Python for Data Science

Branch: B. Tech. Honours (AI and DS)

Subject Code: A000275(022)

By: Swati Dewangan, Assistant Professor, CSVTU, BHILAI

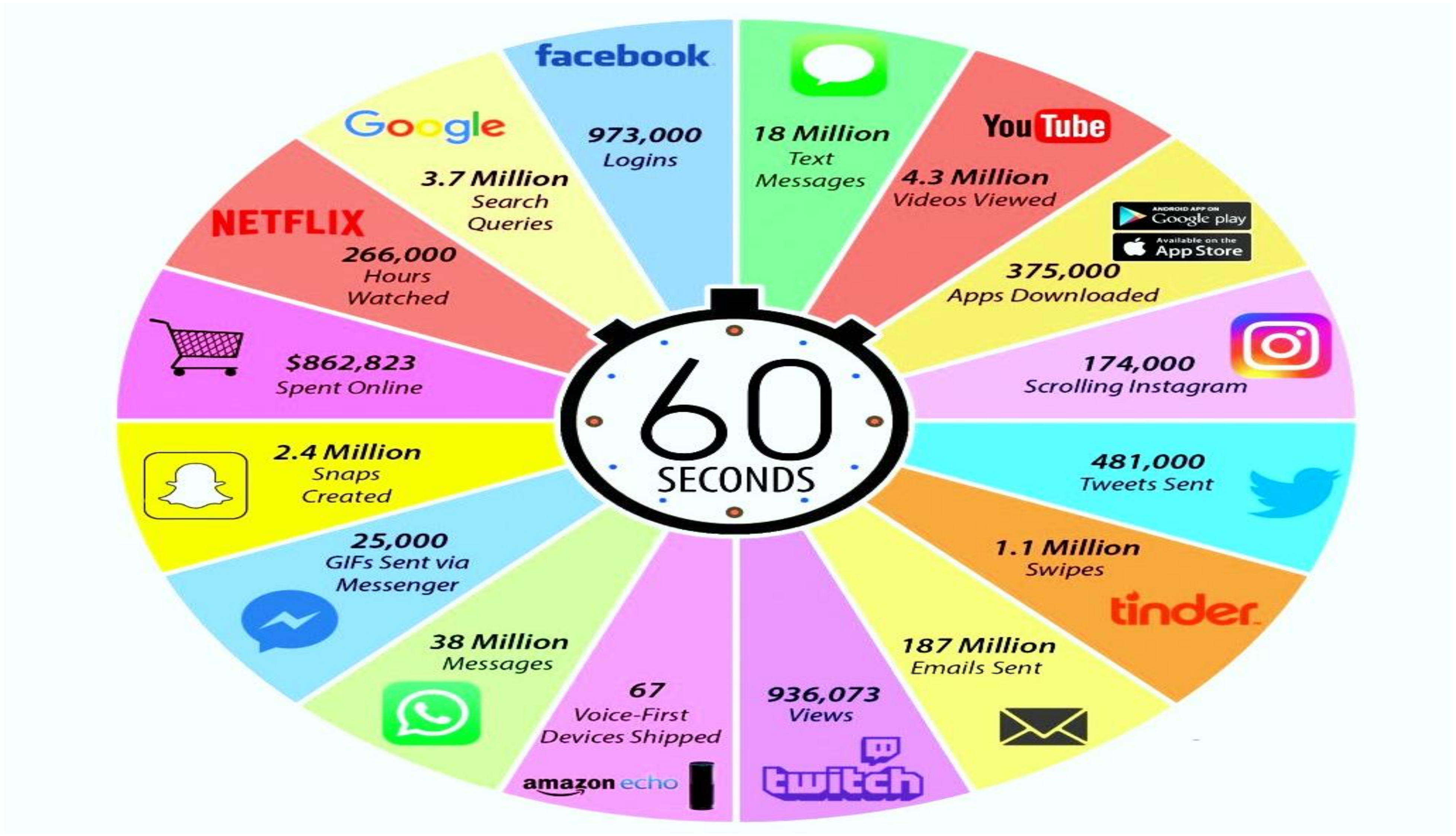
UNIT 1 : Introduction to Data Science and Python Programming

- Introduction to Data Science
- Why Python?
- Essential Python Libraries
- Python Introduction
- Features
- Identifiers
- Reserved Words
- Indentation
- Comments
- Type Conversion
- Operators

UNIT 1 : Introduction to Data Science and Python Programming

- Built in Data Types and their methods:
    - String
    - List
    - Tuples
    - Dictionary
    - Set
- Decision Making
- Looping
- Loop control Statement
- Math and Random Number Functions
- User-Defined Functions
- Function Arguments and its types

Introduction to Data Science



Introduction to Data Science

- ✓ **Internet/Online Data**
  - Clicks and searches, server requests, web logs, cell phone logs, mobile GPS locations, user generated content, etc.

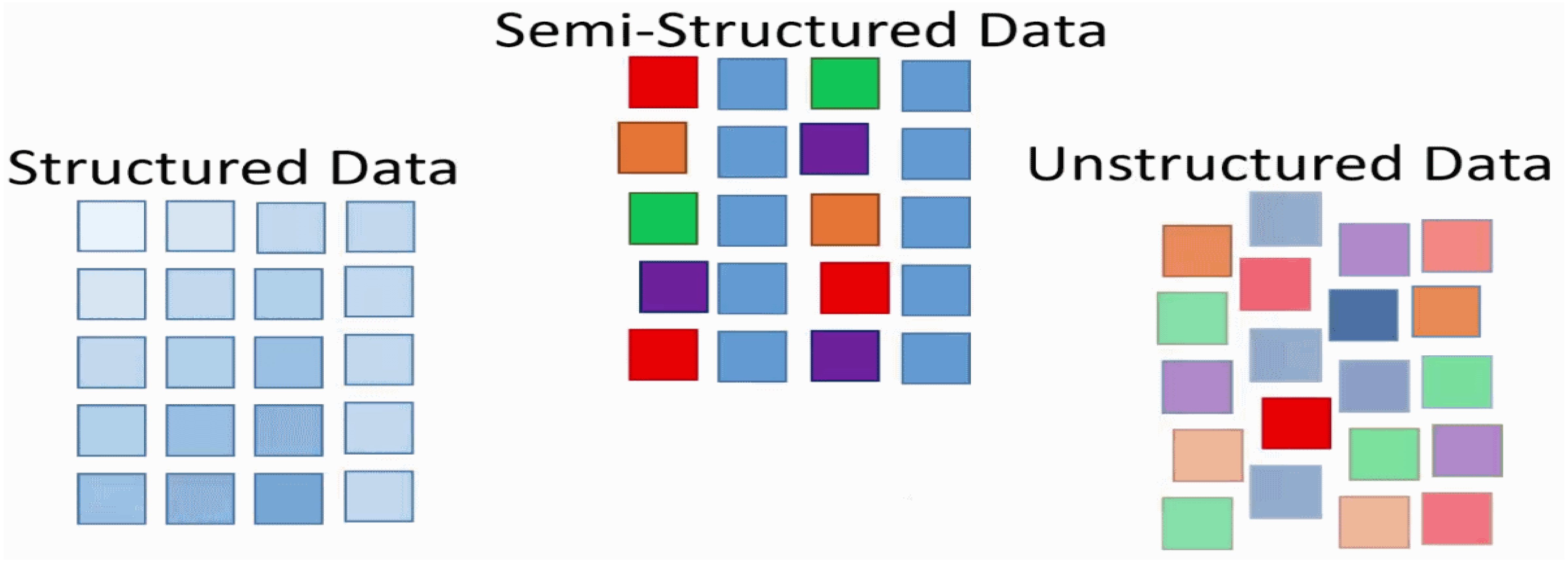
- ✓ **Healthcare**
  - Medical Images, Healthcare data, Billing data, Pharmaceutical industries, etc.

- ✓ **Telecommunications network**
  - Televisions, Mobile computing, Wireless Communications, etc.

- ✓ **Social networks**
  - Facebook, Twitter, LinkedIn, YouTube, Instagram, NetFlix, WhatsApp, Snap Chat, etc.)

- ✓ **Internet of Things**
  - Sensors (Military applications, Home applications, Disaster Management applications, etc.)

Types of Data



Types of Data

1. Structured Data

- Any data that can be stored in pre-specified tabular format.
- These data are well organized in database.
- Accessed and processed in the fixed format.
- **Example:**
  - ✓ Library Catalogues (date, author, place, subject, etc)
  - ✓ An 'Employee' table in a database as:

Employee_ID	Employee_Name	Gender	Department	Salary (Per Year)
2365	Rajesh Kulkarni	Male	Finance	650000
3398	Pratibha Joshi	Female	Admin	650000
7465	Shushil Roy	Male	Admin	500000
7500	Shubhojit Das	Male	Finance	500000
7699	Priya Sane	Female	Finance	550000

Types of Data

2. Un-structured Data

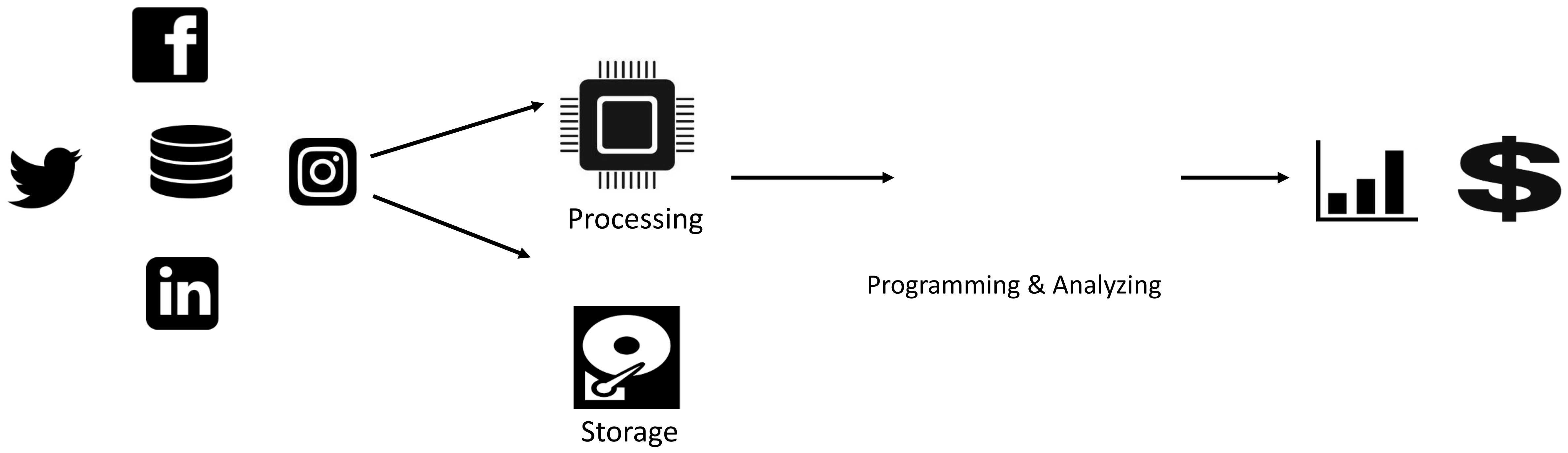
- Data which are not organized in a pre-defined format.
- They doesn't fit neatly in a database.
- Resides in Applications.
- **Example:**
  - ✓ Media (digital photos, audio and video files)
  - ✓ Social Media Posts (Data from Facebook, Twitter, Youtube)
  - ✓ Survey Responses.
  - ✓ Satellite data, Machine data, Sensor data

Types of Data

3. Semi-structured Data

- Semi-structured data cannot be stores in the formal structure of data tables.
- These data doesnot have well-defined structure.
- Maintains tags, markings to separate data elements.
- **Example:**
  - ✓ HTML page, E-mail document, etc.
  - ✓ Personal data stored in a XML file as:

```
<rec><name>Prashant Rao</name><gender>Male</gender><age>35</age></rec>
<rec><name>Seema R.</name><gender>Female</gender><age>41</age></rec>
<rec><name>Satish Mane</name><gender>Male</gender><age>29</age></rec>
<rec><name>Subrato Roy</name><gender>Male</gender><age>26</age></rec>
<rec><name>Jeremiah J.</name><gender>Male</gender><age>35</age></rec>
```


Processing
Storage
Programming & Analyzing

Data Science

- Data Science is the field that deals with:
  - ✓ **analysing huge vast volumes of data** – structured, semi-structured or unstructured data
  - ✓ **using modern tools and techniques** – different methods and algorithms
  - ✓ **to find meaningful information** - making predictions and business decisions.

Data Science

- ➢ By using Data Science, software companies are able to make:
  - ✓ Better decisions (should we choose A or B?)
  - ✓ Predictive analysis (what will happen next?)
  - ✓ Pattern discoveries (find pattern i.e. hidden information in the data)

Data Science

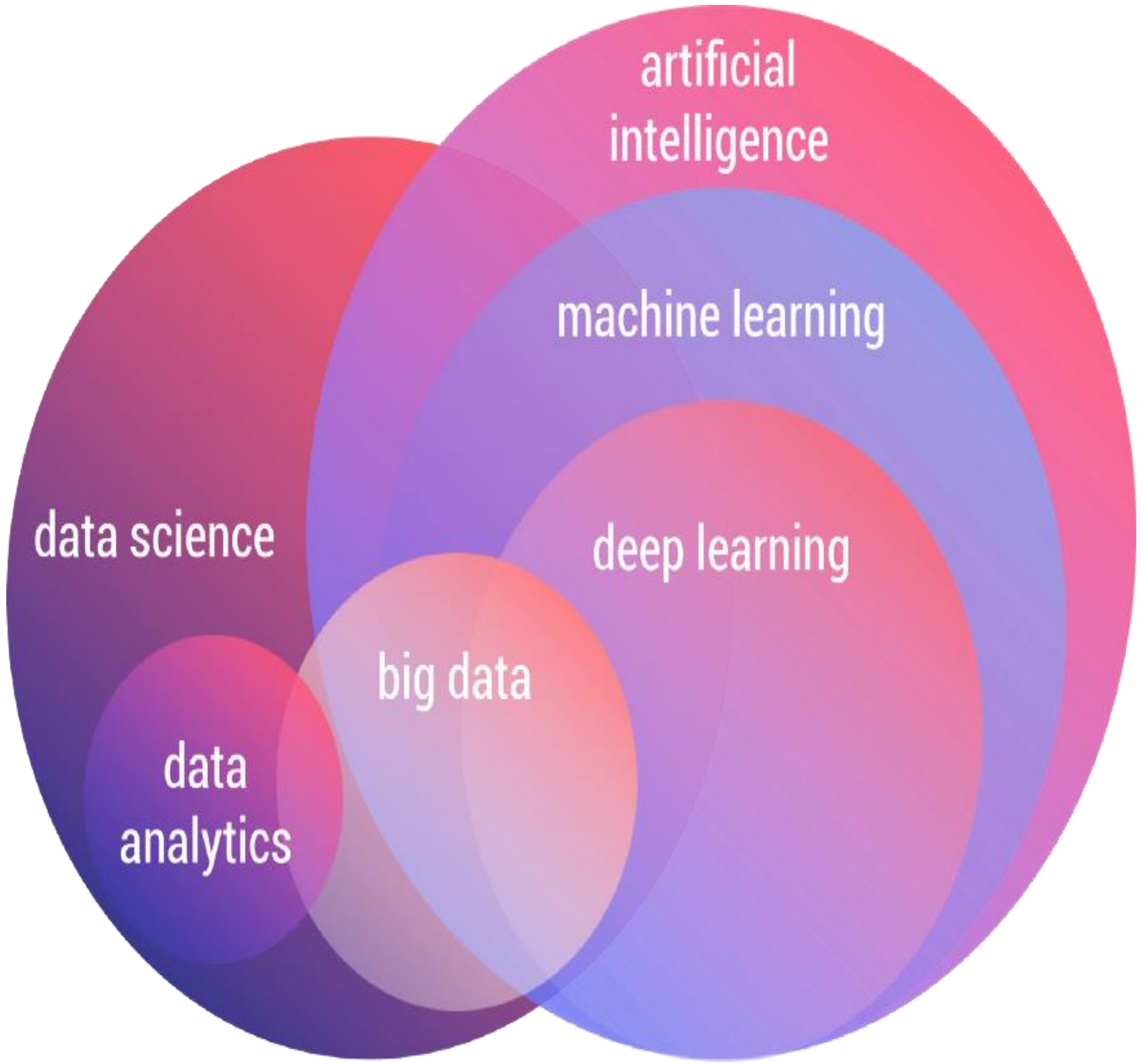
- ➢Data science practitioners:
- ✓apply machine learning algorithms to numbers, text, images, video, audio, etc.
- ✓to produce artificial intelligence (AI) systems to perform tasks same as human intelligence.

How Data Science Works?
(Lifecycle of Data Science)

1. **Problem Statement:** correctly defining the problem to be solved.
2. **Data Collection:** gathers structured and unstructured data.
3. **Data Cleaning:** remove erroneous, missing, redundant, duplicate values from the data using tools - *PYTHON*
4. **Data Analysis:** Process of examining data to find hidden patterns using graphs.
5. **Data Modelling:** building a model using algorithm for accurate prediction of new data.
6. **Optimization:** test your data to check how well it is performing
7. **Deployment:** launching of the application for the end user.

Machine Learning

- Machine Learning is a **subset of Artificial Intelligence.**
- Allows **computer systems** to **learn from data using algorithms and make predictions**.
- **Make predictions about new and unknown data**.



Programming

- **Program - **an ordered **set of instructions to be executed by a computer** to carry out a specific task is called a program.
- **Programming Language - the language used to specify** this **set of instructions to the computer** is called a programming language.
- **Machine language -** uses 1s and 0s to write instructions which are directly understood and executed by the computer.

✓Low Level Languages – Machine Dependent, uses assembler as translator.

✓High Level Languages – Machine Independent, uses compiler or interpreter as translator.

Programming

- **Source code**

  A program written in a high-level language is called source code.

- **Object Code**

  The source code is converted by a translator into the machine understandable form called object (machine) code.

Programming

- **Language Translators**

  Language translators like compilers and interpreters are needed to translate the source code into machine language.

- ✓ **Assembler**

  The translator used to convert the code written in **assembly language to machine language** is called *assembler*.

- ✓ **Interpreter**

  An interpreter translates **one line at a time** instead of the whole program at one go.

- ✓ **Compiler**

  a compiler translates the **entire source code**, as a whole, into the object code. After scanning the whole program, it generates error messages, if any.

Introduction to Python

- Python is widely used multi-purpose high level programming language.
- It was invented by Guido van Rossum in 1991.
- There are two major Python versions:
  - ✓**Python 2**
  - ✓**Python 3**

Features of Python
(Why Python?)

- ✓ **Object Oriented** - Python supports Object-Oriented programming that encapsulates code within objects.
- ✓ **Interpreted language** - as the programs are executed by an interpreter, code can be executed line by line as soon as it is written.
- ✓ **Platform Independent -** Python works on different platforms such as Windows, MacOS, Linux etc.
- ✓ **Open source** – it is free to use and distribute, even for commercial purposes.

Features of Python
(Why Python?)

- ✓ **Portable** – Python can run on a wide variety of hardware platforms.
- ✓ **Less Complex** - allows developers to solve complex problems in less time with fewer lines of code than some other programming languages.
- ✓ **Versatile** - Python can be used for many different tasks, from web development to machine learning.
- ✓ **User-friendly syntax:** Python has a simple syntax similar to the English language so it’s easier to read and understand.

Features of Python (Why Python?)

- ✓ **Easy-to-learn** – Python has few keywords, simple structure, and a clearly defined syntax.
- ✓ **Easy to read and maintain** - It uses Indentation (whitespace) to define scope (such as the scope of loops, functions and classes). Other programming languages often use curly-brackets for this purpose.
- ✓ **Broad standard library** – It has rich libraries of pre-defined functions for numerous applications like:
  - ✓ Desktop Applications
  - ✓ Web Applications
  - ✓ Data Analysis and Preprocessing (text, images, videos, etc.)
  - ✓ Machine Learning
  - ✓ Artificial Intelligence
  - ✓ Data Science
  - ✓ Robotics
  - ✓ Gaming
  - ✓ Mobile Apps
- ✓ Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, etc.

Essential Python Libraries

- **Python Library**

  - A library is a **collection of utility methods, classes and modules** that your application code can use **to perform specific tasks** without writing the functionalities from scratch.

- **Why do we require libraries?**

  - For code reusability.
  - Code Reusability means using code that has already been written down by other people for our own purpose.
  - ***To list all the installed libraries in python:***

help("modules")

Essential Python Libraries

- **Pandas (Panel Data/ Python Data Analysis) -** They are mostly used for **analyzing, cleaning, exploring, and manipulating data**.
- **NumPy (Numerical Python) – enables** with collection **of mathematical functions to operate on array and matrices.**
- **SciPy (Scientific Python) - used for scientific computation**. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing.

Essential Python Libraries

- **Matplotlib** **-** It is a **data visualization** and graphical plotting library.
- **Seaborn** **-** It is an extension of Matplotlib library used **to create more attractive and informative statistical graphics**.
- **Scikit-learn** **-** It is a **machine learning library** that enables tools for used for many other machine learning algorithms such as classification, prediction, etc.
- **Tensorflow** **-** a collection of **multiple machine learning libraries to develop and train and test the models.**
- **Keras** **-** It is built on top of Tensorflow that uses neural network library **for implementing deep learning algorithms.**

Loading Python Libraries

```
#Import Python Libraries
import numpy as np
import scipy as sp
import pandas as pd
import matplotlib as mpl
import seaborn as sns
```

Python Keywords

- In programming, a keyword is a "**reserved word**" by the language which conveys **special meaning to the interpreter**.
- Python is case-sensitive. Example COMPUTER and computer is not same.
- Python has a set of keywords that are **reserved words that cannot be used as variable names, function names, or any other identifiers.**
- As Python is case sensitive, keywords used are given as:

False	await	else	import	pass
None	break	except	in	raise
True	class	finally	is	return
and	continue	for	lambda	try
as	def	from	nonlocal	while
assert	del	global	not	with
async	elif	if	or	yield

Identifiers

- ❑In programming languages, **identifiers are names used to identify a variable, function, or other entities in a program.**
- ❑The rules for naming an identifier in Python are as follows:

✓The name **should begin with:**

- an uppercase or a lowercase alphabet or an underscore sign (_).
- This may be followed by any combination of characters a–z, A–Z, 0–9 or underscore (_).
- Thus, an identifier cannot start with a digit.

✓It can be of any length. (However, it is preferred to keep it short and meaningful).

✓It should **not** be a keyword or reserved word.

✓Special symbols like !, @, #, $, %, etc., are **not** used in identifiers.

Identifiers

For Example:

- To calculate the area of a rectangle:
- Using identifier names, such as `area, length, breadth`
- Instead of single alphabets as identifiers for clarity and more readability.

```
area = length * breadth
```

Variables

- Variables are the names to the memory locations used for **storing data values while executing the program.**
- Variables **do not need to be declared with any particular data type**, and can even change type after they have been set.
- In Python we use **an assignment statement** **to create new variables** and assign specific values to them.
- Variable **declaration is implicit** in Python, means variables are automatically declared and defined when they are assigned a value the first time.
- Variables **must always be assigned values** **before they are used** in expressions as otherwise it will lead to an error in the program. Wherever a variable name occurs in an expression, **the interpreter replaces it with the value of that particular variable.**

Variables

Assigning Values to Variables

- To assign a value to a variable equal sign (=) is used.
- The operand to the left of the = operator is the variable name.
- The operand to the right of the = operator is the value stored in the variable.

```
counter = 100 # An integer assignment
miles = 1000.0 # A floating point
name = "John" # A string
print counter
print miles
print name
```

Variables

- **Rules for Python Variables:**

✓A variable name must start with a letter or the underscore character

✓A variable name cannot start with a number

✓A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)

✓Variable names are case-sensitive (age, Age and AGE are three different variables)

Variables

- Legal Variable Names:
  - myvar = “John”
  - my_var = “John”
  - _my_var = “John”
  - myVar = “John”
  - Myvar = “John”
  - MYVAR2 = “John”

- Illegal Variable Names:
  - 1myvar = “John”
  - $myvar = “John”
  - my-var = “John”
  - My var = “John”
  - myvar@ = “John”
  - 123myvar = “John”

Variables

Assigning One Value to Multiple Variables:

Python allows you to assign a single value to several variables simultaneously. For example –

```
a = b = c = “data science”
print(a)
print(b)
print(c)
```

Here, an integer object is created with the value 1, and all three variables are assigned to the same memory location.

Variables

Assigning Multiple Values to Multiple Variables

Python allows to assign multiple values to multiple variables.

For example –

```
x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)
```

Variables

- Write a program to display values of variables in Python.

- **Program:**

```
#To display values of variables
message = "Keep Smiling"
print(message)
userNo = 101
print('User Number is', userNo)
```

- **Output:**

```
Keep Smiling
User Number is 101
```

Variables

Output Variables

- The print() function is often used to output variables.

```
x = "Python for Data Science”
print(x)
```

- In the print() function, you output multiple variables, separated by a comma:

```
x = "Python"
y = "is"
z = "awesome"
print(x, y, z)
```

- You can also use the + operator to output multiple variables:

```
x = "Python "
y = "is "
z = "awesome"
print(x + y + z)
```

Variables

- For numbers, the + character works as a mathematical operator:

```
x = 5
y = 10
print(x + y)
```

- In the print() function, when you try to combine a string and a number with the + operator, Python will give you an error:

```
x = 5
y = "John"
print(x + y)
```

- The best way to output multiple variables in the print() function is to separate them with commas, which even support different data types:

```
x = 5
y = "John"
print(x, y)
```

Indentation

- ✓ Python uses indentation for block as well as for nested block structures.
- ✓ Leading whitespace (spaces and tabs) at the beginning of a statement is called indentation.
- ✓ The interpreter checks indentation levels very strictly and throws up syntax errors if indentation is not correct.
- ✓ It is a common practice to use a single tab for each level of indentation.

Comments

- Comments are **used to add a remark** or a note in the source code.
- Comments are **not executed by interpreter.**
- They are added with the purpose of **making the source code easier for humans to understand**.

Types of Comments:

A) Single line comment

✓ It starts with # (hash sign) , everything following the # till the end of that line is treated as a comment by the interpreter and simply ignores it while executing the statement.

B) Multiple line comment

- Multiline String (enclosed within triple double quotes or triple single quotes) can be used to place the comments inside it.

NOTE: If this line is assigned to a variable then it will be interpreted as string and hence the name Multiline String.

Using triple double quotes:

#This is a comment
#written in
#more than just one line
print("Hello, World!")
	"""
	This is a comment
	written in
	more than just one line
	"""
	print("Hello, World!")
		a="""
		This is a Multiline String
		written in
		more than just one line
		"""
		print("Hello, World!")
		print(a)

Output	**Output**	**Output**
Hello, World!	Hello World:	Hello, World! This is a Multiline String written in more than just one line

B) Multiple line comment

Using triple single quotes:

#This is a comment
#written in
#more than just one line
print("Hello, World!")
	'''
	This is a comment
	written in
	more than just one line
	'''
	print("Hello, World!")
		a='''
		This is a Multiline String
		written in
		more than just one line
		'''
		print("Hello, World!")
		print(a)

Output	**Output**	**Output**
Hello, World!	Hello World:	Hello, World! This is a Multiline String written in more than just one line