Course:	**Maschinelles Lernen und künstliche Intelligenz/ Machine Learning and Artificial Intelligence**		
(Group) Member:	*Nr.*	*Name*	*Matrikel*
	1	**Miguel Angel Lopez Mejia**	**5197617**
	2		
	3		

Task Reporting Template

Please read the given Task carefully. You can find all necessary information in Moodle.

1. Definition

1.1. Introduction and Explanations

This task aims to familiarize you as students with the basics of Python and VS-Code to fulfill following tasks:

1. Program a first "Hello World" program with Python. Feel free to create an extended version also which combines some strings to a major string that is printed.
2. Program a "dice-algorithm" program with Python. Using required Python packages and confirming the randomness.

The expected result is two working Python applications including documentation and source code. The "Hello World" program is intended to show the functionality of the development environment. The “dice program” is your first program. Use standard packages and functions of Python. The dice-algorithm must prove through mathematical algorithms that the random number generator used works correctly. For example, by a diagram of the statistical distribution.

1.2. Allowed Tools

- The script is going to be executed using ***Python version 3.13.2***
- ***Visual Studio Code*** is going to be used as IDE (Integrated Development Environment)
- Libraries: random, sys, matplotlib.pyplot, logging, numpy, collections, statistics

2. Your Preparation/ Concept

2.1. Structure your Problem

- **Task 1.1 - I have to work on:**
  - My task is to print 'Hello World from <*my_name*>' in the Command line/ Terminal
  - In order to solve the problem, I created a script that works in a virtual environment, once executed, asks the user to introduce a name and by using a class called ***'HelloWorld',*** it concatenates the string 'Hello World from' + <user_name> and displays it in the console.

- **Task 1.2 - I will work on following points for first part:**
  - The problem to solve in this task is to create an algorithm that simulates dice by using python.
  - In order to solve the problem, I created a script that works in a virtual environment, once executed, asks the user to introduce the number of dice by using a class called ***DiceGenerator,*** depending on the number of dice, a for loop is used, and with help of the library 'random' the script simulates the behavior of a die by generating numbers between 1 and 6. Then the result of each dice is displayed in the console.

- **Task 1.3 - I will work on following points for second part:**
  - The problem to solve in this task is to create an algorithm that simulates dice by using python and prove randomness, which, in statistics, implies that the occurrence of one event does not influence the occurrence of another.
  - In order to solve the problem, I created a script that works in a virtual environment, once executed, the script will follow these steps:
    - Asks the user to introduce the number of times that the die is going to be rolled by using a class called ***DiceGenerator.***
    - Depending on the number of rolls, a for loop is used, and with help of the library 'random' the script simulates the behavior of a die by generating numbers between 1 and 6, each number is stored in a list.
    - The next step is to count how many times each face of the die appears in the list.
    - ***Randomness***, I used the ***Chi-Square*** method to determine if there is a significant difference between the expected frequencies and the observed frequencies. The formula of Chi-square is represented by $\chi 2 = \sum \frac{(Oi-Ei)2}{Ei}$ where:
      - Oi is the observed frequency, in this case, the frequency of each face of the die.
      - Ei is the expected frequency of each face of the die, in this case ⅙.
      - In order to use the Chi-square method, we need to consider:
        - ***Null hypothesis*** (Ho) = The die is fair and each face has the same probability of ⅙.
        - ***Critic Value*** = 11.07 (Considering a significance level $(\alpha) = 0.05$, It is the threshold defined to decide when a difference between what is observed and what is expected is *"too large"* to be attributed to chance, 0.05 is the standard -

Check Images/3_Dice_Chi_square_probability_table.png for reference)

- To reject Ho, we expect that $\chi 2 > Critic\ Value$

- Once randomness is proved, a graphic is display, showing the $mean$ values and the result of each roll number
- Finally, an histogram is displayed, comparing the die values vs the frequency.

Flows:

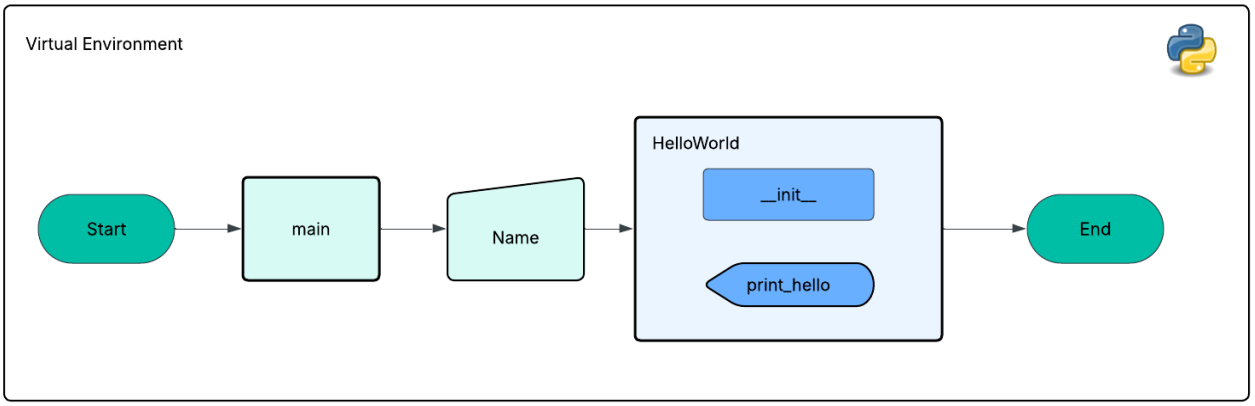

Figure 1: Hello world flow diagram

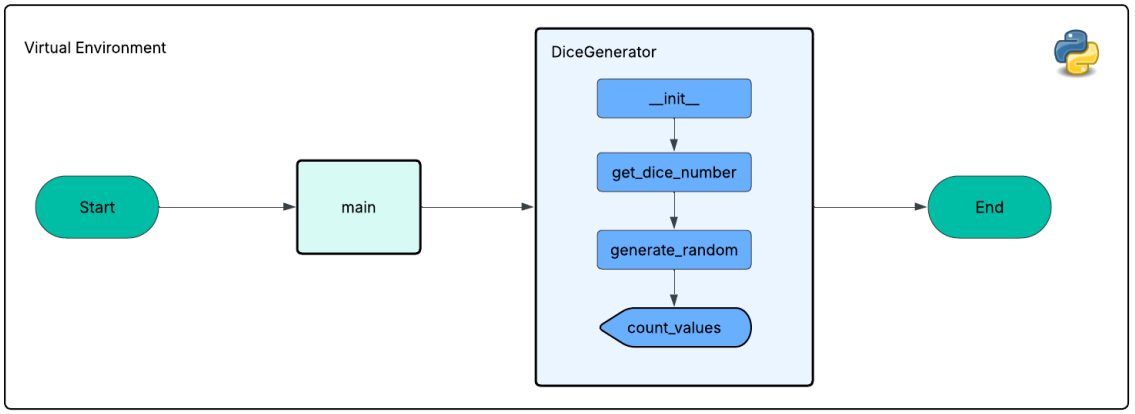

Figure 2: Dice flow diagram

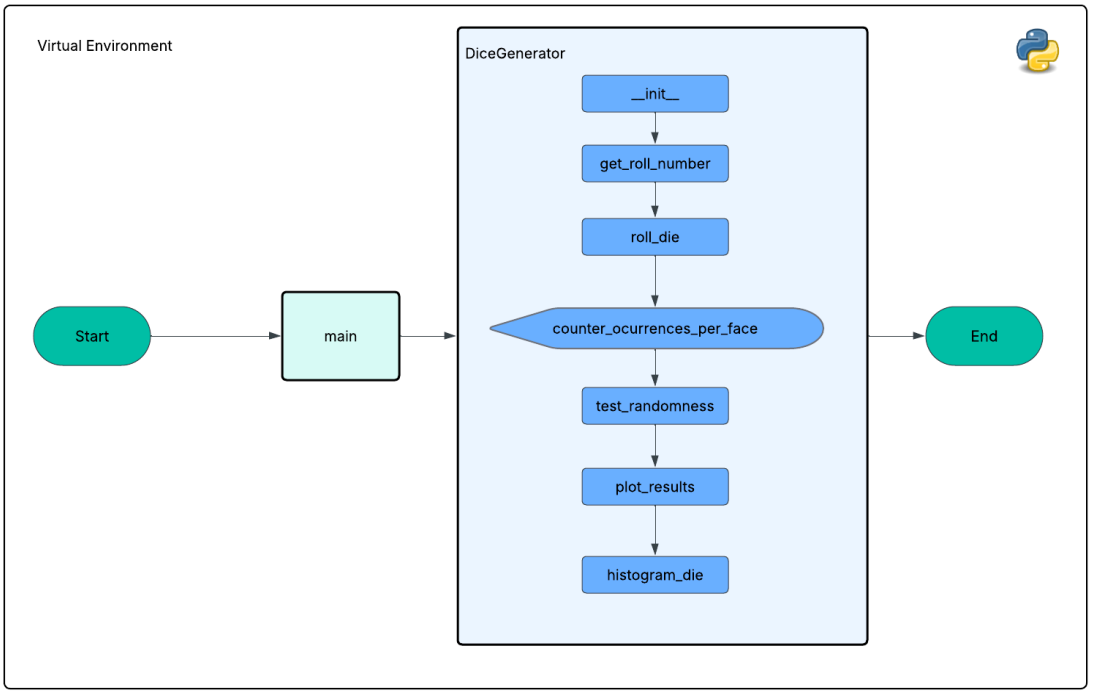

Figure 3: Die flow diagram

Libraries:

- **random** is a python library that provides functions to generate random numbers, commonly used for simulations, games and applications that require unpredictable behavior.
- **matplotlib** is a comprehensive library for creating static, animated, and interactive visualizations.
- **logging**: this module defines functions and classes which implement a flexible event logging system for applications and libraries.
- **numpy** is a Python library providing a multidimensional array object, derived objects like masked arrays and matrices, and tools for fast array operations, including math, logic, shape manipulation, sorting, I/O, Fourier transforms, linear algebra, statistics, and random simulation.
  - **sys** is a library that allows to interact with the run-time environment of python
  - **collections**: This module implements specialized container data types providing alternatives to Python's general purpose built-in containers, dict, list, set, and tuple.
  - **statistics**: this module provides functions for calculating mathematical statistics of numeric (Real-valued) data.

```python
# region Libraries

import random
import matplotlib.pyplot as plt
import logging
import numpy as np
import sys
from collections import Counter
from statistics import mean

# endregion
```

2.2. Your Implementation

Virtual Environment

- In order to run the script, a virtual environment must be created, open a new terminal and create it as it is shown:

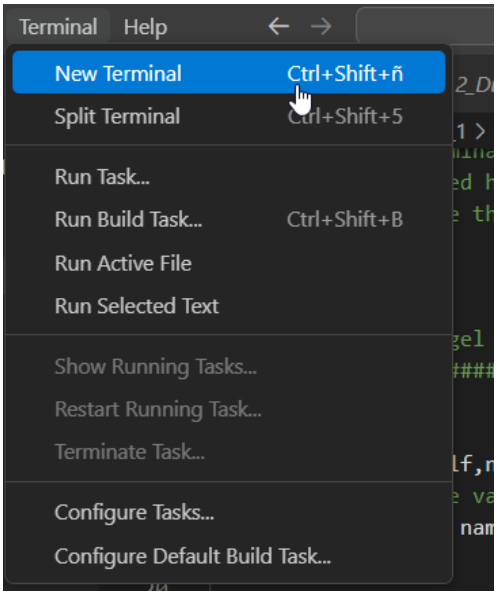

Figure 4: Create a new terminal in VSCODE

- Execute the following command in the console in order to create a new virtual environment: ***python -m venv venv*** once the virtual environment is created, it will be displayed in the explorer section:

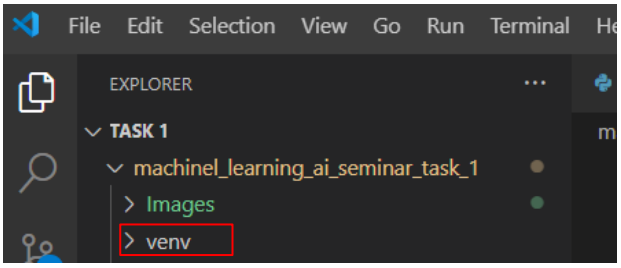

Figure 5: Virtual environment folder

- In order to activate the virtual environment, execute this script in the console: ***venv/scripts/activate*** .

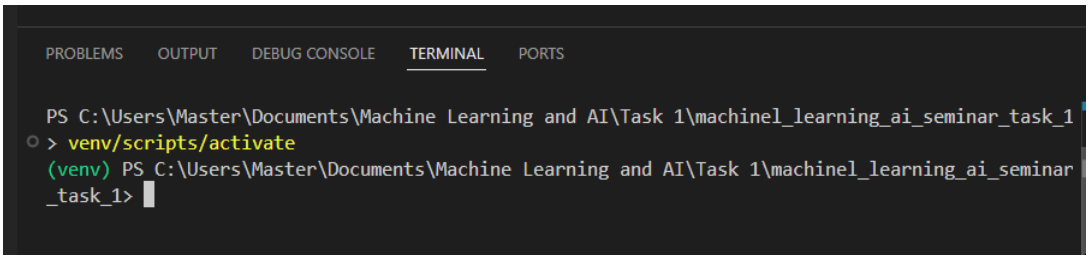

Figure 6: Activation of virtual environment

Task 1.1 - Hello World

- In order to execute the script, run this command in the console: ***python .\1_HelloWorld.py.py***, the name of the user will be requested:

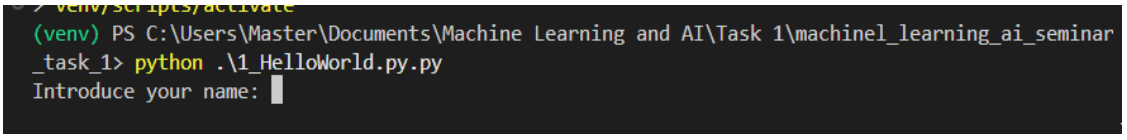

Figure 7: Introduce Name

- The script will show the name of the user concatenated with 'Hello World from':

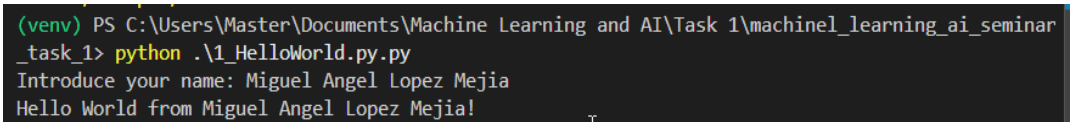

Figure 8: Display name of the user

Task 1.2 - Dice

- In order to execute the script, run this command in the console: ***python .\2_Dice.py*** , the number of dice will be requested:

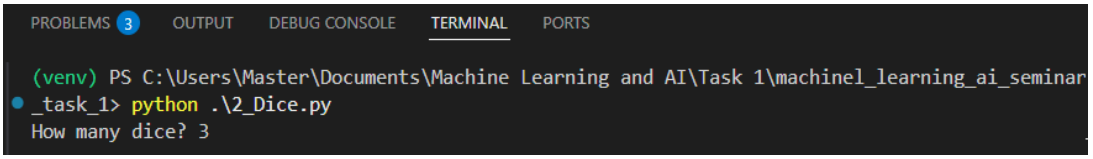

Figure 9: Introduce number of dice

- The script contains a class called ***DiceGenerator*** within 3 methods:
  - ***get_dice_number***: Ask the user to introduce a number of dice
  - ***generate_random:*** Generates random values (1-6) for a specified number of dice.
  - ***count_values***: Prints the value of each dice from a dictionary of dice results.

- Once executed, the script will show the results:

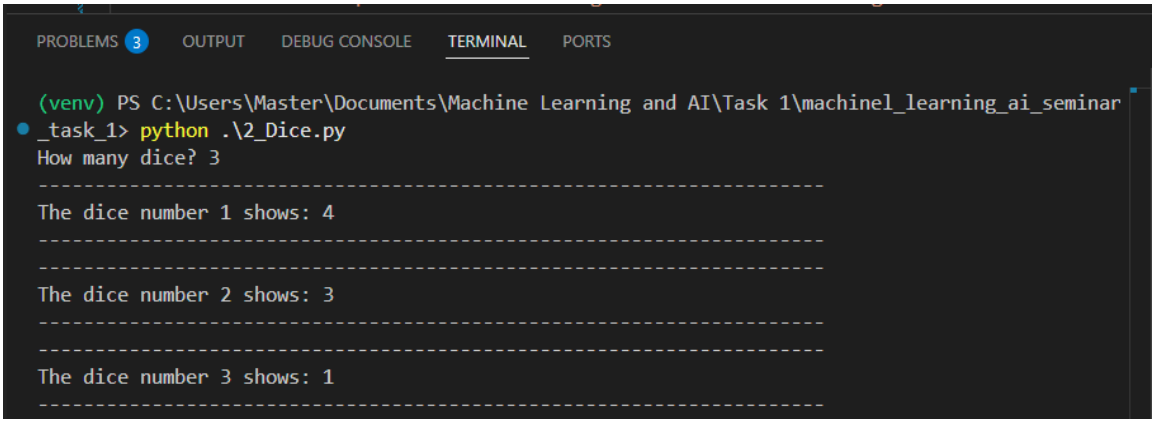

Figure 10: Display dice values

Task 1.3 - Dice Ext

- In order to execute the script, run this command in the console: ***python .\3_Dice_Ext.py*** . The number of rolls is requested:

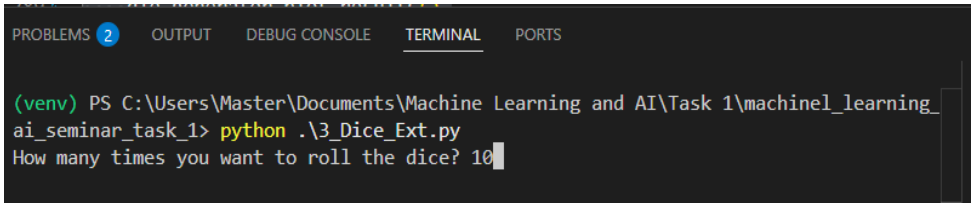

Figure 11: Introduce number of rolls

- The script contains a class called ***DiceGenerator*** within 6 methods, each method contains the description of they work and examples:
  - ***get_roll_number***: Ask the user to introduce a number of rolls
  - ***roll_die:*** Generates random values (1-6) for a specified number of rolls.
  - ***count_ocurrences_per_face***: Counts the occurrences of each die face in the results and logs the statistics.
  - **test_randomness**: Uses the ***Chi-Square*** method to determine if there is a significant difference between the expected frequencies and the observed frequencies.
  - **plot_results**: Generates a bar chart of dice roll results with a mean line.
  - **histogram_die**: Generates a histogram showing the frequency distribution of dice roll results.

- The output of the script is going to show the following ***(in this case the number of rolls is 10)***:
- The number of times that each face of the die appeared and the percentage

```
(venv) PS C:\Users\Master\Documents\Machine Learning and AI\Task 1\machinel_learning_ai_seminar_task_1> python
\3_Dice_Ext.py
How many times you want to roll the dice? 10
INFO:__main__:-------------------------------------------------------------------------------------------------
INFO:__main__: Face 1: Appears 2 times - (20.0 %)
INFO:__main__: Face 2: Appears 1 times - (10.0 %)
INFO:__main__: Face 3: Appears 3 times - (30.0 %)
INFO:__main__: Face 4: Appears 2 times - (20.0 %)
INFO:__main__: Face 5: Appears 0 times - (0.0 %)
INFO:__main__: Face 6: Appears 2 times - (20.0 %)
```

Figure 12: Display face values and percentage

- The conclusion of the test of randomness, by using the Chi-square method, as well as the expected mean vs the observed mean .

```
INFO:__main__:-------------------------------------------------------------------------------------------------
INFO:__main__:-------------------------------------------------------------------------------------------------
INFO:__main__:------------------------- Test of Randomness - Chi-square method ----------------------------
INFO:__main__:-------------------------------------------------------------------------------------------------
INFO:__main__: Degrees of freedom: 5 (6 faces - 1)
INFO:__main__: Compare with critical value (e.g., 11.07 for alpha=0.05, df=5)
INFO:__main__: The value of Chi-square (3.1999999999999997) < Critic Value (11.07)
INFO:__main__: The conclusion is that there is no enough information to reject the Null hypothesis
INFO:__main__:-------------------------------------------------------------------------------------------------
INFO:__main__:-------------------------------------------------------------------------------------------------
INFO:__main__: The Expected mean is 3.5
INFO:__main__: The Observed mean is 3.3
```

Figure 13: Test of randomness with Chi-square and Mean values

- A graphic that shows the dice number and the value that was associated to it, as well as the mean that was obtained.:

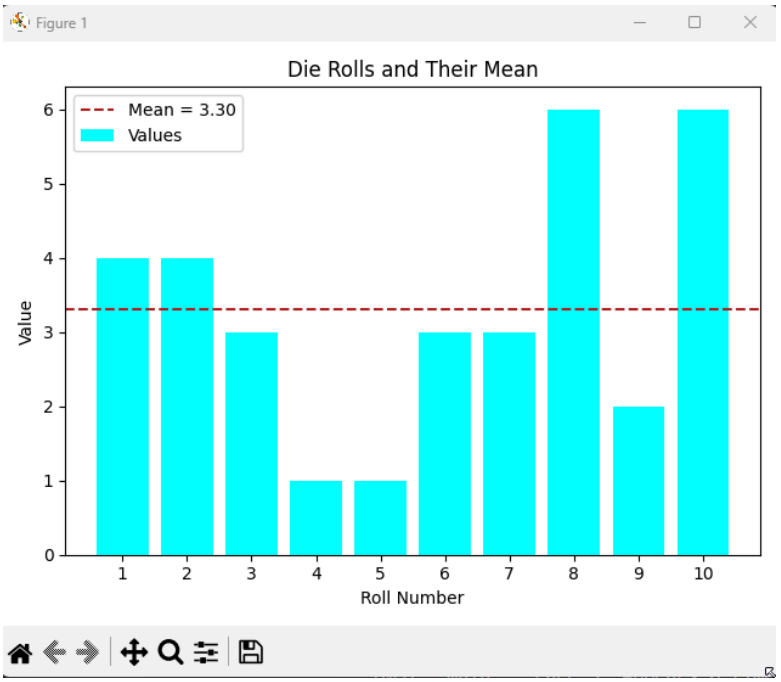

Figure 14: Graphic - Roll Number vs - Value (Mean)

- An histogram that shows the face/die value vs the frequency:

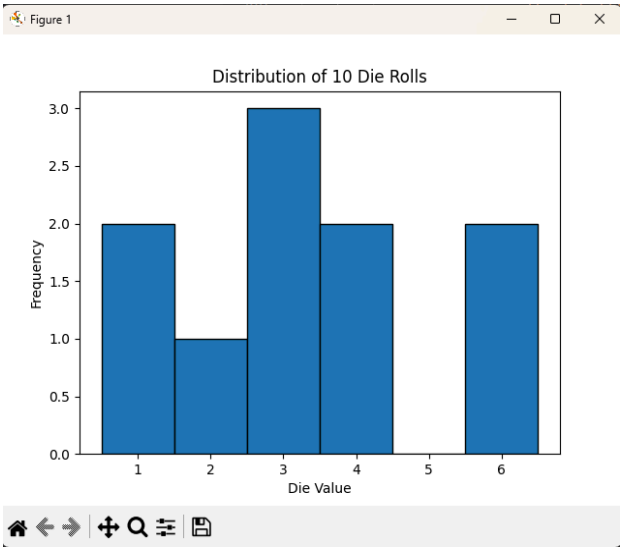

Figure 15: Histogram -Distribution 0f Die Rolls

Analysis

I created 3 more tests for this exercise in order to analyze the data and create a conclusion of the behavior of the script/die.

Test - 20 rolls

- Test with 20 rolls:

```
(venv) PS C:\Users\Master\Documents\Machine Learning and AI\Task 1\machinel_learning_ai_seminar_task_1> python
\3_Dice_Ext.py
How many times you want to roll the dice? 20
INFO:__main__:------------------------------------------------------------------------------------------------------------
INFO:__main__: Face 1: Appears 5 times - (25.0 %)
INFO:__main__: Face 2: Appears 4 times - (20.0 %)
INFO:__main__: Face 3: Appears 2 times - (10.0 %)
INFO:__main__: Face 4: Appears 1 times - (5.0 %)
INFO:__main__: Face 5: Appears 4 times - (20.0 %)
INFO:__main__: Face 6: Appears 4 times - (20.0 %)
INFO:__main__:------------------------------------------------------------------------------------------------------------
INFO:__main__:------------------------------------------------------------------------------------------------------------
INFO:__main__:------------------------- Test of Randomness - Chi-square method -------------------------
INFO:__main__:------------------------------------------------------------------------------------------------------------
INFO:__main__: Degrees of freedom: 5 (6 faces - 1)
INFO:__main__: Compare with critical value (e.g., 11.07 for alpha=0.05, df=5)
INFO:__main__: The value of Chi-square (3.4000000000000004) < Critic Value (11.07)
INFO:__main__: The conclusion is that there is no enough information to reject the Null hypothesis
INFO:__main__:------------------------------------------------------------------------------------------------------------
INFO:__main__:------------------------------------------------------------------------------------------------------------
INFO:__main__: The Expected mean is 3.5
INFO:__main__: The Observed mean is 3.35
INFO:__main__:------------------------------------------------------------------------------------------------------------
```

Figure 16: Algorithm with 20 rolls

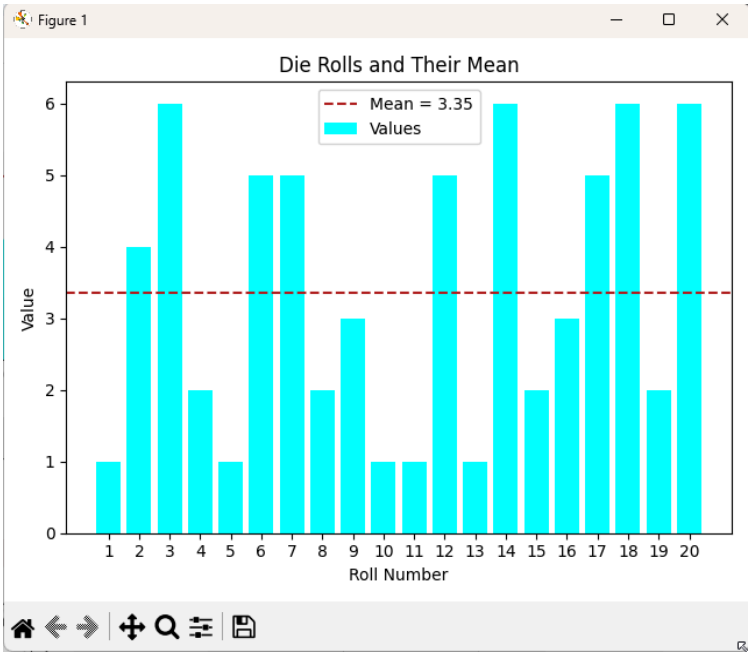

Figure 17: Roll number vs values (20 rolls)

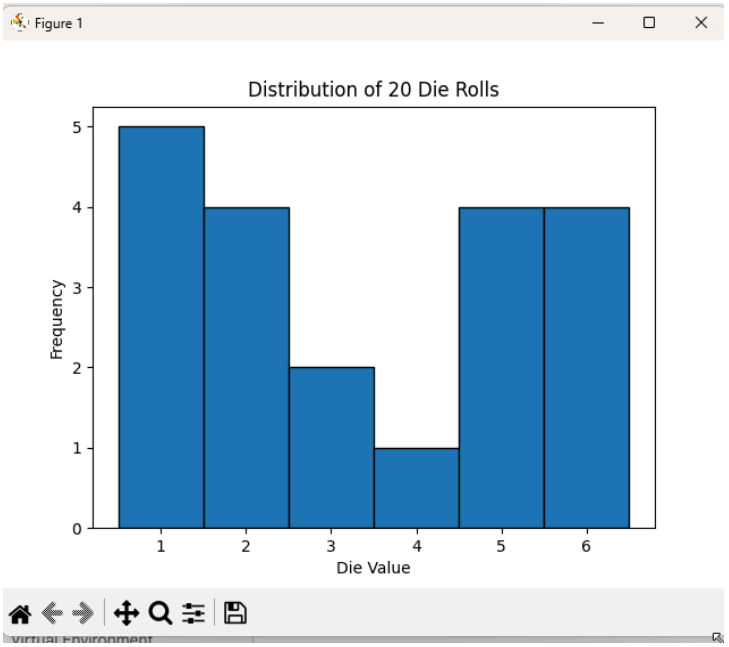

Figure 18: Die value vs Frequency (20 rolls)

Test - 10,000 rolls:

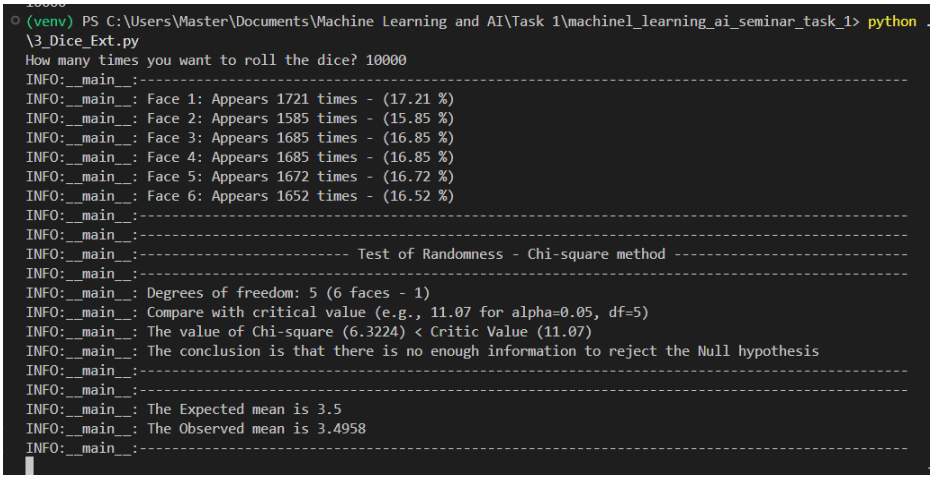

Figure 19: Algorithm with 10,000 rolls

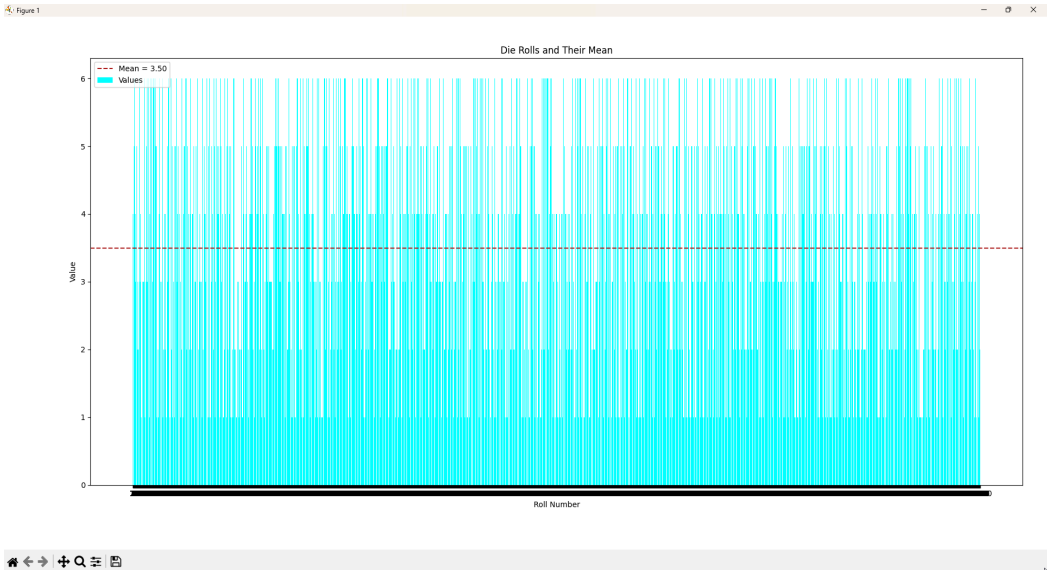

Figure 20: Roll number vs values
(10,000 rolls)

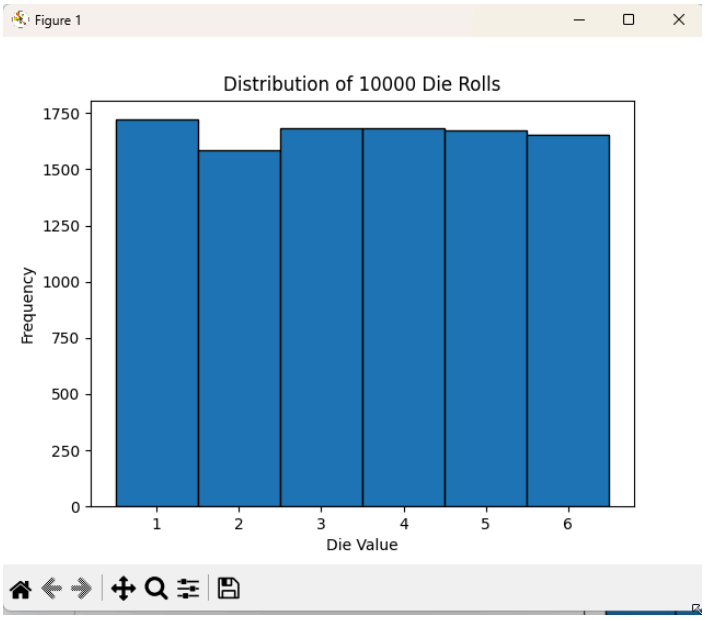

Figure 21: Die value vs Frequency
(10,000 rolls)

Test - 100,000 rolls:

```
INFO:__main__:------------------------------------------------------------------------------------------------------------
(venv) PS C:\Users\Master\Documents\Machine Learning and AI\Task 1\machinel_learning_ai_seminar_task_1> python
\3_Dice_Ext.py
How many times you want to roll the dice? 100000
INFO:__main__:------------------------------------------------------------------------------------------------------------
INFO:__main__: Face 1: Appears 16795 times - (16.79 %)
INFO:__main__: Face 2: Appears 16655 times - (16.66 %)
INFO:__main__: Face 3: Appears 16470 times - (16.47 %)
INFO:__main__: Face 4: Appears 16718 times - (16.72 %)
INFO:__main__: Face 5: Appears 16736 times - (16.74 %)
INFO:__main__: Face 6: Appears 16626 times - (16.63 %)
INFO:__main__:------------------------------------------------------------------------------------------------------------
INFO:__main__:------------------------------------------------------------------------------------------------------------
INFO:__main__:------------------------- Test of Randomness - Chi-square method -------------------------------------
INFO:__main__:------------------------------------------------------------------------------------------------------------
INFO:__main__: Degrees of freedom: 5 (6 faces - 1)
INFO:__main__: Compare with critical value (e.g., 11.07 for alpha=0.05, df=5)
INFO:__main__: The value of Chi-square (3.8627599999999993) < Critic Value (11.07)
INFO:__main__: The conclusion is that there is no enough information to reject the Null hypothesis
INFO:__main__:------------------------------------------------------------------------------------------------------------
INFO:__main__:------------------------------------------------------------------------------------------------------------
INFO:__main__: The Expected mean is 3.5
INFO:__main__: The Observed mean is 3.49823
INFO:__main__:------------------------------------------------------------------------------------------------------------
```

Figure 22: Algorithm with 100,000 rolls

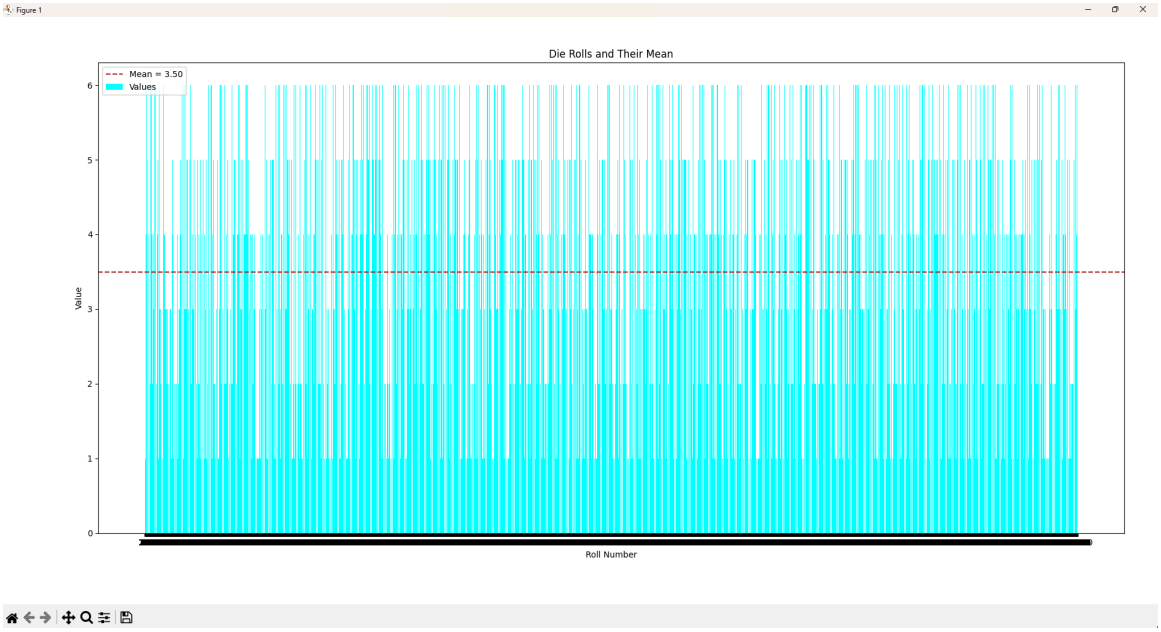

Figure 23: Roll number vs values (100,000 rolls)

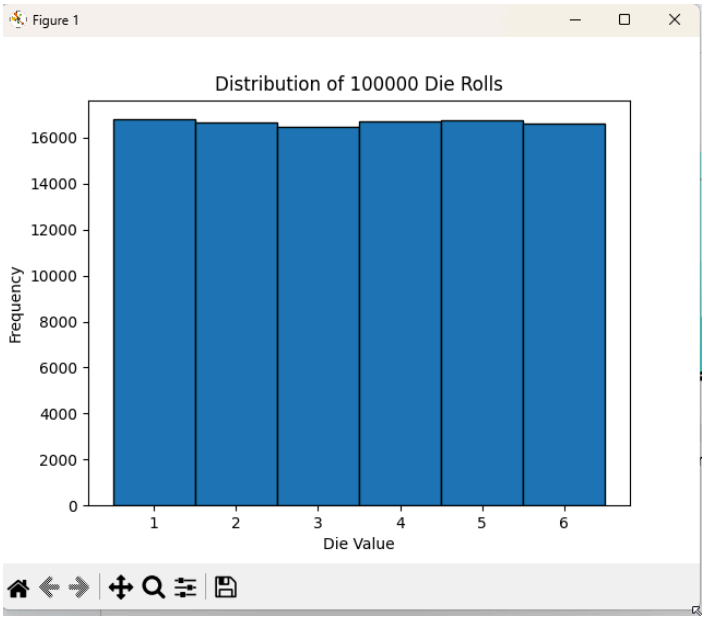

Figure 24: Die value vs Frequency (100,000 rolls)

Conclusion

I created a table that contains the results obtained for distinct number of rolls:

Face	Number of rolls	Frequency	Percentage	Mean	Chi-square
1	10	2	20	3.3	3.199
2	10	1	10	3.3	3.199
3	10	3	30	3.3	3.199
4	10	2	20	3.3	3.199
5	10	0	0	3.3	3.199
6	10	2	20	3.3	3.199
1	20	5	25	3.35	3.40
2	20	4	20	3.35	3.40
3	20	2	20	3.35	3.40
4	20	1	5	3.35	3.40
5	20	4	20	3.35	3.40
6	20	4	20	3.35	3.40
1	10000	1721	17.21	3.4958	6.3224
2	10000	1585	15.85	3.4958	6.3224
3	10000	1685	16.85	3.4958	6.3224
4	10000	1685	16.85	3.4958	6.3224
5	10000	1672	16.72	3.4958	6.3224
6	10000	1652	16.52	3.4958	6.3224
1	100000	16795	16.79	3.49823	3.8627
2	100000	16655	16.66	3.49823	3.8627
3	100000	16470	16.47	3.49823	3.8627
4	100000	16718	16.72	3.49823	3.8627
5	100000	16737	16.74	3.49823	3.8627
6	100000	16626	16.63	3.49823	3.8627

Table 1: Results Die algorithm

According to the ***Table 1: Results Die algorithm***, I created this graphic, which allows me to conclude that the *obtained mean* tends to be closer to the *expected mean (3.5)* in relation with the number of rolls:

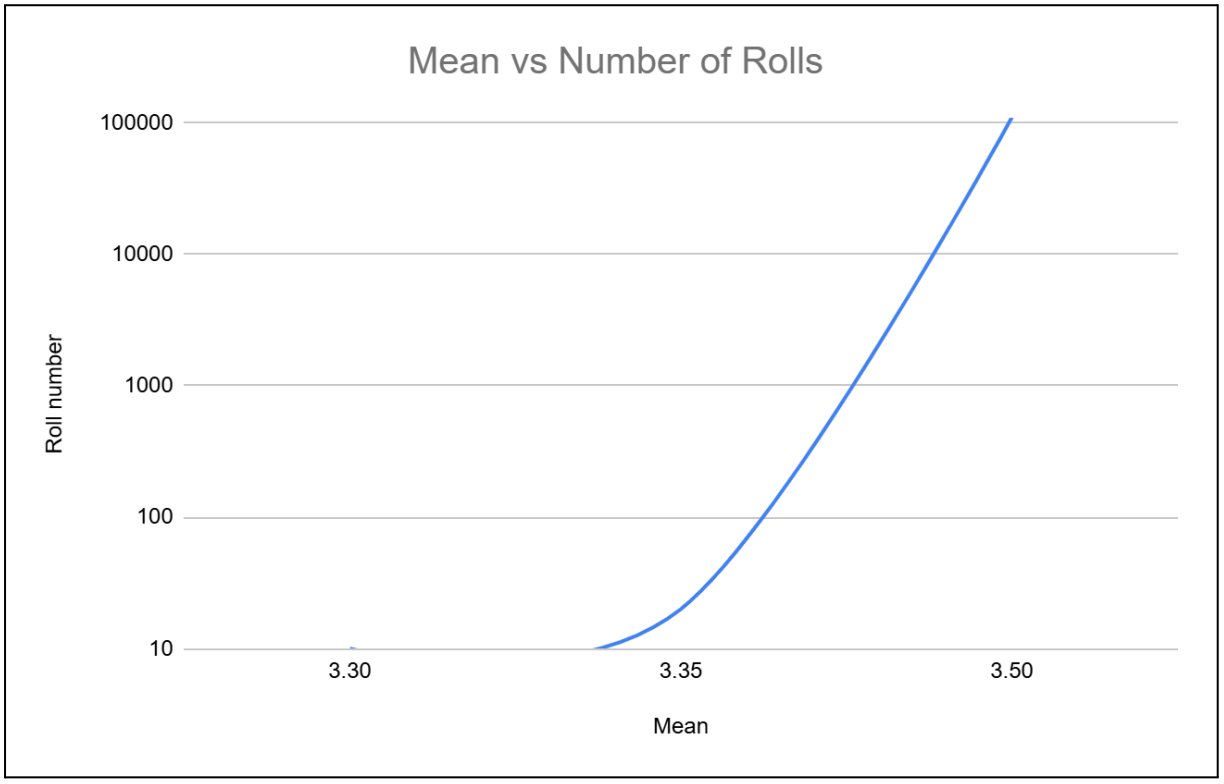

Figure 25: Mean vs Number of rolls

In the ***Table 2: Ranges of results per Number of rolls***, I can appreciate that the difference between the *minimum percentage* and the *maximum percentage* in each roll, tends to decrease when the number of rolls increase. I expect that the percentage of each face of the dice to be $1/6 = 0.1666 = 16.66\%$ and the column **'Range (Max - Min)' shows how the difference between the percentages of each face decreases when the number of rolls increase.**

Number of rolls	Minimum Percentage	Maximum Percentage	Range (Max - Min)
10	0.0	30.0	30.0
20	5.0	25.0	20.0
10000	15.85	17.21	1.36
100000	16.47	16.80	0.33

Table 2: Ranges of results per Number of rolls

Finally, regarding the Chi-square values obtained, from ***Table 1: Results Die algorithm,*** the results don't follow the same pattern as the mean and percentage/frequency, this is due to the formula sums square deviations, and even small proportional deviations (e.g., 1721 vs. 1666.67) produce larger contributions when scaled by a larger sample size. I can take the test with 10,000 rolls as an example.

For 10,000 rolls:

- **Frequencies**: [1721, 1585, 1685, 1685, 1672, 1652].
- **Expected frequency**: $10,000/6 \approx 1666.6667$
- **Chi-square calculation**:
  - Face 1: $(1721 - 1666.6667)2/1666.6667 \approx 1.7712$
  - Face 2: $(1585 - 1666.6667)2/1666.6667 \approx 4.0017$
  - Face 3: 0.2017
  - Face 4: 0.2017
  - Face 5: 0.0171
  - Face 6: 0.1291
  - $\chi 2 = 1.7712 + 4.0017 + 0.2017 + 0.2017 + 0.0171 + 0.1291 = 6.3224$

That is why we can see that in the test with 10,000 rolls, the Chi-square is bigger than the other tests.

Github Repository

MiguelAnhalt/machinel_learning_ai_seminar_task_1: This repository contains the exercises from the task 1 of the subject Machine Learning and AI

References, extra reading links, and resources

- Corder, G. W., & Foreman, D. I. (2009). *Nonparametric statistics for non-statisticians: A step-by-step approach*. Wiley.
- Python Software Foundation. (n.d.). *random — Generate pseudo-random numbers*. Python 3 Documentation. https://docs.python.org/3/library/random.html
- Python Software Foundation. (n.d.). *sys — System-specific parameters and functions*. Python 3 Documentation. https://docs.python.org/3/library/sys.html
- PythonBasics.org. (n.d.). *Try Except in Python*. https://pythonbasics.org/try-except/
- W3Schools. (n.d.). *Python Classes and Objects*. https://www.w3schools.com/python/python_classes.asp