

Social Network Analysis
Prof. Shivani Kumar
Department of Computer Science and Engineering
Indraprastha Institute of Information Technology, Delhi

Lecture - 04
Introduction to Python/Colab

Hello all, I am Shivani and I am going to take you through some coding exercises and lectures for this course Social Network Analysis. So, today we will be beginning with an introduction to Python with the help of Google Colab.

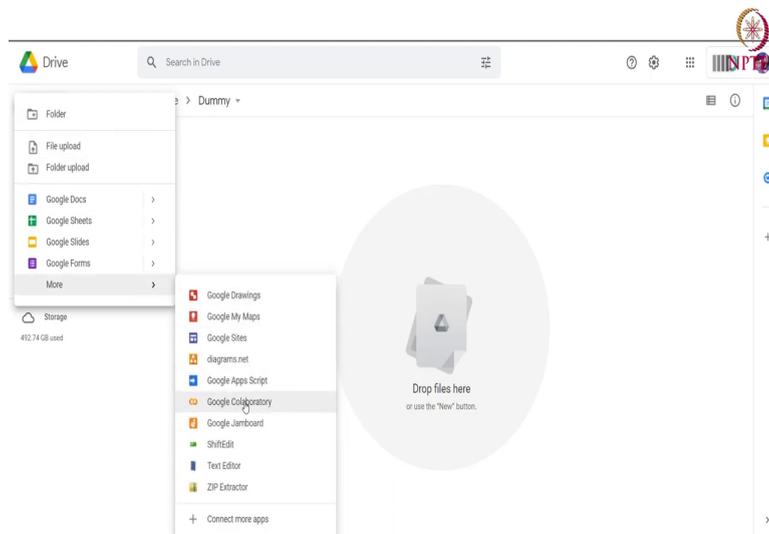
(Refer Slide Time: 00:40)

```
GettingStarted.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
Agenda
1. Google Colab
2. Introduction to Python
Google Colab
• Online interactive coding environment.
• Access to hardware accelerators like GPU and TPU.
• No system resources are used.
• Notebooks saves on Google drive
[1] 1 import os
    2 from google.colab import drive
[2] 1 MOUNTPOINT = "/content/gdrive"
    2 DATA_PATH = os.path.join(MOUNTPOINT, "mydrive")
    3 drive.mount(MOUNTPOINT)
[3] 1 !ls
[] 1 !cd gdrive/mydrive/NPTEL
0s completed at 10:20 AM
```

To begin with we must understand what Google Colab is. So, basically Google Colab is an online interactive platform which allows us to do Python programming on the go.

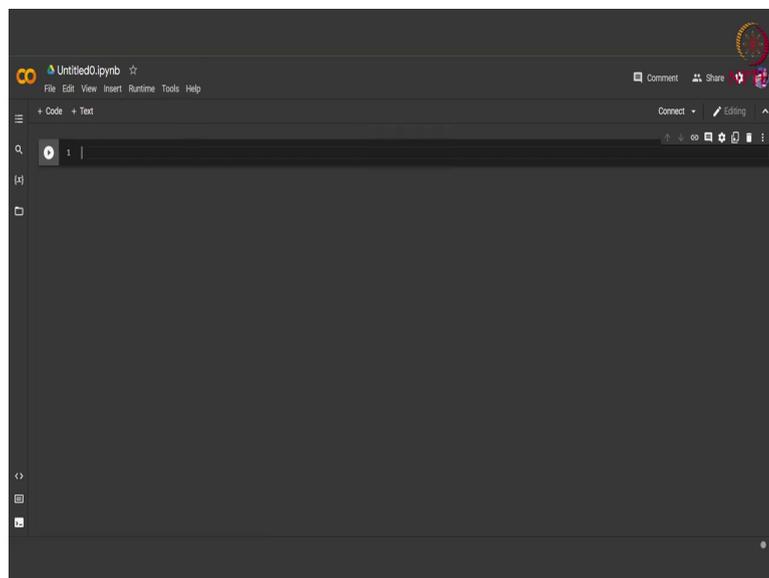
Google Colab provides us with some hardware accelerators like GPUs and TPUs along with cloud CPUs. Essentially if you are working on Google Colab; that means, you are using 0 percent of your own PCs resources except for the internet of course. So, before we get started with the coding in Google Colab, we must know how to create a notebook here, to do that you need two things; first an internet connection and another thing is a Google Drive account.

(Refer Slide Time: 01:36)



If you have a Google Drive account simply go to the folder where you want the a notebook to be and press new. Then if you go to the option More, you will see the Google Colaboratory option here, simply click here and you will be taken to a new brand new notebook.

(Refer Slide Time: 01:56)

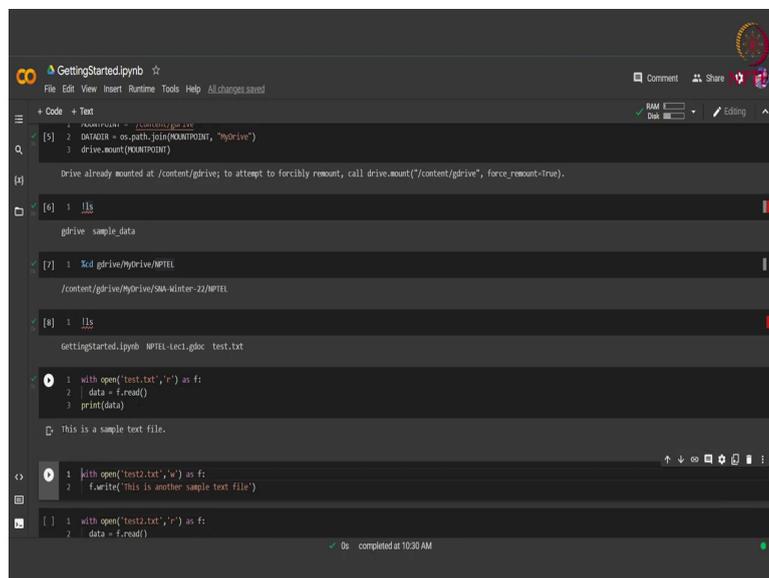


This will be an empty notebook, in which you can write any code that you want. We already have created a notebook here and we will be using this for today's session.

Since we are working on Google Drive and we may want to access files that are already present in this folder, to do that we must connect our notebook to the drive since we are getting assigned a new Cloud CPU each time we create the notebook. So, to mount the Google Drive to our Google Colab notebook we must use this mount function this function is available in this Python library called google dot colab.

We import this library and then we simply call this drive dot mount function giving it as input, the path that we want to mount. So, here our path is simply a drive and we also append this my drive path to the main path.

(Refer Slide Time: 03:09)



```
1 from google.colab import drive
2 drive.mount('/content/gdrive')
3 drive.mount('/content/gdrive')

Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).

[6] 1 !ls
      gdrive sample_data

[7] 1 !cd gdrive/MyDrive/NPTEL
      /content/gdrive/MyDrive/5NA-Winter-22/NPTEL

[8] 1 !ls
      gettingstarted.ipynb NPTEL-lect1.gdoc test.txt

1 with open('test.txt', 'r') as f:
2   data = f.read()
3   print(data)
This is a sample text file.

1 with open('test2.txt', 'w') as f:
2   f.write('This is another sample text file')

[ ] 1 with open('test2.txt', 'r') as f:
2   data = f.read()
```

We run this cell and this cell will generally ask us for a permission for which we just select accept and yes and then we are shown this message that the drive is mounted.

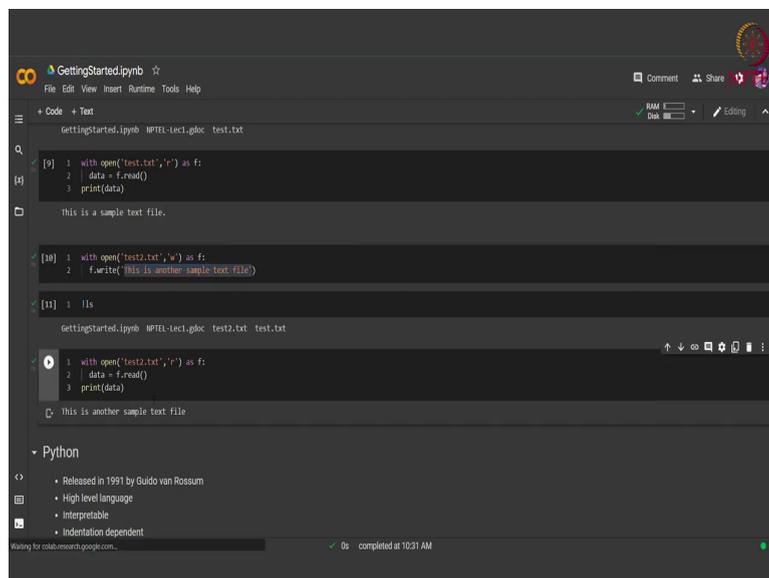
Now, since this is a Python notebook, we cannot use a terminal directly here, but we can use shell commands by using this exclamation marks symbol. We simply run this shell command and we can see what all files and folders we have on the path that is mounted to our drive. Except for exclamation mark we can also use this percentage sign in order to run shell scripts. So, we simply move to the required folder where we have our data and we again do ls here and we see that we have 3 files in this particular folder.

Now, what we can do is, now that we have our folder mounted with the notebook, we can also create and read information from this folder. So, suppose we have this text file already

present in our folder as can be seen from this ls command, that we have this text test dot txt already present in our folder. We can read this by calling this function, with open name of the file in what mode we want to open it.

So, here we are opening it in the reading mode, then the file descriptor and then we simply call the read function from the descriptor in order to read the contents of the file. After we have the contents in this variable called data, we simply print this variable using the print function. Let us run this cell and we can see that we are shown the content of the file that is this is a sample text file.

(Refer Slide Time: 05:26)



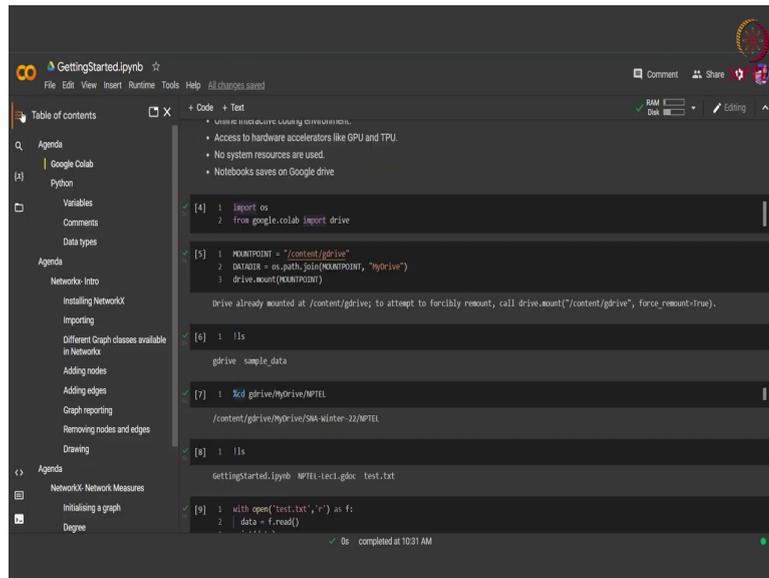
The screenshot shows a Jupyter Notebook interface with the following content:

```
GettingStarted.ipynb
+ Code + Text
GettingStarted.ipynb NPTEL-lect1.gdoc test.txt
[9] 1 with open("test.txt","r") as f:
    2 | data = f.read()
    3 | print(data)
This is a sample text file.
[10] 1 with open("test2.txt","w") as f:
    2 | f.write("This is another sample text file")
[11] 1 !ls
GettingStarted.ipynb NPTEL-lect1.gdoc test2.txt test.txt
[12] 1 with open("test2.txt","r") as f:
    2 | data = f.read()
    3 | print(data)
This is another sample text file
Python
- Python
  - Released in 1991 by Guido van Rossum
  - High level language
  - Interpretable
  - Indentation dependent
Waiting for colab.research.google.com... 0s completed at 10:31 AM
```

Now, instead of just reading we can also write to the directory that is mounted with our notebook we can simply call the a text file, but with the option of writing. So, we write with open, the name of the file the option write and the file descriptor and then we call the write function with the contents that we want to write in the file. We run this cell and now we will open this file in the read mode to see whether we have the contents now in or not.

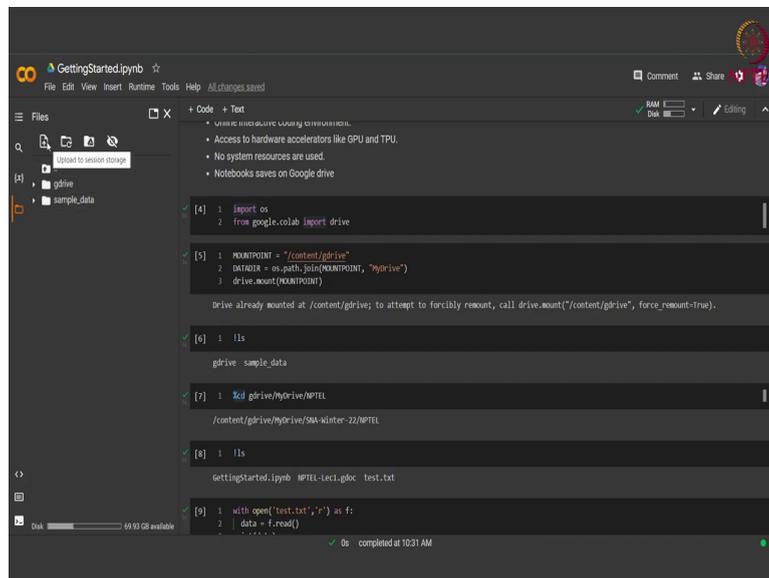
But before that let us first see whether our file is actually present in this folder or not. We see that now we also have this test2 dot txt present in our folder. We read this using the same command as we read the test dot txt, we read this and see that whatever content that we that we have written inside the file that is being read here. Instead of mounting our file using this these functions, Google also provides us with a ui that is that can help us in mounting.

(Refer Slide Time: 06:52)



So, here if we click at the left menu, we can see the table of contents. If we go down we have this folder option.

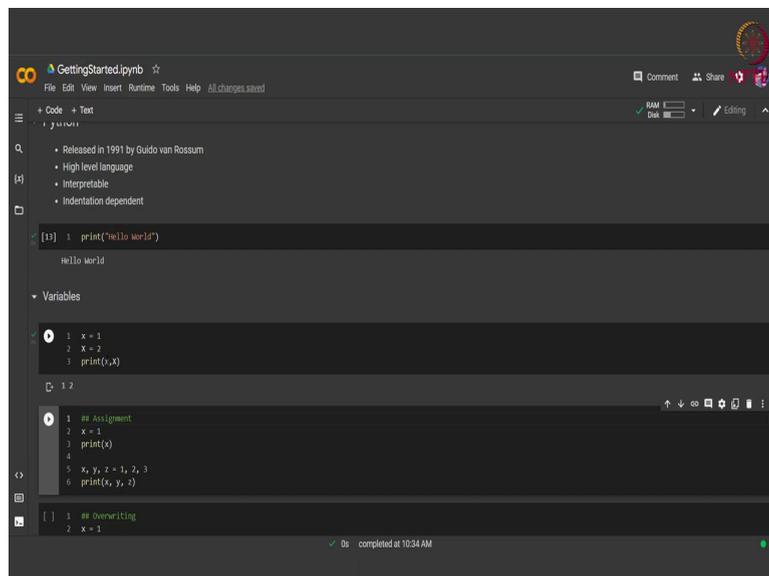
(Refer Slide Time: 07:03)



We click here and we can see that since this is already mounted, we can see the contents of the drive. We can refresh this contents or we can mount the drive again. We can also directly upload any file from here. Now, that we are familiar with Google Colab we would encourage you to explore Google Colab more on your own, since we only covered a very small amount

of functionality that it provides. But this is enough for us to get started with Python. Let us move on to Python.

(Refer Slide Time: 07:50)



The screenshot shows a Jupyter Notebook interface with the following content:

- Search results for Python: Released in 1991 by Guido van Rossum, High level language, Interpretable, Indentation dependent.
- Code cell 1: `print("Hello world")` with output: `Hello world`.
- Variables section showing a list of variables: `x = 1`, `X = 2`, and `print(x,X)`.
- Code cell 2: `1 # Assignment`, `2 x = 1`, `3 print(x)`, `4`, `5 x, y, z = 1, 2, 3`, `6 print(x, y, z)`.
- Code cell 3: `1 # Overwriting`, `2 x = 1`.

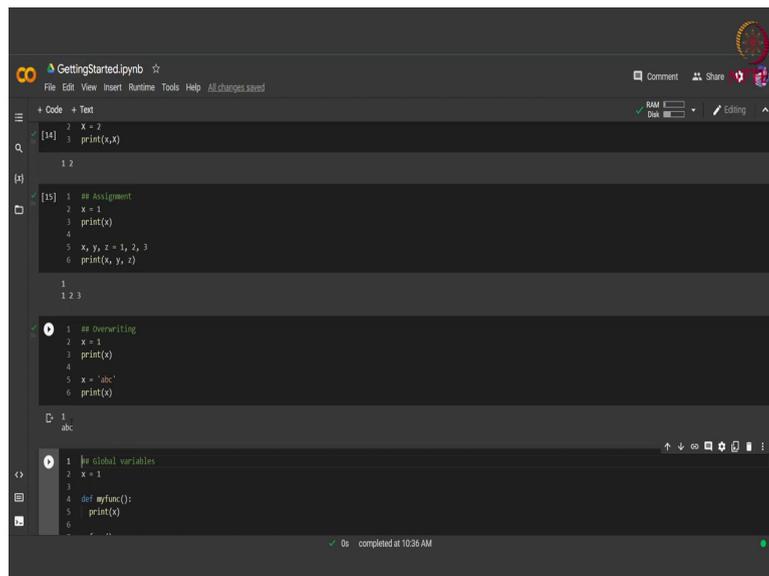
So, Python is basically a high level interpretable and a general purpose programming language. So, by general purpose I mean that it can be used for various purposes like web development software development or server side scripting. We; so let us first start with our “Hello World” function of Python.

So, we can simply call this print function and give the input as “Hello World” and we are given the “Hello World” as the output. Now Python it like any other programming languages it has variables, data types, comments and such things. So, let us first begin with the variables that are involved in Python. So, like as I said like any other programming language Python also has variables with a similar naming convention. For example, a variable name cannot start with a digit, it has to start with an alphabet or an underscore.

Also, Python variables are case sensitive that is small x and capital X will represent two different variables. So, this can be seen here if we have small x equal to 1 and capital X equal to 2 and we print x and capital X together, we will get the output as 1 and 2 because the variables are case sensitive. Now as seen in the above code snippet, what we did was assignment that is we assigned some value to the variable x.

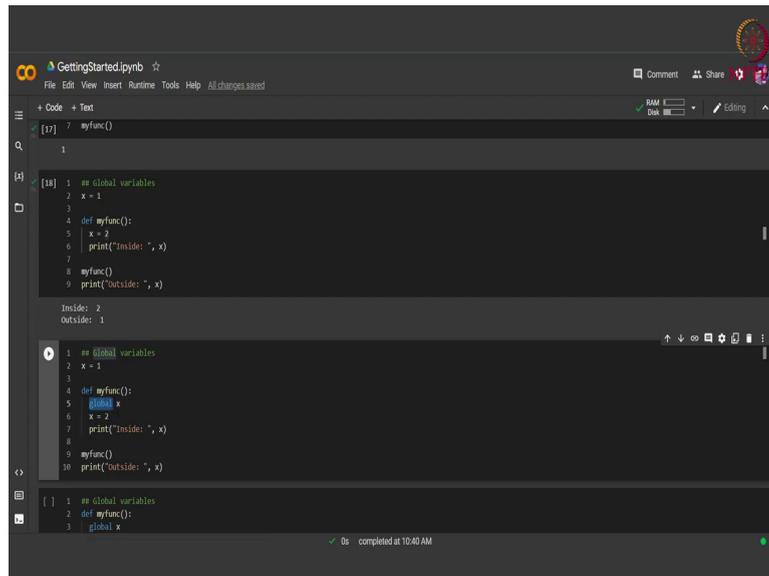
So, here also we are assigning the value of integer 1 and then we are simply printing the variable x. Apart from just this single assignment we can also assign different variables with different values by simply doing this comma separation. So, we have 3 variables, x, y and z separated by comma and they are assigned with three different values 1, 2 and 3 also separated by comma.

(Refer Slide Time: 10:21)



```
GettingStarted.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
RAM
Disk
Editing
[14] 1 x = 2
     2 print(x)
     3
     4
     5
     6
     7
     8
     9
    10
    11
    12
    13
    14
    15
    16
    17
    18
    19
    20
    21
    22
    23
    24
    25
    26
    27
    28
    29
    30
    31
    32
    33
    34
    35
    36
    37
    38
    39
    40
    41
    42
    43
    44
    45
    46
    47
    48
    49
    50
    51
    52
    53
    54
    55
    56
    57
    58
    59
    60
    61
    62
    63
    64
    65
    66
    67
    68
    69
    70
    71
    72
    73
    74
    75
    76
    77
    78
    79
    80
    81
    82
    83
    84
    85
    86
    87
    88
    89
    90
    91
    92
    93
    94
    95
    96
    97
    98
    99
   100
   101
   102
   103
   104
   105
   106
   107
   108
   109
   110
   111
   112
   113
   114
   115
   116
   117
   118
   119
   120
   121
   122
   123
   124
   125
   126
   127
   128
   129
   130
   131
   132
   133
   134
   135
   136
   137
   138
   139
   140
   141
   142
   143
   144
   145
   146
   147
   148
   149
   150
   151
   152
   153
   154
   155
   156
   157
   158
   159
   160
   161
   162
   163
   164
   165
   166
   167
   168
   169
   170
   171
   172
   173
   174
   175
   176
   177
   178
   179
   180
   181
   182
   183
   184
   185
   186
   187
   188
   189
   190
   191
   192
   193
   194
   195
   196
   197
   198
   199
  200
  201
  202
  203
  204
  205
  206
  207
  208
  209
  210
  211
  212
  213
  214
  215
  216
  217
  218
  219
  220
  221
  222
  223
  224
  225
  226
  227
  228
  229
  230
  231
  232
  233
  234
  235
  236
  237
  238
  239
  240
  241
  242
  243
  244
  245
  246
  247
  248
  249
  250
  251
  252
  253
  254
  255
  256
  257
  258
  259
  260
  261
  262
  263
  264
  265
  266
  267
  268
  269
  270
  271
  272
  273
  274
  275
  276
  277
  278
  279
  280
  281
  282
  283
  284
  285
  286
  287
  288
  289
  290
  291
  292
  293
  294
  295
  296
  297
  298
  299
  300
  301
  302
  303
  304
  305
  306
  307
  308
  309
  310
  311
  312
  313
  314
  315
  316
  317
  318
  319
  320
  321
  322
  323
  324
  325
  326
  327
  328
  329
  330
  331
  332
  333
  334
  335
  336
  337
  338
  339
  340
  341
  342
  343
  344
  345
  346
  347
  348
  349
  350
  351
  352
  353
  354
  355
  356
  357
  358
  359
  360
  361
  362
  363
  364
  365
  366
  367
  368
  369
  370
  371
  372
  373
  374
  375
  376
  377
  378
  379
  380
  381
  382
  383
  384
  385
  386
  387
  388
  389
  390
  391
  392
  393
  394
  395
  396
  397
  398
  399
  400
  401
  402
  403
  404
  405
  406
  407
  408
  409
  410
  411
  412
  413
  414
  415
  416
  417
  418
  419
  420
  421
  422
  423
  424
  425
  426
  427
  428
  429
  430
  431
  432
  433
  434
  435
  436
  437
  438
  439
  440
  441
  442
  443
  444
  445
  446
  447
  448
  449
  450
  451
  452
  453
  454
  455
  456
  457
  458
  459
  460
  461
  462
  463
  464
  465
  466
  467
  468
  469
  470
  471
  472
  473
  474
  475
  476
  477
  478
  479
  480
  481
  482
  483
  484
  485
  486
  487
  488
  489
  490
  491
  492
  493
  494
  495
  496
  497
  498
  499
  500
  501
  502
  503
  504
  505
  506
  507
  508
  509
  510
  511
  512
  513
  514
  515
  516
  517
  518
  519
  520
  521
  522
  523
  524
  525
  526
  527
  528
  529
  530
  531
  532
  533
  534
  535
  536
  537
  538
  539
  540
  541
  542
  543
  544
  545
  546
  547
  548
  549
  550
  551
  552
  553
  554
  555
  556
  557
  558
  559
  560
  561
  562
  563
  564
  565
  566
  567
  568
  569
  570
  571
  572
  573
  574
  575
  576
  577
  578
  579
  580
  581
  582
  583
  584
  585
  586
  587
  588
  589
  590
  591
  592
  593
  594
  595
  596
  597
  598
  599
  600
  601
  602
  603
  604
  605
  606
  607
  608
  609
  610
  611
  612
  613
  614
  615
  616
  617
  618
  619
  620
  621
  622
  623
  624
  625
  626
  627
  628
  629
  630
  631
  632
  633
  634
  635
  636
  637
  638
  639
  640
  641
  642
  643
  644
  645
  646
  647
  648
  649
  650
  651
  652
  653
  654
  655
  656
  657
  658
  659
  660
  661
  662
  663
  664
  665
  666
  667
  668
  669
  670
  671
  672
  673
  674
  675
  676
  677
  678
  679
  680
  681
  682
  683
  684
  685
  686
  687
  688
  689
  690
  691
  692
  693
  694
  695
  696
  697
  698
  699
  700
  701
  702
  703
  704
  705
  706
  707
  708
  709
  710
  711
  712
  713
  714
  715
  716
  717
  718
  719
  720
  721
  722
  723
  724
  725
  726
  727
  728
  729
  730
  731
  732
  733
  734
  735
  736
  737
  738
  739
  740
  741
  742
  743
  744
  745
  746
  747
  748
  749
  750
  751
  752
  753
  754
  755
  756
  757
  758
  759
  760
  761
  762
  763
  764
  765
  766
  767
  768
  769
  770
  771
  772
  773
  774
  775
  776
  777
  778
  779
  780
  781
  782
  783
  784
  785
  786
  787
  788
  789
  790
  791
  792
  793
  794
  795
  796
  797
  798
  799
  800
  801
  802
  803
  804
  805
  806
  807
  808
  809
  810
  811
  812
  813
  814
  815
  816
  817
  818
  819
  820
  821
  822
  823
  824
  825
  826
  827
  828
  829
  830
  831
  832
  833
  834
  835
  836
  837
  838
  839
  840
  841
  842
  843
  844
  845
  846
  847
  848
  849
  850
  851
  852
  853
  854
  855
  856
  857
  858
  859
  860
  861
  862
  863
  864
  865
  866
  867
  868
  869
  870
  871
  872
  873
  874
  875
  876
  877
  878
  879
  880
  881
  882
  883
  884
  885
  886
  887
  888
  889
  890
  891
  892
  893
  894
  895
  896
  897
  898
  899
  900
  901
  902
  903
  904
  905
  906
  907
  908
  909
  910
  911
  912
  913
  914
  915
  916
  917
  918
  919
  920
  921
  922
  923
  924
  925
  926
  927
  928
  929
  930
  931
  932
  933
  934
  935
  936
  937
  938
  939
  940
  941
  942
  943
  944
  945
  946
  947
  948
  949
  950
  951
  952
  953
  954
  955
  956
  957
  958
  959
  960
  961
  962
  963
  964
  965
  966
  967
  968
  969
  970
  971
  972
  973
  974
  975
  976
  977
  978
  979
  980
  981
  982
  983
  984
  985
  986
  987
  988
  989
  990
  991
  992
  993
  994
  995
  996
  997
  998
  999
 1000
 1001
 1002
 1003
 1004
 1005
 1006
 1007
 1008
 1009
 1010
 1011
 1012
 1013
 1014
 1015
 1016
 1017
 1018
 1019
 1020
 1021
 1022
 1023
 1024
 1025
 1026
 1027
 1028
 1029
 1030
 1031
 1032
 1033
 1034
 1035
 1036
 1037
 1038
 1039
 1040
 1041
 1042
 1043
 1044
 1045
 1046
 1047
 1048
 1049
 1050
 1051
 1052
 1053
 1054
 1055
 1056
 1057
 1058
 1059
 1060
 1061
 1062
 1063
 1064
 1065
 1066
 1067
 1068
 1069
 1070
 1071
 1072
 1073
 1074
 1075
 1076
 1077
 1078
 1079
 1080
 1081
 1082
 1083
 1084
 1085
 1086
 1087
 1088
 1089
 1090
 1091
 1092
 1093
 1094
 1095
 1096
 1097
 1098
 1099
 1100
 1101
 1102
 1103
 1104
 1105
 1106
 1107
 1108
 1109
 1110
 1111
 1112
 1113
 1114
 1115
 1116
 1117
 1118
 1119
 1120
 1121
 1122
 1123
 1124
 1125
 1126
 1127
 1128
 1129
 1130
 1131
 1132
 1133
 1134
 1135
 1136
 1137
 1138
 1139
 1140
 1141
 1142
 1143
 1144
 1145
 1146
 1147
 1148
 1149
 1150
 1151
 1152
 1153
 1154
 1155
 1156
 1157
 1158
 1159
 1160
 1161
 1162
 1163
 1164
 1165
 1166
 1167
 1168
 1169
 1170
 1171
 1172
 1173
 1174
 1175
 1176
 1177
 1178
 1179
 1180
 1181
 1182
 1183
 1184
 1185
 1186
 1187
 1188
 1189
 1190
 1191
 1192
 1193
 1194
 1195
 1196
 1197
 1198
 1199
 1200
 1201
 1202
 1203
 1204
 1205
 1206
 1207
 1208
 1209
 1210
 1211
 1212
 1213
 1214
 1215
 1216
 1217
 1218
 1219
 1220
 1221
 1222
 1223
 1224
 1225
 1226
 1227
 1228
 1229
 1230
 1231
 1232
 1233
 1234
 1235
 1236
 1237
 1238
 1239
 1240
 1241
 1242
 1243
 1244
 1245
 1246
 1247
 1248
 1249
 1250
 1251
 1252
 1253
 1254
 1255
 1256
 1257
 1258
 1259
 1260
 1261
 1262
 1263
 1264
 1265
 1266
 1267
 1268
 1269
 1270
 1271
 1272
 1273
 1274
 1275
 1276
 1277
 1278
 1279
 1280
 1281
 1282
 1283
 1284
 1285
 1286
 1287
 1288
 1289
 1290
 1291
 1292
 1293
 1294
 1295
 1296
 1297
 1298
 1299
 1300
 1301
 1302
 1303
 1304
 1305
 1306
 1307
 1308
 1309
 1310
 1311
 1312
 1313
 1314
 1315
 1316
 1317
 1318
 1319
 1320
 1321
 1322
 1323
 1324
 1325
 1326
 1327
 1328
 1329
 1330
 1331
 1332
 1333
 1334
 1335
 1336
 1337
 1338
 1339
 1340
 1341
 1342
 1343
 1344
 1345
 1346
 1347
 1348
 1349
 1350
 1351
 1352
 1353
 1354
 1355
 1356
 1357
 1358
 1359
 1360
 1361
 1362
 1363
 1364
 1365
 1366
 1367
 1368
 1369
 1370
 1371
 1372
 1373
 1374
 1375
 1376
 1377
 1378
 1379
 1380
 1381
 1382
 1383
 1384
 1385
 1386
 1387
 1388
 1389
 1390
 1391
 1392
 1393
 1394
 1395
 1396
 1397
 1398
 1399
 1400
 1401
 1402
 1403
 1404
 1405
 1406
 1407
 1408
 1409
 1410
 1411
 1412
 1413
 1414
 1415
 1416
 1417
 1418
 1419
 1420
 1421
 1422
 1423
 1424
 1425
 1426
 1427
 1428
 1429
 1430
 1431
 1432
 1433
 1434
 1435
 1436
 1437
 1438
 1439
 1440
 1441
 1442
 1443
 1444
 1445
 1446
 1447
 1448
 1449
 1450
 1451
 1452
 1453
 1454
 1455
 1456
 1457
 1458
 1459
 1460
 1461
 1462
 1463
 1464
 1465
 1466
 1467
 1468
 1469
 1470
 1471
 1472
 1473
 1474
 1475
 1476
 1477
 1478
 1479
 1480
 1481
 1482
 1483
 1484
 1485
 1486
 1487
 1488
 1489
 1490
 1491
 1492
 1493
 1494
 1495
 1496
 1497
 1498
 1499
 1500
 1501
 1502
 1503
 1504
 1505
 1506
 1507
 1508
 1509
 1510
 1511
 1512
 1513
 1514
 1515
 1516
 1517
 1518
 1519
 1520
 1521
 1522
 1523
 1524
 1525
 1526
 1527
 1528
 1529
 1530
 1531
 1532
 1533
 1534
 1535
 1536
 1537
 1538
 1539
 1540
 1541
 1542
 1543
 1544
 1545
 1546
 1547
 1548
 1549
 1550
 1551
 1552
 1553
 1554
 1555
 1556
 1557
 1558
 1559
 1560
 1561
 1562
 1563
 1564
 1565
 1566
 1567
 1568
 1569
 1570
 1571
 1572
 1573
 1574
 1575
 1576
 1577
 1578
 1579
 1580
 1581
 1582
 1583
 1584
 1585
 1586
 1587
 1588
 1589
 1590
 1591
 1592
 1593
 1594
 1595
 1596
 1597
 1598
 1599
 1600
 1601
 1602
 1603
 1604
 1605
 1606
 1607
 1608
 1609
 1610
 1611
 1612
 1613
 1614
 1615
 1616
 1617
 1618
 1619
 1620
 1621
 1622
 1623
 1624
 1625
 1626
 1627
 1628
 1629
 1630
 1631
 1632
 1633
 1634
 1635
 1636
 1637
 1638
 1639
 1640
 1641
 1642
 1643
 1644
 1645
 1646
 1647
 1648
 1649
 1650
 1651
 1652
 1653
 1654
 1655
 1656
 1657
 1658
 1659
 1660
 1661
 1662
 1663
 1664
 1665
 1666
 1667
 1668
 1669
 1670
 1671
 1672
 1673
 1674
 1675
 1676
 1677
 1678
 1679
 1680
 1681
 1682
 1683
 1684
 1685
 1686
 1687
 1688
 1689
 1690
 1691
 1692
 1693
 1694
 1695
 1696
 1697
 1698
 1699
 1700
 1701
 1702
 1703
 1704
 1705
 1706
 1707
 1708
 1709
 1710
 1711
 1712
 1713
 1714
 1715
 1716
 1717
 1718
 1719
 1720
 1721
 1722
 1723
 1724
 1725
 1726
 1727
 1728
 1729
 1730
 1731
 1732
 1733
 1734
 1735
 1736
 1737
 1738
 1739
 1740
 1741
 1742
 1743
 1744
 1745
 1746
 1747
 1748
 1749
 1750
 1751
 1752
 1753
 1754
 1755
 1756
 1757
 1758
 1759
 1760
 1761
 1762
 1763
 1764
 1765
 1766
 1767
 1768
 1769
 1770
 1771
 1772
 1773
 1774
 1775
 1776
 1777
 1778
 1779
 1780
 1781
 1782
 1783
 1784
 1785
 1786
 1787
 1788
 1789
 1790
 1791
 1792
 1793
 1794
 1795
 1796
 1797
 1798
 1799
 1800
 1801
 1802
 1803
 1804
 1805
 1806
 1807
 1808
 1809
 1810
 1811
 1812
 1813
 1814
 1815
 1816
 1817
 1818
 1819
 1820
 1821
 1822
 1823
 1824
 1825
 1826
 1827
 1828
 1829
 1830
 1831
 1832
 1833
 1834
 1835
 1836
 1837
 1838
 1839
 1840
 1841
 1842
 1843
 1844
 1845
 1846
 1847
 1848
 1849
 1850
 1851
 1852
 1853
 1854
 1855
 1856
 1857
 1858
 1859
 1860
 1861
 1862
 1863
 1864
 1865
 1866
 1867
 1868
 1869
 1870
 1871
 1872
 1873
 1874
 1875
 1876
 1877
 1878
 1879
 1880
 1881
 1882
 1883
 1884
 1885
 1886
 1887
 1888
 1889
 1890
 1891
 1892
 1893
 1894
 1895
 1896
 1897
 1898
 1899
 1900
 1901
 1902
 1903
 1904
 1905
 1906
 1907
 1908
 1909
 1910
 1911
 1912
 1913
 1914
 1915
 1916
 1917
 1918
 1919
 1920
 1921
 1922
 1923
 1924
 1925
 1926
 1927
 1928
 1929
 1930
 1931
 1932
 1933
 1934
 1935
 1936
 1937
 1938
 1939
 1940
 1941
 1942
 1943
 1944
 1945
 1946
 1947
 1948
 1949
 1950
 1951
 1952
 1953
 1954
 1955
 1956
 1957
 1958
 1959
 1960
 1961
 1962
 1963
 1964
 1965
 1966
 1967
 1968
 1969
 1970
 1971
 1972
 1973
 1974
 1975
 1976
 1977
 1978
 1979
 1980
 1981
 1982
 1983
 1984
 1985
 1986
 1987
 1988
 1989
 1990
 1991
 1992
 1993
 1994
 1995
 1996
 1997
 1998
 1999
 2000
 2001
 2002
 2003
 2004
 2005
 2006
 2007
 2008
 2009
 2010
 2011
 2012
 2013
 2014
 2015
 2016
 2017
 2018
 2019
 2020
 2021
 2022
 2023
 
```

(Refer Slide Time: 11:46)



The screenshot shows a Jupyter Notebook interface with the following content:

```
GettingStarted.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text RAM 1.00 GB Disk 100 MB editing
[17] 7 myfunc()
1
[18] 1 ## Global variables
2 x = 1
3
4 def myfunc():
5     x = 2
6     print("Inside: ", x)
7
8 myfunc()
9 print("Outside: ", x)
Inside: 2
Outside: 1
1 ## Global variables
2 x = 1
3
4 def myfunc():
5     global x
6     x = 2
7     print("Inside: ", x)
8
9 myfunc()
10 print("Outside: ", x)
[ ] 1 ## Global variables
2 def myfunc():
3     global x
0s completed at 10:40 AM
```

Moving on, we might also want some global variables to be present in our program, global variables are variables that have the scope throughout the program, that is we can access them anywhere and probably modify them also, we will see about that. So, first we start with x as the global variable this is or this is by default a global variable, because it is defined outside of any function. So, we provide it with the value 1 and then we define a function called myfunc and simply print this value of x inside the function.

Then we call this function. Let us see what happens when we run this cell. We run this cell and we see that this print statement inside the function which is outputting the value of x, actually gives us the value that is present that was assigned to this x globally, that is we are outputting the value 1. Now, this happens when we do not modify the value of x anywhere inside the function.

But what happens when we have another x inside the function and we modify that value of x. So, here in this code snippet you can see that we have a global variable x with the value 1 here. And in this function called myfunc what we do is we simply assign the value 2 to the same variable x, but is it the same variable. So, in this scenario this is not the same variable, when we are declaring this or this assignment that is present inside the function it is creating another local variable x that has a scope only inside the function. That is if we have modified the value or we have assigned the value 2 to this x and print this x inside the function and again print an x outside the function, these 2 values will be different.

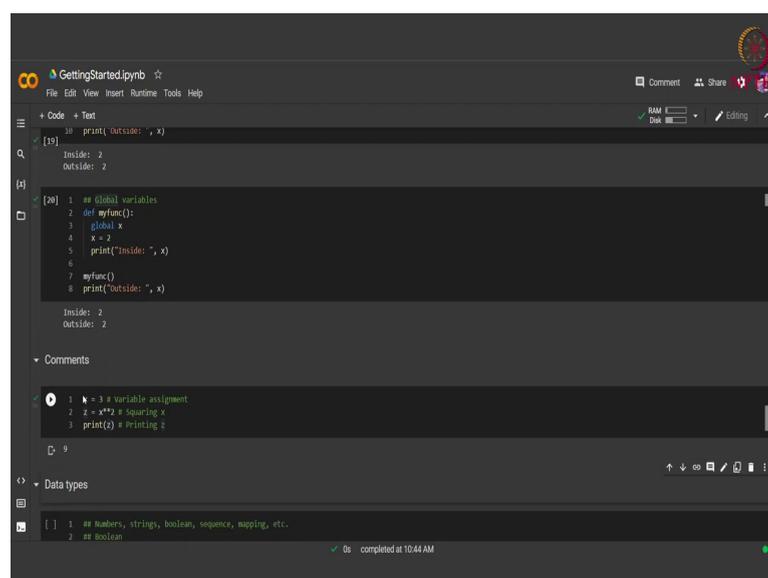
Because inside the function we will be referring to the local version of x, but outside the function since, the scope of the local x has finished we will be referring to the global x that was already present in our in our program. So, let us run this cell to verify our claim, we run this and we see that inside the function the value of x is coming out to be 2 as expected, since we already assigned it the value 2 inside the function.

So, this is a local variable as can be seen by the value of x that is coming outside the function. So, outside the function the value still remains 1, because the global x had the value 1, but it can also happen that we want to modify the global value inside the function. In order to do that what we do is we use the keyword global. We simply define x normally outside the functions as a global variable, this is by default a global variable since it is defined outside of any function.

But inside the function we need to declare that the x that we are using inside the function is actually a global variable to do that we use the keyword global. So, here we write global x and then modify the value of x, that is we are already telling the interpreter that the x that you are going to deal with is a global variable. We assigned the value 2 to this x and then again print x inside as well as outside.

Now, according to our hypothesis the value of x inside should be 2, but outside also it should be 2. Since, we already have defined that the x that we are modifying is the global x.

(Refer Slide Time: 16:34)



```
GettingStarted.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
RAM
Disk
Editing
[19]
print('Outside: ', x)
Inside: 2
Outside: 2
[20]
1 ## Global variables
2 def myfunc():
3     global x
4     x = 2
5     print('Inside: ', x)
6
7 myfunc()
8 print('Outside: ', x)
Inside: 2
Outside: 2
Comments
1 # z = 3 # variable assignment
2 z = x**2 # Squaring x
3 print(z) # Printing z
9
Data types
1 ## Numbers, strings, boolean, sequence, mapping, etc.
2 ## Boolean
0/ completed at 10:44 AM
```

We run this cell and we see that as expected we are getting the value of x as to both inside and outside the functions. Then again even if we do not initialize this x outside of any function we can initialize a new global variable inside the function using the global keyword.

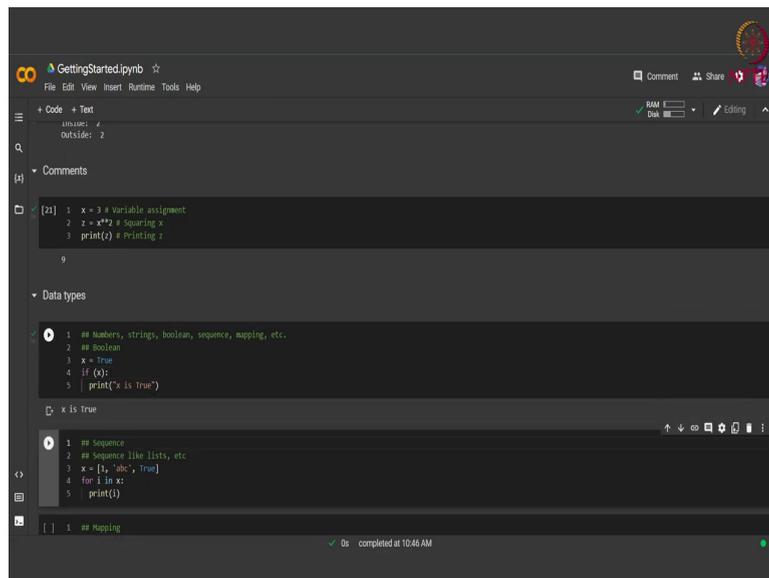
So, we just mentioned here, that we are defining a global variable x assign some value to that variable. And now again if we call this x outside this function we will still get x as a valid variable and with the value 2. Even though we have not defined this variable outside of this function, but inside we have defined it using the global keyword. We run this cell and we see that there are no errors and we are getting the value of x inside as 2 as well as outside as 2.

Now, like any other programming language Python also has comments. So, these comments they are ignored by the interpreter, but they are mostly used to improve the readability of the code so that sharing of code becomes easy. After all we are using Google Colab which allows us to share the code. So, we simply write comments in Python using the hash sign, as we already saw in the previous code snippets, I have mentioned the functionality of each cell using the comments here.

So, the number of hashes do not matter, just after the first hash whatever follows is considered as a comment. So, in this cell we can see that this that we have assigned a comment with each line to describe the functionality of each line separately. So, for example, here the first line is simply a variable assignment, the second line squares the value of x. So, for squaring we use this double asterisk operator then finally, we print the value of this square which we capture in another variable called z.

We run this and we see that whatever is written other than the comment is actually handled by the interpreter and the part that follows after hash that is the comment part is just simply ignored and is useful for the programmers to understand the code.

(Refer Slide Time: 19:30)



```
GettingStarted.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Test
lines: 4
outsides: 2
RAM
Disk
Editing

[2]: 1 x = 3 # Variable assignment
     2 z = x**2 # Squaring x
     3 print(z) # Printing z
     9

Data types

1 1 # Numbers, strings, boolean, sequence, mapping, etc.
  2 # Boolean
  3 x = True
  4 if (x):
  5 | print('x is True')
  x is True

1 1 # Sequence
  2 # Sequence like lists, etc
  3 x = ['a', 'b', 'c', True]
  4 for i in x:
  5 | print(i)

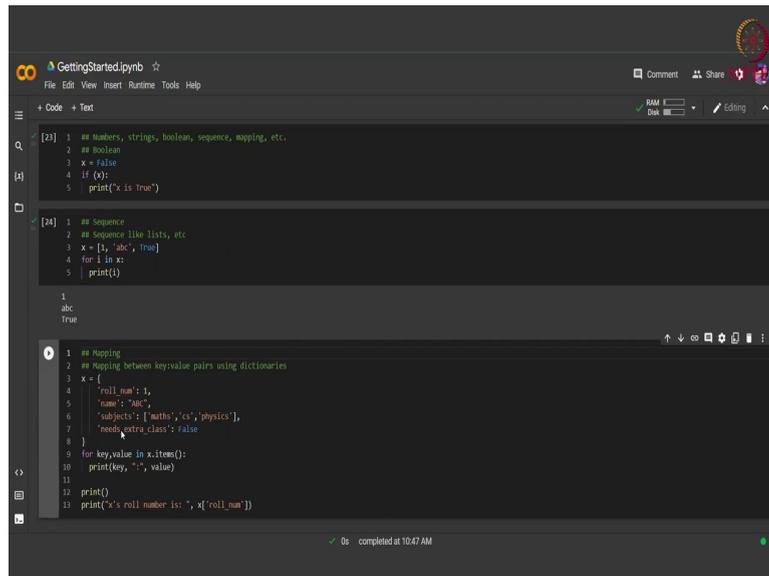
1 1 # Mapping
  0s completed at 10:45 AM
```

Moving on Python also supports various data types. So, in our previous cell we already saw the integer and the string data types. We saw that we can assign the same variable with different data types and there we saw integer and string.

Apart from integer and strings Python also supports Boolean, sequence, mapping type data types and a lot more. So, let us first see how Boolean is handled in Python. So, Boolean variable is simply a variable that can have either of the two values that is true or false. So, here we assign the variable x the value true. So, here our x becomes a Boolean variable, we can use such variable in a conditional statement like if.

So, we check if x; that means, if x is true then the following code statement will be executed. For example, here if x is true then the code will print that x is true and we have already assigned the value of x as true.

(Refer Slide Time: 20:55)



```
GettingStarted.ipynb
File Edit View Insert Runtime Tools Help
+ Code + Text
RAM 100%
Disk 100%
Editing

[23] 1 ## Numbers, strings, boolean, sequence, mapping, etc.
      2 ## Boolean
      3 x = False
      4 if (x):
      5     print("x is true")

[24] 1 ## Sequence
      2 ## Sequence like lists, etc
      3 x = [1, 'abc', True]
      4 for i in x:
      5     print(i)

1
abc
True

[25] 1 ## Mapping
      2 ## Mapping between keyvalue pairs using dictionaries
      3 x = {
      4     'roll_num': 1,
      5     'name': 'ABC',
      6     'subjects': ['maths', 'cs', 'physics'],
      7     'needs_extra_class': False
      8 }
      9 for key,value in x.items():
     10     print(key, ":", value)
     11
     12 print()
     13 print("x's roll number is: ", x['roll_num'])

0s completed at 10:47 AM
```

So, when we run this code this will be printed that x is true. We can also try with the value false and we will see that nothing will be printed because this conditional statement will come as false.

Now, apart from Boolean variables a very interesting data type of Python is a sequential data type. So, sequential data type can have lists sets or tuples in it and other things also. Let us focus on lists here. So, lists basically are like arrays, but in Python a very interesting thing to observe is that in a list we can have different elements of different data types. As you can see here we have a list x where we have three different data types in a single list.

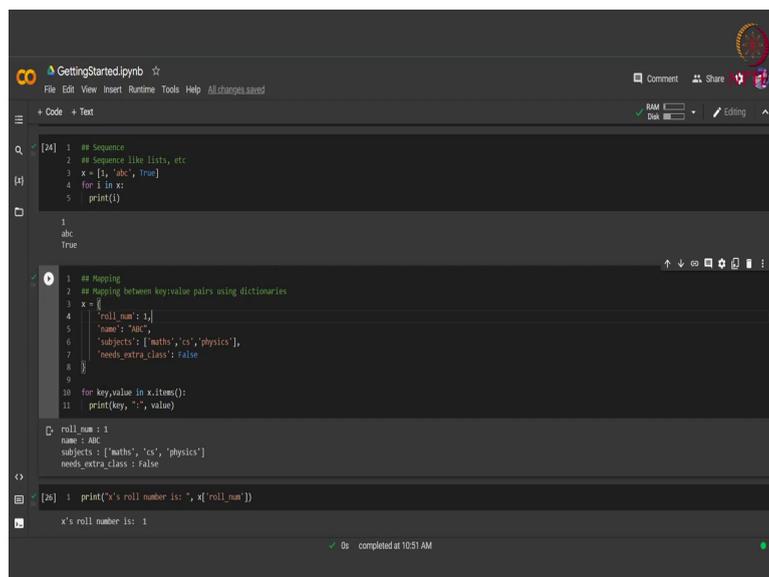
This is unlike an array which has the same data type. So, here list contains an integer, a string and a Boolean value. We can iterate over this list just like we iterate over an array by using a for loop we simply write for i in x that is. So, I will take on the value of one element at a time sequentially and we are simply printing this I, let us run this code and we see that each of the element of the list is printed in a sequential manner.

Moving on from sequences, we have the mapping data type. Now, mapping can be in a key value format. In Python we use the data type dictionaries to perform the mapping. So, dictionaries are basically a data type that has a key and then a value associated with each key. For example, here you can see that we are defining x as a dictionary and to define a dictionary we use these curly brackets and inside these brackets we have defined our keys followed by a colon followed by the value for the corresponding key.

So, for example, here x might contain information of a student. So, it has keys such as roll number, name subjects and or Boolean variable called needs extra class. Now, here we can see that each key can have different data type, data type values. For example, here roll number is an integer whereas, name is a string subjects is actually a sequential data type of three strings, then we have a Boolean type called needs extra class.

Now, in order to iterate over this dictionary, we simply write x dot items which gives us a tuple of key value pairs, in this tuple to access each key value we write for key comma value in x dot items. So, the variable key will take on the values of each key sequentially. Whereas, the variable value will take on the corresponding value, we then print this key and value pairs separated by a colon.

(Refer Slide Time: 24:56)



```
GettingStarted.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text
[24] 1 ## Sequence
2 ## Sequence like lists, etc
3 x = [1, 'abc', True]
4 for i in x:
5     print(i)

1
abc
True

[25] 1 ## Mapping
2 ## Mapping between key:value pairs using dictionaries
3 x = {
4     'roll_num': 1,
5     'name': 'abc',
6     'subjects': ['maths', 'cs', 'physics'],
7     'needs_extra_class': False
8 }
9
10 for key,value in x.items():
11     print(key, ':', value)

roll_num : 1
name : abc
subjects : ['maths', 'cs', 'physics']
needs_extra_class : False

[26] 1 print(x's roll number is: ', x['roll_num'])

x's roll number is: 1
0s completed at 10:51 AM
```

So, let us first print this and then we will see how we can access a particular key from the dictionary. So, when we iterate over all the items, we see that each key and value is being printed as we wanted separated by a colon and followed by the value. It is simply printed like this, but we can also access each key in a separate manner. So, to do that we simply write the name of the dictionary followed by the name of the key that we want to access.

So, here if we want to access the key roll number, we write x and in square brackets the string roll number that is the key name. So, here we write x's roll number is followed by this format. We run this and we see that x's roll number is whatever the value is here the value is

1. So, now, we it is important to note that a dictionary can have the same name of a key only once.

For example, we cannot have 2 roll numbers inside the same dictionary. So, that is each key must be unique inside the dictionary, although the value of the key can be changed over time. So, today we saw about how to get started with Colab and how to get started with Python with by learning some basic variables and basic data types we encourage you to go home and dwell deeper into this Python's world, because there is a plethora of information to explore there.