Machine Learning Introduction and Linear Algebra

AcadView

May 16, 2018

1 Overview

Welcome to this course on machine learning!

Machine learning is the science of getting computers to act without being explicitly programmed. In the past decade, machine learning has given us self-driving cars, practical speech recognition, effective web search, and a vastly improved understanding of the human genome. Machine learning is so pervasive today that you probably use it dozens of times a day without knowing it. Many researchers also think it is the best way to make progress towards human-level AI.
Tom M. Mitchell provided a widely quoted, more formal definition of the algorithms studied in the machine learning field.The simple definition of the Machine Learning is **"Machine Learning is said to learn from experience E w.r.t some class of task T and a performance measure P if learners performance at the task in the class as measured by P improves with experiences."** as Tom M. Mitchell defined.

2 What will you learn?

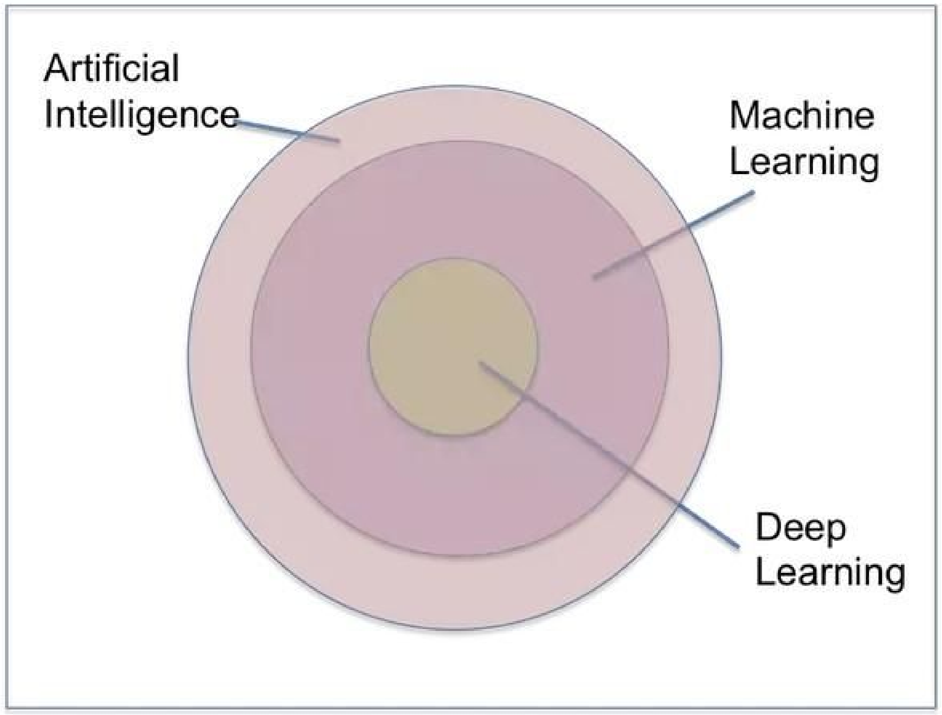
In this course, you will learn about the most effective machine learning techniques, and gain practice implementing them and getting them to work for yourself. More importantly, you'll learn about not only the theoretical underpinnings of machine learning, but also gain the practical know-how needed to quickly and powerfully apply these techniques to new problems. Finally, you'll learn about some of Silicon Valley's best practices in innovation as it pertains to machine learning and AI.

3 Machine Leaning vs AI vs Deep Learning

AI: "It is the study of how to train the computers so that computers can do things which at present human can do better."

Deep Learning: “Deep Learning is a sub-field of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks.”

Well let’s put all the definitions to one side. What we can do is take a look at the diagram which is kind of self explanatory of where AI,Machine Leaning and Deep Learning lie.



4 Some common terms

As you might have encountered some of the terms now like “training examples”,”trained” etc.
Let’s see some of them in details that we will be using throughout the course.

- **Data:** This is the most essential component required to carry out any machine learning algorithm.This is what the algorithms works on and produces the desired results.
- **Dataset:** A set of pre-processed data that is ready to be fed into the machine learning algorithms.

- **Training:**During training the machine learning algorithm tries to identify a general rule or relation that maps the inputs to the desired outputs.
- **Training Examples:** In case of supervised machine learning models , we have a set of data in which each entry consists of a input and desired output pairs.Each entry is called a training example.
- **Train set:** A dataset of training examples that is used for training the model.
- **Model:** A wrapped package containing the entire machine learning algorithm which takes the data as the input while training.
- **Test set:** A small dataset of trainign examples that the algoritms has not seen.This is used to test how well the trained model generalizes on the given task.

5 Machine Learning Tasks

Among the different types of ML tasks, a crucial distinction is drawn between supervised and unsupervised learning:

- **Supervised machine learning:** The program is “trained” on a pre-defined set of “training examples”, which then facilitate its ability to reach an accurate conclusion when given new data.Or more simply put ,The computer is presented with example inputs and their desired outputs, given by a ”teacher”, and the goal is to learn a general rule that maps inputs to outputs.

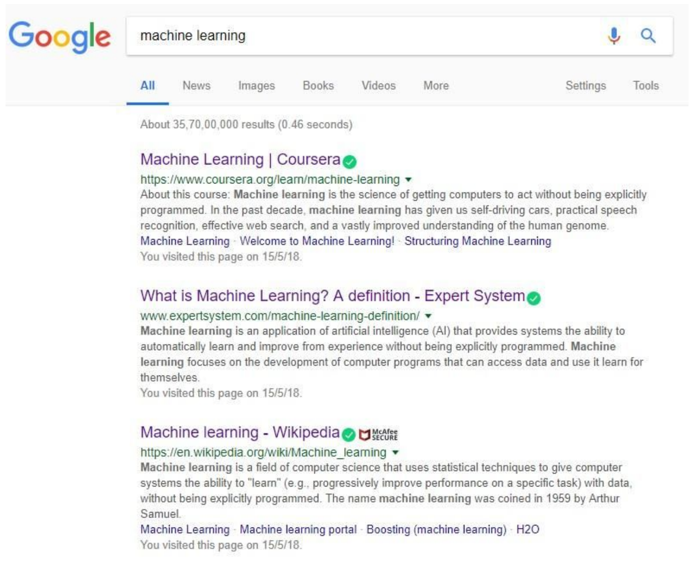
For example, with supervised learning, an algorithm may be fed data with images of sharks labeled as fish and images of oceans labeled as water. By being trained on this data, the supervised learning algorithm should be able to later identify unlabeled shark images as fish and unlabeled ocean images as water.

- **Unsupervised machine learning:** The program is given a bunch of data and must find patterns and relationships therein.No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).

For example,A friend invites you to his party where you meet totally strangers. Now you will classify them using unsupervised learning (no prior knowledge) and this classification can be on the basis of gender, age group, dressing, educational qualification or whatever way you would like.

6 Applications of Machine Learning

- Most readers will be familiar with the concept of web page ranking. That is, the process of submitting a query to a search engine, which then finds webpages relevant to the query and which returns them in their order of relevance. Below is an example of the query results for "machine learning"



About 35,70,00,000 results (0.46 seconds)

Machine Learning | Coursera
https://www.coursera.org/learn/machine-learning
About this course: **Machine learning** is the science of getting computers to act without being explicitly programmed. In the past decade, **machine learning** has given us self-driving cars, practical speech recognition, effective web search, and a vastly improved understanding of the human genome.
Machine Learning · Welcome to Machine Learning! · Structuring Machine Learning
You visited this page on 15/5/18.

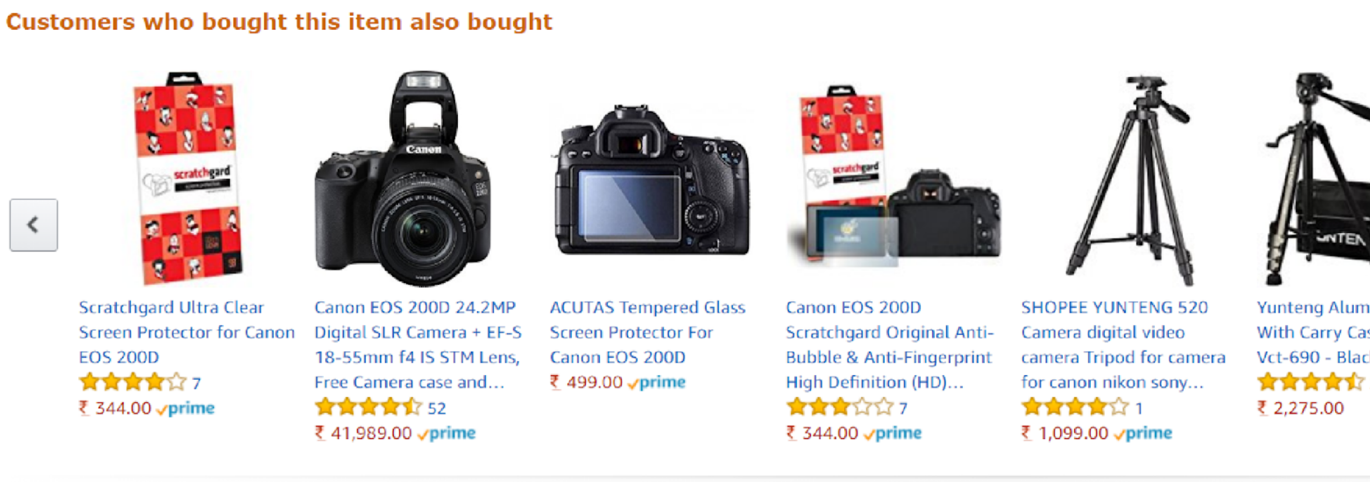
What is Machine Learning? A definition - Expert System
www.expertsystem.com/machine-learning-definition/
Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. **Machine learning** focuses on the development of computer programs that can access data and use it learn for themselves.
You visited this page on 15/5/18.

Machine learning - Wikipedia
https://en.wikipedia.org/wiki/Machine_learning
Machine learning is a field of computer science that uses statistical techniques to give computer systems the ability to "learn" (e.g., progressively improve performance on a specific task) with data, without being explicitly programmed. The name **machine learning** was coined in 1959 by Arthur Samuel.
Machine Learning · Machine learning portal · Boosting (machine learning) · H2O
You visited this page on 15/5/18.

Increasingly machine learning rather than guesswork and clever engineering is used to automate the process of designing a good search engine.

- A rather related application is collaborative filtering. Internet stores such as Amazon, or video content sites such as Netflix use this information extensively to entice users to recommend additional products based on their browsing interests.

- Today, the machine learning algorithms are extensively used to find the solutions to various challenges arising in manufacturing self-driving cars.Below is a link to a fun



Customers who bought this item also bought

Scratchgard Ultra Clear Screen Protector for Canon EOS 200D
★★★★☆ 7
₹ 344.00 prime

Canon EOS 200D 24.2MP Digital SLR Camera + EF-S 18-55mm f4 IS STM Lens, Free Camera case and…
★★★★☆ 52
₹ 41,989.00 prime

ACUTAS Tempered Glass Screen Protector For Canon EOS 200D
₹ 499.00 prime

Canon EOS 200D Scratchgard Original Anti-Bubble & Anti-Fingerprint High Definition (HD)…
★★★☆☆ 7
₹ 344.00 prime

SHOPEE YUNTENG 520 Camera digital video camera Tripod for camera for canon nikon sony…
★★★★☆ 1
₹ 1,099.00 prime

Yunteng Alum
With Carry Ca
Vct-690 - Blac
★★★★☆
₹ 2,275.00

video demonstrating a fully autonomous drive in a rainy night. https://www.youtube.com/watch?v=GMvgtPN2IBU

- Recently Google's assistant is launched with a feature update that can call and book appointments for you. https://www.youtube.com/watch?v=2V6NHKmfnW0

All of these cool things can be done using machine learning . Today it's so widespread that we use it all around us even without noticing it . Pick up your phone the auto text suggestion works because of machine learning .Your SIRI or Google Assistant works on machine learning.It's all around you...

7 Lists

A list is a data structure in Python that is a mutable, or changeable, ordered sequence of elements. Each element or value that is inside of a list is called an item. Just as strings are defined as characters between quotes, lists are defined by having values between square brackets [ ].

Lists are great to use when you want to work with many related values. They enable you to keep data together that belongs together, condense your code, and perform the same methods and operations on multiple values at once.

When thinking about Python lists and other data structures that are types of collections, it is useful to consider all the different collections you have on your computer: your assortment of files, your song playlists, your browser bookmarks, your emails, the collection of videos you can access on a streaming service, and more.

To get started, lets create a list that contains items of the string data type: **sea creatures** = [^{0}**shark**^{0},^{0} **cuttlefish**^{0},^{0} **squid**^{0},^{0} **mantisshrimp**^{0},^{0} **anemone**^{0}]

When we print out the list, the output looks exactly like the list we created: **print**(**sea _creatures**)

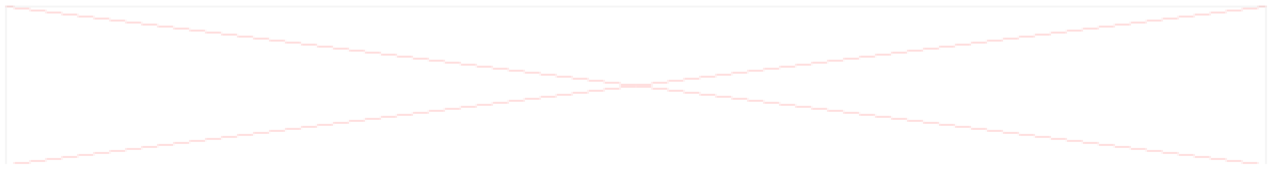
Output:
[^{0}**shark**^{0},^{0} **cuttlefish**^{0},^{0} **squid**^{0},^{0} **mantisshrimp**^{0},^{0} **anemone**^{0}]

As an ordered sequence of elements, each item in a list can be called individually, through indexing. Lists are a compound data type made up of smaller parts, and are very flexible because they can have values added, removed, and changed. When you need to store a lot of values, and you want to be able to readily modify those values, youll likely want to work with list data types.

In this class, well go through some of the ways that we can work with lists in Python.

7.1 Indexing Lists

Each item in a list corresponds to an index number, which is an integer value, starting with the index number 0.
For the list **sea creatures**, the index breakdown looks like this:

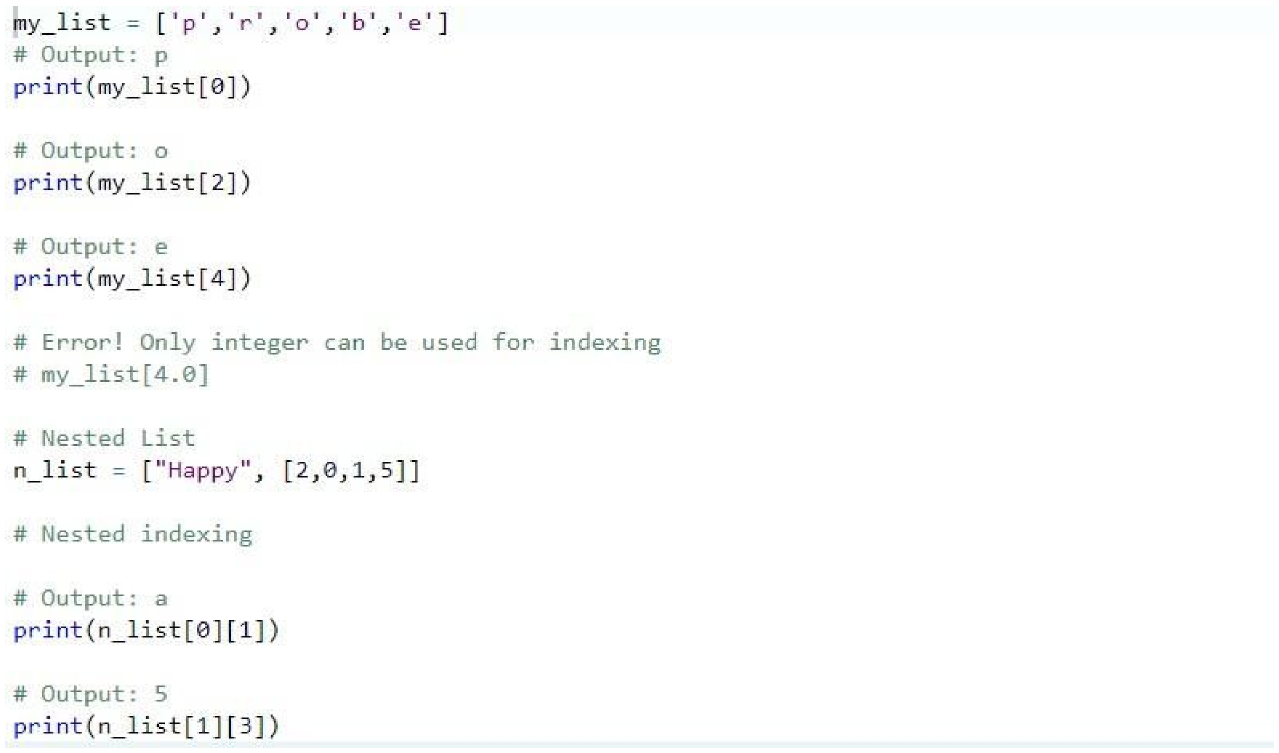


The first item, the string ’shark’ starts at index 0, and the list ends at index 4 with the item ’anemone’.
Because each item in a Python list has a corresponding index number, were able to access and manipulate lists very easily.

Now we can call any item of the list by referring to its index number:
print(**sea creatures**) Output: **cuttlefish**

The index numbers for this list range from 0-4, as shown in the table above. So to call any of the items individually, we would refer to the index numbers like this:
sea creatures[**0**] =^{0} **shark**^{0} **sea**
creatures[1] =^{0} **cuttlefish**^{0} **sea**
creatures[2] =^{0} **squid**^{0} **sea**
creatures[3] =^{0} **mantisshrimp**^{0} **sea**
creatures[4] =^{0} **anemone**^{0}

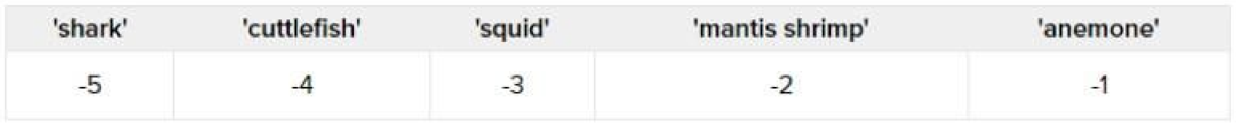
If we call the list sea creatures with an index number of any that is greater than 4, it will be out of range as it will not be valid:
print(**sea creatures**[**18**])
Output:
IndexError : **listindexoutofrange**



```
my_list = ['p','r','o','b','e']
# Output: p
print(my_list[0])

# Output: o
print(my_list[2])

# Output: e
print(my_list[4])

# Error! Only integer can be used for indexing
# my_list[4.0]

# Nested List
n_list = ["Happy", [2,0,1,5]]

# Nested indexing

# Output: a
print(n_list[0][1])

# Output: 5
print(n_list[1][3])
```

In addition to positive index numbers, we can also access items from the list with a negative index number, by counting backwards from the end of the list, starting at -1. This is especially useful if we have a long list and we want to pinpoint an item towards the end of a list.

For the same list sea creatures, the negative index breakdown looks like this:

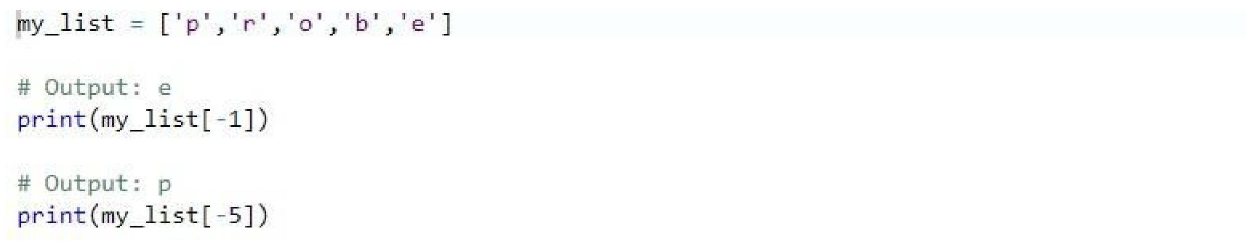


'shark'	'cuttlefish'	'squid'	'mantis shrimp'	'anemone'
-5	-4	-3	-2	-1

So, if we would like to print out the item ’squid’ by using its negative index number, we can do so like this:

print(**sea creatures**[−**3**])

Output: **squid example:**



```
my_list = ['p','r','o','b','e']

# Output: e
print(my_list[-1])

# Output: p
print(my_list[-5])
```

7.2 Modifying Items in Lists

: We can use indexing to change items within the list, by setting an index number equal to a different value. This gives us greater control over lists as we are able to modify and update the items that they contain.
If we want to change the string value of the item at index 1 from ’cuttlefish’ to ’octopus’, we can do so like this: **sea creatures**[**1**] =^{0} **octopus**^{0}

Now when we print sea _creatures, the list will be different: **print**(**sea creatures**) Output:
[^{0}**shark**^{0},^{0} **octopus**^{0},^{0} **squid**^{0},^{0} **mantisshrimp**^{0},^{0} **anemone**^{0}]

We can also change the value of an item by using a negative index number instead:
sea creatures[−**3**] =^{0} **blobfish**^{0}
print(**sea creatures**) Output:
[^{0}**shark**′,′ **octopus**′,′**blobfish**′,′ **mantisshrimp**′,′**anemone**^{0}]
Now ’blobfish’ has replaced ’squid’ at the negative index number of -3 (which corresponds to the positive index number of 2). Being able to modify items in lists gives us the ability to change and update lists in an efficient way.

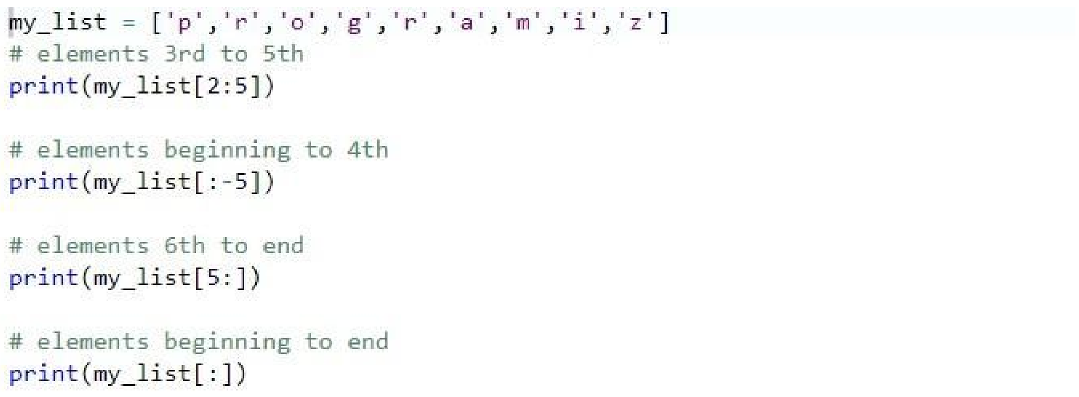
7.3 Slicing Lists

We can also call out a few items from the list. Lets say we would like to just print the middle items of sea _creatures, we can do so by creating a slice. With slices, we can call multiple values by creating a range of index numbers separated by a colon [x:y]:
print(**sea creatures**[**1** : **4**]) Output:
[^{0}**octopus**′,′**blobfish**′,′**mantisshrimp**^{0}]

When creating a slice, as in [1:4], the first index number is where the slice starts (inclusive), and the second index number is where the slice ends (exclusive), which is why

in our example above the items at position, 1, 2, and 3 are the items that print out. If we want to include either end of the list, we can omit one of the numbers in the list[x:y] syntax.
For example, if we want to print the first 3 items of the list sea creatures– which would be ’shark’, ’octopus’, ’blobfish’ – we can do so by typing: **print**(**sea creatures**[: **3**]) Output:
[^{0}**shark**$'$,$'$**octopus**$'$,$'$**blobfish**^{0}]

This printed the beginning of the list, stopping right before index 3. To include all the items at the end of a list, we would reverse the syntax:
print(**sea creatures**[**2** :]) Output:
[^{0}**blobfish**$'$,$'$**mantisshrimp**$'$,$'$**anemone**^{0}]



```
my_list = ['p','r','o','g','r','a','m','i','z']
# elements 3rd to 5th
print(my_list[2:5])

# elements beginning to 4th
print(my_list[:-5])

# elements 6th to end
print(my_list[5:])

# elements beginning to end
print(my_list[:])
```

We can also use negative index numbers when slicing lists, just like with positive index numbers:
print(**sea creatures**[−**4** : −**2**])
print(**sea creatures**[−**3** :])
Output:
[^{0}**octopus**$'$,$'$**blobfish**^{0}]
[^{0}**blobfish**$'$,$'$**mantisshrimp**$'$,$'$**anemone**^{0}]

Here are some other common list methods.

- **list.append(elem)** – adds a single element to the end of the list. Common error: does not return the new list, just modifies the original.
- **list.insert(index, elem)** – inserts the element at the given index, shifting elements to the right.
- **list.extend(list2)**– adds the elements in list2 to the end of the list. Using + or +=

on a list is similar to using extend().

- **list.index(elem)** – searches for the given element from the start of the list and returns its index. Throws a ValueError if the element does not appear (use ”in” to check without a ValueError).
- **list.remove(elem)** – searches for the first instance of the given element and removes it (throws ValueError if not present)
- **list.sort()** – sorts the list in place (does not return it). (The sorted() function shown below is preferred.)
- **list.reverse()** – reverses the list in place (does not return it)
- **list.pop(index)** – removes and returns the element at the given index. Returns the rightmost element if index is omitted (roughly the opposite of append()).
- **list.count(x)**–Return the number of times x appears in the list.

An example that uses most of the list methods:

You might have noticed that methods like insert, remove or sort that only modify the list have no return value printed - they return the default None. This is a design principle for all mutable data structures in Python.

8 Linear Algebra

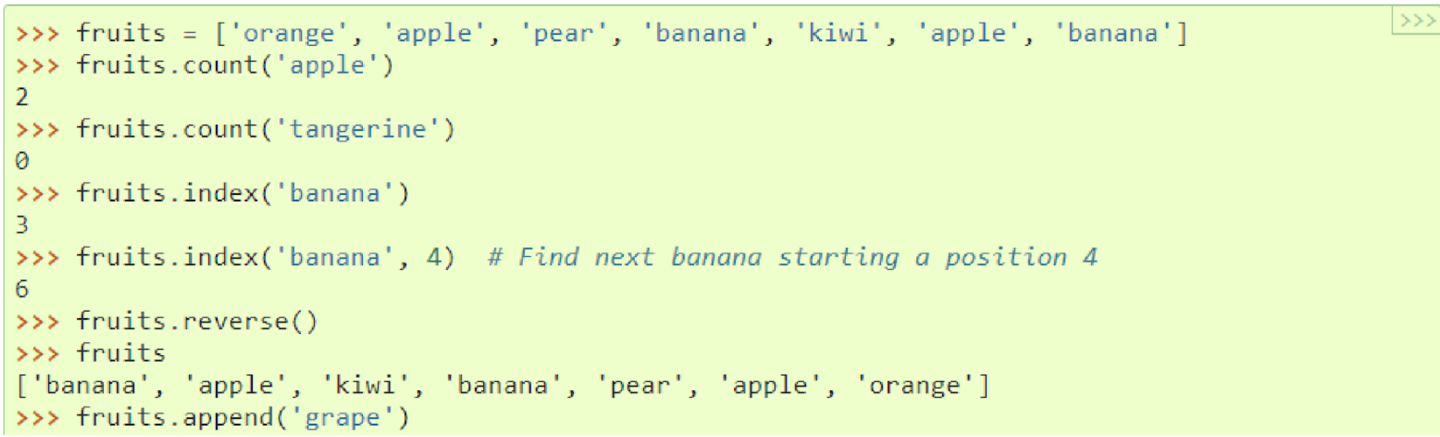
8.1 Basic Concepts and Notation

Linear algebra provides a way of compactly representing and operating on sets of linear equations. For example, consider the following system of equations:

$$4x_1 - 5x_2 = 13$$
$$2x_1 + 3x_2 = 9$$

This is two equations and two variables, so as you know from high school algebra, you can find a unique solution for x_1 and x_2 (unless the equations are somehow degenerate, for example if the second equation is simply a multiple of the first, but in the case above there is in fact a unique solution). In matrix notation, we can write the system more compactly

$$Ax = b$$



```
>>> fruits = ['orange', 'apple', 'pear', 'banana', 'kiwi', 'apple', 'banana']
>>> fruits.count('apple')
2
>>> fruits.count('tangerine')
0
>>> fruits.index('banana')
3
>>> fruits.index('banana', 4)  # Find next banana starting a position 4
6
>>> fruits.reverse()
>>> fruits
['banana', 'apple', 'kiwi', 'banana', 'pear', 'apple', 'orange']
>>> fruits.append('grape')
```

with

A= $\begin{bmatrix} 4 & -5 \\ -2 & 3 \end{bmatrix}$ B= $\begin{bmatrix} -13 \\ 9 \end{bmatrix}$ $x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

8.1.1 Notation

We use the following notation:

- By $A \in \mathrm{R}^{m\times n}$ we denote a matrix with m rows and n columns, where the entries of A are real numbers.
- By $x \in \mathrm{R}^{n}$, we denote a vector with n entries. By convention, an n-dimensional vector is often thought of as a matrix with n rows and 1 column, known as a column vector. If we want to explicitly represent a row vector a matrix with 1 row and n columns –we typically write x^T (here x^T denotes the transpose of x, which we will define shortly).
- The ith element of a vector x is denoted x_i:

x_1

x_2

A= ...

x_n

- We use the notation a_{ij} to denote the entry of A in the ith row and jth column:

a_{11} a_{13} ... a_{1n}

a_{12}

a_{23} ... a_{2n}

a_{21} a_{22} A=

... ...

... ...

a_{m3}

... a_{mn}

a_{m1} a_{m2}

- We denote the jth column of A by a_j:

| $a_2 \ldots$ |

A= a_1 | | a_n |

8.2 Matrix Multiplication

The product of two matrices $A \in \mathrm{R}^{m \times n}$ and $B \in \mathrm{R}^{n \times p}$ is the matrix

$$C = AB \in \mathrm{R}^{m \times p},$$

where

$$C_{ij} = \sum_{n}^{k=1} A_{ik} B_{kj}$$

Note that in order for the matrix product to exist, the number of columns in A must equal the number of rows in B. There are many ways of looking at matrix multiplication, and well start by examining a few special cases.

8.3 Vector-Vector Products

Given two vectors $x,y \in \mathrm{R}^n$, the quantity $x^T y$, sometimes called the inner product or dot product of the vectors, is a real number given by

$$x^T y \in \mathbb{R} = \begin{bmatrix} x_1 & x_2 & \ldots & x_n \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \sum_{i=1}^{n} x_i y_i$$

.

Observe that inner products are really just special case of matrix multiplication. Note that it is always the case that $x_T y = y_T x$. Given vectors $x \in \mathrm{R}^m$, $y \in \mathrm{R}^n$ (not necessarily of the same size), $xy_T \in \mathrm{R}^{m \times n}$ is called the outer product of the vectors. It is a matrix whose entries are given by $(xy_T)_{ij} = x_i y_j$ i.e.,

$$xy^T \in \mathbb{R}^{m\times n} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \begin{bmatrix} y_1 & y_2 & \dots & y_n \end{bmatrix} = \begin{bmatrix} x_1y_1 & x_1y_2 & x_1y_3 & \dots & x_1y_n \\ x_2y_1 & x_1y_2 & x_2y_3 & \dots & x_2y_n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ x_my_1 & x_my_2 & x_my_3 & \dots & x_my_n \end{bmatrix}$$

8.4 Matrix-Vector Products

Given a matrix $A \in \mathrm{R}^{m\times n}$ and a vector$x \in \mathrm{R}^n$, their product is a vector $y = Ax \in \mathrm{R}^m$. There are a couple ways of looking at matrix-vector multiplication, and we will look at each of them in turn. If we write A by rows, then we can express Ax as,

$$y = Ax = \begin{bmatrix} --- & a_1^T & --- \\ --- & a_2^T & --- \\ & \vdots & \\ --- & a_m^T & --- \end{bmatrix} x = \begin{bmatrix} a_1^T x \\ a_2^T x \\ \vdots \\ a_n^T x \end{bmatrix}$$

In other words, the ith entry of y is equal to the inner product of the *i*th row of *A* and $x,\ y_i = a_i^T x$ Alternatively, lets write *A* in column form. In this case we see that,

x^1

x_2 $\quad$ 1 a

$y = Ax = a_2 \ldots$ $\quad a_n \ldots = a_1x_1 + a_2x_2 + \ldots + a_nx_n$

x_n

In other words, y is a linear combination of the columns of A, where the coefficients of the linear combination are given by the entries of x.

8.5 Operations and properties

8.5.1 The Identity Matrix and Diagonal Matrices

The ***identity matrix***, denoted $I \in \mathbb{R}^{n\times n}$, is a square matrix with ones on the diagonal and zeros everywhere else. That is,

$$I_{ij} = \begin{cases} 1 & i = j \\ 0 & i \neq j \end{cases}$$

It has the property that for all $A \in \mathbb{R}^{m\times n}$,

$$AI = A = IA.$$

Note that in some sense, the notation for the identity matrix is ambiguous, since it does not specify the dimension of I. Generally, the dimensions of I are inferred from context so as to make matrix multiplication possible. For example, in the equation above, the I in $AI = A$ is an $n \times n$ matrix, whereas the I in $A = IA$ is an $m \times m$ matrix.

A ***diagonal matrix*** is a matrix where all non-diagonal elements are 0. This is typically denoted $D = \text{diag}(d_1, d_2, \ldots, d_n)$, with

$$D_{ij} = \begin{cases} d_i & i = j \\ 0 & i \neq j \end{cases}$$

Clearly, $I = \text{diag}(1, 1, \ldots, 1)$.

8.6 The Transpose

The ***transpose*** of a matrix results from "flipping" the rows and columns. Given a matrix $A \in \mathbb{R}^{m\times n}$, its transpose, written $A^T \in \mathbb{R}^{n\times m}$, is the $n \times m$ matrix whose entries are given by

$$(A^T)_{ij} = A_{ji}.$$

We have in fact already been using the transpose when describing row vectors, since the transpose of a column vector is naturally a row vector.

The following properties of transposes are easily verified:

- $(A^T)^T = A$
- $(AB)^T = B^T A^T$
- $(A + B)^T = A^T + B^T$

8.7 Norm

A ***norm*** of a vector $\|x\|$ is informally a measure of the "length" of the vector. For example, we have the commonly-used Euclidean or ℓ_2 norm,

$$\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}.$$

Note that $\|x\|_2^2 = x^T x$.

More formally, a norm is any function $f : \mathbb{R}^n \to \mathbb{R}$ that satisfies 4 properties:

1. For all $x \in \mathbb{R}^n$, $f(x) \geq 0$ (non-negativity).
2. $f(x) = 0$ if and only if $x = 0$ (definiteness).
3. For all $x \in \mathbb{R}^n$, $t \in \mathbb{R}$, $f(tx) = |t| f(x)$ (homogeneity).
4. For all $x, y \in \mathbb{R}^n$, $f(x + y) \leq f(x) + f(y)$ (triangle inequality).

8.8 Inverse