



1337AI

The Entry-Level ML Engineer Boot-camp!

Your First Steps in AI

Summary: The goal of the week is to get started with the Python language and data manipulation.

Chapter I

Introduction

Welcome to the Entry-Level ML Engineer Bootcamp! This bootcamp is designed as an entry point for anyone looking to start their journey in AI and machine learning. You'll learn the fundamentals of ML through structured modules, hands-on exercises, and curated resources.

The bootcamp is divided into four modules, with each module lasting one week, allowing you to complete the program in a month. Every week, you'll find exercises and resources to help you grasp key concepts. Additionally, we've included exploration questions to encourage deeper learning and creative thinking.

Our goal is that by the end of the bootcamp, participants will be able to collect and clean data, train machine learning models, and deploy them. Each week, you will focus on one essential skill that builds towards this goal.

We strongly encourage peer-to-peer learning—you'll gain so much by discussing and collaborating with others!

If you have any questions, feel free to ask in the community. Let's learn and grow together!

Chapter II

Common Instructions


- The version of Python recommended to use is 3.12, you can check the version of Python with the following command: `python -V`
- The norm: during this Boot camp, it is recommended to follow the [PEP 8 standards](#), though it is not mandatory. You can install [pycodestyle](#) which is a tool to check your Python code.
- The function `eval` is never allowed.
- The exercises are ordered from the easiest to the hardest.
- Your exercises are going to be evaluated by someone else, so make sure that your variable names and function names are appropriate and civil.
- Your manual is the internet.
- You can access our community on Whats App and ask your Questions for your peers in the dedicated Boot camp channel.
- If you find any issue or mistake in the subject please get in touch with us or create an issue on 1337-AI repository on Github.
- Submit your work to your git repository. Only the work in the git repository will be graded

Contents

I	Introduction	1
II	Common Instructions	2
III	Exercise 00	4
IV	Exercise 01	6
V	Exercise 02	7
VI	Exercise 03	8
VII	Exercise 04	10
VIII	Exercise 05	13
IX	Exercise 06	14
X	Exercise 07	15

Chapter III

Exercise 00

	Exercise : 00
\$PATH	
Turn-in directory : <i>ex00/</i>	
Files to turn in : answers.txt , requirements.txt	
Forbidden functions : None	

The first thing you need to do is install Python.

Most modern Unix-based systems have a **python** interpreter installed by default, but its version might be lower/higher than the one used for these modules. It is also possible that the default **python** command uses a version 3.x (for legacy reasons). This is obviously very confusing for a new developer.

```
$> python -v
$> python3 -v
```

To deal with those version issues we will use **venv**. This program allows you to manage your Python packages and several working environments.

Note: the actual requirement is to use a Python 3.12.X version. You are free to use a different program/utilities to achieve this goal. At your own risk.

Venv manual installation

1. Download & Install Venv

```
$> python3 -m pip install virtualenv
```

3. Create a dedicated environment !

```
$> mkdir my_project  
$> cd my_project  
$> python3 -m venv .venv  
> source .venv/bin/activate
```

4. Check your Python environment

```
$> pip list  
Package      Version  
-----  
pip          23.0.1
```



Check this for more information about Virtual Env [Resource](#)

(Finally) getting started

Now that your setup is ready to run, here are a few questions that need to be solved using `python`, `pip` or `conda`. Save your answers in a file `answers.txt` (one answer per line and per question), and check them with your peers.

Find the commands to:


- Output a list of installed packages and their versions.
- Show the package metadata of `numpy`.
- Remove the package `numpy`.
- (Re)install the package `numpy`.
- Freeze your `python` packages and their versions in a `requirements.txt` file you have to turn-in.



What is Conda?

Chapter IV

Exercise 01

	Exercise : 01
Rev Alpha	
Turn-in directory : <i>ex01/</i>	
Files to turn in : exec.py	
Forbidden functions : None	

Make a program that takes a string as argument, reverses it, swaps its letters case and prints the result.


- If more than one argument is provided, merge them into a single string with each argument separated by a single space character.
- If no argument is provided, do nothing or print an usage.

Examples

```
$> python3 exec.py 'Hello World!' | cat -e
!DLR0w OLLEh$
$>
$> python3 exec.py 'Hello' 'my Friend' | cat -e
DNEIRf YM OLLEh$
$>
$> python3 exec.py
$>
```

Chapter V

Exercise 02

	Exercise : 02
The Odd, the Even and the Zero	
Turn-in directory : <i>ex02/</i>	
Files to turn in : whois.py	
Forbidden functions : None	

Make a program that takes a number as argument, checks whether it is odd, even or zero, and prints the result.


- If more than one argument is provided or if the argument is not an integer, print an error message.
- If no argument is provided, do nothing or print an usage.

Examples

```
$> python3 whois.py 12
I'm Even.
$>
$> python3 whois.py 3
I'm Odd.
$>
$> python3 whois.py
$>
$> python3 whois.py 0
I'm Zero.
$>
$> python3 whois.py Hello
AssertionError: argument is not an integer
$>
$> python3 whois.py 12 3
AssertionError: more than one argument is provided
$>
```


Chapter VI

Exercise 03

	Exercise : 03
Functional file	
Turn-in directory : <i>ex03/</i>	
Files to turn in : <code>count.py</code>	
Forbidden functions : <code>None</code>	

Part 1. `text_analyzer`

Create a function called `text_analyzer` that takes a single string argument and displays the total number of printable characters, and respectively : the number of upper-case characters, lower-case characters, punctuation characters and spaces.

- If `None` or nothing is provided, the user is prompted to provide a string.
- If the argument is not a string, print an error message.
- This function must have a `docstring` explaining its behavior.

Test your function with the `python` console

Examples

```
$> python3
>>> from count import text_analyzer
>>> text_analyzer("Python 2.0, released 2000, introduced
features like List comprehensions and a garbage collection
system capable of collecting reference cycles.")
The text contains 143 printable character(s):
- 2 upper letter(s)
- 113 lower letter(s)
- 4 punctuation mark(s)
- 18 space(s)
>>> text_analyzer("Python is an interpreted, high-level,
general-purpose programming language. Created by Guido van
Rossum and first released in 1991, Python's design philosophy
emphasizes code readability with its notable use of significant
whitespace.")
```

```
The text contains 234 printable character(s):
- 5 upper letter(s)
- 187 lower letter(s)
- 8 punctuation mark(s)
- 30 space(s)
>>> text_analyzer()
What is the text to analyze?
>> Hello World!
The text contains 12 printable character(s):
- 2 upper letter(s)
- 8 lower letter(s)
- 1 punctuation mark(s)
- 1 space(s)
>>> text_analyzer(42)
AssertionError: argument is not a string
>>> print(text_analyzer.__doc__)

This function counts the number of upper characters, lower characters,
punctuation and spaces in a given text.
```

Part 2. `__name__ == __main__`

In the previous part, you wrote a function that can be used in the console or in another file when imported. Without changing this behavior, update your file so it can also be launched as a standalone program.


- If more than one argument is provided to the program, print an error message.
- Otherwise, use the `text_analyzer` function.

Examples

```
$> python3 count.py 'Hello World!'
The text contains 12 character(s):
- 2 upper letter(s)
- 8 lower letter(s)
- 1 punctuation mark(s)
- 1 space(s)
$> python3
>>> from count import text_analyzer
>>> text_analyzer("Hello World!")
The text contains 12 character(s):
- 2 upper letter(s)
- 8 lower letter(s)
- 1 punctuation mark(s)
- 1 space(s)
```

Chapter VII

Exercise 04

	Exercise : 04
The right format	
Turn-in directory : <i>ex04/</i>	
Files to turn in : kata00.py , kata01.py , kata02.py , kata03.py , kata04.py	
Forbidden functions : None	

Let's get familiar with the useful concept of **string formatting** through a kata series.

Each exercise will provide you with a **kata** variable. This variable can be modified to a certain extent: your program must react accordingly.

kata00

The **kata** variable is always a tuple and can only be filled with integers.

```
# Put this at the top of your kata00.py file
kata = (19,42,21)
```

Write a program that displays this variable content according to the format shown below:

```
$> python3 kata00.py
The 3 numbers are: 19, 42, 21
$>
```

kata01

The **kata** variable is always a dictionary and can only be filled with strings.

```
# Put this at the top of your kata01.py file
kata = {
    'Python': 'Guido van Rossum',
    'Ruby': 'Yukihiro Matsumoto',
    'PHP': 'Rasmus Lerdorf',
}
```

Write a program that displays this variable content according to the format shown below:

```
$> python3 kata01.py
Python was created by Guido van Rossum
Ruby was created by Yukihiro Matsumoto
PHP was created by Rasmus Lerdorf
$>
```

kata02

The `kata` variable is always a tuple that contains 5 non-negative integers. The first integer contains up to 4 digits, the rest up to 2 digits.

```
# Put this at the top of your kata02.py file
kata = (2019, 9, 25, 3, 30)
```

Write a program that displays this variable content according to the format shown below:

```
$> python3 kata02.py | cat -e
09/25/2019 03:30$
$> python3 kata02.py | wc -c
17
$>
```

kata03

The `kata` variable is always a string whose length is not higher than 42.

```
# Put this at the top of your kata03.py file
kata = "The right format"
```

Write a program that displays this variable content according to the format shown below:

```
$> python3 kata03.py | cat -e
-----The right format%
$> python3 kata03.py | wc -c
42
$>
```

kata04

The `kata` variable is always a tuple that contains, in the following order:

- 2 non-negative integers containing up to 2 digits
- 1 decimal
- 1 integer
- 1 decimal

```
# Put this at the top of your kata04.py file
kata = (0, 4, 132.42222, 10000, 12345.67)
```


Write a program that displays this variable content according to the format shown below:

```
$> python3 kata04.py
module_00, ex_04 : 132.42, 1.00e+04, 1.23e+04
$> python3 kata04.py | cut -c 10,18
,:

```

Chapter VIII

Exercise 05

	Exercise : 05
Shorter, faster, pythonic	
Turn-in directory : <i>ex05/</i>	
Files to turn in : filterwords.py	
Forbidden functions : filter	

Make a program that takes a string *S* and an integer *N* as argument and prints the list of words in *S* that contains more than *N* non-punctuation characters.

- Words are separated from each other by space characters
- Punctuation symbols must be removed from the printed list: they are neither part of a word nor a separator
- The program must contain at least one **list comprehension** expression.


If the number of argument is different from 2, or if the type of any argument is invalid, the program prints an error message.

Examples

```
$> python3 filterwords.py 'Hello, my friend' 3
['Hello', 'friend']
$> python3 filterwords.py 'Hello, my friend' 10
[]
$> python3 filterwords.py 'A robot must protect its own existence as long as such protection does not
    conflict with the First or Second Law' 6
['protect', 'existence', 'protection', 'conflict']
$> python3 filterwords.py Hello World
ERROR
$> python3 filterwords.py 3 'Hello, my friend'
ERROR
$> python3 filterwords.py
ERROR
```

Chapter IX

Exercise 06

	Exercise : 06
Loading bar!	
Turn-in directory : <i>ex06/</i>	
Files to turn in : <code>loading.py</code>	
Forbidden functions : <code>tqdm</code> or any library for automatic loading bar	

You are about to discover the `yield` operator!
So let's create a function called `ft_progress(lst)`.
The function will display the progress of a `for` loop.

Examples

```
listy = range(1000)
ret = 0
for elem in ft_progress(listy):
    ret += (elem + 3) % 5
    sleep(0.01)
print()
print(ret)
```

```
$> python loading.py
ETA: 8.67s [ 23%][====>                ] 233/1000 | elapsed time 2.33s
...
2000
```


```
listy = range(3333)
ret = 0
for elem in ft_progress(listy):
    ret += elem
    sleep(0.005)
print()
print(ret)
```

```
$> python loading.py
ETA: 14.67s [ 9%][=>                    ] 327/3333 | elapsed time 1.33s
...
5552778
```

{We advise you to go take a look at the wonderful `tqdm` library, it will come in handy in many situations

Chapter X

Exercise 07

	Exercise : 07
Scraper	
Turn-in directory : <i>ex07/</i>	
Files to turn in : scraper.py, requirements.txt, data.csv	
Forbidden functions : None	

By the end of this bootcamp, you will build an AI model capable of predicting house prices based on features such as the number of bathrooms, area, parking spaces, and more. To achieve this, we first need to gather the data required for training our model. That's why we have provided a dataset available at this [link](#).

Your Task:

- Scrape the website to extract the house price data.
- Save the extracted data in a CSV file.

This dataset will be used later to train and evaluate your machine learning model.

Examples

```
$> python3 scraper.py data.csv  
data scrapped successfully and saved in data.csv
```

```
$> cat data.csv | less  
price,area,bedrooms,bathrooms,stories,mainroad,guestroom,basement,hotwaterheating,airconditioning,parking,  
prefarea,furnishingstatus  
13300000,7420,4,2,3,yes,no,no,no,yes,2,yes,furnished  
12250000,8960,4,4,4,yes,no,no,no,yes,3,no,furnished  
12250000,9960,3,2,2,yes,no,yes,no,no,2,yes,semi-furnished  
12215000,7500,4,2,2,yes,no,yes,no,yes,3,yes,furnished  
11410000,7420,4,1,2,yes,yes,yes,no,yes,2,no,furnished  
:
```



you can use `beautifulsoup`



Now that you have scraped and saved the dataset, take a moment to analyze it. What patterns do you notice in the data?

Contact

You can contact 1337AI organization by email: contact@1337ai.org

Find all the relevant and up-to-date information about 1337AI on [our Website](#) !

Thank you for attending the Entry-Level ML Engineer Boot-camp!