KANDULA SRINIVASA REDDY MEMORIAL COLLEGE OF ENGINEERING
(AUTONOMOUS)

KADAPA-516003. AP

(Approved by AICTE, Affiliated to JNTUA, Ananthapuramu, Accredited by NAAC)

(An ISO 9001-2008 Certified Institution)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATION COURSE
ON

"Python for Everybody"

Resource Person : 1. Mrs. B. Gouri, Assistant Professor, Dept. of CSE, KSRMCE

2. Mrs. V. Sudha, Assistant Professor, Dept. of CSE, KSRMCE

Course Coordinator: Dr. V. Lokeswara Reddy, Professor, Dept. of CSE, KSRMCE

Duration : 10/09/2018 to 29/09/2018

K.S.R.M. COLLEGE OF ENGINEERING

(UGC - AUTONOMOUS)

Kadapa, Andhra Pradesh, India - 516003

Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.

An ISO 14001:2004 & 9001: 2015 Certified Institution

Lr./KSRMCE/ (Department of CSE)/2018-19

Date: 05/09/2018

To
The Principal
KSRM College of Engineering
Kadapa, AP.

Sub: KSRMCE - (Department of CSE) – Permission to conduct certification course on Python for Everybody - Requested – reg.

---***---

Respected Sir,

With reference to the cited, the Department of CSE is planning to conduct certificate course on Python for Everybody" for B.Tech students from **10/09/2018 to 29/09/2018**. So I request you to grant permission to conduct the certificate course. This is submitted for your kind perusal.

Thanking you sir,

Yours Faithfully,

Coordinator,
Dr. V. Lokeswara Reddy,
Professor, CSE Dept.,
KSRMCE.

Forwarded to the principal sir,

Cc:

To The Director for Information

To All Deans/HODs

Permitted
V.S.S. Mu/5

K.S.R.M. COLLEGE OF ENGINEERING

(UGC - AUTONOMOUS)

Kadapa, Andhra Pradesh, India - 516003

Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.

An ISO 14001:2004 & 9001: 2015 Certified Institution

Dated: 06/09/2018

Circular

All the B.Tech students are here by informed that department of Computer Science & Engineering is going to organize certification course on **Python for Everybody"** from **10/09/2018 to 29/09/2018**. Interested students do register their names with the below mentioned coordinator on or before 07/09/2018, 5PM.

For any queries contact,

Coordinator:
Dr. V. Lokeswara Reddy,
Professor,
CSE Dept.,
KSRMCE.

HoD

Dr. M. Sreenivasulu,
M. E., Ph. D.,
Professer & HOD CSE
K. S. R. M. College of Engineering
KADAPA - 516 003

Cc to:

The Management /Director / All Deans / All HODS/Staff / Students for information

The IQAC Cell for Documentation

K.S.R.M. COLLEGE OF ENGINEERING

(UGC - AUTONOMOUS)

Kadapa, Andhra Pradesh, India - 516003
Approved by AICTE, New Delhi & Affiliated to JNTUA,
Ananthapuramu.
An ISO 14001:2004 & 9001: 2015 Certified Institution

Date: 06/09/2018

Department of Computer Science & Engineering
Certificate Course on Python for Everybody from 10/09/2018 to 29/09/2018
Registered Student List

S.NO	ROLL NUMBER	NAME OF THE STUDENT	YEAR & BRANCH	SIGNATURE

Coordinator **HoD**

K.S.R.M. COLLEGE OF ENGINEERING

(UGC - AUTONOMOUS)

Kadapa, Andhra Pradesh, India - 516003
Approved by AICTE, New Delhi & Affiliated to JNTUA,
Ananthapuramu.
An ISO 14001:2004 & 9001: 2015 Certified Institution

Date: 06/09/2018

Department of Computer Science & Engineering

Certificate Course on Python for Everybody from 10/09/2018 to 29/09/2018

Registered Student List

S.NO	ROLL NUMBER	NAME OF THE STUDENT	YEAR & BRANCH	SIGNATURE
1	159Y1A0504	A. Venkata Bhavana	B.Tech VII sem	A. Venkata Bhavana
2	159Y1A0503	A. Lavanya	B.Tech VII Sem	A. Lavanya
3.	159Y1A0508	B. Narasimha Reddy	VII Semester	
4	159y1A0512	B. Ashok Vinay Kumar	7th Sem	Vinay
5	159Y1A0514	B. Sireesha	7 sem	Sireesha
6	159Y1A0515	B. Nagendra Bhavani	VII sem	Bhavani
7	159Y1A0523	B. Manjusha	VII Sem	B. Manjusha
8	159Y1A0501	A. Srikanth	7th sem	A..
9	159Y1A0505	A. Shobhasankar Reddy	7th sem	
10	159y1a0507	B. Vijaya Kumar Reddy	VII sem	B. Vijaya Kumar Reddy
11	159Y1A0513	B. Bharat Kumar Reddy	7th sem	Bharat
12	159Y1A0534	C. Archana.	VII sem	C. Archana
13	159Y1A0502	A. Venkata Rohitha Reddy	7th Semester	

14	159Y1A0509	B.KRISHNA KISHORE	B.TECH VII SEM	B. KRISHNA KISHORE
15	159Y1A0516	B. Uma Devi	VII sem	UmaDevi
16	159Y1A0531.	D. Anil Kumar Yadav	VII Sem	D.Anil
17	159Y1A0536	E.Sai Hanuman.	VII sem	Saij.
18	159Y1A0525	C. Pavan Kumar Reddy	VII sem	C.pavan kumarReddy.
19	159Y1A0534	D.Raja Ramani	VII sem	DRR
20	159Y1A0526	C.Lakshmi priya	VII sem	C.Lakshmi priya
21	159Y1a0517	B. Sravani	7 sem	Sravani
22	159Y1a0519	B. Vardhan	7 sem	Vardhan
23	159Y1A0510	B. sravani	B.Tech VII sem	B. Sravani
24	159Y1A0518	B. Madhav Reddy	VII	madavReddy
25	159Y1A0520	B. Kavitha	.VIIsem	Kavitha.
26	159Y1a0521	B.V. Rajasree	7 sem	Rajasree
27	159Y1A0522	C. HARIKRISHNA	VII Sem	Harikrishna
28	159Y1A0532	D.SABAREESH	VII sem	DS
29	159Y1A0527	C MADHURI	VII sem.	C.Madhuri
30	159Y1A0535	D. YASASWINI	VII sem	DYas
31	159Y1A0528	C. Nagamani	VII sem	C.Nagamani
32	159Y1A0533	D.Ratesh.	VII sem	DRatesh
33	159Y1A0529	C. Nikhitha	VII Sem	C. Nikhitha
34	159Y1A0530	C. HarshavardhanReddy	VIIsem	C.HarshavardhanReddy.
35	159Y1A0511	B. Geethanjali	B.Tech VII sem	B. Geethanjali

Coordinator

HoD

Dr. M. Sreenivasulu,
M. E., Ph. D.
Professer & HOD CSE
K.S R.M. College of Engineering
KADAPA - 516 003

K.S.R.M. COLLEGE OF ENGINEERING

(UGC - AUTONOMOUS)

Kadapa, Andhra Pradesh, India - 516003
Approved by AICTE, New Delhi & Affiliated to JNTUA,
Ananthapuramu.
An ISO 14001:2004 & 9001: 2015 Certified Institution

Date: 06/09/2018

Department of Computer Science & Engineering
Certificate Course on Python for Everybody from 10/09/2018 to 29/09/2018
Registered Student List

S.NO.	ROLL NUMBER	NAME OF THE STUDENT	YEAR & BRANCH	EMAIL ID
1	159Y1A0501	A. Srikanth	B.Tech VII Sem	159Y1A0501@ksrmce.ac.in
2	159Y1A0502	A. Venkata Rohitha Reddy	B.Tech VII Sem	159Y1A0502@ksrmce.ac.in
3	159Y1A0503	A. Lavanya	B.Tech VII Sem	159Y1A0503@ksrmce.ac.in
4	159Y1A0504	A.Venkata Bhavana	B.Tech VII Sem	159Y1A0504@ksrmce.ac.in
5	159Y1A0505	A Shobhasankar Reddy	B.Tech VII Sem	159Y1A0505@ksrmce.ac.in
6	159Y1A0507	B. Vijaya Kumar Reddy	B.Tech VII Sem	159Y1A0507@ksrmce.ac.in
7	159Y1A0508	B. Narasimha Reddy	B.Tech VII Sem	159Y1A0508@ksrmce.ac.in
8	159Y1A0509	B. Krishna Kishore	B.Tech VII Sem	159Y1A0509@ksrmce.ac.in
9	159Y1A0510	B. Sravani	B.Tech VII Sem	159Y1A0510@ksrmce.ac.in
10	159Y1A0511	B. Geethanjali	B.Tech VII Sem	159Y1A0511@ksrmce.ac.in
11	159Y1A0512	B. Ashok Vinay Kumar	B.Tech VII Sem	159Y1A0512@ksrmce.ac.in
12	159Y1A0513	B. Bharath Kumar Reddy	B.Tech VII Sem	159Y1A0513@ksrmce.ac.in
13	159Y1A0514	B. Sireesha	B.Tech VII Sem	159Y1A0514@ksrmce.ac.in
14	159Y1A0515	B. Nagendra Bhavani	B.Tech VII Sem	159Y1A0515@ksrmce.ac.in
15	159Y1A0516	B. Uma Devi	B.Tech VII Sem	159Y1A0516@ksrmce.ac.in
16	159Y1A0517	B. Sravani	B.Tech VII Sem	159Y1A0517@ksrmce.ac.in
17	159Y1A0518	B. Madhava Reddy	B.Tech VII Sem	159Y1A0518@ksrmce.ac.in
18	159Y1A0519	B. Vardhan	B.Tech VII Sem	159Y1A0519@ksrmce.ac.in
19	159Y1A0520	B. Kavitha	B.Tech VII Sem	159Y1A0520@ksrmce.ac.in
20	159Y1A0521	B. Venkata Raja Sree	B.Tech VII Sem	159Y1A0521@ksrmce.ac.in
21	159Y1A0522	C. Harikrishna	B.Tech VII Sem	159Y1A0522@ksrmce.ac.in
22	159Y1A0523	C. Manjusha	B.Tech VII Sem	159Y1A0523@ksrmce.ac.in
23	159Y1A0524	C. Archana	B.Tech VII Sem	159Y1A0524@ksrmce.ac.in
24	159Y1A0525	C. Pavan Kumar Reddy	B.Tech VII Sem	159Y1A0525@ksrmce.ac.in
25	159Y1A0526	C. Lakshmi Priya .	B.Tech VII Sem	159Y1A0526@ksrmce.ac.in
26	159Y1A0527	C. Madhuri	B.Tech VII Sem	159Y1A0527@ksrmce.ac.in

27	159Y1A0528	C. Nagamani	B.Tech VII Sem	159Y1A0528@ksrmce.ac.in
28	159Y1A0529	C. Nikhitha	B.Tech VII Sem	159Y1A0529@ksrmce.ac.in
29	159Y1A0530	C. Harshavardhan Reddy	B.Tech VII Sem	159Y1A0530@ksrmce.ac.in
30	159Y1A0531	D. Anil Kumar Yadav	B.Tech VII Sem	159Y1A0531@ksrmce.ac.in
31	159Y1A0532	D. Sabareesh	B.Tech VII Sem	159Y1A0532@ksrmce.ac.in
32	159Y1A0533	D. Rajesh	B.Tech VII Sem	159Y1A0533@ksrmce.ac.in
33	159Y1A0534	D. Roja Rani	B.Tech VII Sem	159Y1A0534@ksrmce.ac.in
34	159Y1A0535	D. Yasaswini	B.Tech VII Sem	159Y1A0535@ksrmce.ac.in
35	159Y1A0536	E. Sai Hanuman	B.Tech VII Sem	159Y1A0536@ksrmce.ac.in

Coordinator

HoD

Dr. M. Sreenivasulu,
M E , Ph. D.
Professer & HOD CSE
K.S R.M. College of Engineering
KADAPA - 516 003

K.S.R.M. College of Engineering (Autonomous), Kadapa.
Department of CSE
Certificate Course on
Python for Everybody
Syllabus

Overview: Python is **a high-level, interpreted, interactive and object-oriented scripting language**. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

Course Objectives: To learn the basics of algorithm problem solving, read and write simple python programs, develop python programs with conditional and loops, define functions and call them, strings, list, dictionaries.

Course Outcomes: At the end of the course participants will be able to

- Interpret the fundamental Python syntax and semantics and be fluent in the use of Python control flow statements.
- Determine the methods to create and manipulate Python programs by utilizing the data structures like lists, dictionaries, tuples and sets.

Module 1: Algorithms, building blocks of algorithms, notation, algorithm problem solving, simple strategies for developing algorithms, python interpreter and interactive mode, values and types, statements, assignment, precedence of operators, comments.

Module 2: Conditionals: Boolean values and operators, conditional (if), alternative (if-else), chained conditional (if-elif-else), Iterations: state, while, for, break, continue, pass, functions: return values, parameters.

Module 3: Strings: string slices, immutability, string functions and methods, string module, lists as arrays. Lists: list operations, list slices, list methods, list loop, mutability, aliasing, cloning lists, list parameters.

Module 4: Dictionaries: operations and methods, advanced list processing, illustrative programs: selection sort, insertion sort, merge sort, histogram.

Module 5: Files and exceptions: text files, reading and writing files, format operator, handling exceptions, modules, packages.

Textbook:

1. Core python programming by Wesley J Chun, Prentice Hall, Second edition.
2. Introduction to Computation and Programming using Python, by John Guttag, PHI Publisher.
3. Learning python, Mark Lutz, O'Reilly publications, 5th edition, 2013.
4. Core python programming by Dr. R. Nageswara Rao, Dreamtech press, second edition, 2018

K.S.R.M. COLLEGE OF ENGINEERING

(UGC - AUTONOMOUS)

Kadapa, Andhra Pradesh, India - 516003
Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.
An ISO 14001:2004 & 9001: 2015 Certified Institution

Department of Computer Science & Engineering
Certificate Course on Python for Everybody from 10/09/2018 to 29/09/2018
Schedule

S.No	Date	Time	Faculty	Topic
1	10/09/2018	4PM t0 5PM	Dr. V. Lokeswara Reddy, Mrs. B. Gouri, Mrs. V. Sudha	Inauguration, Introduction to python programming language.
		5PM to 6PM	Mrs. B. Gouri	Algorithms, building blocks of algorithms, notation
2	11/09/2018	4PM to 5PM	Mrs. B. Gouri	Algorithm problem solving, simple strategies for developing algorithms
		5PM to 6PM	Mrs. B. Gouri	python interpreter and interactive mode
3	12/09/2018	3PM to 5PM	Mrs. B. Gouri	Values and types, statements, assignment, precedence of operators, comments.
		5PM to 6PM	Mrs. B. Gouri	Practice programs on lab
4	13/09/2018	4PM to 5PM	Mrs. B. Gouri	Conditionals: Boolean values and operators, conditional (if), alternative (if-else), chained conditional (if-elif-else)
		5PM to 6PM	Mrs. B. Gouri	Practice programs on lab
5	14/09/2018	3PM to 5PM	Mrs. V. Sudha	Iterations: state, while, for, break, continue, pass, functions: return values, parameters.
		5PM to 6PM	Mrs. V. Sudha	Practice programs on lab
6	15/09/2018	3PM to 5PM	Mrs. V. Sudha	Strings: string slices, immutability, string functions and methods
		5PM to 6PM	Mrs. V. Sudha	Practice programs on lab
7	17/09/2018	3PM to 5PM	Mrs. B. Gouri	string module, lists as arrays
		5PM to 6PM	Mrs. B. Gouri	Practice programs on lab
8	18/09/2018	4PM to 5PM	Mrs. B. Gouri	Lists: list operations, list slices, list methods
		5PM to 6PM	Mrs. B. Gouri	Practice programs on lab
9	19/09/2018	4PM to 5PM	Mrs. B. Gouri	list loop, mutability, aliasing, cloning lists, list parameters.
		5PM to 6PM	Mrs. B. Gouri	Practice programs on lab
10	21/09/2018	3PM to 5PM	Mrs. B. Gouri	Dictionaries: operations and methods,
		5PM to 6PM	Mrs. B. Gouri	Practice programs on lab

11	22/09/2018	4PM to 5PM	Mrs. B. Gouri	Advanced list processing
		5PM to 6PM	Mrs. B. Gouri	Practice programs on lab
12	25/09/2018	4PM to 5PM	Mrs. B. Gouri	Selection sort, insertion sort
		5PM to 6PM	Mrs. B. Gouri	Practice programs on lab
13	26/09/2018	4PM to 5PM	Mrs. B. Gouri	Merge sort, histogram
		5PM to 6PM	Mrs. B. Gouri	Practice programs on lab
14	27/09/2018	3PM to 5PM	Mrs. B. Gouri	Files and exceptions: text files, reading and writing files, format operator
		5PM to 6PM	Mrs. B. Gouri	Practice programs on lab
15	28/09/2018	4PM to 5PM	Mrs. B. Gouri	Handling exceptions, modules, packages
		5PM to 6PM	Mrs. B. Gouri	Practice programs on lab
16	29/09/2018	4PM to 6PM	Dr. V. Lokeswara Reddy, Mrs. B. Gouri, Mrs. V. Sudha	Exam, Certificate distribution

Coordinator

HoD

Dr. M. Sreenivasulu,
M. E., Ph. D.
Professer & HOD CSE
K.S.R.M. College of Engineering
KADAPA - 516 003

K.S.R.M. COLLEGE OF ENGINEERING

(UGC - AUTONOMOUS)

Kadapa, Andhra Pradesh, India - 516003
Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.
An ISO 14001:2004 & 9001: 2015 Certified Institution

Department of Computer Science & Engineering

Certificate Course on Python for Everybody from 10/09/2018 to 29/09/2018

Attendance Sheet

S.No	Roll Num	Name of the Student	10/09/2018	11/09/2018	12/09/2018	13/09/2018	14/09/2018	15/09/2018	17/09/2018	18/09/2018	19/09/2018	21/09/2018	22/09/2018	25/09/2018	26/09/2018	27/09/2018	28/09/2018	29/09/2018
1	159Y1A0501	A. Srikanth	P	P	A	P	P	P	P	P	P	P	P	P	P	P	P	P
2	159Y1A0502	A. Venkata Rohitha Reddy	P	P	P	P	P	P	P	P	P	A	P	P	P	P	P	P
3	159Y1A0503	A. Lavanya	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
4	159Y1A0504	A.Venkata Bhavana	P	P	P	P	P	P	A	P	P	P	A	P	P	P	P	P
5	159Y1A0505	A Shobhasankar Reddy	P	P	P	A	P	P	P	P	P	P	P	P	P	P	P	P
6	159Y1A0507	B. Vijaya Kumar Reddy	P	P	P	P	P	P	P	P	A	P	P	P	P	P	A	P
7	159Y1A0508	B. Narasimha Reddy	P	P	P	P	P	A	P	P	P	P	P	P	P	P	P	P
8	159Y1A0509	B. Krishna Kishore	P	P	P	P	P	P	P	P	P	A	P	P	P	P	P	P
9	159Y1A0510	B. Sravani	P	P	P	P	P	P	P	A	P	P	P	P	P	P	P	P
10	159Y1A0511	B. Geethanjali	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
11	159Y1A0512	B. Ashok Vinay Kumar	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
12	159Y1A0513	B. Bharath Kumar Reddy	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
13	159Y1A0514	B. Sireesha	P	A	P	P	A	P	P	P	A	P	P	P	P	P	P	P
14	159Y1A0515	B. Nagendra Bhavani	P	P	A	P	P	P	P	P	P	P	P	P	P	P	P	P

15	159Y1A0516	B. Uma Devi	P	P	P	A	P	P	P	P	P	P	P	P	P	P	P	P
16	159Y1A0517	B. Sravani	P	P	P	P	P	P	P	P	P	P	P	A	P	P	P	P
17	159Y1A0518	B. Madhava Reddy	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
18	159Y1A0519	B. Vardhan	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
19	159Y1A0520	B. Kavitha	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
20	159Y1A0521	B. Venkata Raja Sree	P	P	P	P	P	P	P	A	P	P	P	P	P	P	P	P
21	159Y1A0522	C. Harikrishna	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
22	159Y1A0523	C. Manjusha	P	P	P	P	P	P	P	P	P	P	A	P	P	P	P	P
23	159Y1A0524	C. Archana	P	P	P	P	P	A	P	P	P	P	P	P	P	P	P	P
24	159Y1A0525	C. Pavan Kumar Reddy	P	P	P	A	P	P	P	P	P	P	P	P	P	P	P	P
25	159Y1A0526	C. Lakshmi Priya .	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
26	159Y1A0527	C. Madhuri	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
27	159Y1A0528	C. Nagamani	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
28	159Y1A0529	C. Nikhitha	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
29	159Y1A0530	C. Harshavardhan Reddy	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
30	159Y1A0531	D. Anil Kumar Yadav	P	P	P	P	P	P	P	P	P	P	A	P	P	P	P	P
31	159Y1A0532	D. Sabareesh	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
32	159Y1A0533	D. Rajesh	P	P	P	P	P	P	A	P	P	P	P	P	P	P	P	P
33	159Y1A0534	D. Roja Rani	P	P	P	P	P	P	P	A	P	P	P	A	P	P	P	P
34	159Y1A0535	D. Yasaswini	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P	P
35	159Y1A0536	E. Sai Hanuman	P	P	P	A	P	P	P	P	P	P	P	P	P	P	P	P

Coordinator

HoD

Dr. M. Sreenivasulu,
M E, Ph. D.
Professer & HOD CSE
K.S.R.M. College of Engineering
KADAPA - 516 003

K.S.R.M. COLLEGE OF ENGINEERING

(UGC - Autonomous)

Kadapa, Andhra Pradesh, India– 516 003

Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.

LIGHTED TO LIGHTEN

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Course on

Python for everybody

from 10-09-2018
to 29-09-2019

Venue : MB 110

Coordinator : **Dr.V.Lokeshwara reddy**

Resource Person: **Mrs. B. Gouri,**
Mr. M.Purushothama

K.S.R.M. COLLEGE OF ENGINEERING

(AUTONOMOUS)

Pulivendala Road, Kadapa-516 005
Andhra Pradesh, India
Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.
An ISO 14001:2004 & 9001: 2015 Certified Institution

ACTIVITY REPORT

Certification Course

On

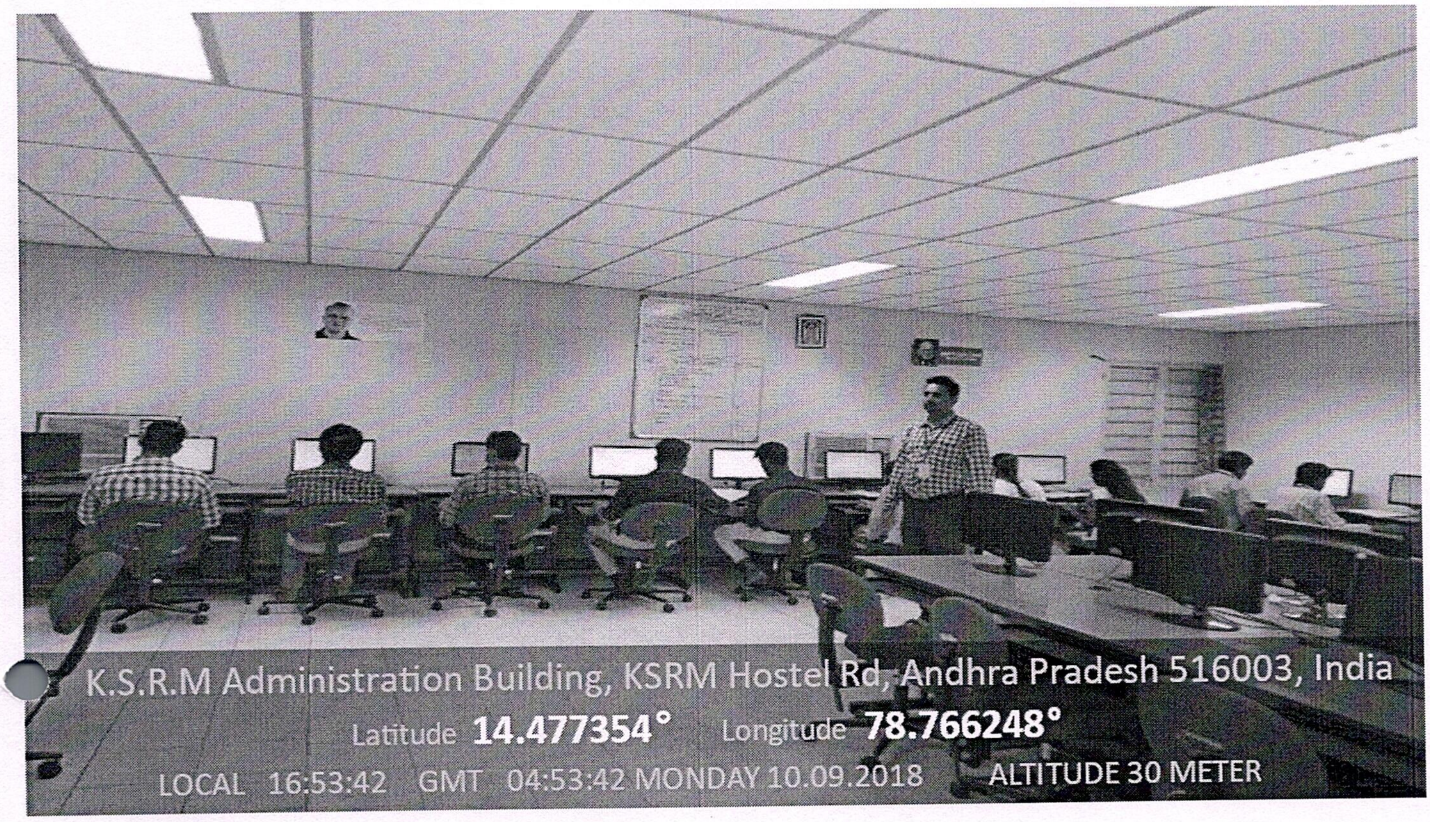
Python for Everybody

10/09/2018 to 29/09/2018

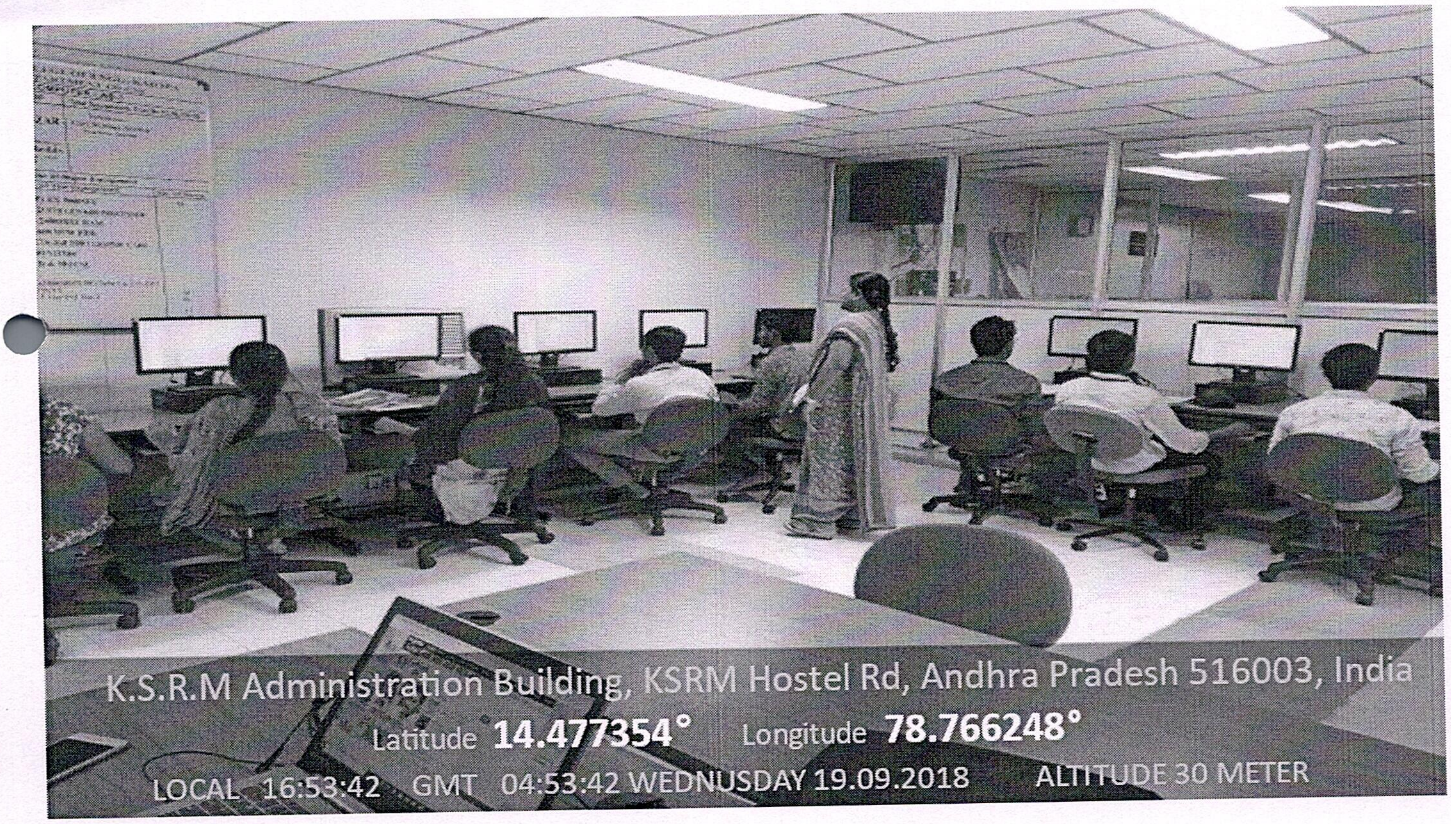
Target Group	:	**B. Tech Students**
Details of Participants	:	**35 Students**
Coordinator	:	**Dr. V. Lokeswara Reddy** **Professor, Dept. of CSE, KSRMCE**
Organizing Department	:	**Computer Science & Engineering**
Venue	:	**Programming in C Lab(MB 110)**

Description: Certification course on Python for Everybody" was organized by Dept. of CSE from 10/09/2018 to 29/09/2018 (4.00PM to 6.00PM). Dr. V. Lokeswara Reddy acted as Course Coordinator and Mrs. B. Gouri and Mrs. V. Sudha acted as resource persons. The course is designed to provide knowledge of Python Programming and to enhance the programming skills of the students by giving practical assignments to be done in labs. Thirty Eight hours course was successfully completed and participation certificates were provided to the participants.

Photo :


K.S.R.M Administration Building, KSRM Hostel Rd, Andhra Pradesh 516003, India
Latitude 14.477354° Longitude 78.766248°
LOCAL 16:53:42 GMT 04:53:42 MONDAY 10.09.2018 ALTITUDE 30 METER

Coordinator gives brief overview about Python for Everybody


K.S.R.M Administration Building, KSRM Hostel Rd, Andhra Pradesh 516003, India
Latitude 14.477354° Longitude 78.766248°
LOCAL 16:53:42 GMT 04:53:42 WEDNUSDAY 19.09.2018 ALTITUDE 30 METER

Resource Person clarify doubts in practical session

Students Participated in practical session

Coordinator

HoD

Dr. M. Sreenivasulu,
M. E., Ph. D.
Professer & HOD CSE
K.S.R.M. College of Engineering
KADAPA - 516003

K.S.R.M. COLLEGE OF ENGINEERING

(UGC - Autonomous)

Kadapa, Andhra Pradesh, India– 516 003

Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.

LIGHTED TO LIGHTEN

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE OF PARTICIPATION

This is to certify that Mr/Miss. A. Srikanth bearing Roll Number. 159Y1A0501 participated in a certification course on "**Python for Everybody**" organized by department of Computer Science and Engineering from 10-09-2018 to 29-09-2018.

V. S. S. Murty

COORDINATOR HOD PRINCIPAL

K.S.R.M. COLLEGE OF ENGINEERING

(UGC - Autonomous)
Kadapa, Andhra Pradesh, India– 516 003
Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.

LIGHTED TO LIGHTEN

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE OF PARTICIPATION

This is to certify that Mr/Miss. B. Ashok Vinay Kumar bearing Roll Number. 159Y1A0512 participated in a certification course on "**Python for Everybody**" organized by department of Computer Science and Engineering from 10-09-2018 to 29-09-2018.

V. S. S. Murty

COORDINATOR HOD PRINCIPAL

K.S.R.M. COLLEGE OF ENGINEERING

(UGC - Autonomous)
Kadapa, Andhra Pradesh, India– 516 003
Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE OF PARTICIPATION

This is to certify that Mr/Miss. B. Sireesha bearing Roll Number. 159Y1A0514 participated in a certification course on "**Python for Everybody**" organized by department of Computer Science and Engineering from 10-09-2018 to 29-09-2018.

V. S. S. Murty

COORDINATOR HOD PRINCIPAL

K.S.R.M. COLLEGE OF ENGINEERING

UGC - AUTONOMOUS

Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.
Kadapa, Andhra Pradesh, India– 516 003

FEEDBACK FORM

Certificate Course on "Python for Everybody", from 10/09/2018 to 29/09/2018
Organized
by
Department of Computer Science & Engineering

NAME:

Roll No:

S.No	Feedback Item	Excellent	Very Good	Good	Average	Below Average
1	Organization of certificate course and session planning by instructor.					
2	Clarity in content delivery.					
3	Content is relevant and useful.					
4	Adequate opportunity to interact with trainer.					
5	Judicious mix of concepts. Principles and practices.					
6	Assignments and tasks are interesting and challenging.					
7	Overall rating					

Any suggestions for improvement.

Signature

K.S.R.M. COLLEGE OF ENGINEERING

UGC - AUTONOMOUS

Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.
Kadapa, Andhra Pradesh, India– 516 003

FEEDBACK FORM

Certificate Course on "Python for Everybody", from 10/09/2018 to 29/09/2018
Organized
by
Department of Computer Science & Engineering

NAME: A. Venkata Bhavana

Roll No: 159Y1A0504

S.No	Feedback Item	Excellent	Very Good	Good	Average	Below Average
1	Organization of certificate course and session planning by instructor.	✓				
2	Clarity in content delivery.	✓				
3	Content is relevant and useful.	✓				
4	Adequate opportunity to interact with trainer.	✓				
5	Judicious mix of concepts. Principles and practices.	✓				
6	Assignments and tasks are interesting and challenging.	✓				
7	Overall rating	✓				

Any suggestions for improvement.

Need such type of courses, to improve our knowledge.

Bhavana

Signature

(A. Venkata Bhavana)

K.S.R.M. COLLEGE OF ENGINEERING

UGC - **AUTONOMOUS**

Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.
Kadapa, Andhra Pradesh, India– 516 003

FEEDBACK FORM

Certificate Course on "Python for Everybody", from 10/09/2018 to 29/09/2018
Organized
by
Department of Computer Science & Engineering

NAME: A. Lavanya

Roll No: 15941A0503

S.No	Feedback Item	Excellent	Very Good	Good	Average	Below Average
1	Organization of certificate course and session planning by instructor.		✓			
2	Clarity in content delivery.	✓				
3	Content is relevant and useful.	✓				
4	Adequate opportunity to interact with trainer.		✓			
5	Judicious mix of concepts. Principles and practices.	✓				
6	Assignments and tasks are interesting and challenging.	✓				
7	Overall rating	✓				

Any suggestions for improvement.

A. Lavanya
Signature

K.S.R.M. COLLEGE OF ENGINEERING

UGC - AUTONOMOUS

Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.
Kadapa, Andhra Pradesh, India– 516 003

FEEDBACK FORM

Certificate Course on "Python for Everybody", from 10/09/2018 to 29/09/2018
Organized
by
Department of Computer Science & Engineering

NAME: B· KRISHNA KISHORE

Roll No: 159Y1A0509

S.No	Feedback Item	Excellent	Very Good	Good	Average	Below Average
1	Organization of certificate course and session planning by instructor.	✓				
2	Clarity in content delivery.	✓				
3	Content is relevant and useful.	✓				
4	Adequate opportunity to interact with trainer.	✓				
5	Judicious mix of concepts. Principles and practices.	✓				
6	Assignments and tasks are interesting and challenging.	✓				
7	Overall rating	✓				

Any suggestions for improvement.

—

B·K
Signature

K.S.R.M. COLLEGE OF ENGINEERING

UGC - **AUTONOMOUS**

Approved by AICTE, New Delhi & Affiliated to JNTUA, Ananthapuramu.
Kadapa, Andhra Pradesh, India– 516 003

FEEDBACK FORM

Certificate Course on "Python for Everybody", from 10/09/2018 to 29/09/2018
Organized
by
Department of Computer Science & Engineering

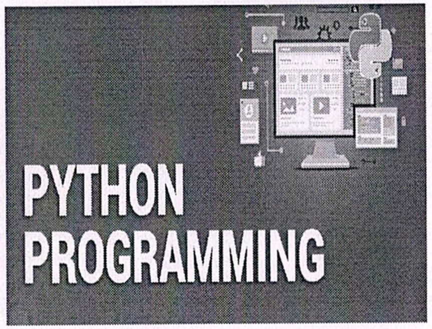
NAME: B·Sravani

Roll No: 159Y1A0510

S.No	Feedback Item	Excellent	Very Good	Good	Average	Below Average
1	Organization of certificate course and session planning by instructor.	✓				
2	Clarity in content delivery.	✓				
3	Content is relevant and useful.	✓				
4	Adequate opportunity to interact with trainer.		✓			
5	Judicious mix of concepts. Principles and practices.	✓				
6	Assignments and tasks are interesting and challenging.		✓			
7	Overall rating	✓				

Any suggestions for improvement.

B·Sravani
Signature



PYTHON PROGRAMMING

Python

- Python is an open source, cross-platform, general purpose programming language.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

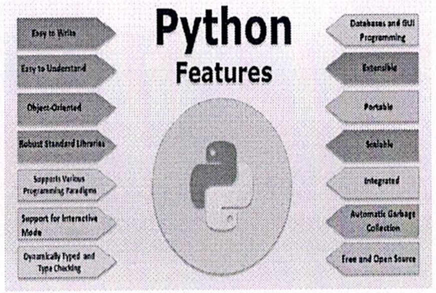
Applications of P[illegible]n

- Python is used to develop video players like **YouTube**, power apps like **Instagram**, run a search engine at **Google**

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.


Python
Features
Easy to Write
Easy to Understand
Object-Oriented
Robust Standard Libraries
Supports Various Programming Paradigms
Support for Interactive Mode
Dynamically Typed and Type Checking
Databases and GUI Programming
Extensible
Portable
Scalable
Integrated
Automatic Garbage Collection
Free and Open Source

ANACONDA

- **Anaconda is a distribution of Python and R programming languages for data science and Machine learning**
- **Install from**

  https://www.anaconda.com/

JUPYTER

- **Jupyter notebook is an open-source IDE.**
- It is a web-based interactive computational environment. The Jupyter notebook can support various languages that are popular in data science such as Python, Julia, Scala, R, etc

Python Comments

- Comments can be used to explain Python code.
- Comments can be used to make the code more readable.
- Comments can be used to prevent execution when testing code.
- Comments starts with a #, and Python will ignore them

Ex: #This is a comment

Output Statement:

- We use print() function to display output

Ex: print("Python")

Read Input from User

- **input ( ) :** This function first takes the input from the user and convert it into string. Type of the returned object always will be <type 'str'>. It does not evaluate the expression it just return the complete statement as String.

Ex:

```
val = input("Enter your value: ")
print(val)
```

Python Variables

- A variable is created the moment you first assign a value to it.

Ex:

```
x = 5
y = "IBM"
print(x)
print(y)
```

Variables

- Variables do not need to be declared with any particular *type*, and can even change type after they have been set.

Ex:

```
x = 10      # x is of type int
x = "IBM"  # x is now of type str
print(x)
```

Casting VARIABLES

- We can specify the data type of a variable.

Ex:

```
x = str(10)    # x will be '10'
y = int(10)    # y will be 10
z = float(10)  # z will be 10.0
```

Multiple Assignment

Python allows you to assign a single value to several variables simultaneously.

For example:

```
a = b = c = 1
a,b,c = 1,2,"john"
```

Variables are **Case-Sensitive**

- Variable names are case-sensitive.

```
a = 10
A = "IBM"

# A will not overwrite a
```

Python Data Types

- Python has the following data types built-in by default, in these categories:
- Text Type: str
- Numeric Types: int, float, complex
- Sequence Types: list, tuple, range
- Mapping Type: dict
- Set Types: set, frozenset
- Boolean Type: bool
- Binary Types: bytes, bytearray, memoryview

Python Strings

- Strings in python are surrounded by either single quotation marks, or double quotation marks.
- 'hello' is the same as "hello".
- Ex:

```
print("Hello")
print('Hello')
```

Multiline Strings

- You can assign a multiline string to a variable by using three quotes:

Ex:

```
a = """I am a Beginner in PYTHON,
    Machine Learning,
    Data Science."""

print(a)
```

Sub String

- b = "Hello, World!"
  print(b[2:5]) O/P: llo (begin with index – length)

- **Slice From the Start**
  b = "Hello, World!"
  print(b[:5]) O/P: Hello

- **Slice To the End**
  b = "Hello, World!"
  print(b[2:]) O/P: llo, World!

- **Upper Case**
  a = "Hello, World!"
  print(a.upper())

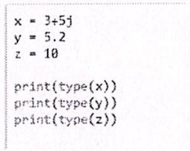
- **Lower Case**
  a = "Hello, World!"
  print(a.lower())

- **Remove Whitespace**
  a = " Hello, World! "
  print(a.strip()) # returns "Hello, World!"

- **Replace String (All Chars)**
- a = "Hello, World!"
  print(a.replace("H", "J"))

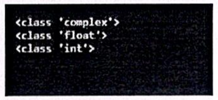
- **String Concatenation**
  a = "Hello"
  b = "World"
  c = a + b
  print(c)

Numeric Data Types:



```
x = 3+5j
y = 5.2
z = 10

print(type(x))
print(type(y))
print(type(z))
```

Output:



```
<class 'complex'>
<class 'float'>
<class 'int'>
```

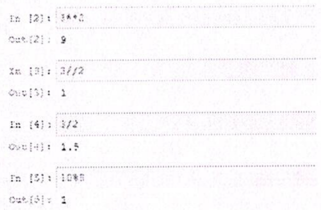
Python Operators

- Arithmetic operators
- Assignment operators
- Comparison operators
- Logical operators
- Identity operators
- Membership operators
- Bitwise operators

Python Arithmetic Operators

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Examples:



```
In [2]: 3**2
Out[2]: 9

In [3]: 3//2
Out[3]: 1

In [4]: 3/2
Out[4]: 1.5

In [5]: 10%3
Out[5]: 1
```

Exercise

1. Read 4 Subjects marks and display total and average
2. Read X value from user and display X^3

Python Assignment Operators

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3

Python Comparison Operators

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Python Logical Operators

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and x < 10
or	Returns True if one of the statements is true	x < 5 or x < 4
not	Reverse the result, returns False if the result is true	not(x < 5 and x < 10)

Python Identity Operators

Operator	Description	Example
is	Returns True if both variables are the same object	x is y
is not	Returns True if both variables are not the same object	x is not y

Python Membership Operators

Operator	Description	Example
in	Returns True if a sequence with the specified value is present in the object	x in y
not in	Returns True if a sequence with the specified value is not present in the object	x not in y

Python Bitwise Operators

Operator	Name	Description
&	AND	Sets each bit to 1 if both bits are 1
|	OR	Sets each bit to 1 if one of two bits is 1
^	XOR	Sets each bit to 1 if only one of two bits is 1
~	NOT	Inverts all the bits
<<	Zero fill left shift	Shift left by pushing zeros in from the right and let the leftmost bits fall off
>>	Signed right shift	Shift right by pushing copies of the leftmost bit in from the left, and let the rightmost bits fall off

Conditional Statements

- Simple IF:

```
a = 33
b = 200

if b > a:
  print("b is greater than a")
```

Indentation

- Python relies on indentation (whitespace at the beginning of a line) to define scope in the code. Other programming languages often use curly-brackets for this purpose.

```
a = 33
b = 200
if b > a:
print("b is greater than a") # you will get an error
```

IF ... ELSE

```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

ELSE

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

Short Hand If ... Else

- If you have only one statement to execute, one for if, and one for else, you can put it all on the same line:

```
a = 2
b = 330
print("A") if a > b else print("B")
```

Nested If

- You can have if statements inside if statements, this is called *nested* if statements.

```
x = 41

if x > 10:
  print("Above ten,")
  if x > 20:
    print("and also above 20!")
  else:
    print("but not above 20.")
```

Python Loops

- Python has two primitive loop commands:
  1. while loop
  2. for loop

while Loop

- With the while loop we can execute a set of statements as long as a condition is true.

```
i = 1
while i < 6:
  print(i)
  i += 1
```

break Statement

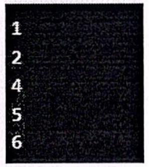
- With the break statement we can stop the loop even if the while condition is true:

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

continue Statement

- With the continue statement we can stop the current iteration, and continue with the next iteration.

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```



```
1
2
4
5
6
```

For Loop

- A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  print(x)
```

The range() Function

- The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and ends at a specified number.

```
for x in range(6):
  print(x)
```

```
for x in range(2, 6):
  print(x)
```

Step variable

- The range() function defaults to increment the sequence by 1, however it is possible to specify the increment value by adding a third parameter: range(2, 30, **3**):

```
for x in range(2, 30, 3):
  print(x)
```

Programs

1. Program to check whether given number is even or odd
2. Program to count no.of even and odd numbers in range 50 to 200
3. Python program that prints all the numbers from 1 to 20 except those divisible by 3.
4. Write a Python program to get the Fibonacci series between 0 to 50 Ex: 0, 1, 1, 2, 3, 5, 8, 13, 21, ...

5. Write a Python program to check whether an alphabet is a vowel or consonant.

6.

Python Collections (Arrays)

There are four collection data types in the Python programming language:

- **List:** is a collection which is ordered and changeable. Allows duplicate members.
- **Tuple:** is a collection which is ordered and unchangeable. Allows duplicate members.
- **Set:** is a collection which is unordered, unchangeable. No duplicate members.
- **Dictionary:** is a collection which is ordered and changeable. No duplicate members.

Python Lists

- Lists are used to store multiple items in a single variable. List items are ordered, changeable, and allow duplicate values.
- List items are indexed, the first item has index [0], the second item has index [1] etc.

```
thislist = ["apple", "banana", "cherry"]
print(thislist)
```

List Length

- To determine how many items a list has, use the len() function:

  Ex:

  len(thislist)

```
thislist = ["apple", "banana", "cherry"]
print(len(thislist))
```

The list() Constructor

- It is also possible to use the list() constructor when creating a new list.

Ex:

```
thislist = list(("apple", "banana", "cherry"))

# note the double round-brackets
print(thislist)
```

Access List Items

```
thislist = ["apple", "banana", "cherry"]

print(thislist[1])
print(thislist[2:3])
```

Check if Item Exists

- To determine if a specified item is present in a list use the "in" keyword:

-
  ```
  thislist = ["apple", "banana", "cherry"]
  if "apple" in thislist:
    print("Yes, 'apple' is in the fruits list")
  ```

Change List Items

-
  ```
  thislist = ["apple", "banana", "cherry"]
  thislist[1] = "blackcurrant"
  print(thislist)
  ```

-
  ```
  thislist = ["apple", "banana", "cherry", "orange",
  "kiwi", "mango"]
  thislist[1:3] = ["blackcurrant", "watermelon"]
  print(thislist)
  ```

Append List Items

- To add an item to the end of the list, use the append() method:

Ex:

```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

Insert List Items

- To insert a list item at a specified index, use the insert() method.
- The insert() method inserts an item at the specified index:

Ex:

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(1, "orange")
print(thislist)
```

Extend List

- To append elements from *another list* to the current list, use the extend() method.

Ex:

```
thislist = ["apple", "banana", "cherry"]
tropical = ["mango","pineapple","papaya"]
thislist.extend(tropical)
print(thislist)
```

Remove List Items

- The remove() method removes the specified item.

Ex:

```
thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)
```

Remove Specified Index

- The pop() method removes the specified index.

Ex:

```
thislist = ["apple", "banana", "cherry"]
thislist.pop(1)
print(thislist)
```

Sort Lists

- List objects have a sort() method that will sort the list alphanumerically, ascending, by default:

Ex:

```
thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
thislist.sort()
print(thislist)
```

Sort Descending - List

- To sort descending, use the keyword argument reverse = True:

Ex:

```
thislist = ["orange", "mango", "kiwi", "pineapple", "banana"]
thislist.sort(reverse = True)
print(thislist)
```

Reverse Order - List

- The reverse() method reverses the current sorting order of the elements.

Ex:

```
thislist = ["banana", "Orange", "Kiwi", "cherry"]
thislist.reverse()
print(thislist)
```

Copy Lists

- You cannot copy a list simply by typing list2 = list1, because: list2 will only be a *reference* to list1, and changes made in list1 will automatically also be made in list2.
- to make a copy, one way is to use the built-in List method copy().
- Ex:

```
thislist = ["apple", "banana", "cherry"]
mylist = thislist.copy()
print(mylist)
```

List Methods

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Programs

1. Python program to interchange first and last elements in a list
2. Python program to swap two elements in a list
3. Reversing a List
4. Count occurrences of an element in a list
5. Multiply all numbers in the list
6. Python program to find largest number in a list

Python Tuples

- A tuple is a collection which is ordered and **unchangeable.**
- Tuples are written with round brackets.

```
thistuple = ("apple", "banana", "cherry")
print(thistuple)
```

Tuple Items

- Tuple items are ordered, unchangeable, and allow duplicate values.
- Tuple items are indexed, the first item has index [0], the second item has index [1] etc.
- When we say that tuples are ordered, it means that the items have a defined order, and that order will not change.
- Tuples are unchangeable, meaning that we cannot change, add or remove items after the tuple has been created.

Create a Tuple

- A tuple is a collection which is ordered and **unchangeable.**
- Tuples are written with round brackets.

```
tup = ("CSE", "ECE", "EEE")
print(tup)
```

Tuple Length

- To determine how many items a tuple has, use the len() function:

```
tup = ("CSE", "ECE", "EEE")
print(len(tup))
```

The tuple() Constructor

- It is also possible to use the tuple() constructor to make a tuple.

```
tup = tuple(("CSE", "ECE", "EEE"))
print(tup)
```

Negative Indexing

- Negative indexing means start over from the end.
- -1 refers to the last item, -2 refers to the second last item etc.

```
tup = tuple(("CSE", "ECE", "EEE"))
print(tup[-1])
```

```
EEE
```

Change Tuple Values

- Once a tuple is created, you cannot change its values. Tuples are **unchangeable**, or **immutable** as it also is called.
- You can convert the tuple into a list, change the list, and convert the list back into a tuple.

```
x = ("CSE", "ECE", "EEE")
y = list(x)
y[1] = "Mech"
x = tuple(y)

print(x)
```

Add Items

- Since tuples are immutable, they do not have a build-in append() method, so convert tuple into list and add item.

```
thistuple = ("CSE", "ECE", "EEE")
y = list(thistuple)
y.append("Mech")
thistuple = tuple(y)

print(thistuple)
```

Add tuple to a tuple

- create a new tuple with the item(s), and add it to the existing tuple:

```
tup = ("CSE", "ECE", "EEE")
y = ("Mech",)
tup += y

print(tup)
```

Remove Items

- Tuples are **unchangeable,** so you cannot remove items from it. So, convert to list and remove item

```
tup = ("CSE", "ECE", "EEE")
y = list(tup)
y.remove("EEE")
tup = tuple(y)

print(tup)
```

Loop Tuples

- You can loop through the tuple items by using a for loop.

```
tup = ("CSE", "ECE", "EEE")
for x in tup:
  print(x)
```

Loop Through the Index Numbers

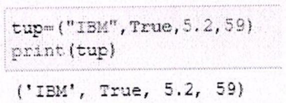
```
tup = ("CSE", "ECE", "EEE")
for i in range(len(tup)):
  print(tup[i])
```

Using a While Loop

```
tup = ("CSE", "ECE", "EEE")
i = 0
while i < len(tup):
  print(tup[i])
  i = i + 1
```

Programs on Tuple

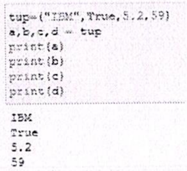
1. Python program to create a tuple with different data types



```
tup=("IBM",True,5.2,59)
print(tup)
```

```
('IBM', True, 5.2, 59)
```

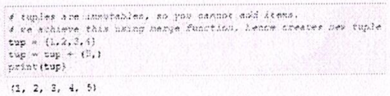
2. Python program to assign tuple values to multiple variables



```
tup=("IBM",True,5.2,59)
a,b,c,d = tup
print(a)
print(b)
print(c)
print(d)
```

```
IBM
True
5.2
59
```

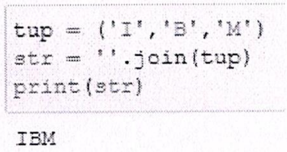
3. Add an item in a tuple (without list convertion)



```
# tuples are immutables, so you cannot add items.
# we achieve this using merge function, hence creates new tuple
tup = (1,2,3,4)
tup = tup + (5,)
print(tup)
```

```
(1, 2, 3, 4, 5)
```

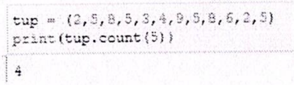
4. Python pogram to convert tuple to string



```
tup = ('I','B','M')
str = ''.join(tup)
print(str)
```

```
IBM
```

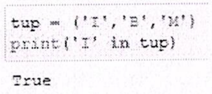
5. Python program to find repeated item in tuple



```
tup = (2,5,8,5,3,4,9,5,8,6,2,5)
print(tup.count(5))
```

```
4
```

6. Python program to check whether an item exists in tuple

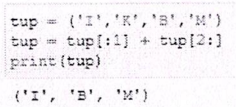


```
tup = ('I','B','M')
print('I' in tup)
```

```
True
```

7. Program to convert list to tuple

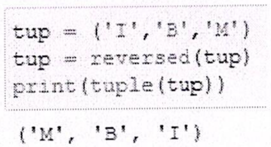
8. Python program to remove an item from tuple



```
tup = ('I','K','B','M')
tup = tup[:1] + tup[2:]
print(tup)
```

```
('I', 'B', 'M')
```

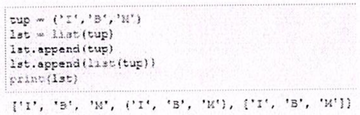
9. Program to reverse a tuple



```
tup = ('I','B','M')
tup = reversed(tup)
print(tuple(tup))
```

```
('M', 'B', 'I')
```

10. Program to add list and tuple to a list



```
tup = ('I','B','M')
lst = list(tup)
lst.append(tup)
lst.append(list(tup))
print(lst)
```

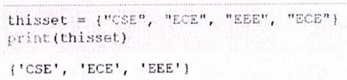
```
['I', 'B', 'M', ('I', 'B', 'M'), ['I', 'B', 'M']]
```

Python Sets

- A set is a collection which is *unordered, unchangeable*, and *unindexed.*
- Set *items* are unchangeable, but you can remove items and add new items.

```
myset = {"CSE", "ECE", "EEE"}
print(myset)

# Note: the set list is unordered, meaning: the items will appear in a random order.
```

- Set items in a set do not have a defined order.
- Set items can appear in a different order every time you use them, and cannot be referred to by index or key.
- Sets cannot have two items with the same value.

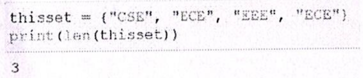


```
thisset = {"CSE", "ECE", "EEE", "ECE"}
print(thisset)
```

```
{'CSE', 'ECE', 'EEE'}
```

Length of a Set

- Use len() function to get length of a set

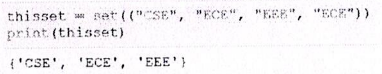


```
thisset = {"CSE", "ECE", "EEE", "ECE"}
print(len(thisset))
```

```
3
```

set() Constructor

- It is also possible to use the set() constructor to make a set.

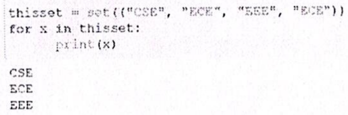


```
thisset = set(("CSE", "ECE", "EEE", "ECE"))
print(thisset)
```

```
{'CSE', 'ECE', 'EEE'}
```

Access set items

- We cannot access items in a set by referring to an index or a key.
- Use for loop to access items in a set

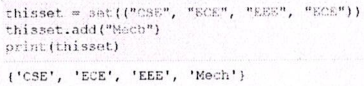


```
thisset = set(("CSE", "ECE", "EEE", "ECE"))
for x in thisset:
      print(x)
```

```
CSE
ECE
EEE
```

Add new items

- Once a set is created, you cannot change its items, but you can add new items.



```
thisset = set(("CSE", "ECE", "EEE", "ECE"))
thisset.add("Mech")
print(thisset)
```

```
{'CSE', 'ECE', 'EEE', 'Mech'}
```

Add Sets

- To add items from another set into the current set, use the update() method.

```
set1 = set(("CSE", "ECE", "EEE", "ECE"))
set2 = set(("Mech", "CE"))
set1.update(set2)
print(set1)
```

```
{'CSE', 'ECE', 'Mech', 'CE', 'EEE'}
```

Remove Set Item

- User remove() function to remove an item from set

```
set1 = set(("CSE", "ECE", "EEE", "ECE"))
set1.remove("EEE")
print(set1)
```

```
{'CSE', 'ECE'}
```

Programs on Sets

1. Program to create a set

```
st = {1,2,'IBM'}
print(st)
```

```
{1, 2, 'IBM'}
```

2. Program to add items to a set

```
st = {1,2}
print(st)
# add a single value
st.add("IBM")
print(st)
# add multiple values
st.update([1,2,3])
print(st)
```

```
{1, 2}
{1, 2, 'IBM'}
{3, 1, 2, 'IBM'}
```

3. Program to create a union of sets

```
st1 = {1,2}
st2 = {2,3}
st3 = {1,5}
st = st1.union(st2)
print(st)
st = st.union(st3)
print(st)
```

```
{1, 2, 3}
{1, 2, 3, 5}
```

4. Program to create intersection of sets

```
st1 = {1,2,5,6}
st2 = {2,3,5,8}
st = st1 & st2
print(st)
```

```
{2, 5}
```

5. Program to create set difference

```
st1 = {1,2,5,6}
st2 = {2,3,5,8}
st = st1.difference(st2)
print(st)
```

```
{1, 6}
```

Python Dictionaries

- Dictionaries are used to store data values in key:value pairs.
- A dictionary is a collection which is ordered, changeable and do not allow duplicates.

```
mydict = {
  "brand": "TATA",
  "model": "Altroz",
  "year": 2020
}
print(mydict)
```

```
{'brand': 'TATA', 'model': 'Altroz', 'year': 2020}
```

Duplicates Not Allowed

- Dictionaries cannot have two items with the same key.
- Duplicate values will overwrite existing values

```
mydict = {
  "brand": "TATA",
  "model": "Altroz",
  "year": 2020,
  "year": 2021
}
print(mydict)
```

```
{'brand': 'TATA', 'model': 'Altroz', 'year': 2021}
```

Dictionary Length

- To determine how many items a dictionary has, use the len() function:
- Ex: print(len(thisdict))

Access Dict items

```
mydict = {
  "brand": "TATA",
  "model": "Altroz",
  "year": 2020,
  "year": 2021
}
x = mydict["model"]
print(x)
```

```
Altroz
```

Change Values

- You can change the value of a specific item by referring to its key name:

```
mydict = {
  "brand": "TATA",
  "model": "Altroz",
  "year": 2020
}
mydict["year"] = "2021"
print(mydict)
```

```
{'brand': 'TATA', 'model': 'Altroz', 'year': '2021'}
```

Add New Item

```
mydict = {
  "brand": "TATA",
  "model": "Altroz",
  "year": 2020
}
mydict["color"] = "red"
print(mydict)
```

```
{'brand': 'TATA', 'model': 'Altroz', 'year': 2020, 'color': 'red'}
```

Remove an Item from Dict

- Use pop() method to remove item

```
mydict = {
  "brand": "TATA",
  "model": "Altroz",
  "year": 2020
}
mydict.pop("model")
print(mydict)
```

```
{'brand': 'TATA', 'year': 2020}
```

Access using For loop

```
mydict = {
  "brand": "TATA",
  "model": "Altroz",
  "year": 2020
}
for x in mydict:
     print(x)
```

```
brand
model
year
```

Copy a Dictionary

- use the built-in Dictionary method copy().

```
mydict = {
  "brand": "TATA",
  "model": "Altroz",
  "year": 2020
}
dict2 = mydict.copy()
print(dict2)
```

```
{'brand': 'TATA', 'model': 'Altroz', 'year': 2020}
```

Programs on Dictionary

1. Python program to check whether the key already exists in dictionary

```
d = {'a': 1, 'b': 2, 'c':3, 'd':4}
if 'a' in d:
    print("Key is present")
else:
    print("Key is not present")
```

```
Key is present
```

2. Python program to iterate over dictionaries

```
d = {'a': 1, 'b': 2, 'c':3, 'd':4}
for dict_key, dict_value in d.items():
    print(dict_key,':',dict_value)
```

```
a : 1
b : 2
c : 3
d : 4
```

3. Python program to remove a key from dictionary

```
d = {'a': 1, 'b': 2, 'c':3, 'd':4}
print(d)
if 'a' in d:
    del d['a']
print(d)
```

```
{'a': 1, 'b': 2, 'c': 3, 'd': 4}
{'b': 2, 'c': 3, 'd': 4}
```

4. Python program to remove duplicates

```
d = {'a': 1, 'b': 2, 'c':3, 'd':4, 'e':1, 'f':2 }
print(d)
result = {}
for key,value in d.items():
    if value not in result.values():
        result[key] = value
print(result)
```

```
{'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 1, 'f': 2}
{'a': 1, 'b': 2, 'c': 3, 'd': 4}
```

5. Python program to map two lists into a dictionary

```
keys = ["sid","sname","dept"]
vals = ["520","Kumar","AI ML"]
student = dict(zip(keys,vals))
print(student)
```

```
{'sid': '520', 'sname': 'Kumar', 'dept': 'AI ML'}
```

Python Functions

- A function is a sub program or block of statements with a name called function name.
- A function gets executed only when it is called from somewhere in the program.

```
def funct():
  print("Hello from a function")

funct()
```

```
Hello from a function
```

Function Parameter and Arguments

- A parameter is the variables listed inside the parentheses in the function definition.
- An argument is the values that are sent to the function when it is called.

```
def funct(a,b):
  print(a+b)

funct(2,3)
```

```
5
```

Arbitrary Arguments, *args

- If you do not know how many arguments that will be passed into your function, add a * before the parameter name in the function definition.

```
def funct(*params):
  print(params[0]+params[1])

funct(10,15)
```

```
25
```

Passing a List as an Argument

- You can send any data types of argument to a function (string, number, list, dictionary etc.), and it will be treated as the same data type inside the function.

```
def funct(lst):
  for x in lst:
    print(x)

depts = ["CSE","ECE","EEE"]
funct(depts)
```

```
CSE
ECE
EEE
```

Return Values

- A Function can also return a value as and when required. Use return keyword to return value from function.

```
def funct(val):
    res = 5 * val
    return res

funct(10)
```

```
50
```

Programs on Functions

1. Write a program using functions to perform arithmetic operations based on user choice
2. Write a function to sum all numbers in a list
3. Write a function to calculate factorial of given number
4. Write function to find given string is palindrome or not
5. Write a function to find given number is prime or not

Python Classes and Objects

- Python is an object oriented programming language.
- Almost everything in Python is an object, with its properties and methods.
- A Class is like an object constructor, or a "blueprint" for creating objects.

Create a Class

- Use class keyword to create a class

```
class MyClass:
  x = 5
```

Create Object

- An Object is a variable of class. Use class name to create an object

```
class MyClass:
  x = 5
obj = MyClass()
```

Access Members of a class

- Use . Operator to access members of a class

```
class MyClass:
  x = 5
obj = MyClass()
print(obj.x)
```

```
5
```

The __init__() Function

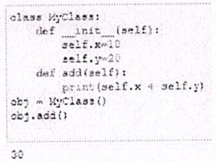
- The __init__() function is called automatically every time the class is being used to create a new object.
- Use this function to initialize values to variables

```
class MyClass:
     def __init__(self, x, y):
         print(x+y)
obj = MyClass(1,2)
```

```
3
```

Self parameter

- The self parameter is a reference to the current instance of the class, and is used to access variables that belong to the class.



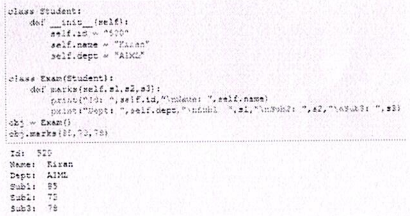
```
class MyClass:
    def __init__(self):
        self.x=10
        self.y=20
    def add(self):
        print(self.x + self.y)
obj = MyClass()
obj.add()
```

```
30
```

Python Inheritance

- Inheritance allows us to define a class that inherits all the methods and properties from another class.
- **Parent class** is the class being inherited from, also called base class.
- **Child class** is the class that inherits from another class, also called derived class.

Inheritance



```
class Student:
    def __init__(self):
        self.id = "520"
        self.name = "Kiran"
        self.dept = "AIML"

class Exam(Student):
    def marks(self,s1,s2,s3):
        print("Id: ",self.id,"\nName: ",self.name)
        print("Dept: ",self.dept,"\nSub1: ",s1,"\nSub2: ",s2,"\nSub3: ",s3)
obj = Exam()
obj.marks(85,73,78)
```

```
Id:  520
Name:  Kiran
Dept:  AIML
Sub1:  85
Sub2:  73
Sub3:  78
```

Libraries in Python

- Pandas
- Numpy
- Matplotlib

Pandas

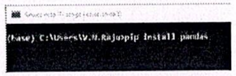
- Pandas is a Python library used for working with data sets.
- It has functions for analyzing, cleaning, exploring, and manipulating data.
- The name "Pandas" has a reference to both "Panel Data", and "Python Data Analysis"

Why Pandas?

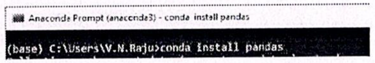
- Pandas allows us to analyze big data and make conclusions based on statistical theories.
- Pandas can clean messy data sets, and make them readable and relevant.
- Relevant data is very important in data science.

Installation of Pandas

- Open ANACONDA prompt and type in command:



```
(base) C:\Users\V.N.Raju>pip install pandas
```



```
(base) C:\Users\V.N.Raju>conda install pandas
```

Pandas Data Structures

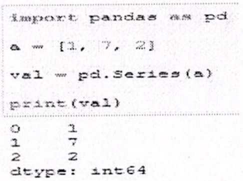
- Pandas has two main data structures for data storage:

1. Series
2. DataFrame

Series

- A series is similar to a one-dimensional array. It can store data of any type. The values of a Pandas Series are mutable but the size of a Series is immutable and cannot be changed.
- The first element in the series is assigned the index 0, while the last element is at index N-1, where N is the total number of elements in the series.

- To create a Pandas Series, we must first import the Pandas package via the Python's import command:

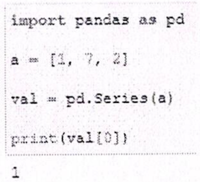


```
import pandas as pd

a = [1, 7, 2]

val = pd.Series(a)

print(val)
```

```
0    1
1    7
2    2
dtype: int64
```

Labels

- This label can be used to access a specified value.

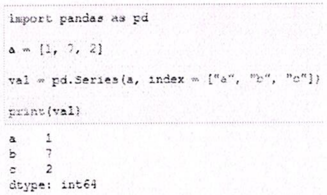


```
import pandas as pd

a = [1, 7, 2]

val = pd.Series(a)

print(val[0])
```

```
1
```

Create Labels

- We can create labels with the index argument, you can name your own labels.

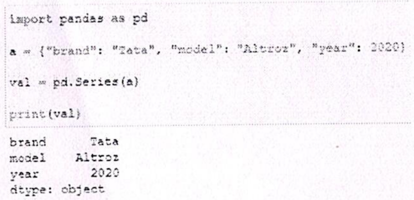


```
import pandas as pd

a = [1, 7, 2]

val = pd.Series(a, index = ["a", "b", "c"])

print(val)
```

```
a    1
b    7
c    2
dtype: int64
```

Key:value pairs as series

- We can also make series using key value pair

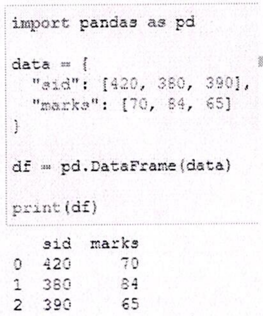


```
import pandas as pd

a = {"brand": "Tata", "model": "Altroz", "year": 2020}

val = pd.Series(a)

print(val)
```

```
brand      Tata
model    Altroz
year       2020
dtype: object
```

Data Frames

- A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.

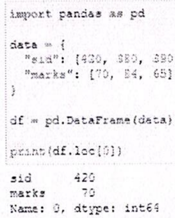


```
import pandas as pd

data = {
  "sid": [420, 380, 390],
  "marks": [70, 84, 65]
}

df = pd.DataFrame(data)

print(df)
```

```
   sid  marks
0  420     70
1  380     84
2  390     65
```

Locate Row

- Pandas use the loc attribute to return one or more specified row(s)



```
import pandas as pd

data = {
  "sid": [420, 380, 390],
  "marks": [70, 84, 65]
}

df = pd.DataFrame(data)

print(df.loc[0])
```

```
sid      420
marks     70
Name: 0, dtype: int64
```

- When using [ ], the result is a Pandas **DataFrame.**

```
import pandas as pd

data = {
  "sid": [420, 380, 390],
  "marks": [70, 84, 65]
}

df = pd.DataFrame(data)

print(df.loc[[0]])
```

```
   sid  marks
0  420     70
```

Named Indexes

- With the index argument, you can name your own indexes.

```
import pandas as pd

data = {
  "sid": [420, 380, 390],
  "marks": [70, 84, 65]
}

df = pd.DataFrame(data,index=["Sub1","Sub2","Sub3"])

print(df)
```

```
      sid  marks
Sub1  420     70
Sub2  380     84
Sub3  390     65
```

Load Files Into a DataFrame

- If your data sets are stored in a file, Pandas can load them into a DataFrame.
- Create a CSV file using MS Excel

```
import pandas as pd

df = pd.read_csv('data.csv')

print(df)
```

max_rows

- The number of rows returned is defined in Pandas option settings.
- We can check your system's maximum rows with the pd.options.display.max_rows statement.
- The default number is 60 rows, which means that if the DataFrame contains more than 60 rows, the print(df) statement will return only the headers and the first and last 5 rows.

- You can change the maximum rows number with the same statement.

```
import pandas as pd

pd.options.display.max_rows = 9999
df = pd.read_csv('data.csv')

print(pd)
```

Read JSON

- Big data sets are often stored, or extracted as JSON.
- JSON is plain text, but has the format of an object.

```
import pandas as pd

df = pd.read_json('data.json')

print(pd)
```

Viewing the Data

- One of the most used method for getting a quick overview of the DataFrame, is the head() method.
- The head() method returns the headers and a specified number of rows, starting from the top.

```
import pandas as pd

df = pd.read_csv('data.csv')

print(df.head(5))
```

```
   Sl no  Student Id   Name  Marks
0      1         501  Vinay     78
1      2         502  Kiran     79
2      3         503   Ravi     80
3      4         504   Siva     81
4      5         505   Mani     82
```

- There is also a tail() method for viewing the *last* rows of the DataFrame.
- The tail() method returns the headers and a specified number of rows, starting from the bottom.

```
import pandas as pd

df = pd.read_csv('data.csv')

print(df.tail())
```

```
   Sl no  Student Id    Name  Marks
5      6         506  Harish     83
6      7         507   Satya     84
7      8         508   Karna     85
8      9         509  Dinesh     86
9     10         510   Kumar     87
```

Pandas - Cleaning Data

Data cleaning means fixing bad data in your data set.

Bad data could be:

1. Empty cells
2. Data in wrong format
3. Wrong data
4. Duplicates

Pandas - Cleaning Empty Cells

- Empty cells can potentially give you a wrong result when you analyze data.
- One way to deal with empty cells is to remove rows that contain empty cells.
- since data sets can be very big, and removing a few rows will not have a big impact on the result.

Remove Empty rows

- By default, the dropna() method returns a *new* DataFrame, and will not change the original.

```
import pandas as pd

df = pd.read_csv('data.csv')
new_df = df.dropna()

print(new_df)
```

- If you want to change the original DataFrame, use the inplace = True argument:

```
import pandas as pd

df = pd.read_csv('data.csv')
df.dropna(inplace = True)


print(new_df)
```

Replace Empty Values

- Another way of dealing with empty cells is to insert a *new* value instead.
- This way you do not have to delete entire rows just because of some empty cells.
- The fillna() method allows us to replace empty cells with a value:

```
import pandas as pd

df = pd.read_csv('data.csv')
df.fillna(130, inplace = True)


print(new_df)
```

Replace Only For Specified Columns

- To only replace empty values for one column, specify the *column name* for the DataFrame:

```
import pandas as pd

df = pd.read_csv('data.csv')
df["Marks"].fillna(35, inplace=True)


print(df)
```

Using Mean, Median, or Mode

- A common way to replace empty cells, is to calculate the mean, median or mode value of the column.
- Pandas uses the mean() median() and mode() methods to calculate the respective values for a specified column:
- **Mean** = the average value (the sum of all values divided by number of values).
- **Median** = the value in the middle, after you have sorted all values ascending.
- **Mode** = the value that appears most frequently.

```
import pandas as pd

df = pd.read_csv('data.csv')
x = df["Marks"].mean()
y = df["Marks"].median()
z = df["Marks"].mode()
df["Marks"].fillna(x, inplace=True)


print(df)
```

Removing Duplicates

- Duplicate rows are rows that have been registered more than one time.
- To discover duplicates, we can use the duplicated() method.
- The duplicated() method returns a Boolean values for each row:

```
import pandas as pd

df = pd.read_csv('data.csv')

print(df.duplicated())

0      False
1      False
```

- To remove duplicates, use the drop_duplicates() method.

```
import pandas as pd

df = pd.read_csv('data.csv')
df.drop_duplicates(inplace = True)

print(df)
```

Pandas - Data Correlations

Finding Relationships:

- A great aspect of the Pandas module is the corr() method.
- The corr() method calculates the relationship between each column in your data set.

```
import pandas as pd

df = pd.read_csv('data.csv')
print(df.corr())

                Sl no  Student Id
Sl no        1.000000   -0.018847
Student Id  -0.018847    1.000000
```

- The Result of the corr() method is a table with a lot of numbers that represents how well the relationship is between two columns.
- The number varies from -1 to 1.
- 1 means that there is a 1 to 1 relationship (a perfect correlation), and for this data set, each time a value went up in the first column, the other one went up as well.
- 0.9 is also a good relationship, and if you increase one value, the other will probably increase as well.
- -0.9 would be just as good relationship as 0.9, but if you increase one value, the other will probably go down.
- 0.2 means NOT a good relationship, meaning that if one value goes up does not mean that the other will.

What is a good correlation?

- It depends on the use, but I think it is safe to say you have to have at least 0.6 (or -0.6) to call it a good correlation.

Plotting

- Pandas uses the plot() method to create diagrams.
- We can use Pyplot, a submodule of the Matplotlib library to visualize the diagram on the screen.

```
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('data.csv')
df.plot()

print(plt)
```

Error Handling

- There are mainly three kinds of distinguishable errors in Python: **syntax errors, exceptions and logical errors.**

Syntax Error

- **Syntax errors** are Missing symbols (such as comma, bracket, colon), misspelling a keyword, having incorrect indentation are common **syntax errors** in Python.

```
# Syntax Error-1: Misusing the Assignment Operator
len("data") = 4
Output:
  File "ErrorsAndExceptions.py", line 1
    len("data") = 4
SyntaxError: can't assign to function call
```

Exception

2. **Exceptions** may occur in syntactically correct code blocks at run time. When Python cannot execute the requested action, it terminates the code and raises an error message.

```
# Exception-1: ZeroDivisionError
print (5/0)
Output:
Traceback (most recent call last):
  File "ErrorsAndExceptions.py", line 12, in <module>
    print (5/0)
ZeroDivisionError: integer division or modulo by zero
```

Logical Error

3. **Logical errors** are the most difficult errors to fix as they don't crash your code and you don't get any error message.If you have logical errors, your code does not run as you expected.Using incorrect variable names, code that is not reflecting the algorithm logic properly, making mistakes on boolean operators will result in **logical errors.**

Regression analysis in Python

- The term regression is used when you try to find the relationship between variables.
- In Machine Learning, and in statistical modeling, that relationship is used to predict the outcome of future events.
- For example, you can observe several employees of some company and try to understand how their salaries depend on the **features,** such as experience, level of education, role, city they work in, and so on.
- Similarly, you can try to establish a mathematical dependence of the prices of houses on their areas, numbers of bedrooms, distances to the city center, and so on.

Linear Regression

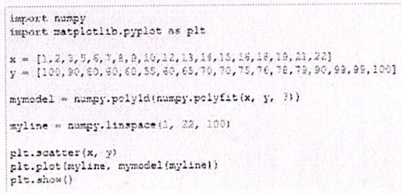
- Linear regression uses the relationship between the data-points to draw a straight line through all them.
- This line can be used to predict future values.
- Python has methods for finding a relationship between data-points and to draw a line of linear regression.

```
import matplotlib.pyplot as plt
from scipy import stats

x = [5,7,8,7,2,17,2,9,4,11,12,9,6]
y = [99,86,87,88,111,86,103,87,94,78,77,85,86]

slope, intercept, r, p, std_err = stats.linregress(x, y)

def myfunc(x):
    return slope * x + intercept

mymodel = list(map(myfunc, x))

plt.scatter(x, y)
plt.plot(x, mymodel)
plt.show()
```

- It is important to know how the relationship between the values of the x-axis and the values of the y-axis is, if there are no relationship the linear regression can not be used to predict anything.
- This relationship - the coefficient of correlation - is called r.
- The r value ranges from -1 to 1, where 0 means no relationship, and 1 (and -1) means 100% related.
- Python and the Scipy module will compute this value.

Polynomial Regression

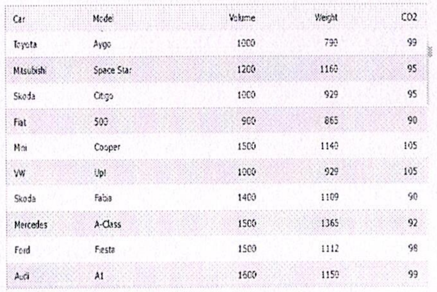
- Polynomial regression, like linear regression, uses the relationship between the variables x and y to find the best way to draw a line through the data points.
- Import numpy and matplotlib then draw the line of Polynomial Regression:



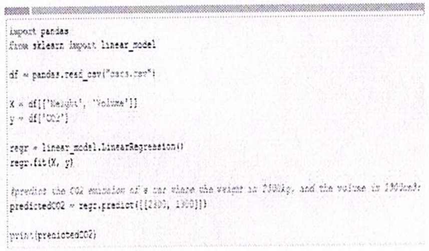
```
import numpy
import matplotlib.pyplot as plt

x = [1,2,3,5,6,7,8,9,10,12,13,14,15,16,18,19,21,22]
y = [100,90,80,60,60,55,60,65,70,70,75,76,78,79,90,99,99,100]

mymodel = numpy.poly1d(numpy.polyfit(x, y, 3))

myline = numpy.linspace(1, 22, 100)

plt.scatter(x, y)
plt.plot(myline, mymodel(myline))
plt.show()
```

Multiple regression

- Multiple regression is like linear regression, but with more than one independent value, meaning that we try to predict a value based on **two or more** variables.



Car	Model	Volume	Weight	CO2
Toyota	Aygo	1000	790	99
Mitsubishi	Space Star	1200	1160	95
Skoda	Citigo	1000	929	95
Fiat	500	900	865	90
Mini	Cooper	1500	1140	105
VW	Up!	1000	929	105
Skoda	Fabia	1400	1109	90
Mercedes	A-Class	1500	1365	92
Ford	Fiesta	1500	1112	98
Audi	A1	1600	1150	99



```
import pandas
from sklearn import linear_model

df = pandas.read_csv("cars.csv")

X = df[['Weight', 'Volume']]
y = df['CO2']

regr = linear_model.LinearRegression()
regr.fit(X, y)

#predict the CO2 emission of a car where the weight is 2300kg, and the volume is 1300cm3:
predictedCO2 = regr.predict([[2300, 1300]])

print(predictedCO2)
```

What is Data Analytics?

- Data analytics is the process of exploring and analyzing large datasets to make predictions and boost data-driven decision making. Data analytics allows us to collect, clean, and transform data to derive meaningful insights. It helps to answer questions, test hypotheses, or disprove theories.

Applications of Data Analytics

- Data analytics is used in the banking and e-commerce industries to detect fraudulent transactions.
- The healthcare sector uses data analytics to improve patient health by detecting diseases before they happen. It is commonly used for cancer detection.
- Data analytics finds its usage in inventory management to keep track of different items.

- Logistics companies use data analytics to ensure faster delivery of products by optimizing vehicle routes.
- Marketing professionals use analytics to reach out to the right customers and perform targeted marketing to increase ROI.
- Data analytics can be used for city planning, to build smart cities.

Types of Data Analytics

1. **Descriptive Analytics**
2. **Predictive Analytics**
3. **Prescriptive Analytics**

Descriptive Analytics

- It tells you what has happened. It can be done using an exploratory data analysis.
- Example: Studying the total units of chairs sold and the profit that was made in the past.

Predictive Analytics

- It tells you what will happen. It can be achieved by building predictive models.
- Example: Predicting the total units of chairs that would sell and the profit we can expect in the future.

Prescriptive Analytics

- It tells you how to make something happen. It can be done by deriving key insights and hidden patterns from the data.
- Example: Finding ways to improve sales and profit of chairs.

Data Analytics Process Steps

- **Data Collection:** The first step in data analytics is to collect or gather relevant data from multiple sources. Data can come from different databases, web servers, log files, social media, excel and CSV files, etc.
- **Data Preparation:** The next step in the process is to prepare the data. It involves cleaning the data to remove unwanted and redundant values, converting it into the right format, and making it ready for analysis. It also requires data wrangling.

- **Data Exploration:** After the data is ready, data exploration is done using various data visualization techniques to find unseen trends from the data.
- **Data Modeling:** The next step is to build your predictive models using machine learning algorithms to make future predictions.
- **Result interpretation:** The final step in any data analytics process is to derive meaningful results and check if the output is in line with your expected results.

Why Data Analytics Using Python?

- There are many programming languages available, but Python is popularly used by statisticians, engineers, and scientists to perform data analytics.

Reasons for using Python

- Python is easy to learn and understand and has a simple syntax.
- The programming language is scalable and flexible.
- It has a vast collection of libraries for numerical computation and data manipulation.
- Python provides libraries for graphics and data visualization to build plots.
- It has broad community support to help solve many kinds of queries.

Python Libraries for Data Analytics

- **NumPy:** NumPy supports n-dimensional arrays and provides numerical computing tools. It is useful for Linear algebra and Fourier transform.
- **Pandas:** Pandas provides functions to handle missing data, perform mathematical operations, and manipulate the data.
- **Matplotlib:** Matplotlib library is commonly used for plotting data points and creating interactive visualizations of the data.

- **SciPy:** SciPy library is used for scientific computing. It contains modules for optimization, linear algebra, integration, interpolation, special functions, signal and image processing.
- **Scikit-Learn:** Scikit-Learn library has features that allow you to build regression, classification, and clustering models.

Advanced analytics

- Advanced analytics uses data science beyond traditional business intelligence (BI) methods to predict patterns and estimate the likelihood of future events. This in turn can help an organization be more responsive and significantly increase its accuracy in decision-making.
- Often used by data scientists, advanced analytics tools both combine and extend prescriptive analytics and predictive analytics while adding various options for enhanced visualization and predictive models.

Benefits of advanced analytics

- **Accurate forecasting.** Using advanced analytics can confirm or refute prediction and forecast models with a greater level of accuracy than traditional BI tools that still carry an element of uncertainty.
- **Faster decision-making.** With predictions that feature a high level of accuracy, executives can act more quickly, confident their business decisions will achieve the desired results and that favorable outcomes can be repeated.
- **Deeper insight.** Advanced analytics offers a deeper level of actionable insight from data, including customer preference, market trends and key business processes, which empowers stakeholders to make data-driven decisions that can directly affect their strategy.

- **Improved risk management.** The higher level of accuracy provided by advanced analytics predictions can help businesses reduce their risk of costly mistakes.
- **Anticipate problems and opportunities.** Advanced analytics uses statistical models to reveal potential problems on the business' current trajectory, or identify new opportunities, so stakeholders can quickly change course and achieve better outcomes.

Some advanced analytics techniques?

- **Data mining.** This process sorts through large data sets to identify patterns and establish relationships to solve problems through data analysis.
- **Sentiment analysis.** This technique uses natural language processing, text analysis and biometrics to identify the emotional tone behind a body of text.
- **Cluster analysis.** This process matches pieces of unstructured data based on similarities found between them.

- Complex event processing. This technique uses technology to predict high-level events likely to result from specific sets of low-level factors.
- Big data analytics. This is the process of examining large volumes of structured, semi-structured and unstructured data to uncover information such as hidden patterns, correlations, market trends and customer preferences.

- **Machine learning.** The development of machine learning has dramatically increased the speed at which data can be processed and analyzed, facilitating disciplines like predictive analytics.
- **Data visualization.** This process of presenting data in graphical format makes data analysis and sharing more accessible across organizations.

Use cases for advanced analytics?

- **Marketing metrics.** With advanced analytics, marketing organizations can create customized, targeted marketing campaigns and avoid wasting money on ineffective strategies. Analyzing future outcomes also can help an organization identify opportunities to up-sell and optimize the marketing funnel.
- **Supply chain optimization.** Advanced analytics can help an organization factor demand, cost fluctuations and changing consumer preferences to create an agile supply chain that can quickly adapt to changing market conditions.

- **Risk management.** By examining particular data sets and data streams in real time, advanced analytics can help data scientists identify patterns that may indicate high levels of risk, for example by identifying possible payment fraud or insurance liabilities.
- **Business operations.** Advanced analytics can help organizations streamline their operations and adapt them to better suit predictions on changing market conditions or trends and ultimately increase revenue.