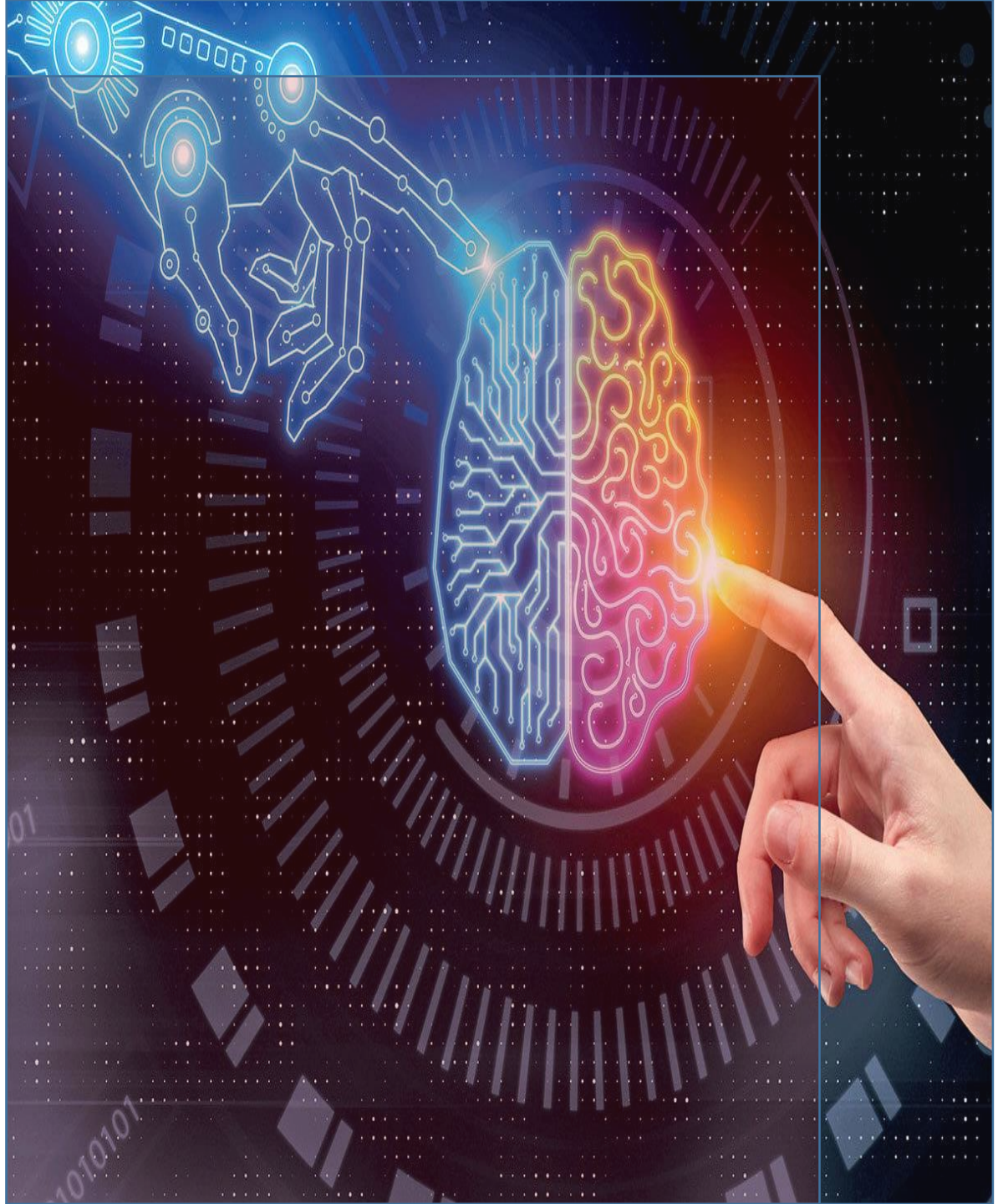


PYTHON

(Lab)

Engr. Tajummal Hussain
engr.tajummalhussain@gmail.com

Python & Environment Setup

Why python

- Simple syntax & less coding
- Inbuilt libraries for AI projects
- Open source
- Can be used for broad range of programming

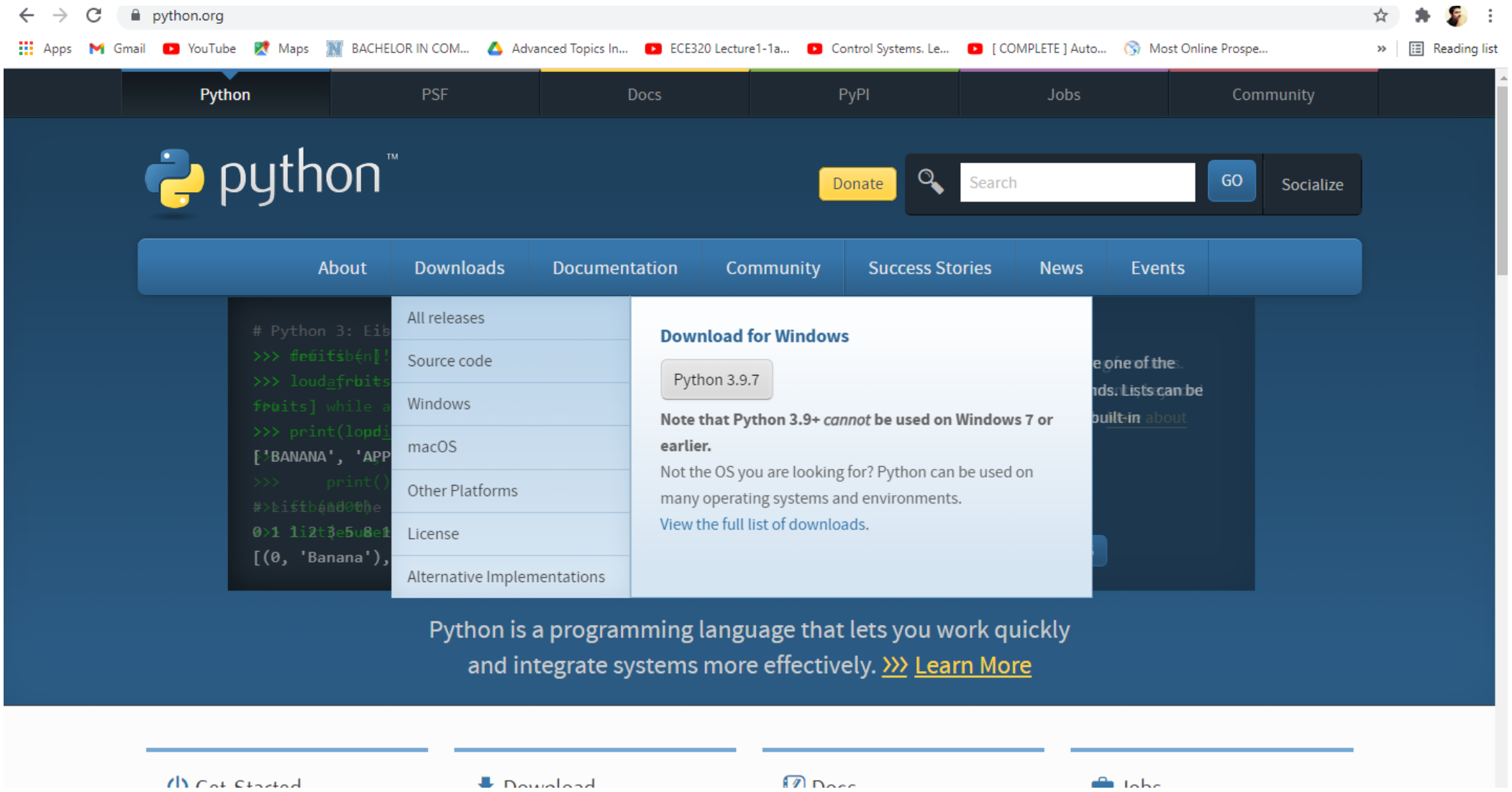
Features of Python

- Python is a high-level, interpreted, interactive and object-oriented scripting language
- **Easy-to-learn** , **Easy-to-read** , **Easy-to-maintain**
- **A broad standard library**
- **Interactive Mode**
- **Portable**
- **Extendable**
- **Databases**
- **GUI Programming**

Setting Environment

(Download & Installation)

https://www.python.org/



IDLE

SPYDER

PyCharm

eric

Vim

Atom

jupyter

Anaconda

PyDev

Thonny
Python IDE for beginners

whats app web - Google Search | Anaconda | Individual Edition

anaconda.com/products/individual-d

Apps | Gmail | YouTube | Maps | BACHELOR IN COM... | Advanced Topics In... | ECE320 Lecture1-1a... | Control Systems. Le... | [COMPLETE] Auto... | Most Online Prospe... | Reading list

ANACONDA.

Products | Pricing | Solutions | Resources | Blog | Company | Get Started

Individual Edition
Open Source Distribution

Commercial Edition
Premium Package Repository

Team Edition
On-prem Package Repository

Enterprise Edition
Full Data Science Platform

Professional Services
Data Experts Work Together

Individual Edit

Your c
toolkit

With over 25 millio ce Individual Edition (Distribution) is the easiest way to perform Python/R data science and machine learning on a single machine. Developed for solo practitioners, it is the toolkit that equips you to work with thousands of open-source packages and libraries.

Anaconda Individual Edition

Download

For Windows

Python 3.8 • 64-Bit Graphical Installer • 477 MB

Get Additional Installers

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

Python | Anaconda | I... | Microsoft Te... | Smadav 202... | Intro_to_AI (... | AI LAB-01 - ... | Lecture_1 M... | artificial_int... | 2:48 PM


download
KICSIT
sign
Emulated Turbo C...
wp2601087
EasyEDA
ML-Project
Microsoft Edge
Foxit Reader
Recycle Bin
Google Drive
Precision 3930 Rack ...
IMG_20180...
Recently added
Google Drive
#
3D Viewer
A
Alarms & Clock
Allegorithmic
Anaconda3 (64-bit)
Anaconda Navigator (anacond)
Anaconda Powershell Pron
Anaconda Prompt (anacon
Jupyter Notebook (anacon
Reset Spyder Settings (anacond)
Spyder (anacond)
Arduino
AutoCAD 2021 - English
Autodesk
Autodesk Maya 2020
Pin to Start
More
Uninstall
Productivity
Office
See all your mail in one place
Mail
Microsoft Edge
Photos
Explore
Mostly Clear
Movies & TV
Play
This PC
KALI LINUX
THE QUIETER YOU BECOME, THE MORE YOU ARE ABLE TO HEAR.
Python
anaconda navi...
Microsoft Teams
Intro_to_AI (1) ...
AI LAB-01 - Mi...
Lecture_1 ML (...
artificial_intelli...
2:53 PM

User Interface

Anaconda Navigator

File Help

ANACONDA.NAVIGATOR

Sign in

Home

Environments

Learning

Community

Applications on base (root) Channels Refresh

CMD.exe Prompt
0.1.1
Run a cmd.exe terminal with your current environment from Navigator activated
Launch

Datalore
Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team.
Launch

IBM Watson Studio Cloud
IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.
Launch

JupyterLab
3.0.14
An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.
Launch

Notebook
6.3.0
Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.
Launch

Powershell Prompt
0.0.1
Run a Powershell terminal with your current environment from Navigator activated
Launch

PyCharm Community
2020.3.3
An IDE by JetBrains for pure Python development. Supports code completion, listing, and debugging.
Launch

Qt Console
5.0.3
PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.
Launch

Documentation

Anaconda Blog

Python Learn... This PC NumPy ufun... Microsoft Te... PowerPoint ... artificial_inte... Anaconda N... Jupyter Note... 3:27 PM

Home Page - Select or

localhost:8888/tree

jupyter

Quit Logout

Files Running Clusters

Select items to perform actions on them.

Upload New

Notebook:
Python 3

Other:
Text File
Folder
Terminal

0 /

Name

3D Objects		
Contacts		
Desktop		
digit-recognizer		
Documents	18 days ago	
Downloads	20 minutes ago	
Favorites	7 months ago	
Links	8 months ago	
Music	9 months ago	
OneDrive	a month ago	
Pictures	7 months ago	
PycharmProjects	2 months ago	
Saved Games	9 months ago	
Searches	9 months ago	
Videos	6 months ago	
irisclustering.ipynb	a month ago	126 kB
K-means-1-UNI.ipynb	a month ago	54.8 kB

Home Page -... Python anaconda na... Microsoft Te... PowerPoint ... artificial_inte... Anaconda N... Jupyter Note... 2:55 PM

Home Page - Select or creat | Untitled3 - Jupyter Note

localhost:8888/notebooks/Untitled3.ipynb?kernel_name=python3

jupyter Untitled3 Last Checkpoint: a minute ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Trusted | Python 3

Run | Code

In [1]:
```
print('Welcome to Artificial Intelligence')
```

```
Welcome to Artificial Intelligence
```

In [ ]:

Untitled3 - Ju... | Python | anaconda na... | Microsoft Te... | PowerPoint ... | artificial_inte... | Anaconda N... | Jupyter Note... | 2:57 PM

Home Page - Select or creat | Untitled3 - Jupyter Note

localhost:8888/notebooks/Untitled3.ipynb?kernel_name=python3

jupyter Untitled3 Last Checkpoint: 3 minutes ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Markdown

Artificial Intelligence Lab

In [1]:
```
print('Welcome to Artificial Intelligence')
```

```
Welcome to Artificial Intelligence
```

In [ ]:

In [ ]:

Untitled3 - Ju... Python anaconda na... Microsoft Te... PowerPoint ... artificial_inte... Anaconda N... Jupyter Note... 2:59 PM

Python Basic Programming-I

Python Basic Programming-I

- Python Identifiers
- Variables & Data Types
- Python Operators
- Python strings & strings functions
- Data structures in python
  - Tuple ,Lists ,Dictionaries, Sets
- Decision making statements

Python Basic Programming

Identifiers: A Python identifier is a name used to identify a variable, function, class, module or other object.

- identifier starts with a letter A to Z or a to z
- an underscore (_) followed by zero or more letters
- underscores and digits (0 to 9).

```
a=10
A=20
_0123=False
_A23=5.2
```

Variables & Data Types

Variables: **A temporary storage used to store the data, each variable have some name and memory address.**

Data types: **Each variable is associated with some type of data but not mention while defining a variable**

Integer , float, Boolean ,string, complex

```
a=10
b=3.5
st='Artificial intelligence'
Ok=True
comNum=2+3j
```

Reserve Words

and	exec	Not
as	finally	or
assert	for	pass
break	from	print
class	global	raise
continue	if	return
def	import	try
del	in	while
elif	is	with
else	lambda	yield
except		

Python Strings & String Functions

Python String: **Sequence of a characters enclosed in single quotes (‘ ’), Double quotes (“ ”) or triple quotes (“’ ’”)**

```
S1='Artificial intelligence'
S2="Artificial intelligence"
S3=''' Welcome
        To
Artificial intelligence'''
```

```
print(S1)
print(S2)
print(S3)
```

```
Artificial intelligence
Artificial intelligence
 Welcome
        To
Artificial intelligence
```

Operators

- Arithmetic Operators
- Comparison (Relational) Operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

Arithmetic Operators
a=10, b=2

- Addition
- Subtraction
- Multiplication
- Division
- Modulus
- Exponent

```
a=10
b=2
```

```
print('Addition:',a+b)
print('Subtraction:',a-b)
print('Multiplication:',a*b)
print('Division:',a/b)
print('Modulus:',a%b)
print('Exponent:',a**b)
print('Divide (Floor):',a//b)
```

Comparison Operators
a=10, b=2

- **Equal (==)**
- **Not Equal (!=)**
- **Greater Then (>)**
- **Less Then(<)**
- **Greater Then Equal(>=)**
- **Less Then Equal(<=)**

```
print('a Equal b: ',a==b)
print('a Not Equal to b: ',a!=b)
print('a greater then b: ',a>b)
print('a less then b: ',a<b)
print('a greater then Equal to b: ',a>=b)
print('a less then Equal to b: ',a<=b)
```

```
a Equal b:  False
a Not Equal to b:  True
a greater then b:  True
a less then b:  False
a greater then Equal to b:  True
a less then Equal to b:  False
```

Bitwise Operators

a=10, b=2

- **Binary and (&)**
- **Binary or (|)**
- **Binary Not (~)**
- **Binary XOR (^)**
- **Binary Left Shift(<<)**
- **Binary Right Shift(>>)**

```
print('Binary of A=10',bin(a))
print('Binary of B=2',bin(b))
print('Binary AND (&)',bin(a&b))
print('Binary OR (|)',bin(a|b))
print('Binary NOT (~)',bin(~a))
print('Binary XOR (^)',bin(a^b))
print('Binary lEFT SHIFT (<<)',bin(a<<1))
print('Binary RIGHT SHIFT (>>) ',bin(a>>1))
```

```
Binary of A=10 0b1010
Binary of B=2 0b10
Binary AND (&) 0b10
Binary OR (|) 0b1010
Binary NOT (~) -0b1011
Binary XOR (^) 0b1000
Binary lEFT SHIFT (<<) 0b10100
Binary RIGHT SHIFT (>>)  0b101
```

Logical Operators
a=True, b=False

- **AND (and)**
- **OR (or)**
- **NOT (not)**

```python
a= True
b= False
print('Value of A:',a)
print('Value of B:',b)
print('A OR B ',a or b)
print('A AND B',a and b)
print('Not B ',not(b))
```

```
Value of A: True
Value of B: False
A OR B  True
A AND B False
Not B  True
```

Python Strings & String Functions

Python String: **Sequence of a characters enclosed in single quotes (‘ ’), Double quotes (“ ”) or triple quotes (“’ ’”)**

```
S1='Artificial intelligence'
S2="Artificial intelligence"
S3=''' Welcome
         To
Artificial intelligence'''
```

```
print(S1)
print(S2)
print(S3)
```

```
Artificial intelligence
Artificial intelligence
 Welcome
         To
Artificial intelligence
```

Python Strings & String Functions

```
S1='Artificial intelligence'
```

Extract Character from String

```
S1[0]
```

```
'A'
```

```
S1[-1]
```

```
'e'
```

Extract Sequence of Character

```
S1[0:10]
```

```
'Artificial'
```

```
S1[0:3]
```

```
'Art'
```

Replace Characters in a String

```
S1.replace('i','j')
```

```
'Artjfjcjal jntelljgence'
```

```
S1.replace('intelligence','Networks')
```

```
'Artificial Networks'
```

Python Strings & String Functions

```
S1='Artificial intelligence'
```

Convert to lower case

```
S1.lower()
```

```
'artificial intelligence'
```

Convert to upper case

```
S1.upper()
```

```
'ARTIFICIAL INTELLIGENCE'
```

Count Character in a String

```
S1.count('i')
```

```
5
```

Python Strings & String Functions

```
S1='Artificial intelligence'
```

Finding index

```
S1.find('i')
```

```
3
```

```
S1.find('intel')
```

```
11
```

String Splitting

```
S1.split(' ')
```

```
['Artificial', 'intelligence']
```

```
S1.split('i')
```

```
['Art', 'f', 'c', 'al ', 'ntell', 'gence']
```

Count Character in a String

```
S1.count('i')
```

```
5
```

Python Strings & String Functions

```
S1='Artificial intelligence'
```

Center

```
S1.center(40,'*')
```

```
'********Artificial intelligence*********'
```

String min/max

```
min(S1)
```

```
' '
```

```
max(S1)
```

```
't'
```

String is digit/alpha

```
S1.isdigit()
```

```
False
```

```
S1.isalpha()
```

```
False
```

Python Strings & String Functions

```
S1='Artificial intelligence'
```

Some other is functions:

- **isnumaric()**
- **islower()**
- **isspace()**
- **isupper()**
- **istitle()**
- **istrip()**

Python Strings & String Functions

```
S1='Artificial intelligence'
```

String starts with

```
S1.startswith('A')
```

```
True
```

String ends with

```
S1.endswith('e')
```

```
True
```

```
S1.endswith('!')
```

```
False
```

```
S1.endswith('e',5,len(S1))
```

```
True
```

String is digit/alpha

```
S1.isdigit()
```

```
False
```

```
S1.isalpha()
```

```
False
```

Data structures in Python

There are four basic data structures are in python:

- Tuple
- List
- Dictionary
- sets

Tuple

- A tuple is an ordered collection of elements
- immutable Python objects, tuples cannot be changed
- Can Store heterogeneous elements
- Tuples use parentheses ()
- Elements separated by comma (,)

```
tup1=(1,2,3,4)
tup2=('A','I',1,2,3)
tup3=('Artificial','Intelligence ',7,'Fall',2021)
```

Tuple

```
tup3=('Artificial','Intelligence ',7,'Fall',2021)
```

Accessing Elements from Tuple:

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain the value available at that index.

```
print(tup3)
print(tup3[0])
print(tup3[0:2])
```

```
('Artificial', 'Intelligence ', 7, 'Fall', 2021)
Artificial
('Artificial', 'Intelligence ')
```

Tuple

```
tup3=('Artificial','Intelligence ',7,'Fall',2021)
```

Update Elements in Tuple:

Tuple is immutable ,we can not modify the elements

```
tup3[1]='Networks'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-94-131abc003515> in <module>
----> 1 tup3[1]='Networks'

TypeError: 'tuple' object does not support item assignment
```

Tuple

```
tup3=('Artificial','Intelligence ',7,'Fall',2021)
```

Tuple Length:

```
print('Tuple contain Elements:',len(tup3))
```

```
Tuple contain Elements: 5
```

```
print('Tuple contain Elements:',len((1,2,5,7)))
```

```
Tuple contain Elements: 4
```

Tuple

```
tup3=('Artificial','Intelligence ',7,'Fall',2021)
```

Tuple Min/Max Element: will work if Tuple contains elements of same type

```
print('Minimum Value',min(8,7,5,6,8,0,4,1))
print('Maximum Value',max(8,7,5,6,8,0,4,1))
```

```
Minimum Value 0
Maximum Value 8
```

Tuple

Tuples Concatenation: join 2 or more tuples

```
print(tup1)
print(tup2)
print(tup1+tup2)
```

```
(1, 2, 3, 4)
('A', 'I', 1, 2, 3)
(1, 2, 3, 4, 'A', 'I', 1, 2, 3)
```

Tuple Repetition: Repeat tuple many time

```
print(tup1*3)
```

```
(1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4)
```

Tuple

Tuples with membership:

- Membership operators are used to either specific elements is present IN or NOT IN inside a variable.
- Return Boolean value true or false
- Mostly used with decision making statements (if-else) and flow control (loops)

```
tup=(1,5,8,32,3,4,87,12)
print('Element 3 in Tuple ?', 3 in tup)
print('Element 12 Not in Tuple ?',12 not in tup)
```

```
Element 3 in Tuple ? True
Element 12 Not in Tuple ? False
```

List

- The list is the most versatile data type available in Python its an order collection of elements
- Elements written inside square brackets [ ] , comma – separated (,) elements.
- List is Mutable
- Important thing about a list is that the items in a list need not be of the same type.

```
list1=[1,2,3,4]
list2=['A','I',1,2,3]
list3=['Artificial','Intelligence ',7,'Fall',2021]
```

List

```
list3=['Artificial','Intelligence ',7,'Fall',2021]
```

Accessing Elements from List:

To access values in list, use the square brackets for slicing along with the index or indices to obtain the value available at that index. Just like a tuple

```
print(list3)
```

```
['Artificial', 'Intelligence ', 7, 'Fall', 2021]
```

```
print('List First Element',list3[0])
print('List Last Element',list3[-1])
print('Slicing',list3[1:4])
print('Slicing',list3[2:])
```

```
List First Element Artificial
List Last Element 2021
Slicing ['Intelligence ', 7, 'Fall']
Slicing [7, 'Fall', 2021]
```

List

```
list3=['Artificial','Intelligence ',7,'Fall',2021]
```

Updating Lists :

List is Mutable so we can modify elements in the list

```
list3[1]='Networks'
print(list3)
```

```
['Artificial', 'Networks', 7, 'Fall', 2021]
```

List

```
list3=['Artificial','Intelligence ',7,'Fall',2021]
```

List Length:

```
print('List contain Elements:',len(list3))
```

```
List contain Elements: 5
```

```
print('List contain Elements:',len([1,5,7,8,9,6]))
```

```
List contain Elements: 6
```

List

```
list3=['Artificial','Intelligence ',7,'Fall',2021]
```

List Min/Max Element: will work if list contains elements of same type

```
print('Minimum Value',min(list3))
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-65-215c454a4152> in <module>
----> 1 print('Minimum Value',min(list3))

TypeError: '<' not supported between instances of 'int' and 'str'
```

```
print('Minimum Value',min([8,7,5,6,8,0,4,1]))
print('Maximum Value',max([8,7,5,6,8,0,4,1]))
```

```
Minimum Value 0
Maximum Value 8
```

List

List Concatenation: join 2 or more list

```
print(list1)
print(list2)
print(list1+list2)
```

```
[1, 2, 3, 4]
['A', 'I', 1, 2, 3]
[1, 2, 3, 4, 'A', 'I', 1, 2, 3]
```

List Repetition: Repeat list many time

```
print(list1*2)
```

```
[1, 2, 3, 4, 1, 2, 3, 4]
```

List

List with membership:

- Membership operators are used to either specific elements is present IN or NOT IN inside a variable.
- Return Boolean value true or false
- Mostly used with decision making statements (if-else) and flow control (loops)

```
lis=[11,54,8,32,73,84,85,12]
print('Element 18 in List ?', 18 in lis)
print('Element 73 Not in Tuple ?',73 not in lis)
```

```
Element 18 in List ? False
Element 73 Not in Tuple ? False
```

List

insert Element: Used to insert data in the list associated with Index

```
list3.insert(1,5)
list3.insert(1,5)
list3.insert(1,5)
print(list3)
```

```
['Artificial', 5, 5, 5, 'Networks', 7, 'Fall', 2021]
```

Pop Element: Used to remove and return data from the list associated with Index

```
list3.pop(1)
```

```
5
```

```
print(list3)
```

```
['Artificial', 5, 5, 'Networks', 7, 'Fall', 2021]
```

List

Remove Element: Used to remove data from the list

```
print(list3)
list3.remove('Networks')
print(list3)
```

```
['Artificial', 5, 5, 'Networks', 7, 'Fall', 2021]
['Artificial', 5, 5, 7, 'Fall', 2021]
```

Index of Element: Used to remove and return data from the list associated with Index

```
print(list3)
print(list3.index('Fall'))
```

```
['Artificial', 5, 5, 7, 'Fall', 2021]
4
```

Dictionary

Dictionary is a key-value pairs of data

Unordered collection of data

Enclosed in curly braces {}

Dictionary is Mutable

Key-value pairs (Elements) separated using comma (,)

```
dict={1:'Enginnering',
      2:'Technology',
      3:2021}
dict={1:'Enginnering' ,2:'Technology',3:2021}
```

Dictionary

Create and print Dictionary:

```
dict={1:'Enginnering' ,2:'Technology',3:2021}
dict2={1:'A' ,2:'B',3:'C',4:'D',5:'E'}
print(dict)
print(dict2)
type(dict)
```

```
{1: 'Enginnering', 2: 'Technology', 3: 2021}
{1: 'A', 2: 'B', 3: 'C', 4: 'D', 5: 'E'}

dict
```

Dictionary

Access element: you can access value using key

```
print(dict)
print('Value of Key 2:',dict[1])
```

```
{1: 'Enginnering', 2: 'Technology', 3: 2021}
Value of Key 2: Enginnering
```

Access items from:

```
print(dict.items())
```

```
dict_items([(1, 'Enginnering'), (2, 'Technology'), (3, 2021)])
```

Access Keys:

```
print(dict.keys())
```

```
dict_keys([1, 2, 3])
```

Access Values:

```
print(dict.values())
```

```
dict_values(['Enginnering', 'Technology', 2021])
```

Dictionary

Update: append elements from one dictionary to second

```
dict={1:'Enginnering' ,2:'Technology',3:2021}
dict2={'A':'Apple' ,5:'Ball',6:'Cat',7:'Dog'}
print(dict)
print(dict2)
dict.update(dict2)
print(dict)
```

```
{1: 'Enginnering', 2: 'Technology', 3: 2021}
{'A': 'Apple', 5: 'Ball', 6: 'Cat', 7: 'Dog'}
{1: 'Enginnering', 2: 'Technology', 3: 2021, 'A': 'Apple', 5: 'Ball', 6: 'Cat', 7: 'Dog'}
```

Copy : Copy elements of one dictionary to second

```
print(dict2)
new_copy=dict2.copy()
print(new_copy)
```

```
{'A': 'Apple', 5: 'Ball', 6: 'Cat', 7: 'Dog'}
{'A': 'Apple', 5: 'Ball', 6: 'Cat', 7: 'Dog'}
```

Sets

Sets:

Unordered and unindexed sequence of data

Enclosed in curly braces { }

Repetition (duplication) not allowed in sets

```
s1={1,2,4,5}
s2={1,2,5,"AI"}
s3={'AI','Computer','Engineering'}
```

Sets

Check duplication:

```
s1={1,2,4,5}
s2={1,2,5,"AI"}
s3={'AI','Computer','Engineering'}
s4={1,1,2,4,2,1,3,7,6,}
print(s1)
print(s2)
print(s3)
print(s4)
```

```
{1, 2, 4, 5}
{1, 2, 5, 'AI'}
{'AI', 'Computer', 'Engineering'}
{1, 2, 3, 4, 6, 7}
```

Sets

Type and Length of sets:

```
s1={1,2,4,5}
s2={1,2,5,"AI"}
s3={'AI','Computer','Engineering'}
s4={1,1,2,4,2,1,3,7,6,}
print('Type of S1: ',type(s1),' Length of S1: ',len(s1))
print('Type of S2 ',type(s2),' Length of S2 ',len(s2))
print('Type of S3 ',type(s3),' Length of S3 ',len(s3))
print('Type of S4 ',type(s4), ' Length of S4',len(s4))
```

```
Type of S1:  <class 'set'>  Length of S1:  4
Type of S2  <class 'set'>  Length of S2  4
Type of S3  <class 'set'>  Length of S3  3
Type of S4  <class 'set'>  Length of S4 6
```

Sets

Add Element:

```
s1={1,2,4,5}
print(s1)
s1.add(6)
print(s1)
```

```
{1, 2, 4, 5}
{1, 2, 4, 5, 6}
```

Remove Element :

```
print(s1)
s1.remove(5)
print(s1)
```

```
{1, 2, 4, 5, 6}
{1, 2, 4, 6}
```

Sets

Union of Sets:

```
s1={1,2,4,5}
s2={4,8,5,12,4}
print(s1.union(s2))
```

```
{1, 2, 4, 5, 8, 12}
```

Intersection of Sets:

```
s1={1,2,4,5}
s2={4,8,5,12,4}
print(s1.intersection(s2))
```

```
{4, 5}
```

Decision Making

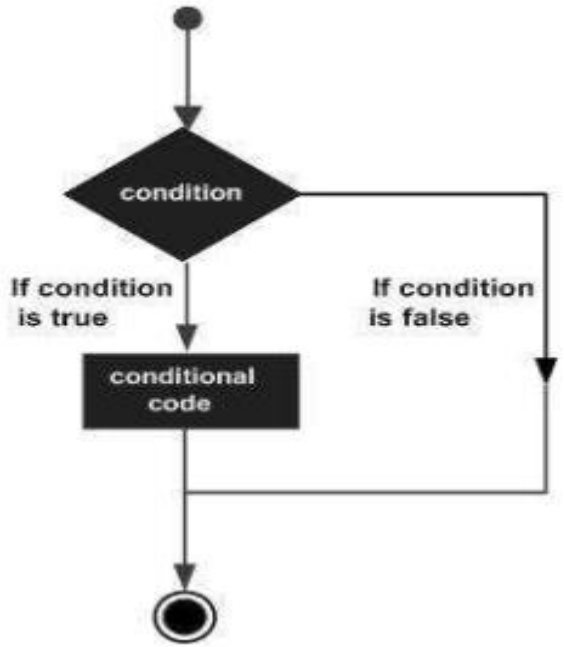
Decision-making is the anticipation of conditions occurring during the execution of a program and specified actions taken according to the conditions.

Performs action when some specified condition is occurred

Returns Boolean value TRUE or FASLE

Decision statements are

- If statements
- If-else statements
- Nested if-else statements



If statements in python

Simple If-Statement: Check A is equal 10

```
A=10
if A==10:
    print('Yes A is 10')
```

```
Yes A is 10
```

If statements in python

If-Statement with Tuple: Check value 10 is present in tuple

```
tup=(1,4,1,7,12,4,10,14,7,75)
A=10

if A in tup:
    print("Yes, Value of A is present in The Tuple")
```

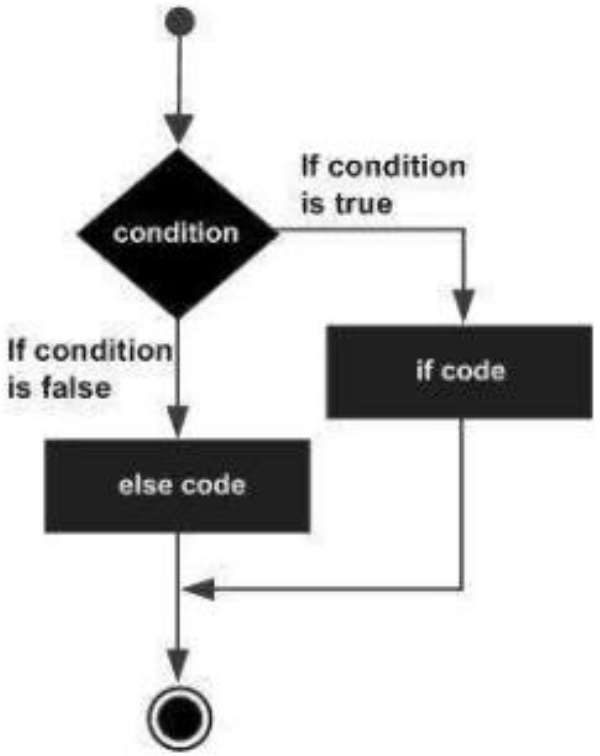
```
Yes, Value of A is present in The Tuple
```

If-Else statements in python

Single If-else Statement: Check A is equal 10 or Not Equal

```
A=9
if A==10:
    print('Yes A is 10')
else:
    print('No, A is 10')
```

```
No, A is 10
```



If-Else statements in python

Multi IF-ELSE Statements:

```
A=9
if A>9:
    print('No, A is Not Greater then 10')
elif A==9:
    print('Yes, A is equal to 9')
else:
     print('A is Less then 9')
```

```
Yes, A is equal to 9
```

Python Basic Programming-II

Python Basic Programming-II

- Decision making statements
  - If, If-Else, Nested if-else
- Flow Control Statements (Loops)
  - For, While loops
- Functions
- Lambda function
- Filtering and mapping
- Program Practices

Decision Making

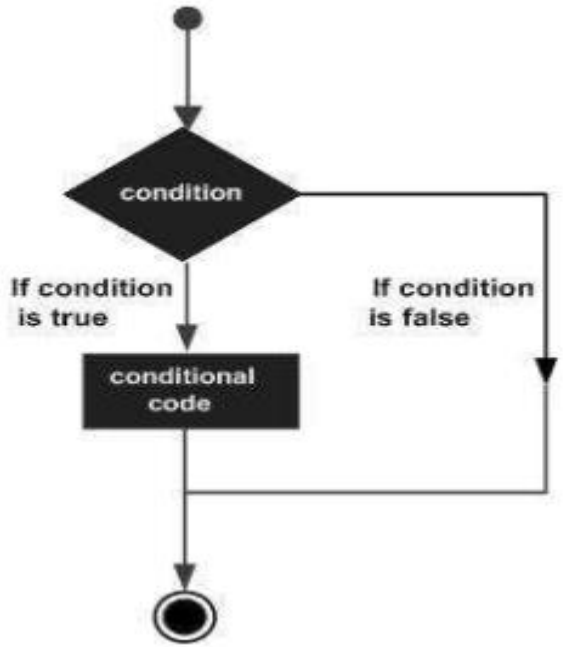
Decision-making is the anticipation of conditions occurring during the execution of a program and specified actions taken according to the conditions.

Performs action when some specified condition is occurred

Returns Boolean value TRUE or FASLE

Decision statements are

- If statements
- If-else statements
- Nested if-else statements



If statements in python

Simple If-Statement: Check A is equal 10

```
A=10
if A==10:
    print('Yes A is 10')
```

```
Yes A is 10
```

If statements in python

If-Statement with Tuple: Check value 10 is present in tuple

```
tup=(1,4,1,7,12,4,10,14,7,75)
A=10

if A in tup:
    print("Yes, Value of A is present in The Tuple")
```

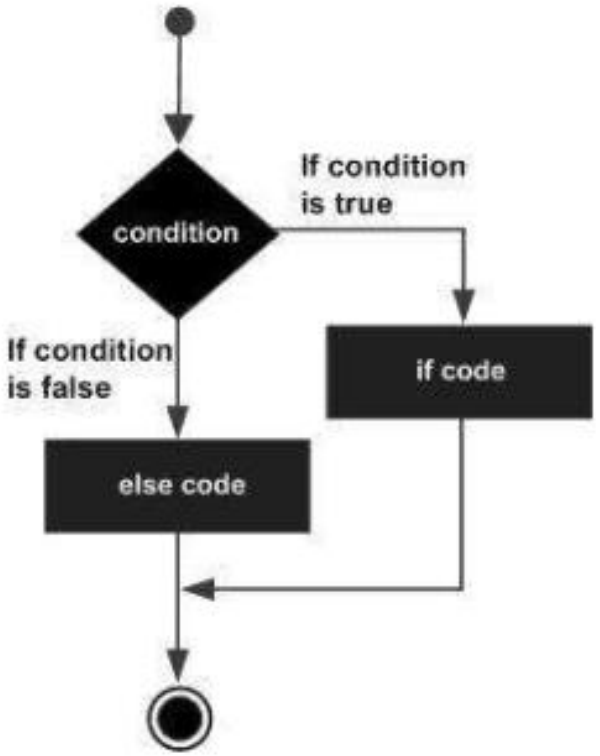
```
Yes, Value of A is present in The Tuple
```

If-Else statements in python

Single If-else Statement: Check A is equal 10 or Not Equal

```
A=9
if A==10:
    print('Yes A is 10')
else:
    print('No, A is 10')
```

```
No, A is 10
```



If-Else statements in python

Multi IF-ELSE Statements:

```
A=9
if A>9:
    print('No, A is Not Greater then 10')
elif A==9:
    print('Yes, A is equal to 9')
else:
     print('A is Less then 9')
```

```
Yes, A is equal to 9
```

Loops

In general, statements are executed sequentially- The first statement in a function is executed first, followed by the second, and so on. There may be a situation when you need to execute a block of code several number of times.
Programming languages provide various control structures that allow more complicated execution paths.
A loop statement allows us to execute a statement or group of statements multiple times.
The following diagram illustrates a loop statement.