National University of Computer & Emerging Sciences, Karachi
Computer Science Department
Spring 2022, Lab Manual – 02

Course Code: AI-2002	**Course : Artificial Intelligence Lab**
Instructor(s):	**Kariz Kamal, Erum Shaheen, Mafaza Mohi, Danish Waseem, Ali Fatmi**

Contents:

Objective

1. Introduction to Some Useful Python Libraries, understanding need of those libraries.
2. To acquire programming skills in Python and its relevant libraries.
3. Solve some basic AI problem using the python programming language.

Introduction to Python Libraries

A Python library is a reusable chunk of code that you may want to include in your programs/ projects. Compared to languages like C++ or C, a Python libraries do not pertain to any specific context in Python. Here, a 'library' loosely describes a collection of core modules. Essentially, then, a library is a collection of modules. A package is a library that can be installed using a package manager like rubygems or npm.

Python libraries list that will take you places in your journey with Artificial Intelligence. These are also the libraries for Data Science, Machine Learning, Deep Learning, Graph learning etc.

1. numpy

NumPy is a module for Python. The name is an acronym for "Numeric Python" or "Numerical Python". It is an extension module for Python, mostly written in C. This makes sure that the precompiled mathematical and numerical functions and functionalities of Numpy guarantee great execution speed.
Furthermore, NumPy enriches the programming language Python with powerful data structures, implementing multi-dimensional arrays and matrices. These data structures guarantee efficient calculations with matrices and arrays. The implementation is even aiming at huge matrices and arrays, better known under the heading of "big data". Besides that the module supplies a large library of high-level mathematical functions to operate on these matrices and arrays.

You can include the given code in your ipython notebook to include functionalities of Numpy:

```
#Code#1 multiplying the numpy #array
a(matrix) by 2
import numpy as np
a = np.array([[1,2,3],[4,5,6]])
b = 2*a
print(b)
```

```
#Code#2 Convert a 1-D array into a
3_D array
import numpy as np
a = np.array([x for x in range(27)])
o = a.reshape((3,3,3))
print(o)
```

```
#Code#3 multiplying the numpy #array
a(matrix) by 2
import numpy as np
np.random.seed(123) # setting the seed
o = np.random.randint(0, 10, size = (5,5))
print(o)
```

```
#Code#4 Create a NumPy ndarray Object
import numpy as np
np.random.seed(123) # setting the seed
o = np.random.randint(0, 10, size = (5,5))
print(o)
```

```
#Code#5 Use a tuple to create a
NumPy array:
import numpy as np
arr = np.array((1, 2, 3, 4, 5))
print(arr)
```

```
#Code#6 Check dimensions the arrays
import numpy as np
a = np.array(42)
b = np.array([1, 2, 3, 4, 5])
c = np.array([[1, 2, 3], [4, 5, 6]])
d = np.array([[[1, 2, 3], [4, 5, 6]],
 [[1, 2, 3], [4, 5, 6]]])
print(a.ndim)
print(b.ndim)
print(c.ndim)
print(d.ndim)
```

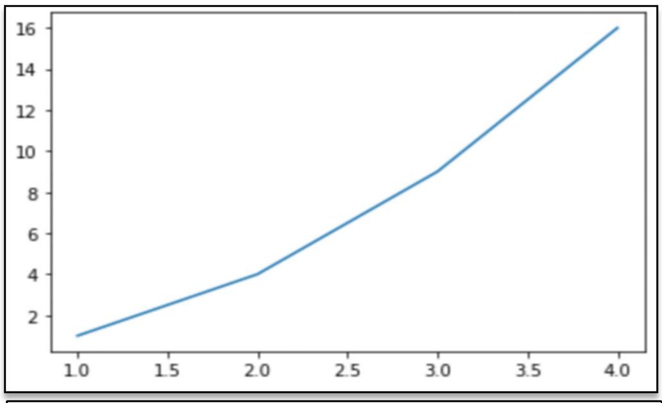
2. b.Matplotlib

It is a very powerful plotting library useful for those working with Python and NumPy. The most used module of Matplotib is Pyplot which provides an interface like MATLAB but instead, it uses Python and it is open source

What is PIP

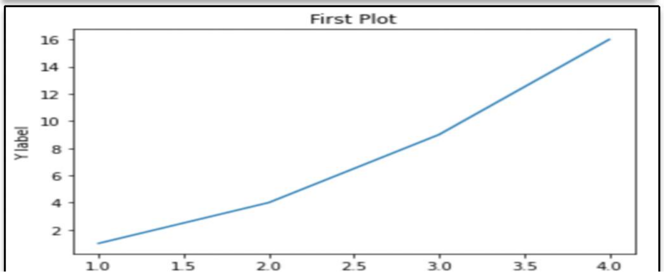
PIP is a package manager for Python packages, or modules if you like. Note: If you have Python version 3.4 or later, PIP is included by default.

```
#Code#7 Installation Packages
pip install -U pip
pip install matplotlib
```

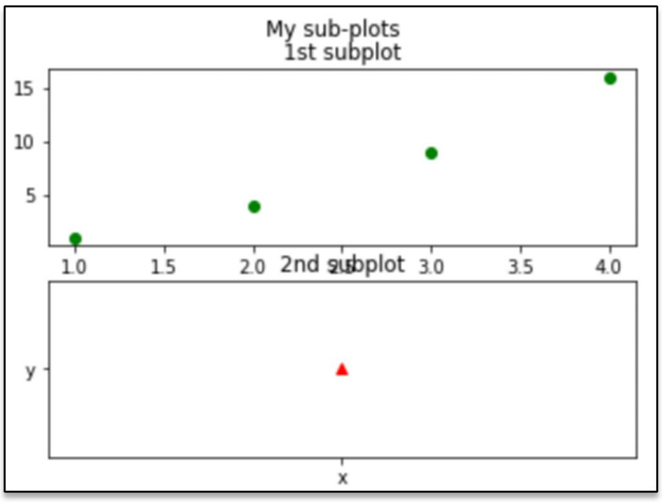


```
#Code#7 a simple plot using array
import matplotlib.pyplot as plt
import numpy as np
plt.plot([1,2,3,4],[1,4,9,16])
plt.show()
```

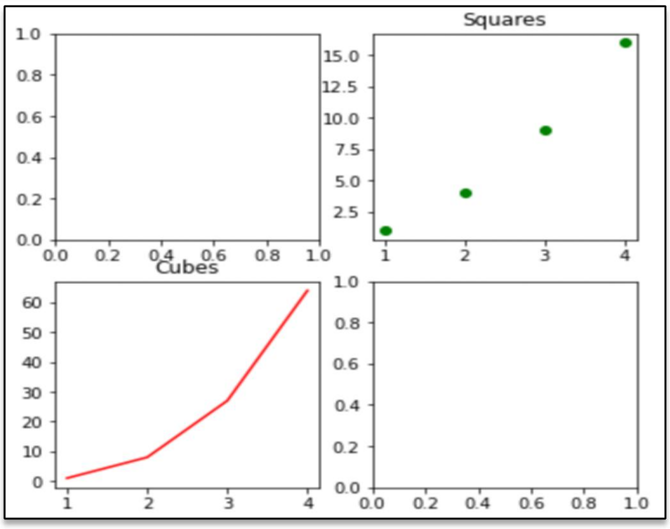
```
#Code#7 X-axis , Y-Axis and Title
import matplotlib.pyplot as plt
import numpy as np
plt.title("First Plot")
plt.xlabel("X Label")
plt.ylabel("Y Label")
plt.show()
```



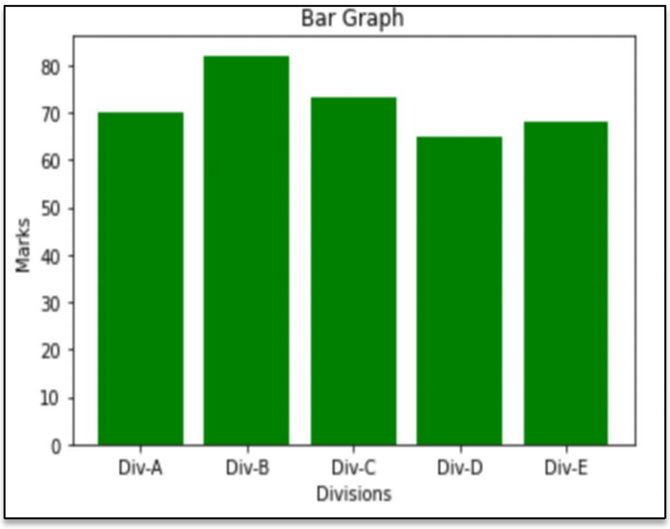
```
#Code#9 Two Plot

import matplotlib.pyplot as plt
import numpy as np
plt. subplot(2,1,1)
plt.plot([1,2,3,4],[1,4,9,16],"go")
plt.title("1st subplot")
plt.subplot(2,1,2)
plt.plot("x","y","r^")
plt.title("2nd subplot")
plt.suptitle("My sub-plots")
plt.show()
```



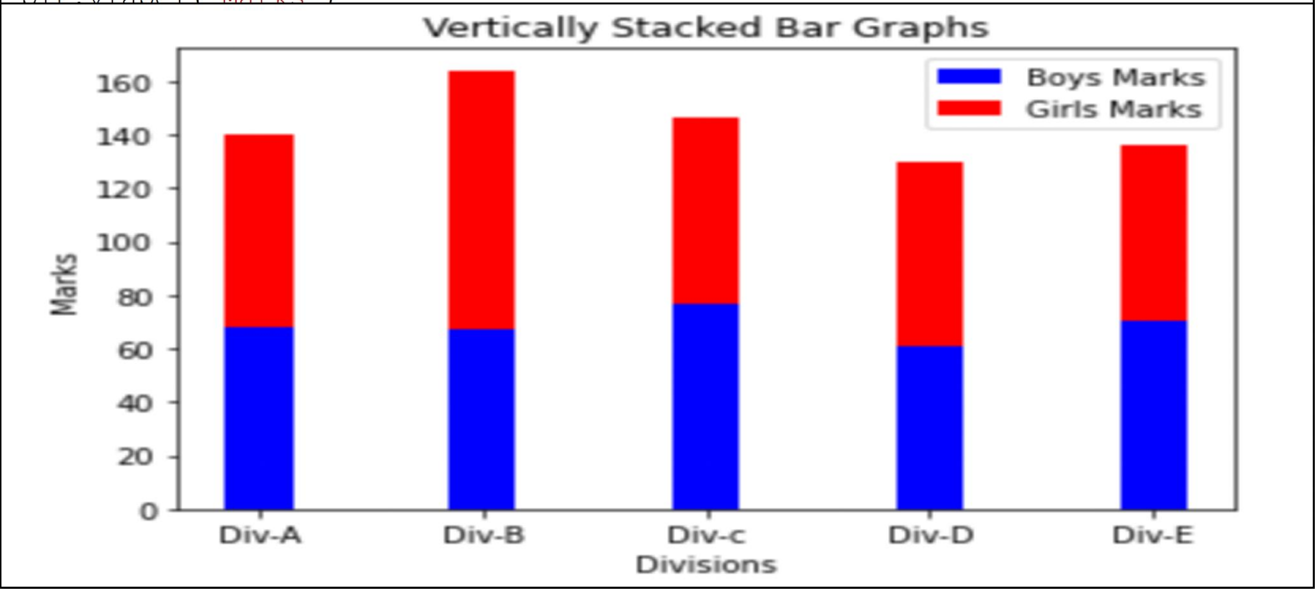
```
#Code#10 Square Plot
import matplotlib.pyplot as plt
import numpy as np
x=np.arange(1,5)
y=x**3
fig,ax=plt.subplots(nrows=2,ncols=2,
figsize=(6,6))
ax[0,1].plot([1,2,3,4],[1,4,9,16], "
go")
ax[1,0].plot(x,y, 'r')
ax[0,1].set_title("Squares")
ax[1,0].set_title("Cubes")
plt. show
```



```
#Code#11 Bar Graph
import matplotlib.pyplot as plt
import numpy as np
divisions = ["Div-A", "Div-B", "Div-
C", "Div-D", "Div-E"]
division_average_marks = [70, 82, 73
, 65, 68]
plt.bar (divisions, division_average
_marks, color='green')
plt.title("Bar Graph")
plt.xlabel("Divisions")
plt.ylabel("Marks")
plt.show()
```

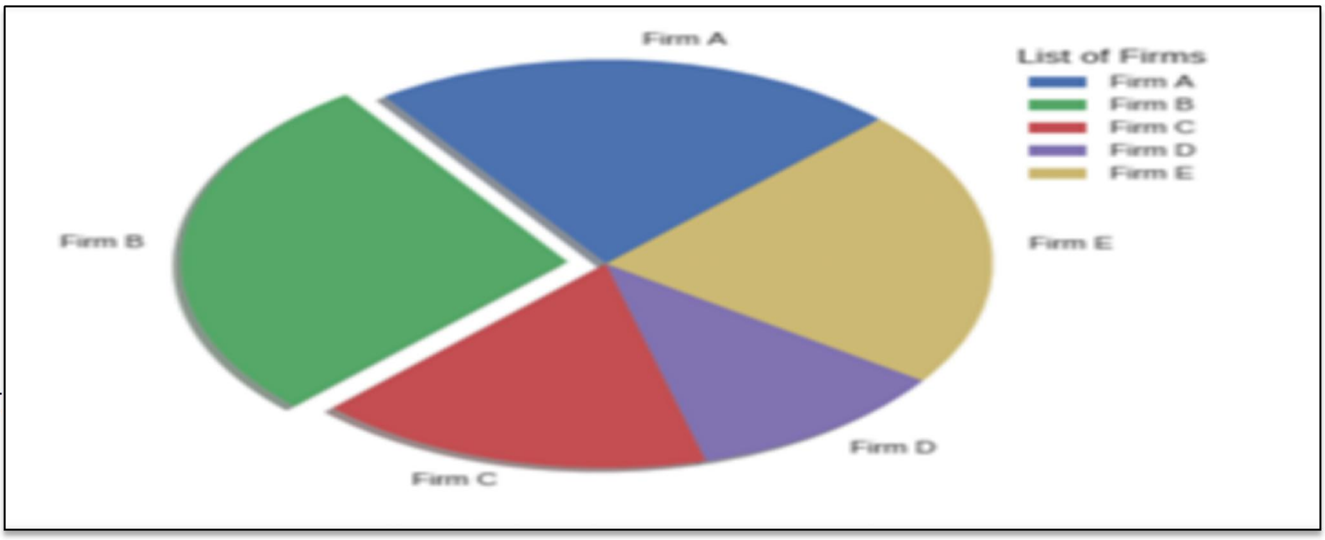


```
#Code#13 Vertically  stacked Bar Graph
import matplotlib.pyplot as plt
import numpy as np
divisions =["Div-A", "Div-B", "Div-c", "Div-D", "Div-E"]
boys_average_marks = [68, 67, 77, 61, 70]
girls_average_marks = [72, 97, 69, 69, 66]
index = np.arange(5)
width = 0.30
plt.bar(index, boys_average_marks, width, color="blue", label="Boys Marks")

plt.bar(index, girls_average_marks, width, color="red", label="Girls Marks"
, bottom=boys_average_marks)
plt. title("Vertically Stacked Bar Graphs")
plt.xlabel("Divisions")
plt.ylabel("Marks")
```



```
#Code#14 Pie Charts
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.pyplot as plt
import numpy as np
firms = ["Firm A", "Firm B", "Firm C", "Firm D","Firm E"]
market_share = [20, 25, 15, 10, 20]
Explode = [0,0.1,0,0,0]
plt.pie(market_share, explode=Explode, labels=firms, shadow=True, startangl
e=45)
plt.axis('equal')
plt.legend (title="List of Firms")
plt.show()
```

T



3. Spacy

Spacy is an open-source software python library used in advanced natural language processing and machine learning. It will be used to build information extraction, natural language understanding systems, and to pre-process text for deep learning

Features Of Spacy:

a. Non-destructive tokenization
b. Named entity recognition
c. Support for 61+ languages
d. Part-of-speech tagging
e. Built-in visualizers for syntax and NER
f. Export to NumPy data arrays

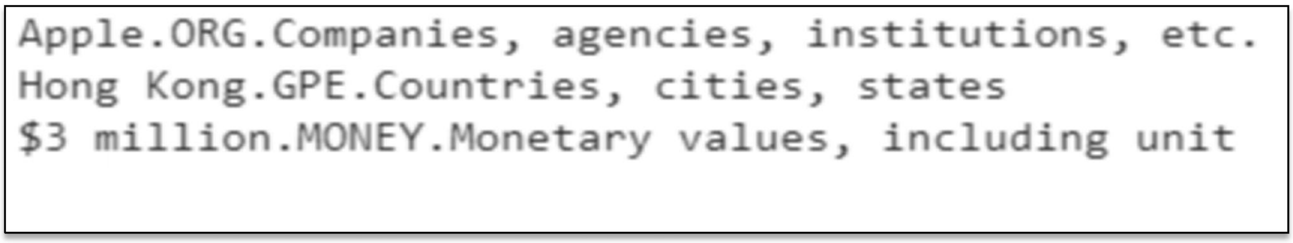
Install Spacy: pip install spacy

```
#Code#15 Working with Spacy
import spacy
nlp = spacy.load('en_core_web_sm')
string = '"I\'m with you for the rest of my life in PKK.!"'
print(string)
```

```
#Code#16 Working with Tokens in string
import spacy
nlp = spacy.load('en_core_web_sm')
string = '"I\'m with you for the rest of my life in PKK.!"'
doc= nlp(string)
for token in doc:
  print (token.text, end=' | ')
```

```
#Code#16 Working with Tokens using loop in string
import spacy
nlp = spacy.load('en_core_web_sm')
string = '"I\'m with you for the rest of my life in PKK.!"'
doc= nlp(string)
for token in doc:
  print (token)
```

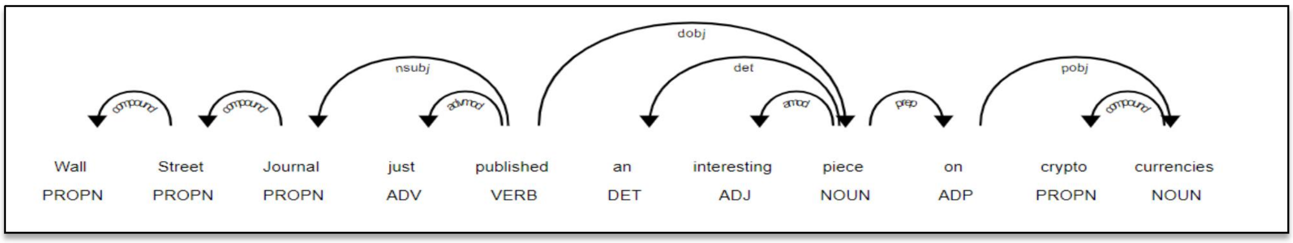
```
#Code#17 Working with Tokens using loop in string
import spacy
doc8=nlp(u"Apple to build a Hong Kong factory for $3 million")
for ent in doc8.ents:
  print(ent.text+ '.'+ent. label_+ '.' +str(spacy.explain(ent.label_)))
len(doc8.ents)
print(doc8.ents)
```



```
Apple.ORG.Companies, agencies, institutions, etc.
Hong Kong.GPE.Countries, cities, states
$3 million.MONEY.Monetary values, including unit
```

A modern syntactic dependency visualizer. Visualize spaCy's guess at the syntactic structure of a sentence. Arrows point from children to heads, and are labelled by their relation type.

```
#Code#18 Working with Tokens using loop in string
from spacy import displacy
doc = nlp('Wall Street Journal just published an interesting piece on cry
pto currencies')
displacy.render(doc, style='dep', jupyter=True, options={'distance': 90})
```



5. Aima3 Library

aima3 stands for 'Artificial Intelligence: A Modern Approach'

Install pip install aima3

THE VACUUM WORLD

In this notebook, we will be discussing the structure of agents through an example of the vacuum agent. The job of AI is to design an agent program that implements the agent function: the mapping from percepts to actions. We assume this program will run on some sort of computing device with physical sensors and actuators: we call this the architecture:

agent = architecture + program

Random Agent Program

A random agent program, as the name suggests, chooses an action at random, without taking into account the percepts.Here, we will demonstrate a random vacuum agent for a trivial vacuum environment, that is, the two-state environment. Let's begin by importing all the functions from the agents module:

```
from aima3.agents import *
from aima3.notebook import psource
#Let us first see how we define the TrivialVacuumEnvironment.
#Run the next cell to see how abstract class TrivialVacuumEnvironment is
defined in agents module.
```

```
class TrivialVacuumEnvironment(Environment):

"""This environment has two locations, A and B. Each can be Dirty or
Clean. The agent perceives its location and the location's status. This
serves as an example of how to implement a simple Environment."""

def init (self): super(). init ()
self.status = {loc_A: random.choice(['Clean', 'Dirty']), loc_B:
random.choice(['Clean', 'Dirty'])}

def thing_classes(self):
return [Wall, Dirt, ReflexVacuumAgent, RandomVacuumAgent,
TableDrivenVacuumAgent, ModelBasedVacuumAgent]

def percept(self, agent):
"""Returns the agent's location, and the location status
(Dirty/Clean)."""
return (agent.location, self.status[agent.location])

def execute_action(self, agent, action):
"""Change agent's location and/or location's status; track performance.
Score 10 for each dirt cleaned; -1 for each move."""
if action == 'Right': agent.location = loc_B agent.performance -= 1
elif action == 'Left': agent.location = loc_A

agent.performance -= 1 elif action == 'Suck':
if self.status[agent.location] == 'Dirty': agent.performance += 10
self.status[agent.location] = 'Clean'

def default_location(self, thing):
"""Agents start in either location at random."""
return random.choice([loc_A, loc_B]):
```

```
# These are the two locations for the two-state environment
   loc_A, loc_B = (0, 0), (1, 0)

# Initialize the two-state environment
   trivial_vacuum_env = TrivialVacuumEnvironment()

# Check the initial state of the environment
   print("State of the Environment: {}.".format(trivial_vacuum_env.status))
      State of the Environment: {(0, 0): 'Dirty', (1, 0): 'Clean'}.
```

TASKS

TASKS

1) Create an array with 5 dimensions and verify that it has 5 dimensions.

2) Plot the histogram showing the heights of 250 people using the random module's normal method.

3) Plot a scatter diagram to show the disadvantages of smoking with increasing age and weight.

The age vector: [10, 20, 30, 40, 50]
The weight vector: [35, 45, 55, 65, 75].

4) Consider an *Interactive Cognitive Environment (ICE)* in which autonomous robot is performing cleaning task in the big room that appears to be a matrix of ***N * M***. Each index referred to as a cell of the matrix is valued as dirty "*D*" or clean "*C*". The cells which are occupied by the stuff in the room are blocked and valued "B". The vacuum can move in all four directions (up, down, left, right), and if the cell status is D, it will clean the cell and change the status to "C", if the cell status is either C, it will not enter the cell. The vacuum will stop working if the whole room is cleaned, i.e., the status of all the cells is either C. The vacuum may start cleaning the room from the first cell (0, 0) or any random location. You will trace the path of the vacuum and display at each step of the program. * Represent the location of the vacuum cleaner. Develop a Python code of the above describe scenario of the autonomous robot.

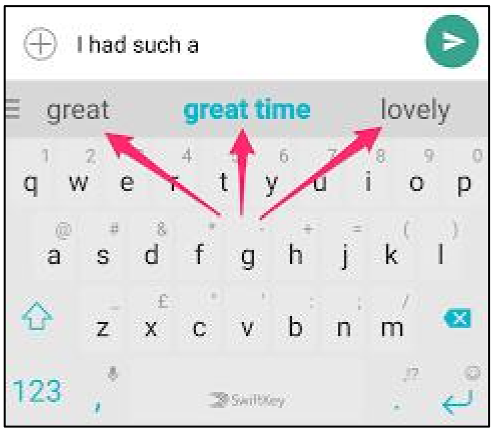
D*	D	D	D	D	D	D	■
■	■	C	D	■	■	■	■
D	D	C	D	■	D	D	D
D	D	D	D	C	C	C	D
D	D	D	D	D	D	D	D
D	D	D	D	D	D	D	D
D	D	D	■	■	D	D	D

$\rightarrow$

C	C	C	C	C	C	C	■
■	■	C	C	■	■	■	■
D	D	C	C	■	D	D	D
D	D	D	C	C	C	C	D
D	D	D	C*	D	D	D	D
D	D	D	D	D	D	D	D
D	D	D	■	■	D	D	D

If vacuum is in a location where it's all neighbors (up, down, left and right) are clean (with status C) it will move in any one of the directions and keep searching the Dirt (D). It will stop it execution if it does not sense any dirt after 10 movements.
If vacuum is in a location where it's one more neighbor (up, down, left and right) is dirty it will move in any one of the directions and will return to the location when it cleans all the dirty cell of its neighbors. e.g., cell (0, 3) where it's three neighbors are dirty.

5) Next Word Prediction or what is also called Language Modeling is the task of predicting what word comes next. It is one of the fundamental tasks of NLP and has many applications. You might be using it daily when you write texts or emails without realizing it.



Your devices or applications keep the track of your writing style or text in memory and apply different algorithm and models. These models break the paragraph/text into group words e.g. group may contain 1 to N word based on the model. Your task is to build the same program for breaking the texts/paragraph in to group of words and counts which group of words occurring together in text.
e.g.
Sentence = 'the purpose of our life is to happy'
('The', 'purpose')
('Purpose', 'of')
('Of', 'our')
('Our', 'life')
('Life', 'is')
('is', 'to')
('To', 'happy')
Sentence = 'Whoever is happy will make others happy too'
('Whoever', 'is', 'happy')
('is', 'happy', 'will')
('happy', 'will', 'make')
('will', 'make', 'others')
('make', 'others', 'happy')
('others', 'happy', 'too')