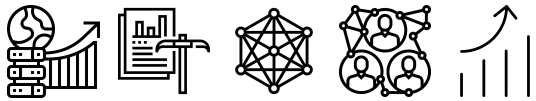


CSC 5741 (2020/21)
Data Mining and Warehousing
Lecture 2: Python for Data Mining and Machine Learning

Lighton Phiri
Department of Library & Information Science
University of Zambia
http://lis.unza.zm/~lightonphiri

Updates on Activities—March 29, 2021 (1/2)

- Trial paper reading discussion moved to next week
  - Review paper and aligned with grading rubric
- Trial Talk moved to next week
- Invited Speakers to be confirmed

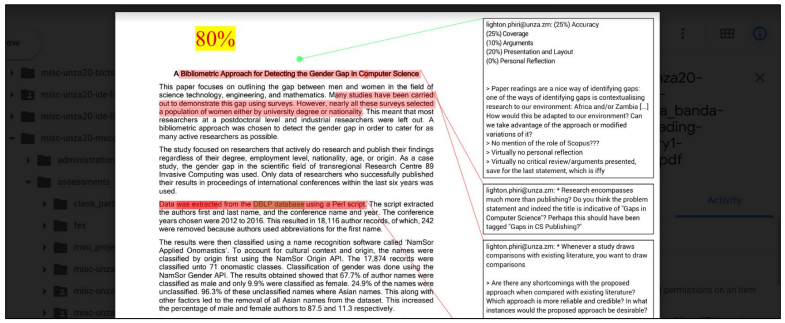
Automatic Classification of Digital Objects for Improved Metadata Quality of Electronic Theses and Dissertations in Institutional Repositories

Lighton Phiri

Department of Library and Information Science,
University of Zambia,
Lusaka, Zambia
E-mail: lighton.phiri@unza.zm

Abstract: Higher Education Institutions typically employ Institutional Repositories (IRs) in order to curate and make available Electronic Theses and Dissertations (ETDs). While most of these IRs are implemented with self-archiving functionalities, self-archiving practices are still a challenge. This arguably leads to inconsistencies in the tagging of digital objects with descriptive metadata, potentially compromising searching and browsing of scholarly research output in IRs. This paper proposes an approach to automatically classify ETDs in IRs, using supervised machine learning techniques, by extracting features from the minimum possible input expected from document authors: the ETD manuscript. The experiment results demonstrate the feasibility of automatically classifying IR ETDs and, additionally, ensuring that repository digital objects are appropriately structured. Automatic classification of repository objects has the obvious benefit of improving the searching and browsing of content in IRs and further presents opportunities for the implementation of third-party tools and extensions that could potentially result in effective self-archiving strategies.

Updates on Activities—March 29, 2021 (2/2)



Lecture Series Outline

- Part I: Jupyter Notebooks
- Part II: Google Colab
- Part III: Getting Started With Python
- Part IV: Core Python Libraries

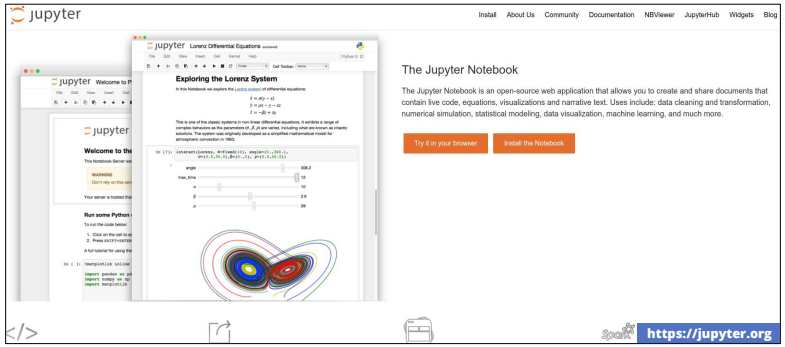
Lecture Series Outline

- Part I: Jupyter Notebooks
- Part II: Google Colab
- Part III: Getting Started With Python
- Part IV: Core Python Libraries

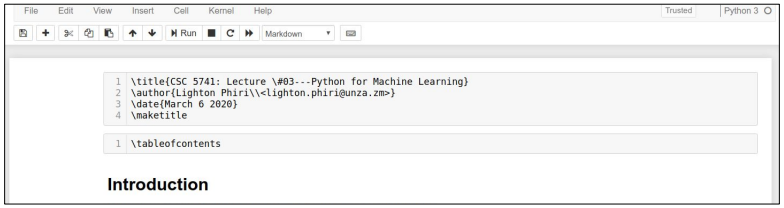
Lecture Series Outline

- Part I: Jupyter Notebooks
  - Jupyter Notebooks Interface
  - Textual Content
  - Live Code
  - Visualisations
- Part II: Google Colab
- Part III: Getting Started With Python
- Part IV: Core Python Libraries

About Jupyter Notebooks (1/2)



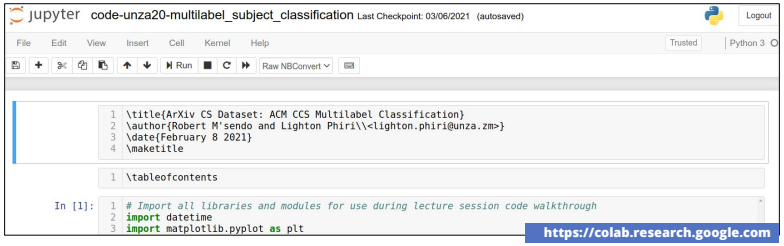
About Jupyter Notebooks (2/2)



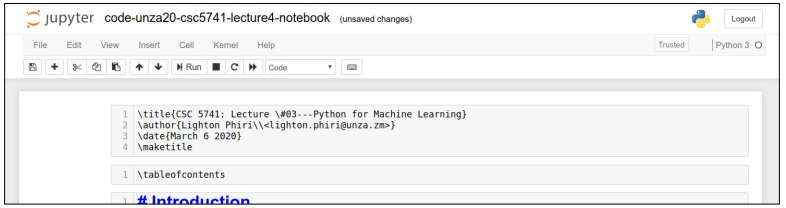
- A notebook is a web application that contains descriptive textual content, live code, equations and visualizations.
  - While we shall predominantly use the Python kernel, additional kernels for other languages like R can be installed.

About Jupyter Notebooks: Installation

- Installation instructions are available online (https://jupyter.org/install)

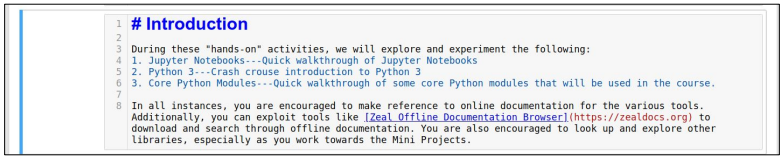


About Jupyter Notebooks: UI



- Text and live code is specified in cells and menubar and/or toolbar are used to execute cell contents
- Output appears immediately below the input cell

About Jupyter Notebooks: Text (1/4)



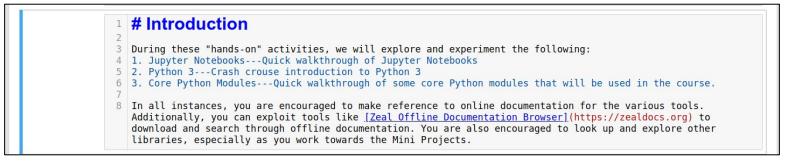
```
# Introduction

During these "hands-on" activities, we will explore and experiment the following:
1. Jupyter Notebooks---Quick walkthrough of Jupyter Notebooks
2. Python 3---Crash crouse introduction to Python 3
3. Core Python Modules---Quick walkthrough of some core Python modules that will be used in the course.

In all instances, you are encouraged to make reference to online documentation for the various tools.
Additionally, you can exploit tools like [Zeal Offline Documentation Browser](https://zealdocs.org) to
download and search through offline documentation. You are also encouraged to look up and explore other
libraries, especially as you work towards the Mini Projects.
```

- Textual content is primarily specified using the markup language “Markdown”
  - You essentially specify the structure of your text, similar to HTML

About Jupyter Notebooks: Text (2/4)



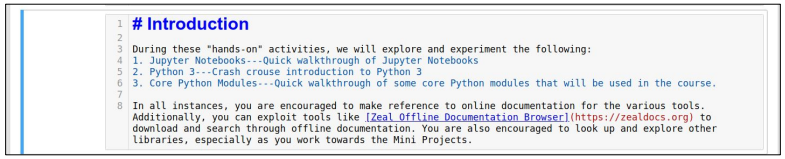
```
# Introduction

During these "hands-on" activities, we will explore and experiment the following:
1. Jupyter Notebooks---Quick walkthrough of Jupyter Notebooks
2. Python 3---Crash crouse introduction to Python 3
3. Core Python Modules---Quick walkthrough of some core Python modules that will be used in the course.

In all instances, you are encouraged to make reference to online documentation for the various tools.
Additionally, you can exploit tools like [Zeal Offline Documentation Browser](https://zealdocs.org) to
download and search through offline documentation. You are also encouraged to look up and explore other
libraries, especially as you work towards the Mini Projects.
```

- Common markup: Headings
  - # h1
  - ## h2
  - ### h3
  - #### h4

About Jupyter Notebooks: Text (3/4)



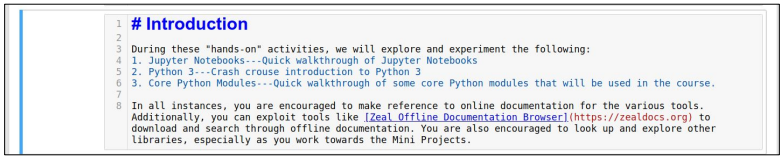
```
# Introduction

During these "hands-on" activities, we will explore and experiment the following:
1. Jupyter Notebooks---Quick walkthrough of Jupyter Notebooks
2. Python 3---Crash crouse introduction to Python 3
3. Core Python Modules---Quick walkthrough of some core Python modules that will be used in the course.

In all instances, you are encouraged to make reference to online documentation for the various tools.
Additionally, you can exploit tools like [Zeal Offline Documentation Browser](https://zealdocs.org) to
download and search through offline documentation. You are also encouraged to look up and explore other
libraries, especially as you work towards the Mini Projects.
```

- **Common markup: Lists—Unordered**
  - * Jupyter Notebooks
  - * Python 3
  - * Core Python Libraries

About Jupyter Notebooks: Text (4/4)



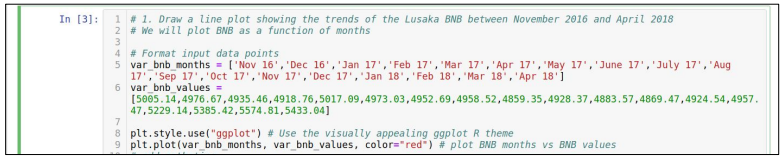
```
# Introduction

During these "hands-on" activities, we will explore and experiment the following:
1. Jupyter Notebooks---Quick walkthrough of Jupyter Notebooks
2. Python 3---Crash crouse introduction to Python 3
3. Core Python Modules---Quick walkthrough of some core Python modules that will be used in the course.

In all instances, you are encouraged to make reference to online documentation for the various tools.
Additionally, you can exploit tools like [Zeal Offline Documentation Browser](https://zealdocs.org) to
download and search through offline documentation. You are also encouraged to look up and explore other
libraries, especially as you work towards the Mini Projects.
```

- **Common markup: Lists—Unordered**
  - 1. Jupyter Notebooks
  - 2. Python 3
  - 3. Core Python Libraries

About Jupyter Notebooks: Code (1/2)



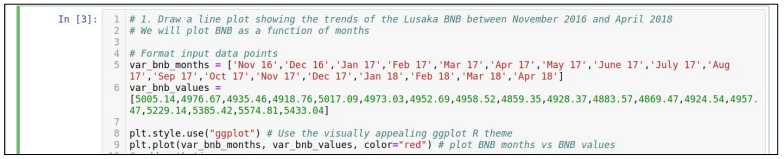
```
In [3]:
# 1. Draw a line plot showing the trends of the Lusaka BNB between November 2016 and April 2018
# We will plot BNB as a function of months

# Format input data points
var_bnb_months = ['Nov 16','Dec 16','Jan 17','Feb 17','Mar 17','Apr 17','May 17','June 17','July 17','Aug
17','Sep 17','Oct 17','Nov 17','Dec 17','Jan 18','Feb 18','Mar 18','Apr 18']
var_bnb_values =
[5005.14,4976.67,4935.46,4918.76,5017.09,4973.03,4952.69,4958.52,4859.35,4928.37,4883.57,4869.47,4924.54,4957.
47,5229.14,5385.42,5574.81,5433.04]

plt.style.use("ggplot") # Use the visually appealing ggplot R theme
plt.plot(var_bnb_months, var_bnb_values, color="red") # plot BNB months vs BNB values
```

- **Python code, shell commands and magics are the most common type of code**
  - Python code is specified in its raw form in the cells

About Jupyter Notebooks: Code (2/2)



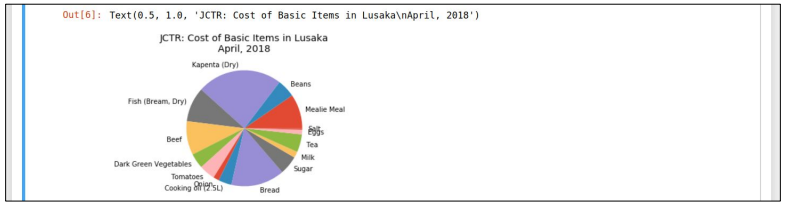
```
In [3]:
# 1. Draw a line plot showing the trends of the Lusaka BNB between November 2016 and April 2018
# We will plot BNB as a function of months

# Format input data points
var_bnb_months = ['Nov 16','Dec 16','Jan 17','Feb 17','Mar 17','Apr 17','May 17','June 17','July 17','Aug
17','Sep 17','Oct 17','Nov 17','Dec 17','Jan 18','Feb 18','Mar 18','Apr 18']
var_bnb_values =
[5005.14,4976.67,4935.46,4918.76,5017.09,4973.03,4952.69,4958.52,4859.35,4928.37,4883.57,4869.47,4924.54,4957.
47,5229.14,5385.42,5574.81,5433.04]

plt.style.use("ggplot") # Use the visually appealing ggplot R theme
plt.plot(var_bnb_months, var_bnb_values, color="red") # plot BNB months vs BNB values
```

- **Python code, shell commands and magics are the most common type of code**
  - Shell commands are prefixed with “!”
  - Cell magics are prefixed with “%%”
    - Available magics specified with “%lsmagic”

About Jupyter Notebooks: Visualisations



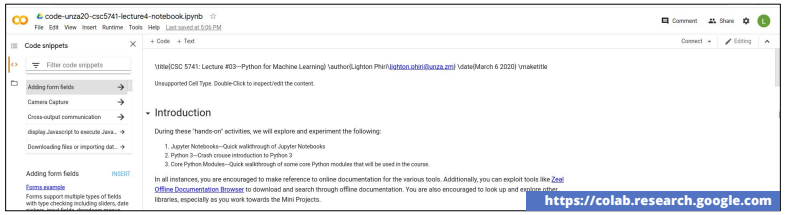
- Visualisations can be generated using plotting libraries like matplotlib or using HTML via the "%%html" magic

Lecture Series Outline

- Part I: Jupyter Notebooks
- Part II: Google Colab
  - Google Colab Interface
- Part III: Getting Started With Python
- Part IV: Core Python Libraries

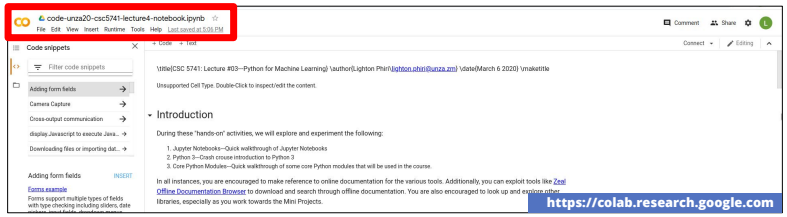
Google Colab Interface (1/5)

- Google Colaboratory is a cloud-based alternative to Jupyter Notebook



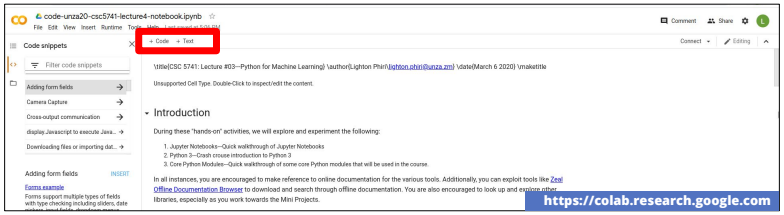
Google Colab Interface (2/5)

- Google Colaboratory is a cloud-based alternative to Jupyter Notebook



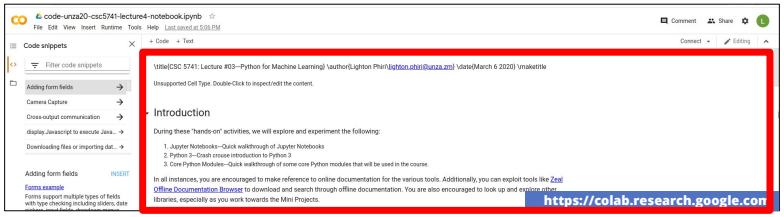
Google Colab Interface (3/5)

- Google Colaboratory is a cloud-based alternative to Jupyter Notebook



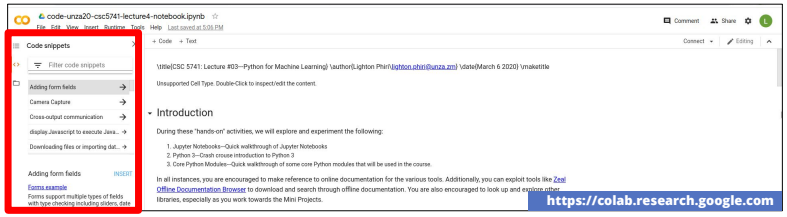
Google Colab Interface (4/5)

- Google Colaboratory is a cloud-based alternative to Jupyter Notebook



Google Colab Interface (5/5)

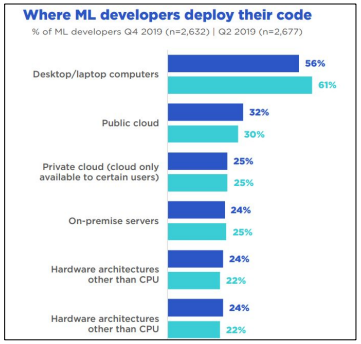
- Google Colaboratory is a cloud-based alternative to Jupyter Notebook

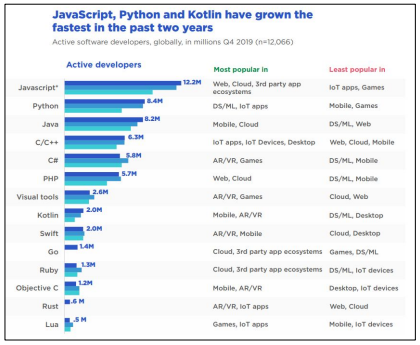


Lecture Series Outline

- Part I: Jupyter Notebooks
- Part II: Google Colab
- Part III: Getting Started With Python
  - Introduction
  - Installation and Setup
  - Basics
  - Data Structures
  - Flow Control
  - Functions and Modules
- Part IV: Core Python Libraries

Motivation





Getting Started With Python (1/3)

```
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently
```


Getting Started With Python (2/3)

- Python is an interpreted language
- Python is a scripting language
- Python is a general purpose language
- Python is an Object Oriented language
- [...]
- [...]
- We recommend using Python 3

Getting Started With Python (3/3)

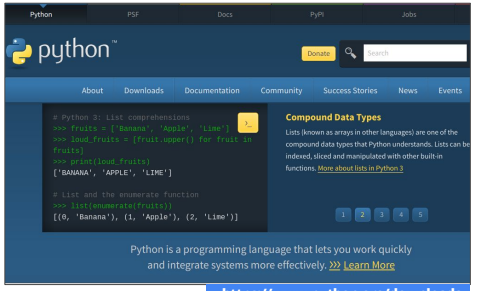
- Python statements can be executed directly from the interpreter
- Python scripts can be executed as shell commands

Installation and Setup

- [...]

Installation and Setup (1/3)

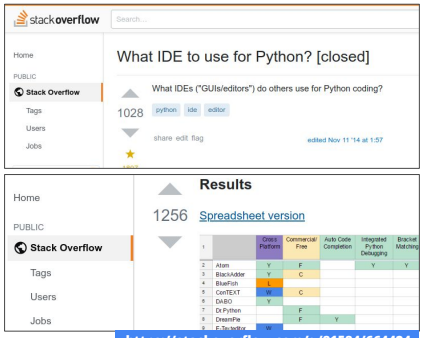
- **Download and install the latest version of Python 3**
  - Installers also available on course Web page, in the resources directory
- **Download and install the latest version of pip**



https://www.python.org/downloads

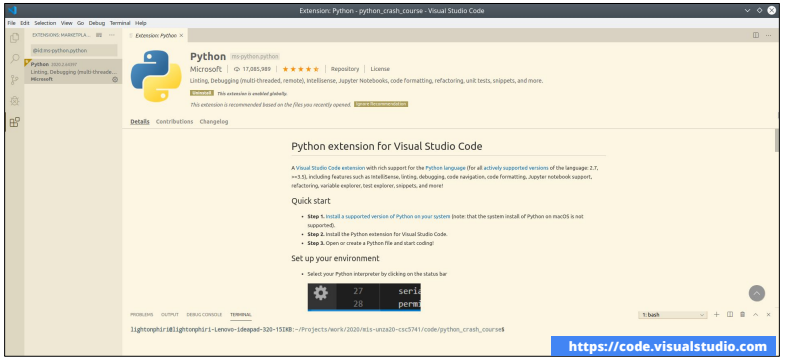
Installation and Setup (2/3)

- **Any text editor will be sufficient for scripting.**
  - Vim, Notepad [...]
- **On IDEs**
  - There are plenty of IDEs to choose from
  - In the recent past, I have worked with Wing 101 and Kate



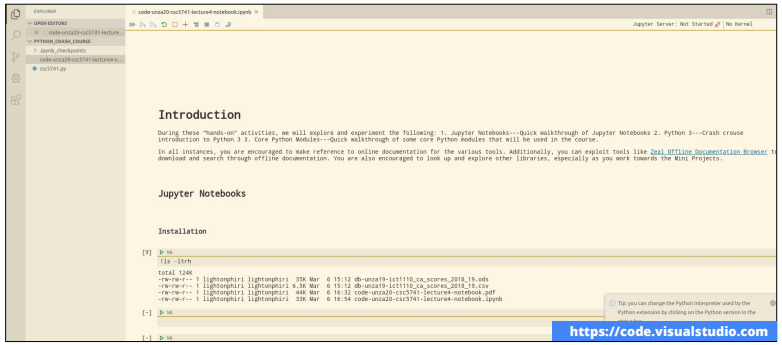
https://stackoverflow.com/q/81584/664424

Installation and Setup (3/3)



https://code.visualstudio.com

Installation and Setup (3/3)



Basics

- No need to specify data types on variable declaration
- Indentation is important

Identifiers (1/2)

- Python is case-sensitive, meaning uppercase and lowercase are considered as different
  - age is different from AGE
  - favourite_course is different from Favourite_Course
- Variable names, like other identifiers, follow rules
  - can use letters, numbers or underscores
  - can't use other punctuation
  - can't start with a number
  - can't use Python keywords (reserved words)
- The assignment operator in Python is the equals sign =
  - **>>> age = 19**

Identifiers (2/2)

- Python keywords (reserved words) can't be used when naming identifiers
- **>>> import keyword**
- **>>> keyword.kwlist**
- **['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']**

Comments (1/2)

- **Comments are useful in explaining your code, and are ignored by the Python interpreter**
- **Single line comments are simply indicated with a hash # character**
- **Everything to the right of the hash is ignored**
  - >>> course_code = “csc5741” # creates a variable course_code

Comments (2/2)

- **Multiple line comments are specified between sets of three quotes, ' ' ' or " " "**

```
' ' ' Author: Mwangala Sikota
Course: CSC 5741
Lecture #04 ' ' '

" " " Author: Mumbi Mumbi
Course: CSC 5741
Lecture #04 " " "
```


Data Types (1/3)

- **Variables don’t require explicit type declaration in Python, as in other programming languages**
  - **>>> x = 5**
- **There are a few basic data types in Python**

Integers	**int**
Float	**float**
String	**str**
Boolean	**bool**

Data Types (2/3)

- **Integer, whole numbers**
  - **>>> i = 23**
- **Float, floating point numbers**
  - full stop indicates decimal point
  - **>>> d = 2.345**
- **String, piece of text**
  - enclosed in single (‘) or double quotes (“”)
  - **>>> x = ‘CSC 5741’**
  - **>>> y = “CSC 5741”**

Data Types (3/3)

- **Boolean, true or false**
  - values True and False, start with capital letter
  - 0, '', [], (), {}, None are considered False, everything else is True
  - **>>> weekday = True**

Functions (1/3)

- **Functions are used to perform simple operations, sometimes on values**
- **Functions are called with round brackets ()**
  - function_name()
- **Functions can be passed certain values, which are referred to as parameters (or arguments) separated by commas**
  - function_name(parameter)
  - function_name(parameter1, parameter2, ...)

Functions (2/3)

- **Python has many built-in functions, here are some:**
  - print() function prints information to the screen
  - input() function gets information from the user
  - type() function returns data type of variable or value
  - **>>> x = 3**
  - **>>> type(x)**
  - **<class 'int'>**

Functions (3/3)

```
def csc5741(x, y='Y', z='Z'):
    print(x + ' ' + y)
return 0

csc5741('Xxxx', 'Yyyyy')
```

- **All arguments are named**
- **Naming useful for optional arguments**
- **Return is optional**

Data Structures

- **Tuple**
  - var = (1, 2, 3, 4, 5)
- **List**
  - var = [1, 2, 3, 4, 5]
- **Dictionary**
  - var = {“one”:1, “two”:2, “three”:3, “four”:4, “five”:5}
- **Set**
  - var = {1, 2, 3, 4, 5}

Loops

```
for i in [1,2,3]:
    print(i)


while i < 5:
    i += 1
    print(i)
```

- **No curly braces or "end for"**
- **Structure is derived from level of indentation**
- **One statement per line**
- **No semicolons required**

Modules (1/2)

- **Modules facilitate extensibility and reusability**
- **Modules are collections of functions adding functionality to Python**
- **Modules can be imported using import keyword**
  - Once modules are imported, their functions can be accessed by using the module name
  - The help() function displays what is contained in a module

```
from math import sqrt
import  math
```


Modules (2/2)

- **Single functions can be imported using the from statement**
  - **>>> from math import sqrt**
- **When using the from statement functions can be accessed without the module name**
  - **>>> sqrt(16)**
- **Everything from the module can be imported using an asterisk with the from statement**
  - **>>> from math import ***

Lecture Series Outline

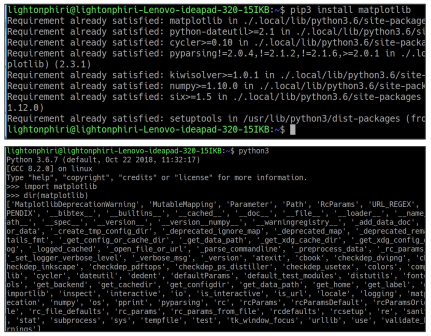
- Part I: Jupyter Notebooks
- Part II: Google Colab
- Part III: Getting Started With Python
- Part IV: Core Python Libraries
  - Datasets
  - matplotlib
  - pandas
  - Scikit-learn

Datasets

- See Jupyter Notebook "2020/21 CSC 5741: Lecture #04 Notebook—Python for Machine Learning" (http://bit.ly/2Q2T2Lw)

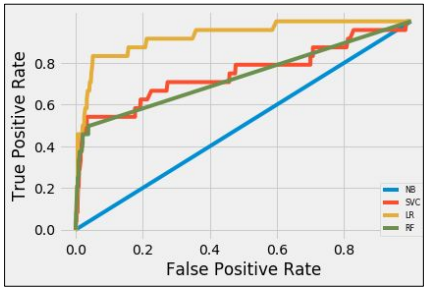
Matplotlib (1/7)

- The matplotlib library is best installed using pip, as with all libraries or using apt-get, if on Mac or Linux
  - pip3 install matplotlib
  - sudo apt-get install python-matplotlib
- Test installation by importing a library module



Matplotlib (2/7)

- Basic elements of a plot
  - Plot title
  - Axis labels
  - Legend



Matplotlib (3/7)

- **Creating plots is a four-step process**
  1) Import matplotlib
  2) Draw the plot
  3) Specify plot aesthetics
  4) Render plot

Matplotlib (4/7)

- **Creating plots is a four-step process**
  1) Import matplotlib
     import matplotlib.pyplot as plt
  2) Draw the plot
  3) Specify plot aesthetics
  4) Render plot

Matplotlib (5/7)

- **Creating plots is a four-step process**
  1) Import matplotlib
     import matplotlib.pyplot as plt
  2) Draw the plot
     plt.plot([...])
     plt.hist([...])
  3) Specify plot aesthetics
  4) Render plot
- **Illustration**
  - Simple plots
  - Plots using pandas dataframe

Matplotlib (6/7)

- **Creating plots is a four-step process**
  1) Import matplotlib
     import matplotlib.pyplot as plt
  2) Draw the plot
     plt.plot([...])
     plt.hist([...])
  3) Specify plot aesthetics
     plt.xlabel("[...]")
     plt.ylabel("[...]")
  4) Render plot
- **Illustration**
  - Simple plots
  - Plots using pandas dataframe

Matplotlib (7/7)

- **Creating plots is a four-step process**
  1) Import matplotlib
     import matplotlib.pyplot as plt
  2) Draw the plot
     plt.plot([...])
     plt.hist([...])
  3) Specify plot aesthetics
     plt.xlabel("[...]"); plt.ylabel("[...]")
     plt.legend()
  4) Render plot
     plt.show()
- **Illustration**
  - Simple plots
  - Plots using pandas dataframe

Matplotlib—Exercises

- **See Jupyter Notebook "2020/21 CSC 5741: Lecture #04 Notebook—Python for Machine Learning" (http://bit.ly/2Q2T2Lw)**

Lecture Series Outline

- **Part I: Jupyter Notebooks**
- **Part II: Google Colab**
- **Part III: Getting Started With Python**
- **Part IV: Core Python Libraries**
  - Datasets
  - matplotlib
  - pandas
  - Scikit-learn

Pandas (1/9)

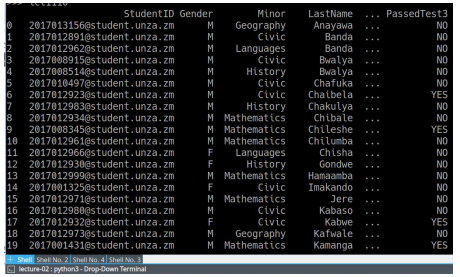
- **Why use pandas instead a spreadsheet for data analysis**
  - Efficiency as data scales
  - Very user-friendly
  - Dataframe similar to spreadsheet

Pandas (2/9)

- **Why use pandas instead a spreadsheet for data analysis**
  - Efficiency as data scales
  - Very user-friendly
  - Dataframe similar to spreadsheet

Pandas (3/9)

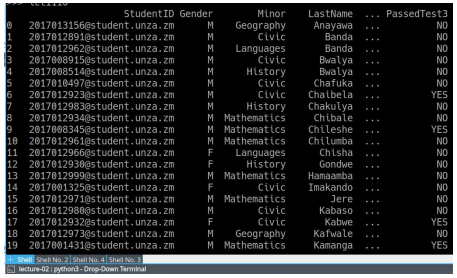
- **Pandas DataFrame**
  - Two dimensional labeled data structure
  - DataFrame can be viewed as a representation of a Spreadsheet worksheet



	StudentID	Gender	Minor	LastName	...	PassedTest3
0	2017013156@student.unza.zm	M	Geography	Anayawa	...	NO
1	2017012891@student.unza.zm	M	Civic	Banda	...	NO
2	2017012962@student.unza.zm	M	Languages	Banda	...	NO
3	2017008915@student.unza.zm	M	Civic	Bwalya	...	NO
4	2017008514@student.unza.zm	M	History	Bwalya	...	NO
5	2017010497@student.unza.zm	M	Civic	Chafuka	...	NO
6	2017012923@student.unza.zm	M	Civic	Chaibela	...	YES
7	2017012983@student.unza.zm	M	History	Chakulya	...	NO
8	2017012934@student.unza.zm	M	Mathematics	Chibale	...	NO
9	2017008345@student.unza.zm	M	Mathematics	Chileshe	...	YES
10	2017012961@student.unza.zm	M	Mathematics	Chilumba	...	NO
11	2017012966@student.unza.zm	F	Languages	Chisha	...	NO
12	2017012930@student.unza.zm	F	History	Gondwe	...	NO
13	2017012999@student.unza.zm	M	Mathematics	Hamaamba	...	NO
14	2017001325@student.unza.zm	F	Civic	Imakando	...	NO
15	2017012971@student.unza.zm	M	Mathematics	Jere	...	NO
16	2017012980@student.unza.zm	M	Civic	Kabaso	...	NO
17	2017012932@student.unza.zm	F	Civic	Kabwe	...	YES
18	2017012973@student.unza.zm	M	Geography	Kafwale	...	NO
19	2017001431@student.unza.zm	M	Mathematics	Kamanga	...	YES

Pandas (4/9)

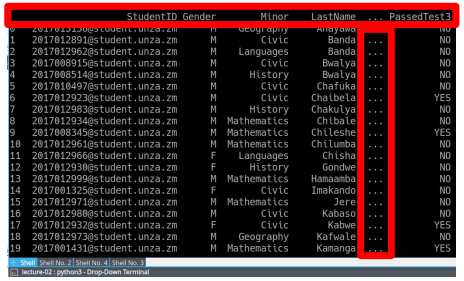
- **Pandas series**
  - One dimensional labeled array that can hold any data type.
  - Similar to column in Spreadsheet applications



	StudentID	Gender	Minor	LastName	...	PassedTest3
0	2017013156@student.unza.zm	M	Geography	Anayawa	...	NO
1	2017012891@student.unza.zm	M	Civic	Banda	...	NO
2	2017012962@student.unza.zm	M	Languages	Banda	...	NO
3	2017008915@student.unza.zm	M	Civic	Bwalya	...	NO
4	2017008514@student.unza.zm	M	History	Bwalya	...	NO
5	2017010497@student.unza.zm	M	Civic	Chafuka	...	NO
6	2017012923@student.unza.zm	M	Civic	Chaibela	...	YES
7	2017012983@student.unza.zm	M	History	Chakulya	...	NO
8	2017012934@student.unza.zm	M	Mathematics	Chibale	...	NO
9	2017008345@student.unza.zm	M	Mathematics	Chileshe	...	YES
10	2017012961@student.unza.zm	M	Mathematics	Chilumba	...	NO
11	2017012966@student.unza.zm	F	Languages	Chisha	...	NO
12	2017012930@student.unza.zm	F	History	Gondwe	...	NO
13	2017012999@student.unza.zm	M	Mathematics	Hamaamba	...	NO
14	2017001325@student.unza.zm	F	Civic	Imakando	...	NO
15	2017012971@student.unza.zm	M	Mathematics	Jere	...	NO
16	2017012980@student.unza.zm	M	Civic	Kabaso	...	NO
17	2017012932@student.unza.zm	F	Civic	Kabwe	...	YES
18	2017012973@student.unza.zm	M	Geography	Kafwale	...	NO
19	2017001431@student.unza.zm	M	Mathematics	Kamanga	...	YES

Pandas (5/9)

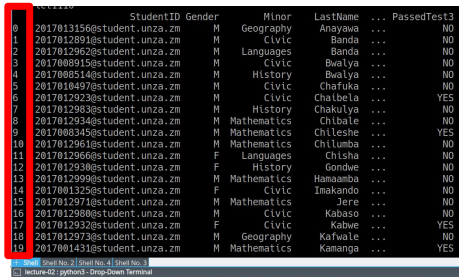
- **Columns**
  - Ellipse indicate more columns. Structure of data frame indicated on last line of output



	StudentID	Gender	Minor	LastName	...	PassedTest3
0	2017013156@student.unza.zm	M	Geography	Anayawa	...	NO
1	2017012891@student.unza.zm	M	Civic	Banda	...	NO
2	2017012962@student.unza.zm	M	Languages	Banda	...	NO
3	2017008915@student.unza.zm	M	Civic	Bwalya	...	NO
4	2017008514@student.unza.zm	M	History	Bwalya	...	NO
5	2017010497@student.unza.zm	M	Civic	Chafuka	...	NO
6	2017012923@student.unza.zm	M	Civic	Chaibela	...	YES
7	2017012983@student.unza.zm	M	History	Chakulya	...	NO
8	2017012934@student.unza.zm	M	Mathematics	Chibale	...	NO
9	2017008345@student.unza.zm	M	Mathematics	Chileshe	...	YES
10	2017012961@student.unza.zm	M	Mathematics	Chilumba	...	NO
11	2017012966@student.unza.zm	F	Languages	Chisha	...	NO
12	2017012930@student.unza.zm	F	History	Gondwe	...	NO
13	2017012999@student.unza.zm	M	Mathematics	Hamaamba	...	NO
14	2017001325@student.unza.zm	F	Civic	Imakando	...	NO
15	2017012971@student.unza.zm	M	Mathematics	Jere	...	NO
16	2017012980@student.unza.zm	M	Civic	Kabaso	...	NO
17	2017012932@student.unza.zm	F	Civic	Kabwe	...	YES
18	2017012973@student.unza.zm	M	Geography	Kafwale	...	NO
19	2017001431@student.unza.zm	M	Mathematics	Kamanga	...	YES

Pandas (6/9)

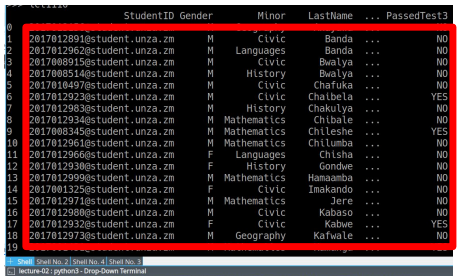
- **Index**
  - Automatically generated, but can be changed
  - Uniquely identifies rows in the DataFrame



	StudentID	Gender	Minor	LastName	...	PassedTest3
0	2017013156@student.unza.zm	M	Geography	Anayawa	...	NO
1	2017012891@student.unza.zm	M	Civic	Banda	...	NO
2	2017012962@student.unza.zm	M	Languages	Banda	...	NO
3	2017008915@student.unza.zm	M	Civic	Bwalya	...	NO
4	2017008514@student.unza.zm	M	History	Bwalya	...	NO
5	2017010497@student.unza.zm	M	Civic	Chafuka	...	NO
6	2017012923@student.unza.zm	M	Civic	Chaibela	...	YES
7	2017012983@student.unza.zm	M	History	Chakulya	...	NO
8	2017012934@student.unza.zm	M	Mathematics	Chibale	...	NO
9	2017008345@student.unza.zm	M	Mathematics	Chileshe	...	YES
10	2017012961@student.unza.zm	M	Mathematics	Chilumba	...	NO
11	2017012966@student.unza.zm	F	Languages	Chisha	...	NO
12	2017012930@student.unza.zm	F	History	Gondwe	...	NO
13	2017012999@student.unza.zm	M	Mathematics	Hamaamba	...	NO
14	2017001325@student.unza.zm	F	Civic	Imakando	...	NO
15	2017012971@student.unza.zm	M	Mathematics	Jere	...	NO
16	2017012980@student.unza.zm	M	Civic	Kabaso	...	NO
17	2017012932@student.unza.zm	F	Civic	Kabwe	...	YES
18	2017012973@student.unza.zm	M	Geography	Kafwale	...	NO
19	2017001431@student.unza.zm	M	Mathematics	Kamanga	...	YES

Pandas (7/9)

- **Data**



	StudentID	Gender	Minor	LastName	...	PassedTest3
1	2017012891@student.unza.zm	M	Civic	Banda	...	NO
2	2017012962@student.unza.zm	M	Languages	Banda	...	NO
3	2017008915@student.unza.zm	M	Civic	Bwalya	...	NO
4	2017008514@student.unza.zm	M	History	Bwalya	...	NO
5	2017010497@student.unza.zm	M	Civic	Chafuka	...	NO
6	2017012923@student.unza.zm	M	Civic	Chaibela	...	YES
7	2017012983@student.unza.zm	M	History	Chakulya	...	NO
8	2017012934@student.unza.zm	M	Mathematics	Chibale	...	NO
9	2017008345@student.unza.zm	M	Mathematics	Chileshe	...	YES
10	2017012961@student.unza.zm	M	Mathematics	Chilumba	...	NO
11	2017012966@student.unza.zm	F	Languages	Chisha	...	NO
12	2017012930@student.unza.zm	F	History	Gondwe	...	NO
13	2017012999@student.unza.zm	M	Mathematics	Hamaamba	...	NO
14	2017001325@student.unza.zm	F	Civic	Imakando	...	NO
15	2017012971@student.unza.zm	M	Mathematics	Jere	...	NO
16	2017012980@student.unza.zm	M	Civic	Kabaso	...	NO
17	2017012932@student.unza.zm	F	Civic	Kabwe	...	YES
18	2017012973@student.unza.zm	M	Geography	Kafwale	...	NO
19						

Pandas (8/9)

- **Some common operations**
  - Reading data files
    - df.read_csv([...])
    - df.read_html([...])
    - df.read_json([...])
    - df.read_*
  - Inspecting dataframes
    - df.head([...])
    - df.tail([...])
    - df.columns
    - df[‘[...]’]

Pandas (9/9)

- **Some common operations**
  - Converting to different file formats
    - df.to_csv([...])
    - df.to_excel([...])
    - df.to_sql([...])
    - df.to_*
  - Renaming columns
    - df.rename(columns={[...]})
  - Aggregating data
    - df.groupby([‘[...]’]).mean()
    - df.groupby(‘[...]’).max()

Pandas—Exercise

- See Jupyter Notebook "2020/21 CSC 5741: Lecture #04 Notebook—Python for Machine Learning" (http://bit.ly/2Q2T2Lw)

Lecture Series Outline

- Part I: Jupyter Notebooks
- Part II: Google Colab
- Part III: Getting Started With Python
- Part IV: Core Python Libraries
  - Datasets
  - matplotlib
  - pandas
  - Scikit-learn

Scikit-learn

- Scikit-learn
  - Ensure that the module is installed by using the import statement

```
lightonphiri@lightonphiri-Lenovo-ideapad-320-15IKB:~$ pip3 install sklearn
Collecting sklearn
  Downloading https://files.pythonhosted.org/packages/1e/7a/dbb3be0ce9bd5c8b7e3d
sklearn-0.0.tar.gz
Collecting scikit-learn (from sklearn)
  Downloading https://files.pythonhosted.org/packages/5e/82/c0de5839d613b82bddd0
scikit_learn-0.20.3-cp36-cp36m-manylinux1_x86_64.whl (5.4MB)
    0% ||                                | 20kB 55kB/s eta 0:01:38
```

```
lightonphiri@lightonphiri-Lenovo-ideapad-320-15IKB:~$ python3
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import sklearn
>>> dir(sklearn)
['__SKLEARN_SETUP__', '__all__', '__builtins__', '__cached__', '__check_build', '__doc__',
ame__', '__package__', '__path__', '__spec__', '__version__', '_config', 'base', 'clone',
 'externals', 'get_config', 'logger', 'logging', 're', 'set_config', 'setup_module', 'show
rnings']
>>>
>>>
```


scikit-learn—Exercises

- See Jupyter Notebook "2020/21 CSC 5741: Lecture #04 Notebook—Python for Machine Learning" (http://bit.ly/2Q2T2Lw)

Bibliography

[1] A Byte of Python
https://python.swaroopch.com

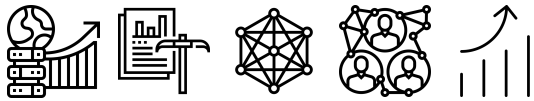
[2] Python 3.4 Programming Tutorials
https://www.youtube.com/playlist?list=PL6gx4Cwl9DGAcbMi1sH6oAMk4JHw91mC_

[3] Python for Beginners | Python.org
https://www.python.org/about/gettingstarted

[4] Pyplot tutorial – Matplotlib 3.0.3 documentation
https://matplotlib.org/tutorials/introductory/pyplot.html

[5] 10 Minutes to pandas – pandas 0.22.0 documentation
https://pandas.pydata.org/pandas-docs/version/0.22/10min.html

csc5741@unza.zm
http://bit.ly/39HTdTK
http://bit.ly/2kK2ZkA



CSC 5741 (2020/21)
Data Mining and Warehousing
Lecture 2: Python for Data Mining and Machine Learning

Lighton Phiri
Department of Library & Information Science
University of Zambia
http://lis.unza.zm/~lightonphiri