

CSC 5741 (2021/22)

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 28, 2022 (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**

Controlled Vocabularies in Digital Libraries: Challenges and Solutions for Increased Discoverability of Digital Objects

Bertha Chipangila^{1†}, Eric Liswaniso^{1†}, Andrew Mawila^{1†}, Philomena Mwanza^{1†}, Daisy Nawila^{1†}, Robert M'sendo², Mayumbo Nyirenda² and Lighton Phiri^{1*}

¹*Department of Library and Information Science, University of Zambia, P.O Box 32379, Lusaka, 10101, Zambia.

²Department of Computer Science, University of Zambia, P.O Box 32379, Lusaka, 10101, Zambia.

*Corresponding author(s). E-mail(s): lighton.phiri@unza.zm; Contributing authors: 13000438@student.unza.zm; 15058590@student.unza.zm; 15014576@student.unza.zm; 15018148@student.unza.zm; 15019551@student.unza.zm; 20171520216@student.unza.zm; mayumbo.nyirenda@cs.unza.zm;

†These authors contributed equally to this work.

Abstract

Digital Library Systems are widely used in the Higher Education sector, through the use of Institutional Repositories (IRs), to collect, store, manage and make available scholarly research output produced by Higher Education Institutions (HEIs). This wide application of IRs is a direct response to the increase of scholarly research output produced. In order to facilitate discoverability of digital content in IRs, accurate, consistent and comprehensive association of descriptive metadata to digital

Updates on Activities—March 28, 2022 (2/2)

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.

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?

Lecture Series Outline

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

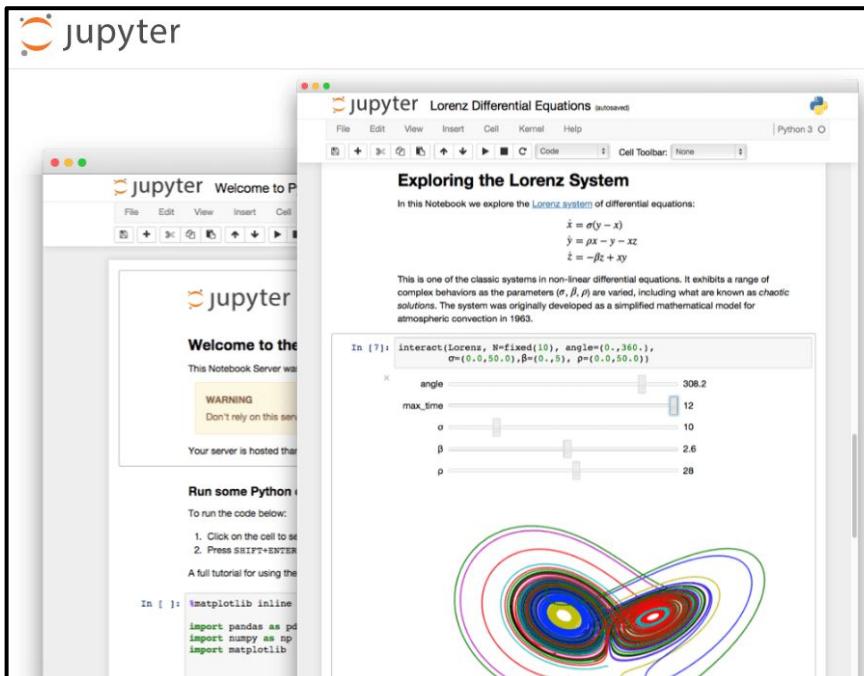
Lecture Series Outline

- 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)

A screenshot of the Jupyter Notebook application. On the left, there's a sidebar with the Jupyter logo and links like 'Welcome to the', 'Run some Python', and 'In []: %matplotlib inline'. The main area shows 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 3D plot of the Lorenz attractor, showing a complex, butterfly-shaped trajectory. A control panel with sliders for 'angle' (308.2), 'max_time' (12), 'a' (10), 'beta' (2.6), and 'p' (28) is also visible.

[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](#)







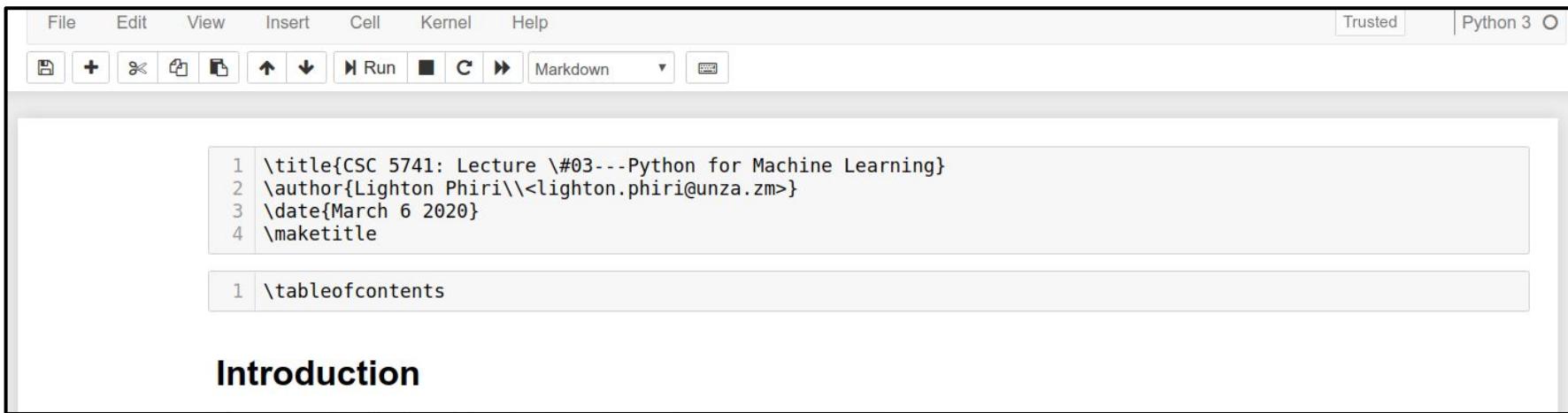


<https://jupyter.org>

March 28, 2022

CSC 5741 (2021/22) 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:

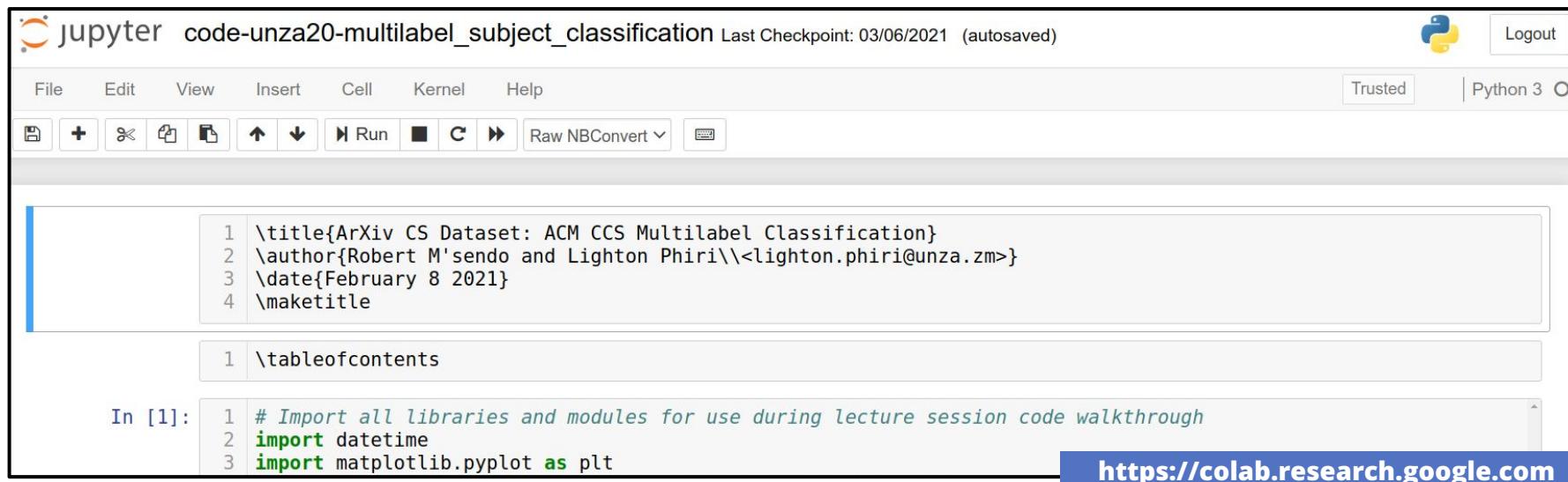
```
1 \tableofcontents
```

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.

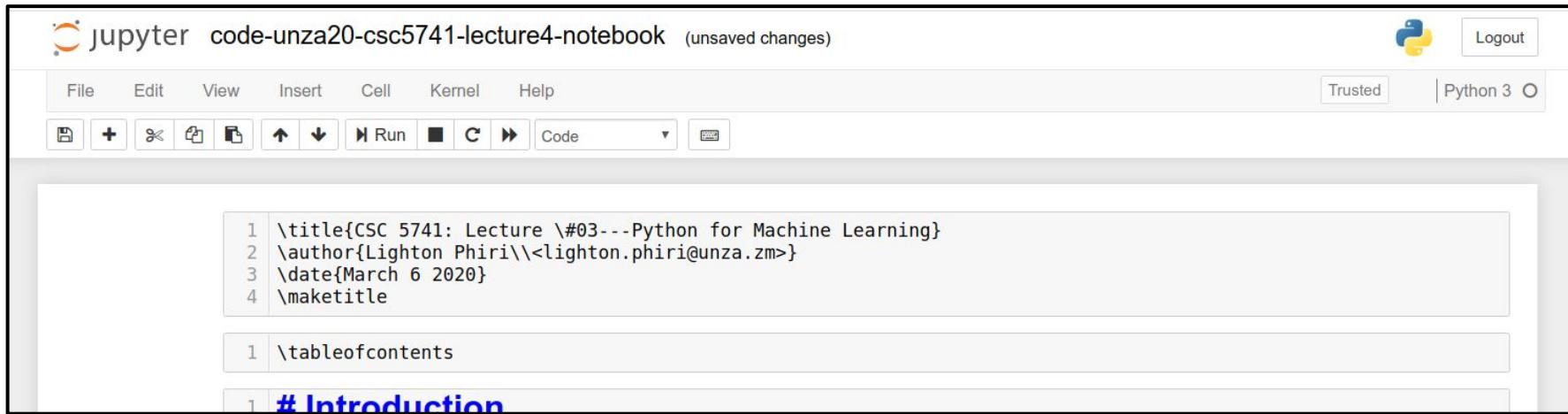
About Jupyter Notebooks: Installation

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



The screenshot shows a Jupyter Notebook interface. At the top, there's a toolbar with various icons for file operations like opening, saving, and running cells. Below the toolbar is a menu bar with File, Edit, View, Insert, Cell, Kernel, and Help. To the right of the menu bar are buttons for Trusted, Python 3, and Logout. The main area contains several code cells. The first cell 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 \maketitleThe second cell contains a command to generate a table of contents:1 \tableofcontentsThe third cell, labeled "In [1]", contains Python code:1 # Import all libraries and modules for use during lecture session code walkthrough
2 import datetime
3 import matplotlib.pyplot as pltA blue bar at the bottom right of the notebook window displays the URL <https://colab.research.google.com>.

About Jupyter Notebooks: UI



The screenshot shows a Jupyter Notebook interface. At the top, there's a header bar with the 'jupyter' logo, the notebook title 'code-unza20-csc5741-lecture4-notebook' (with '(unsaved changes)' in parentheses), a Python logo icon, and 'Logout' button. Below the header is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', and 'Help'. To the right of the menu are 'Trusted' and 'Python 3' buttons. A toolbar below the menu contains icons for file operations like new, open, save, and run, along with a 'Code' dropdown. The main area contains three code cells. The first cell has four lines of LaTeX-like code: 1 \title{CSC 5741: Lecture #03---Python for Machine Learning}, 2 \author{Lighton Phiri\\<lighton.phiri@unza.zm>}, 3 \date{March 6 2020}, and 4 \maketitle. The second cell has one line: 1 \tableofcontents. The third cell has one line: 1 **# Introduction**, which is highlighted in blue.

- **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 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.
```

- **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 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 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.
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.  
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 # 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

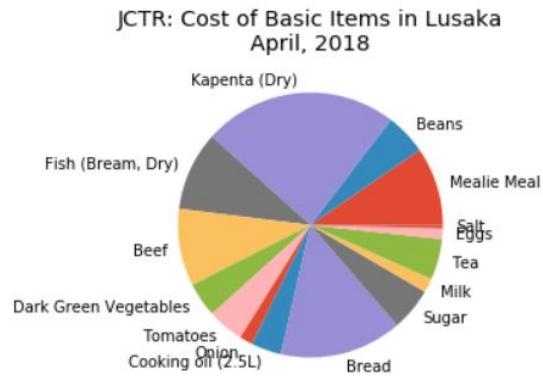
About Jupyter Notebooks: Code (2/2)

```
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 notebook titled "code-unza20-csc5741-lecture4-notebook.ipynb". The left sidebar displays a "Code snippets" section with various options like "Adding form fields", "Camera Capture", and "Cross-output communication". The main content area shows a code cell starting with a title block and an "Introduction" section. A blue bar at the bottom right contains the URL <https://colab.research.google.com>.

\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 'File' menu, followed by 'Edit', 'View', 'Insert', 'Runtime', 'Tools', 'Help', and a status message 'Last saved at 5:06 PM'. A red box highlights the 'File' menu. To the right of the menu are 'Comment', 'Share', 'Settings', and a user profile icon. Below the menu is a sidebar titled 'Code snippets' with a 'Filter code snippets' input field. Under this, there are several items: 'Adding form fields' (selected), 'Camera Capture', 'Cross-output communication', 'display.Javascript to execute Java...', 'Downloading files or importing dat...', 'Adding form fields' again (with an 'INSERT' button), and 'Forms example'. A blue box highlights the 'INSERT' button. The main workspace contains 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
```

Below the code cell is a note: 'Unsupported Cell Type. Double-Click to inspect/edit the content.'

Under the workspace, there's a section titled 'Introduction' with a minus sign icon. It contains the following text:

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

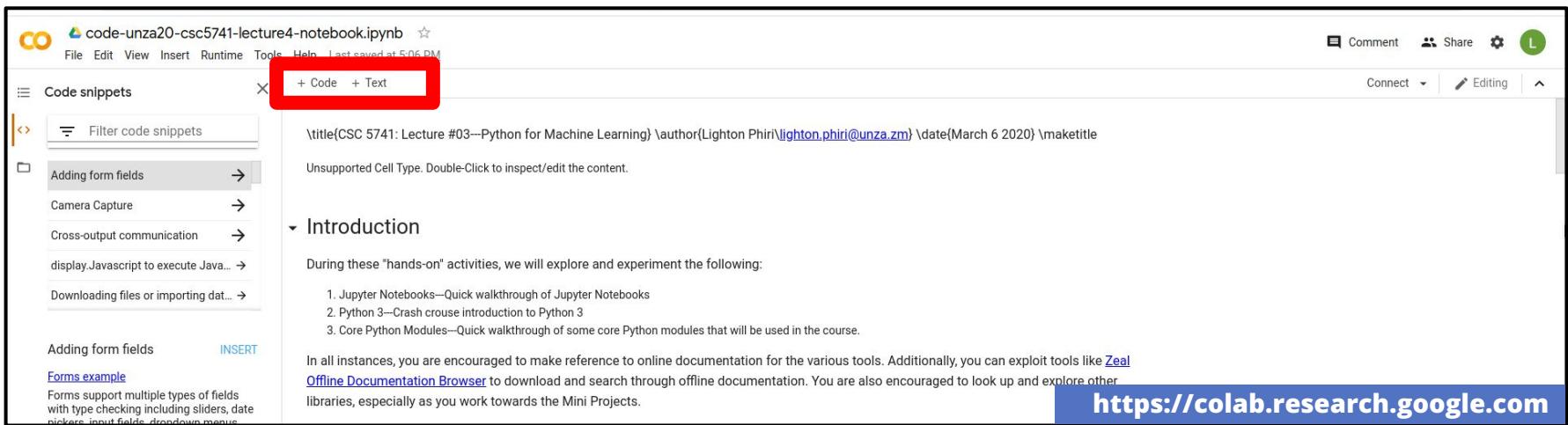
1. Jupyter Notebooks—Quick walkthrough of Jupyter Notebooks
2. Python 3—Crash course introduction to Python 3
3. Core Python Modules—Quick walkthrough of some core Python modules that will be used in the course.

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 (3/5)

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



The screenshot shows the Google Colab interface for a notebook titled "code-unza20-csc5741-lecture4-notebook.ipynb". On the left, there's a sidebar titled "Code snippets" with various options like "Adding form fields", "Camera Capture", and "Cross-output communication". A red box highlights the "+ Code" button at the top of this sidebar. The main content area displays a single cell with the following code:

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

Below the code, a message says "Unsupported Cell Type. Double-Click to inspect/edit the content.". Under the heading "Introduction", it says "During these "hands-on" activities, we will explore and experiment the following:" followed by a numbered list: 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. At the bottom, it encourages users to refer to online documentation and use tools like Zeal Offline Documentation Browser. A blue bar at the bottom right contains the URL <https://colab.research.google.com>.

Google Colab Interface (4/5)

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

The screenshot shows the Google Colab interface with a red box highlighting the main content area. The left sidebar contains a 'Code snippets' section with various options like 'Adding form fields', 'Camera Capture', and 'Cross-output communication'. The main content area displays 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>

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 a search bar "Filter code snippets" and a list of snippet categories: "Adding form fields", "Camera Capture", "Cross-output communication", "display.Javascript to execute Java...", "Downloading files or importing dat...", "Adding form fields", and "Forms example". Below the sidebar, a note states: "Forms support multiple types of fields with type checking including sliders, date". The main workspace shows a notebook 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**

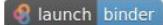
CSC 5741 Code and Datasets

 **lightonphiri** Initial commit 1ecff39 7 minutes ago ⏲ 1 commit

 README.md Initial commit 7 minutes ago

 README.md Edit

About 2021/22 CSC 5741: Data Mining and Warehousing



2021/22 CSC 5741 is a graduate-level course, offered in the [Department of Computer Science \[1\]](#) at [The University of Zambia \[2\]](#). CSC 5741 addresses the concepts, skills, methodologies, and models of data warehousing and data mining. The students are introduced to appropriate techniques for designing data warehouses for various business domains and, concepts for potential uses of the data warehouse and mining opportunities. CSC 5741 also provides students with fundamental concepts and algorithms in the knowledge discovery process such as data pre-processing, data mining and post-process evaluation.

2021/22 CSC 5741: Data Mining and Warehousing

 Readme
 0 stars
 1 watching
 0 forks

Releases
No releases published [Create a new release](#)

Packages
No packages published [Publish your first package](#)

Learning Outcomes

<https://github.com/lightonphiri/misc-unza22-csc5741>

March 28, 2022

CSC 5741 (2021/22) L03 - 25

CSC 5741 Code and Datasets

lightonphiri Added code and datasets ... db7069e 5 minutes ago 3 commits

	code	Added code and datasets	5 minutes ago
	slides	Added slides and notes	17 minutes ago
	README.md	Initial commit	21 minutes ago

README.md

About 2021/22 CSC 5741: Data Mining and Warehousing

[launch binder](#)

2021/22 CSC 5741 is a graduate-level course, offered in the [Department of Computer Science \[1\]](#) at [The University of Zambia \[2\]](#). CSC 5741 addresses the concepts, skills, methodologies, and models of data warehousing and data mining. The students are introduced to appropriate techniques for designing data warehouses for various business domains and, concepts for potential uses of the data warehouse and mining opportunities. CSC 5741 also provides students with fundamental concepts and algorithms in the knowledge discovery process.

<https://github.com/lightonphiri/misc-unza22-csc5741>

2021/22 CSC 5741: Data Mining and Warehousing

Readme
 0 stars
 1 watching
 0 forks

Releases

No releases published [Create a new release](#)

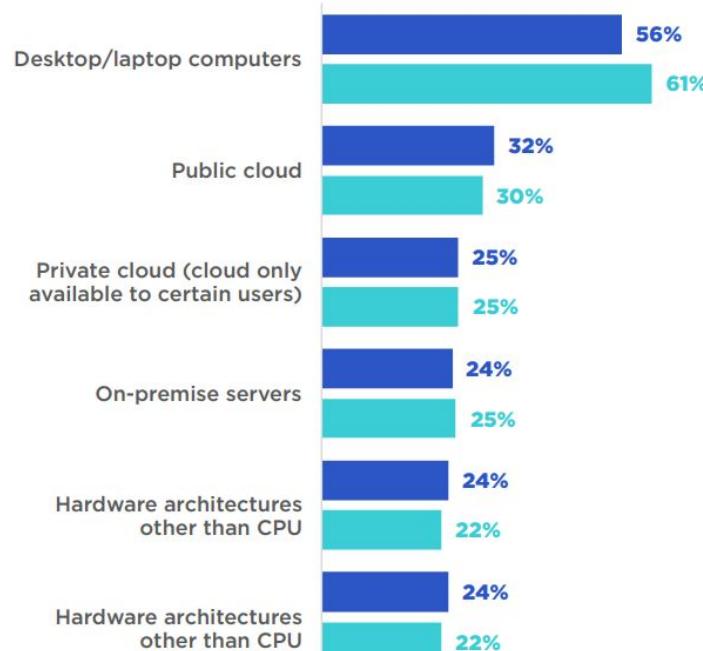
Packages

No packages published [Publish your first package](#)

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

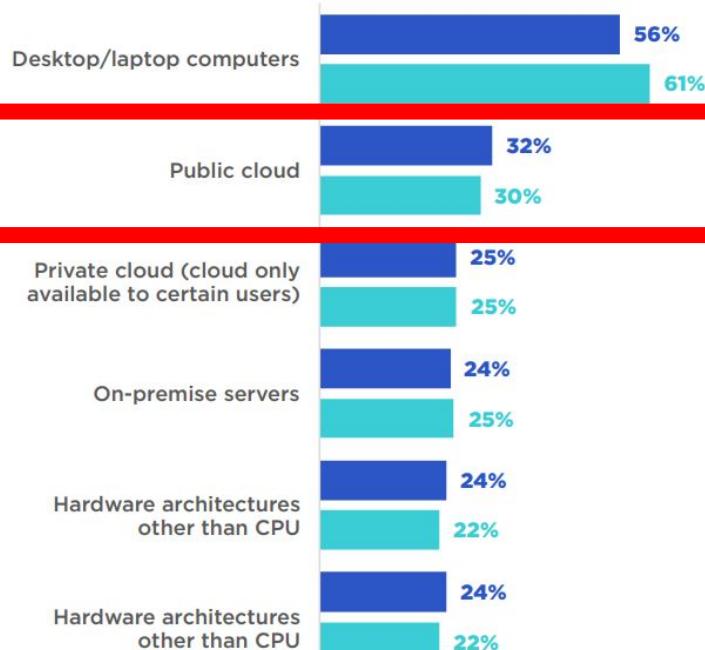
Active software developers, globally, in millions Q4 2019 (n=12,066)



Motivation

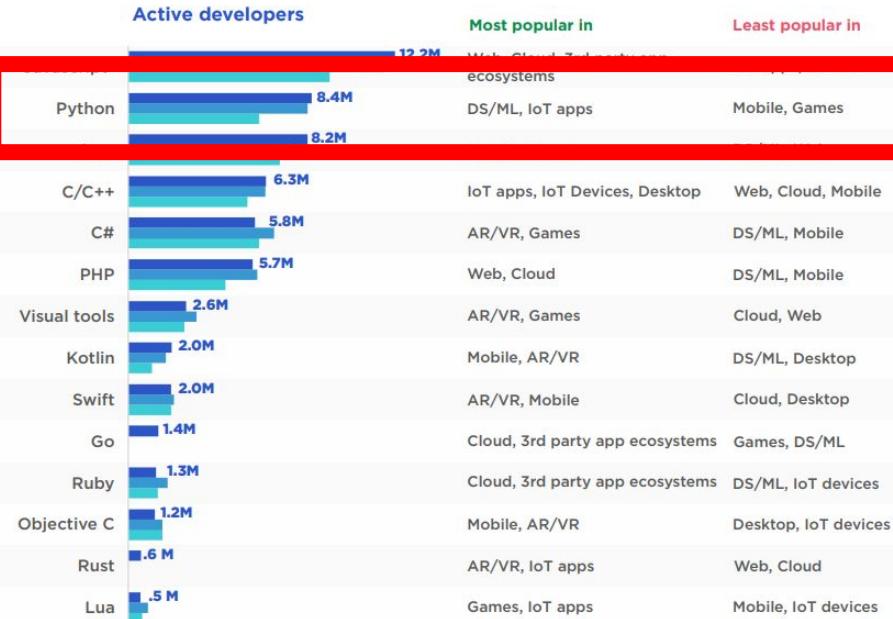
Where ML developers deploy their code

% 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. At the top, there's a navigation bar with tabs for Python, PSF, Docs, PyPI, and Jobs. Below the navigation bar is the Python logo and the word "python™". To the right of the logo are buttons for "Donate" and a search bar. The main content area has a dark blue background. On the left, there's a code editor window displaying Python code:

```
# 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 of the code editor, there's a section titled "Compound Data Types" with the following text:

Lists (known as arrays in other languages) are one of the compound data types that Python understands. Lists can be indexed, sliced and manipulated with other built-in functions. [More about lists in Python 3](#)

At the bottom of the page, there are five small numbered buttons (1, 2, 3, 4, 5). Below them is a brief description of Python: "Python is a programming language that lets you work quickly and integrate systems more effectively. [» Learn More](#)".

<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

What IDE to use for Python? [closed]

What IDEs ("GUIs/editors") do others use for Python coding?

1028 python ide editor

share edit flag

edited Nov 11 '14 at 1:57

1256 Spreadsheet version

		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, version (2020.2.64397), developer (Microsoft), download count (17,085,989), rating (5 stars), repository, and license. It also mentions features like Linting, Debugging (multi-threaded, remote), Intellisense, Jupyter Notebooks, code formatting, refactoring, unit tests, snippets, and more. A note indicates the extension is enabled globally. Below the details are tabs for "Details", "Contributions", and "Changelog".

Python extension for Visual Studio Code

A Visual Studio Code extension with rich support for the Python language (for all actively supported versions of the language: 2.7, >=3.5), including features such as IntelliSense, linting, debugging, code navigation, code formatting, Jupyter notebook support, refactoring, variable explorer, test explorer, snippets, and more!

Quick start

- Step 1. Install a supported version of Python on your system (note: that the system install of Python on macOS is not supported).
- Step 2. Install the Python extension for Visual Studio Code.
- Step 3. Open or create a Python file and start coding!

Set up your environment

- Select your Python interpreter by clicking on the status bar

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

lightonphiri@lightonphiri-Lenovo-ideapad-320-15IKB:~/Projects/work/2020/mis-unza20-csc5741/code/python_crash_course\$

<https://code.visualstudio.com>

Installation and Setup (3/3)

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

- Explorer View:** Shows files in the current workspace, including `code-unza20-csc5741-lecture4-notebook.ipynb`, `csc5741.py`, and other Python files.
- Top Bar:** Displays "Jupyter Server: Not Started" and "No Kernel".
- Content Area:**
 - ## 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.
 - ## Jupyter Notebooks
 - ### Installation
 - Terminal:** Shows command-line output from cell [9] and [-].
- Bottom Status Bar:** Includes a tip about changing the Python interpreter and 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 “2021/22 CSC 5741” Astria and/or Google Drive folder
(https://drive.google.com/drive/folders/1JUcWNxB_pYsncEx3do9ZrpLODX-KHsgL)

Matplotlib (1/7)

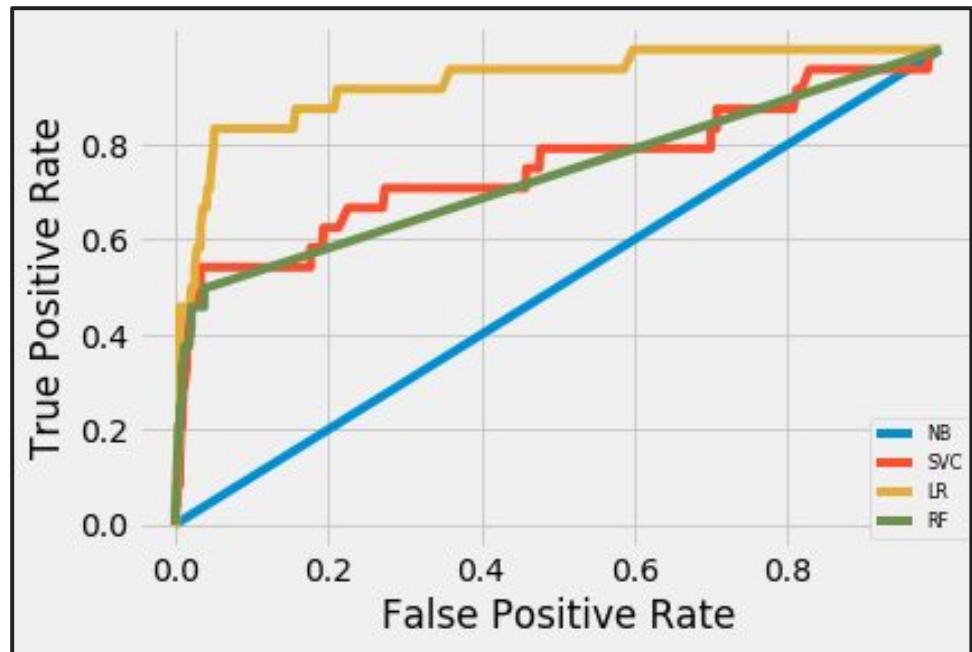
- The matplotlib library is best installed using pip, as with all libraries or using apt-get, if on Mac or Linux
 - pip3 install matplotlib
 - sudo apt-get install python-matplotlib
- Test installation by importing a library module

```
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:~$ █
```

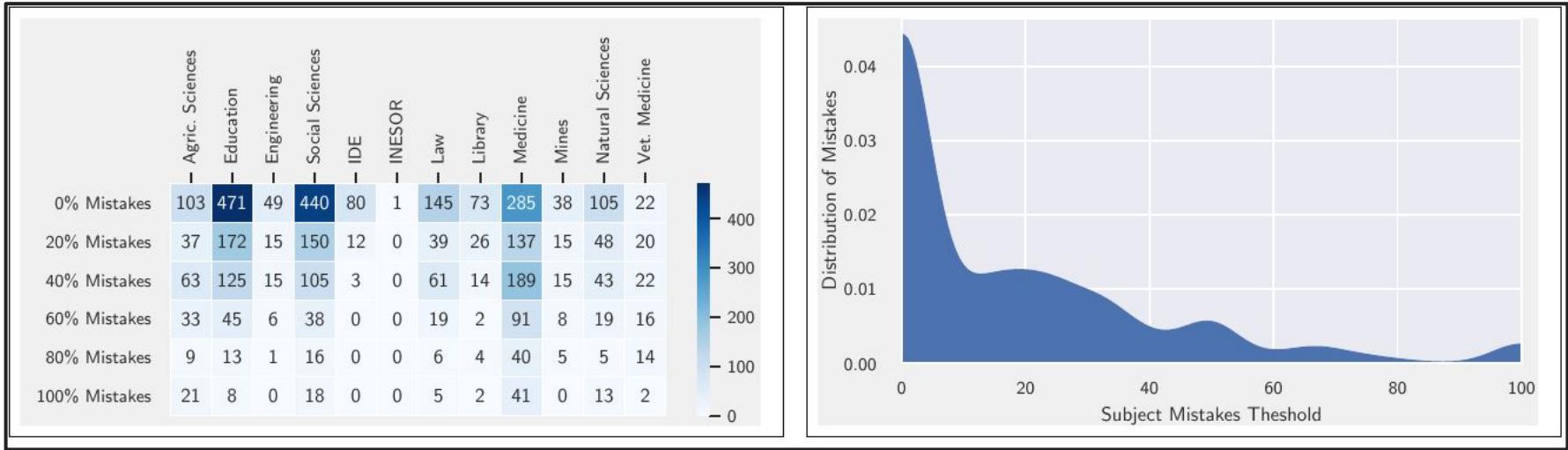
```
lightonphiri@lightonphiri-Lenovo-ideapad-320-15IKB:~$ python3
Python 3.6.7 (default, Oct 22 2018, 11:32:17)
[GCC 8.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import matplotlib
>>> dir(matplotlib)
['_MatplotlibDeprecationWarning', 'MutableMapping', 'Parameter', 'Path', 'RcParams', 'URL_REGEX',
PENDIX', '__bibTeX__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__',
'__path__', '__spec__', '__version__', '__version__numpy__', '__warningregistry__', '__add_data_doc',
'__or_data__', '__create_tmp_config_dir', '__deprecated_ignore_map', '__deprecated_map',
'__deprecated_rem
etails_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', 'dotted', 'defaultParams', 'default_test_modules', 'distutils', 'font
ols', 'get_backend', 'get_cachedir', 'get_configdir', 'get_data_path', 'get_home', 'get_label',
'importlib', 'inspect', 'interactive', 'io', 'is_interactive', 'is_url', 'locale', 'logging',
'mat
plotlib', 'numpy', 'os', 'pprint', 'pyparsing', 'rc', 'RcParams', 'RcParamsDefault',
'rcParamsOri
nial', 'rc_file_defaults', 'rc_params', 'rc_params_from_file', 'rcdefaults', 'rcsetup', 're',
'sani
', 'stat', 'subprocess', 'sys', 'tempfile', 'test', 'tk_window_focus', 'urllib', 'use',
'validate_l
rnings']
```

Matplotlib (2/7)

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



Matplotlib (2/7)



- **Basic elements of a plot**
 - Plot title, Axis labels and 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 “2021/22 CSC 5741” Astria and/or Google Drive folder
(https://drive.google.com/drive/folders/1JUcWNxB_pYsncEx3do9ZrpLODX-KHsgL)

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

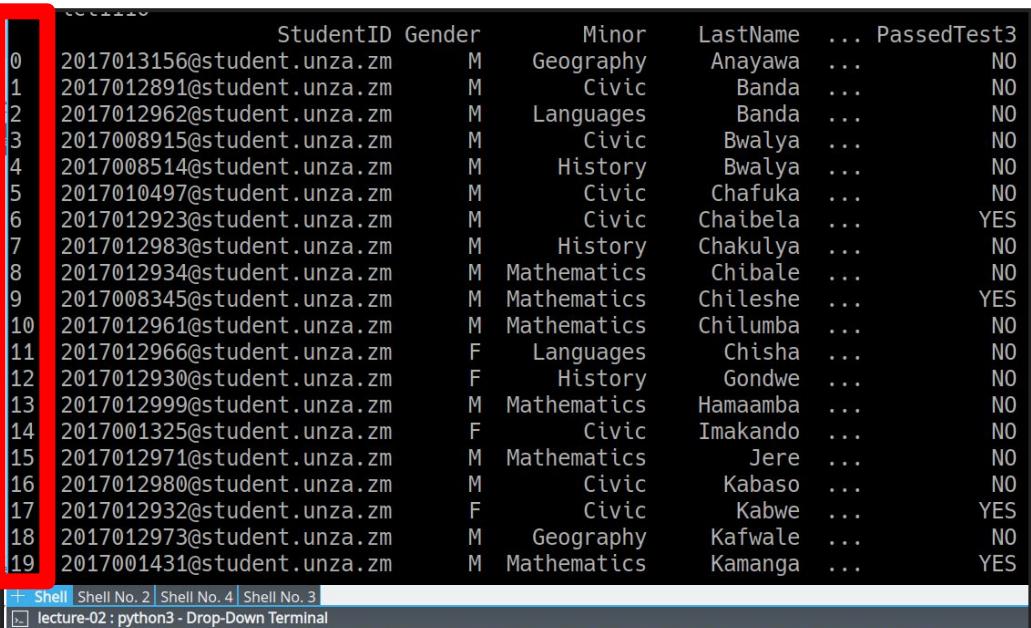
	StudentID	Gender	Minor	LastName	...	PassedTest3
0	2017015158@student.unza.zm	M	Geography	Anayawa	...	NO
1	2017012891@student.unza.zm	M	Civic	Banda	...	NO
2	2017012962@student.unza.zm	M	Languages	Banda	...	NO
3	2017008915@student.unza.zm	M	Civic	Bwalya	...	NO
4	2017008514@student.unza.zm	M	History	Bwalya	...	NO
5	2017010497@student.unza.zm	M	Civic	Chafuka	...	NO
6	2017012923@student.unza.zm	M	Civic	Chaibela	...	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 (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
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 “2021/22 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 “2021/22 CSC 5741: Lecture #04 Notebook—Python for Machine Learning” (<http://bit.ly/2Q2T2Lw>)

Bibliography

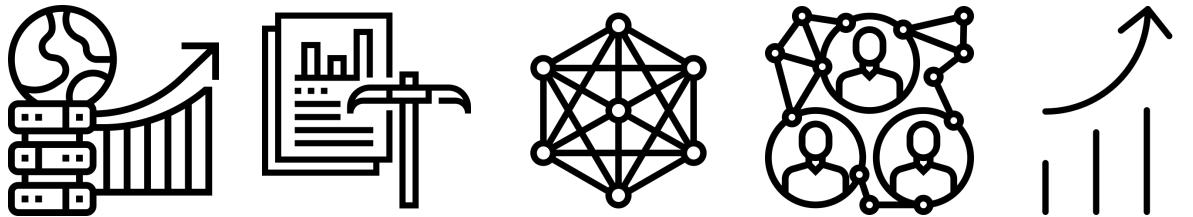
- [1] **A Byte of Python**
<https://python.swaroopch.com>
- [2] **Python 3.4 Programming Tutorials**
<https://www.youtube.com/playlist?list=PL6gx4Cwl9DGAcbMi1sH6oAMk4JHw91mC>
- [3] **Python for Beginners | Python.org**
<https://www.python.org/about/gettingstarted>
- [4] **Pyplot tutorial – Matplotlib 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 (2021/22)

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>