TELKOM
DIGITAL TALENT
INCUBATOR 2020

Telkom University

Telkom Indonesia

DATA SCIENTIST

M. Yudha Febrianta, ST., MM.

Organized by:

CAE

TELKOM
DIGITAL TALENT
INCUBATOR 2020

Modul 2: Introduction to Python

Organized by:
CAE

TELKOM DIGITAL TALENT INCUBATOR 2020

Module Overview

Topics

- Introduction to Python
- Python Operators
- Python Data Type
- Python Control Flow
- Python Function
- Python Libraries

Activities

- Group Discussion
- Coding Practice

Organized by: CAE

Module Objectives

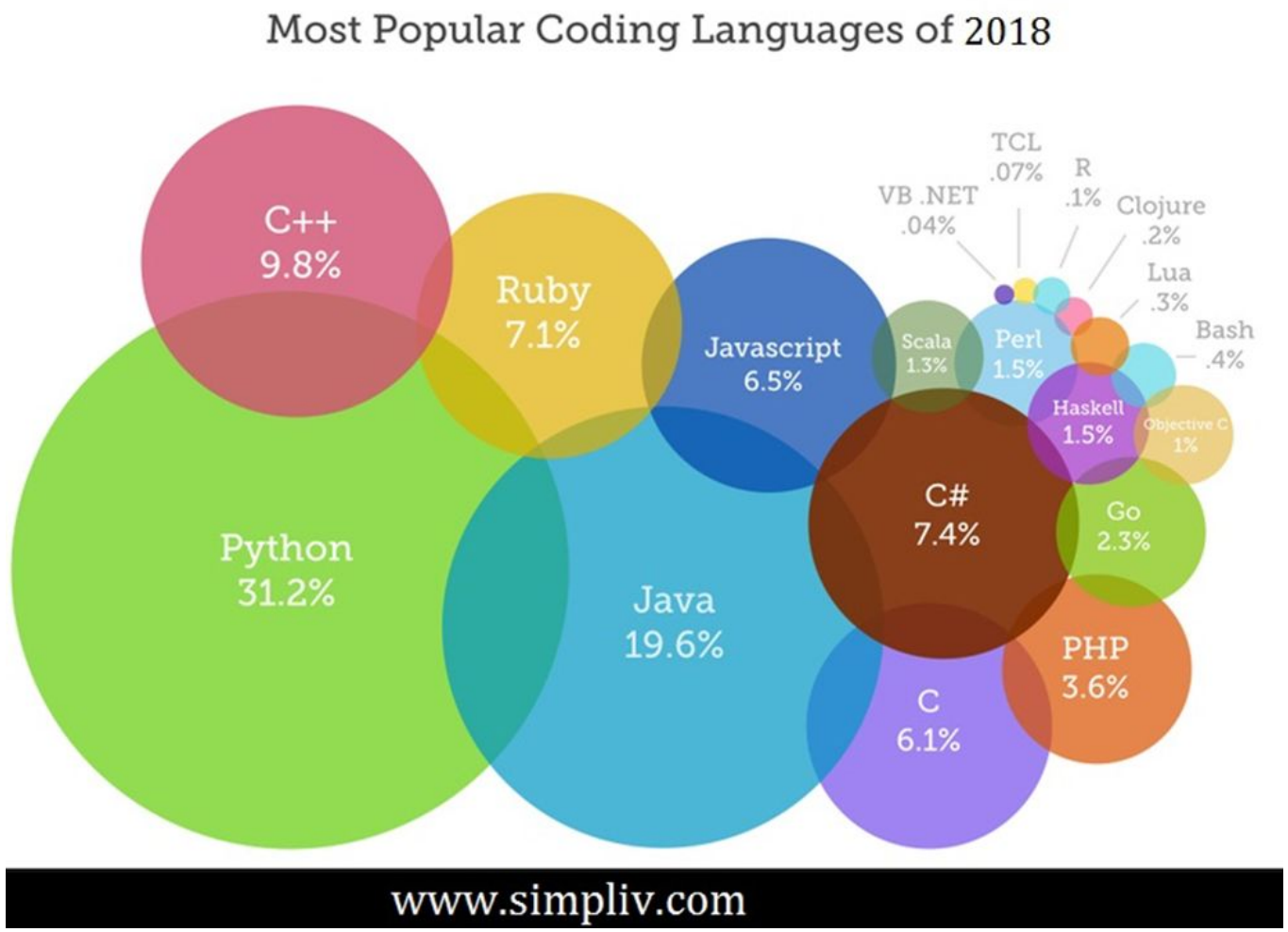
- Understand Basic Syntax of Python Programming
- Understand the python package for data science
- Understand how to use Python in Data Science process

What is Python

- It's been said that "The best things in life are free!" Python is an **open source**, **high-level**, **object-oriented**, **interpreted**, and **general purpose** dynamic programming language. It has a **community-based** development model. Its core design theory accentuates code **readability**, and its coding structure enables programmers to articulate computing concepts in **fewer lines** of code compared with other high-level programming languages such as Java, C, or C++.

TELKOM DIGITAL TALENT INCUBATOR 2020

Why Learn Python?



Organized by: CAE

TELKOM DIGITAL TALENT INCUBATOR 2020

25 of the Most Popular Python and Django Websites

https://www.shuup.com/django/25-of-the-most-popular-python-and-django-websites/

Brief history Python

- Python was officially born on February 20, 1991, with version number 0.9.0. by Guido van Rossum
- Named after Monty Python
- Open sourced from the beginning
- Considered a scripting language, but is much more
- Scalable, object oriented and functional from the beginning
- Used by Google from the beginning
- Increasingly popular
- Its application cuts across various areas such as website development, mobile apps development, scientific and numeric computing, desktop GUI, and complex software development.

Born	31 January 1956 (age 64)[1] Haarlem, Netherlands[2][3]
Nationality	Dutch
Alma mater	University of Amsterdam
Occupation	Computer programmer, author
Known for	Creating the Python programming language
Spouse(s)	Kim Knapp (m. 2000)
Children	1[4]
Awards	Award for the Advancement of Free Software (2001)
Website	gvanrossum.github.io

Python Advantages

Readable

- Intuitive and strict syntax

Productive

- saves a lot of code.

Portable

- For every operating system.

Reloaded

- It comes with many libraries.

TELKOM DIGITAL TALENT INCUBATOR 2020

The Zen of Python

- Beautiful is better than ugly—**be consistent**.
- Complex is better than complicated—**use existing libraries**.
- Simple is better than complex—**keep it simple, stupid (KISS)**.
- Flat is better than nested—**avoid nested ifs**.
- Explicit is better than implicit—**be clear**.
- Sparse is better than dense—**separate code into modules**.
- Readability counts—**indent for easy readability**.
- Special cases aren't special enough to break the rules—everything is an object.
- Errors should never pass silently—**use good exception handling**.
- Although practicality beats purity—if required, break the rules.
- Unless explicitly silenced—**use error logging and traceability**.
- In ambiguity, refuse the temptation to guess—Python syntax is simpler; however, many times we might take a longer time to decipher.
- Although the way may not be obvious at first—there is not only one way of achieving something.
- There should be, preferably, only one obvious way to do it—use existing libraries.
- If the implementation is hard to explain, it's a bad idea—**if you can't explain in simple terms, then you don't understand it well enough**.
- Now is better than never—**there are quick/dirty ways to get the job done rather than trying too much to optimize**.
- Although never is often better than right now—although there is a quick/dirty way, don't head on a path that will not allow a graceful way back.
- Namespaces are one honking great idea, so let's do more of those! Be specific.
- If the implementation is easy to explain, it may be a good idea—**simplicity is good**.

https://www.python.org/dev/peps/pep-0020/

Python Identifiers

As the name suggests, identifiers help us to differentiate one entity from another. Python entities such as class, functions, and variables are called identifiers.

- It can be a combination of upper or lower case letters (a to z or A to Z).
- It can be any digits (0 to 9) or an underscore (_).
- The general rules to be followed for writing identifiers in Python:
  - It cannot start with a digit. For example, 1variable is not valid, whereas variable1 is valid.
  - Python reserved keywords (refer to Table 1-2) cannot be used as identifiers.
  - Except for underscore (_), special symbols like !, @, #, $, %, etc. cannot be part of the identifiers.

Python Keywords

Table 1-2 lists the set of reserved words used in Python to define the syntax and structure of the language. Keywords are case sensitive, and all the keywords are in lowercase except *True, False,* and *None.*

Table 1-2. *Python Keywords*

False	class	finally	Is	return
None	continue	for	lambda	try
True	Def	From	nonlocal	while
and	Del	Global	Not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Python Operators

There are several operators in python, such as:

- Arithmetic operators
- Relational Operators
- Logical operators
- Bitwise operators
- Assignment operators
- Special operators
- Precedence and Associativity of Operators

Operators: Arithmetic

Arithmetic operators are used to perform mathematical operations like addition, subtraction, multiplication and division.

OPERATOR	DESCRIPTION	SYNTAX
+	Addition: adds two operands	x + y
-	Subtraction: subtracts two operands	x - y
*	Multiplication: multiplies two operands	x * y
/	Division (float): divides the first operand by the second	x / y
//	Division (floor): divides the first operand by the second	x // y
%	Modulus: returns the remainder when first operand is divided by the second	x % y
**	Power : Returns first raised to power second	x ** y

Operators: Relational

Relational operators compares the values. It either returns True or False according to the condition.

OPERATOR	DESCRIPTION	SYNTAX
>	Greater than: True if left operand is greater than the right	x > y
<	Less than: True if left operand is less than the right	x < y
==	Equal to: True if both operands are equal	x == y
!=	Not equal to - True if operands are not equal	x != y
>=	Greater than or equal to: True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to: True if left operand is less than or equal to the right	x <= y

Operators: Logical

Logical operators perform Logical AND, Logical OR and Logical NOT operations.

OPERATOR	DESCRIPTION	SYNTAX
and	Logical AND: True if both the operands are true	**x and y**
or	Logical OR: True if either of the operands is true	**x or y**
not	Logical NOT: True if operand is false	**not x**

Operators: Bitwise

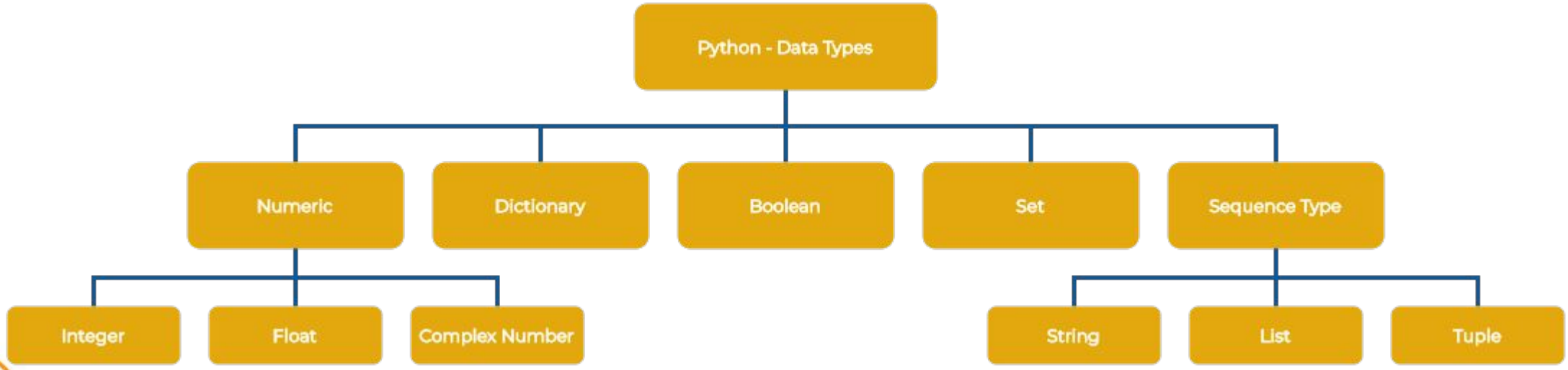
Bitwise operators acts on bits and performs bit by bit operation.

OPERATOR	DESCRIPTION	SYNTAX
&	Bitwise AND	x & y
|	Bitwise OR	x | y
~	Bitwise NOT	~x
^	Bitwise XOR	x ^ y
>>	Bitwise right shift	x>>
<<	Bitwise left shift	x<<

Python Data Types

- Data types are the classification or categorization of data items.
- It represents the kind of value that tells what operations can be performed on a particular data.
- Standard or built-in data type of Python, such as:
  - Numeric
  - Sequence Type
  - Boolean
  - Set
  - Dictionary

Python Data Types



Data Types: Numeric

- Numeric data type represent the data which has numeric value.
- Numeric value can be integer, floating number or even complex numbers.
  - **Integers** – Represented by int class. Contains positive or negative whole numbers (without fraction or decimal).
  - **Float** – Represented by float class. It is a real number with floating point representation. It is specified by a decimal point. Optionally, the character e or E followed by a positive or negative integer may be appended to specify scientific notation.
  - **Complex Numbers** – Represented by complex class. It is specified as (real part) + (imaginary part)j. For example – 2+3j

TELKOM DIGITAL TALENT INCUBATOR 2020

Data Types: Sequence Type

- Sequence is the ordered collection of similar or different data types.
- Sequences allows to store multiple values in an organized and efficient fashion.
- There are several sequence types in Python, such as:
  - **String** - Represented by str class. Strings are arrays of bytes representing Unicode characters.
  - **List** - Lists are just like the arrays, declared in other languages which is a ordered collection of data. It is very flexible as the items in a list do not need to be of the same type.
  - **Tuple** - Represented by tuple class. Tuple is just like list, tuple is also an ordered collection of Python objects. The only difference between type and list is that tuples are immutable i.e. tuples cannot be modified after it is created.

Data Types: Boolean

- Boolean is a data type with one of the two built-in values, True or False.
- Boolean objects that are equal to True are truthy (true), and those equal to False are falsy (false).
- But non-Boolean objects can be evaluated in Boolean context as well and determined to be true or false. It is denoted by the class bool.

Data Types: Set

- Set is an unordered collection of data type that is iterable, mutable and has no duplicate elements.
- The order of elements in a set is undefined though it may consist of various elements.
- Sets can be created by using the built-in set() function with an iterable object or a sequence by placing the sequence inside curly braces, separated by ‘comma’.

TELKOM DIGITAL TALENT INCUBATOR 2020

Data Types: Dictionary

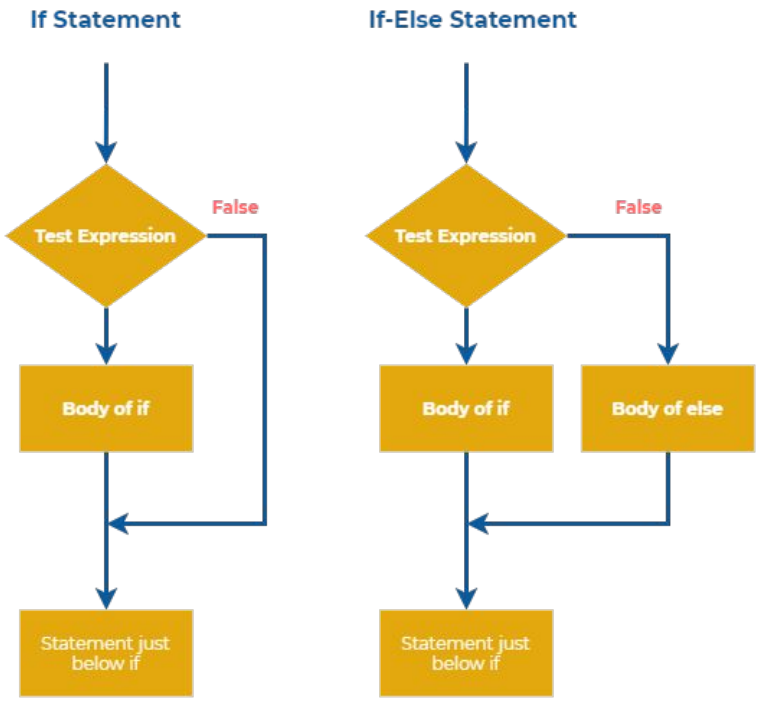
- Dictionary in Python is an unordered collection of data values, used to store data values like a map, which unlike other Data Types that hold only single value as an element, Dictionary holds key:value pair.
- Key-value is provided in the dictionary to make it more optimized. Each key-value pair in a Dictionary is separated by a colon :, whereas each key is separated by a ‘comma’.
- Dictionary can be created by placing a sequence of elements within curly {} braces, separated by ‘comma’.

Python Control Flow

- Control flow (or flow of control) is the order in which individual statements, instructions or function calls of an imperative program are executed or evaluated.
- There are three control flow statements in Python which is: If, while, and for.

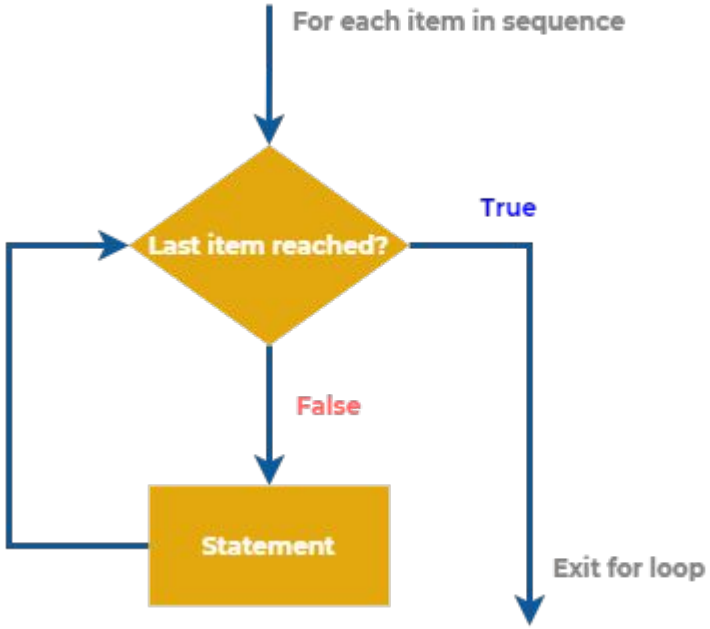
Control Flow: If Statement

- The if statement is used to check a condition: if the condition is true, we run a block of statements (called the if-block), else we process another block of statements (called the else-block).
- The else clause is optional.



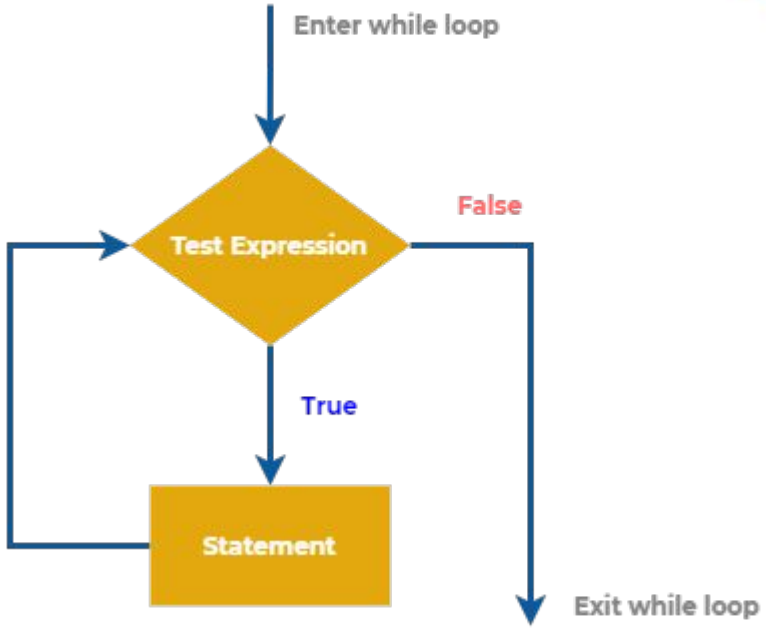
Control Flow: For Statement

- The for .. in statement is another looping statement which iterates over a sequence of objects i.e. go through each item in a sequence.



Control Flow: While Statement

- The while statement allows you to repeatedly execute a block of statements as long as a condition is true.
- A while statement is an example of what is called a looping statement. A while statement can have an optional else clause.



Python Function

- Functions are reusable pieces of programs.
- Function allow us to give a name to a block of statements and allowing us to run that block using the specified name anywhere in our program and any number of times. This is known as calling the function.
- Python provides built-in functions like print(), etc. but we can also create your own functions.

Function: Default Argument

- A default argument is a parameter that assumes a default value if a value is not provided in the function call for that argument.
- The following example illustrates Default arguments.

```
# Default arguments
def myFun(x, y=50):
    print("x: ", x)
    print("y: ", y)

# Driver code (We call myFun() with only argument)
myFun(10)
```

Function: Keyword Argument

- The idea is to allow caller to specify argument name with values so that caller does not need to remember order of parameters.

```
# Python program to demonstrate Keyword Arguments
def student(firstname, lastname):
    print(firstname, lastname)


# Keyword arguments
student(firstname ='Muhammad', lastname ='Apriandito')
student(lastname ='Apriandito', firstname ='Muhammad')
```

Function: Anonymous Functions

- Anonymous function means that a function is without a name.
- Previously, we know that def keyword is used to define the normal functions and the lambda keyword is used to create anonymous functions.

```
# Anonymous function using labmda
cube = lambda x: x*x*x
print(cube(7))
```

Python Modules

- Module is a Python object with arbitrarily named attributes that we can bind and reference.
- Module allow us to logically organize our Python code. Grouping related code into a module makes the code easier to understand and use.
- Module can define functions, classes and variables. A module can also include runnable code.

Python Libraries

- A library is a collection of files (called modules) that contains functions for use by other programs. It may also contain data values (e.g., numerical constants) and other things.
- The Python standard library is an extensive suite of modules that comes with Python itself.
- Many additional libraries are available from PyPI (the Python Package Index).
- We must import a library module before using it

Python Libraries for Data Science

- Many popular Python libraries:
  - NumPy
  - Pandas
  - Scikit-Learn
  - Tensorflow

- Visualization libraries
  - Matplotlib
  - Seaborn

Python Libraries: Numpy

- NumPy is a free for numerical computing on data that can be in the form of **large arrays** and **multidimensional matrices**.
- These multidimensional matrices are the main objects in NumPy **where their dimensions are called axes and the number of axes is called a rank**.
- NumPy also provides **various tools** to work with these arrays and **high-level mathematical functions** to manipulate this data with linear algebra, Fourier transforms, random number crunchings, etc.
- Some of the basic array operations that can be performed using NumPy include **adding, slicing, multiplying, flattening, reshaping, and indexing the arrays**. Other advanced functions include **stacking the arrays, splitting them into sections, broadcasting arrays, etc**.

Python Libraries: Pandas

- Pandas is a free Python library for **data analysis and data handling**.
- Pandas provides **various high-performance and easy-to-use data structures and operations** for **manipulating data in the form of numerical tables and time series**.
- Pandas also has multiple tools for **reading and writing data between in-memory data structures and different file formats**
- Pandas can also take in data from different types of files such as **CSV, excel etc. or a SQL database** and **create a Python object known as a data frame**. A data frame contains **rows and columns** and it can be used for **data manipulation with operations such as join, merge, groupby, concatenate** etc.

Python Libraries: Scikit Learn

- Scikit-learn is a free python library for **Machine Learning coding** primarily in the Python programming language.
- Scikit-learn is built on top of other Python libraries like **NumPy, SciPy, Matplotlib, Pandas, etc**. and so it **provides full interoperability** with these libraries.
- While Scikit-learn is written mainly in Python, it has also used **Cython to write some core algorithms in order to improve performance**.
- We can implement **various Supervised and Unsupervised Machine learning models** on Scikit-learn like **Classification, Regression, Support Vector Machines, Random Forests, Nearest Neighbors, Naive Bayes, Decision Trees, Clustering, etc**. with Scikit-learn.

Python Libraries: Matplotlib

- Matplotlib is a **data visualization library and 2-D plotting library** of Python
- Matplotlib is the **most popular and widely-used plotting library** in the Python community.
- Matplotlib can be use to **create plots, bar charts, pie charts, histograms, scatterplots, error charts, power spectra, stemplots, and other visualization chart**.
- The Pyplot module also provides a **MATLAB-like interface** that is just as versatile and **useful as MATLAB** while being totally free and open source.

Python Libraries: Seaborn

- Seaborn is a Python **data visualization library that is based on Matplotlib** and **closely integrated with the numpy and pandas data structures**.
- Seaborn has **various dataset-oriented plotting functions** that **operate on data frames and arrays** that have whole datasets within them.
- It is a **high-level interface** for creating beautiful and **informative statistical graphics** that are integral to exploring and understanding data.
- The Seaborn data graphics can **include bar charts, pie charts, histograms, scatterplots, error charts, etc**. Seaborn also has **various tools for choosing color palettes** that can reveal patterns in the data.

Python Libraries: Plotly

- Plotly is a free open-source graphing library that can be used to form data visualizations. Plotly (plotly.py)
- Plotly is built on top of the Plotly JavaScript library (plotly.js) and can be used to create web-based data visualizations that can be displayed in Jupyter notebooks or web applications using Dash or saved as individual HTML files.
- Plotly provides more than 40 unique chart types like scatter plots, histograms, line charts, bar charts, pie charts, error bars, box plots, multiple axes, sparklines, dendrograms, 3-D charts, etc. Plotly also provides contour plots, which are not that common in other data visualization libraries

TELKOM DIGITAL TALENT INCUBATOR 2020

Practice

- Practice Link: https://github.com/yudhafebrianta/intro-python

Organized by: CAE

Google Collaboratory

- Colaboratory, or “Colab” for short, is a product from Google Research.
- Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.
- More technically, Colab is a hosted Jupyter notebook service that requires no setup to use, while providing free access to computing resources including GPUs.

TELKOM DIGITAL TALENT INCUBATOR 2020

Module Summary

- Python is an interpreted, object-oriented, high-level programming language with dynamic semantics.
- Standard or built-in data type of Python: Numeric, Sequence Type, Boolean, Set, Dictionary.
- Popular Python packages for data science: Pandas, Matplotlib, Numpy, Scipy, Bokeh.

TELKOM DIGITAL TALENT INCUBATOR 2020

References/Additional Resources

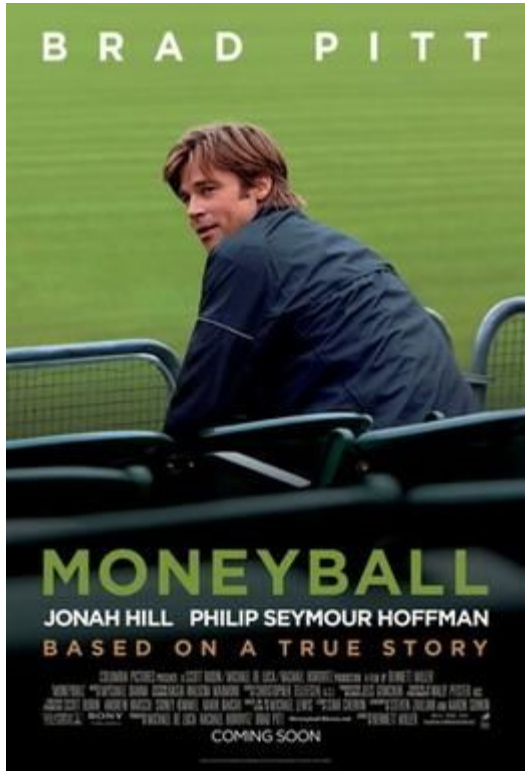
- Python Data Science Handbook: Essential Tools for Working with Data by Jake VanderPlas
- Mastering Machine Learning with Python in Six Steps by Manohar Swamynathan
- Kelas Terbuka Channel Youtube https://www.youtube.com/watch?v=iA8lLwmtKQM&list=PLZS-MHyEIRo59lUBwU-XHH7Ymmb04ffOY

Organized by: CAE

TELKOM DIGITAL TALENT INCUBATOR 2020

Assignment Week 1

- Please Watch Moneyball (2011) Movie.
- Make Summary about the film, and explain how they use Data Science
  - Write it in Indonesian
  - Post it on Medium, Blog, or other online writing media.
  - Not allowed to take other people's writings.
  - Make it as informative as possible
  - Submit the link through: http://bit.ly/dti-ds-1
  - **Deadline: 1st October 2020, 23.59 (Extended)**



Organized by:
CAE

TELKOM DIGITAL TALENT INCUBATOR 2020

Assignment Week 2

- Please Create a GitHub Account
- Submit the link through: http://bit.ly/dti-ds-2
  - **Deadline: 1st October 2020, 23.59**

GitHub

Organized by:
CAE