BASIC PYTHON FOR DATA SCIENCE

Engr. Chinelo Okafor

BASIC PYTHON FOR DATA SCIENCE

Engr. Chinelo Okafor

BASIC PYTHON
FOR
DATA SCIENCE

Volume One

For Beginners with No Coding Knowledge

BASIC PYTHON
FOR
DATA SCIENCE

Volume One

For Beginners with No Coding Knowledge

Engr. Chinelo Okafor

ISBN: 978-978-60211-1-9

Published in Nigeria by Winepress Publishing

Winepress Publishing
Suite 223, Ogun-Oshun River Basin Development Authority, Opposite Palms Shopping Mall, Off Oni Memorial Children’s Hospital, Ring-Road, Ibadan
Telephone: +234 809 816 4359 | +234 805 316 4359
Email: hello@noirledge.com | www.noirledge.com
twitter.com/noirledge | facebook.com/noirledge | instagram.com/noirledge

Winepress Publishing is an imprint of Noirledge Limited. For information regarding discounts on bulk purchases and special editions of our titles, please contact our Sales Department via hello@noirledge.com or +234 809 8164 359.

Cover Design: Dhee Sylvester
Book Design: Servio Gbadamosi
Typesetting: Rukayat Amudah
Technical Team: Solomon Ogungbenro
Printed and bound in Nigeria by Noirledge Limited

DEDICATION

This book is dedicated to God's greater glory. To my loving husband, Ikechukwu, who never gave up on my potential, and to my son, Obichukwu, for being very supportive.

WHO CAN BENEFIT FROM THIS BOOK?

Basic Python for Data Science (Volume One) serves as a foundational course ideal for individuals aspiring to pursue careers in the following domains:

- Data Science
- AI Engineering
- Data Engineering
- Cloud Computing
- Machine Learning Engineering
- Python Programming
- Robotics Engineering, and more.

Basic Python for Data Science stands out for several reasons:

1. It offers a video version accessible online, providing an interactive learning experience. (Find more information at www.daisytech limited.org)
2. The coding is presented in a simplified manner and incorporates locally relevant content, making it accessible to learners without prior knowledge.
3. The course caters to a diverse audience, including:
   - Secondary School Students
   - Undergraduates
   - Graduates
   - Postgraduates
   - Professionals

Now that you're informed, it's time to embark on your learning journey.

We value your input; feel free to share your feedback with us via email: info@daisytechlimited.org.

HOW TO USE THIS BOOK

This book is easy to use, THE CODES ARE WRITTEN IN DARKER COLORS while the OUTPUT is separated with a COLON (:).

The book is divided into five sections; and the reader is advised to study it systematically in the arranged order. This is because each new section builds on the knowledge of the preceding one. The sections are:

Section One: Overview of Data Analytics—This section introduces you to the world of Data Analytics and why it is important in business. It also guides you on how to get started with the installation of the software and other options you can explore if you do not want to install the software yet. It guides you on how to write your first code too. Yay!

Section Two: Python Data Types—This section helps you understand the major data types in Python and how you can work with them. You are also given lots of examples and practice questions, you would also learn how to use Python as a calculator, supercool right?

Section Three: Data Structures—These are other classes of Data Types that you will need to write your data analysis codes. Here we talk about lists, tuples, dictionaries, etc. and how you can work with them.

Section Four: Conditions and Branching—Now that you have familiarized yourself with these codes, it is time to put them into practice, this section will guide you on how to achieve that.

Section Five: Reading and Writing Files with Open—This is the last section that guides you on how to read and write files in Python, and create your own folder, is this not cool?

You can access the video version of this book, Basic Python for Data Science (Volume One) using this link: http:\\daisytechlimited .org\courses\basic-python-for-data-science

A 50% Discount Coupon for the course is included on the last page of the book.

SECTION ONE

OVERVIEW OF DATA ANALYTICS

- **What is Data Analytics?**
- **Importance of Data Analytics**
- **Five Data Analytics Processes**

- **Explanation of Data Analytics Processes**
- **Installation of Anaconda**
- **If I Cannot Install Python?**
- **Exploring the Jupyter Environment**

WHAT IS DATA ANALYTICS?

Data is a collection of facts or observations.

TYPES OF DATA

Structured Data: Highly organized information such as databases, spreadsheets, and any information that is easily accessible.
Unstructured Data: Not organized information such as Text, video, audio, and pictures.
Data Analytics is the science of examining raw data in order to draw conclusions about the information. It impacts tremendously how the organization makes decisions.

IMPORTANCE OF DATA ANALYTICS

Data Analytics helps organizations to achieve the following:

- Identify growth opportunities.
- Drive innovation
- Operate more efficiently
- Manage risks in new ways

FIVE DATA ANALYTICS PROCESSES

EXPLANATION OF DATA ANALYTICS PROCESSES

1. Ask Questions

Either you're given data and ask questions based on it, or you ask questions first and gather data based on that later. In both cases, great questions help

you focus on relevant parts of your data and direct your analysis towards meaningful insights.

2. Wrangle Data

You get the data you need in a form you can work with in three steps: **gather, assess, and clean.** You gather the data you need to answer your questions, assess your data to identify any problems in your data's quality or structure and clean your data by modifying, replacing, or removing data to ensure that your dataset is of the highest quality and well-structured.

3. Perform EDA (Exploratory Data Analysis)

You explore and then augment your data to maximize the potential of your analyses, visualizations, and models. Exploring **involves finding patterns in your data,** visualizing relationships in your data, and building intuition about what you're working with. After exploring, you can do things like remove outliers and create better features from your data, also known as feature engineering.

4. Draw Conclusions (or even make predictions)

This step is typically approached with machine learning or descriptive statistics.

5. Communicate Your Results

You often need to justify and convey meaning in the insights you've found. **Data visualization** will always be very valuable.

INSTALLATION OF ANACONDA

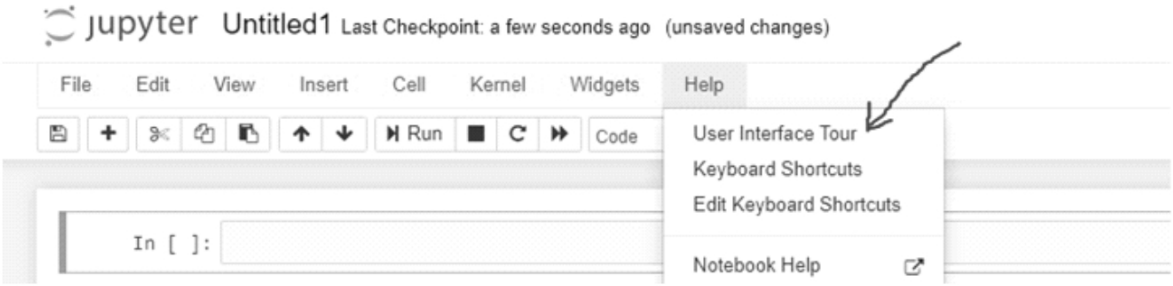
Click on the link to install the Python package

https://www.anaconda.com/products/individual

Scroll down on the site to choose the bits and window type for your computer.

Then click to install.
Install based on recommended settings.

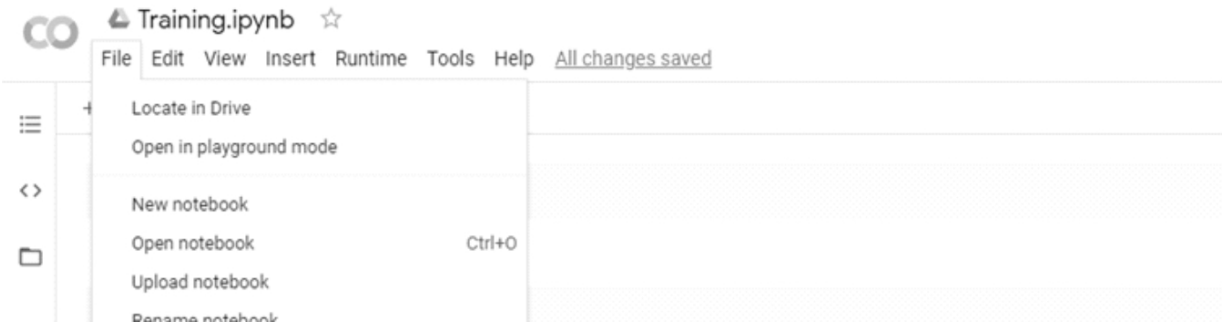
As you launch the Jupyter environment click new -> Python 3.



IF I CANNOT INSTALL PYTHON?

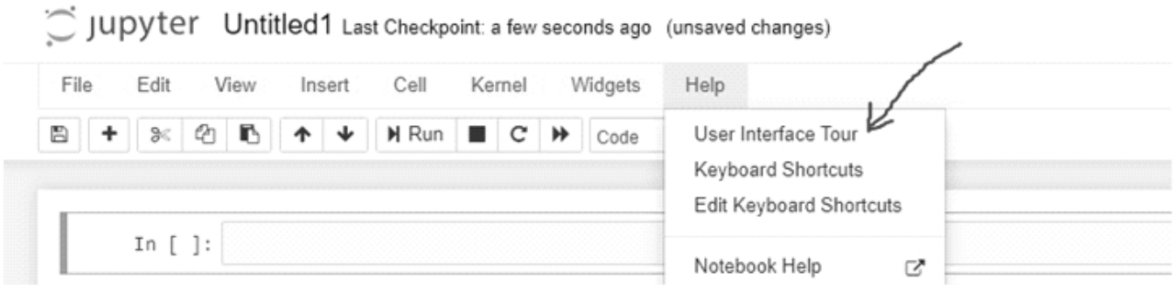
You do not need to worry.

Go to Google -> Login with your Gmail -> Type Google Colab -> click on new notebook -> rename the file. It can be downloaded.



EXPLORING THE JUPYTER ENVIRONMENT

To get a tour of the environment, click on the user interface tour. That will be your tour guide.

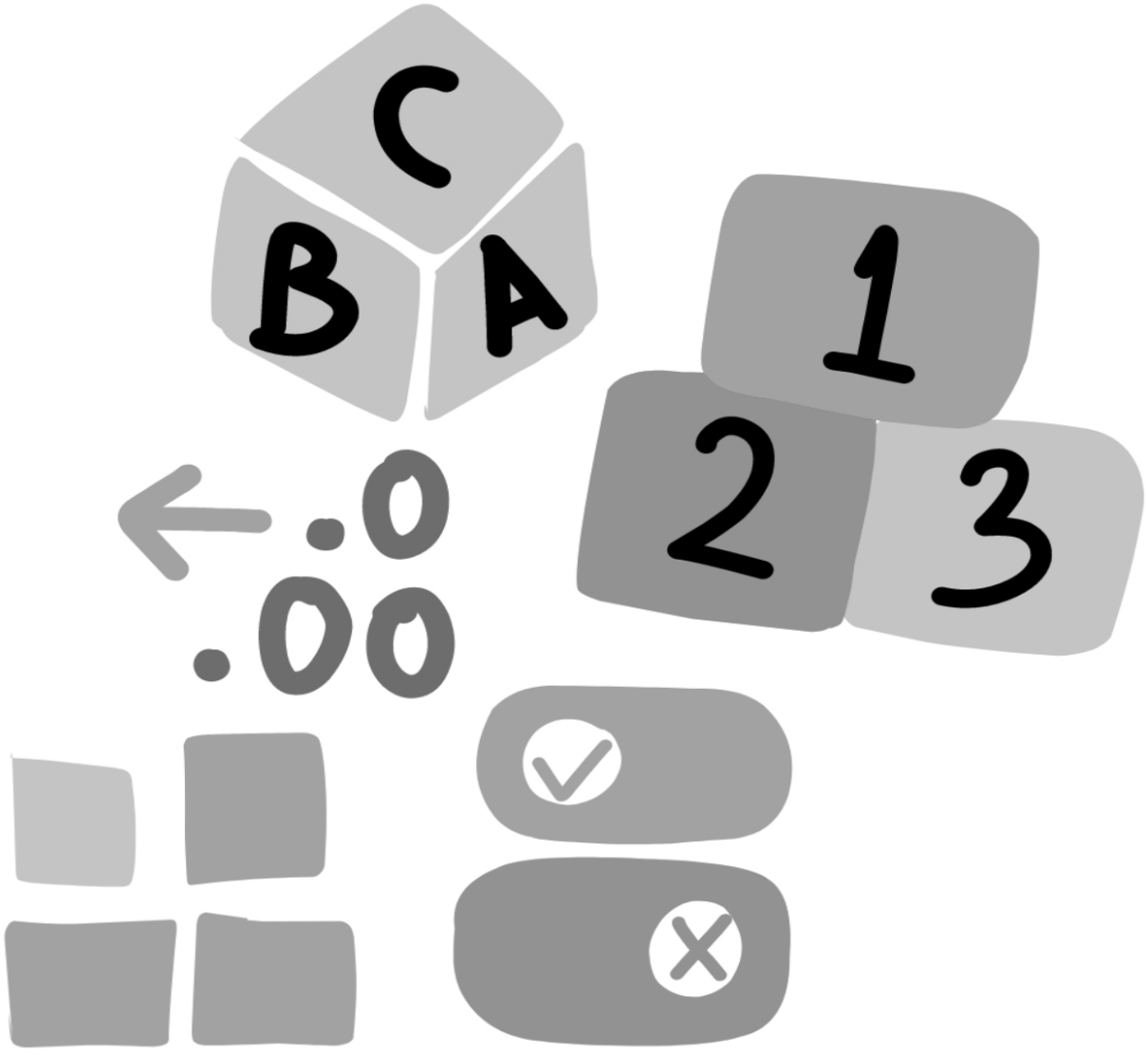


ACTIVITY 1

1. What do you hope to achieve at the end of this course ?
2. Define each Data Analytics process in your own words.
3. Type your first code. Print(“Hello Nigeria”), snap and attach to send.

SECTION TWO

PYTHON DATA TYPES



- **Data Types**

  Conversion From A Data Type To Another

Boolean: A Data Type
Activity 2.1

- **Expression and Variables**
  - Expressions
  - Special Operators
  - Activity 2.21
  - Variables
  - Activity 2.22
- **Mutable and Immutable Objects**
  - String
  - Activity 2.23

DATA TYPES

A type is different ways Python represents different types of data.

3 Data Types in python

1.Integer – int e.g. 23 (negative or positive numbers)
2. Float – float e.g. 23.12 (negative or positive numbers decimal numbers)
3.String – str e.g. “Lagos is great”

Note: Python is case-sensitive, Low letters are encouraged as good practice.

SHIFT + ENTER to run the code or click on

CONVERSION FROM ONE DATA TYPE TO ANOTHER

float(5) : 5.0
int(6.3) : 6
int(“45”) : 45
int(“Imo”) : Error
str(67) : “67”
str(45.3) : “45.3”

BOOLEAN: A DATA TYPE

A boolean takes 2 values which are True and False, Upper cases are used.

Example

type(True) : bool

True	1
False	0

ACTIVITY 2.1

Check the Data Type in the Python environment.

1 type(23)

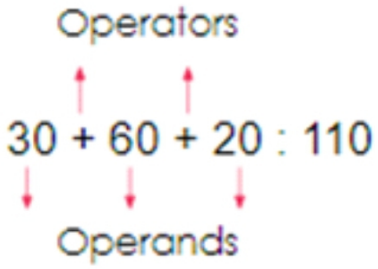
2.type(23.3)
3.type(“Calabar is good for tourism”)

Boolean Values

1.int(False)
2. int(True)
3.bool(1)
4.bool(0)

EXPRESSIONS AND VARIABLES

Expression: defines the type of operations the computer performs. It defines the type of operation Python performs. Example



EXPRESSIONS

SYMBOL	OPERATOR
+	Addition
-	Subtraction
*	Multiplication
**	Exponentiation
\ (Foward Slash)	Division
\\	Integer Division
%	Modulo

SPECIAL OPERATORS

** called exponentiation is a symbol for square root in Python.

Example:
22** : 4
/ called forward slash is a symbol of division in Python.
Example:
86/2 : 43.0
When 2 integers divide themselves, the answer is in float.
// called Integer division, the result is rounded up to a whole number.
Example:
23//4 : 5

Python Follows mathematical conventions when performing mathematical expressions.
BODMAS or PEDMAS

SPECIAL OPERATORS CONT'D

BODMAS	PEDMAS
• Bracket()	• Parenthesis()
• Of	• Exponentiation
• Division	• Division
• Multiplication	• Multiplication
• Addition	• Addition
• Subtraction	• Subtraction

Example
50+20*3-6 **Note** : Adding brackets makes the expression readable -> (50+ (20*3))-6

ACTIVITY 2.21

Perform the following operations in the Jupyter environment.

50+25
48-23
625*223
60/6
17893/114
50*100-200+4**3

VARIABLES

Variables are used to store values in Python. Assignment of variables reduces code modification when we should change values in a code. You can call up a value through a variable name.

Example:

daisy_tech = 15

daisy_tech : 15

When you reassign another value to a variable, the old one is discarded, and only the new one is recognized.

Example:

daisy_tech = 23

daisy_tech : 23

It is advised to use variable names related to your data description.

Example

total_min = 60+20+50

total_hr = total_min/60

total_hr : 2.16666

ACTIVITY 2.22

height = 5

weight = 75

BMI = weight
height2**

BMI: output

What will be the BMI when the height is

a) 6 b) 7 c) 4

MUTABLE AND IMMUTABLE OBJECTS

Immutable Objects	Mutable Objects
• Integer	• List
• Float	• Set
• String	• Dictionary
• Tuple	

Immutable means the value of the object cannot be changed but a new on can be created.

STRING

To work with text in python you have to use string.

A string is an immutable sequence of characters such as Letters, spaces, numbers, and symbols. A string can be created by using single or double quotes.

Example:

“This is Abuja” or 'This is Abuja’

Rules for writing string

• Backslash is used for escaping quotes in a sentence.

Example:

calabar_tour = “I think calabar is a good place for touring, isn't it?”

print(calabar_tour)

Note: The double quote used will accept the apostrophe sign in isn't it?

• \t represents a tab or white space. Example:

print(“First \t Second”) : First Second

• \n represents a new line. Example:

print('First \nSecond') : First Second

• + is used to concatenate strings together while * is used to repeat strings.

Example:

letter_one = “Come soon”
letter_two = “Bye”
Print (letter_one + “ “+ letter_two) : Come soon Bye

word = “Lagos”
print(word*3) : lagoslagoslagos

• Len() is a built-in function that tells us the length of an object.
Example:

```
abuja_way = len('Airport')
      print(abuja_way)
```

• If you want to place \ in your string, it will pass.
Example:

```
print('Tranquility\Peaceful')
```

STRING METHODS

1. Upper()
Example:

```
nig_pop = “nigeria has the largest population in the world”
                  pop_stat = nig_pop.upper()
                   pop_stat
```

2. Replace()
Example:

```
nig_pop = “Nigeria has the largest population in the world”
  world_stat = nig_pop.replace('Nigeria', 'China')
  world_stat
```

3. find() the output is the first value of the sequence.
Example:

```
author = 'CHINUA  ACHEBE'
```

C	H	I	N	U	A		A	C	H	E	B	E
0	1	2	3	4	5	6	7	8	9	10	11	12

Author.find('EB') : ?

Note: if the string is not found in the sequence of letters, it will output -1.

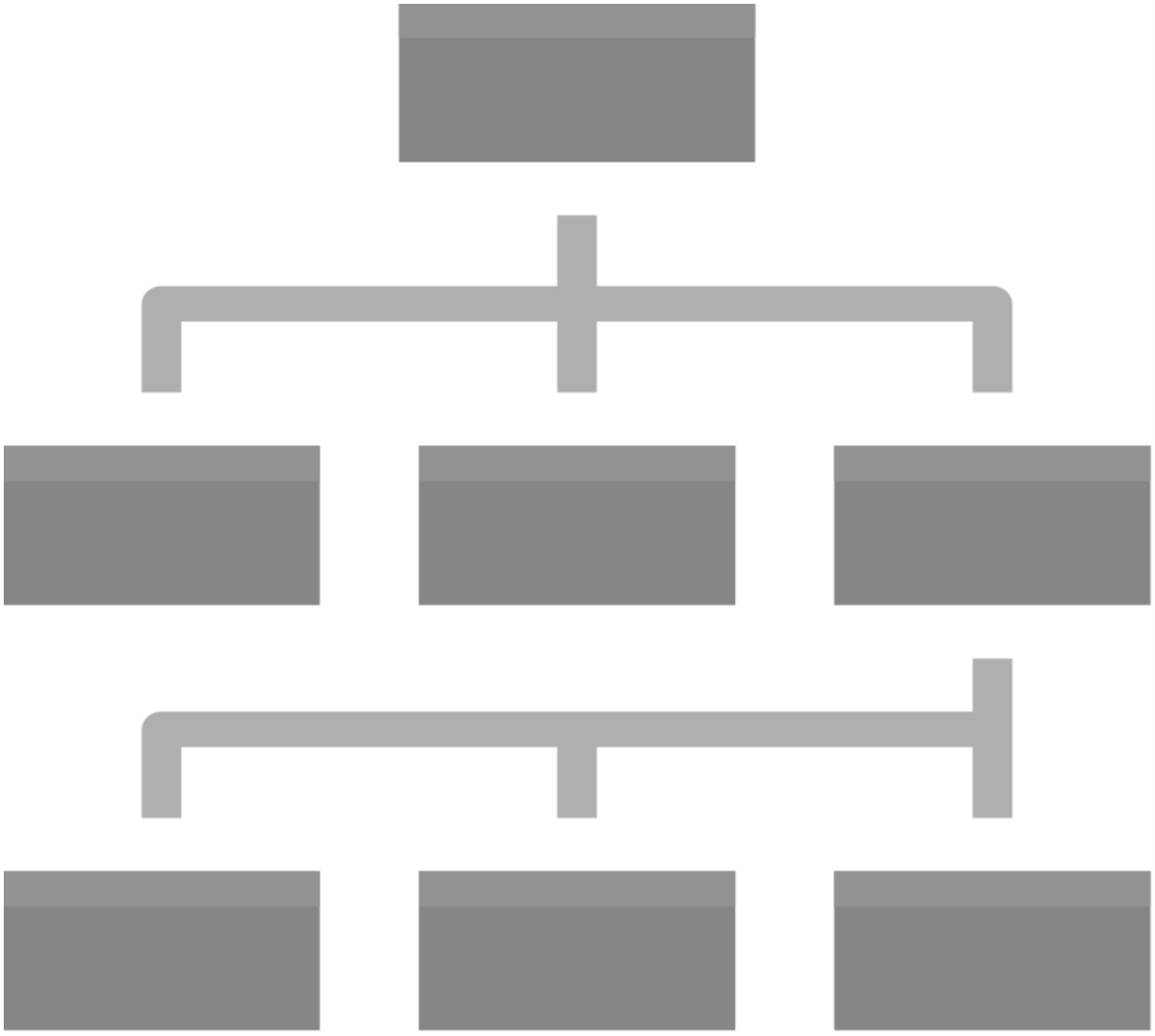
ACTIVITY 2.23

Lab Exercise

1. welcome_message = “Hello, welcome to Daisy Tech!”
print(welcome_message)
2. length_word = len(“Hello, welcome to Daisy Tech!”)
print(length_word)

SECTION THREE

DATA STRUCTURE



- Tuples
- Lists

- Dictionaries
- Sets

DATA STRUCTURES

There are four (4) collection data types in a python programming language also known as data structures:

1. Tuple
2. List
3. Dictionary
4. Set

TUPLES

Tuples are ordered sequence separated by commas enclosed in brackets or parenthesis ( ).

Example:

scores = (35, 40, 45, 50, 55, 60, 65, 70, 75, 80)

Tuples can contain Strings, Integers, floats

Example:

basket = ('fish', 30, 100.56)

basket

type(basket)

Tuples can be accessed by **indexing** the elements in the tuple.

Example

Fish	30	100.56
0	1	2

Basket[2] : ?

Negative indexing :

Fish	30	100.56
0	1	2
-3	-2	-1

Basket[-3] : ?

- Concatenate or join 2 or more tuples, use the + operator

Example

basket = ('fish' ,30, 100.56)

basket_add = basket + ('bread' , 8)
basket_add : ('fish' , 30, 100.56, 'bread' , 8)

0	1	2	3	4

• Slicing of Tuples
Example
Basket_add[0:2] : ('fish' , 30)

• Length of the Tuple
Example
len(('fish' ,30, 100.56, 'bread' , 8)) : 5 or **len(basket_add)** : 5

• Sorting Tuple
Example
alphabet : ('a' , 'b' , 'n', 'j' , 'e' , 't' , 'u' , 'o' , 'p' , 'a' , 'd')
alphabet_sorted = sorted(alphabet)
alphabet_sorted

• Nesting : A Tuple can contain other tuples which is called nesting.
Example
TG = (('fish' , 30,100.56), 'a' , 'd' , (1,6), ('apple', ('rice', 'dodo')))

0	1	2	3	4

TG[0:2] : (('fish' , 30, 100.56), 'a'))
TG[3] : (1, 6)[1] :6 -->**TG [3][1]** : 6

LISTS

List are ordered sequence. They are enclosed in square brackets [ ]. It can contain strings,integars,floats and we can get nest other lists.
Example
Z = ['Kenya', 4. 5, 8, 2000] **Z[0]** : “Kenya” or **Z[-3]** : 4.5

0	1	2	3
-4	-3	-2	-1

• Slicing a list

Example
Z[2:3] : 8 Note : It ignores the last number

• Concatenate lists by adding them.
Example
Z1 =Z + [67, 1999]
Z1 : ['Kenya' , 4, 5, 8, 2020, 67, 1999]

• Extend(): This is another way of adding elements in Python.
Example:
Z1 = ['Kenya', 4.5, 8, 2020, 67, 1999]
Z1.extend(['Nairobi', 234])
Z1 : ['Kenya', 4.5, 8, 2020, 67, 1999, 'Nairobi', 234]

Append(): It adds only one element to the list.
Example:
Z1 = ['Kenya', 4.5, 8, 2020, 67, 1999]
Z1.append([“Kano”, 18])
Z1: ['Kenya', 4.5, 8, 2020, 67, 1999, [“Kano”, 18]]
Z1.append(“Asaba”)
Z1: ['Kenya', 4.5, 8, 2020, 67, 1999, ”Asaba”]

Since lists are mutable, it can be changed.
Example:
M = ['Joy', 3,4,5,6.5]
M[1] = 28
M : ['Joy', 28,4,5,6.5]

• Del() is used to delete an element in the list
Example:
M = ['Joy', 3,4,5,6.5]
del(M[0])
M : [3,4,5,6.5]

• Split(): This method splits is used to convert a string into list, separating every group of characters separated by a space into an element a list.
Example:
“Eko oni baje!”.split()
Output : [“Eko”, “oni”, baje!”)

We can split string through a specific character called delimiter.
Example:
'1,2,3,4,5,6,7'.split(',') : ['1','2','3','4','5','6','7']

DICTIONARIES

It stores the mapping of unique keys to values.

List		Dictionary	
Index	**Element**	**Key**	**Value**
0	Element 1	Key1	Value1
1	Element 2	Key2	Value2
2	Element 3	Key3	Value3
…..	…..	……	……

- Dictionaries are denoted by curly brackets {}
- The keys are immutable and unique.
- Each key is followed by a value, and both are separated by a comma.
- The value can be immutable, mutable and duplicates.

Example:

movies = {“Living in Bondage”:1993, “Lionheart”:2018, “Chief Daddy”:2018, “Isoken”:2017, “Ije”:2010}

“Ije” in movies : True

- To add an element in a dictionary.

Example:

movies ['October 1”] =2014

movies : {“Living in Bondage”:1993, “Lionheart”:2018, “Chief Daddy”:2018, “Isoken”:2017, “Ije”:2010, 'October 1':2014}

- To add multiple elements in a dictionary use method .update()

Example:

movies: **{“Living in Bondage”:1993, “Lionheart”:2018, “Chief Daddy”:2018, “Isoken”:2017, “Ije”:2010, 'October 1':2014}**

movies.update({“Blood Sister”:2022, “The Witcher”:2021})

movies : {“Living in Bondage”:1993, “Lionheart”:2018, “Chief Daddy”:2018, “Isoken”:2017, “Ije”:2010, 'October 1':2014, “Blood Sister”:2022, “The Witcher”:2021}

Forming a table can help you visualize the dictionary.

Key	Value
Living in Bondage	1993
Lionheart	2018
Chief Daddy	2018
Isoken	2017
Ije	2010

- Get (): to get a Key from a Dictionary

Example:

movies = {“Living in Bondage”:1993, “Lionheart”:2018, “Chief Daddy”:2018, “Isoken”:2017, “Ije”:2010}

movies.get('Ije') : 2010

- Del (): to delete a key in dictionary.

Example:

del(movies[“Isoken”])

movies : {'Living in Bondage': 1993, 'Lionheart': 2018, 'Chief Daddy': 2018, 'Ije': 2010}

- Values (): to see all the values in the dictionary.

Example:

movies = {“Living in Bondage”:1993, “Lionheart”:2018, “Chief Daddy”:2018, “Isoken”:2017, “Ije”:2010}

movies.values() : dict_values([1993, 2018, 2018, 2017, 2010])

- Keys (): to see all the keys in the dictionary.

Example:

movies.keys() : dict_keys([“Living in Bondage”, “Lionheart”, “Chief Daddy”, “Isoken”, “Ije”)

SETS

Sets also can contain different Python types.

• They are unordered collections which means they do not record elements position.

• Sets have only unique elements.

• Set is defined using curly brackets. {}

Example:

set_new = {'rock star', 'afro', 'rock', 'blues', 'afro beat', 'afro beat', 'slow jamz', 'rock', 'slow'}

even as some words were repeated, the output will ignore the repeated words.

set_new: {'rock star', 'afro', 'rock', 'blues', 'afro beat', 'slow jamz', 'slow'}

CREATING SETS

• Set(): to create a set from a list, and a set from a dictionary with only the key.

Example:

afro_list = ['Fela', 'Femi', 'afro beat', 'Fela']

To convert the list to set -> afro_set = set(afro_list)

afro_set : {'Fela', 'Femi', 'afrobeat'}

movies = {'Living in Bondage':1993, 'Lionheart':2018, 'Chief Daddy':2018, 'Isoken':2017, 'Ije':2010}

set(movies)

• Add(): to add an element to a set.

Example:

afro_set.add('Burna boy')

afro_set : {'Fela', 'Femi', 'afrobeat', 'Burna boy'}

• Remove(): to remove an element from a set

Example:

afro_set = afro_set: {'Fela', 'Femi', 'afrobeat', 'Burna boy'}

afro_set.remove('Femi')

afro_set: {'Burna boy', 'Fela', 'afro beat'}

• To check if an element is in a set: 'Celine Dion' in afro_set: False

SET METHODS

• Intersection(&): When two sets have some values in common, the intersection operation & can be used to get the common values from the 2

sets.

Example:

afro_set = {'Fela', 'Femi', 'afrobeat', 'Burna boy'}

set_new = {'rock star', 'afro', 'rock', 'blues', 'afro beat', 'slow jamz', 'slow'}

new_intersection = afro_set & set_new

new_intersection : { 'afro beat'}

• Union(): This is the union of 2 sets which output contains all the elements in both sets.

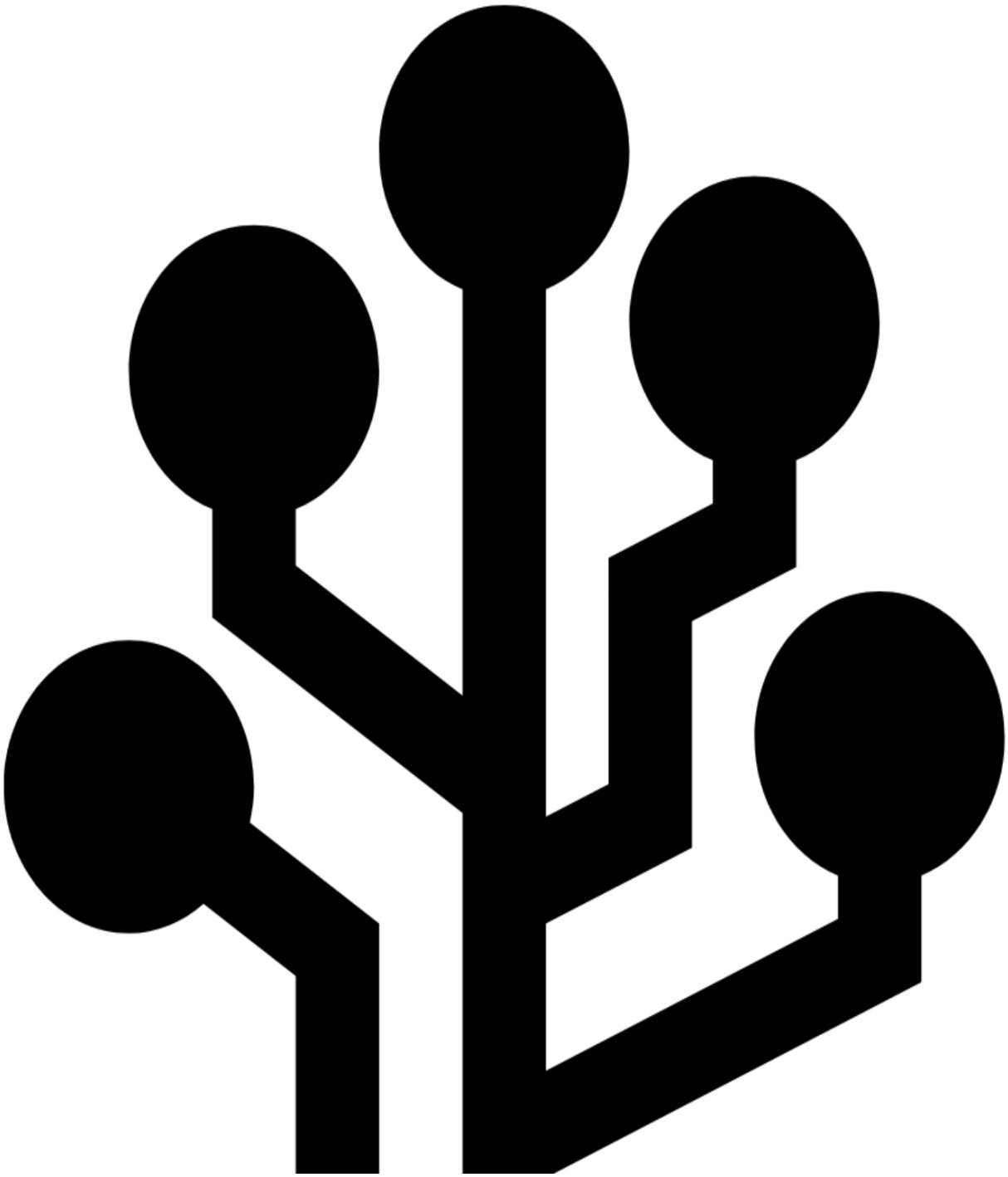
Example: **afro_set = {'Fela', 'Femi', 'afrobeat', 'Burna boy'}**

new_beat = {'blues', 'afro beat', 'slow jamz', 'slow'}

afro_set.union(new_beat) : {'Fela', 'Femi', 'afro beat', 'Fela', 'afro beat', 'blues', 'afro beat', 'slow jamz', 'slow'

SECTION FOUR

CONDITION AND BRANCHING



- **Conditions**
  - Assignment Operators
  - Conditions Examples

- **Branching**
- **Indentation**
    - Else Statement
    - Elif Statement
- **Logical Operations and Statement**
- **Loops**
    - For Loop
    - Create New List
    - Modify List
    - While Loop
    - Activity 4.1

CONDITIONS

Logical conditions are supported in Python. They are:

Comparison Operators	Symbols
Equals	a == b
Not Equals	a != b
Less than	a < b
Less than or equal to	a < = b
Greater than	a > b
Greater than or equal to	a > =b

The output using the comparison operator is a **boolean** which is either **True or False.**

ASSIGNMENT OPERATORS

Operator	Example	Equivalent
=	b=4	b= 4
+=	b+=4	b=b+4
-=	b-=4	b=b-4
*=	b*=4	b=b*4
\=	b\=4	b=b\4
%=	b%=4	b=b%4
\\=	b\\=4	b=b\\4
**=	b**=4	b=b**4
&=	b&=4	b=b&4
|=	b|=4	b=b|4
^=	b^=4	b=b^4
>>=	b>>=4	b=b>>4
<<=	b<<=3	b=b<<4

CONDITIONS EXAMPLES

Example:

m = 3 **i = 100** **f = 22**

m == 8 **i > 200** **f != 21**

Output: False False True

'Ada' == 'Obi' **'Ada' != 'Obi'**
Output: False True

BRANCHING

Branching allows us to run different statements for different inputs.

If Statement: it is a logical statement written by using the 'if' keyword.

If a is… do something, otherwise do something else.

Example:

```
  x= 20
  y= 90
if x !=y :
    print(“Learning Python Language is cool”)
print('not again') -> this code will print irrespective if True/False.
```

INDENTATION

Before we continue, we need to understand indentation as it will be used to write conditional statements.

Indentation is used in Python to identify what code is in the body of a conditional statement and what code is outside the body of the statement. However, some other languages use braces… {….} to show where a block of code begins and ends. But in Python, indentations are used.

Note: Indentation comes in the multiple of 4 spaces. Following the indentation, the convention is crucial as changing it can change the meaning of the code.

ELSE STATEMENT

Example:

```
age = 16
if (age>=18):
    print('Drink responsibly!')
else:
    print('Children do not drink')
print('Next')
```

ELIF STATEMENT

The elif statement a short form of else if allows us to check for additional conditions if the proceeding condition is false.

```
age = 16
if (age>=18):
    print('Drink responsibly!')
elif(age==17):
    print('Hang on, your age is near to drink')
else:
    print('Children do not drink')
print('Next')
```

LOGICAL OPERATORS

Operator	Description	Example
And	Returns True if both statements are true	b<4 and b<3
Or	Returns True if one of the statements is true	b<4 or b<3
Not	Reverse the result. Returns False if the result is true.	not(b<4 and b<3)

LOGICAL STATEMENTS : USING OR

```
  afro_beat = 1960
if (afro_beat<1970) or (afro_beat>1980):
            print("This is old school")
else:
           print("This album is latest release")
```

LOGICAL STATEMENTS: USING AND

```
  afro_beat = 1960
if (afro_beat<1970) and (afro_beat>1980):
            print("This is old school")
else:
```

print(“This album is latest release”)

LOOPS

Loops allow us to repeat blocks of code. There are two kinds of Loop names;

1. For Loop

2. While Loop

For Loop: This is used to iterate over an iterable.

Iterable: An iterable is an object that can return one element at a time. Such as a sequence type e.g. List, tuple, string or non-sequence types such as set, dictionary and files.

Example :

```
cities = ['lagos', 'abuja', 'kano', 'enugu', 'aba']
for city in cities:
        print(city.upper())
```

Output: LAGOS, ABUJA, KANO, ENUGU, ABA

FOR LOOP

In the earlier example,

Cities are the iterable

The city here is the loop iteration variable; this means the variable that represents the element in the iterable.

The indented print action is repeated for each element in cities. You can name the iteration variable whatever you like. However, a pattern like an example above is best practice.

For loops can be used to create lists and modify lists.

CREATE A NEW LIST

Append() method can be used to create a new list.

Example:

```
cities = ['lagos', 'abuja', 'kano', 'enugu', 'aba']
new_cities = [ ]
for city in cities:
        new_cities.append(city.title())
new_cities
```

MODIFY A LIST

To modify a list, a new function is needed -> Range.

Rang(start, stop, step): It is used to create immutable sequences of numbers with 3 arguments which must all be integers.

Start – beginning number. Default is 0

Stop – second to the last number

Step – the difference between the numbers in the sequence. Default is 1

Example:

print(list(range(10)))

Output: [0,1,2,3,4,5,6,7,8,9]

A range with 1 integer value, value will be a stop argument. Since the example has one value, 10 is the stop argument. It returns 10-1 = 9.

FOR LOOP: USING RANGE

A range with 2 integer value, values will be a start and stop the argument.

Example:

```
 for i in range(3,8):
        print (i)
```

A range with 3 integer values, values will be a start, stop and step argument.

Example:

```
for i in range(1,20,2):
        print (i)
```

FOR LOOP: USING ENUMERATE

Enumerate (): used to obtain the index and the element in a Data collection type.

Example:

```
colors = ['blue', 'pink', 'grey', 'brown']
for m, color in enumerate(colors):
            print(m,color)
```

WHILE LOOP

A while loop repeats executions when certain stated conditions are met.

Example:

```
m= 3
while m !=10:
        print(m)
        m +=1
```

ACTIVITY 4.1

Lab Exercise

1.
```
colors = ['blue', 'pink', 'grey', 'brown']
    x=len(colors)
for m in range(x):
        print(colors[m])
```

2.
```
for m in range(0,20):
        print(m)
```

3.
```
for m in ['A','B','C','D','E']:
        print(m+'D')
```

4. What is the difference between for and while loop?

SECTION FIVE

READING AND WRITING FILES WITH OPEN

- **Reading file**
- **Data.Txt**

- **Using a Loop Print Out Each Line**
- **Creating and Writing file**
- **Writing File with Open**

READING FILES WITH OPEN

Open(): This is Python built-in function used to get a file object.

Example:

File1 = open(“C:/users/obi/Desktop/data.txt”, “w”)

Open -> open function

(“C:/user/obi/Desktop/data.txt”, “w”) –> File path

Directory file name mode

Modes are;

r = reading x = create

w = write

a = appending

DATA.TXT

Write the following sentence in the Data.TXT file you created on your desktop

Daisy Tech is a training Company.

We train on analytical skills.

We build people to acquire skills not just certificates.

File1.name -> get the name of the file

File1.mode -> mode of the file

File1.close() -> close the file

Example: Here we use with the statement because it closes the file automatically.

print(File1.closed) -> check if the file is closed

```
with open (“C:/user/obi/Desktop/data.txt”,”r”) as File1:
        training_file = File1.read() -> stores the file as a string
        print(training_file)
```

\n -> new line

Example:

```
with open (“C:/user/obi/Desktop/data.txt”,”r”) as File1:
        training_file = File1.readline() -> store file under training_file
        print(training_file)
        training_file = File1.readline()
        print(training_file)
```

USING A LOOP TO PRINT OUT EACH LINE

Example:

```
with open(“C:/user/obi/Desktop/data.txt”, “r”) as File1:
        for line in File1:
            print(line)
```

To print specified numbers of lines, we do the following;

```
 with open('data.txt','r') as File1:
        training_file = File1.readline(5)
        print(training_file)
        training_file = File1.readline(8)
        print(training_file)
```

CREATING AND WRITING FILE

To create a file, we use 'x' or 'w'. 'x' will return error if the file does exist, but 'w' will create a file if the file does not exist. Open() function is also used to create, write or append a file. However, write(w) will overwrite existing text.

Example:

Creating a file with 'x’

```
F1 = open('C:/user/obi/Desktop/valentine.txt', 'x')
```

Creating a file with 'w’

```
F1 = open('“C:/user/obi/Desktop/my_file.txt', 'w')
F1.write(“I am writing my first content in Python')
F1.close()
```

WRITING FILES WITH OPEN

Using append('a')

```
F1 = open('“C:/user/obi/Desktop/my_file.txt', 'a')
F1.write(“….writing more content”)
F1.close()
```

Read the file

```
F11 = open('C:/user/obi/Desktop/my_file.txt','r')
```

print(F1.read())

For additional resources, please visit the W3 School website:
https:\\www.w3schools.com\python\default.asp

You can access the video version of this book, Basic Python for Data Science (Volume One) using this link: http:\\daisytechlimited .org\courses\basic-python-for-data-science

A 50% Discount Coupon for the course is included below for your use.

Basic Python for Data Science was birthed as a result of the author's struggle with coding while studying Data Analytics with Python. In this book, she demystified coding in general and Python in particular, so that anyone aspiring towards a career in Data Science but having zero coding knowledge can find the journey not only easy but also interesting. The uniqueness of this book is how relatable it is; she infused African content into the various sections. For instance, you could see "Eko o ni baje", or other familiar catchphrases in some of the codes, or popular song lyrics in some of the contents thereby creating an enjoyable learning experience.

50% coupon code to access the video version of the book inside.
Learn more here about the online video here: www.daisytechlimited.org.

Chinelo Okafor is a COREN-registered Electrical and Electronics Engineer. She holds a B.Sc in Electronics/ Telecommunication Engineering from Nnamdi Azikiwe University, Awka (UNIZIK) and a Master's from the University of Lagos, Akoka (UNILAG). She also did a Business Management program with Lagos Business School (LBS) sponsored by the Bank of Industry (BOI).

She is a certified Data Analyst with over ten (10) years experience. She has worked as a Solar Analyst and Performance Engineer in a telecommunications company, trained over 200 professionals and graduates in Data Science and over 100 children in ICT.

She founded Daisy Tech Limited, a Data Science training company that empowers youths with Data Science skills to enable them to be employable or entrepreneurs. She also offers ICT training to children to help prepare them for the digital world.

Winepress
Educational
www.noirledge.com

Available as an eBook

ISBN: 978-978-60211-1-9

9 789786 021119