



Lab Report

Only for course Teacher						
		Needs Improvement	Fair	Good	Excellent	Total Mark
Allocate mark & Percentage		25%	50%	75%	100%	25
Understanding/Analysis	7					
Implementation	8					
Report Writing	10					
Total obtained mark						
Comments						

Semester: Spring 25

Student Name: Nur Ahmed

Student ID: 222-35-1111

Batch: 38th

Section: A2

Course Code: SE 334

Course Name: Artificial Intelligence Lab

Course Teacher Name: Dr. Mohammad Azam Khan

Designation: Assistant Professor

Submission Date: 13/04/2025

Lab Report for Artificial Intelligence (SE334)

Nur Ahmed 222-35-1111

January 2025

Abstract

From this course, we will learn some of the most important parts of Artificial Intelligence using Python. We will explore from environment setup to advanced Python programming for various algorithms of Artificial Intelligence.

Contents

1	Lab-1: Setup Environment for Python	3
2	Lab-2: Basic Python Code	5
3	Lab-3: Document String in Python	7
4	Lab-4: Implementation of OOP in Python	8
5	Lab-5: Solving two problems in Codeforces using Python	9
6	Lab-6: Maximizing the cost of $f(x) = -x^2 + 4x$ using Hill Climbing algorithm	10
7	Lab-7: Additional Lab Classes	11

List of Figures

1.1	Downloading Anaconda from anaconda.org	3
1.2	Default Installation Settings	3
1.3	Creating a Virtual Environment	4
1.4	Installing Libraries and Jupyter Notebook	4
1.5	Opening Jupyter Notebook	4
2.1	Basic Python Code Output 1	5
2.2	Basic Python Code Output 2	6
2.3	Basic Python Code Output 3	6
3.1	Usage of Document String in Python	7
4.1	Creating Class and Object along with Getter and Setter in Python	8
5.1	Two mild difficult problems solved in Codeforces using Python	9
6.1	Hill Climbing algorithm for maximizing cost in Python	10
7.1	Using Pandas Library in Python	11
7.2	Using Matplotlib Library in Python	12

Chapter 1

Lab-1: Setup Environment for Python

Installation Steps

Step 1: Visit `anaconda.org` and download the latest version of Anaconda.

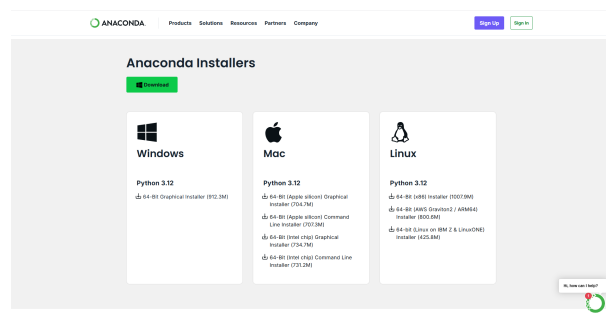


Figure 1.1: Downloading Anaconda from `anaconda.org`

Step 2: Install the software with all the default settings.

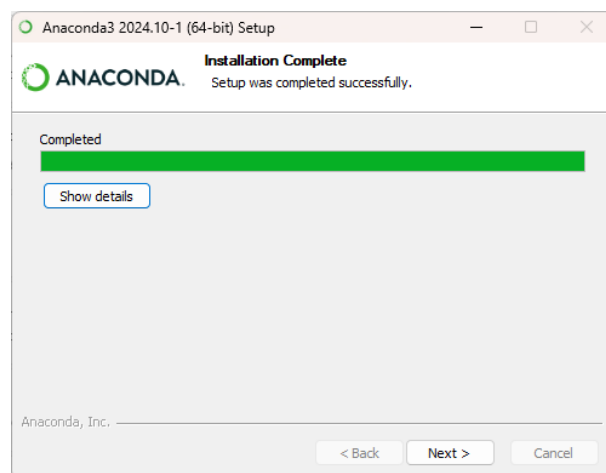


Figure 1.2: Default Installation Settings

Step 3: Create a Virtual Environment for Python using Anaconda Prompt.

```

(Users) C:\Users\User\anaconda create --name bmpi python=3.13.1
Retrieving notices: done
Channels:
 - defaults
Platform: win-64
Collecting package metadata (repodata.json): done
Solving environment: done

## Package Plan ##

environment location: D:\Anaconda\envs\bmpi

added / updated specs:
 - python=3.13.1

The following packages will be downloaded:

package                                     build
-----
numpy-2.0.2                                h8db2796_0    287 KB
libm2dec-4.0.0                             h27c369_0     95 KB
pip-24.2                                   py313haa95532_0    2.6 MB
python-3.13.1                             ha0c380_100_cp313    16.6 MB
python_abi-3.13                            0_cp313         7 KB
setuptools-75.8.0                         py313haa95532_0    2.2 MB
tornado-2025a                             h8d1a81_0       117 KB
vc-14.40                                   haa95532_2        10 KB
vs2015_runtime-10.0.270323                h9511a66_2        1.2 MB
  
```

Figure 1.3: Creating a Virtual Environment

Step 4: After activating the Virtual Environment, install necessary libraries and Jupyter Notebook.

```

(Users) C:\Users\User\pip install numpy
Collecting numpy
  Downloading numpy-2.2.2-cp313-cp313-win_amd64.whl.metadata (60 kB)
  Downloading numpy-2.2.2-cp313-cp313-win_amd64.whl (12.6 MB)
Installing collected packages: numpy
Successfully installed numpy-2.2.2

(Users) C:\Users\User\pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-3.10.0-cp313-cp313-win_amd64.whl.metadata (11 kB)
  Collecting contourpy>=1.0.1 (from matplotlib)
  Downloading contourpy-1.3.1-cp313-cp313-win_amd64.whl.metadata (5.4 kB)
  Collecting cycler>=0.10 (from matplotlib)
  Using cached cycler-0.12.1-py3-none-any.whl.metadata (3.8 kB)
  Collecting fonttools>=4.35.0 (from matplotlib)
  Downloading fonttools-4.35.0-cp313-cp313-win_amd64.whl.metadata (103 kB)
  Collecting kiwisolver>=1.3.1 (from matplotlib)
  Downloading kiwisolver-1.4.0-cp313-cp313-win_amd64.whl.metadata (6.3 kB)
  Requirement already satisfied: numpy>=1.23 in d:\anaconda\envs\bmpi\lib\site-packages (from matplotlib) (2.2.2)
  Collecting packaging>=20.0 (from matplotlib)
  Downloading packaging-24.2-py3-none-any.whl.metadata (3.2 kB)
  Collecting pillow>=8.1.0 (from matplotlib)
  Downloading pillow-11.1.0-cp313-cp313-win_amd64.whl.metadata (9.3 kB)
  Collecting pyparsing>=2.3.1 (from matplotlib)
  Downloading pyparsing-3.2.1-py3-none-any.whl.metadata (5.0 kB)
  Collecting python-dateutil>=2.7 (from matplotlib)
  Using cached python_dateutil-2.9.0-py2.py3-none-any.whl.metadata (3.0 kB)
  
```

Figure 1.4: Installing Libraries and Jupyter Notebook

Step 5: Open Jupyter Notebook using the command `jupyter notebook` and create files in `.ipynb` format.

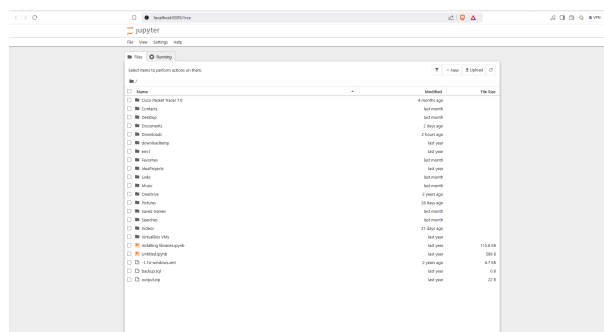
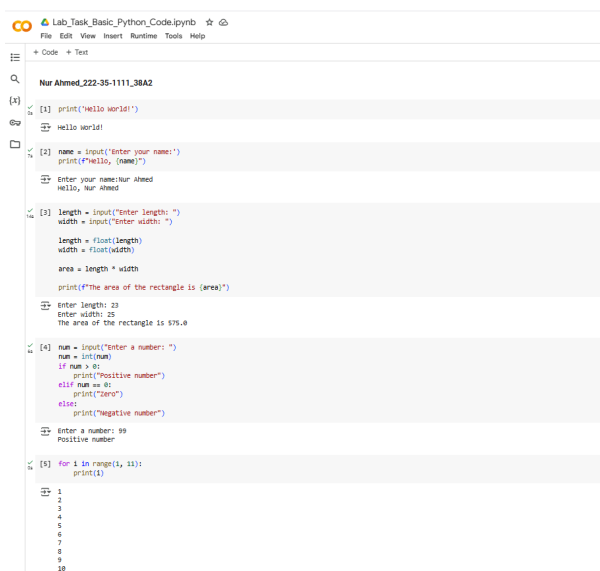


Figure 1.5: Opening Jupyter Notebook

Chapter 2

Lab-2: Basic Python Code

Here are the screenshots of the basic Python codes that I have practiced with outputs:



```
Lab_Task_Basic_Python_Code.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text

Nur Ahmed,222-35-1111,38A2

[1] print('hello world!')
hello world!

[2] name = input('Enter your name:')
print(f'Hello, {name}')
Enter your name:Nur Ahmed
Hello, Nur Ahmed

[3] length = input('Enter length: ')
width = input('Enter width: ')
length = float(length)
width = float(width)
area = length * width
print(f'The area of the rectangle is {area}')
Enter length: 23
Enter width: 25
The area of the rectangle is 575.0

[4] num = input('Enter a number: ')
num = int(num)
if num > 0:
    print("Positive number")
elif num == 0:
    print("Zero")
else:
    print("Negative number")
Enter a number: 99
Positive number

[5] for i in range(1, 11):
    print(i)
1
2
3
4
5
6
7
8
9
10
```

Figure 2.1: Basic Python Code Output 1


```

Lab_Task_Basic_Python_Code.ipynb
File Edit View Insert Runtime Tools Help

+ Code + Text

[6] def add_numbers(a, b):
    return a + b

num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))
result = add_numbers(num1, num2)
print(f"The sum is {result}")

Enter first number: 2
Enter second number: 5
The sum is 7.0

def add(a, b):
    return a + b

def subtract(a, b):
    return a - b

def multiply(a, b):
    return a * b

def divide(a, b):
    return a / b

print("Options: 1-Add, 2-Subtract, 3-Multiply, 4-Divide")
choice = int(input("Enter your choice: "))
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))

if choice == 1:
    print("Result:", add(num1, num2))
elif choice == 2:
    print("Result:", subtract(num1, num2))
elif choice == 3:
    print("Result:", multiply(num1, num2))
elif choice == 4:
    if num2 != 0:
        print("Result:", divide(num1, num2))
    else:
        print("Error: Division by zero")
else:
    print("Invalid choice")

Options: 1-Add, 2-Subtract, 3-Multiply, 4-Divide
Enter your choice: 2
Enter first number: 9
Enter second number: 4
Result: 5.0

```

Figure 2.2: Basic Python Code Output 2

```

Lab_Task_Basic_Python_Code.ipynb
File Edit View Insert Runtime Tools Help

+ Code + Text

[8] def add(a, b):
    return a + b

def subtract(a, b):
    return a - b

def multiply(a, b):
    return a * b

def divide(a, b):
    if b == 0:
        raise ZeroDivisionError("Division by zero is not allowed.")
    return a / b

[9] print("Options: 1-Add, 2-Subtract, 3-Multiply, 4-Divide")
choice = int(input("Enter your choice: "))
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))

if choice == 1:
    print("Result:", add(num1, num2))
elif choice == 2:
    print("Result:", subtract(num1, num2))
elif choice == 3:
    print("Result:", multiply(num1, num2))
elif choice == 4:
    if num2 != 0:
        print("Result:", divide(num1, num2))
    else:
        print("Error: Division by zero")
else:
    print("Invalid choice")

Options: 1-Add, 2-Subtract, 3-Multiply, 4-Divide
Enter your choice: 1
Enter first number: 7
Enter second number: 9
Result: 16.0

Start coding or generate with AI.

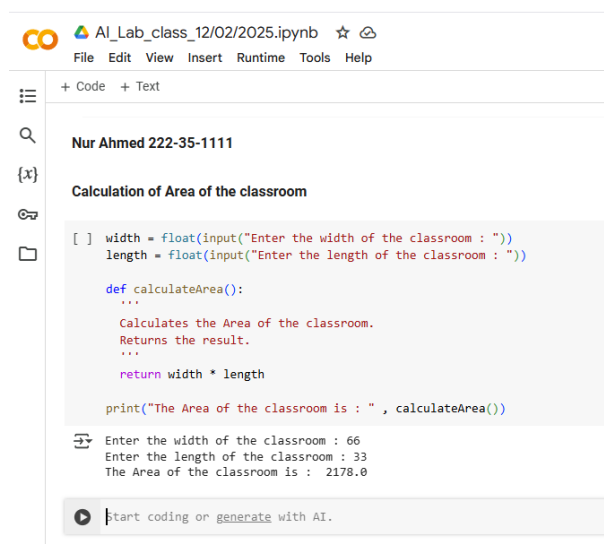
```

Figure 2.3: Basic Python Code Output 3

Chapter 3

Lab-3: Document String in Python

Here is the screenshot of the usage of Document String in Python:



The screenshot shows a Jupyter Notebook interface with a file named 'AI_Lab_class_12/02/2025.ipynb'. The notebook contains a Python script with a docstring for a function named 'calculateArea()'. The docstring describes the function's purpose and return value. The script also includes input prompts for width and length, and a print statement to display the calculated area.

```
[ ] width = float(input("Enter the width of the classroom : "))
length = float(input("Enter the length of the classroom : "))

def calculateArea():
    """
    Calculates the Area of the classroom.
    Returns the result.
    """
    return width * length

print("The Area of the classroom is : " , calculateArea())
```

Enter the width of the classroom : 66
Enter the length of the classroom : 33
The Area of the classroom is : 2178.0

Start coding or generate with AI.

Figure 3.1: Usage of Document String in Python

Chapter 4

Lab-4: Implementation of OOP in Python

Here is the screenshot of the implementation of OOP in Python:

```
class Car:
    def __init__(self, color, mileage):
        self.color = color
        self.mileage = mileage

    def drive(self):
        '''Drive the car'''
        print(f"The {self.color} car goes faster.")

my_car = Car("red", 37281)
print(my_car.mileage)
my_car.drive()

red
37281
The red car drives.
```

```
Getters and Setters

[ ] # Book: Title,
class Book:
    def __init__(self):
        pass

    def get_title(self):
        '''Get the title of the book'''
        return self.title

    def set_title(self, title):
        '''Set the title of the book'''
        self.title = title

book = Book()
book.set_title("GO There")
print(book.get_title())

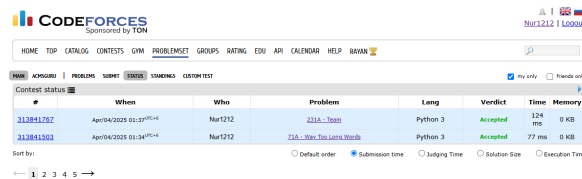
GO There
```

Figure 4.1: Creating Class and Object along with Getter and Setter in Python

Chapter 5

Lab-5: Solving two problems in Codeforces using Python

Here is the screenshot of the two problems that I solved in Codeforces using Python:



The screenshot shows the Codeforces website interface. At the top, there's a navigation bar with links like HOME, TOP, CATALOG, CONTESTS, GYM, PROBLEMS, GROUPS, RATING, EJO, API, CALENDAR, HELP, and BAYAN. Below this, there's a section for 'Contest status' with a table of submissions. The table has columns for ID, When, Who, Problem, Lang, Verdict, Time, and Memory. Two submissions are listed, both with a verdict of 'Accepted'.

#	When	Who	Problem	Lang	Verdict	Time	Memory
313841262	Apr/04/2025 01:32:00+04	Nur1212	231A - Team	Python 3	Accepted	124 ms	0 KB
313841263	Apr/04/2025 01:34:00+04	Nur1212	231A - Team	Python 3	Accepted	77 ms	0 KB

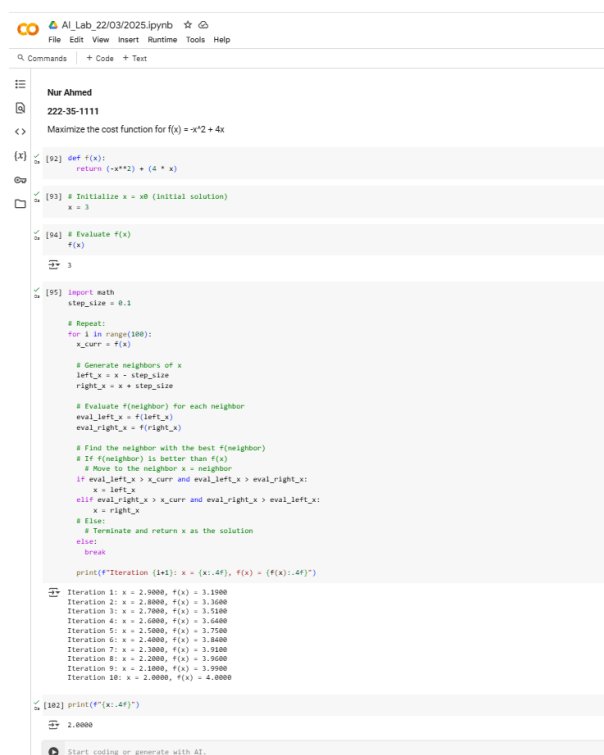
Sort by: ☐ Default order ☒ Submission time ☐ Judging Time ☐ Solution Size ☐ Execution Time

Figure 5.1: Two mild difficult problems solved in Codeforces using Python

Chapter 6

Lab-6: Maximizing the cost of $f(x) = -x^2 + 4x$ using Hill Climbing algorithm

Here is the screenshot of the implementation of Hill Climbing algorithm for maximizing cost in Python:



```
AI Lab 22/03/2025.ipynb
File Edit View Insert Runtime Tools Help
Commands Code Test

Nur Ahmed
222-35-1111
Maximize the cost function for f(x) = -x^2 + 4x

In [92]: def f(x):
         return (-x**2) + (4 * x)

In [93]: # Initialize x = x0 (initial solution)
         x = 3

In [94]: # Evaluate f(x)
         f(x)
         3

In [95]: import math
         step_size = 0.1

         # Repeat:
         for i in range(100):
             x_curr = f(x)

             # Generate neighbors of x
             left_x = x - step_size
             right_x = x + step_size

             # Evaluate f(neighbors) for each neighbor
             eval_left_x = f(left_x)
             eval_right_x = f(right_x)

             # Find the neighbor with the best f(neighbors)
             # If f(neighbors) is better than f(x)
             # Move to the neighbor x = neighbor
             if eval_left_x > x_curr and eval_left_x > eval_right_x:
                 x = left_x
             elif eval_right_x > x_curr and eval_right_x > eval_left_x:
                 x = right_x
             # Else:
             # Terminate and return x as the solution
             else:
                 break

             print(f"Iteration [{i+1}]: x = {x:.4f}, f(x) = {f(x):.4f}")

         Iteration 1: x = 2.0000, f(x) = 3.1000
         Iteration 2: x = 2.8000, f(x) = 3.3600
         Iteration 3: x = 2.7000, f(x) = 3.5100
         Iteration 4: x = 2.6000, f(x) = 3.6400
         Iteration 5: x = 2.5000, f(x) = 3.7500
         Iteration 6: x = 2.4000, f(x) = 3.8400
         Iteration 7: x = 2.3000, f(x) = 3.9100
         Iteration 8: x = 2.2000, f(x) = 3.9600
         Iteration 9: x = 2.1000, f(x) = 3.9900
         Iteration 10: x = 2.0000, f(x) = 4.0000

In [102]: print(f"({x:.4f})")
         2.0000

Start coding or generate with AI.
```

Figure 6.1: Hill Climbing algorithm for maximizing cost in Python

Chapter 7

Lab-7: Additional Lab Classes

Here are the screenshots of the additional lab class activities (Pandas and Matplotlib Libraries):

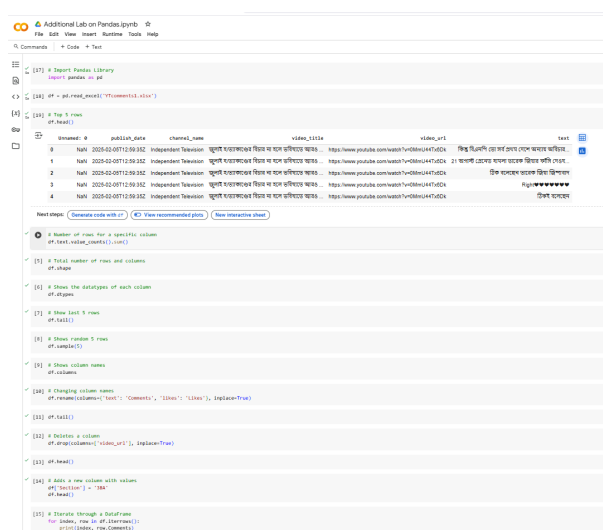


Figure 7.1: Using Pandas Library in Python



Additional Lab Class on Matplotlib.ipynb ☆ Saving failed since 12:59 AM

File Edit View Insert Runtime Tools Help

Commands + Code + Text

```
[16] %matplotlib inline
# Import Matplotlib
import matplotlib.pyplot as plt

[17] import numpy as np

Line plot

[18] # Creating an empty list
l1 = []

# Adding elements to the list
for i in range(10):
    l1.append(i/2)
l1

[19] # Generating numbers between 1 to 39
x = range(1, 40)
y = [i/2 for i in x]

plt.plot(x, y, 'o')
plt.show()

[20] # Creating a plot figure
fig = plt.figure()

# Creating an array of 0 to 30 numbers with 500 points
x = np.linspace(0, 30, 500)

# Plots the x
plt.plot(x, np.sin(x))

[21] plt.plot(x, np.cos(x))

x = np.linspace(0, 10, 1000)
plt.plot(x, np.sin(x), x, np.cos(x))

plt.xlabel('x')
plt.ylabel('y')
plt.title("Sin/Cos Plot")
plt.legend(['sin(x)', 'cos(x)'])

plt.grid(True) # Add grids
plt.show()

x = np.linspace(0, 10, 10)
# creates a plot figure
```

Figure 7.2: Using Matplotlib Library in Python