

Deep Dive into Engineering

Presented by
Bunyodbek Ibrokhimov

21.01.2023

```
def operation == "MIRROR_X":  
    mirror_mod.use_x = False  
    mirror_mod.use_y = True  
    mirror_mod.use_z = False  
elif operation == "MIRROR_Z":  
    mirror_mod.use_x = False  
    mirror_mod.use_y = False  
    mirror_mod.use_z = True
```

```
#selection at the end -add back the deselected mirror modifier  
mirror_ob.select= 1  
modifier_ob.select=1  
bpy.context.scene.objects.active = modifier_ob  
print("Selected" + str(modifier_ob)) # modifier ob is the active ob  
mirror_ob.select = 0  
bpy.context.selected_objects[0]  
bpy.data.objects[mirror_ob.name].select = 0
```

Agenda

01

What is
software
engineering?

02

Where to
start:
A Roadmap

03

Different
levels of
software
engineer

04

Tips to
become a
pro

Software Engineering



ChatGPT

Software engineering:

The design and development of software systems, including the use of [programming languages](#), software development [methodologies](#), and [software testing](#).

From coding to ML engineer

Computer engineering:

Software Engineering +

- **Computer architecture:** The design and organization of computer systems, including the design of processors, memory systems, and other components.
- **Computer networks:** The design, implementation, and maintenance of computer networks and communication systems, including local area networks (LANs), wide area networks (WANs), and the internet.
- **Embedded systems:** The design and development of computer systems that are integrated into other devices or products, such as cars, appliances, and medical equipment.

Where to start: A Roadmap



Programming language

Python

- Syntax
- Basics (variables, data types, casting)
- Loops, iterators, scope, modules, function



Algorithms and data structures

- **Sequence of instructions**
- **A process or set of rules**
- Dictionaries
- Linked lists
- Binary search tree
- Arrays, Lists, Sets



Core ML/DL concepts


- Matrix multiplication
- Neural network structure (layers)
- Inference
- Backpropagation
- Activation functions
- Error functions
- Regularizations



Tools and frameworks

- Pytorch
- Tensorflow
- Keras
- Pandas
- NumPy
- Matplotlib
- OpenCV
- ROS
- Git/GitLab
- Linux

Python

Tutorials▼References▼Exercises▼Videos

Sign UpUpgradeGet CertifiedCreate WebsiteLog in

HTMLCSSJAVASCRIPTSQLPYTHONJAVAPHBOOTSTRAPHOW TOW3.CSSCC#REACTRJQUERYDJANGOTYPESCRIPT

Python Tutorial

Python HOME

Python Intro

Python Get Started

Python Syntax

Python Comments

Python Variables

Python Data Types

Python Numbers

Python Casting

Python Strings

Python Booleans

Python Operators

Python Lists

Python Tuples

Python Sets

Python Dictionaries

Python If...Else

Python While Loops

Python For Loops

Python Functions

Python Lambda

Python Arrays

Python Classes/Objects

Python Inheritance

Python Iterators

Python Scope

Python Modules

Python Tutorial

< Home

Next >

Learn Python

Python is a popular programming language.

Python can be used on a server to create web applications.


Start learning Python now »

Learning by Examples

With our "Try it Yourself" editor, you can edit Python code and view the result.

Example


Unlock POWERFUL features!







Upgrade your learning journey for only \$4.99

Find out more

COLOR PICKER





Get certified by completing a Python course today!

Python

- YouTube
- Coursera
- Udemy
- Codecademy
- Codefinity (from zero to hero)
- W3Schools

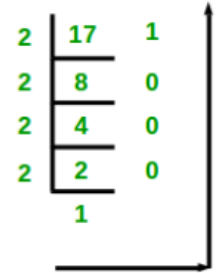


Algorithm

- **Covert Decimal (17) to binary (10001)**
- **Valid palindrome**
- **Remove zeros from string**
- ...

- 1 read (number)
- 2 loop (number > 0)
 - 1 digit = number modulo 2
 - 2 print (digit)
 - 3 number = number / 2

Decimal number : 17



Binary number: 10001

```
test.cpp
1 void decToBinary(int n){
2     int binaryNum;
3     int i = 0;
4     while (n > 0) {
5         binaryNum[i] = n % 2;
6         n = n / 2;
7         i++;
8     }
9     for (int j = i - 1; j >= 0; j--)
10        cout << binaryNum[j];
11 }
12 int main(){
13     int n = 17;
14     decToBinary(n);
15     return 0;
16 }
```

```
1 def decToBinary(n):
2     binaryNum = [0] * n
3     i = 0
4     while (n > 0):
5         binaryNum[i] = n % 2
6         n = int(n / 2)
7         i += 1
8     for j in range(i - 1, -1, -1):
9         print(binaryNum[j], end = "")
10 n = 17
11 decToBinary(n)
```

Algorithm

- Covert Decimal (17) to binary (10001)
- Valid palindrome
- Remove zeros from string
- ...

Integer: 1221

1221 // 1000 == 1221 % 10

1221 // 100 == 1221 % 100

1221 // 10 == 1221 % 1000

127809721

```
1 def isPalindrome(str):
2     for i in range(0, int(len(str)/2)):
3         if str[i] != str[len(str)-i-1]:
4             return False
5     return True
6
7 s = "malayalam"
8 ans = isPalindrome(s)
9
10 if (ans):print("Yes")
11 else:print("No")
```

Input : malayalam

Output : Yes

Input : geeks

Output : No

```
def isPalindrome(s):
    return s == s[::-1]

s = "malayalam"
ans = isPalindrome(s)

if ans:print("Yes")
else:print("No")
```

Time complexity: $O(n)$

Auxiliary Space: $O(1)$

Data structure

- Search from sorted array

Input: arr = [-1,0,3,5,9,12], target = 9

Output: 4

```
1 arr = [-1,0,3,5,9,12]
2 target = 13
3 def search(target):
4     for i in range(len(arr)):
5         if arr[i] == target:
6             return(i)
7     return -1
8 print(search(target))
```

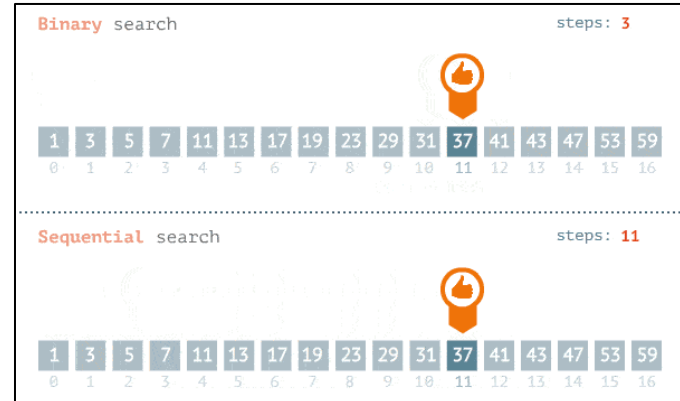
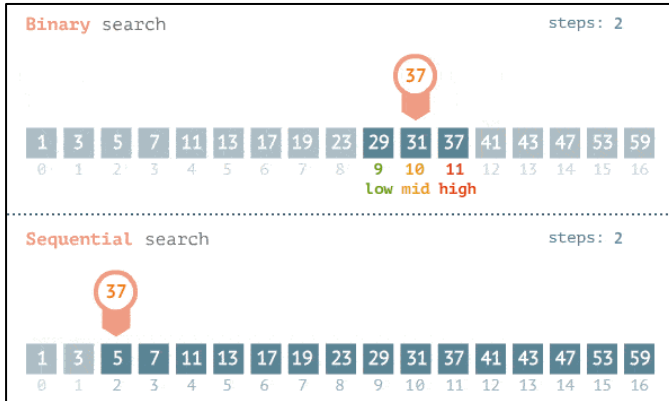
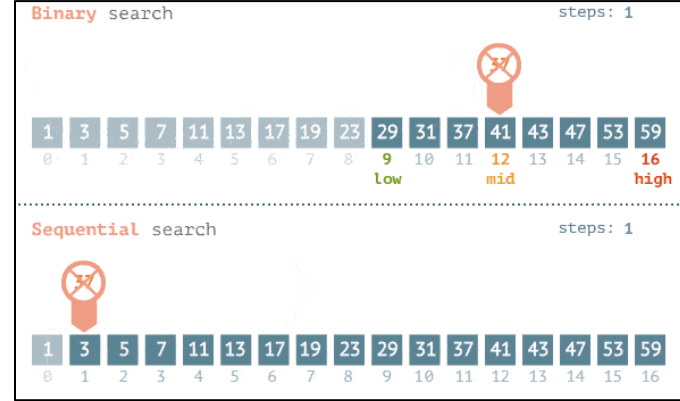
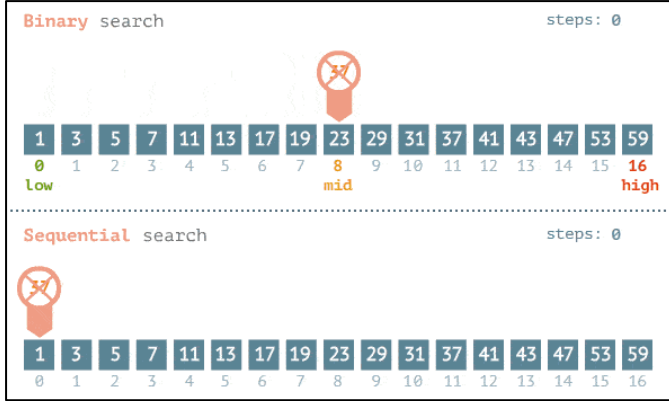
Time complexity: $O(n)$

```
1 arr = [-1,0,3,5,9,12]
2 target = 9
3 def search(self, arr, target):
4     left = 0
5     right = len(arr)-1
6     while left<=right:
7         mid = (left+right)//2
8         if arr[mid]==target:
9             return mid
10        elif arr[mid]>target:
11            right = mid-1
12        else:
13            left = mid+1
14    return -1
```

```
1 arr = [-1,0,3,5,9,12]
2 target = 9
3 def search(arr, target):
4     if not target in arr:
5         return -1
6     else:
7         return arr.index(target)
```

Time complexity: $O(\log n)$

Data structure



Data structure

<https://www.programiz.com/dsa/data-structure-types>

DSA Introduction

What is an algorithm?

Data Structure and Types

Why learn DSA?

Asymptotic Notations

Master Theorem

Divide and Conquer Algorithm

Data Structures (I)

Data Structures (II)


Tree based DSA (I)

Tree based DSA (II)

Graph based DSA

Sorting and Searching Algorithms

Greedy Algorithms

Get 10 Free Images From Adobe Stock. Start Now.

ADS VIA CARBON

Data Structure and Types

In this article, you will learn about data structure and its types.

What are Data Structures?

Data structure is a storage that is used to store and organize data. It is a way of arranging data on a computer so that it can be accessed and updated efficiently.

Depending on your requirement and project, it is important to choose the right data structure for your project. For example, if you want to store data sequentially in the memory, then you can go for the Array data structure.

memory locations						
1004	1005	1006	1007	1008	1009	1010
...	2	1	5	3	4	...

Where to start: A Roadmap



Programming language

Python

- Syntax
- Basics (variables, data types, casting)
- Loops, iterators, scope, modules, function



Algorithms and data structures

- **Sequence of instructions**
- **A process or set of rules**
- Dictionaries
- Linked lists
- Binary search tree
- Arrays, Lists, Sets
- Heap, stack
- Monotonic stack



Core ML/DL concepts

- Matrix multiplication
- Neural network structure (layers)
- Inference
- Backpropagation
- Activation functions
- Error functions
- Regularizations

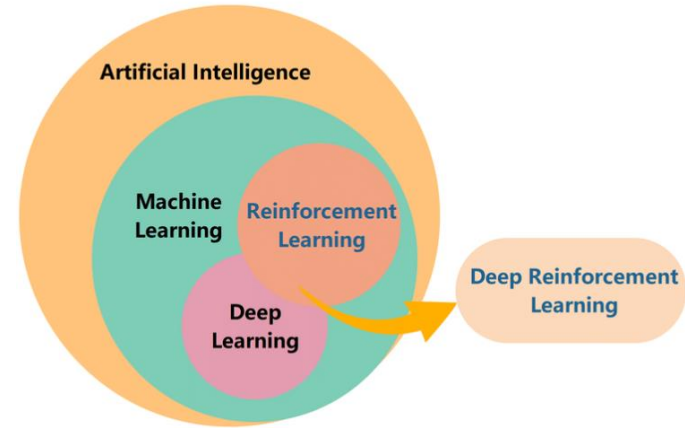


Tools and frameworks

- Pytorch
- Tensorflow
- Keras
- Pandas
- NumPy
- Matplotlib
- OpenCV
- ROS
- Git/GitLab
- Linux

Core ML/DL concepts

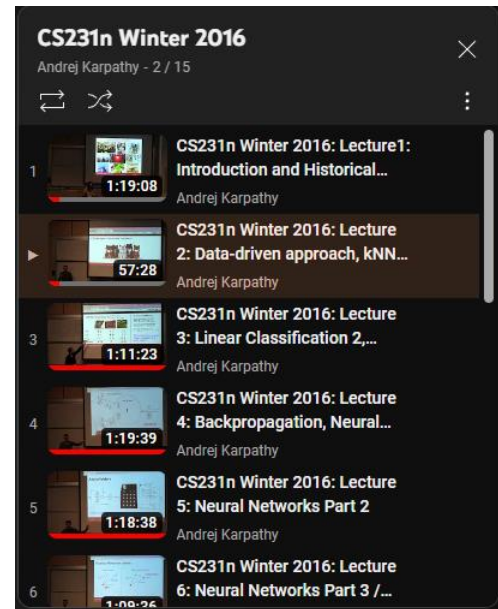
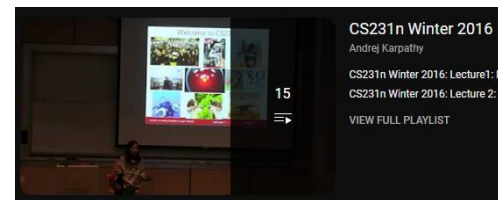
- Matrix multiplication
 - Neural network structure (layers)
 - Inference
 - Backpropagation
 - Activation functions
 - Error functions
 - Regularizations
- ML solution types
 - DNN types



Core ML/DL concepts

Must-learn courses:

- Coursera: Machine learning by Andrew Ng (ML and DL concepts)
- YouTube: cs231n 2016 winter by Andrej Karpathy (theory, impl., types)




Browse > Data Science > Machine Learning

Machine Learning Specialization

#BreakIntoAI with Machine Learning Specialization. Master fundamental AI concepts and develop practical machine learning skills in the beginner-friendly, 3-course program by AI visionary Andrew Ng

★★★★★ 4.9 7,480 ratings

 Andrew Ng +3 more instructors **TOP INSTRUCTORS**

Enroll for Free
Starts Jan 21

Financial aid available

123,947 already enrolled

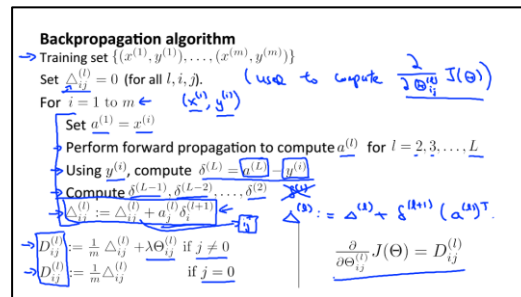
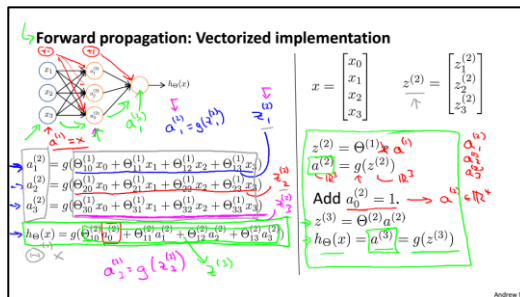
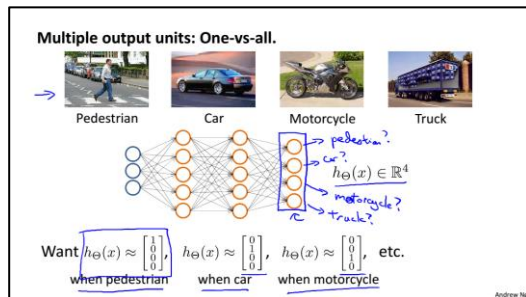
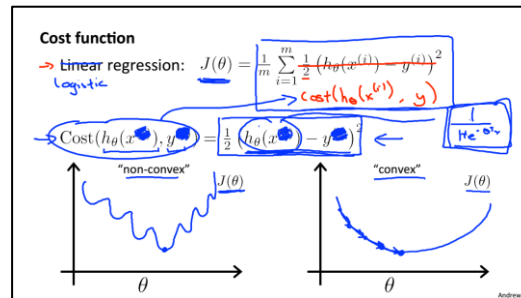
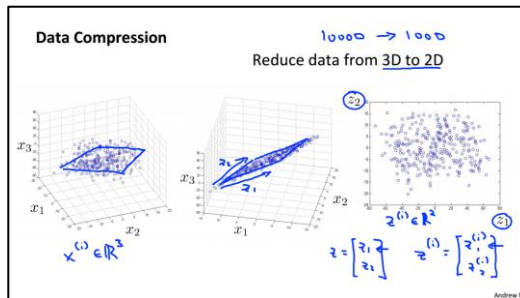
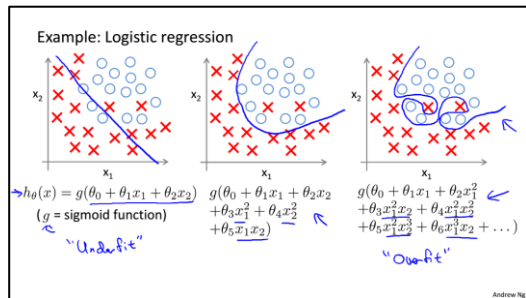
Core ML/DL concepts

- Coursera: Machine learning by Andrew Ng

COURSE		WEEK	
1	<p>Supervised Machine Learning: Regression and Classification</p> <p>★★★★★ 4.9 6,584 ratings</p> <p>In the first course of the Machine Learning Specialization, you will:</p> <ul style="list-style-type: none">• Build machine learning models in Python using popular machine learning libraries NumPy and scikit-learn. <p>SHOW ALL</p>	1	<p>7 hours to complete</p> <p>Week 1: Introduction to Machine Learning</p> <p>Welcome to the Machine Learning Specialization! You're joining millions of others who have helped millions of other learners, like you, take a look at the exciting world of machine learning.</p> <p>20 videos (Total 147 min) See All</p>
2	<p>Advanced Learning Algorithms</p> <p>★★★★★ 4.9 1,549 ratings</p> <p>In the second course of the Machine Learning Specialization, you will:</p> <ul style="list-style-type: none">• Build and train a neural network with TensorFlow to perform multi-class classification <p>SHOW ALL</p>	2	<p>10 hours to complete</p> <p>Week 2: Regression with multiple input variables</p> <p>This week, you'll extend linear regression to handle multiple input features. You'll learn about vectorization, feature scaling, feature engineering and polynomial regression.</p> <p>10 videos (Total 66 min) See All</p>
3	<p>Unsupervised Learning, Recommenders, Reinforcement Learning</p> <p>★★★★★ 4.9 729 ratings</p> <p>In the third course of the Machine Learning Specialization, you will:</p> <ul style="list-style-type: none">• Use unsupervised learning techniques for unsupervised learning: including clustering and anomaly detection. <p>SHOW ALL</p>	3	<p>16 hours to complete</p> <p>Week 3: Classification</p> <p>This week, you'll learn the other type of supervised learning, classification. You'll learn about overfitting, and how to handle this problem with a method called regularization.</p> <p>11 videos (Total 98 min), 1 reading, 5 quizzes See All</p>

Core ML/DL concepts

- Coursera: Machine learning by Andrew Ng



Core ML/DL concepts

- YouTube: cs231n 2016 winter by Andrej Karpathy

Implementation: forward/backward API

Graph (or Net) object. (Rough pseudo code)

```

class ComputationalGraph(object):
    # ...
    def forward(self):
        # 1. pass inputs to input gates...
        # 2. forward the computational graph
        for gate in self.graph.nodes.topologically_sorted():
            gate.forward()
        return [g for g in self.graph.nodes if g.is_output()]

    def backward(self):
        for gate in reversed(self.graph.nodes.topologically_sorted()):
            gate.backward() # little piece of backprop (chain rule applied)
        return [g for g in self.graph.nodes if g.is_input()]
    
```

Fei-Fei Li & Andrej Karpathy & Justin Johnson Lecture 4 - 46 13 Jan 2016

Implementation: forward/backward API

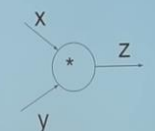
class MultiplyGate(object):

```

def forward(x,y):
    z = x*y
    self.x = x # must keep these around!
    self.y = y
    return z

def backward(dz):
    dx = self.y * dz # {dz/dx = dL/dz}
    dy = self.x * dz # {dz/dy = dL/dz}
    return [dx, dy]
    
```

(x,y,z are scalars)



Full implementation of training a 2-layer Neural Network needs ~11 lines:

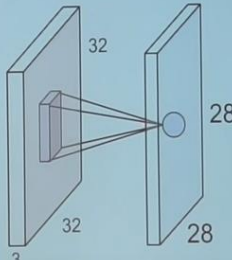
```

01. X = np.array([[0,0,1],[0,1,1],[1,0,1],[1,1,1]])
02. y = np.array([[0,1,0]])
03. syn0 = 2*np.random.random((3,4)) - 1
04. syn1 = 2*np.random.random((4,1)) - 1
05. for j in xrange(60000):
06.     l1 = 1/(1+np.exp(-(np.dot(X,syn0))))
07.     l2 = 1/(1+np.exp(-(np.dot(l1,syn1))))
08.     l2_delta = (y - l2)*(l2*(1-l2))
09.     l1_delta = l2_delta.dot(syn1.T) * (l1 * (1-l1))
10.     syn1 += l1.T.dot(l2_delta)
11.     syn0 += X.T.dot(l1_delta)
    
```

from @iamtrask, <http://iamtrask.github.io/2015/07/12/basic-python-network/>

Fei-Fei Li & Andrej Karpathy & Justin Johnson Lecture 4 - 67 13 Jan 2016

The brain/neuron view of CONV Layer



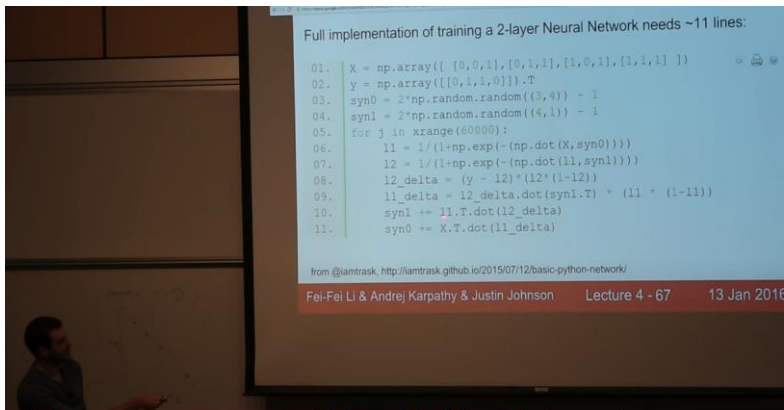
An activation map is a 28x28 sheet of neuron outputs:

1. Each is connected to a small region in the input
2. All of them share parameters

"5x5 filter" -> "5x5 receptive field for each neuron"

Core ML/DL concepts

- YouTube: cs231n 2016 winter by Andrej Karpathy



```
1 import tensorflow as tf
2
3 input_layer = tf.keras.layers.Input(shape=(28,28))
4 hidden_layer = tf.keras.layers.Dense(units=32, activation='relu')(input_layer)
5 output_layer = tf.keras.layers.Dense(units=2, activation='softmax')(hidden_layer)
6
7 model = tf.keras.Model(inputs=input_layer, outputs=output_layer)
8 model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
9 model.fit(x_train, y_train, epochs=10, batch_size=4)
```

Tools and frameworks

- Pytorch
- Tensorflow
- Keras

- Pandas
- NumPy
- Matplotlib
- OpenCV
- ROS

- Git/GitLab
- Linux

The screenshot displays the Keras website. At the top left is the Keras logo (a red square with a white 'K') and the word 'Keras'. Below it is a GitHub Star button showing 57,122 stars. A navigation menu on the left includes links for 'About Keras', 'Getting started' (which is highlighted), 'Developer guides', 'Keras API reference', 'Code examples', 'Why choose Keras?', 'Community & governance', 'Contributing to Keras', 'KerasTuner', 'KerasCV', and 'KerasNLP'. The main content area is titled 'Getting started' and contains three paragraphs of introductory text. The first paragraph is for engineers, the second for researchers, and the third for beginners. To the right of the main content is a sidebar with a 'Getting started' section containing links for 'Further starter resources' and 'Installing Keras'. At the bottom of the main content area, there is a section titled 'Installing Keras' with a code block showing the import statement.

Keras

Star 57,122

About Keras

Getting started

Introduction to Keras for engineers

Introduction to Keras for researchers

The Keras ecosystem

Learning resources

Frequently Asked Questions

Developer guides

Keras API reference

Code examples

Why choose Keras?

Community & governance

Contributing to Keras

KerasTuner

KerasCV

KerasNLP

Search Keras documentation...

» Getting started

Getting started

Are you an engineer or data scientist? Do you ship reliable and performant applied machine learning solutions? Check out our [Introduction to Keras for engineers](#).

Are you a machine learning researcher? Do you publish at NeurIPS and push the state-of-the-art in CV and NLP? Check out our [Introduction to Keras for researchers](#).

Are you a beginner looking for both an introduction to machine learning and an Introduction to Keras and TensorFlow? You're going to need more than a one-pager. And you're in luck: **we've got just the book for you.**

Further starter resources

- The Keras ecosystem
- Learning resources
- Frequently Asked Questions

Installing Keras

To use Keras, will need to have the TensorFlow package installed. [See detailed instructions](#).

Once TensorFlow is installed, just import Keras via:

```
from tensorflow import keras
```

Getting started

- ◆ Further starter resources
- ◆ Installing Keras

Tools and frameworks

- Pytorch
- Tensorflow
- Keras

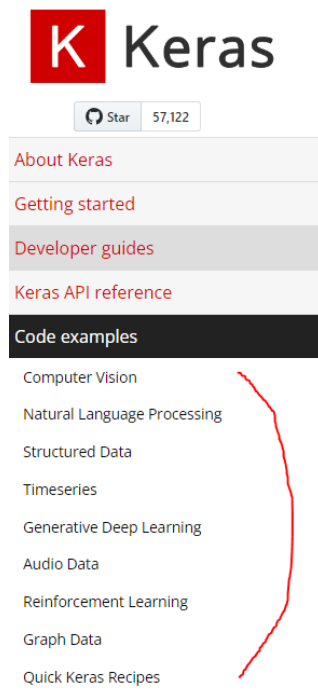
- Pandas
- NumPy
- Matplotlib
- OpenCV
- ROS

- Git/GitLab
- Linux

The screenshot displays the Keras website's API reference page. On the left, a sidebar contains a list of navigation links: 'About Keras', 'Getting started', 'Developer guides', 'Keras API reference' (highlighted), 'Models API', 'Layers API', 'Callbacks API', 'Optimizers', 'Metrics', 'Losses', 'Data loading', 'Built-in small datasets', 'Keras Applications', 'Mixed precision', 'Utilities', 'KerasTuner', 'KerasCV', 'KerasNLP', 'Code examples', 'Why choose Keras?', and 'Community & governance'. A red arrow originates from the 'Keras API reference' link in the sidebar and points to the main content area. The main content area features a search bar at the top, followed by the heading 'Keras API reference'. Below this, there are three sections: 'Models API' (listing 'The Model class', 'The Sequential class', 'Model training APIs', and 'Model saving & serialization APIs'), 'Layers API' (listing various layer types like 'The base Layer class', 'Layer activations', 'Layer weight initializers', etc.), and 'Callbacks API' (listing 'Base Callback class', 'ModelCheckpoint', 'BackupAndRestore', 'TensorBoard', and 'EarlyStopping').

Tools and frameworks

- Pytorch
 - Tensorflow
 - Keras
-
- Pandas
 - NumPy
 - Matplotlib
 - OpenCV
 - ROS
-
- Git/GitLab
 - Linux



Search Keras documentation...

» Code examples

Code examples

Our code examples are short (less than 300 lines of code), focused demonstrations of vertical deep learning workflows.

All of our examples are written as Jupyter notebooks and can be run in one click in [Google Colab](#), a hosted notebook environment that requires no setup and runs in the cloud. Google Colab includes GPU and TPU runtimes.

★ = Good starter example

Computer Vision

Image classification

- ★ [Image classification from scratch](#)
- ★ [Simple MNIST convnet](#)
- ★ [Image classification via fine-tuning with EfficientNet](#)

[Image classification with Vision Transformer](#)

[Image Classification using BigTransfer \(BiT\)](#)

[Classification using Attention-based Deep Multiple Instance Learning](#)

Tools and frameworks

- Pytorch
 - Tensorflow
 - Keras
-
- Pandas
 - NumPy
 - Matplotlib
 - OpenCV
 - ROS
-
- Git/GitLab
 - Linux

Developer guides
Keras API reference
Code examples
Computer Vision
Image classification from scratch
Simple MNIST convnet
Image classification via fine-tuning with EfficientNet
Image classification with Vision Transformer
Image Classification using BigTransfer (BIT)
Classification using Attention-based Deep Multiple Instance Learning
Image classification with modern MLP models
A mobile-friendly Transformer-based model for image classification
Pneumonia Classification on TPU
Compact Convolutional Transformers
Image classification with ConvMixer
Image classification with EAnet (External Attention Transformer)
Involutional neural networks
Image classification with Perceiver
Few-Shot learning with Reptile
Semi-supervised image classification using contrastive pretraining with SimCLR
Image classification with Swin Transformers

Date created: 2015/06/19

Last modified: 2020/04/21

Description: A simple convnet that achieves ~99% test accuracy on MNIST.

[View in Colab](#) · [GitHub source](#)

Setup

```
import numpy as np
from tensorflow import keras
from tensorflow.keras import layers
```

Prepare the data

```
# Model / data parameters
num_classes = 10
input_shape = (28, 28, 1)

# Load the data and split it between train and test sets
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()

# Scale images to the [0, 1] range
x_train = x_train.astype("float32") / 255
x_test = x_test.astype("float32") / 255
# Make sure images have shape (28, 28, 1)
x_train = np.expand_dims(x_train, -1)
x_test = np.expand_dims(x_test, -1)
print("x_train shape:", x_train.shape)
print(x_train.shape[0], "train samples")
print(x_test.shape[0], "test samples")

# convert class vectors to binary class matrices
y_train = keras.utils.to_categorical(y_train, num_classes)
y_test = keras.utils.to_categorical(y_test, num_classes)
```

```
x_train shape: (60000, 28, 28, 1)
```

Tools and frameworks

- Pytorch
- Tensorflow
- Keras

- Pandas
- NumPy
- Matplotlib
- OpenCV
- ROS

- Git/GitLab
- Linux

Image segmentation with a U-Net-like architecture

Multiclass semantic segmentation using DeepLabV3+

Object Detection with RetinaNet

Keypoint Detection with Transfer Learning

Object detection with Vision Transformers

3D image classification from CT scans

Monocular depth estimation

3D volumetric rendering with NeRF

Point cloud classification

OCR model for reading Captchas

Handwriting recognition

Convolutional autoencoder for image denoising

Low-light image enhancement using MIRNet

Image Super-Resolution using an Efficient Sub-Pixel CNN

Enhanced Deep Residual Networks for single-image super-resolution

Zero-DCE for low-light image enhancement

CutMix data augmentation for image classification

MixUp augmentation for image classification

RandAugment for Image Classification for Improved Robustness

Build the model

```
model = keras.Sequential([
    keras.Input(shape=input_shape),
    layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
    layers.MaxPooling2D(pool_size=(2, 2)),
    layers.Flatten(),
    layers.Dropout(0.5),
    layers.Dense(num_classes, activation="softmax"),
])

model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
conv2d (Conv2D)	(None, 26, 26, 32)	320

max_pooling2d (MaxPooling2D)	(None, 13, 13, 32)	0

conv2d_1 (Conv2D)	(None, 11, 11, 64)	18496

max_pooling2d_1 (MaxPooling2D)	(None, 5, 5, 64)	0

flatten (Flatten)	(None, 1600)	0

dropout (Dropout)	(None, 1600)	0

dense (Dense)	(None, 10)	16010
=====		
Total params: 34,826		
Trainable params: 34,826		
Non-trainable params: 0		

Tools and frameworks

Train the model

```
batch_size = 128
epochs = 15

model.compile(loss="categorical_crossentropy", optimizer="adam", metrics=["accuracy"])

model.fit(x_train, y_train, batch_size=batch_size, epochs=epochs, validation_split=0.1)
```

```
Epoch 1/15
422/422 [=====] - 13s 29ms/step - loss: 0.7840 - accuracy: 0.7643 - val_
Epoch 2/15
422/422 [=====] - 13s 31ms/step - loss: 0.1199 - accuracy: 0.9639 - val_
Epoch 3/15
422/422 [=====] - 14s 33ms/step - loss: 0.0845 - accuracy: 0.9737 - val_
Epoch 4/15
422/422 [=====] - 14s 33ms/step - loss: 0.0762 - accuracy: 0.9756 - val_
Epoch 5/15
422/422 [=====] - 15s 35ms/step - loss: 0.0621 - accuracy: 0.9812 - val_
Epoch 6/15
422/422 [=====] - 17s 40ms/step - loss: 0.0547 - accuracy: 0.9825 - val_
Epoch 7/15
422/422 [=====] - 17s 41ms/step - loss: 0.0497 - accuracy: 0.9840 - val_
Epoch 8/15
422/422 [=====] - 16s 39ms/step - loss: 0.0443 - accuracy: 0.9862 - val_
Epoch 9/15
422/422 [=====] - 17s 39ms/step - loss: 0.0436 - accuracy: 0.9860 - val_
Epoch 10/15
422/422 [=====] - 16s 38ms/step - loss: 0.0407 - accuracy: 0.9865 - val_
Epoch 11/15
422/422 [=====] - 16s 37ms/step - loss: 0.0406 - accuracy: 0.9874 - val_
Epoch 12/15
237/422 [=====>.....] - ETA: 7s - loss: 0.0398 - accuracy: 0.9877
< [ ] >
```

Available optimizers

- SGD
- RMSprop
- Adam
- AdamW
- Adadelta
- Adagrad
- Adamax
- Adafactor
- Nadam
- Ftrl

Tools and frameworks

- Pytorch
- Tensorflow
- Keras

- Pandas
- NumPy
- Matplotlib
- OpenCV
- ROS

- Git/GitLab
- Linux

Evaluate the trained model

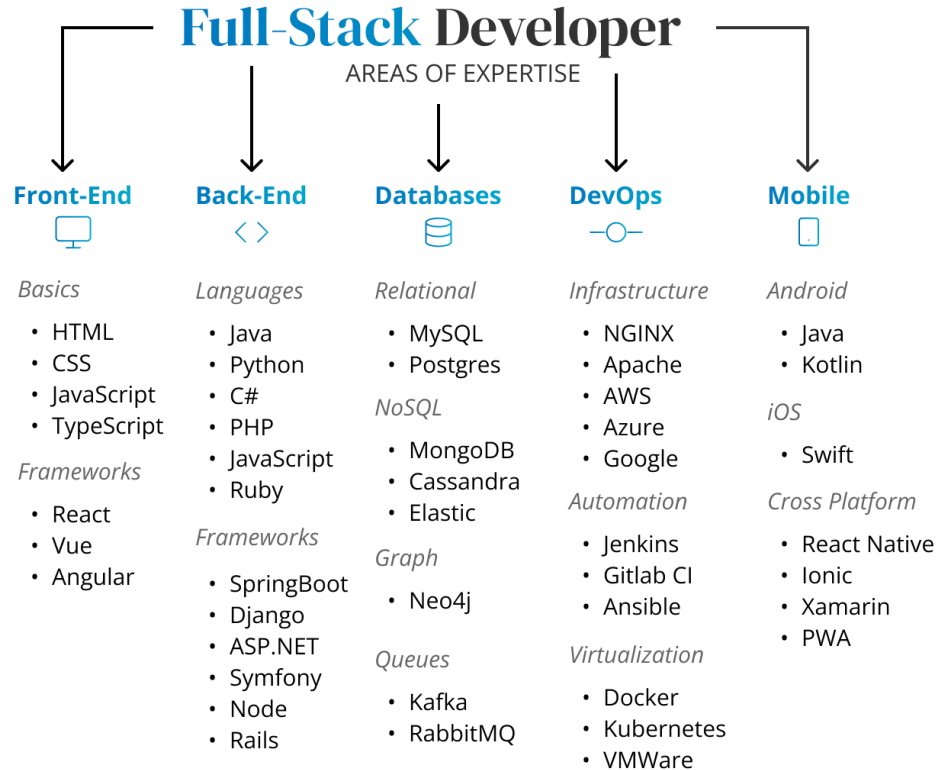
```
score = model.evaluate(x_test, y_test, verbose=0)
print("Test loss:", score[0])
print("Test accuracy:", score[1])
```

```
Test loss: 0.023950600996613503
Test accuracy: 0.9922000169754028
```

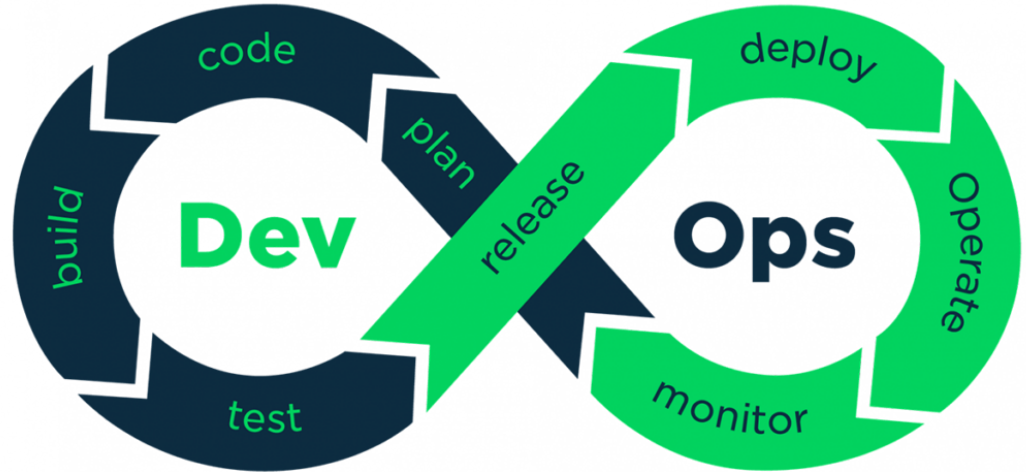
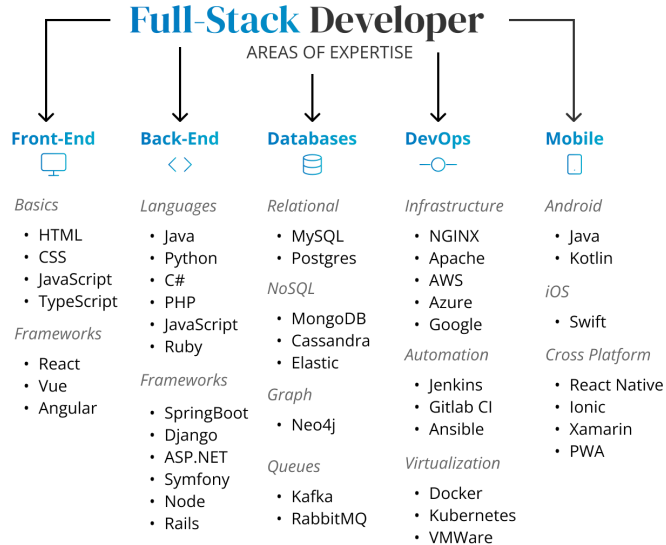
Different levels of engineers

- **Junior engineer/developer:** great coder, knows fundamentals, returns what is asked
- **Medior/middle:** junior + good understanding of the software development process, and is able to work independently
- **Senior:** Middle + birds-eye view skills, deeper technical skillset, knows about team roles and processes, system design expert
- **Lead:** Senior + management

Different levels of engineers




Different levels of engineers



Different levels of engineers



Different levels of engineers

 Vision

- Project information
- Repository
- Files**
- Commits
- Branches
- Tags
- Contributors
- Graph
- Compare
- Issues 0
- Merge requests 0
- CI/CD**
- Security & Compliance
- Deployments
- Monitor
- Infrastructure
- Packages & Registries
- Analytics
- Wiki
- Snippets
- Settings

<< Collapse sidebar

Pythonistas >  Vision > Repository

artifact_update

vision / backend /

+ ▾

History

Find file

Web IDE

↓

Clone ▾



box scaling fixed

Bunyod Ibrokhimov authored 1 week ago

72d1dabc



Name	Last commit	Last update
..		
arc_facerec	face not fully shown bug fixed	1 week ago
eye_detector	remove old unuseful models	2 months ago
facenet_pytorch	face not fully shown bug fixed	1 week ago
fasnet_liveness	face not fully shown bug fixed	1 week ago
mtcnort	Master fix	3 months ago
nnetwork	yolo trash deleted, yolov5_2 added	1 month ago
scripts	--worker-class gthread --preload	1 month ago
vectors	Remove trash files	1 month ago
yolov5	box scaling fixed	1 week ago
.dockerignore	Face rec/dlib	2 years ago
.gitignore	Face rec/threshold	2 years ago
Dockerfile	Added yolo models and new rose_final_231 model	1 month ago
Dockerfile.own	Optimization	5 months ago
Dockerfile.stag	Optimization	5 months ago

Different levels of engineers

Recognition

Liveness

Pictures:

Launch Camera

Upload Files

1 files selected

☒ 75 blurriness threshold: 75

☐ single face only

☒ open eyes only

☐ color image only

☐ head rotation

☐ blink threshold

rose_full_374

rose_epoch_231

final_rose_231

yolov5_1

yolov5_2

yolov5_3

yolov5_4

Check

Reset

Result:

Download

100006_camera-photos-bucket-14c13c5e043a4eacab9802e67afa3f4a.png

BBNet_v3	rose_full_374	yolov5_1
Успешно	Успешно	Успешно

Tips to become a pro

Road to become a senior engineer:

- Write well-design code (clean code: maximize cohesion, minimize coupling)
- Learn dependency injection and abstraction
- Document what you learn
- Always have side projects

Tips to become a pro

- OOP – object-oriented programming
 - Inheritance, abstraction, protocols
- Functional programming
- Optimization (make working codes more efficient, e.g., space, time)
- Modular programming
- SOLID principles
- Test-driven development

Tips to become a pro

Road to become a senior engineer:

- Write well-design code (clean code: maximize cohesion, minimize coupling)
- Learn dependency injection and abstraction
- Document what you learn
- Always have side projects

SOLID principle (Uncle Bob principle)

SOLID

From Wikipedia, the free encyclopedia

This article is about the SOLID principles of object-oriented programming. For the fundamental state of matter, see [Solid](#). For other uses, see [Solid \(disambiguation\)](#).

In [software engineering](#), **SOLID** is a [mnemonic acronym](#) for five design principles intended to make [object-oriented](#) designs more understandable, flexible, and [maintainable](#). The principles are a subset of many principles promoted by American software engineer and instructor [Robert C. Martin](#),^{[1][2][3]} first introduced in his 2000 paper *Design Principles and Design Patterns* discussing [software](#) [rot](#).^{[2][4]:2–3}

SOLID
Principles
Single responsibility
Open–closed
Liskov substitution
Interface segregation
Dependency inversion

V · T · E

The SOLID ideas are

- The [Single-responsibility principle](#): "There should never be more than one reason for a [class](#) to change."^[5] In other words, every class should have only one responsibility.^[6]
- The [Open–closed principle](#): "Software entities ... should be open for extension, but closed for modification."^[7]
- The [Liskov substitution principle](#): "Functions that use pointers or references to base classes must be able to use objects of derived classes without knowing it."^[8] See also [design by contract](#).^[8]
- The [Interface segregation principle](#): "Clients should not be forced to depend upon interfaces that they do not use."^{[9][4]}
- The [Dependency inversion principle](#): "Depend upon abstractions, [not] concretions."^{[10][4]}

The SOLID acronym was introduced later, around 2004, by Michael Feathers.^[11]

Although the SOLID principles apply to any object-oriented design, they can also form a core philosophy for methodologies such as [agile development](#) or [adaptive software development](#).^[3]

```

# origin.py > ...
1 class Order:
2     items = []
3     quantities = []
4     prices = []
5     status = 'open'
6
7     def add_item(self, name, quantity, price):
8         self.items.append(name)
9         self.quantities.append(quantity)
10        self.prices.append(price)
11
12    def total_price(self):
13        total = 0
14        for i in range(len(self.prices)):
15            total += self.quantities[i] * self.prices[i]
16        return total
17
18    def pay(self, payment_type, security_code):
19        if payment_type == 'debit':
20            print('Processing debit payment type')
21            print(f'Verifying security code: {security_code}')
22            self.status = 'paid'
23        elif payment_type == 'credit':
24            print('Processing credit payment type')
25            print(f'Verifying security code: {security_code}')
26            self.status = 'paid'
27        else:
28            raise Exception(f"Unknown payment type: {payment_type}")
29
30    order = Order()
31    order.add_item("iPhone 12 Pro", 1, 1290)
32    order.add_item("Macbook Air", 1, 1100)
33
34    print("Total price: ", order.total_price())
35    order.pay('debit', '03214')

```

Total price: 2390

Processing debit payment type

Verifying security code: 03214

The Single-responsibility principle:

There should never be more than one reason for a class to change.

In other words, every class should have only one responsibility.

In the previous code, Order class has more than one responsibility, lets fix that.

```

origin.py > ...
1 class Order:
2     items = []
3     quantities = []
4     prices = []
5     status = 'open'
6
7     def add_item(self, name, quantity, price):
8         self.items.append(name)
9         self.quantities.append(quantity)
10        self.prices.append(price)
11
12    def total_price(self):
13        total = 0
14        for i in range(len(self.prices)):
15            total += self.quantities[i] * self.prices[i]
16        return total
17
18    def pay(self, payment_type, security_code):
19        if payment_type == 'debit':
20            print('Processing debit payment type')
21            print(f'Verifying security code: {security_code}')
22            self.status = 'paid'
23        elif payment_type == 'credit':
24            print('Processing credit payment type')
25            print(f'Verifying security code: {security_code}')
26            self.status = 'paid'
27        else:
28            raise Exception(f"Unknown payment type: {payment_type}")
29
30    order = Order()
31    order.add_item("iPhone 12 Pro", 1, 1290)
32    order.add_item("Macbook Air", 1, 1100)
33
34    print("Total price: ", order.total_price())
35    order.pay('debit', '03214')

```

```

7 class Order:
8     items = []
9     quantities = []
10    prices = []
11    status = 'open'
12
13    def add_item(self, name, quantity, price):
14        self.items.append(name)
15        self.quantities.append(quantity)
16        self.prices.append(price)
17
18    def total_price(self):
19        total = 0
20        for i in range(len(self.prices)):
21            total += self.quantities[i] * self.prices[i]
22        return total
23
24    class PaymentProcessor:
25        def debit_pay(self, order, security_code):
26            print('Processing debit payment type')
27            print(f'Verifying security code: {security_code}')
28            order.status = 'paid'
29
30        def credit_pay(self, order, security_code):
31            print('Processing credit payment type')
32            print(f'Verifying security code: {security_code}')
33            order.status = 'paid'
34
35
36    order = Order()
37    order.add_item("iPhone 12 Pro", 1, 1290)
38    order.add_item("Macbook Air", 1, 1100)
39
40    payment = PaymentProcessor()
41    print("Total price: ", order.total_price())
42    payment.credit_pay(order, '03214')
43

```

```

7 class Order:
8     items = []
9     quantities = []
10    prices = []
11    status = 'open'
12
13    def add_item(self, name, quantity, price):
14        self.items.append(name)
15        self.quantities.append(quantity)
16        self.prices.append(price)
17
18    def total_price(self):
19        total = 0
20        for i in range(len(self.prices)):
21            total += self.quantities[i] * self.prices[i]
22        return total
23
24    class PaymentProcessor:
25        def debit_pay(self, order, security_code):
26            print('Processing debit payment type')
27            print(f'Verifying security code: {security_code}')
28            order.status = 'paid'
29
30        def credit_pay(self, order, security_code):
31            print('Processing credit payment type')
32            print(f'Verifying security code: {security_code}')
33            order.status = 'paid'
34
35
36    order = Order()
37    order.add_item("iPhone 12 Pro", 1, 1290)
38    order.add_item("Macbook Air", 1, 1100)
39
40    payment = PaymentProcessor()
41    print("Total price: ", order.total_price())
42    payment.credit_pay(order, '03214')
43

```

The Open-closed principle
 # Software entities ... should be open for extension
 but closed for modification.

In the previous code we cannot add new payment
 methods without changing the class. Let's fix
 that...

```

7 class Order:
8     items = []
9     quantities = []
10    prices = []
11    status = 'open'
12
13    def add_item(self, name, quantity, price):
14        self.items.append(name)
15        self.quantities.append(quantity)
16        self.prices.append(price)
17
18    def total_price(self):
19        total = 0
20        for i in range(len(self.prices)):
21            total += self.quantities[i] * self.prices[i]
22        return total
23
24    class PaymentProcessor:
25        def debit_pay(self, order, security_code):
26            print('Processing debit payment type')
27            print(f'Verifying security code: {security_code}')
28            order.status = 'paid'
29
30        def credit_pay(self, order, security_code):
31            print('Processing credit payment type')
32            print(f'Verifying security code: {security_code}')
33            order.status = 'paid'
34
35
36    order = Order()
37    order.add_item("iPhone 12 Pro", 1, 1290)
38    order.add_item("Macbook Air", 1, 1100)
39
40    payment = PaymentProcessor()
41    print("Total price: ", order.total_price())
42    payment.credit_pay(order, '03214')
43

```

6 from abc import ABC, abstractmethod

```

25 class PaymentProcessor(ABC):
26     @abstractmethod
27     def pay(self, order, security_code):
28         pass
29
30 class DebitPaymentProcessor:
31     def pay(self, order, security_code):
32         print('Processing debit payment type')
33         print(f'Verifying security code: {security_code}')
34         order.status = 'paid'
35
36 class CreditPaymentProcessor:
37     def pay(self, order, security_code):
38         print('Processing credit payment type')
39         print(f'Verifying security code: {security_code}')
40         order.status = 'paid'
41
42 class PaypalPaymentProcessor:
43     def pay(self, order, security_code):
44         print('Processing paypal payment type')
45         print(f'Verifying security code: {security_code}')
46         order.status = 'paid'
47
48 order = Order()
49 order.add_item("iPhone 12 Pro", 1, 1290)
50 order.add_item("Macbook Air", 1, 1100)
51
52 payment = PaypalPaymentProcessor()
53 print("Total price:", order.total_price())
54 payment.pay(order, '03214')

```

```
6 from abc import ABC, abstractmethod
```

```
25 class PaymentProcessor(ABC):
26     @abstractmethod
27     def pay(self, order, security_code):
28         pass
29
30 class DebitPaymentProcessor:
31     def pay(self, order, security_code):
32         print('Processing debit payment type')
33         print(f'Verifying security code: {security_code}')
34         order.status = 'paid'
35
36 class CreditPaymentProcessor:
37     def pay(self, order, security_code):
38         print('Processing credit payment type')
39         print(f'Verifying security code: {security_code}')
40         order.status = 'paid'
41
42 class PaypalPaymentProcessor:
43     def pay(self, order, security_code):
44         print('Processing paypal payment type')
45         print(f'Verifying security code: {security_code}')
46         order.status = 'paid'
47
48 order = Order()
49 order.add_item("iPhone 12 Pro", 1, 1290)
50 order.add_item("Macbook Air", 1, 1100)
51
52 payment = PaypalPaymentProcessor()
53 print("Total price:", order.total_price())
54 payment.pay(order, '03214')
```

The Liskov substitution principle:
Functions that use pointers or references to base classes must be able to use objects of derived classes without knowing it.

In the previous code, if we want to change security_code to email_address for paypal payment method, we are abusing PaymentProcessor class for something it is not supposed to do. And this violates Liskov substitution principle. Let's fix that...

6 from abc import ABC, abstractmethod

```
25 class PaymentProcessor(ABC):
26     @abstractmethod
27     def pay(self, order, security_code):
28         pass
29
30 class DebitPaymentProcessor:
31     def pay(self, order, security_code):
32         print('Processing debit payment type')
33         print(f'Verifying security code: {security_code}')
34         order.status = 'paid'
35
36 class CreditPaymentProcessor:
37     def pay(self, order, security_code):
38         print('Processing credit payment type')
39         print(f'Verifying security code: {security_code}')
40         order.status = 'paid'
41
42 class PaypalPaymentProcessor:
43     def pay(self, order, security_code):
44         print('Processing paypal payment type')
45         print(f'Verifying security code: {security_code}')
46         order.status = 'paid'
47
48 order = Order()
49 order.add_item("iPhone 12 Pro", 1, 1290)
50 order.add_item("Macbook Air", 1, 1100)
51
52 payment = PaypalPaymentProcessor()
53 print("Total price:", order.total_price())
54 payment.pay(order, '03214')
```

```
32 class DebitPaymentProcessor(PaymentProcessor):
33     def __init__(self, security_code):
34         self.security_code = security_code
35
36     def pay(self, order):
37         print('Processing debit payment type')
38         print(f'Verifying security code: {self.security_code}')
39         order.status = 'paid'
40
41 class CreditPaymentProcessor(PaymentProcessor):
42     def __init__(self, security_code):
43         self.security_code = security_code
44
45     def pay(self, order):
46         print('Processing credit payment type')
47         print(f'Verifying security code: {self.security_code}')
48         order.status = 'paid'
49
50 class PaypalPaymentProcessor(PaymentProcessor):
51     def __init__(self, email_address):
52         self.email_address = email_address
53
54     def pay(self, order):
55         print('Processing paypal payment type')
56         print(f'Verifying email address: {self.email_address}')
57         order.status = 'paid'
58
59 order = Order()
60 order.add_item("iPhone 12 Pro", 1, 1290)
61 order.add_item("Macbook Air", 1, 1100)
62
63 payment = CreditPaymentProcessor("04132")
64 print("Total price:", order.total_price())
65 payment.pay(order)
```

Homework: finish I and D

```
# The Interface segregation principle:  
# Clients should not be forced to depend  
upon interfaces that they do not use.  
  
# In the previous code, we are abusing  
credit card method to use sms  
authenticate,  
# And this violates Interface segregation  
principle. Let's fix that...
```

```
# The Dependency inversion principle:  
# Depend upon abstractions, [not] concretions.  
  
# In the previous code, if we want to add new  
authentication method, we have to change SMSAuth  
class because it has dependency  
# Instead we can create new authorizer class so  
that we can easily add new auth types in future.  
Let's fix that...
```

Agenda

01

What is
software
engineering?

02

Where to
start:
A Roadmap

03

Different
levels of
software
engineer

04

Tips to
become a
pro

Thank You

Q & A