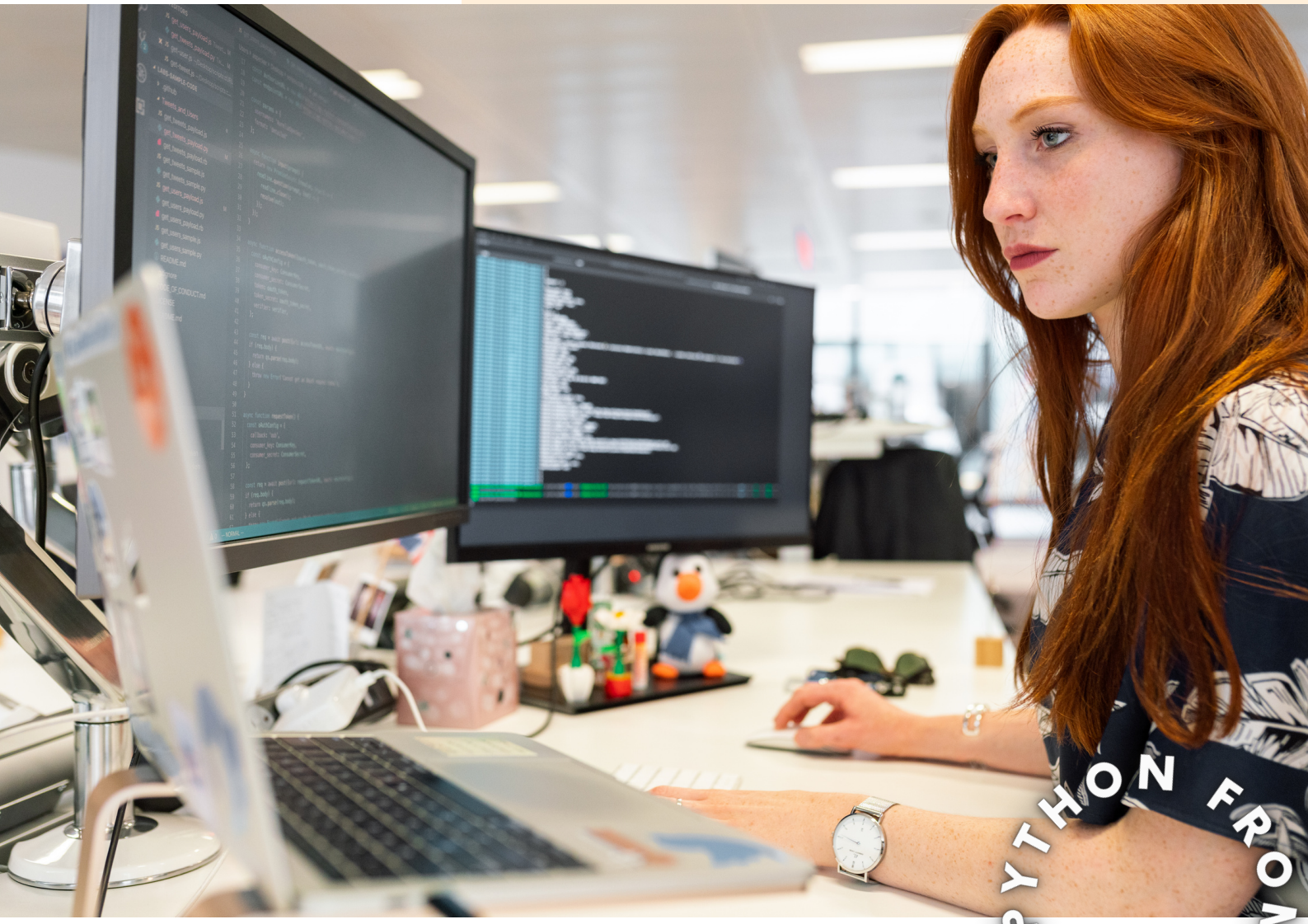
LPWR ACADEMY



PYTHON 101
A CRASH COURSE

Python Quickly and Efficiently

What Makes This Python Crash Course Different?

Did you know that most people don't understand what programming is?

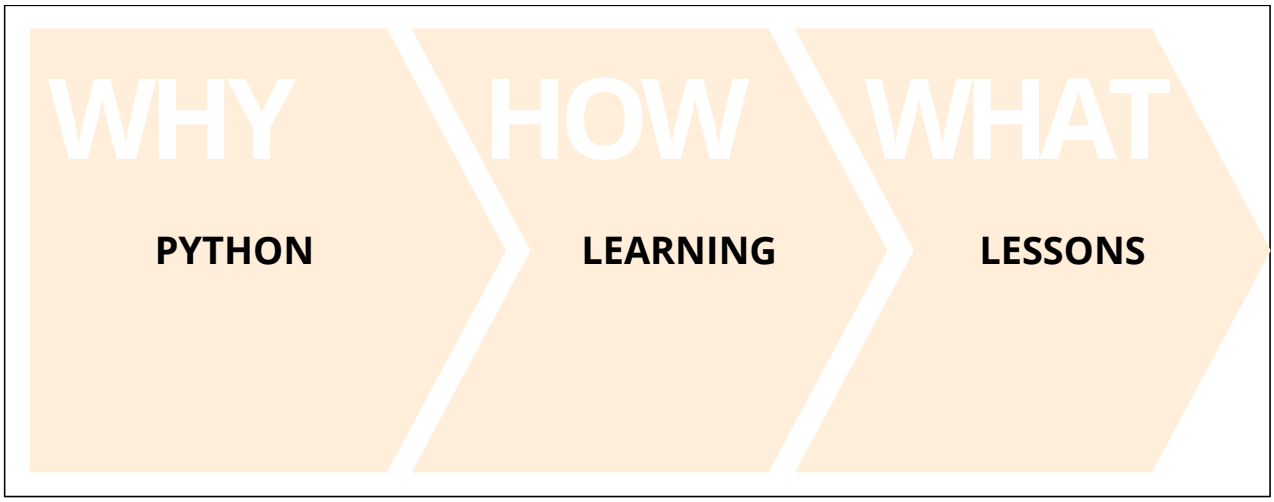
Because of this, many people struggle to learn about programming and spend too much time learning insignificant aspects of it.

A senior programmer knows that code is written once but read hundreds of times.

How does that change their approach to programming?

Well, a programmer knows that it's about creating the most efficient, unintuitive, and complex code constructions. They know that code is about **readability**.

Code should be intuitive and straightforward to understand.



This e-book will cover

- Why Python is a good choice for beginners and scientific studies.
- How the learning approach is tailored for the best learning experience.
- Finally, the 17 lessons with a getting started guide.

Python is ideal for programmers because the syntax is minimal and constructed with a readability as a core focus point. In addition, Python can be used for simple scripts and full-scale object-oriented programming.

Simplicity is the key to Python's success and popularity. Most people do not understand this aspect of Python and, more importantly, programming.

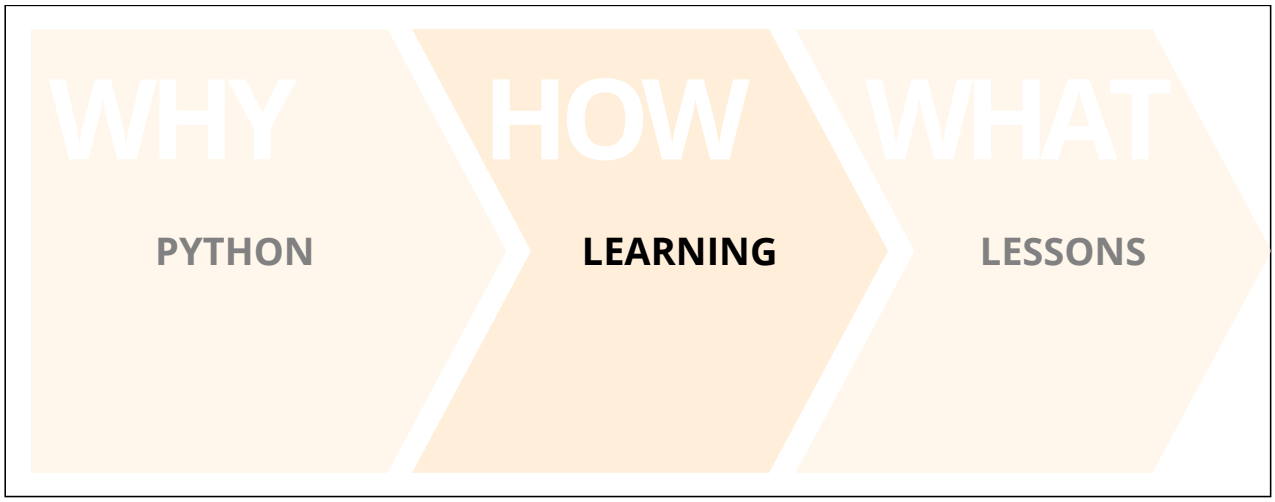
This means that many courses teach Python and programming incorrectly, making it too complex, covering too many topics, and teaching edge-case optimizations.

Senior programmers make their code easy to understand and read.

Why do most teachers not understand this?

Well, because they teach coding and have little to no experience with production code. Their code is written once and then never seen again. They have an academic and not practical approach to coding.

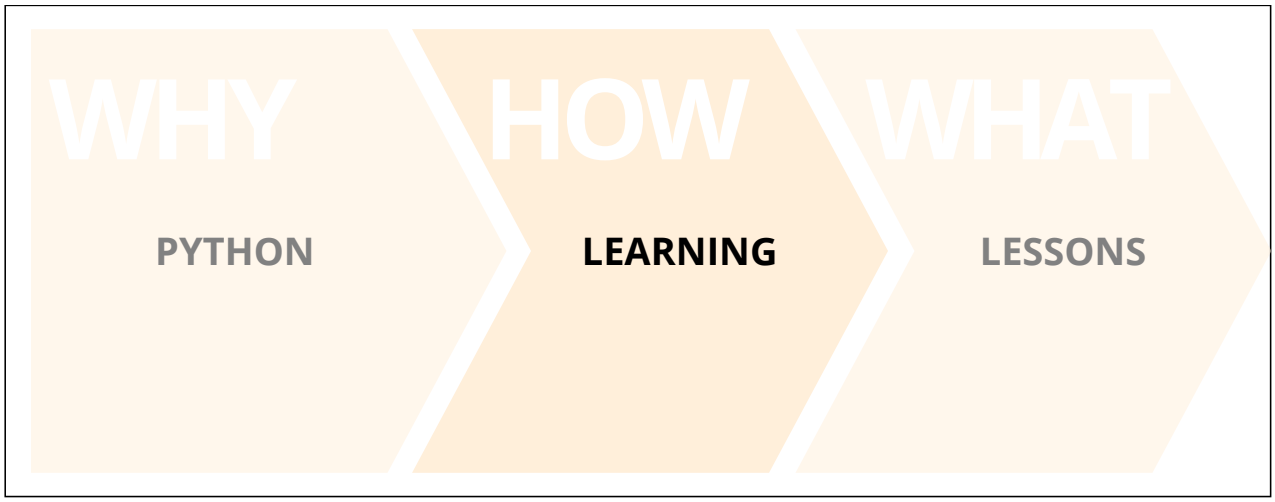
Hello! I am Rune from Denmark, and I have been programming professionally for over 15 years. In addition, I have had a passion for teaching since I began studying at the university for my Ph.D.



Crash Course Design

This course is designed as follows to keep it simple, intuitive, and as coding is intended.

- **One concept at a time** Each lesson will only focus on one new concept at a time to avoid overloading information.
- **Simple explanations** The explanations will relate to things you already know – it will make it easier for you to learn to program.
- **Use-case focused** We will cover the most important use cases, which will allow you to unlock the full power of programming. This will help you stay focused on what is relevant to successful programming.
- **Project-based** Each concept will have a project you will try to make on your own because that's how you learn to program. But do not worry, I present a solution with an explanation for each project.
- **Resource Options** Each lesson has video explanations with code-along instructions, notebooks with the code, and an eBook chapter with a written explanation. This will help you review the concepts in 3 different ways if necessary.



Crash Course Structure

To optimize your learning the course is structured into 17 lessons. It will start from zero helping you to set up your environment with Jupyter Notebook (Python environment) and run your first program.

In your final project, you will create a full Machine Learning model from scratch.

All 17 video lessons include the following.

- An introduction to new Python and programming **concepts**.
- Prepared **Jupyter Notebooks** you can follow along with.
- A **project** is introduced in the video lecture along with a Jupyter Notebook.
- At the end of the video lesson, a **solution to the projec**t is given.
- Each lesson is covered in the **eBook** for easy access.

The best way to learn Python is to program projects tailored to your level. This course comes with 17 projects. One project for each lesson and a new concept learned.

Crash Course Learnings

After this course, you will have achieved the following

- Created **17 projects** including Machine Learning.
- Understand and use **variables**.
- Use **lists** and **dictionaries**.
- Program flow with **if-statements** and **loops**.
- How to create **functions**.
- Using **randomness** in a program.
- Created simple **games**.
- Read and process **CSV files**.
- **Object-Oriented Programming** (OOP).
- **Visualization** of data.
- Created projects with **NumPy** and **Pandas**.
- Created a **Machine Learning** model from scratch.
- **Recursive functions**.
- **List comprehension.**

Setup and Installation

You will use Jupyter Notebooks for your learning.

In the first lesson you will be introduced to them, for now, you need to install Anaconda.

1. Go to **Anaconda** and download the individual edition.
2. It will install **Python** and **Jupyter** notebook. This is all you need to get started and it is free.
3. Launch Anaconda.
    - On Windows, you may need to add **conda** in **User Variable**'s path.
4. In Anaconda launch Jupyter Notebook.
5. Navigate Jupyter Notebook to the downloaded Notebooks from the link (button) above.
    - Alternatively, you can import them.

On the next page, you will download the Notebooks with all the lessons and projects you will do in this course.

Notebooks

IMPORTANT You need to download all the Notebooks and course material to follow along.

CLICK HERE TO DOWNLOAD ZIP FILE

The above link will download a ZIP file with all the Notebooks and files.

- Unzip the file (main.zip)
- You find the first Notebook under
  - Jupyter -> starter
- You find the Notebooks from the recordings under
  - jupyter -> final

Alternatively, you can go directly to my GitHub and get the files.

- https://github.com/LearnPythonWithRune/LearnPython

LESSON

00

WHY PYTHON | HOW LEARNING | WHAT LESSONS

Learn Python

Learn Python for Beginner a 8H full course in 17 lessons. Each lesson will introduce concepts and a project with a solution. This course comes with 34 Jupyter Notebooks, an installation guide, and basically everything you need to get started.

Do you want to Learn Python? Don't know how to get started? Not sure programming is for you? Don't know why you should start with Python?

The goal of this session is how to get started learning Python. It will show you how to get started with Anaconda Python.

This will take you on a learning journey with Python. It will show you what aspects of programming are ideal with Python. Learn everything you need to get started with development, Data Science, Machine Learning, Object-Oriented Programming, Creating Fun Games, NumPy and Pandas, Excel Sheet Automation, and much more...

In this first lesson, we will learn to take input from the user and print it. Also, how to apply string methods that convert the string to upper case or similar.

LESSON 01

WHY PYTHON | HOW LEARNING | WHAT LESSONS

Variables and Types in Python

In this section, you will learn about variables and types in Python.

Variable and types are needed to make program operations, like adding numbers. This will enable you to understand how programs execute operations on data.

Variables and types are the foundation of programming. Without an understanding of them, it is difficult to start programming. A variable has a type, in Python, among them are integer, float, string, list, and dictionaries.

Learn why types in Python are important to master. They enable you to use specific functions, like Math functions, manipulate specific string operations, and more.

In the project, we will learn how to make calculations assigning values to variables. The variables will have types and we will see how the types are important to master.

```
1 type(b)
```

LESSON 02

WHY PYTHON | HOW LEARNING | WHAT LESSONS

Type Conversion in Python

Learn how to convert between types in Python.

Type Conversion in Python is important as input to functions needs to be correct.

The goal of this section is to take user input and convert it to an integer or float. It will also teach you how to convert a Python integer (int) to a string.

In the project, we will make interactive calculations. We will prompt the user for input, and convert the input to an integer or float before further computations are done. This will demonstrate the use of type conversion.

```
1 name = input("What is your name? ")
2 print(f"Hello {name}!")
3 birth_year = input("What is your birht year? ")
4 print(f"You were born in {birth_year}.")
```

LESSON 03

WHY PYTHON | HOW LEARNING | WHAT LESSONS

Conditional Flow in Python

Learn to use conditional flows in your Python program. This enables you to let the program perform different tasks based on input and calculations.

The goal of this section is to teach you how to do if-else statements in Python and master conditional flow in Python. This will enable you to implement a decision tree in Python.

Does it rain? Yes, take an umbrella. No, no need for an umbrella. This is how you make decisions based on the context you are exposed to. Conditional flows in Python enable you to simulate the same behavior. This is done with if-else statements in Python. These can be extended to if-else or if-elif-else statements to make more advanced flows.

In the project, we will implement a decision tree for a car insurance company. This is a great thing to master as a programmer.

```
1 if a == b:
2     print("a equals b")
```

LESSON
04

WHY PYTHON | HOW LEARNING | WHAT LESSONS

Randomness and Simple Games

To make a Game fun you need to add randomness to it. Otherwise, it will be predictable. Learn how to use Python's random number generator.

The Python Random Module is important to master to make things interesting in many aspects. For one, when creating games. Learn how to use the Python random generator to create random integers, a random float, and more.

The goal of this lesson is to teach you how to use the Python random number generator.

In the project, you will create a Math game and the classical Rock-Scissor-Paper game.

```
1 import random
2 a = random.randint(1, 6)
3 b = random.uniform(0, 1)
4 c = random.gauss(0, 1)
```

LESSON 05

WHY PYTHON | HOW LEARNING | WHAT LESSONS

Python Lists and Jumbled Game

Python lists are the most important data structure to master in Python. Learn the most important Python list operations.

Python lists are versatile and easy to use. The Python list can be used for containing any Python types or objects.

The project is to create the classical Jumbled Game. This will use a Python list and demonstrate a use-case of them.

```
1 my_list = ['Apple', 'Orange', 'Banana']
2 another_list = ['France', 'Denmark', 'Sweden']
3 new_list = my_list + another_list
4 ' '.join(new_list)
```

```
1 import random
2 random.choice(new_list)
3 random.shuffle(new_list)
```

LESSON

06

WHY PYTHON | HOW LEARNING | WHAT LESSONS

For and While Loops in Python

The for and while loops help you repeat the same task again and again in Python. Learn how to make loops that terminate based on user input.

The difference between for and while loops can be a bit confusing at first. The for-loop is used when you know how many repetitions you have. The while loop is used when you do not know how many repetitions you have before you enter the loop.

The goal of this section is to teach you how to use Python for and while loops. The while loop is especially powerful when there is user input in form of a string like we will use in the project.

The project will be to implement the classical Hangman Game.

```
1 my_list = ['First', 'Second', 'Third']
2 for i in my_list:
3     print(i)
```

LESSON
07

WHY PYTHON | HOW LEARNING | WHAT LESSONS

Python Functions and Caesar Cipher

Learn how to structure your code with Python functions. How to master arguments and return statements with Python.

We will learn about Python functions in Jupyter notebook. How to call functions with arguments. Master how to return values from Python functions. Also, why and when to use functions.

The objective of this section is to learn about Python functions for beginners. It will contain clear examples and use cases of how and why to use functions in Python.

The project for this section will be to implement the Caesar Cipher. Both encryption and decryption. This will teach you how to use functions as well as how encryption with Caesar Cipher works.

```
1  def print_name(name):
2      print(f"Hello {name}")
3
4  print_name("Rune")
```

LESSON
08

WHY PYTHON | HOW LEARNING | WHAT LESSONS

Dictionaries and Game & Frequency Counting

Learn how to use Python Dictionaries with keys and values. This section will teach you how to use Python Dictionaries to count occurrences of unknown types.

Master one of the most important data structures in Python: The dictionary (dict). Learn how to add key and value pairs in a dictionary. How you can add new items to a dictionary. Also, learn how to iterate over a Python Dictionary with Key-Value pairs.

The project will be to implement a Guess a Capital Game and how to make frequency count using Python Dictionaries.

```
my_dict = {
    'Key 1': 'Value 1',
    'Key 2': 'Value 2'
}
```

LESSON 09

WHY PYTHON | HOW LEARNING | WHAT LESSONS

CSV files with DictReader

What is a CSV file? A great way to store tabular data.
How to read CSV files with Python? This is the focus of this lesson.

Learn how to read CSV files with DictReader in Python. This will read the CSV content into a Python list of dictionaries representing one row each with key-value pairs.

In this lesson, you will learn what a CSV file is, and how do you read a CSV file in a convenient way in Python. Also how to iterate over the CSV content with a for-loop in Python.

This can all be done simply by using the CSV library with the DictReader.

In the project, you will read a huge CSV file and make some nice data processing utilizing what you have just learned.

```
1  import csv
2  with open("files/NameRecords.csv", "r") as f:
3      csv_reader = csv.DictReader(f)
4      name_records = list(csv_reader)
```

LESSON 10

WHY PYTHON | HOW LEARNING | WHAT LESSONS

Recursive Functions & Tower of Hanoi

Learn what Recursive Functions are and why they are powerful. The classical example is solving the Tower of Hanoi, which solves a complex problem in a simple manner with recursion.

In this lesson, we will first learn what recursive functions are, and why they are useful and gives you great power. We will break recursive functions down with a simple counting function. Then learn what the Fibonacci numbers are and how to implement them recursively.

In the project, we will learn the complex mathematical game: Tower of Hanoi. It is a seemingly complex problem to solve, but with recursion it becomes easy. Tower of Hanoi is the best example to learn what recursion is and how it can help craft readable code.

```
1  def count(n):
2      if n <= 0:
3          return n
4      return n + count(n - 1)
```

LESSON 11

WHY PYTHON | HOW LEARNING | WHAT LESSONS

List Comprehension

List Comprehension in Python is something you need to master. When you know how to use them, you will write simpler code.

In this lesson, we will learn what List Comprehension is and how to use it. We will start off with a simple example and explore similar code without list comprehension. Then we will make list comprehension with calculations, with if-statements, if-else-statements, and list comprehension from another list. We will also learn how to make Dict Comprehension.

In the project, we will re-do some of our previous code with list comprehension. Specifically, we will make the Caesar Cipher encryption and decryption with List Comprehension. Also, we will make Frequency count with Dict Comprehension.

```
1 my_list = [i for i in range(10)]
2 my_list = [i for i in range(10) if i % 2 == 0]
3 my_list = [i if i % 2 else -i for i in range(10)]
4 {i: i*i for i in my_list}
```

LESSON 12

WHY PYTHON | HOW LEARNING | WHAT LESSONS

Object-Oriented Programming

Object-Oriented Programming (OOP) is the best tool to implement complex problems in a simple way.

In this lesson, we will introduce Object-Oriented Programming (OOP) with a concrete example of implementing a Card Game. This will teach you Python classes and objects. More precisely, we will implement three classes (Card, Deck, and Hand) from a description of a diagram. This will include implementing string representation of the objects, passing parameters to class instance variables, comparison of objects, and adding custom methods.

The project will be implementing a Card Game using the classes we will have implemented.

```
1 class Card:
2     def __init__(self, suit, rank):
3         self.suit = suit
4         self.rank = rank
```

LESSON 13

WHY PYTHON | HOW LEARNING | WHAT LESSONS

Matplotlib Visualization

When working with data you need to be able to visualize it. This is easy with the Python library Matplotlib.

In this lesson, we will learn how to visualize data using Matplotlib in Python. We will learn how to plot lines and make scatter plots and histograms. Also, we will learn how to use the Matplotlib library in both functional and object-oriented ways. This is important to understand to utilize the library efficiently. We will also learn how to set titles and labels and color scatter plots.

In the project, we will investigate if there is a connection between Horsepower and Torque in a long CSV file full of cars. This investigation will be done visually with Matplotlib in a color full scatterplot. Secondly, we will use a histogram to investigate the value of the roll of two dice.

```
1 import matplotlib.pyplot as plt
2 %matplotlib notebook
3 plt.plot([1, 2, 3, 4])
```

LESSON

14

WHY PYTHON | HOW LEARNING | WHAT LESSONS

Linear Regression with NumPy

We will learn about Machine Learning and use a Linear Regression model to fit it to a real-world dataset.

In this section, we will learn the difference between classical computing and Machine Learning. How Machine Learning works, as well as, the types of Machine Learning. We will get a brief introduction to the NumPy library, which is useful for Machine Learning. Finally, we will learn how to use a Linear Regression model from sklearn, to predict from a dataset.

In the project, we will use real-world data and see how well the Linear Regression model can predict a connection between Horsepower and Torque.

```
from sklearn.linear_model import LinearRegression
lin_regressor = LinearRegression()
lin_regressor.fit(X, Y)
Y_pred = lin_regressor.predict(X)
```

LESSON 15

WHY PYTHON | HOW LEARNING | WHAT LESSONS

Pandas and Excel Automation

Learn how to use pandas to create Excel sheets with charts – all from Python.

In this lesson, we will get a short introduction to pandas data structure DataFrames, which are very similar to Excel sheets. Among the things we will master with pandas, are filtering and GroupBy operations. We will learn how to read data properly from a CSV file into pandas DataFrame. From the DataFrame, we will export it to an Excel sheet and insert charts of the data.

In the project, we will read a sales dataset and make some nice Excel sheets with charts representing the sales reps.

```
d = {
  'Cars': ['Fiat', 'Porsche', 'Ferrari', 'Ferrari'],
  'Speed': [120, 215, 305, 195]
}
```

```
import pandas as pd
df = pd.DataFrame(d)
```

LESSON 16

WHY PYTHON | HOW LEARNING | WHAT LESSONS

Reinforcement Learning from Scratch

Create your own Machine Learning model from Scratch without using any libraries. That is, you will learn how the Reinforcement Learning Machine Learning algorithm works and create it yourself from scratch in Python.

In this last lesson, you will make your Cap Stone project. Creating your own Machine Learning model from scratch. First, we will learn what the Reinforcement model is, how it works, and a description of the algorithm.

Then we will look at the problem we want to solve and use Object-Oriented Programming (OOP) to create a Field class where our algorithm will exist in. The field will contain actions that the model can make, and reward it according to the behavior we want.

Then we will continue to create a naive solution to the problem, to see how well it performs. This will be a baseline for the implementation of our Reinforcement Learning algorithm.

We continue to implement a Q-table, the learning part of the algorithm. Finally, we will see how it performs compared to the naive solution. And I will promise you, you will be impressed by how little code is needed to make such an efficient algorithm.

Congratulations

You have created 17 projects with Python!

You know a lot about Python

- types and variables
- lists, dictionaries
- program flows
- randomness
- CSV files

...and much more.

But...

- **It is the first step**

Don't expect

- to write big programs
- not to need help

I started programming at 12, got a Ph.D. in computer science, coded professionally for almost 15 years, and I still need help and to learn new stuff all the time.

One step at a time... but enjoy the learning journey!

Rune