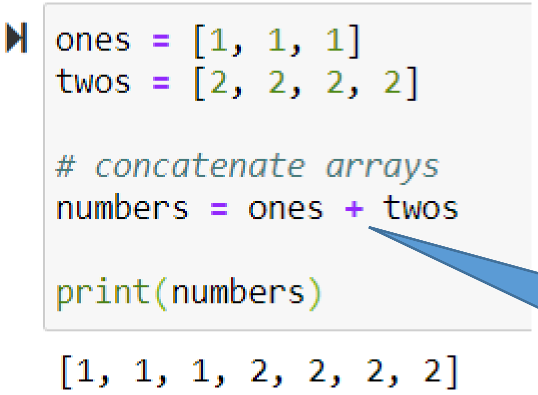
How many?



```
zeros = [0] * 10
print(zeros)
```

```
[0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
```

- Concatenating lists

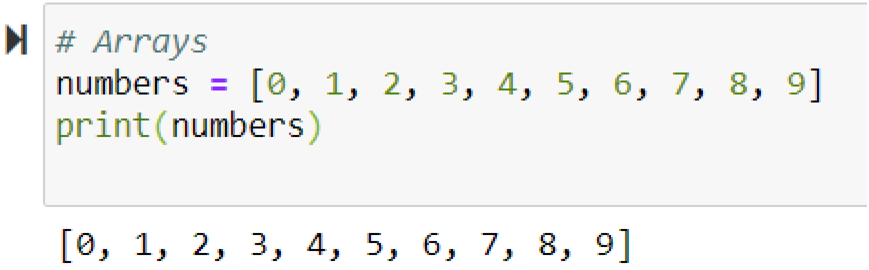


```
ones = [1, 1, 1]
twos = [2, 2, 2, 2]

# concatenate arrays
numbers = ones + twos

print(numbers)
```

```
[1, 1, 1, 2, 2, 2, 2]
```

+ means appending

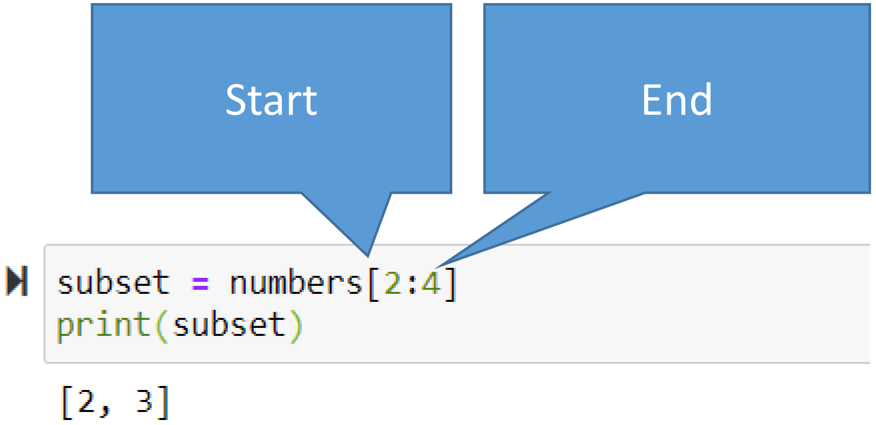
Subsets



```
# Arrays
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
print(numbers)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

- Creating subsets of lists

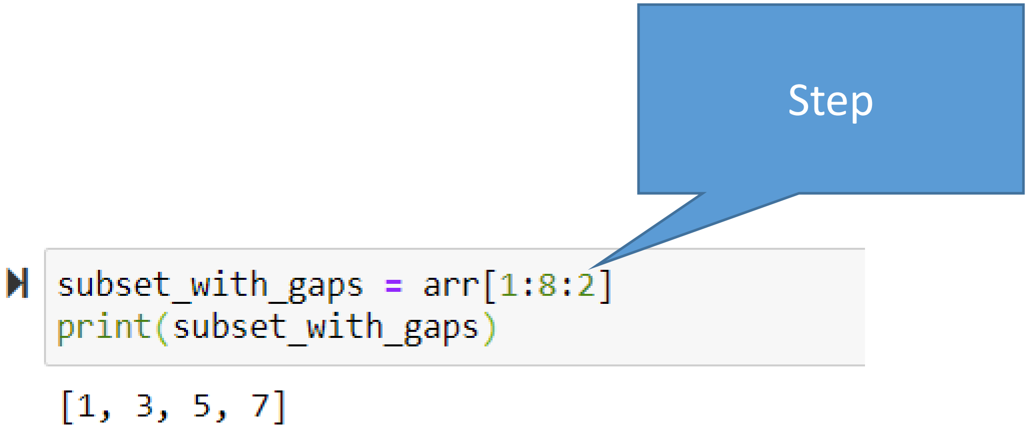


Start

End

```
subset = numbers[2:4]
print(subset)
```

```
[2, 3]
```



Step

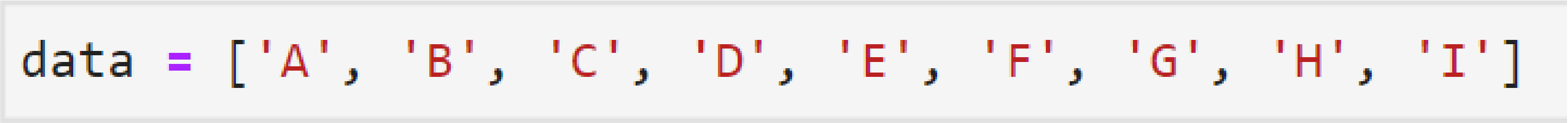
```
subset_with_gaps = arr[1:8:2]
print(subset_with_gaps)
```

```
[1, 3, 5, 7]
```

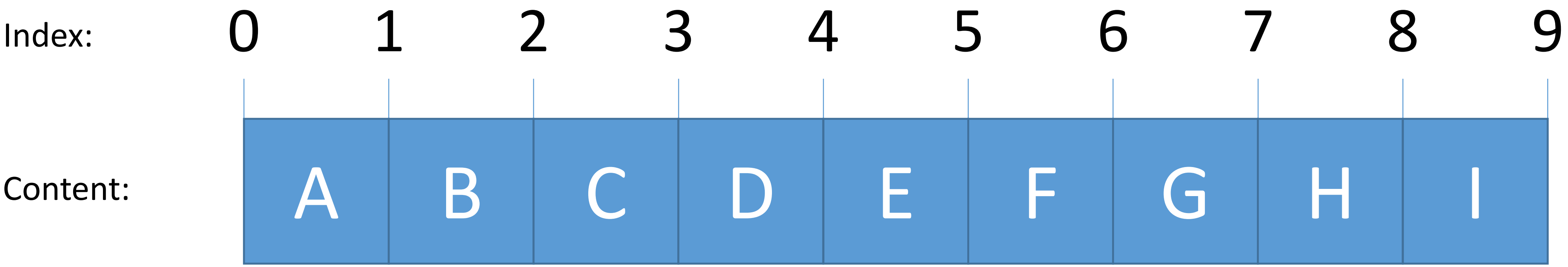
`data[start:stop:step]`

Indexing, cropping, subsets

- “Indexing” is addressing certain elements in lists. The first element is “0” away from the start.



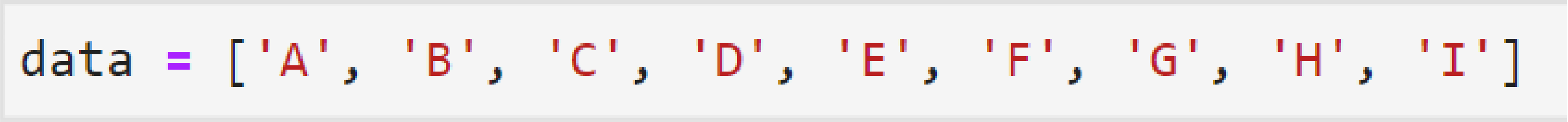
```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```



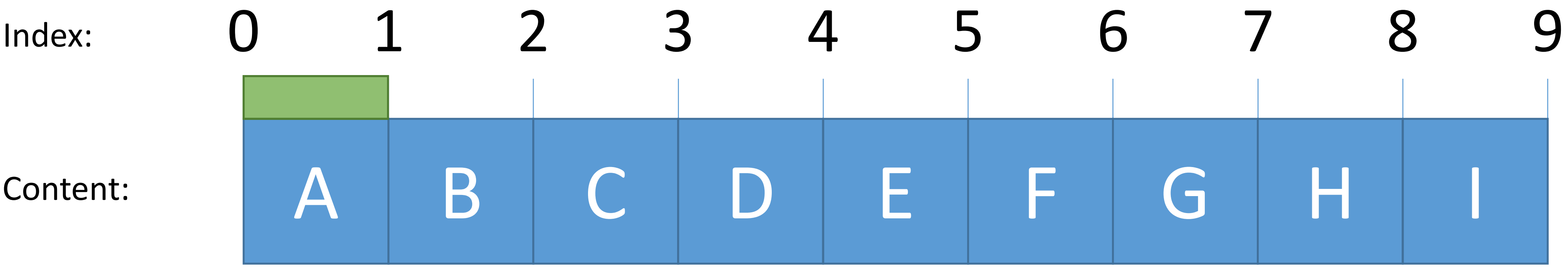
Index:	0	1	2	3	4	5	6	7	8	9
Content:	A	B	C	D	E	F	G	H	I	

Indexing, cropping, subsets

- “Indexing” is addressing certain elements in lists. The first element is “0” away from the start.

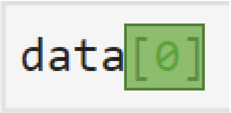


```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```



Index: 0 1 2 3 4 5 6 7 8 9

Content: A B C D E F G H I

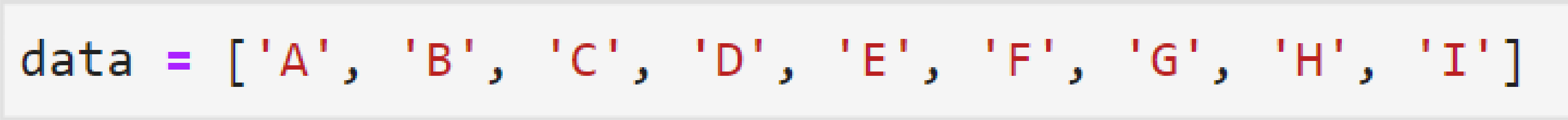


```
data[0]
```

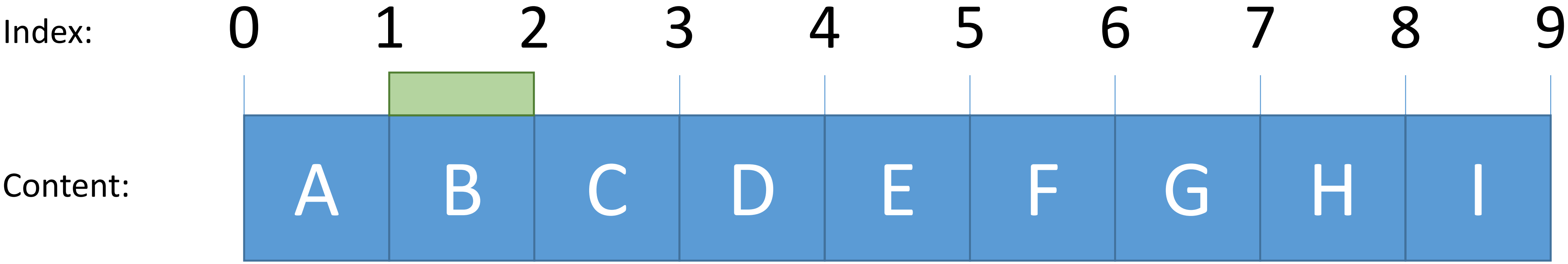
```
'A'
```

Indexing, cropping, subsets

- “Indexing” is addressing certain elements in lists. The first element is “0” away from the start.



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



```
data[0]
```

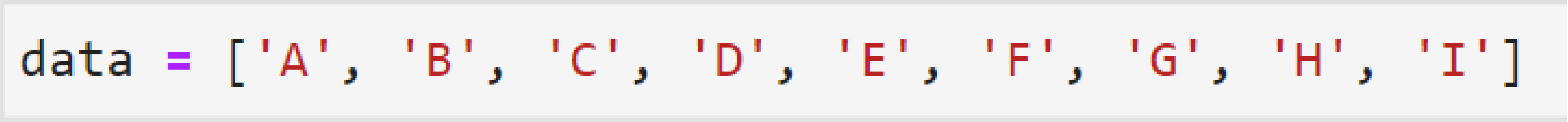
```
'A'
```

```
data[1]
```

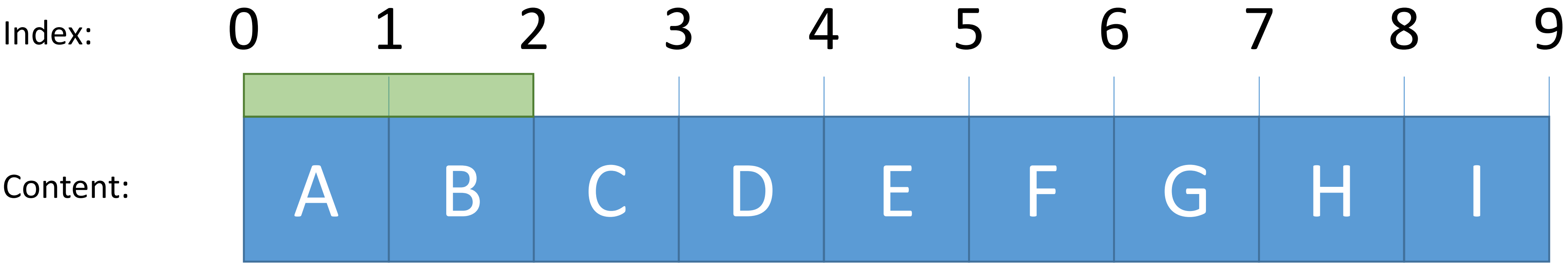
```
'B'
```

Indexing, cropping, subsets

- “Indexing” is addressing certain elements in lists. The first element is “0” away from the start.



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

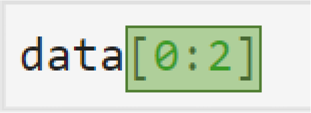


```
data[0]
```

```
'A'
```

```
data[1]
```

```
'B'
```

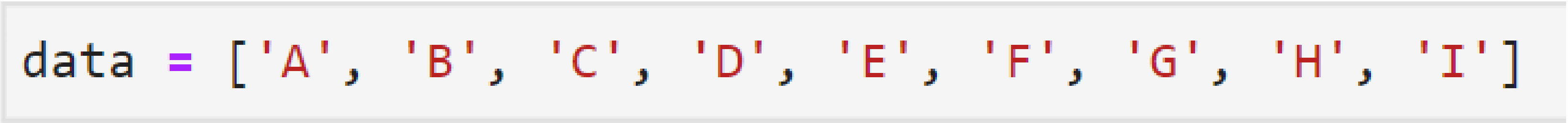


```
data[0:2]
```

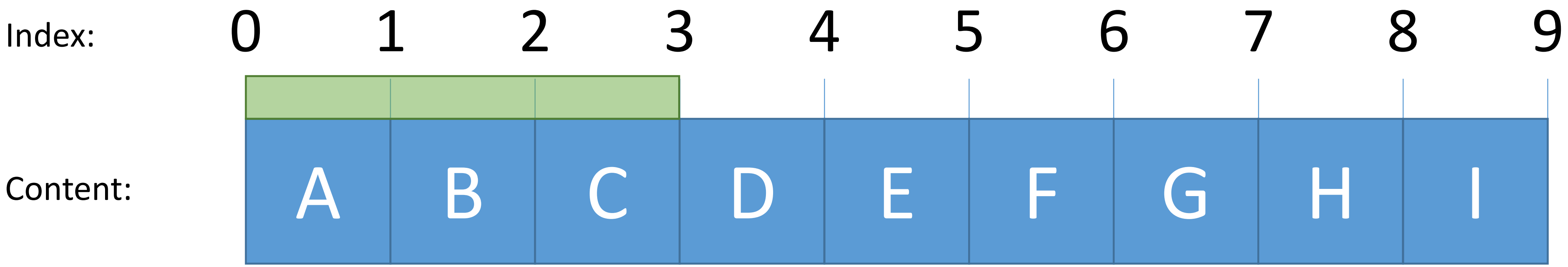
```
['A', 'B']
```

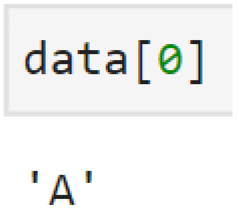
Indexing, cropping, subsets

- “Indexing” is addressing certain elements in lists. The first element is “0” away from the start.



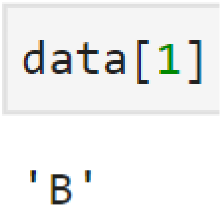
```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```





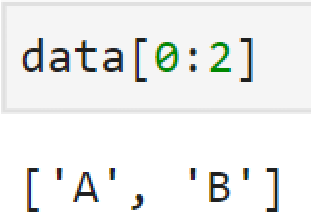
```
data[0]
```

```
'A'
```



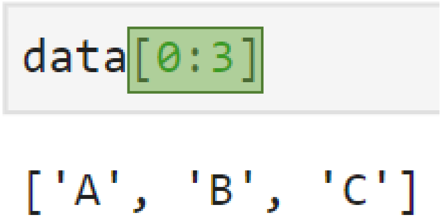
```
data[1]
```

```
'B'
```



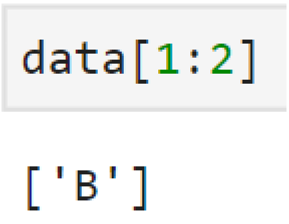
```
data[0:2]
```

```
['A', 'B']
```



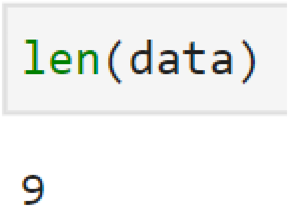
```
data[0:3]
```

```
['A', 'B', 'C']
```



```
data[1:2]
```

```
['B']
```

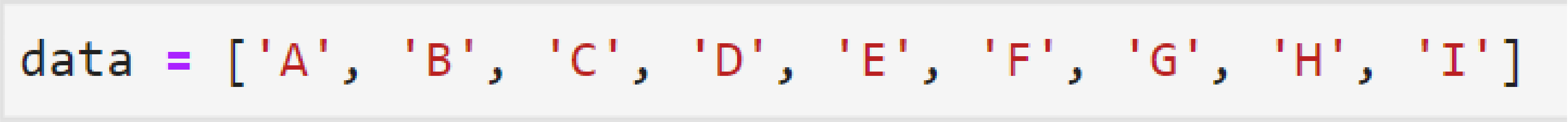


```
len(data)
```

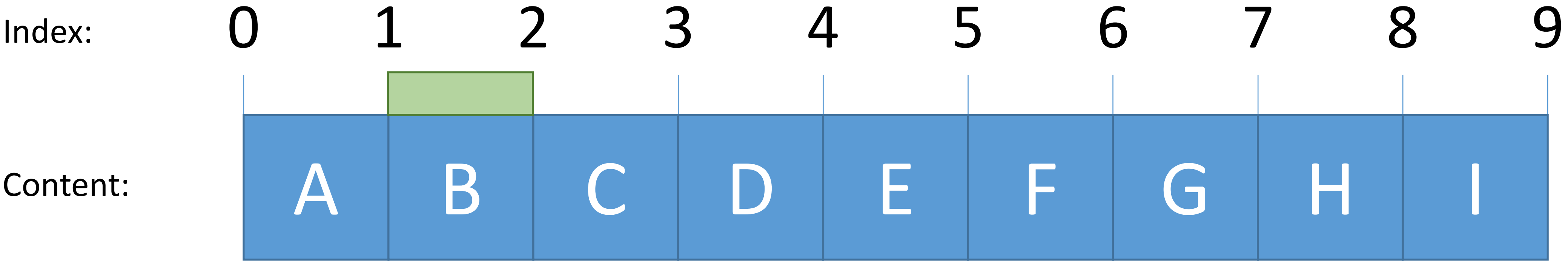
```
9
```

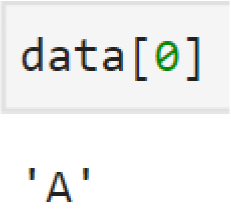
Indexing, cropping, subsets

- “Indexing” is addressing certain elements in lists. The first element is “0” away from the start.



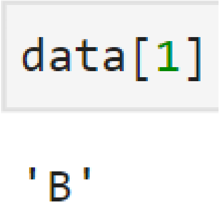
```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```





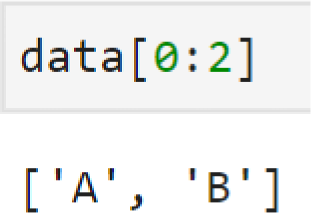
```
data[0]
```

```
'A'
```



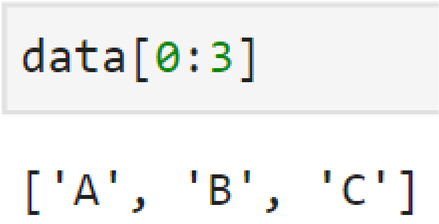
```
data[1]
```

```
'B'
```



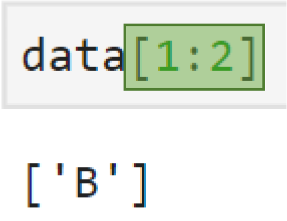
```
data[0:2]
```

```
['A', 'B']
```



```
data[0:3]
```

```
['A', 'B', 'C']
```

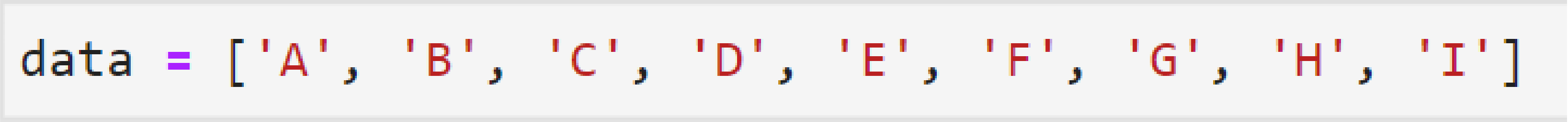


```
data[1:2]
```

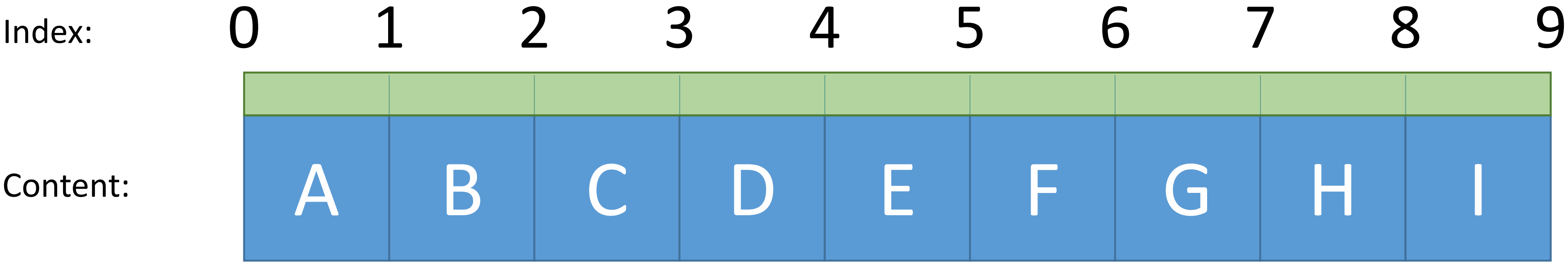
```
['B']
```

Indexing, cropping, subsets

- “Indexing” is addressing certain elements in lists. The first element is “0” away from the start.

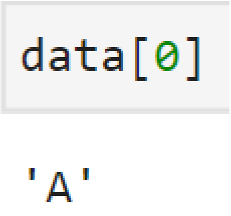


```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```



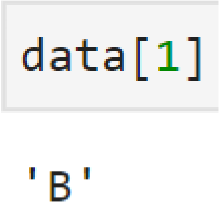
Index: 0 1 2 3 4 5 6 7 8 9

Content: A B C D E F G H I



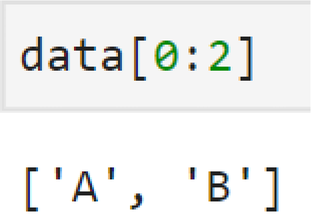
```
data[0]
```

```
'A'
```



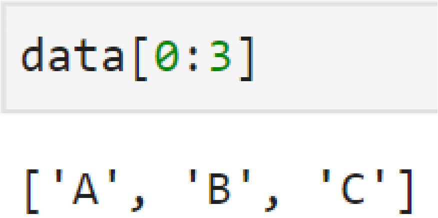
```
data[1]
```

```
'B'
```



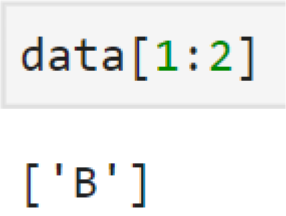
```
data[0:2]
```

```
['A', 'B']
```



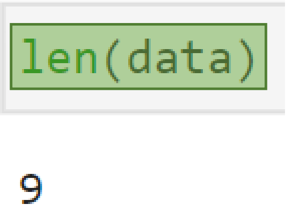
```
data[0:3]
```

```
['A', 'B', 'C']
```



```
data[1:2]
```

```
['B']
```

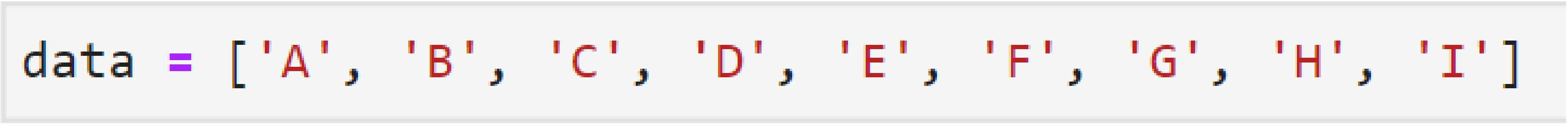


```
len(data)
```

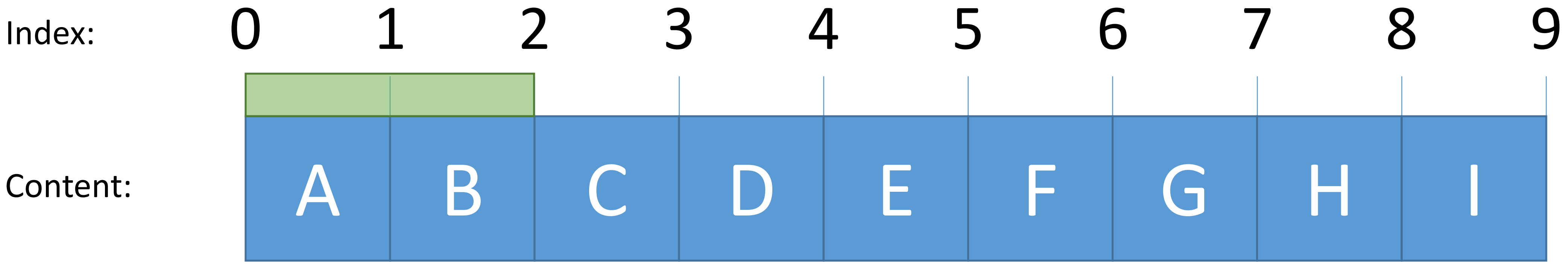
```
9
```

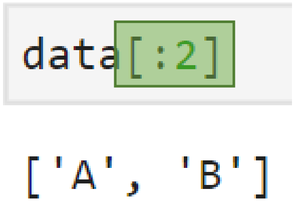
Indexing, cropping, subsets

- You can leave start and end out when specifying index ranges



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





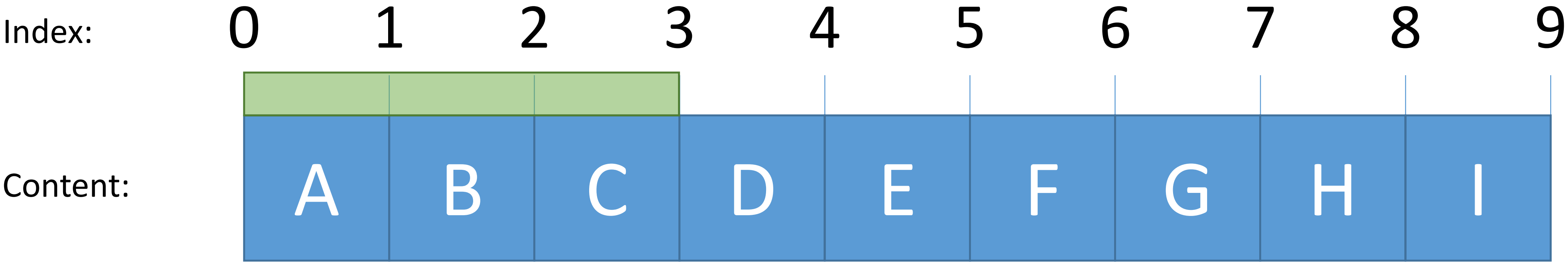
```
data[:2]
```

```
['A', 'B']
```

Indexing, cropping, subsets

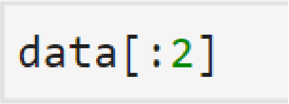
- You can leave start and end out when specifying index ranges

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



Index: 0 1 2 3 4 5 6 7 8 9

Content: A B C D E F G H I



```
data[:2]
```

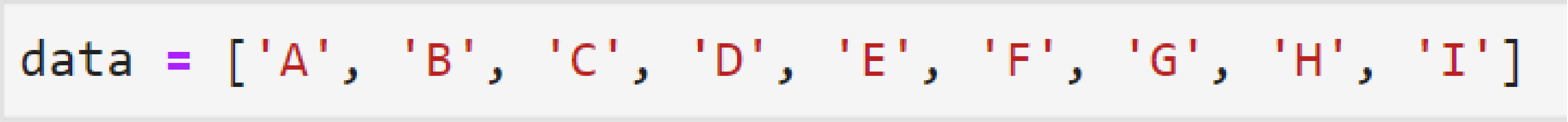
```
data[:3]
```

```
['A', 'B']
```

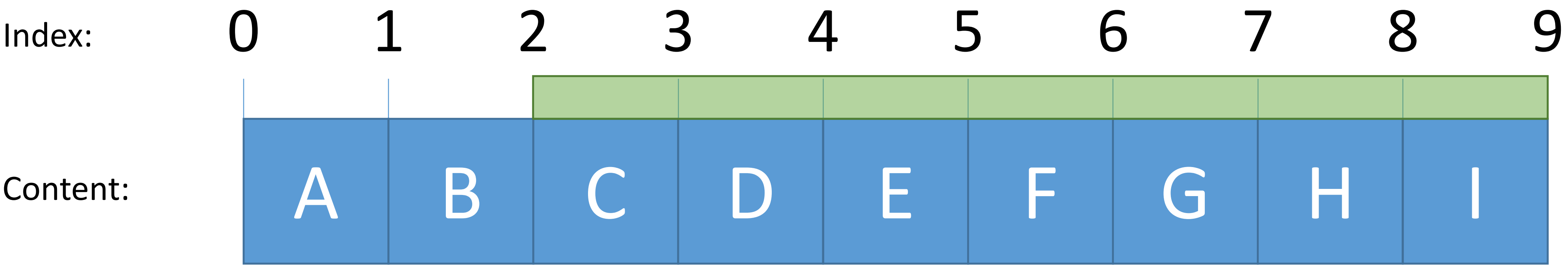
```
['A', 'B', 'C']
```

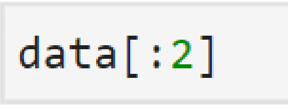
Indexing, cropping, subsets

- You can leave start and end out when specifying index ranges



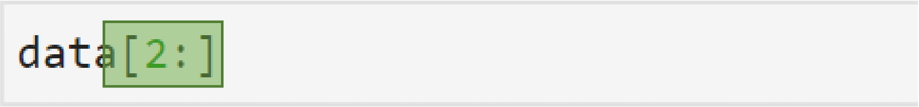
```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```





```
data[:2]
```

```
data[:3]
```



```
data[2:]
```

```
['A', 'B']
```

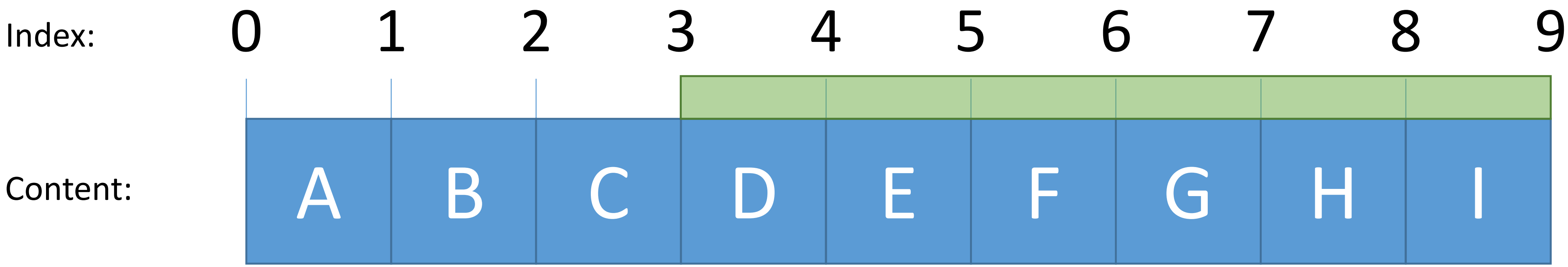
```
['A', 'B', 'C']
```

```
['C', 'D', 'E', 'F', 'G', 'H', 'I']
```

Indexing, cropping, subsets

- You can leave start and end out when specifying index ranges

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



```
data[:2]
```

```
['A', 'B']
```

```
data[:3]
```

```
['A', 'B', 'C']
```

```
data[2:]
```

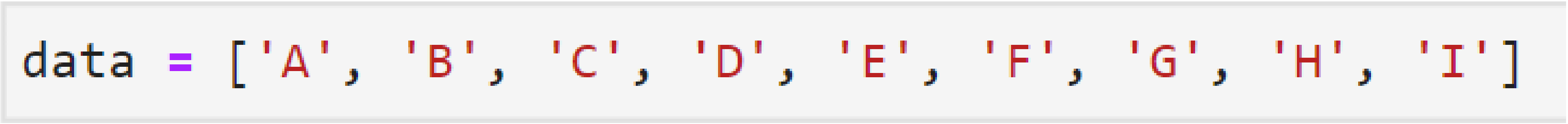
```
['C', 'D', 'E', 'F', 'G', 'H', 'I']
```

```
data[3:]
```

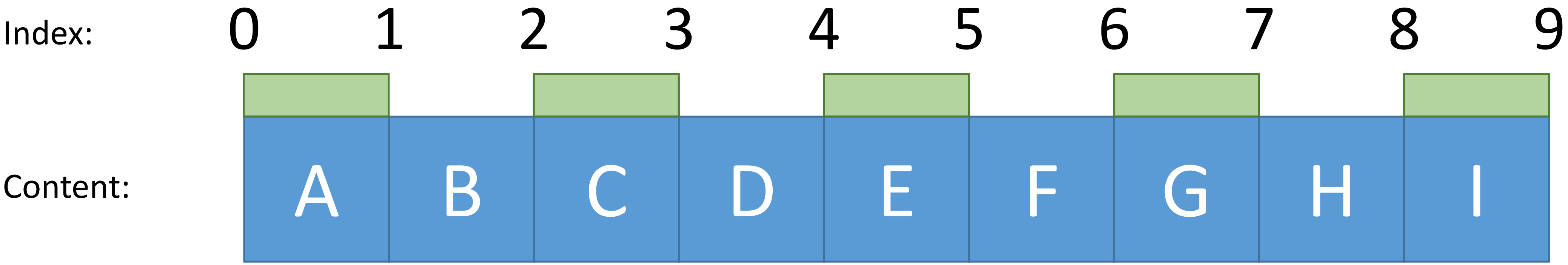
```
['D', 'E', 'F', 'G', 'H', 'I']
```

Indexing, cropping, subsets

- The step-size allows skipping elements

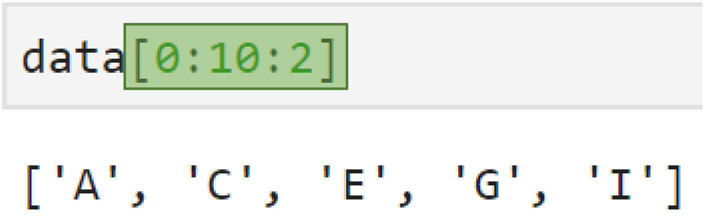


```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```



Index: 0 1 2 3 4 5 6 7 8 9

Content: A B C D E F G H I

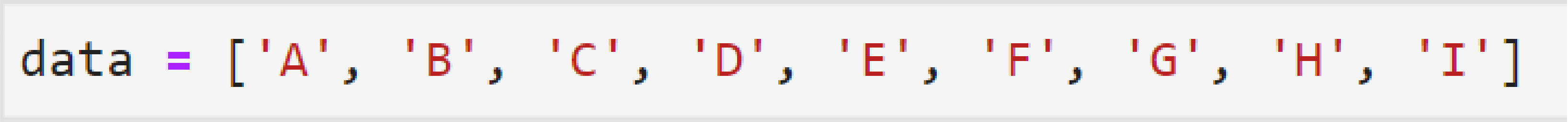


```
data[0:10:2]
```

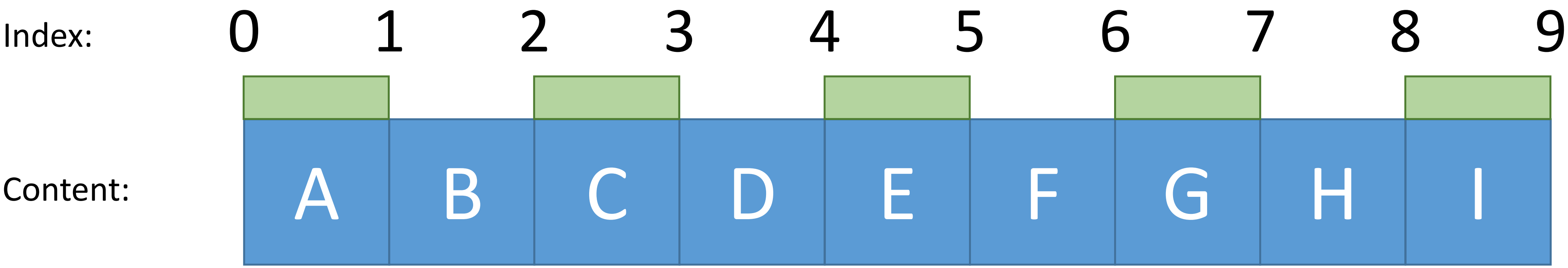
```
['A', 'C', 'E', 'G', 'I']
```

Indexing, cropping, subsets

- The step-size allows skipping elements

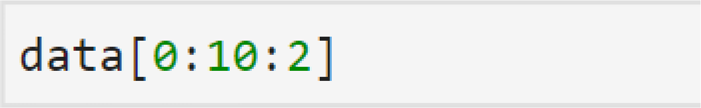


```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```



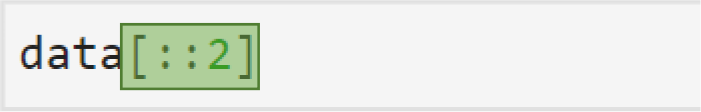
Index: 0 1 2 3 4 5 6 7 8 9

Content: A B C D E F G H I



```
data[0:10:2]
```

```
['A', 'C', 'E', 'G', 'I']
```

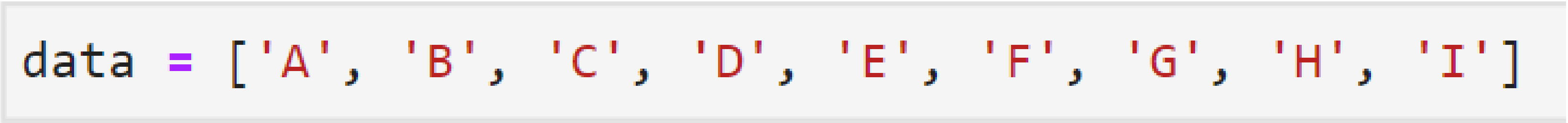


```
data[::2]
```

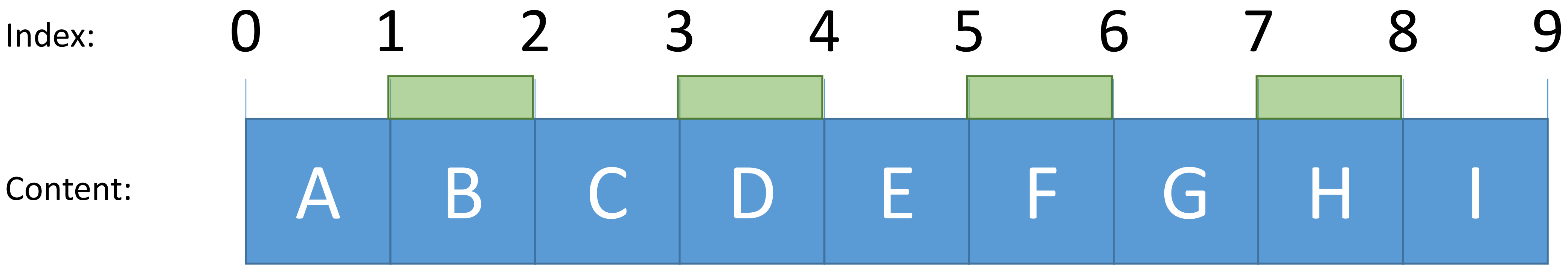
```
['A', 'C', 'E', 'G', 'I']
```

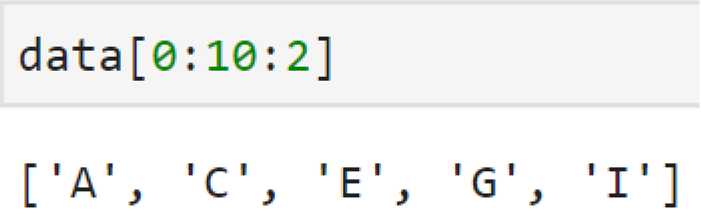
Indexing, cropping, subsets

- The step-size allows skipping elements



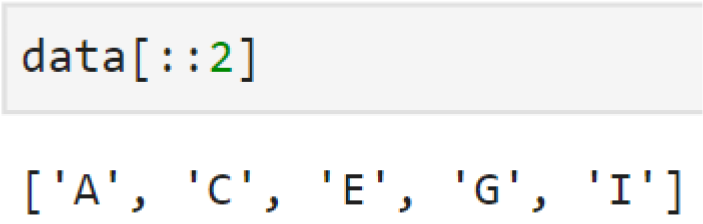
```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```





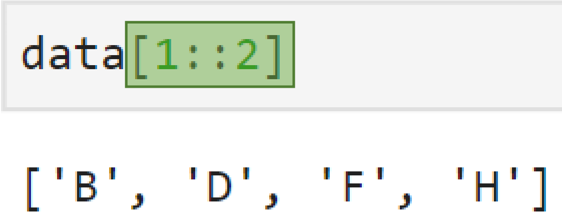
```
data[0:10:2]
```

```
['A', 'C', 'E', 'G', 'I']
```



```
data[::2]
```

```
['A', 'C', 'E', 'G', 'I']
```

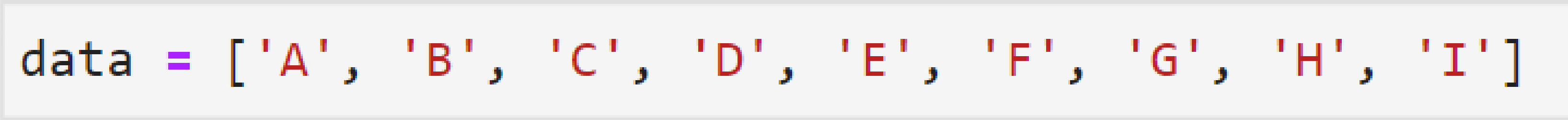


```
data[1::2]
```

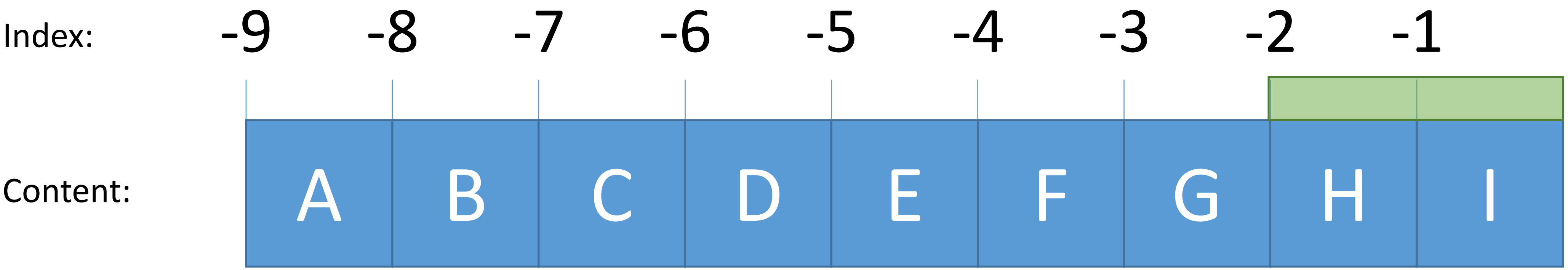
```
['B', 'D', 'F', 'H']
```

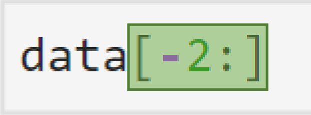
Indexing, cropping, subsets

- Indexing also works with negative indices



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





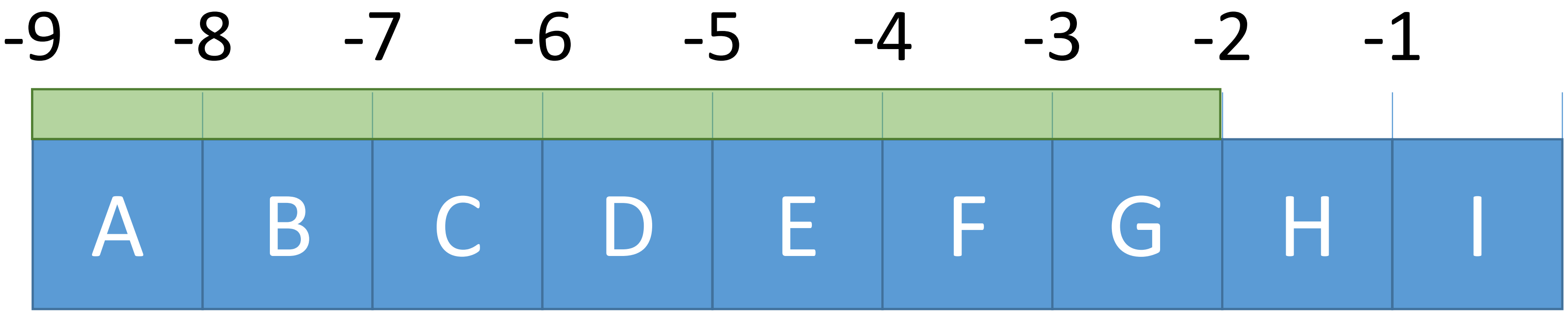
```
data[-2:]
```

```
['H', 'I']
```

Indexing, cropping, subsets

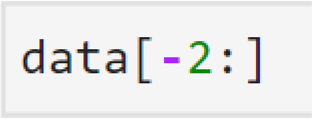
- Indexing also works with negative indices

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

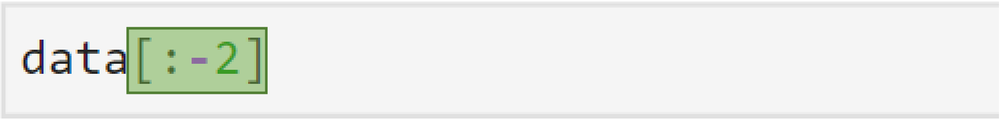


Index: -9 -8 -7 -6 -5 -4 -3 -2 -1

Content: A B C D E F G H I



```python
data[-2:]
```



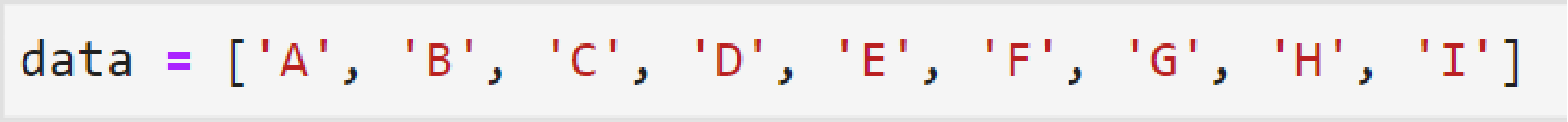
```python
data[:-2]
```

```
['H', 'I']
```

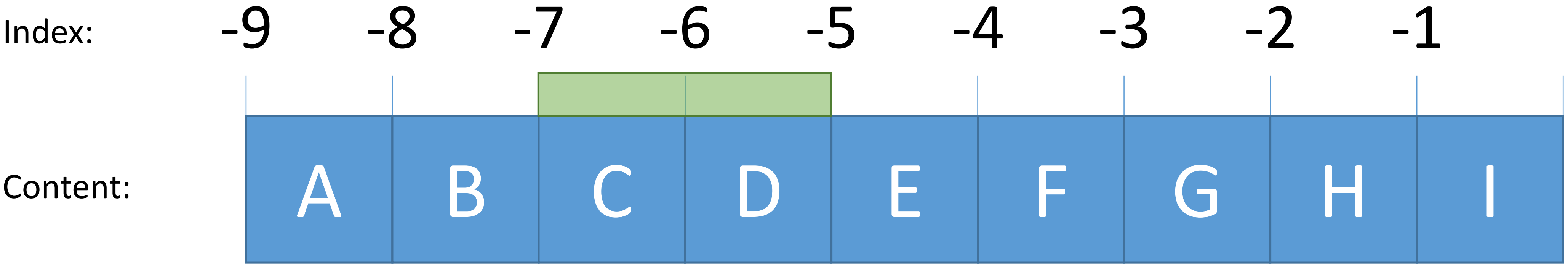
```
['A', 'B', 'C', 'D', 'E', 'F', 'G']
```

Indexing, cropping, subsets

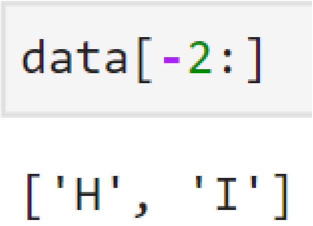
- Indexing also works with negative indices



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

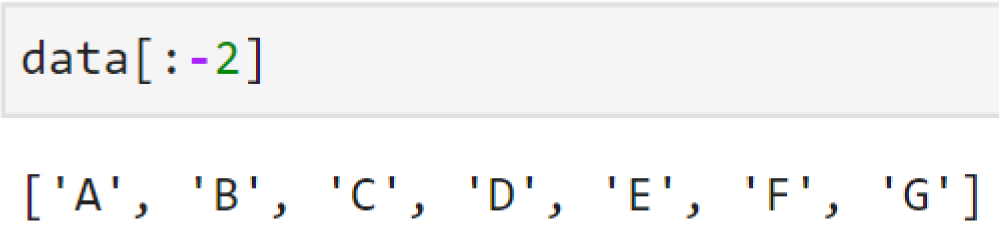


Index:	-9	-8	-7	-6	-5	-4	-3	-2	-1
Content:	A	B	C	D	E	F	G	H	I



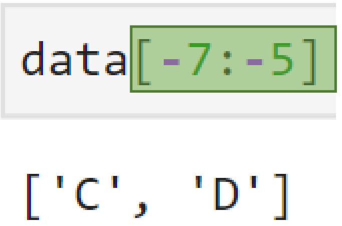
```python
data[-2:]
```

```
['H', 'I']
```



```python
data[:-2]
```

```
['A', 'B', 'C', 'D', 'E', 'F', 'G']
```

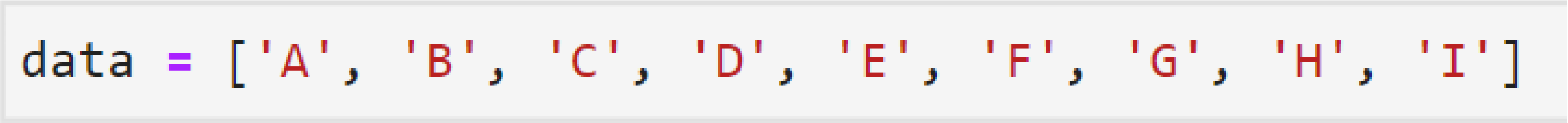


```python
data[-7:-5]
```

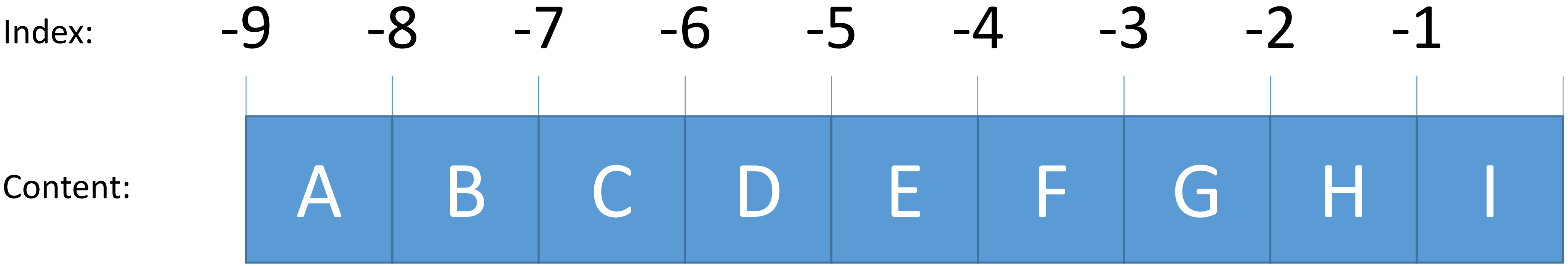
```
['C', 'D']
```

Indexing, cropping, subsets

- Indexing also works with negative indices

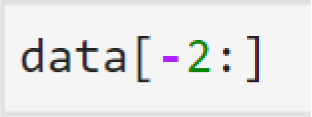


```python
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```



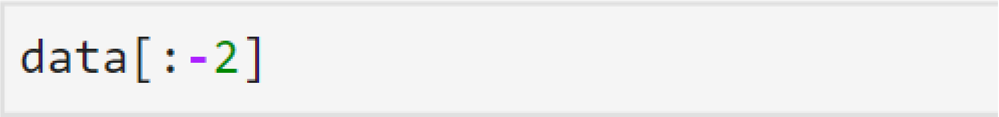
Index: -9 -8 -7 -6 -5 -4 -3 -2 -1

Content: A B C D E F G H I



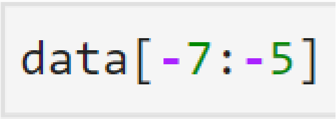
```python
data[-2:]
```

```
['H', 'I']
```



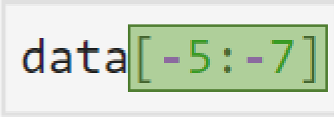
```python
data[:-2]
```

```
['A', 'B', 'C', 'D', 'E', 'F', 'G']
```



```python
data[-7:-5]
```

```
['C', 'D']
```

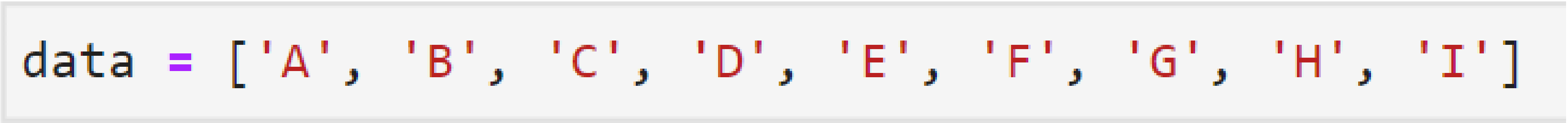


```python
data[-5:-7]
```

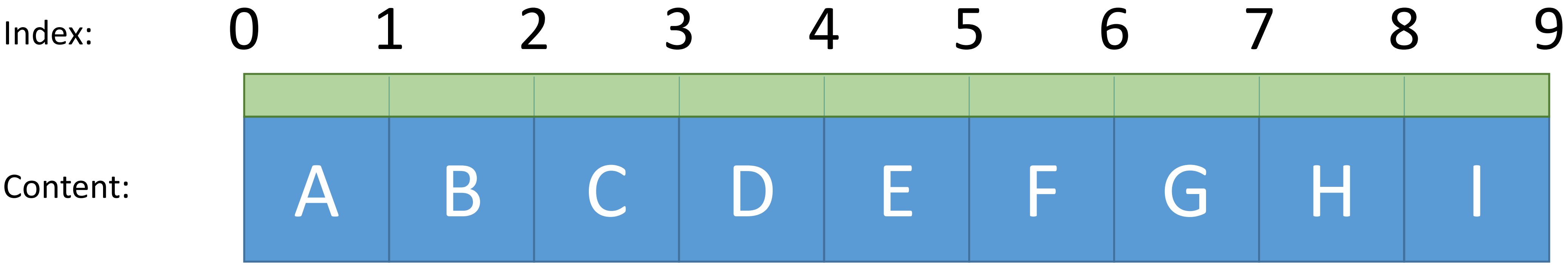
```
[]
```

Indexing, cropping, subsets

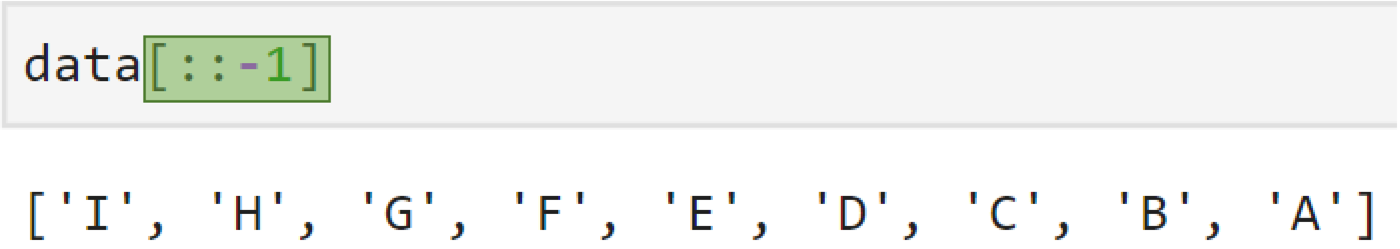
- Negative stepping also works



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



Index:	0	1	2	3	4	5	6	7	8	9
Content:	A	B	C	D	E	F	G	H	I	

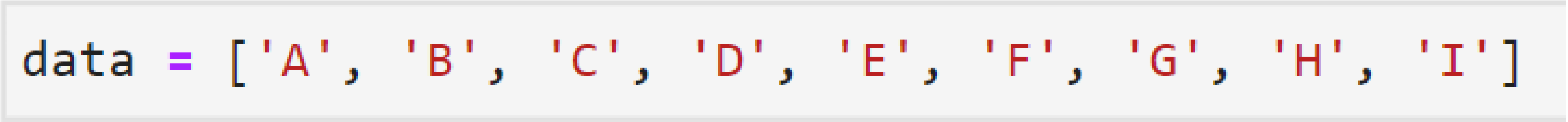


```
data[::-1]
```

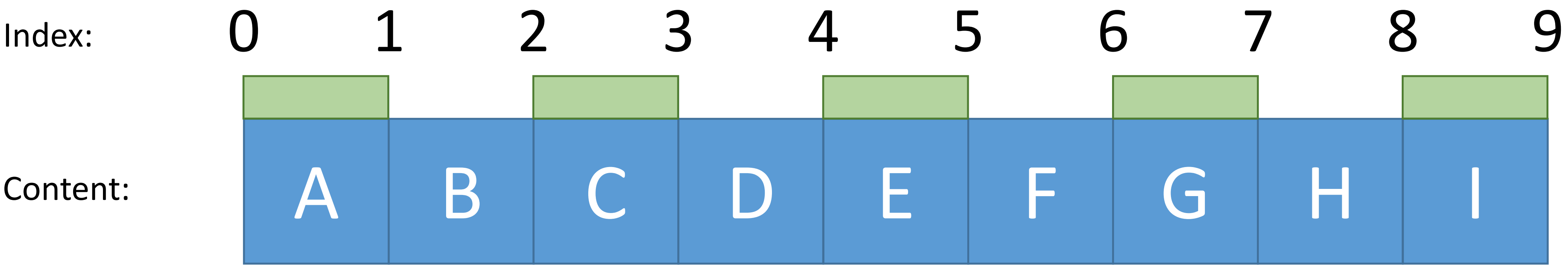
```
['I', 'H', 'G', 'F', 'E', 'D', 'C', 'B', 'A']
```

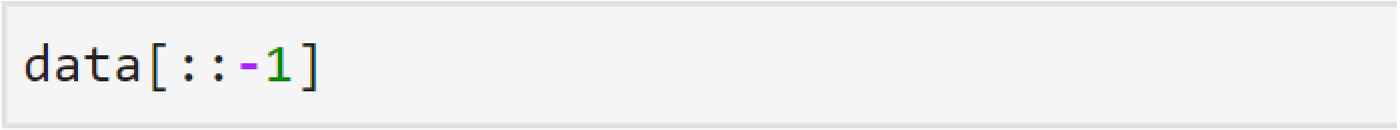
Indexing, cropping, subsets

- Negative stepping also works



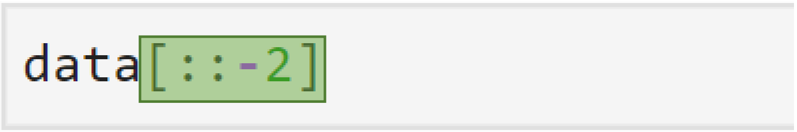
```
data = ['A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I']
```





```
data[::-1]
```

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

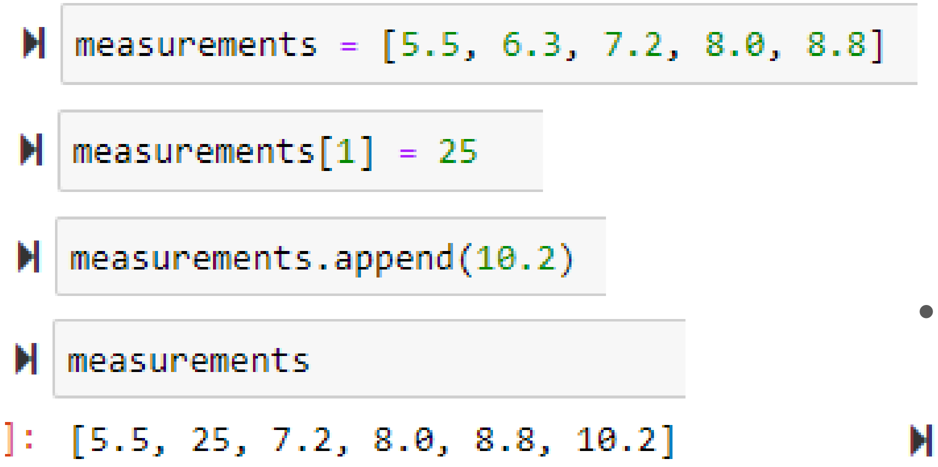


```
data[::-2]
```

```
['I', 'G', 'E', 'C', 'A']
```

Lists versus Tuples

- Lists can be modified



```
measurements = [5.5, 6.3, 7.2, 8.0, 8.8]
```

```
measurements[1] = 25
```

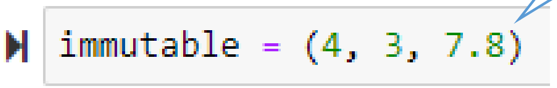
```
measurements.append(10.2)
```

```
measurements
```

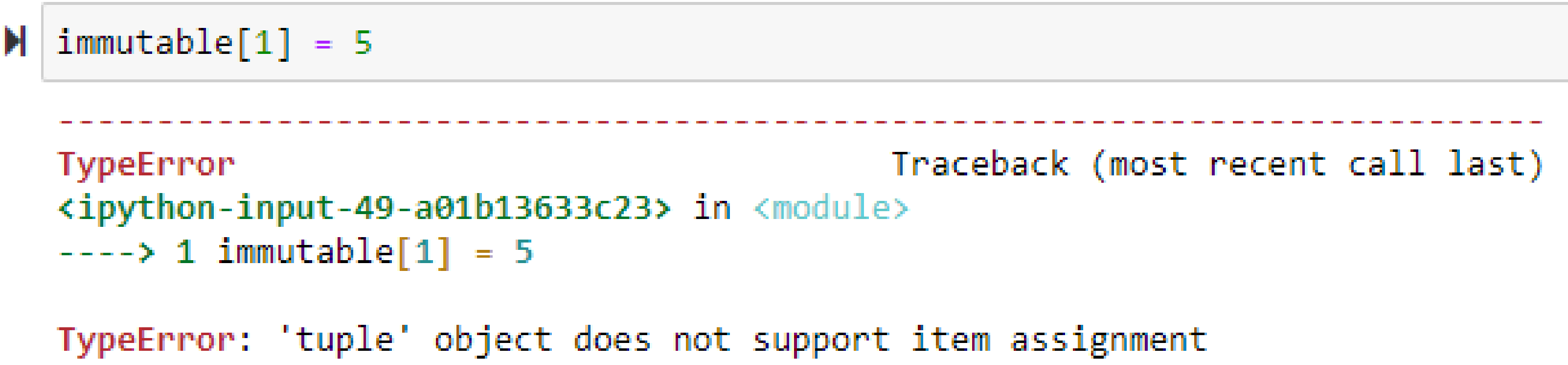
```
]: [5.5, 25, 7.2, 8.0, 8.8, 10.2]
```

- Tuples not

Note: round brackets



```
immutable = (4, 3, 7.8)
```



```
immutable[1] = 5
```

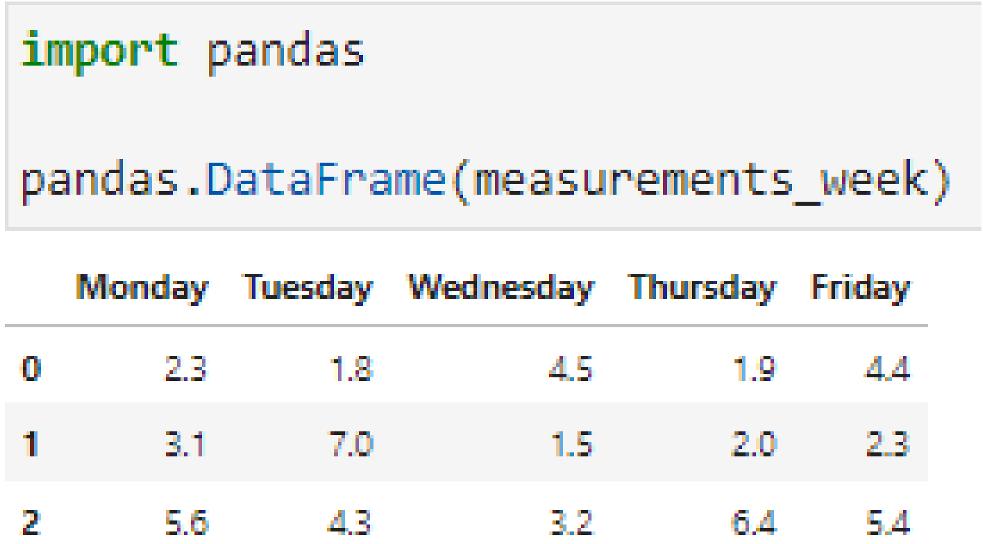
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-49-a01b13633c23> in <module>
----> 1 immutable[1] = 5

TypeError: 'tuple' object does not support item assignment
```

Tables

- Sneak preview: By the mid of the course, we will work with Pandas DataFrames, *fancy* Tables.

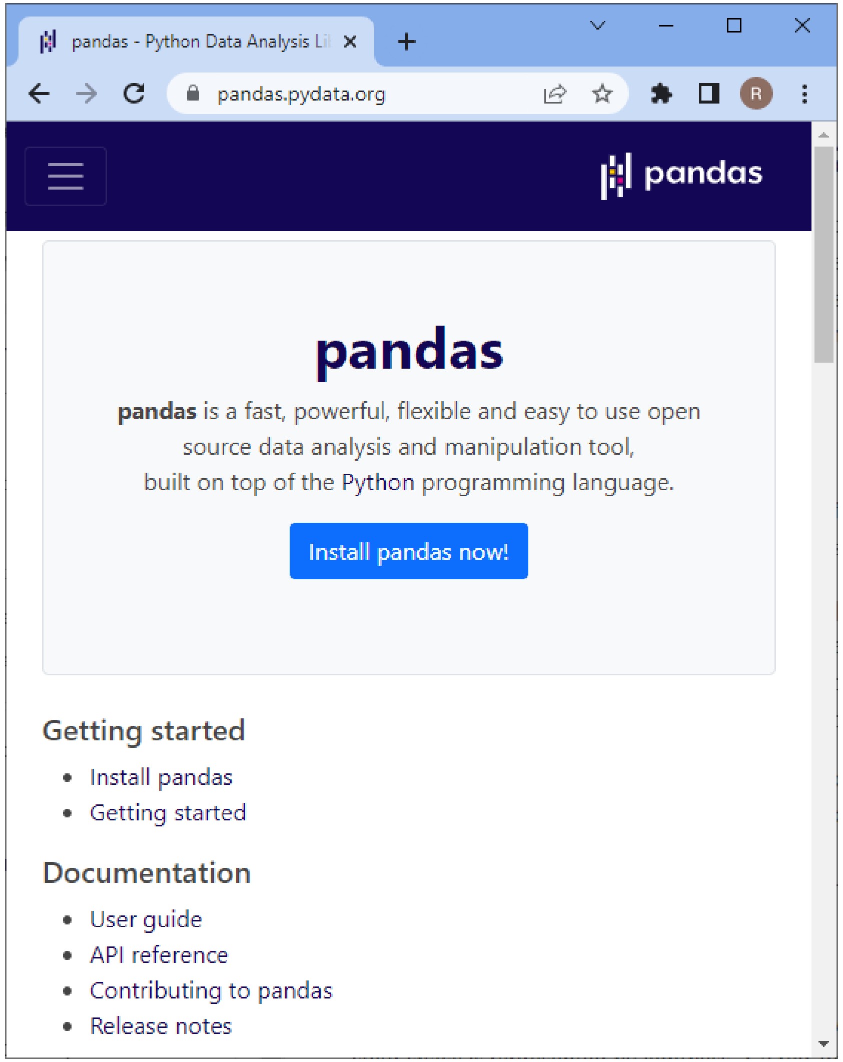
- `conda install pandas`

- Among many other features, Pandas allows to visualize tables nicely in Jupyter notebooks.



```python
import pandas

pandas.DataFrame(measurements_week)
```

	Monday	Tuesday	Wednesday	Thursday	Friday
0	2.3	1.8	4.5	1.9	4.4
1	3.1	7.0	1.5	2.0	2.3
2	5.6	4.3	3.2	6.4	5.4



pandas

pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

Getting started

- Install pandas
- Getting started

Documentation

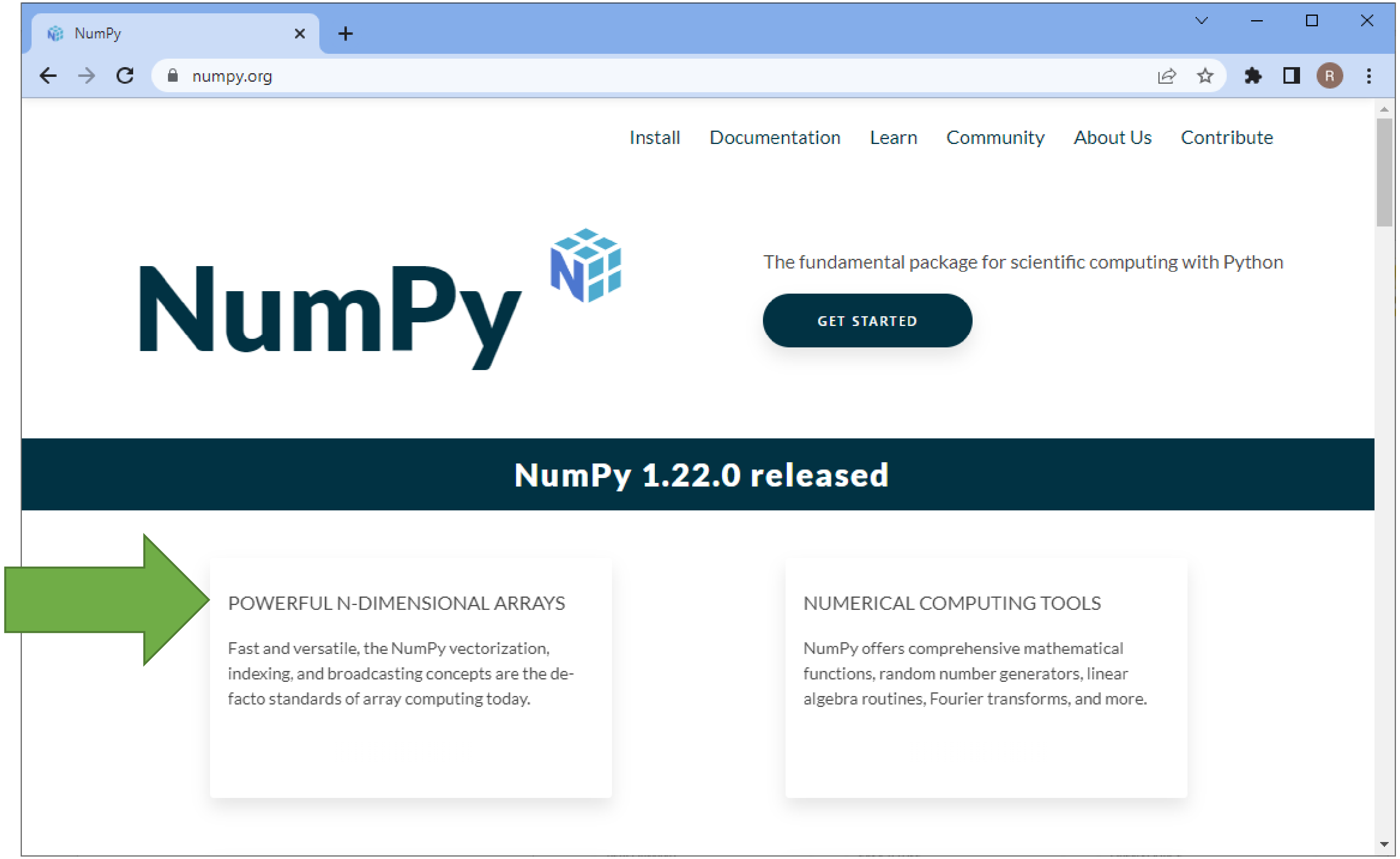
- User guide
- API reference
- Contributing to pandas
- Release notes

https://pandas.pydata.org/

numpy

- The fundamental package for scientific computing with python.

- `conda install numpy`



NumPy

numpy.org

Install Documentation Learn Community About Us Contribute

NumPy

The fundamental package for scientific computing with Python

GET STARTED

NumPy 1.22.0 released

POWERFUL N-DIMENSIONAL ARRAYS

Fast and versatile, the NumPy vectorization, indexing, and broadcasting concepts are the de-facto standards of array computing today.

NUMERICAL COMPUTING TOOLS

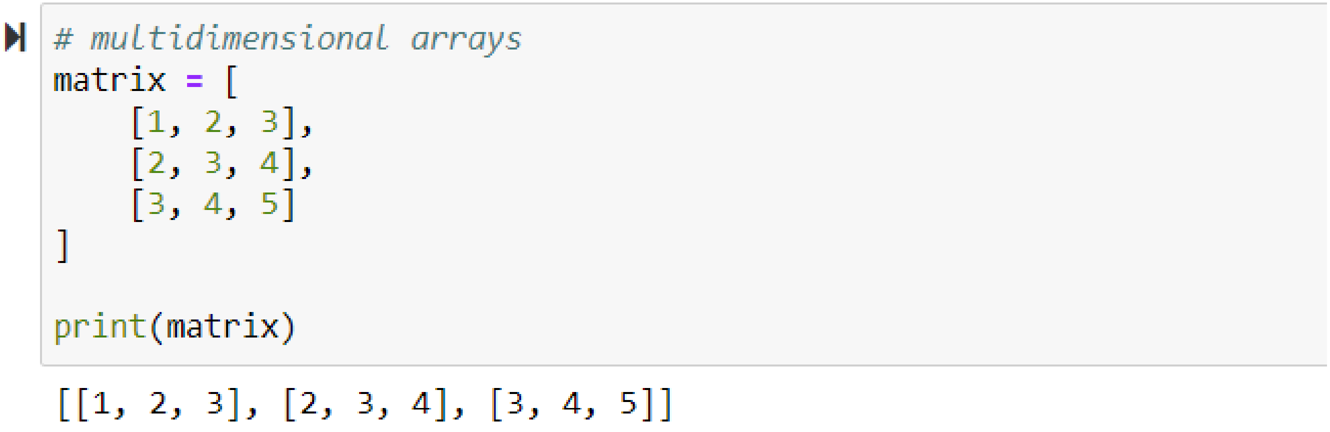
NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.

https://numpy.org/

numpy

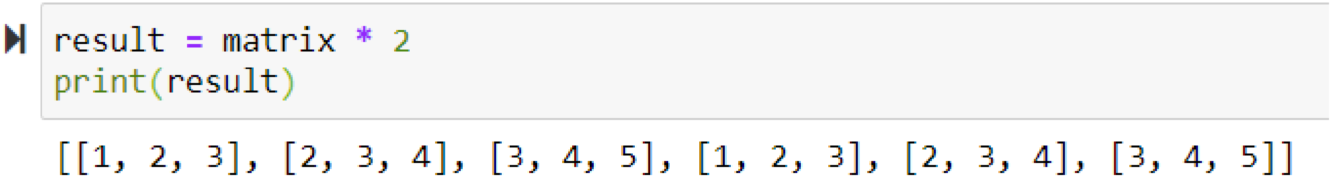
- Simplifying mathematical operations on n-dimensional arrays

- Python arrays of arrays (lists of lists)



```
# multidimensional arrays
matrix = [
    [1, 2, 3],
    [2, 3, 4],
    [3, 4, 5]
]

print(matrix)
```

```
[[1, 2, 3], [2, 3, 4], [3, 4, 5]]
```

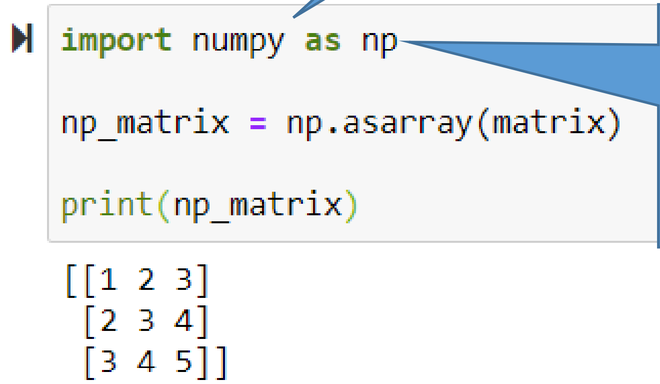


```
result = matrix * 2
print(result)
```

```
[[1, 2, 3], [2, 3, 4], [3, 4, 5], [1, 2, 3], [2, 3, 4], [3, 4, 5]]
```

- numpy arrays

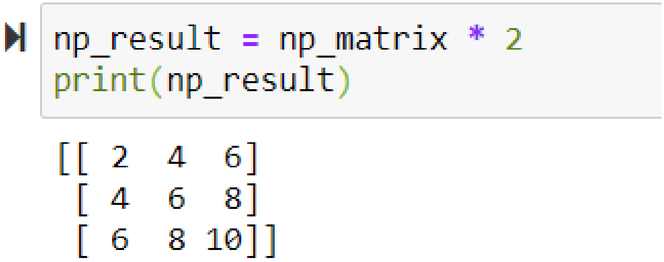
Tell python that you want to use a library called numpy



```
import numpy as np

np_matrix = np.asarray(matrix)

print(np_matrix)
```

```
[[1 2 3]
 [2 3 4]
 [3 4 5]]
```

If “numpy” is to long, you can give an alias “np”

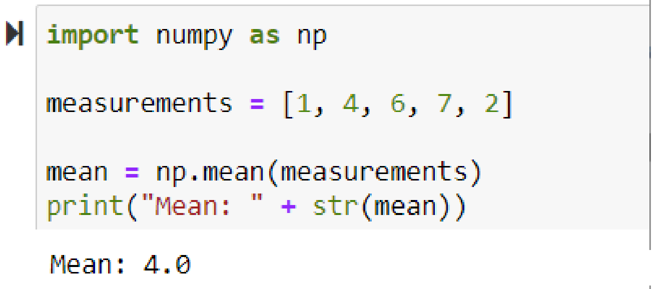


```
np_result = np_matrix * 2
print(np_result)
```

```
[[ 2  4  6]
 [ 4  6  8]
 [ 6  8 10]]
```

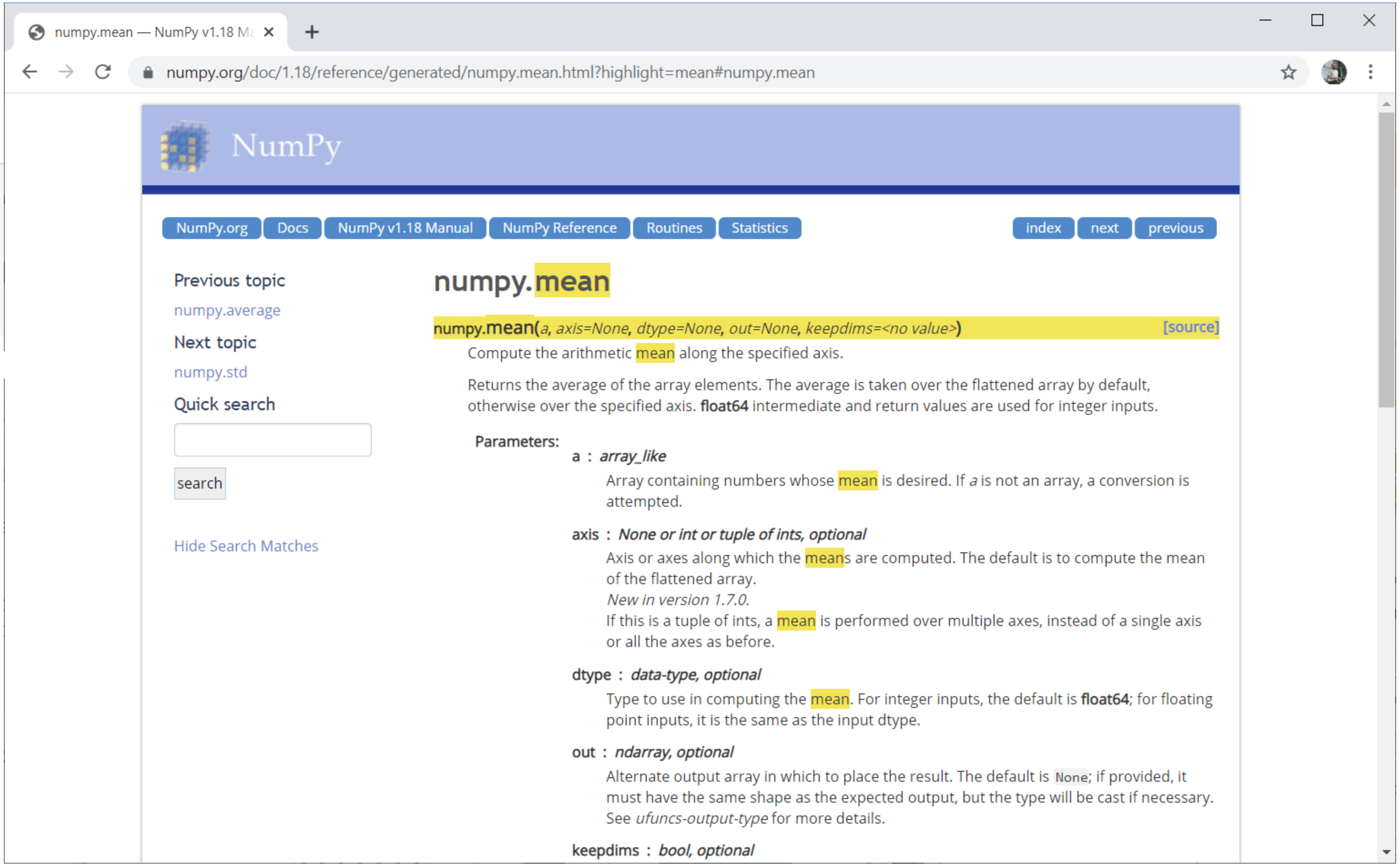
Basic descriptive statistics using numpy

- Basic descriptive statistics



```
import numpy as np

measurements = [1, 4, 6, 7, 2]

mean = np.mean(measurements)
print("Mean: " + str(mean))
```

```
Mean: 4.0
```



numpy.mean

numpy.mean(*a*, *axis=None*, *dtype=None*, *out=None*, *keepdims=<no value>*) [source]

Compute the arithmetic mean along the specified axis.

Returns the average of the array elements. The average is taken over the flattened array by default, otherwise over the specified axis. **float64** intermediate and return values are used for integer inputs.

Parameters:

a : *array_like*

Array containing numbers whose mean is desired. If *a* is not an array, a conversion is attempted.

axis : *None or int or tuple of ints, optional*

Axis or axes along which the means are computed. The default is to compute the mean of the flattened array.

New in version 1.7.0.

If this is a tuple of ints, a mean is performed over multiple axes, instead of a single axis or all the axes as before.

dtype : *data-type, optional*

Type to use in computing the mean. For integer inputs, the default is **float64**; for floating point inputs, it is the same as the input dtype.

out : *ndarray, optional*

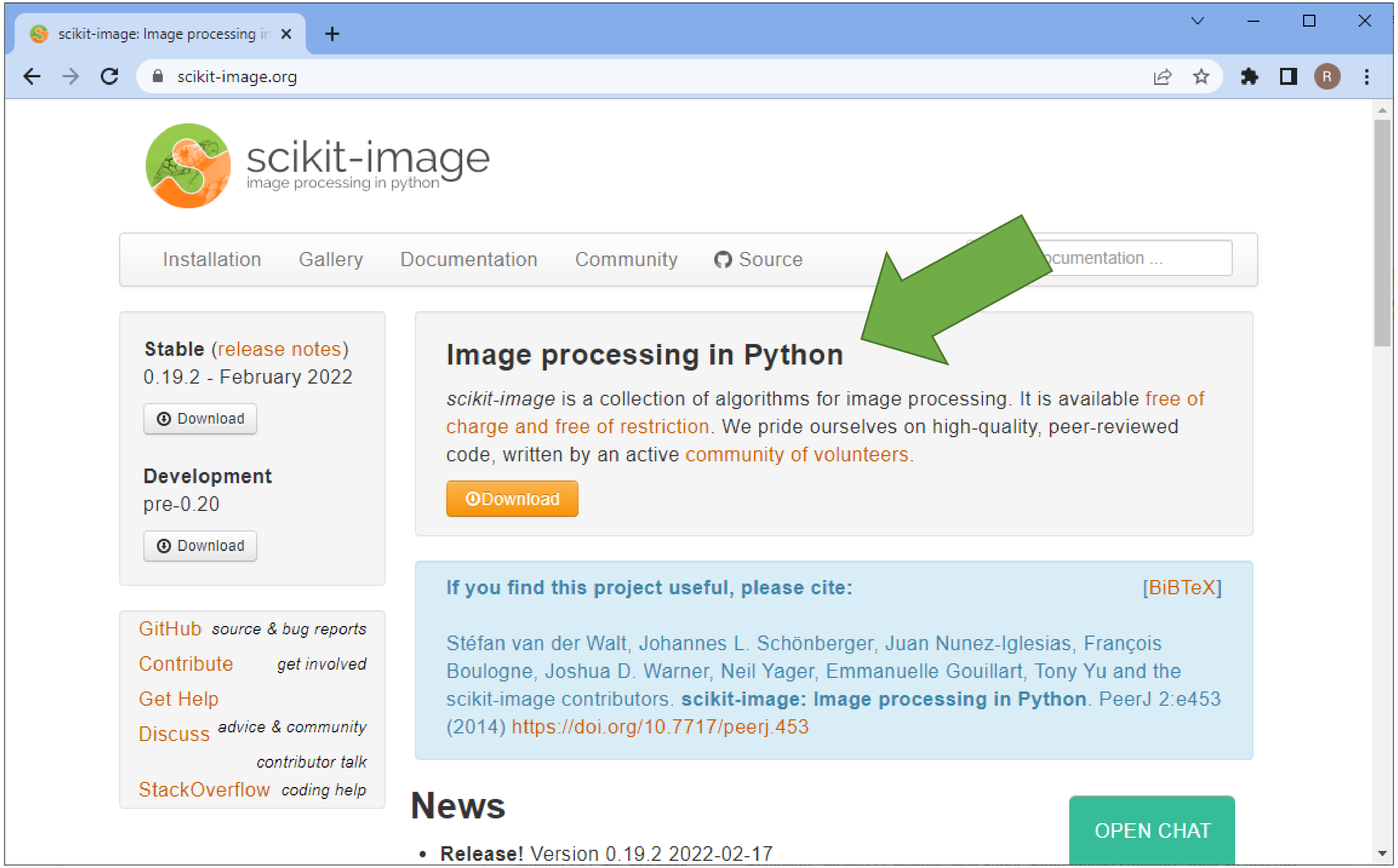
Alternate output array in which to place the result. The default is `None`; if provided, it must have the same shape as the expected output, but the type will be cast if necessary. See *ufuncs-output-type* for more details.

keepdims : *bool, optional*

scikit-image

- *scikit-image* is a collection of algorithms for image processing.

```
conda install scikit-image
```



Stable (release notes)
0.19.2 - February 2022

Development
pre-0.20

Image processing in Python

scikit-image is a collection of algorithms for image processing. It is available free of charge and free of restriction. We pride ourselves on high-quality, peer-reviewed code, written by an active community of volunteers.

Download

If you find this project useful, please cite: [BiBTeX]

Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu and the scikit-image contributors. **scikit-image: Image processing in Python**. PeerJ 2:e453 (2014) https://doi.org/10.7717/peerj.453

News

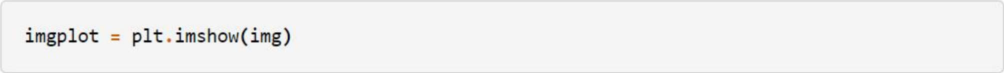
- **Release!** Version 0.19.2 2022-02-17

https://scikit-image.org/

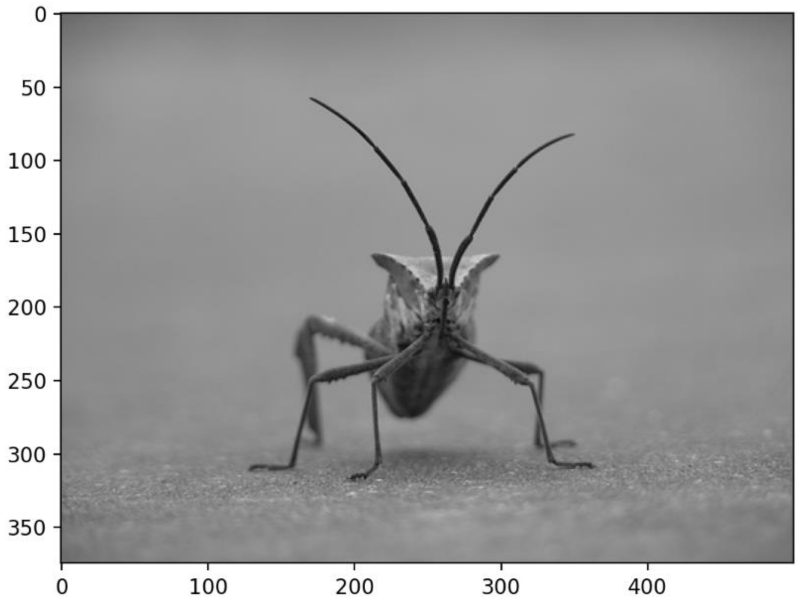
matplotlib

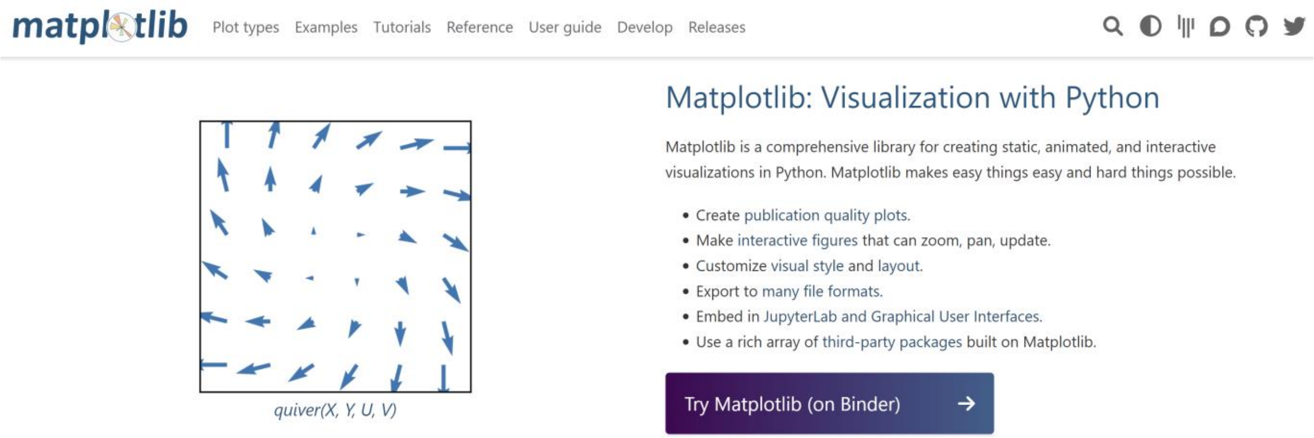
- *matplotlib* is the standard python library for plotting data.

```
conda install matplotlib
```



```
imgplot = plt.imshow(img)
```





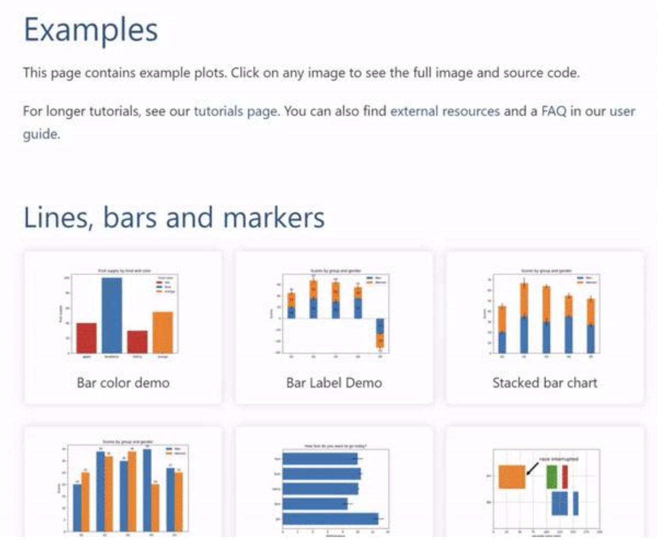
quiver(X, Y, U, V)

Matplotlib: Visualization with Python

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

- Create publication quality plots.
- Make interactive figures that can zoom, pan, update.
- Customize visual style and layout.
- Export to many file formats.
- Embed in JupyterLab and Graphical User Interfaces.
- Use a rich array of third-party packages built on Matplotlib.

Try Matplotlib (on Binder) →



Examples

This page contains example plots. Click on any image to see the full image and source code.

For longer tutorials, see our tutorials page. You can also find external resources and a FAQ in our user guide.

Lines, bars and markers

Bar color demo

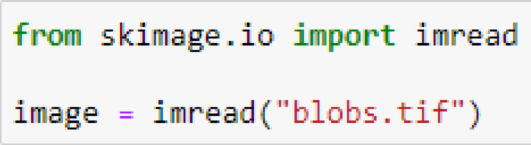
Bar Label Demo

Stacked bar chart

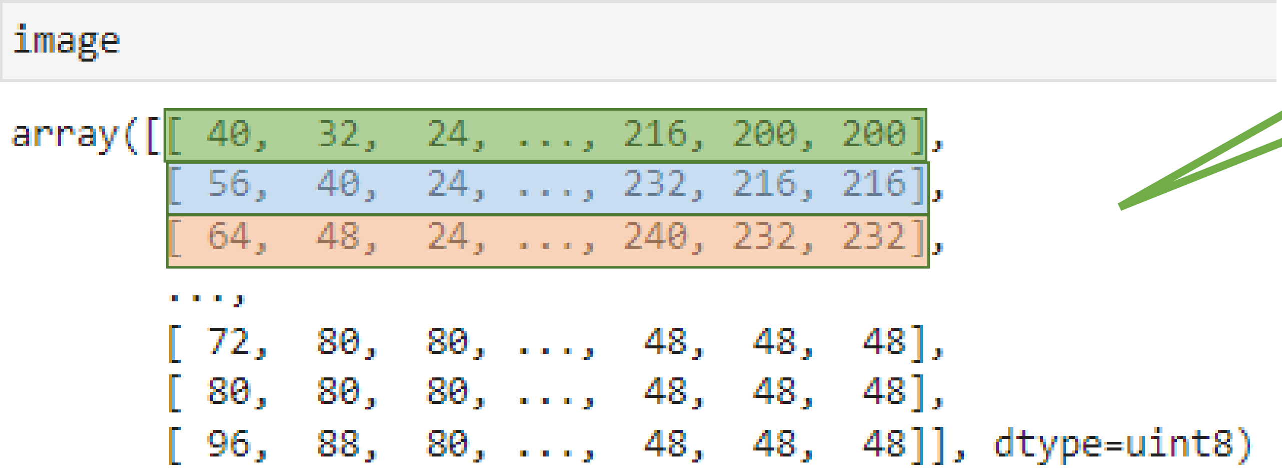
https://matplotlib.org/

Working with images in python

- Open images



```python
from skimage.io import imread

image = imread("blobs.tif")
```



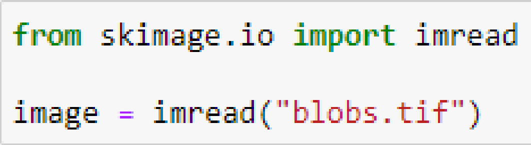
```
image
```

```
array([[ 40,  32,  24, ..., 216, 200, 200],
       [ 56,  40,  24, ..., 232, 216, 216],
       [ 64,  48,  24, ..., 240, 232, 232],
       ...,
       [ 72,  80,  80, ...,  48,  48,  48],
       [ 80,  80,  80, ...,  48,  48,  48],
       [ 96,  88,  80, ...,  48,  48,  48]], dtype=uint8)
```

Images are *just* multi-dimensional arrays or “arrays of arrays”.

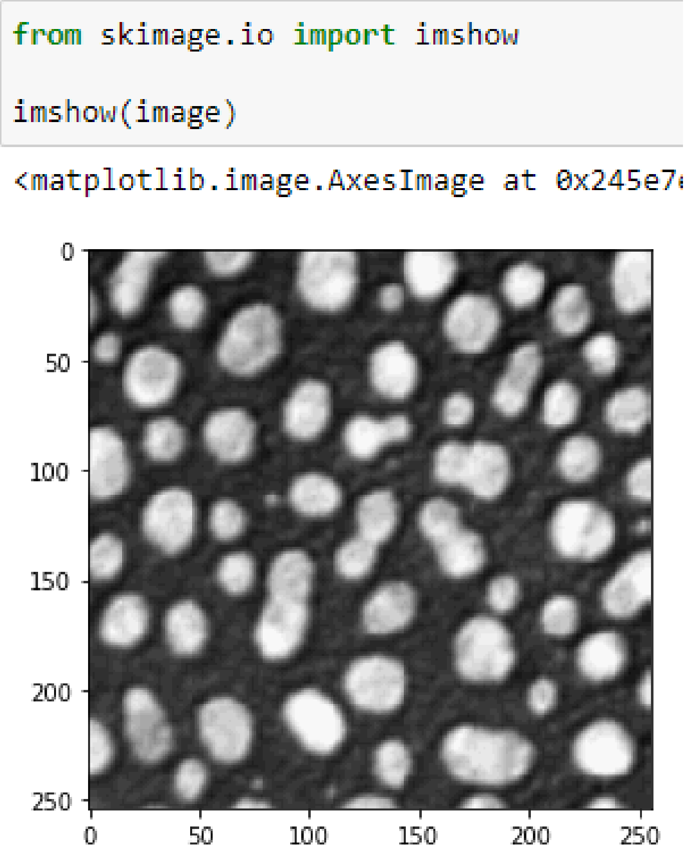
Working with images in python

- Open images



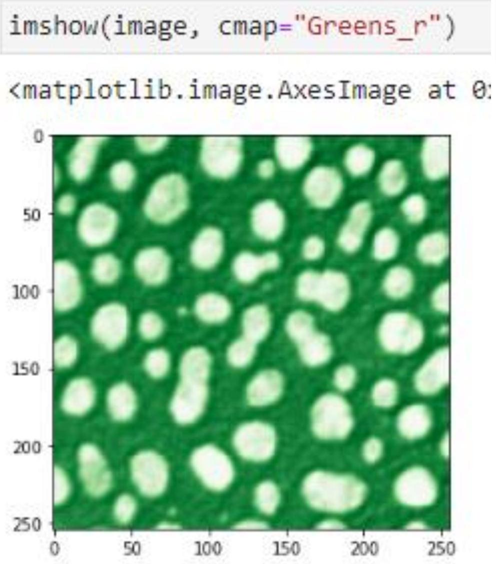
```
from skimage.io import imread

image = imread("blobs.tif")
```

- Visualize images



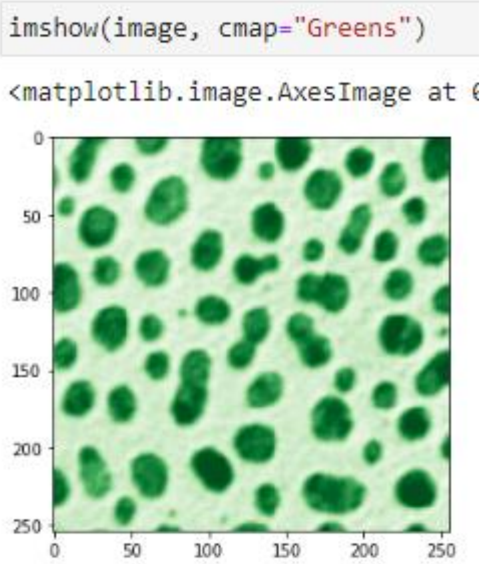
```
from skimage.io import imshow

imshow(image)
```

```
<matplotlib.image.AxesImage at 0x245e7
```



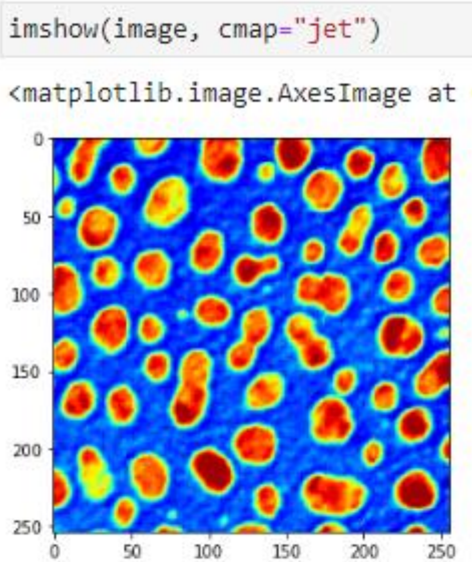
```
imshow(image, cmap="Greens_r")
```

```
<matplotlib.image.AxesImage at 0:
```



```
imshow(image, cmap="Greens")
```

```
<matplotlib.image.AxesImage at
```



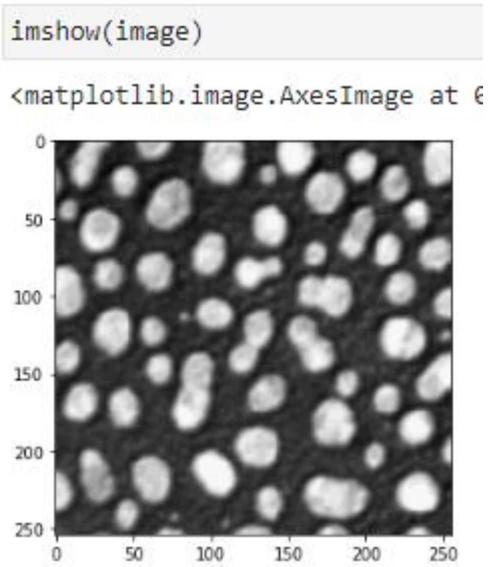
```
imshow(image, cmap="jet")
```

```
<matplotlib.image.AxesImage at
```

This does not modify the image data. The images are just shown with different colors representing the same values.

Cropping, sampling and flipping images

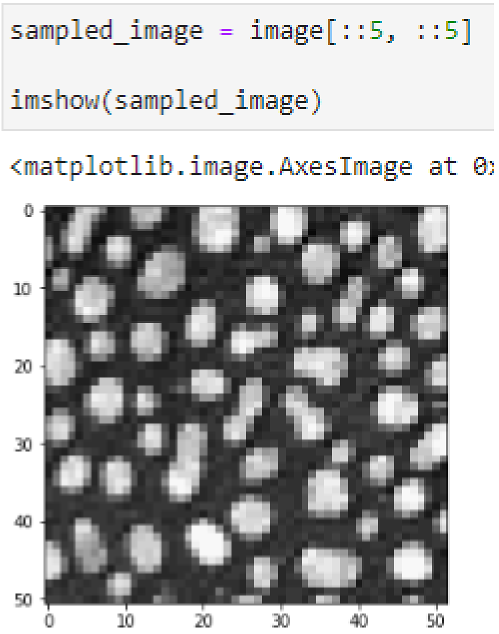
- Indexing and cropping *numpy-arrays* works like with python arrays.



```
imshow(image)
```

```
<matplotlib.image.AxesImage at 0
```

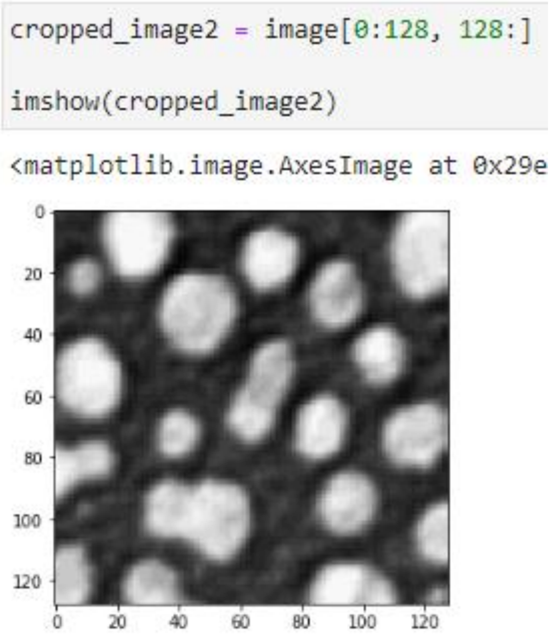
Original image



```
sampled_image = image[::5, ::5]

imshow(sampled_image)
```

```
<matplotlib.image.AxesImage at 0x
```

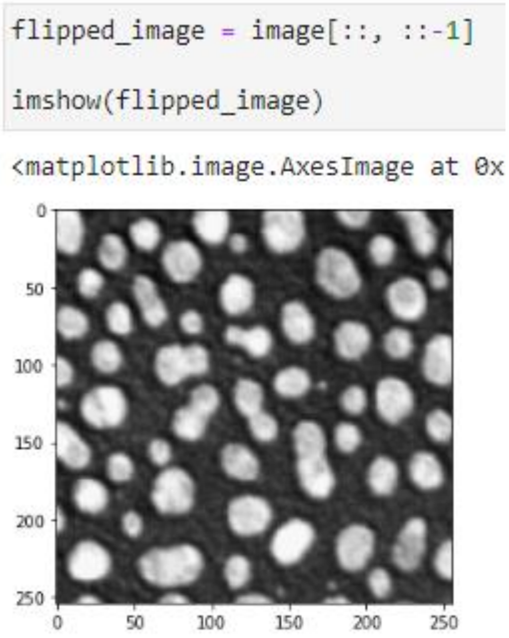
Sub-sampled image



```
cropped_image2 = image[0:128, 128:]

imshow(cropped_image2)
```

```
<matplotlib.image.AxesImage at 0x29e
```

Cropped image



```
flipped_image = image[::, ::-1]

imshow(flipped_image)
```

```
<matplotlib.image.AxesImage at 0x
```

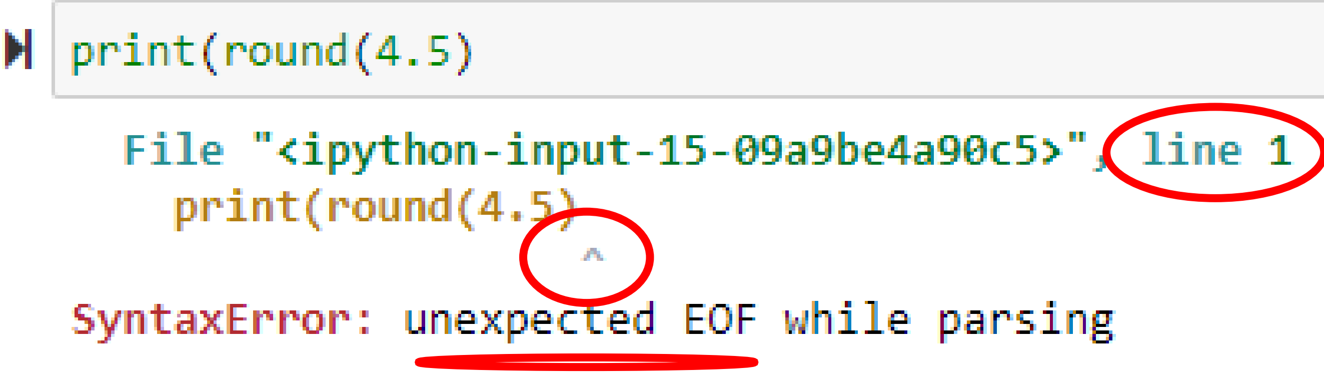
Flipped image

Troubleshooting

If your program throws error messages:

- Don’t panic.
- *“There are two ways to write error-free programs; only the third one works.”*

  Alan J. Perlis, Yale University
- Read where the error happened.
  - You may see your fault immediately, when looking at the right point.
- Read what appears to be wrong.
  - If you know, what’s missing, you may see it, even if it’s missing in a slightly different place.
  - Sometimes, something related is missing



```
print(round(4.5)

  File "<ipython-input-15-09a9be4a90c5>", line 1
    print(round(4.5)
                    ^
SyntaxError: unexpected EOF while parsing
```

Summary

Take home messages

- Lists can be accessed like this:

```
data[start:stop:step]
```

- Strings are lists of characters
- Tuples are immutable lists
- Columns in tables are lists
- Images are multi-dimensional lists or simply numpy arrays

- Learning how to deal with lists in Python is key.