Introduction To Python

Bonface Onyango

Outline: Session I

- What is Python
- Why Python
- Python Interpreter
- Applications of Python in Data Science
- Installation of Python
- Writing your first python code
- Python as a Calculator
- Overview of Jupyter Notebook
- Running Python Code

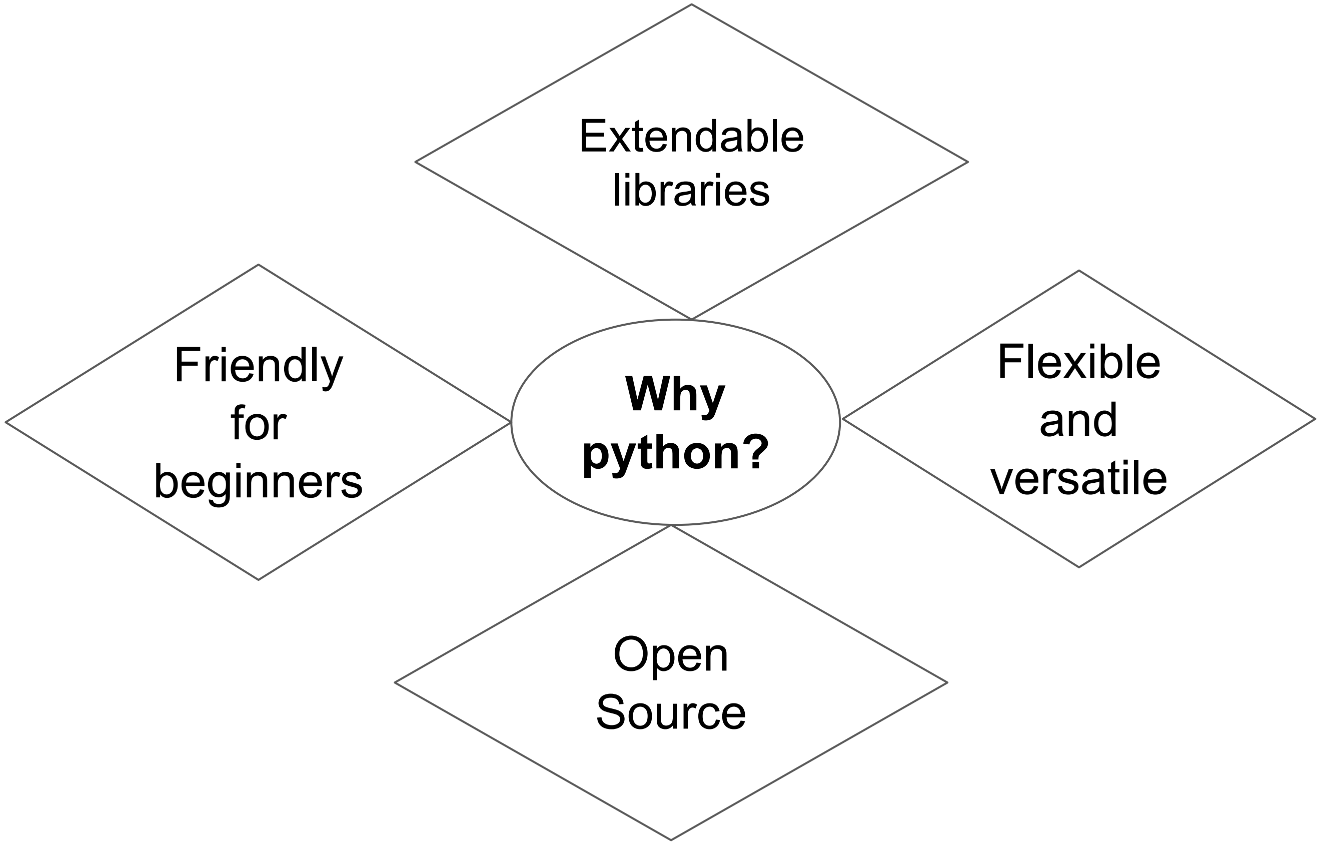
Mode of Lesson Delivery

- Theory lectures and video recordings
- Live coding
- Practicals

What is python

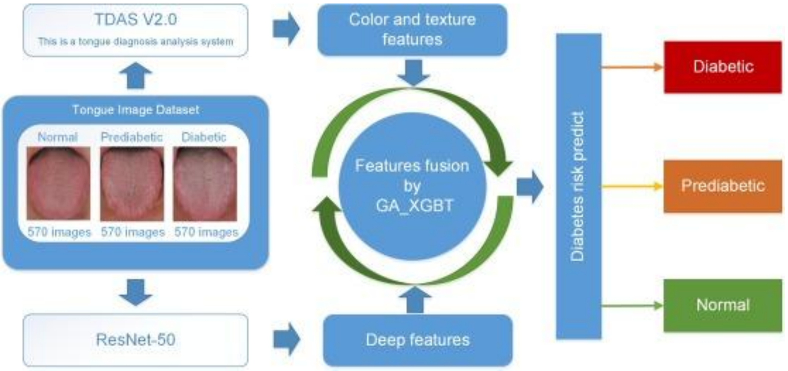
- Python is a general-purpose, high-level, interpreted, object-oriented programming language.

Why python?

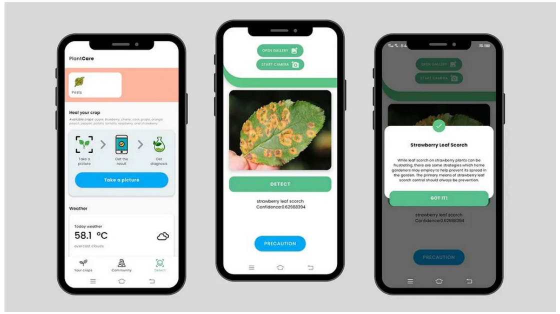

Extendable libraries
Friendly for beginners
Why python?
Flexible and versatile
Open Source

Applications of python in data science

Healthcare

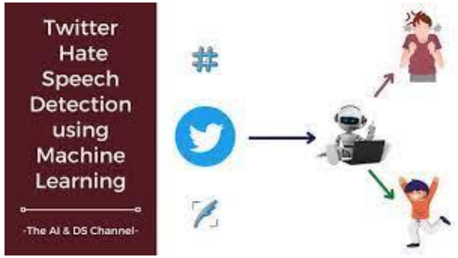


Agriculture

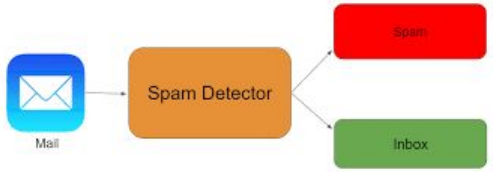


Gaming

Politics and Governance

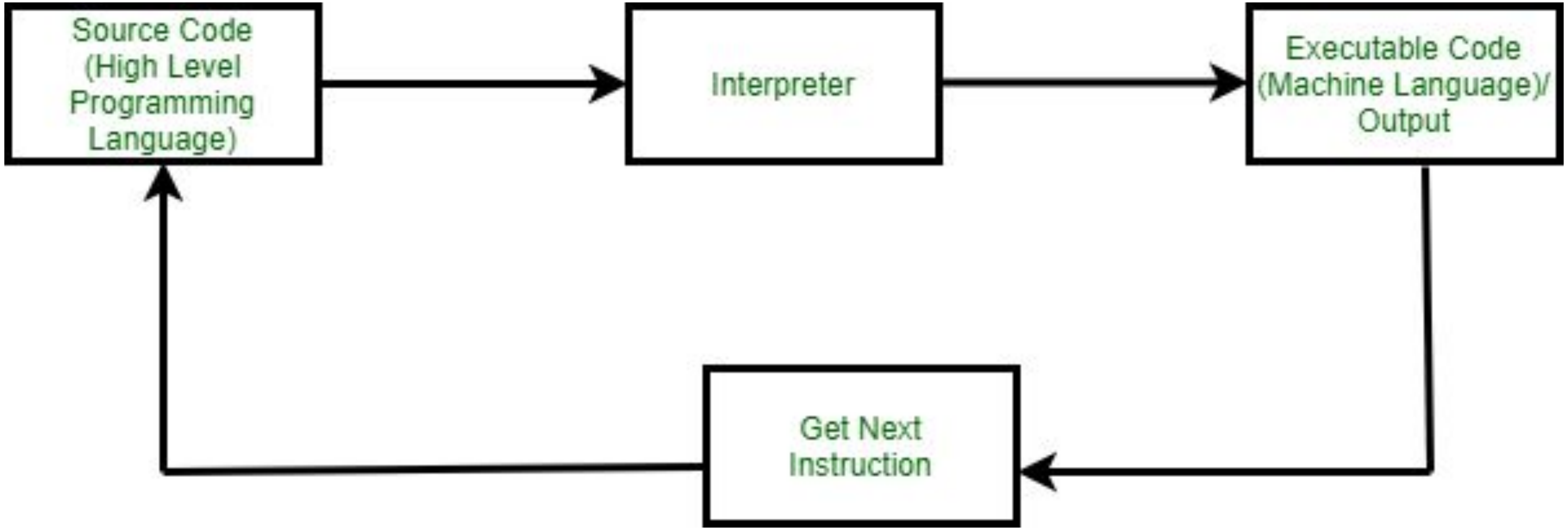


Email Classification



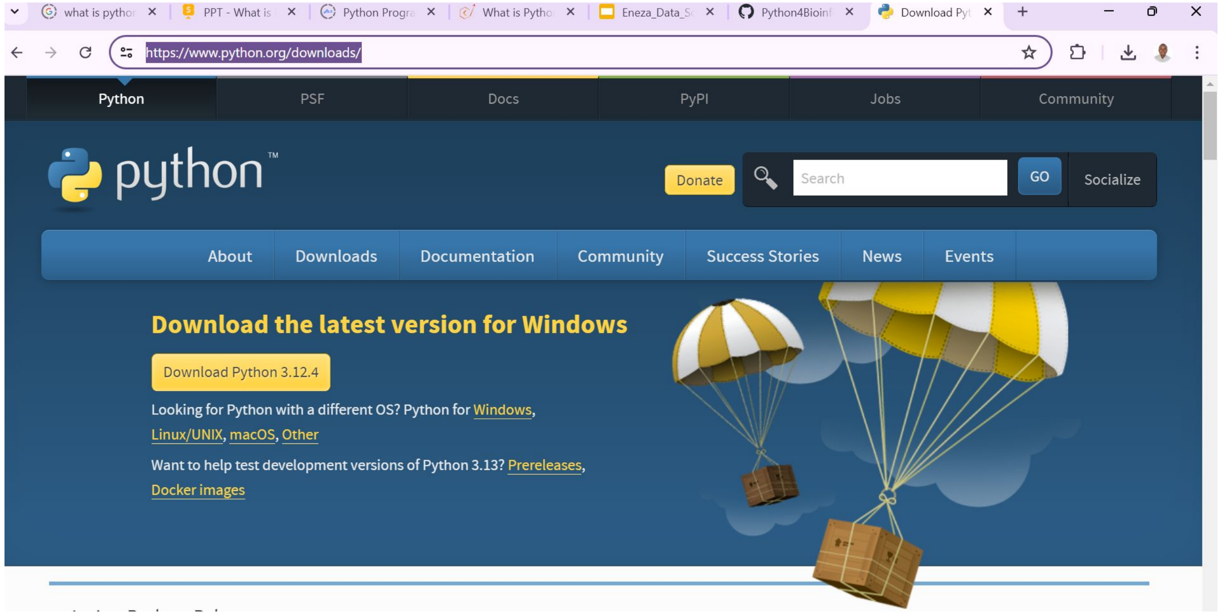
Python interpreter

- Python is an interpreted language because it executes line-by-line instructions
- Bridge the gap between what a human programmer writes and what the machine understands and thus executes


Source Code (High Level Programming Language)
Interpreter
Executable Code (Machine Language)/ Output
Get Next Instruction

Installing python

>> https://www.python.org/downloads/



Download the latest version for Windows

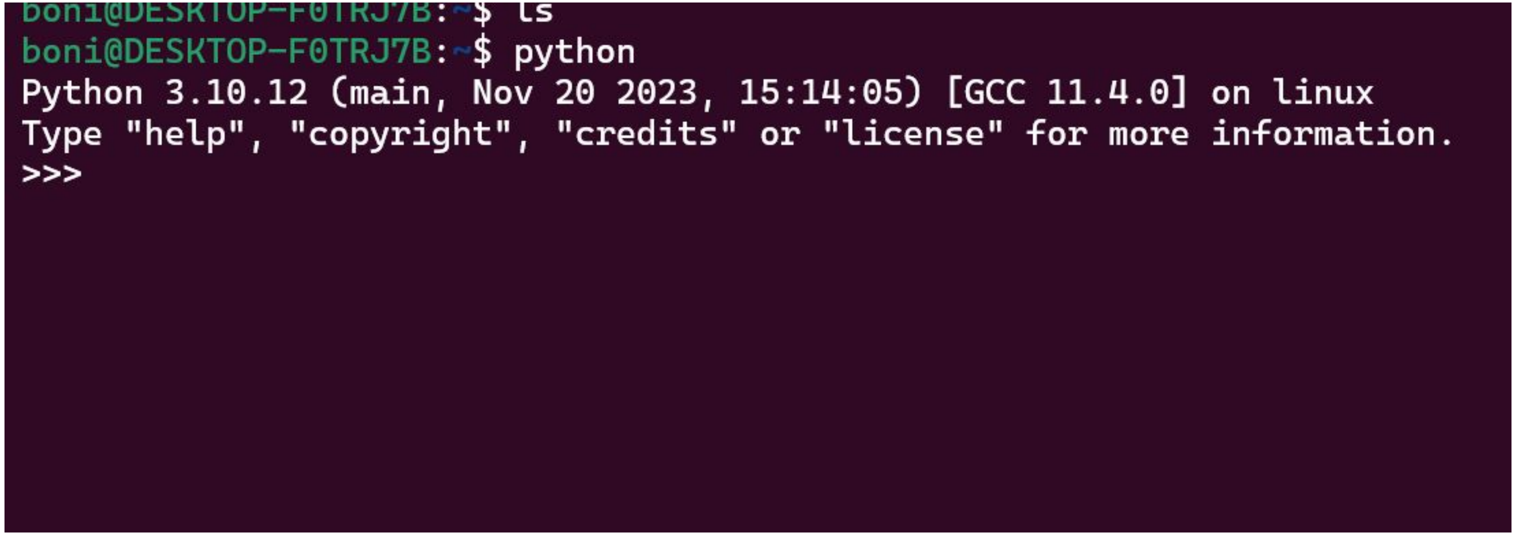
Download Python 3.12.4

Looking for Python with a different OS? Python for Windows, Linux/UNIX, macOS, Other

Want to help test development versions of Python 3.13? Prereleases, Docker images

Checking whether python has been installed

>>> python
>>> python 3

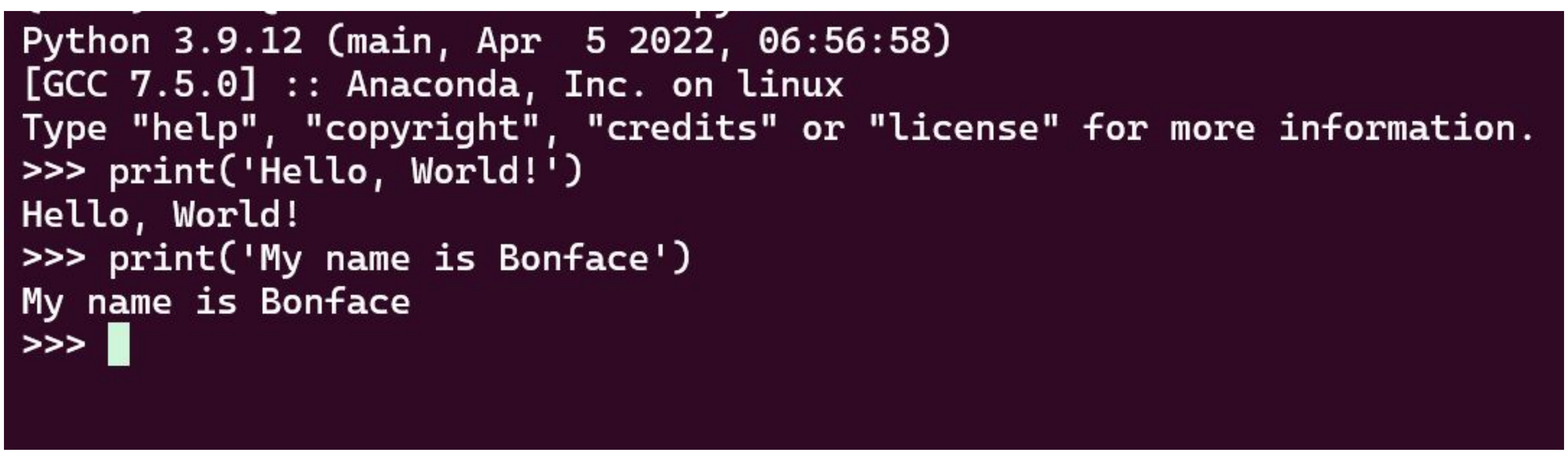


```
boni@DESKTOP-F0TRJ7B:~$ ls
boni@DESKTOP-F0TRJ7B:~$ python
Python 3.10.12 (main, Nov 20 2023, 15:14:05) [GCC 11.4.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

NOTE: ‘python’ should be typed in small letters!
Can you tell the python version?

Your first python code

>>> print(‘Hello world’)



```
Python 3.9.12 (main, Apr  5 2022, 06:56:58)
[GCC 7.5.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hello, World!')
Hello, World!
>>> print('My name is Bonface')
My name is Bonface
>>> 
```

Basic Python syntax >>>

Syntax is a Set of rules for a particular programming language

- Indentation of code block

```
Color=[“red”, ”blue”, “ green” ]
for color in colors:
   print(color)
```

- Variables: no spaces, case sensitive

```
max_num=90
```

- Strings enclosed with double or single quotation marks

```
Var1=‘Hello world”
var2  ‘Hello world’
```

- The new line symbol is ‘\’

```
add=20+50+\
            60+70
```

- Comments start with a #

```
# print my name
print(“my name”)
```

Other Basic Rules

- **Python is Case sensitive**

Example:
input:
```
PRINT(‘hello world!’)
```
Output:
```
NameError: name 'PRINT' is not defined
```

- **Indexing in python starts from 0: ATGC**

Example
```
my_dna=[‘ATGC’]
My_dna[0]
```
Output: A

Using Python as a calculator

Multiply

```
>>> 3*3
9
```

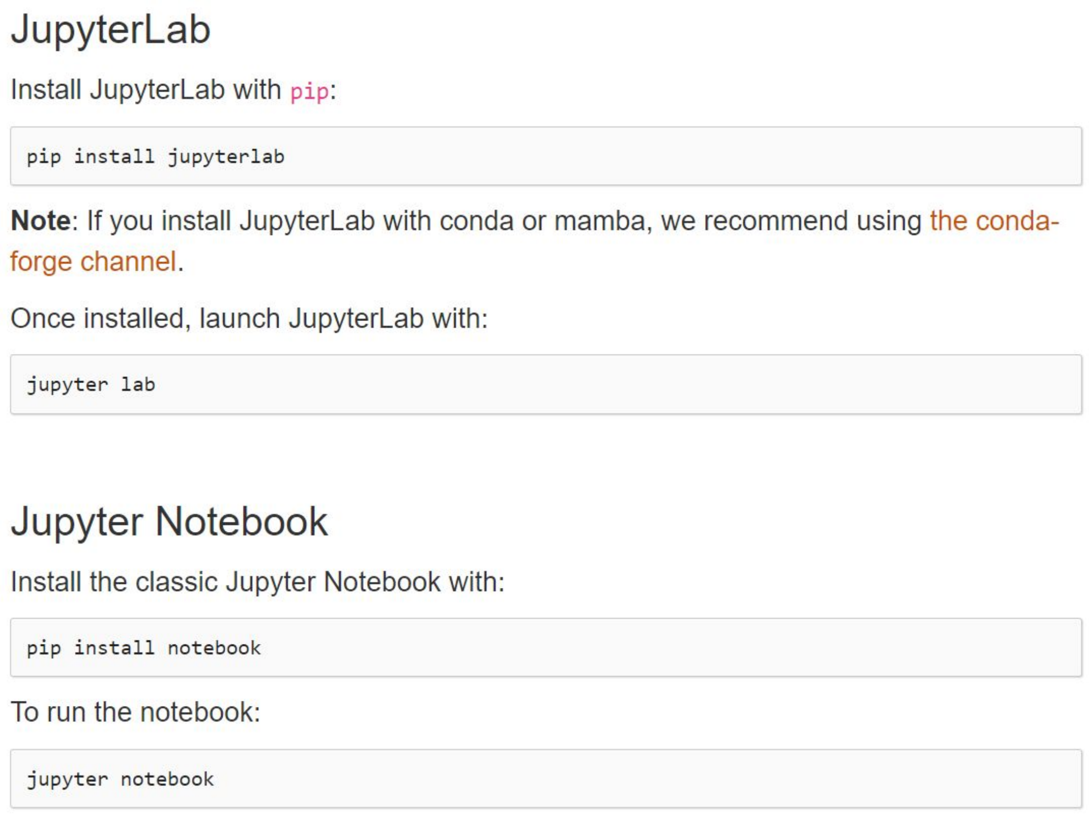
Add

```
>>> 3+3
6
```

Divide

```
>>> 4/2
2
```

Installation of Jupyter Notebook



JupyterLab

Install JupyterLab with `pip`:

```
pip install jupyterlab
```

Note: If you install JupyterLab with conda or mamba, we recommend using the conda-forge channel.

Once installed, launch JupyterLab with:

```
jupyter lab
```

Jupyter Notebook

Install the classic Jupyter Notebook with:

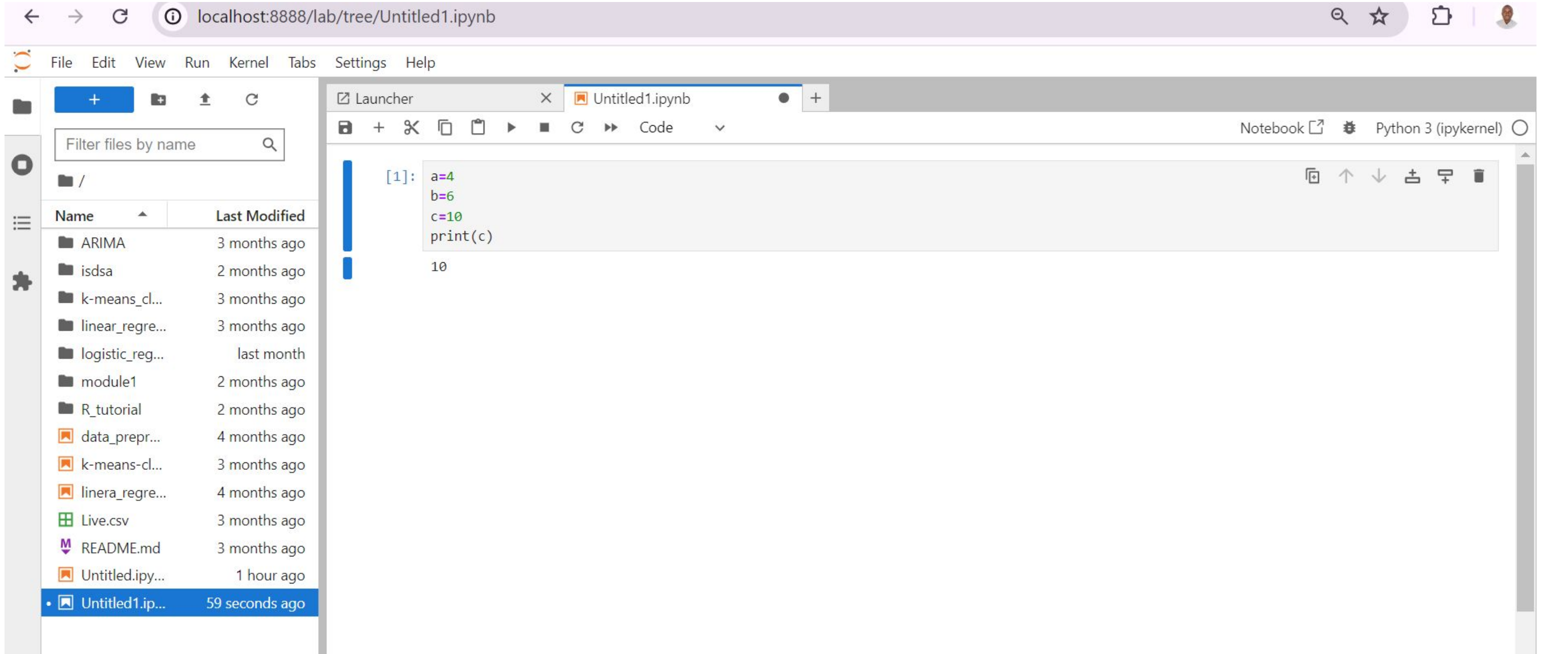
```
pip install notebook
```

To run the notebook:

```
jupyter notebook
```

Jupyter Notebook

- Jupyter Notebook is used to create interactive notebook documents that can contain live code, equations, visualizations, media and other computational output



Name	Last Modified
ARIMA	3 months ago
isdsa	2 months ago
k-means_cl...	3 months ago
linear_regre...	3 months ago
logistic_reg...	last month
module1	2 months ago
R_tutorial	2 months ago
data_prepr...	4 months ago
k-means-cl...	3 months ago
linera_regre...	4 months ago
Live.csv	3 months ago
README.md	3 months ago
Untitled.ipy...	1 hour ago
Untitled1.ip...	59 seconds ago

```
[1]: a=4
     b=6
     c=10
     print(c)

     10
```

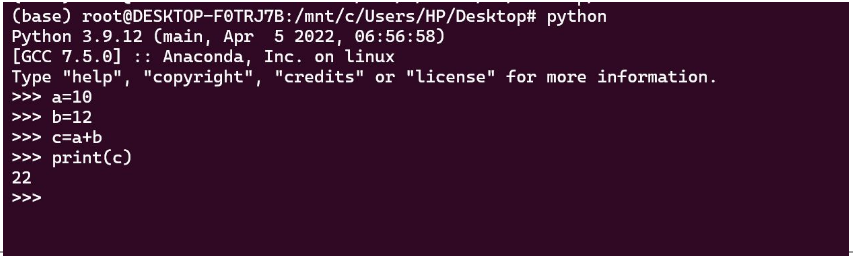
Installing Jupyter Notebook using Miniconda

- It is recommended to install Jupyter Notebook through Anaconda or as an add-on to Miniconda.

- While it's possible to install Jupyter Notebook on its own, using Anaconda or Miniconda simplifies the process.

Running Python Code

Interactive mode



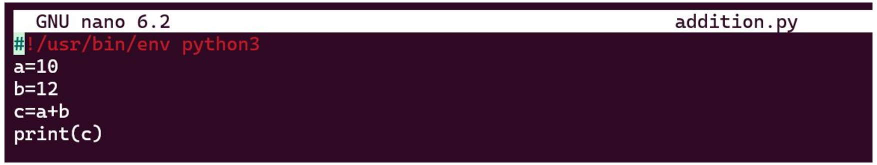
```
(base) root@DESKTOP-F0TRJ7B:/mnt/c/Users/HP/Desktop# python
Python 3.9.12 (main, Apr  5 2022, 06:56:58)
[GCC 7.5.0] :: Anaconda, Inc. on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> a=10
>>> b=12
>>> c=a+b
>>> print(c)
22
>>>
```

Above python program:

- takes two inputs as **a** and **b**
- Prints the sum in the third variable which is **c**.
- It follows sequential as well as functional execution of programs

To escape python console: Control + D

Script mode/Python prompts



```
  GNU nano 6.2                                                    addition.py
#!/usr/bin/env python3
a=10
b=12
c=a+b
print(c)
```

- Create a python file e.g addition.py
- Write each line of the python code using any editor (nano,Vscode..etc)
- Cd to the file directory

Run code below?

Python addition.py

Output: 22

Live coding

- Python console
- Check if python is installed
- Write your first code
- Use python as a calculator

Let’s Recap

- Python is a high-level, interpreted programming language known for its easy-to-read syntax, dynamic typing, and versatility
- Indentation is used to define code blocks instead of braces or keywords.
- Python is case-sensitive, and variables do not require explicit declaration.
- Jupyter notebook combines code, narrative text,visualizations, and other rich media into a single document

Time for Break !

SEE YOU IN THE NEXT SESSION!

Introduction to Python: Session II

Mode of Lesson Delivery

- Theory lectures and video recordings
- Live coding
- Practicals

Outline: Session II

- What is Algorithm in python
- Assigning variables
- Python operators
- Basic data types and operations
- Data structures
- Control statements
- For loop

Algorithms in python

- Are set of instructions for solving problems programmatically

Real life example (cooking ugali) Steps to follow	Code problem example (Find mean of numbers)
Ingredients: 2 cups of maize flour 4 cups of boiling water	numbers=[1,2,3,4] Input: numbers
1. Buy unga 2. Bring 4 cups of water to a boil in a kettle or other pan 3. Pour Unga after the water have boiled 4. Stir briskly with a cooking stick 5. Cook to a firm texture 6. Turn the cooked ugali onto a serving plate	1.Get the length of the numbers 2. Get the sum of the numbers 3. Divide sum by length

Assigning variables

```
a=10
b=20
c=a+b
print(c)
```

A variable names a value that we want to use later in a program.

Note:

- The symbol = is pronounced “gets” not “equals”!
- Avoid using python **keywords** in naming variables e.g list=[1,2,3]

Assigning variables…

- Python allows you to assign values to multiple variables in one line:

```
>>>a,b,c=10,20,30

>>>print(a)

>>>print(b)

>>>print(c)
```

Data Types

Python data types

- Floating point numbers
  float
- String
  str
  my_dna='ATGCT
  in , **not in** , **+** , and *****
- Integers
  int
  +,-,*,/,//
- boleans
  bool
  and,or not

Key:
Black:data type
green : python representation
red:operators

Python operators

Arithmetic python operators

Addition **(+)**
```
>> 7+2
9
```

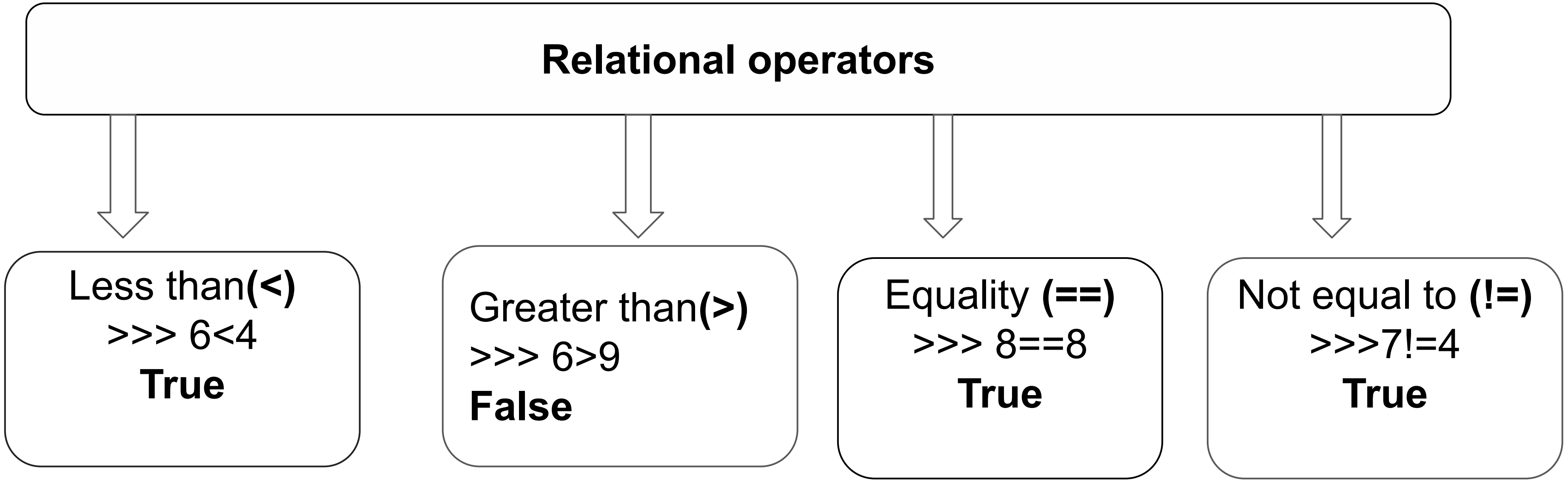
Multiplication **(*)**
```
>>> 7*2
14
```

Division**(/)**
```
>>> 7/2
3.5
```

Floor division**(//)**
```
7//2
3
```

Can you check what this operator (**) do in python? E.g 33=?**

Python operators



Relational operators

Less than**(<)**

```
>>> 6<4
True
```

Greater than**(>)**

```
>>> 6>9
False
```

Equality **(==)**

```
>>> 8==8
True
```

Not equal to **(!=)**

```
>>>7!=4
True
```

You check what these operators (**) do in python: **<= and >=**

Strings

What to know about strings in python

- Sequence of characters
- index starts from 0
- Case sensitive
- Immutable

Example

```
My_fast_string=‘Hello word’
```

String manipulation

- Indexing
- Negative Indexing
- Splicing

my_dna=’TGCGTAGC’

String Built In Functions

Use case: my_protein=“ALTPPPTA”

- len()
- str()
- split()
- strip()
- reverse
- startswith()
- endswith()

Why know data types

- Know which operators to use

Data Structures in Python

[Lists]

- Group of items
- Different data types
- Mutable: items can be added or removed
- Index starts from 0

Example

diseases=['cancer', covid19,diabetes]

{Dictionaries}

- consists of a key and then an associated value
- Heterogeneous objects
- Mutable

Example

my_dict={UUU:'Phe',AGU: 'Ser}

(Tuples)

- Allow duplicate values
- Indexed and starts from 0

Example

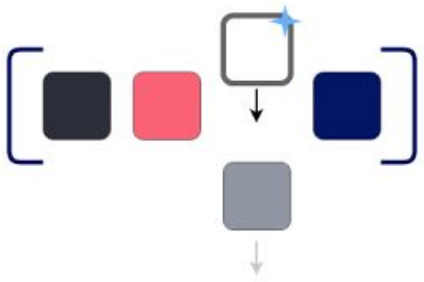
locus=(gene A,2000,7)

Why know data structures

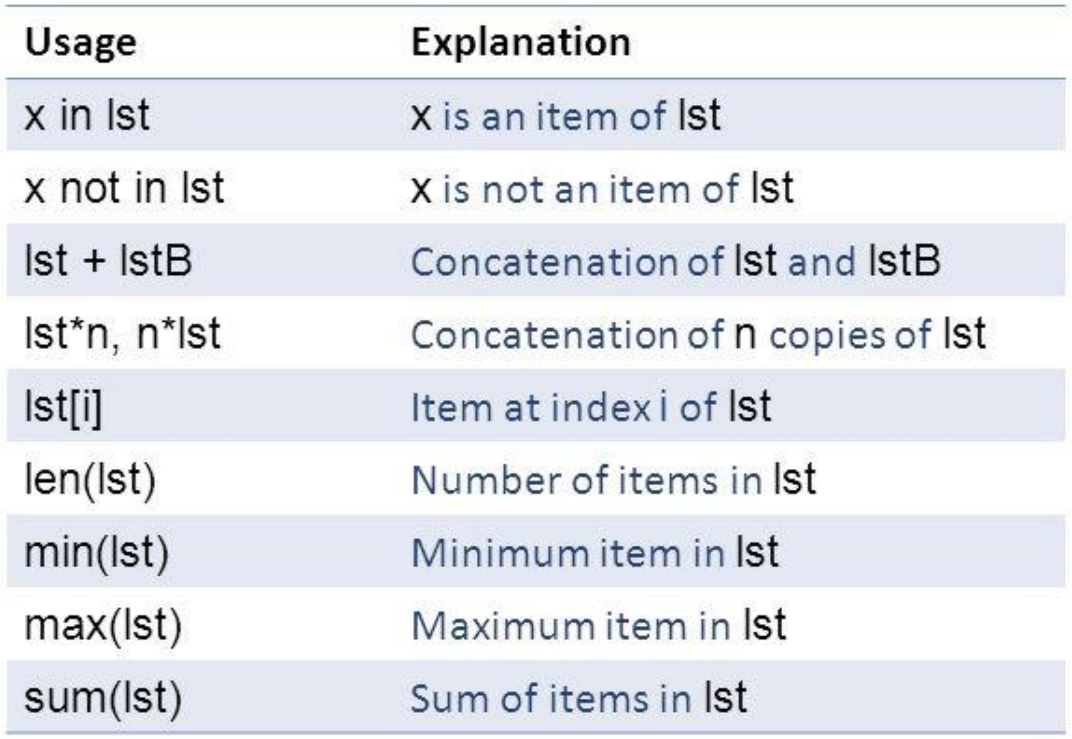
- Acts as a guide on data **storage**,and **manipulation**

[Python Lists]

- Sequence of objects
- Comma used to separate list items



List Operators



Usage	Explanation
x in lst	x is an item of lst
x not in lst	x is not an item of lst
lst + lstB	Concatenation of lst and lstB
lst*n, n*lst	Concatenation of n copies of lst
lst[i]	Item at index i of lst
len(lst)	Number of items in lst
min(lst)	Minimum item in lst
max(lst)	Maximum item in lst
sum(lst)	Sum of items in lst

List Build In Functions

- len()
- max()
- min()
- reverse()
- insert()
- index()

Use case my_list= [1,2,3,’a’,’b’,c,7]

{Python Dictionaries}

- General syntax: {key:value}
- Creating dictionary in python
  my_dict={}

- **Build in Functions for Python dictionary**
  -get()
  - keys()

Loops

For Loop

- Iterate over items in a list, dictionary,tuple
- Used if we know how many iterations must occur

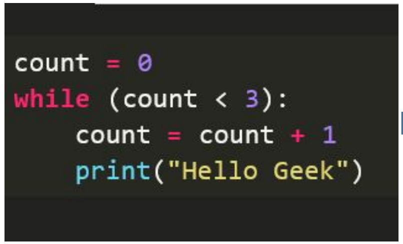
```
List=[“Orange” ,”Pink’ ,’Red” ]
for color in list:
   Print(color)
```

output

```
Orange
Pink
Red
```

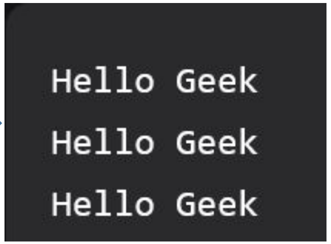
While Loop

- Loops through a series of code until a specific condition is met
- Used if we do not know how many iterations must occur



```
count = 0
while (count < 3):
    count = count + 1
    print("Hello Geek")
```

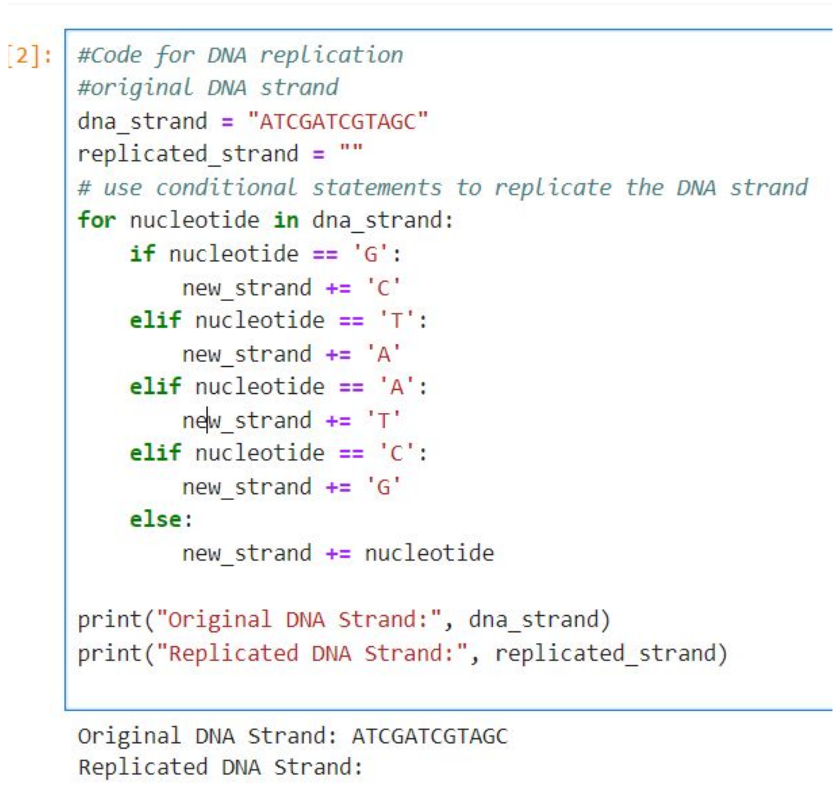
output



```
Hello Geek
Hello Geek
Hello Geek
```

Conditional Statements

if,else, el if:
Example use case: Simulation of how DNA makes a copy of itself



```
[2]: #Code for DNA replication
     #original DNA strand
     dna_strand = "ATCGATCGTAGC"
     replicated_strand = ""
     # use conditional statements to replicate the DNA strand
     for nucleotide in dna_strand:
         if nucleotide == 'G':
             new_strand += 'C'
         elif nucleotide == 'T':
             new_strand += 'A'
         elif nucleotide == 'A':
             new_strand += 'T'
         elif nucleotide == 'C':
             new_strand += 'G'
         else:
             new_strand += nucleotide

     print("Original DNA Strand:", dna_strand)
     print("Replicated DNA Strand:", replicated_strand)
```

```
Original DNA Strand: ATCGATCGTAGC
Replicated DNA Strand:
```

Practice Question

Practice Exercise

- Using strings, lists, and dictionaries concepts, find the DNA copy (replicate) of:

  GTGGGATGGTTTGGGTTGGGCGTG

Practicals

- **Fork** and **clone** the github repository below
  https://github.com/kipkurui/Python4BioinformaticsV2

Contacts

Email:bonfaceb21@gmail.com

GitHub: https://github.com/bonfaceonyango

LinkedIn:https://www.linkedin.com/in/bonface-onyango-927145161

Thank you!

END