

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)

The screenshot shows a digital workspace interface. On the left is a sidebar with a tree view of files and folders, including 'misc-unza20-bicts', 'misc-unza20-ide-l', 'misc-unza20-ide-l', 'misc-unza20-msci', 'administration', 'assessments', 'class_part', 'fex', 'mini_proje', 'misc-unza', 'misc-unza', and 'misc-unza'. The main area displays a presentation slide with the following content:

80%

A Bibliometric Approach for Detecting the Gender Gap in Computer Science

This paper focuses on outlining the gap between men and women in the field of science technology, engineering, and mathematics. Many studies have been carried out to demonstrate this gap using surveys. However, nearly all these surveys selected a population of women either by university degree or nationality. This meant that most researchers at a postdoctoral level and industrial researchers were left out. A bibliometric approach was chosen to detect the gender gap in order to cater for as many active researchers as possible.

The study focused on researchers that actively do research and publish their findings regardless of their degree, employment level, nationality, age, or origin. As a case study, the gender gap in the scientific field of transregional Research Centre 89 Invasive Computing was used. Only data of researchers who successfully published their results in proceedings of international conferences within the last six years was used.

Data was extracted from the DBLP database using a Perl script. The script extracted the authors first and last name, and the conference name and year. The conference years chosen were 2012 to 2016. This resulted in 18,116 author records, of which, 242 were removed because authors used abbreviations for the first name.

The results were then classified using a name recognition software called 'NamSor Applied Onomastics'. To account for cultural context and origin, the names were classified by origin first using the NamSor Origin API. The 17,874 records were classified into 71 onomastic classes. Classification of gender was done using the NamSor Gender API. The results obtained showed that 67.7% of author names were classified as male and only 9.9% were classified as female. 24.9% of the names were unclassified. 96.3% of these unclassified names were Asian names. This along with other factors led to the removal of all Asian names from the dataset. This increased the percentage of male and female authors to 87.5 and 11.3 respectively.

In the top right corner of the slide, there is a green line pointing towards the top right, ending with a green dot. In the bottom right corner of the slide, there is a red line pointing towards the bottom right, ending with a red dot.

On the right side of the slide, there is a sidebar with the following text:

lighton.phiri@unza.zm: (25%) Accuracy
(25%) Coverage
(10%) Arguments
(20%) Presentation and Layout
(0%) Personal Reflection

> Paper readings are a nice way of identifying gaps: one of the ways of identifying gaps is contextualising research to our environment: Africa and/or Zambia [...] How would this be adapted to our environment? Can we take advantage of the approach or modified variations of it?
> No mention of the role of Scopus??
> Virtually no personal reflection
> Virtually no critical review/arguments presented, save for the last statement, which is iffy

lighton.phiri@unza.zm: * Research encompasses much more than publishing? Do you think the problem statement and indeed the title is indicative of "Gaps in Computer Science"? Perhaps this should have been tagged "Gaps in CS Publishing?"

lighton.phiri@unza.zm: * Whenever a study draws comparisons with existing literature, you want to draw comparisons
> Are there any shortcomings with the proposed approach when compared with existing literature? Which approach is more reliable and credible? In what instances would the proposed approach be desirable?

Below the sidebar, the text 'permissions on an item' is visible. At the bottom right of the slide, there is a 'Activity' section with a progress bar.

Lecture Series Outline

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

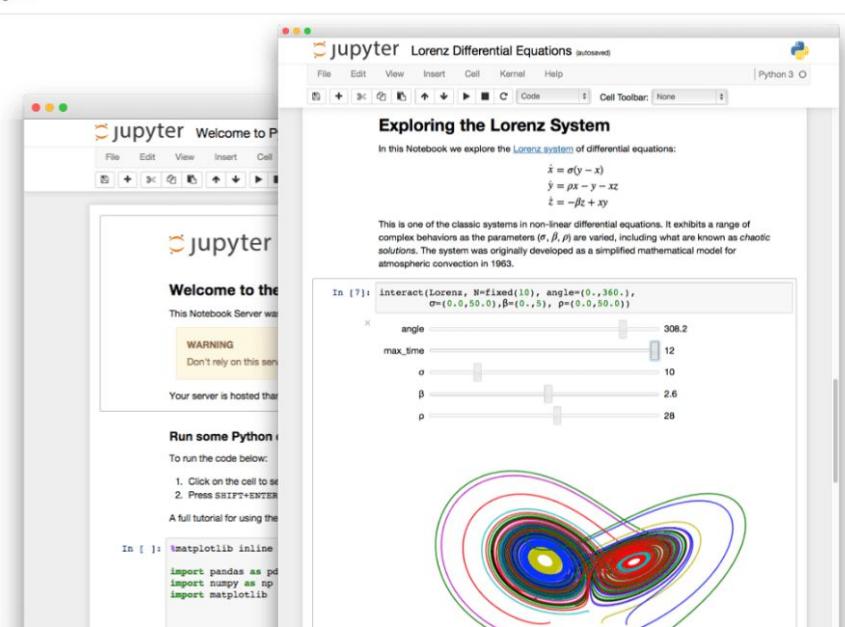
Lecture Series Outline

- 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)



The screenshot shows the Jupyter Notebook interface. On the left, there's a sidebar with the Jupyter logo and a "Welcome to the..." message. The main area displays a notebook titled "Exploring the Lorenz System". It contains text about the Lorenz system, three differential equations, and a paragraph about its history. Below this is a code cell with the command `In [7]: interact(Lorenz, N=fixed(10), angle=(0.,360.), a=(0.0,50.0),beta=(0.,5), p=(0.0,50.0))`. To the right of the code is a plot of the Lorenz attractor, showing a complex, butterfly-shaped trajectory in red, blue, and green. At the bottom of the notebook, there's another code cell with `In []: %matplotlib inline` followed by imports for pandas, numpy, and matplotlib.

Install About Us Community Documentation NBViewer JupyterHub Widgets Blog

The Jupyter Notebook

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Try it in your browser

Install the Notebook







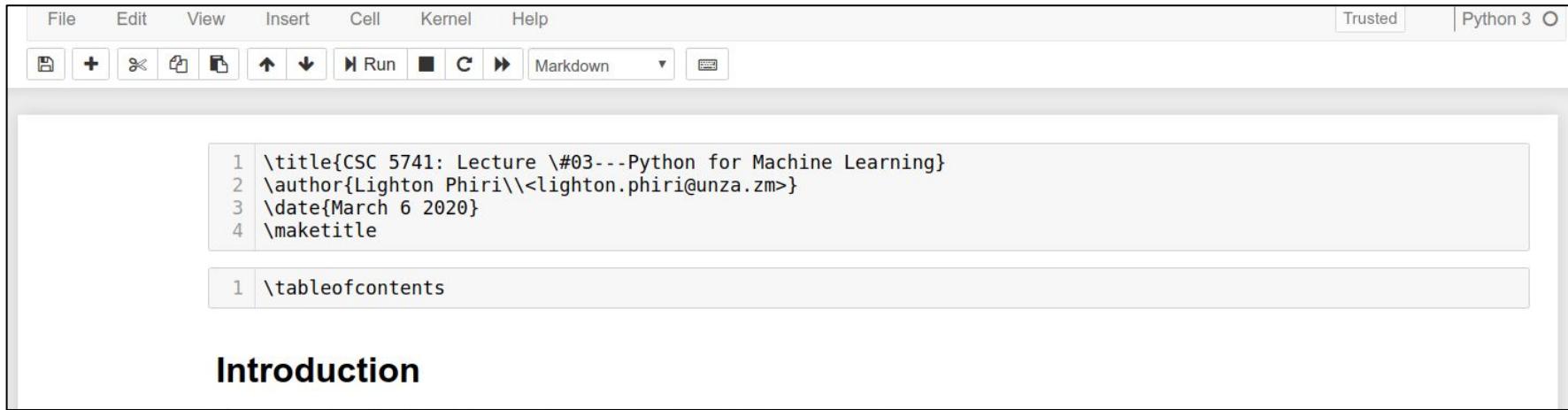


<https://jupyter.org>

March 29, 2021

CSC 5741 (2020/21) L03 - 7

About Jupyter Notebooks (2/2)



The screenshot shows a Jupyter Notebook interface. At the top is a menu bar with File, Edit, View, Insert, Cell, Kernel, and Help. To the right are Trusted and Python 3 buttons. Below the menu is a toolbar with icons for file operations like Open, Save, and New, and cell execution controls like Run, Cell, and Kernel. The main area contains two code cells. The first cell, which is selected, contains the following LaTeX code:

```
1 \title{CSC 5741: Lecture #03---Python for Machine Learning}
2 \author{Lighton Phiri\<lighton.phiri@unza.zm>}
3 \date{March 6 2020}
4 \maketitle
```

The second cell, below it, contains:

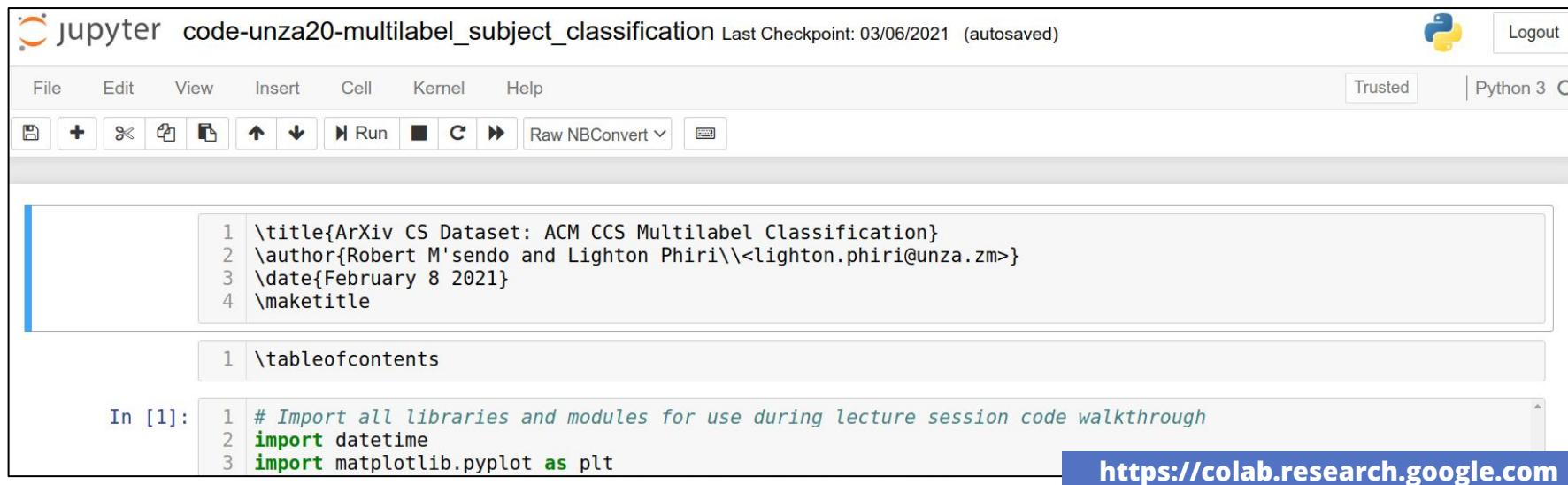
```
1 \tableofcontents
```

Below the cells, the word "Introduction" is visible.

- 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>)



The screenshot shows a Jupyter Notebook interface. At the top, there's a toolbar with icons for file operations like Open, Save, and Run, along with tabs for File, Edit, View, Insert, Kernel, Help, and a Python 3 kernel selector. To the right of the toolbar are buttons for Trusted status and Logout. Below the toolbar, the notebook title is "code-unza20-multilabel_subject_classification" and it indicates "Last Checkpoint: 03/06/2021 (autosaved)". The main area contains several code cells. The first cell (In [1]) contains LaTeX code for a title page:

```
1 \title{ArXiv CS Dataset: ACM CCS Multilabel Classification}
2 \author{Robert M'sendo and Lighton Phiri\<lighton.phiri@unza.zm>}
3 \date{February 8 2021}
4 \maketitle
```

The second cell (In [2]) contains a command to generate a table of contents:

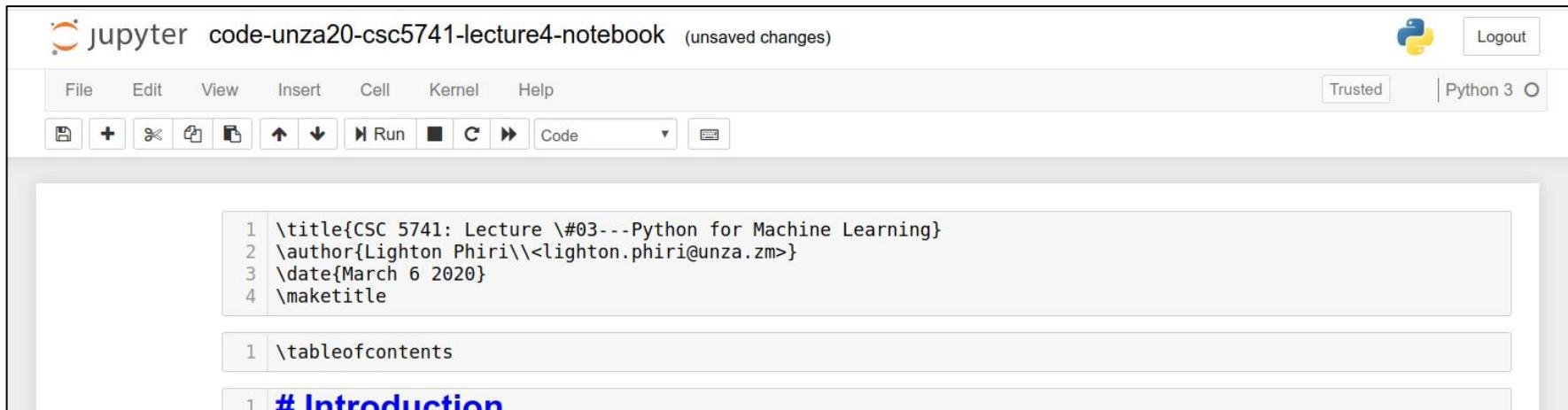
```
1 \tableofcontents
```

The third cell (In [3]) contains Python code to import libraries:

```
1 # Import all libraries and modules for use during lecture session code walkthrough
2 import datetime
3 import matplotlib.pyplot as plt
```

At the bottom right of the interface, there's a URL: <https://colab.research.google.com>.

About Jupyter Notebooks: UI



The screenshot shows a Jupyter Notebook interface. At the top, there's a header with the logo, the notebook title "code-unza20-csc5741-lecture4-notebook" (with "(unsaved changes)" in parentheses), and user information (Python 3). Below the header is a menu bar with File, Edit, View, Insert, Cell, Kernel, and Help. To the right of the menu are buttons for Trusted and Python 3. A toolbar below the menu contains icons for file operations like new, open, save, and run, along with a code dropdown. The main area displays three code cells. The first cell contains LaTeX-like code for a document's metadata:

```
1 \title{CSC 5741: Lecture #03---Python for Machine Learning}
2 \author{Lighton Phiri\\<lighton.phiri@unza.zm>}
3 \date{March 6 2020}
4 \maketitle
```

The second cell contains:

```
1 \tableofcontents
```

The third cell contains:

```
1 # Introduction
```

- **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)

```
1 # Introduction
```

```
2 During these "hands-on" activities, we will explore and experiment the following:
```

- ```
3 1. Jupyter Notebooks---Quick walkthrough of Jupyter Notebooks
4 2. Python 3---Crash course introduction to Python 3
5 3. Core Python Modules---Quick walkthrough of some core Python modules that will be used in the course.
```

```
6
7 8 In all instances, you are encouraged to make reference to online documentation for the various tools.
8 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)

```
1 # Introduction
```

```
2 During these "hands-on" activities, we will explore and experiment the following:
```

- ```
3 1. Jupyter Notebooks---Quick walkthrough of Jupyter Notebooks  
4 2. Python 3---Crash crouse introduction to Python 3  
5 3. Core Python Modules---Quick walkthrough of some core Python modules that will be used in the course.
```

```
6 8 In all instances, you are encouraged to make reference to online documentation for the various tools.  
7 Additionally, you can exploit tools like [Zeal Offline Documentation Browser](https://zealdocs.org) to  
8 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)

```
1 # Introduction
```

```
2 During these "hands-on" activities, we will explore and experiment the following:
```

- ```
3 1. Jupyter Notebooks---Quick walkthrough of Jupyter Notebooks
4 2. Python 3---Crash course introduction to Python 3
5 3. Core Python Modules---Quick walkthrough of some core Python modules that will be used in the course.
```

```
6
7 8 In all instances, you are encouraged to make reference to online documentation for the various tools.
8 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)

## 1 # Introduction

2 During these "hands-on" activities, we will explore and experiment the following:

- 3 1. Jupyter Notebooks---Quick walkthrough of Jupyter Notebooks  
4 2. Python 3---Crash course introduction to Python 3  
5 3. Core Python Modules---Quick walkthrough of some core Python modules that will be used in the course.

6  
7 8 In all instances, you are encouraged to make reference to online documentation for the various tools.  
8 Additionally, you can exploit tools like [\[Zeal Offline Documentation Browser\]](https://zealdocs.org)(<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 # 1. Draw a line plot showing the trends of the Lusaka BNB between November 2016 and April 2018
2 # We will plot BNB as a function of months
3
4 # Format input data points
5 var_bnb_months = ['Nov 16', 'Dec 16', 'Jan 17', 'Feb 17', 'Mar 17', 'Apr 17', 'May 17', 'June 17', 'July 17', 'Aug
6 17', 'Sep 17', 'Oct 17', 'Nov 17', 'Dec 17', 'Jan 18', 'Feb 18', 'Mar 18', 'Apr 18']
7 var_bnb_values =
8 [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.
9 47, 5229.14, 5385.42, 5574.81, 5433.04]
10
11 plt.style.use("ggplot") # Use the visually appealing ggplot R theme
12 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

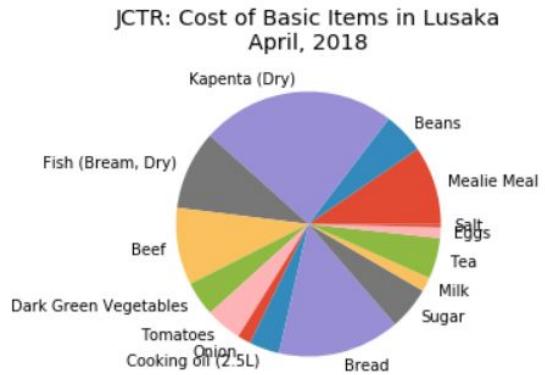
# About Jupyter Notebooks: Code (2/2)

```
In [3]: 1 # 1. Draw a line plot showing the trends of the Lusaka BNB between November 2016 and April 2018
2 # We will plot BNB as a function of months
3
4 # Format input data points
5 var_bnb_months = ['Nov 16', 'Dec 16', 'Jan 17', 'Feb 17', 'Mar 17', 'Apr 17', 'May 17', 'June 17', 'July 17', 'Aug
6 17', 'Sep 17', 'Oct 17', 'Nov 17', 'Dec 17', 'Jan 18', 'Feb 18', 'Mar 18', 'Apr 18']
7 var_bnb_values =
8 [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.
9 47, 5229.14, 5385.42, 5574.81, 5433.04]
10
11 plt.style.use("ggplot") # Use the visually appealing ggplot R theme
12 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

```
Out[6]: Text(0.5, 1.0, 'JCTR: Cost of Basic Items in Lusaka\nApril, 2018')
```



- **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

The screenshot shows the Google Colab interface with a Jupyter notebook titled "code-unza20-csc5741-lecture4-notebook.ipynb". The left sidebar displays a "Code snippets" panel with various options like "Adding form fields", "Camera Capture", and "Cross-output communication". The main content area shows a cell containing a title block and an "Introduction" section. The URL "https://colab.research.google.com" is visible at the bottom right.

code-unza20-csc5741-lecture4-notebook.ipynb

File Edit View Insert Runtime Tools Help Last saved at 5:06 PM

Comment Share L

Code snippets

+ Code + Text

Connect Editing

Filter code snippets

Adding form fields →

Camera Capture →

Cross-output communication →

display.Javascript to execute Java... →

Downloading files or importing dat... →

Adding form fields INSERT

Forms example

Forms support multiple types of fields with type checking including sliders, date pickers, input fields, dropdown menu

\title{CSC 5741: Lecture #03---Python for Machine Learning} \author{Lighton Phiri\lighton.phiri@unza.zm} \date{March 6 2020} \maketitle

Unsupported Cell Type. Double-Click to inspect/edit the content.

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 course 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](#) 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.

<https://colab.research.google.com>

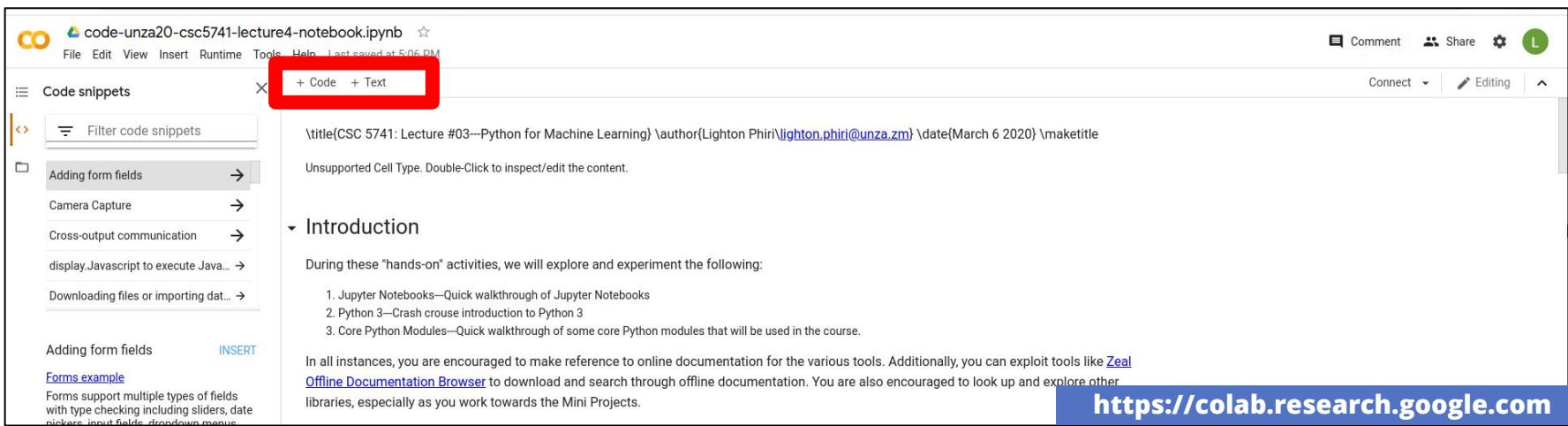
# Google Colab Interface (2/5)

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

The screenshot shows the Google Colab interface. At the top, there's a navigation bar with a 'CO' logo, the file name 'code-unza20-csc5741-lecture4-notebook.ipynb', and a star icon. To the right are 'Comment', 'Share', 'Settings', and a user profile icon. Below the bar is a toolbar with 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a timestamp 'Last saved at 5:06 PM'. A red box highlights the top navigation area. The main workspace contains a sidebar titled 'Code snippets' with sections like 'Adding form fields', 'Camera Capture', 'Cross-output communication', 'display.Javascript to execute Java...', 'Downloading files or importing dat...', and 'Adding form fields' again, with an 'INSERT' button. The main content area shows a code cell starting with '\title{CSC 5741: Lecture #03---Python for Machine Learning} \author{Lighton Phiri\lighton.phiri@unza.zm} \date{March 6 2020} \maketitle'. Below it is a note: 'Unsupported Cell Type. Double-Click to inspect/edit the content.' A section titled 'Introduction' is expanded, containing text about hands-on activities and a numbered list of topics: 1. Jupyter Notebooks—Quick walkthrough of Jupyter Notebooks, 2. Python 3—Crash course introduction to Python 3, and 3. Core Python Modules—Quick walkthrough of some core Python modules that will be used in the course. A note at the bottom encourages referencing online documentation and using tools like Zeal Offline Documentation Browser. A blue bar at the bottom right contains the URL <https://colab.research.google.com>.

# Google Colab Interface (3/5)

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



The screenshot shows the Google Colab interface with a red box highlighting the '+ Code' and '+ Text' buttons in the code snippets sidebar.

Code snippets sidebar:

- + Code
- + Text

Main content area:

```
\title{CSC 5741: Lecture #03---Python for Machine Learning} \author{Lighton Phiri\lighton.phiri@unza.zm} \date{March 6 2020} \maketitle
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

## 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 course 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](#) 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.

<https://colab.research.google.com>

# Google Colab Interface (4/5)

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

The screenshot shows a Google Colab interface with a red box highlighting a specific section of the code snippet notebook.

Code snippets

- + Code
- + Text

\title{CSC 5741: Lecture #03---Python for Machine Learning} \author{Lighton Phiri\lighton.phiri@unza.zm} \date{March 6 2020} \maketitle

Unsupported Cell Type. Double-Click to inspect/edit the content.

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 course 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](#) 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.

<https://colab.research.google.com>

# Google Colab Interface (5/5)

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

The screenshot shows the Google Colab interface. On the left, there is a sidebar titled "Code snippets" which is highlighted with a red box. The sidebar contains several options: "Adding form fields", "Camera Capture", "Cross-output communication", "display.Javascript to execute Java...", "Downloading files or importing dat...", and "Adding form fields" again, which has an "INSERT" button next to it. Below these, there is a link to "Forms example" and a note that "Forms support multiple types of fields with type checking including sliders, date". The main area of the interface shows a code cell with the following content:

```
\title{CSC 5741: Lecture #03---Python for Machine Learning} \author{Lighton Phiri\lighton.phiri@unza.zm} \date{March 6 2020} \maketitle
```

Unsupported Cell Type. Double-Click to inspect/edit the content.

**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 course 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](#) 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.

<https://colab.research.google.com>

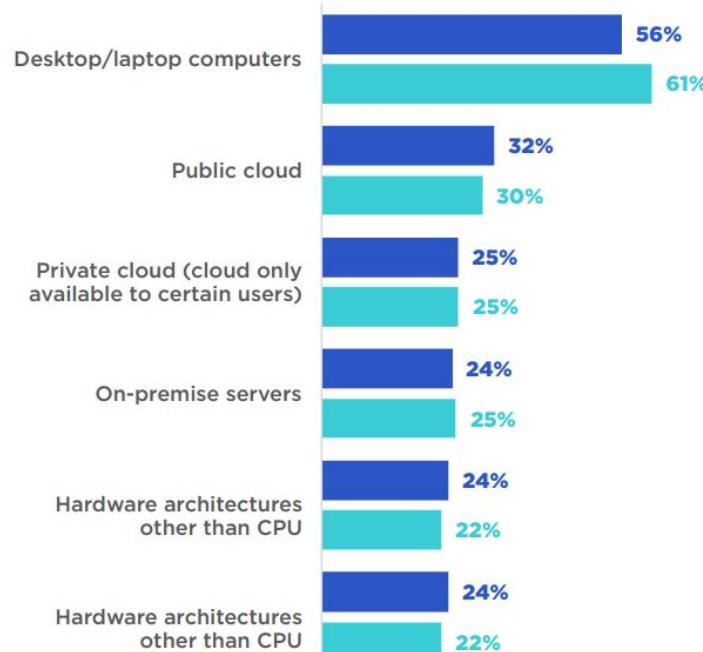
# Lecture Series Outline

- **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

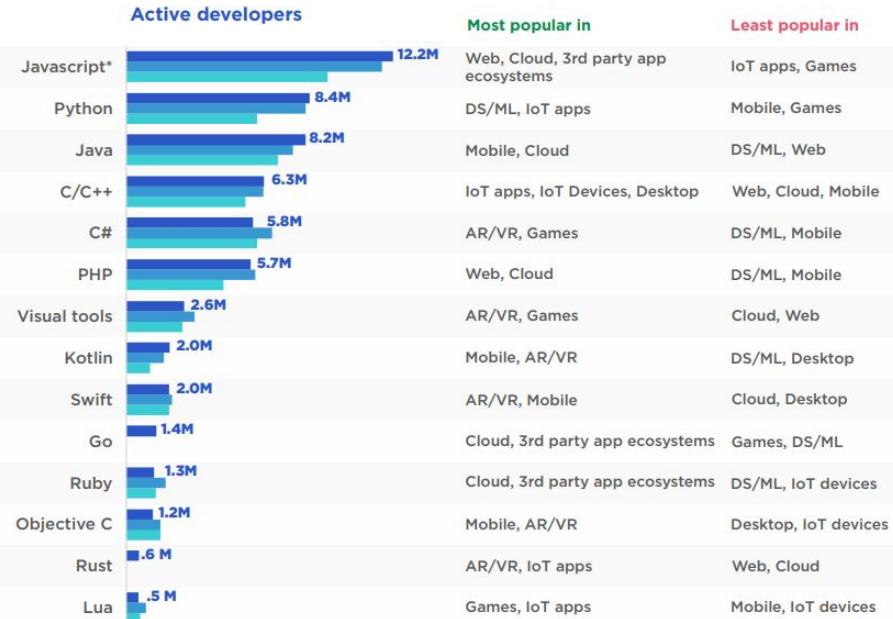
## Where ML developers deploy their code

% of ML developers Q4 2019 (n=2,632) | Q2 2019 (n=2,677)



## JavaScript, Python and Kotlin have grown the fastest in the past two years

Active software developers, globally, in millions Q4 2019 (n=12,066)



# 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**

The screenshot shows the Python.org homepage. The top navigation bar includes links for Python, PSF, Docs (which is highlighted), PyPI, and Jobs. Below the navigation is a search bar and a "Donate" button. The main content area features the Python logo and a navigation menu with links for About, Downloads, Documentation, Community, Success Stories, News, and Events. A central code editor window displays Python code demonstrating list comprehensions and the enumerate function:

```
Python 3: List comprehensions
>>> fruits = ['Banana', 'Apple', 'Lime']
>>> loud_fruits = [fruit.upper() for fruit in
fruits]
>>> print(loud_fruits)
['BANANA', 'APPLE', 'LIME']

List and the enumerate function
>>> list(enumerate(fruits))
[(0, 'Banana'), (1, 'Apple'), (2, 'Lime')]
```

To the right, a sidebar titled "Compound Data Types" explains lists as arrays and provides a link to more information. At the bottom, a quote states: "Python is a programming language that lets you work quickly and integrate systems more effectively." with a "Learn More" link.

<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

The image contains two screenshots from the Stack Overflow website.

The top screenshot shows a question titled "What IDE to use for Python? [closed]". It has 1028 answers and is tagged with "python", "ide", and "editor". The bottom screenshot shows a related question titled "What IDEs ("GUIs/editors") do others use for Python coding?", which has 1256 answers and is also tagged with "python", "ide", and "editor". Both questions were edited on Nov 11 '14 at 1:57.

**Results**

|   |              | Cross Platform | Commercial/Free | Auto Code Completion | Integrated Python Debugging | Bracket Matching |
|---|--------------|----------------|-----------------|----------------------|-----------------------------|------------------|
| 1 |              |                |                 |                      |                             |                  |
| 2 | Atom         | Y              | F               |                      | Y                           | Y                |
| 3 | BlackAdder   | Y              | C               |                      |                             |                  |
| 4 | BlueFish     | L              |                 |                      |                             |                  |
| 5 | ConTEXT      | W              | C               |                      |                             |                  |
| 6 | DABO         | Y              |                 |                      |                             |                  |
| 7 | Dr.Python    |                | F               |                      |                             |                  |
| 8 | DreamPie     |                | F               |                      | Y                           |                  |
| 9 | E-Texteditor | W              |                 |                      |                             |                  |

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

# Installation and Setup (3/3)

The screenshot shows the Visual Studio Code interface with the Python extension details page open. The title bar reads "Extension: Python - python\_crash\_course - Visual Studio Code". The left sidebar shows the Python extension in the extensions marketplace. The main content area displays the Python extension's details, including its name "Python", version "2020.2.64397", developer "Microsoft", download count "17,085,989", and a 5-star rating. It also lists features like Linting, Debugging (multi-threaded, remote), Intellisense, Jupyter Notebooks, code formatting, refactoring, unit tests, snippets, and more. A note indicates it is enabled globally. Below the extension details, there is a "Quick start" section with three steps: installing Python, installing the extension, and opening a Python file. There is also a "Set up your environment" section with instructions to select a Python interpreter from the status bar. The bottom of the screen shows the terminal window with the command "lightonphiri@lightonphiri-Lenovo-ideapad-320-15IKB:~/Projects/work/2020/mis-unza20-csc5741/code/python\_crash\_course\$". A blue bar at the bottom right contains the URL "https://code.visualstudio.com".

# Installation and Setup (3/3)

The screenshot shows the Visual Studio Code interface with the following details:

- Explorer View:** Shows a file tree with the following structure:
  - OPEN EDITORS: code-unza20-csc5741-lecture4-notebook.ipynb
  - PYTHON\_CRASH\_COURSE:
    - .ipynb\_checkpoints
    - code-unza20-csc5741-lecture4-n...
    - csc5741.py
- Code Editor:** The main editor area displays the content of the Jupyter notebook.
- Terminal:** The bottom panel shows two terminal sessions:
  - [9] `ls -ltrh` output:

```
total 124K
-rw-rw-r-- 1 lightonphiri lightonphiri 35K Mar 6 15:12 db-unza19-ict1110_ca_scores_2018_19.ods
-rw-rw-r-- 1 lightonphiri lightonphiri 6.5K Mar 6 15:12 db-unza19-ict1110_ca_scores_2018_19.csv
-rw-rw-r-- 1 lightonphiri lightonphiri 44K Mar 6 16:32 code-unza20-csc5741-lecture4-notebook.pdf
-rw-rw-r-- 1 lightonphiri lightonphiri 33K Mar 6 16:54 code-unza20-csc5741-lecture4-notebook.ipynb
```
  - [-] `ls` output (empty)
- Status Bar:** Shows "Jupyter Server: Not Started" and "No Kernel".
- Tip:** A tooltip in the bottom right corner says: "Tip: you can change the Python interpreter used by the Python extension by clicking on the Python version in the status bar."
- URL:** A blue bar at the bottom right contains the URL <https://code.visualstudio.com>.

# **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)

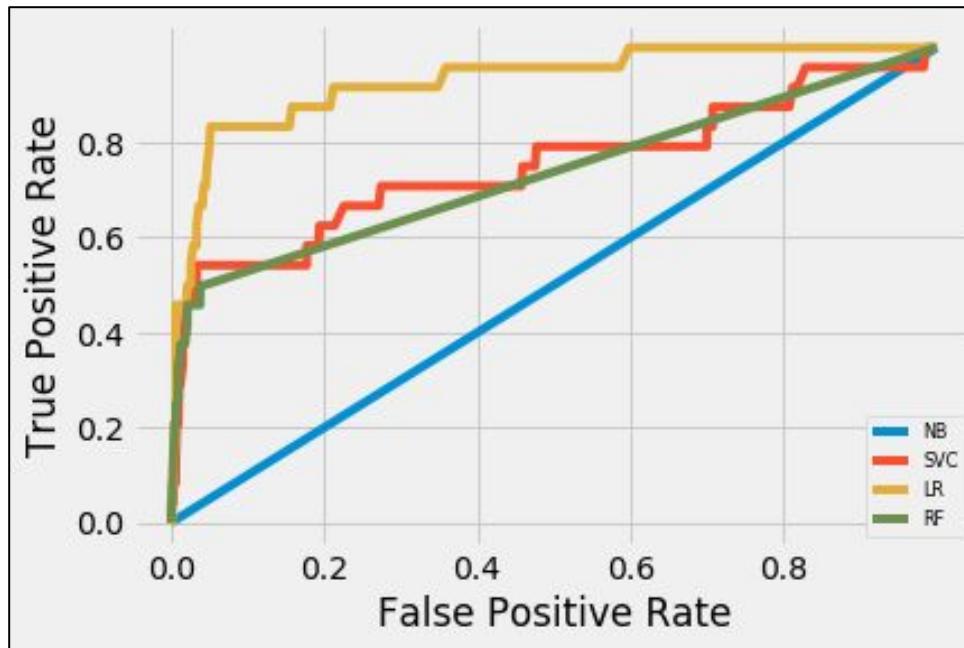
- The matplotlib library is best installed using pip, as with all libraries or using apt-get, if on Mac or Linux
  - pip3 install matplotlib
  - sudo apt-get install python-matplotlib
- Test installation by importing a library module

```
lightonphiri@lightonphiri-Lenovo-ideapad-320-15IKB:~$ pip3 install matplotlib
Requirement already satisfied: matplotlib in ./local/lib/python3.6/site-packages
Requirement already satisfied: python-dateutil>=2.1 in ./local/lib/python3.6/site-pac
Requirement already satisfied: cycler>=0.10 in ./local/lib/python3.6/site-pac
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in ./lo
plotlib) (2.3.1)
Requirement already satisfied: kiwisolver>=1.0.1 in ./local/lib/python3.6/site-
Requirement already satisfied: numpy>=1.10.0 in ./local/lib/python3.6/site-pac
Requirement already satisfied: six>=1.5 in ./local/lib/python3.6/site-packages
1.12.0)
Requirement already satisfied: setuptools in /usr/lib/python3/dist-packages (fr
lightonphiri@lightonphiri-Lenovo-ideapad-320-15IKB:~$ █
```

```
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 matplotlib
>>> dir(matplotlib)
['MatplotlibDeprecationWarning', 'MutableMapping', 'Parameter', 'Path', 'RcParams', 'URL_REGEX',
'PENDIX', '__bibtex__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__',
'__ath__', '__spec__', '__version__', '__version__numpy__', '__warningregistry__', '__add_data_doc__',
'or_data', '__create_tmp_config_dir', '__deprecated_ignore_map', '__deprecated_map', '__deprecated_rema
tials_fmt', '__get_config_or_cache_dir', '__get_data_path', '__get_xdg_cache_dir', '__get_xdg_config_
og', '__logged_cached', '__open_file_or_url', '__parse_commandline', '__preprocess_data', '__rc_params',
'_set_logger_verbose_level', '__verbose_msg', '__version__', '__exit__', 'cbook', 'checkdep_dvipng',
'checkdep_tknscape', 'checkdep_pdftops', 'checkdep_ps_distiller', 'checkdep_usetex', 'colors', 'comp
lib', 'cycler', 'dateutil', 'dedent', 'defaultParams', 'default_test_modules', 'distutils', 'font
ols', 'get_backend', 'get_cachedir', 'get_configdir', 'get_data_path', 'get_home', 'get_label', 'g
importlib', 'inspect', 'interactive', 'io', 'is_interactive', 'is_url', 'locale', 'logging', 'mat
plotlib', 'numpy', 'os', 'pprint', 'pyparsing', 'rc', 'rcParams', 'rcParamsDefault', 'rcParamsOrigi
le', 'rc_file_defaults', 'rc_params', 'rc_params_from_file', 'rcdefaults', 'rcsetup', 're', 'san
', 'stat', 'subprocess', 'sys', 'tempfile', 'test', 'tk_window_focus', 'urllib', 'use', 'validate_
rnings']
```

# 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         |

+ Shell Shell No. 2 Shell No. 4 Shell No. 3

lecture-02 : python3 - Drop-Down Terminal

# 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  | ...             |
| 1  | 2017012891@student.unza.zm | M      | Civic       | Banda    | ...             |
| 2  | 2017012962@student.unza.zm | M      | Languages   | Banda    | ...             |
| 3  | 2017008915@student.unza.zm | M      | Civic       | Bwalya   | ...             |
| 4  | 2017008514@student.unza.zm | M      | History     | Bwalya   | ...             |
| 5  | 2017010497@student.unza.zm | M      | Civic       | Chafuka  | ...             |
| 6  | 2017012923@student.unza.zm | M      | Civic       | Chaibela | ...             |
| 7  | 2017012983@student.unza.zm | M      | History     | Chakulya | ...             |
| 8  | 2017012934@student.unza.zm | M      | Mathematics | Chibale  | ...             |
| 9  | 2017008345@student.unza.zm | M      | Mathematics | Chileshe | ...             |
| 10 | 2017012961@student.unza.zm | M      | Mathematics | Chilumba | ...             |
| 11 | 2017012966@student.unza.zm | F      | Languages   | Chisha   | ...             |
| 12 | 2017012930@student.unza.zm | F      | History     | Gondwe   | ...             |
| 13 | 2017012999@student.unza.zm | M      | Mathematics | Hamaamba | ...             |
| 14 | 2017001325@student.unza.zm | F      | Civic       | Imakando | ...             |
| 15 | 2017012971@student.unza.zm | M      | Mathematics | Jere     | ...             |
| 16 | 2017012980@student.unza.zm | M      | Civic       | Kabaso   | ...             |
| 17 | 2017012932@student.unza.zm | F      | Civic       | Kabwe    | ...             |
| 18 | 2017012973@student.unza.zm | M      | Geography   | Kafwale  | ...             |
| 19 | 2017001431@student.unza.zm | M      | Mathematics | Kamanga  | ...             |

+ Shell Shell No. 2 Shell No. 4 Shell No. 3

lecture-02 : python3 - Drop-Down Terminal

# Pandas (5/9)

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

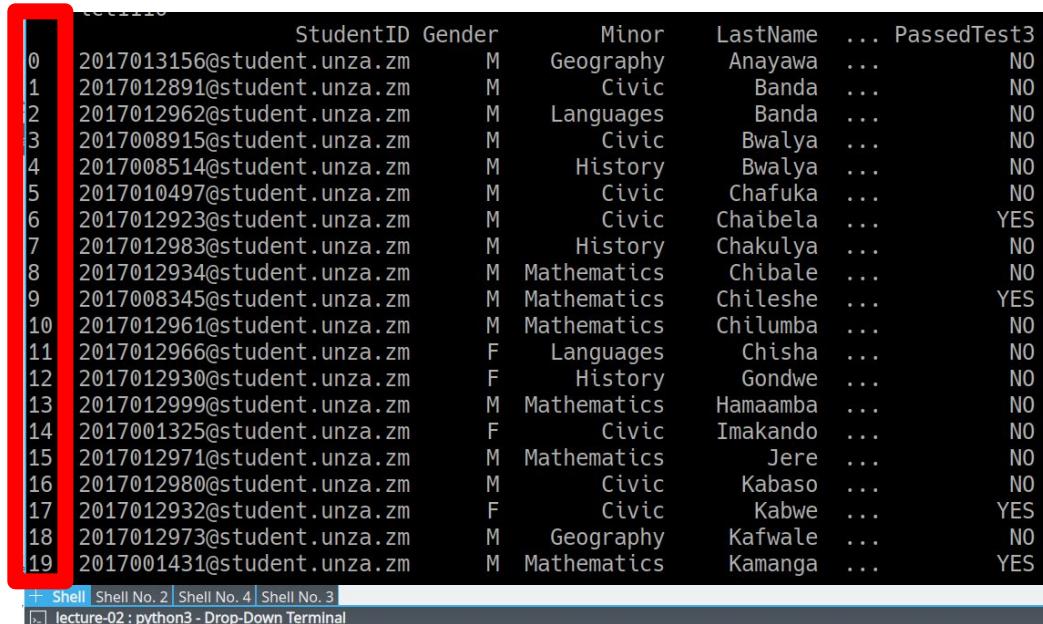
|    |  | StudentID                  | Gender | Minor       | LastName | ... | PassedTest3 |
|----|--|----------------------------|--------|-------------|----------|-----|-------------|
| 0  |  | 2017015158@student.unza.zm | M      | Geography   | Anayawa  | ... | NO          |
| 1  |  | 2017012891@student.unza.zm | 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 | ... | NO          |
| 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         |

+ Shell Shell No. 2 Shell No. 4 Shell No. 3

lecture-02 : python3 - Drop-Down Terminal

# 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 |

+ Shell Shell No. 2 Shell No. 4 Shell No. 3

lecture-02 : python3 - Drop-Down Terminal

# Pandas (7/9)

- Data

|    | StudentID                  | Gender | Minor       | LastName | ... | PassedTest3 |
|----|----------------------------|--------|-------------|----------|-----|-------------|
| 0  |                            |        |             |          |     |             |
| 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/dbb3be0ce9bd5c8b7e3d8
sklearn-0.0.tar.gz
Collecting scikit-learn (from sklearn)
 Downloading https://files.pythonhosted.org/packages/5e/82/c0de5839d613b82bdd08
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__', '__name__', '__package__', '__path__', '__spec__', '__version__', '__config__', 'base', 'clone', 'externals', 'get_config', 'logger', 'logging', 're', 'set_config', 'setup_module', 'showWarnings']
>>>
>>>]
```

# 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

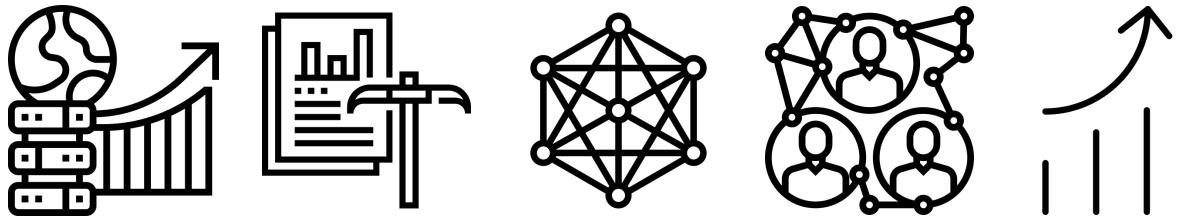
<https://pandas.pydata.org/pandas-docs/version/0.22/10min.html>



✉ @ [csc5741@unza.zm](mailto: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>**