

ARTIFICIAL INTELLIGENCE

Lab Manual

[Fall/ Spring 2017]

|  |  |
| --- | --- |
| Student Name: RASHID RAHIM |  |
| Student Id: 9715 |  |
|  |  |

|  |  |
| --- | --- |
| Prepared By: *Dr. Noman Islam* |  |
| Instructor: *Dr. Noman Islam* |  |

**LIST OF EXPERIMENTS**

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **Date** | **Experiment** |  |
| **1** | \_\_/\_\_/\_\_ | To setup the environment and familiarize with Python |  |
| **2** | \_\_/\_\_/\_\_ | To study and implement algorithms in Python |  |
| **3** | \_\_/\_\_/\_\_ | To study and implement Graph search algorithms in Python |  |
| **4** | \_\_/\_\_/\_\_ | To study and understand numpy library |  |
| **5** | \_\_/\_\_/\_\_ | To study and implement pandas library |  |
| **6** | \_\_/\_\_/\_\_ | To study and implement Artificial Neural Network using Keras |  |
| **7** | \_\_/\_\_/\_\_ |  |  |
| **8** | \_\_/\_\_/\_\_ |  |  |
| **9** | \_\_/\_\_/\_\_ | To study and implement classification algorithms in Tensorflow |  |
| **10** | \_\_/\_\_/\_\_ | To study and implement clustering algorithms in Tensorflow |  |
| **11** | \_\_/\_\_/\_\_ | To study and implement Artificial Neural Network (ANN) in Tensorflow |  |
| **12** | \_\_/\_\_/\_\_ | To study and implement Convolutional Neural Network (CNN) in Tensorflow |  |
| **13** | \_\_/\_\_/\_\_ | To study and implement LSTM in Tensorflow |  |
| **14** | \_\_/\_\_/\_\_ | To study and implement Hidden Markov Model in Tensorflow |  |

**Lab 1: To setup the environment and familiarize with Python**

The objective of this lab is to set up the Python environment and get some familiarity with the language.

To set up the environment, follow the steps below:

1. Download and install Anaconda. Anaconda is the leading open data science platform powered by Python
2. Download and install PyCharm. PyCharm is an Integrated Development Environment (IDE) used in computer programming, specifically for the Python language.

**Lab Tasks:**

1. Write a small program in Python to print your CV.
2. print(**"Name:\n \t\t\t RASHID RAHIM"**)  
   print(**"CGPA:\n \t\t\t3.38"**)  
   print(**"Semester:\n \t\t\t6th"**)  
   print(**"University:\n \t\t\tIqra University Qulshan Campus"**)  
   print(**"Skills:\n \t\t\tGood at in Programming Language java,python,c#"**)  
   print(**"Interested In:\n \t\t\tCricket and FootBall"**)  
   print(**"Future Plans:\n \t\t\tDevelop an AI Base Project for cancer treatment"**)  
   print(**"Experience:\n \t\t\tNone"**)

Name:

Rashid Rahim

CGPA:

3.38

Semester:

6th

University:

Iqra University Qulshan Campus

Skills:

Good at in Programming Language java,python,c#

Interested In:

Cricket and FootBall

Future Plans:

Develop an AI Base Project for cancer treatment

Experience:

None

1. Write a program that takes the month (1…12) as input. Print whether the season is summer, winter, spring or autumn depending upon the input month.

a = int(input(**'Enter the Month\n'**))  
  
**if**(a==1 **or** a==2 **or** a==3):  
 {  
 print(**"Winter\n"**)  
 }  
**elif**(a==4 **or** a==5 **or** a==6):  
 {  
 print(**"Spring"**)  
 }  
**elif** (a == 7 **or** a == 8 **or** a == 9):  
 {  
 print(**"Summer"**)  
 }  
**elif** (a == 10 **or** a == 11 **or** a == 12):  
 {  
 print(**"Autumn"**)  
 }  
**else**:print(**"the month does not exist"**)

Enter the Month

6

Spring

1. To determine whether a year is a leap year, follow these steps:
   1. If the year is evenly divisible by 4, go to step 2. Otherwise, go to step 5.
   2. If the year is evenly divisible by 100, go to step 3. Otherwise, go to step 4.
   3. If the year is evenly divisible by 400, go to step 4. Otherwise, go to step 5.
   4. The year is a leap year (it has 366 days).
   5. The year is not a leap year (it has 365 days).

Write a program to input an year as integer. Using if…else, determines whether the input is a leap year or not.

x = int(input(**'Enter the Year\n'**))  
  
**if**(x%4==0):  
 **if**(x%100==0):  
 **if**(x%400==0):  
 {  
 print(**"leap year"**)  
 }  
 **else**:{  
 print(**"Not leap year"**)  
 }  
 **else**:{  
 print(**"leap year"**)  
 }  
**else**:{  
 print(**"Not leap year"**)  
}

Enter the Year

2005

Not leap year

1. Write a program that takes a line as input and finds the number of letters and digits in the input

x = input(**"enter the String\n"**)  
y=z=0  
  
**for** s **in** x:  
 **if** (s.isdigit()):  
 y=y+1  
 **if** (s.isalpha()):  
 z=z+1  
  
print(**"letter and digit"**,z,y)

enter the String

iqra123

letter and digit 4 3

1. Write a program that takes a sentence as input. Compute the frequency of each words and prints them.

a = input(**'enter the sentence \n'**)

x = len(a.split())

print(**"Number of the word are "**,x)

enter the sentence

this is a boy

Number of the word are 4

**Lab 2: To study and implement basic algorithms in Python**

In this lab, we will familiarize ourselves with functions, classes and other advanced constructs of python.

**Lab Tasks:**

1. Write a program to generate a dictionary that contains (i,sqrt(i)), where *i* is an integer between 1 and n. *n* is a number input by the user.

**from** math **import** sqrt  
  
a = int(input(**"enter a number"**))  
dic = {}  
**for** r **in** range(1,a+1):  
 dic [r] = sqrt(r)  
print(dic)

enter a number3

{1: 1.0, 2: 1.4142135623730951, 3: 1.7320508075688772}

1. Write a simple calculator program using functions add, sub, mul and div. The program should accepts two numbers and an operator and calls the corresponding function to perform the operation.

**import** math  
a = int(input(**"enter the 1st num"**))  
b = int(input(**"enter the 2nd num"**))  
op = input(**"enter the operator"**)  
**def** add(a,b):  
 z = a+b  
 **return** z  
**def** sub(a,b):  
 z = a-b  
 **return** z  
**def** mul(a,b):  
 z = a\*b  
 **return** z  
**def** div(a,b):  
 z = a/b  
 **return** z  
  
**if**(op == **"+"**):  
 z = add(a,b)  
 print(z)  
**if**(op == **"-"**):  
 z = sub(a,b)  
 print(z)  
**if**(op == **"\*"**):  
 z = mul(a,b)  
 print(z)  
**if**(op == **"/"**):  
 z = div(a,b)  
 print(z)

enter the 1st num 3

enter the 2nd num 4

enter the operator \*

12

1. Write a function that generates a list with values that are square of number between 1 and 20.

list = []  
  
**def** sqr(a):  
 p = a\*a  
 **return** p  
**for** r **in** range(1,21):  
 list.append(sqr(r))  
 print(**"the square of "**,r,**" is "**,list[r-1])

The square of 1 is 1

The square of 2 is 4

The square of 3 is 9

The square of 4 is 16

The square of 5 is 25

The square of 6 is 36

The square of 7 is 49

The square of 8 is 64

The square of 9 is 81

The square of 10 is 100

The square of 11 is 121

The square of 12 is 144

The square of 13 is 169

The square of 14 is 196

The square of 15 is 225

The square of 16 is 256

The square of 17 is 289

The square of 18 is 324

The square of 19 is 361

The square of 20 is 400

1. Define a class named Shape with static method printType. Define methods draw() and area(). Now define two class Rectangle and Triangle. Rectangle has two attributes length and width. The Triangle class has attributes a,b and c. Override the two methods of shape class. Demonstrate the functionality of class by creating its objects.

**class** Shape():  
 @staticmethod  
 **def** printType():  
 print(**"static method"**)  
 **def** draw(self):  
 print(**"draw"**)  
 **def** area(self):  
 print(**"area"**)  
  
**class** rectangle(Shape):  
 **def** \_\_init\_\_(self):  
 self.width = 23  
 self.length =34  
  
**class** triangle(Shape):  
 **def** \_\_init\_\_(self):  
 self.a = 2  
 self.b = 3  
 self.c = 4  
 **def** draw(self):  
 print(**"draw again2"**)  
 **def** area(self):  
 print(**"area again2"**)  
  
s = Shape()  
Shape.printType()  
t = triangle()  
r = rectangle()  
s.area()  
s.draw()  
r.draw()  
t.draw()

static method

area

draw

draw

draw again2

1. Using recursion, write a program to calculate the reverse of a string.

str = input(**"Enter the String"**)  
**def** rec(st):  
 **if**(st==**""**):  
 **return ""  
 else**:  
 **return** rec(st[1:])+st[0]  
a = rec(str)  
print(a)

Enter the String

Khan Sahib

bihaS nahK

**Lab 3: To study and implement Graph search algorithms in Python**

In this lab, we are going to implement searching algorithms in Python. There are two popular searching algorithms i.e. Depth First Search (Fig. 3a) and Breadth First Search (Fig 3b).

|  |
| --- |
| DFS(G,v) ( v is the vertex where the search starts )  Stack S := {}; ( start with an empty stack )  for each vertex u, set visited[u] := false;  push S, v;  while (S is not empty) do  u := pop S;  if (not visited[u]) then  visited[u] := true;  for each unvisited neighbour w of u  push S, w;  end if  end while  END DFS() |
| **3a:** Pseudo-code for Depth First Search |
| Breadth-First-Search(Graph, root):  create empty set S  create empty queue Q  root.parent = NIL  add root to S  Q.enqueue(root)  while Q is not empty:  current = Q.dequeue()  if current is the goal:  return current  for each node n that is adjacent to current:  if n is not in S:  add n to S  n.parent = current  Q.enqueue(n) |
| **3b:** Pseudo-code for Breadth First Search |

**Fig 3:** Pseudo-code for Graph Searching algorithms

**Lab Task:**

1. Provide the implementation of DFS and BFS algorithms in Python.

**BFS**

**import** queue  
x = [[0,1,1,1,1,0,0,0,0],[0,0,0,0,1,1,0,0,0],[0,0,0,0,0,1,1,0,0],[0,0,0,0,0,0,0,1,1],[0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0],[0,0,0,0,0,0,0,0,0]]  
frontier = queue.Queue()  
frontier.put(0)  
target = 6  
explored = []  
**while**(**True**):  
 **if**(frontier.empty()):  
 print(**"Target Does not Exist!"**)  
 **break** d = frontier.get()  
 **if**(d **in** explored):  
 **continue** explored.append(d)  
 **if**(d == target):  
 print(**"Node: "**,target,**" Found!!"**)  
 **break  
 for** y **in** range(0,9):  
 p = x[d][y]  
 **if**(p==1):  
 frontier.put(y)  
print(explored)

Node: 6 Found!!

[0, 1, 2, 3, 4, 5, 6]

**DFS:**

x = [[0,1,1,1,0,0,0,0,0],  
 [0,0,0,0,1,1,0,0,0],  
 [0,0,0,0,0,0,1,1,0],  
 [0,0,0,0,0,0,0,1,1],  
 [0,0,0,0,0,1,0,1,0],  
 [0,0,0,0,0,0,0,0,0],  
 [0,0,0,0,0,0,0,0,0],  
 [0,0,0,0,0,0,0,0,0],  
 [0,0,0,0,0,0,0,0,0]]  
**class** depth():  
 **def** DFS(i):  
 explored.append(i)  
 **for** j **in** range(0,9):  
 **if** x[i][j] == 1 **and** j **not in** explored:  
 depth.DFS(j)  
i=0  
j=0  
explored = []  
target = int(input(**"Enter Node Number: "**))  
depth.DFS(i)  
print(**"["**, end=**" "**)  
**for** k **in** range(0,len(explored)):  
 print(explored[k],end=**", "**)  
 **if**(explored[k]==target):  
 print(**"]"**, end=**" "**)  
 print(**"\nNode: "**, explored[k], **" Found!"**)  
 **break  
if** target **not in** explored:  
 print(**"Node not Found!"**)

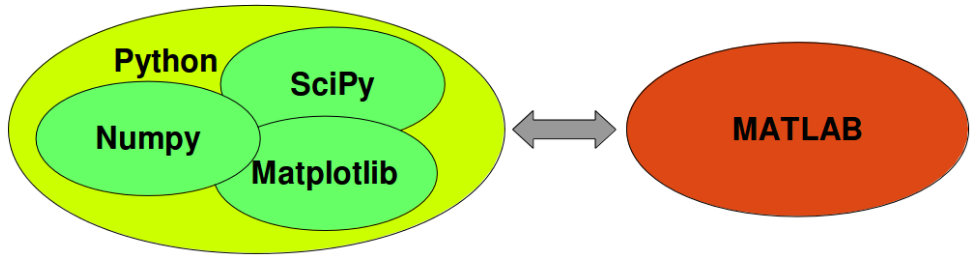
Enter Node Number: 7

[ 0, 1, 4, 5, 7, ]

Node: 7 Found!

**Lab 4: To study and understand numpy library**

In this lab, we are going to explore numpy. NumPy is an acronym for "Numeric Python" or "Numerical Python". It is an open source extension module for Python, which provides fast precompiled functions for mathematical and numerical routines.



**Lab Task:**

Open the Python Notebook provided with this lab and perform the tasks.

**Numpy On Python:**

**import** math **as** m  
**import** numpy **as** np  
**import** matplotlib.pyplot **as** plt  
x = np.linspace(1,2,100)  
y = np.sin(x)  
print(y)  
plt.scatter(x,y)  
plt.show()

[ 0.84147098 0.84688556 0.85221374 0.85745496 0.86260869 0.86767441

0.87265161 0.87753977 0.88233839 0.88704699 0.89166508 0.8961922

0.90062788 0.90497167 0.90922313 0.91338181 0.91744731 0.9214192

0.92529707 0.92908054 0.93276922 0.93636273 0.93986069 0.94326277

0.9465686 0.94977786 0.95289021 0.95590534 0.95882294 0.96164271

0.96436436 0.96698762 0.96951222 0.9719379 0.97426441 0.97649152

0.978619 0.98064663 0.98257421 0.98440153 0.98612842 0.98775469

0.98928018 0.99070474 0.99202822 0.99325047 0.99437139 0.99539085

0.99630876 0.99712501 0.99783952 0.99845223 0.99896306 0.99937197

0.99967891 0.99988386 0.99998679 0.99998769 0.99988657 0.99968342

0.99937828 0.99897117 0.99846214 0.99785123 0.99713852 0.99632407

0.99540796 0.99439029 0.99327116 0.99205069 0.99072901 0.98930624

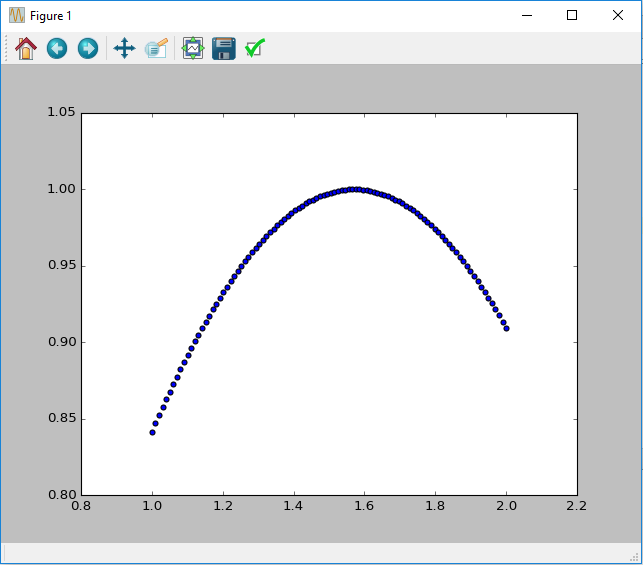
0.98778253 0.98615803 0.98443293 0.98260737 0.98068157 0.9786557

0.97652999 0.97430464 0.97197988 0.96955595 0.9670331 0.96441158

0.96169166 0.95887362 0.95595775 0.95294434 0.94983371 0.94662616

0.94332203 0.93992165 0.93642537 0.93283355 0.92914656 0.92536476

0.92148855 0.91751832 0.91345447 0.90929743]



**import** numpy **as** np  
z = np.linspace(1,5,10)  
print(z)

[ 1. 1.44444444 1.88888889 2.33333333 2.77777778 3.22222222

3.66666667 4.11111111 4.55555556 5. ]

**import** numpy **as** np  
a = np.arange(18)  
a = a.reshape(3, 3, 2)  
print(a)  
print(a.ndim)  
print(a.dtype.name)  
print(a.itemsize)  
print(a.size)

[[[ 0 1]

[ 2 3]

[ 4 5]]

[[ 6 7]

[ 8 9]

[10 11]]

[[12 13]

[14 15]

[16 17]]]

3

int32

4

18

**import** numpy **as** np  
b = np.array([(1.5,2,3),(4,5,6)])  
print(b)  
c = np.zeros((3,4))  
print(c)  
d = np.ones((3,4))  
print(d)  
e = np.ones((3,5))  
e = e\*6  
print(e)  
f = np.empty((3,4))  
print(f)  
b = np.linspace(0 ,2, 9)  
print(b)  
g = np.sin(b)  
print(g)

[[ 1.5 2. 3. ]

[ 4. 5. 6. ]]

[[ 0. 0. 0. 0.]

[ 0. 0. 0. 0.]

[ 0. 0. 0. 0.]]

[[ 1. 1. 1. 1.]

[ 1. 1. 1. 1.]

[ 1. 1. 1. 1.]]

[[ 6. 6. 6. 6. 6.]

[ 6. 6. 6. 6. 6.]

[ 6. 6. 6. 6. 6.]]

[[ 0. 0. 0. 0.]

[ 0. 0. 0. 0.]

[ 0. 0. 0. 0.]]

[ 0. 0.25 0.5 0.75 1. 1.25 1.5 1.75 2. ]

[ 0. 0.24740396 0.47942554 0.68163876 0.84147098 0.94898462

0.99749499 0.98398595 0.90929743]

**Numpy on jupyter Notebook:**

a. Import the "numpy" library as "np".

a. Import the "numpy" library as "np".

x

import numpy as np

×

. . .



b. Create an array of shape (2, 3, 4) of zeros.

b. Create an array of shape (2, 3, 4) of zeros.

In [2]:

a = np.zeros((2,3,4))

print(a)

×

[[[ 0. 0. 0. 0.]

[ 0. 0. 0. 0.]

[ 0. 0. 0. 0.]]

[[ 0. 0. 0. 0.]

[ 0. 0. 0. 0.]

[ 0. 0. 0. 0.]]]

. . .

c. Create an array of shape (2, 3, 4) of ones

c. Create an array of shape (2, 3, 4) of ones

a = np.ones((2,3,4))

print(a)

×

[[[ 1. 1. 1. 1.]

[ 1. 1. 1. 1.]

[ 1. 1. 1. 1.]]

[[ 1. 1. 1. 1.]

[ 1. 1. 1. 1.]

[ 1. 1. 1. 1.]]]

. . .

d. Create an array with values 0 to 999 using the "np.arange" function

d. Create an array with values 0 to 999 using the "np.arange" function

a = np.arange(0,1000)

print(a)

×

[ 0 1 2 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]

e. Create an array from the list [2, 3.2, 5.5, -6.4, -2.2, 2.4] and assign it to the variable "a"

e. Create an array from the list [2, 3.2, 5.5, -6.4, -2.2, 2.4] and assign it to the variable "a"

a = np.array([2,3.2,5.5,-6.4,-2.2,2.4])

print(a)

×

[ 2. 3.2 5.5 -6.4 -2.2 2.4]

f. Do you know what a[1] will equal? Print it to see

f. Do you know what a[1] will equal? Print it to see

print(a[1])

×

3.2

g. Try printing a[1:4] to see what that equals

g. Try printing a[1:4] to see what that equals

print(a[1:4])

×

[ 3.2 5.5 -6.4]

h. Create a 2-D array from the following list and assign it to the variable "a":

[[2, 3.2, 5.5, -6.4, -2.2, 2.4],

[1, 22, 4, 0.1, 5.3, -9],

[3, 1, 2.1, 21, 1.1, -2]]

h. Create a 2-D array from the following list and assign it to the variable "a": [[2, 3.2, 5.5, -6.4, -2.2, 2.4], [1, 22, 4, 0.1, 5.3, -9], [3, 1, 2.1, 21, 1.1, -2]]

b = np.array([[2,3.2,5.5,-6.4,-2.2,2.4],[1,22,4,0.1,5.3,-9],[3,1,2.1,21,1.1,-2]])

print(b)

×

[[ 2. 3.2 5.5 -6.4 -2.2 2.4]

[ 1. 22. 4. 0.1 5.3 -9. ]

[ 3. 1. 2.1 21. 1.1 -2. ]]

i. Can you guess what the following slices are equal to? Print them to check your understanding.

a[:, 3]

a[1:4, 0:4]

a[1:, 2]

i. Can you guess what the following slices are equal to? Print them to check your understanding. a[:, 3] a[1:4, 0:4] a[1:, 2]

print(b[:,3])

print(b[1:3,0:4])

print(b[1:,2])

×

[ -6.4 0.1 21. ]

[[ 1. 22. 4. 0.1]

[ 3. 1. 2.1 21. ]]

[ 4. 2.1]

j. Create a 2-D array of shape (2, 4) containing two lists (range(4), range(10, 14))

and assign it to the variable "arr".Print the shape of the array. Print the size of the array. Print the maximum and minimum of the array

j. Create a 2-D array of shape (2, 4) containing two lists (range(4), range(10, 14)) and assign it to the variable "arr".Print the shape of the array. Print the size of the array. Print the maximum and minimum of the array

arr = np.array([range(0,4),range(10,14)])

print(arr)

print("Shape: ",arr.shape)

print("size: ",arr.size)

×

[[ 0 1 2 3]

[10 11 12 13]]

Shape: (2, 4)

size: 8

. . .

print("Maximum Value: ",np.max(arr))

×

Maximum Value: 13

. . .

print("Minimum Value: ",np.min(arr))

×

Minimum Value: 0

k. Continue to use the array "arr" as defined above.Print the array re-shaped to (2, 2, 2).Print the array transposed.Print the array flattened to a single dimension. Print the array converted to floats.

k. Continue to use the array "arr" as defined above.Print the array re-shaped to (2, 2, 2).Print the array transposed.Print the array flattened to a single dimension. Print the array converted to floats.

arr = arr.reshape((2,2,2))

print(np.transpose(arr))

print(arr.astype(float))

×

[[[ 0 10]

[ 2 12]]

[[ 1 11]

[ 3 13]]]

[[[ 0. 1.]

[ 2. 3.]]

[[ 10. 11.]

[ 12. 13.]]]

l. Create an an array counting from 1 to 20 inclusive

l. Create an an array counting from 1 to 20 inclusive

a = np. arange(1,21)

print(a)

×

[ 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20]

m. The array of multiples of 3 greater than 0 and less than 30

m. The array of multiples of 3 greater than 0 and less than 30

c = np.arange(9)+1

c = c\*3

print(c)

×

[ 3 6 9 12 15 18 21 24 27]

n. The array of 8 equally spaced floats x where 0 ≤ x ≤ 1

n. The array of 8 equally spaced floats x where 0 ≤ x ≤ 1

d = np.linspace(0,1,8)

print(d)

×

[ 0. 0.14285714 0.28571429 0.42857143 0.57142857 0.71428571

0.85714286 1

o. Use np.arange and reshape to create the array A = [[1 2 3 4] [5 6 7 8]]

o. Use np.arange and reshape to create the array A = [[1 2 3 4] [5 6 7 8]]

a = np.arange(8)+1

A = a.reshape(2,4)

print(A)

×

[[1 2 3 4]

[5 6 7 8]]

p. Use np.array to create the array B = [1 2]

p. Use np.array to create the array B = [1 2]

B = np.array([1,2])

print(B)

×

[1 2]

q. Use broadcasting to add B to A to create the final array A + B

q. Use broadcasting to add B to A to create the final array A + B

a = np.arange(8)+1

A = a.reshape(2,4)

print(A)

×

[[1 2 3 4]

[5 6 7 8]]

. . .

B = 1

c = A+B

print(c)

×

[[2 3 4 5]

[6 7 8 9]]

**Lab 5: To study and implement pandas library**

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with “relational” or “labeled” data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.

Open the Python Notebook provided with this lab and perform the tasks.

1. Create a data series with marks of students : 75, 80, 79, 60[¶](http://localhost:8888/notebooks/pandas%20(1).ipynb#1.-Create-a-data-series-with-marks-of-students-:-75,-80,-79,-60)

import pandas as pd;

pd.Series([75,80,79,60])

​Out[2]:

0 75

1 80

2 79

3 60

### 2. Create a data frame with name of students, id and marks

2. Create a data frame with name of students, id and marks[¶](http://localhost:8888/notebooks/pandas%20(1).ipynb#2.-Create-a-data-frame-with-name-of-students,-id-and-marks)

pd.DataFrame(

{"name":["Rashid","Sabahat","Usman"],

"marks":[5,2,20],

"Id":[123,345,534]})

×

Out[3]:

|  | Id | marks | name |
| --- | --- | --- | --- |
| 0 | 123 | 5 | Rashid |
| 1 | 345 | 2 | Sabahat |
| 2 | 534 | 20 | Usman |

.

### 3. Now read the file 'data.csv' in panda

3. Now read the file 'data.csv' in panda[¶](http://localhost:8888/notebooks/pandas%20(1).ipynb#3.-Now-read-the-file-'data.csv'-in-panda)

r=pd.read\_csv("data.csv")

print(r)

×

Student Code Degree Student Name Mid Quiz 1 \

0 022-14-19987 BS(CS) Abdul Basit 28 8.0

1 022-14-110233 BS(CS) Adeel Ahmed 17 NaN

2 022-14-110585 BS(CS) Afrah Zareen 18 5.0

3 022-14-19718 BS(CS) Ahmed Ali Raza 14 7.0

4 022-14-110648 BS(CS) Ahsan Ali Vohra 27 7.0

5 022-14-110232 BS(CS) Ameer Hamza 25 9.0

6 022-14-110588 BS(CS) Anas Ali Khan 28 5.0

7 022-14-110388 BS(CS) Aneebullah Niazi 26 9.0

8 022-14-110601 BS(CS) Areesha Sohail 19 9.0

9 022-14-110599 BS(CS) Arsalan 28 8.0

10 022-14-110214 BS(CS) Fatima Haider Warsi 30 8.0

11 022-14-110591 BS(CS) Habib Ullah 28 8.0

12 022-15-110994 BS(CS) Hafiza Tooba Akbani 23 7.0

13 022-14-110600 BS(CS) Hamza Abdul Jabbar 24 8.0

14 022-14-110389 BS(CS) Hareem Afshan 21 7.0

15 022-14-19916 BS(CS) Haseeb Sajid 18 NaN

16 022-14-110596 BS(CS) Hassam Ahmed 23 5.0

17 022-14-110396 BS(CS) Khalid Anwer 20 8.0

18 022-14-110223 BS(CS) Madiha Jabeen 16 5.0

19 022-14-110222 BS(CS) Mohammad Hunain 27 9.0

20 022-14-110412 BS(CS) Muhammad Aamir 24 7.0

21 022-14-110593 BS(CS) Muhammad Abdul Rehman Siddiqui 20 9.0

22 022-14-110398 BS(CS) Muhammad Abdullah 20 7.0

23 022-14-19983 BS(CS) Muhammad Ali Iqbal 26 5.0

24 022-14-110215 BS(CS) Muhammad Bilal 28 5.0

25 022-14-110105 BS(CS) Muhammad Faraz 26 9.0

26 022-14-110370 BS(CS) Muhammad Ghazali Faridi 27 6.0

27 022-14-110452 BS(CS) Muhammad Osama Khan 27 8.0

28 022-14-110387 BS(CS) Muhammad Saqib Intizar 22 6.0

29 022-14-110217 BS(CS) Muhammad Shahroz Khurshid 17 7.0

30 022-14-110401 BS(CS) Muhammad Shozab 23 8.0

31 022-14-110231 BS(CS) Muhammad Taha Hasnain 27 7.0

32 022-14-110035 BS(CS) Muhammad Wajahat Khan 9 NaN

33 022-14-19919 BS(CS) Muhammad Younus Baig 20 4.0

34 022-14-110413 BS(CS) Nazeer Bin Zafar 25 9.0

35 022-14-19923 BS(CS) Rabi Ahmed 20 8.0

36 022-14-110582 BS(CS) Rida Nasim 25 9.0

37 022-14-110230 BS(CS) Sadaf Nosheen 27 9.0

38 022-14-110229 BS(CS) Sadaquat Rafique 9 6.0

39 022-14-110107 BS(CS) Sania Iqbal 28 4.0

40 022-14-110584 BS(CS) Sharif Taqi 27 9.0

41 022-14-110225 BS(CS) Shariqa Ahmad 20 7.0

42 022-14-110587 BS(CS) Sumbul Rehman 28 5.0

43 022-14-110451 BS(CS) Syed Faizan Uddin 28 9.0

44 022-14-110589 BS(CS) Syed Sohaib 25 7.0

45 022-14-110400 BS(CS) Syeda Sabika Raza 27 9.0

46 022-14-19911 BS(CS) Usman Khan 25 8.0

47 022-14-110219 BS(CS) Waqar Ahmed 11 9.0

Quiz 2 Best of Quizzes Assignment 1 Assignment 2 Best of Assignments \

0 3.0 8 7.0 9.0 9

1 5.0 5 8.0 10.0 10

2 2.0 5 8.0 10.0 10

3 2.0 7 NaN 2.0 2

4 6.0 7 7.0 9.0 9

5 6.0 9 8.0 10.0 10

6 6.0 6 8.0 10.0 10

7 6.0 9 8.0 10.0 10

8 4.0 9 7.0 9.0 9

9 6.0 8 8.0 NaN 8

10 7.0 8 8.0 NaN 8

11 5.0 8 5.0 NaN 5

12 5.0 7 8.0 10.0 10

13 4.0 8 8.0 10.0 10

14 4.0 7 6.0 8.0 8

15 NaN 0 NaN NaN 0

16 5.0 5 7.0 9.0 9

17 5.0 8 9.0 11.0 11

18 2.0 5 8.0 10.0 10

19 6.0 9 8.0 10.0 10

20 6.0 7 7.0 9.0 9

21 6.0 9 9.0 11.0 11

22 6.0 7 7.0 NaN 7

23 NaN 5 8.0 10.0 10

24 4.0 5 9.0 11.0 11

25 6.0 9 8.0 10.0 10

26 NaN 6 6.0 8.0 8

27 6.0 8 8.0 10.0 10

28 6.0 6 9.0 11.0 11

29 1.0 7 6.0 NaN 6

30 6.0 8 9.0 NaN 9

31 5.0 7 10.0 NaN 10

32 2.0 2 8.0 10.0 10

33 2.0 4 8.0 10.0 10

34 6.0 9 8.0 10.0 10

35 5.0 8 NaN 10.0 10

36 5.0 9 9.0 NaN 9

37 5.0 9 9.0 NaN 9

38 6.0 6 5.0 7.0 7

39 4.0 4 9.0 11.0 11

40 6.0 9 8.0 10.0 10

41 6.0 7 8.0 10.0 10

42 6.0 6 8.0 10.0 10

43 4.0 9 6.0 8.0 8

44 5.0 7 9.0 NaN 9

45 6.0 9 9.0 NaN 9

46 5.0 8 8.0 10.0 10

47 5.0 9 5.0 7.0 7

Total Sessional (50) Final (50) Total (100) Grade

0 45 25.0 70 B

1 32 18.0 50 F

2 33 30.0 63 C

3 23 23.0 46 F

4 43 34.0 77 B

5 44 27.0 71 B

6 44 30.0 74 B

7 45 40.0 85 A

8 37 24.0 61 C

9 44 40.0 84 A

10 46 45.0 91 A

11 41 35.0 76 B

12 40 33.0 73 B

13 42 25.0 67 C

14 36 29.0 65 C

15 18 NaN 18 F

16 37 35.0 72 B

17 39 31.0 70 B

18 31 29.0 60 C

19 46 45.0 91 A

20 40 31.0 71 B

21 40 20.0 60 C

22 34 23.0 57 F

23 41 21.0 62 C

24 44 33.0 77 B

25 45 41.0 86 A

26 41 24.0 65 C

27 45 26.0 71 B

28 39 18.0 57 F

29 30 9.0 39 F

30 40 32.0 72 B

31 44 29.0 73 B

32 21 8.0 29 F

33 34 17.0 51 F

34 44 42.0 86 A

35 38 28.0 66 C

36 43 27.0 70 B

37 45 30.0 75 B

38 22 17.0 39 F

39 43 25.0 68 C

40 46 35.0 81 A

41 37 20.0 57 F

42 44 23.0 67 C

43 45 34.0 79 B

44 41 22.0 63 C

45 45 35.0 80 A

46 43 22.0 65 C

47 27 19.0 46 F

### 4. What are the columns in the dataframe?

4. What are the columns in the dataframe?[¶](http://localhost:8888/notebooks/pandas%20(1).ipynb#4.-What-are-the-columns-in-the-dataframe?)

r.columns

×

Out[5]:

Index(['Student Code', 'Degree', 'Student Name', 'Mid', 'Quiz 1', 'Quiz 2',

'Best of Quizzes', 'Assignment 1', 'Assignment 2',

'Best of Assignments', 'Total Sessional (50)', 'Final (50)',

'Total (100)', 'Grade'],

dtype='object')

### 5. Sort the data based on Marks obtained. Fill all the 'na' cells with 0

5. Sort the data based on Marks obtained. Fill all the 'na' cells with 0[¶](http://localhost:8888/notebooks/pandas%20(1).ipynb#5.-Sort-the-data-based-on-Marks-obtained.-Fill-all-the-'na'-cells-with-0)

s = r.fillna(0)

s.sort(['Total (100)'])

×

C:\Users\User\Anaconda3\lib\site-packages\ipykernel\\_\_main\_\_.py:2: FutureWarning: sort(columns=....) is deprecated, use sort\_values(by=.....)

from ipykernel import kernelapp as app

Out[7]:

|  | Student Code | Degree | Student Name | Mid | Quiz 1 | Quiz 2 | Best of Quizzes | Assignment 1 | Assignment 2 | Best of Assignments | Total Sessional (50) | Final (50) | Total (100) | Grade |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 15 | 022-14-19916 | BS(CS) | Haseeb Sajid | 18 | 0.0 | 0.0 | 0 | 0.0 | 0.0 | 0 | 18 | 0.0 | 18 | F |
| 32 | 022-14-110035 | BS(CS) | Muhammad Wajahat Khan | 9 | 0.0 | 2.0 | 2 | 8.0 | 10.0 | 10 | 21 | 8.0 | 29 | F |
| 38 | 022-14-110229 | BS(CS) | Sadaquat Rafique | 9 | 6.0 | 6.0 | 6 | 5.0 | 7.0 | 7 | 22 | 17.0 | 39 | F |
| 29 | 022-14-110217 | BS(CS) | Muhammad Shahroz Khurshid | 17 | 7.0 | 1.0 | 7 | 6.0 | 0.0 | 6 | 30 | 9.0 | 39 | F |
| 47 | 022-14-110219 | BS(CS) | Waqar Ahmed | 11 | 9.0 | 5.0 | 9 | 5.0 | 7.0 | 7 | 27 | 19.0 | 46 | F |
| 3 | 022-14-19718 | BS(CS) | Ahmed Ali Raza | 14 | 7.0 | 2.0 | 7 | 0.0 | 2.0 | 2 | 23 | 23.0 | 46 | F |
| 1 | 022-14-110233 | BS(CS) | Adeel Ahmed | 17 | 0.0 | 5.0 | 5 | 8.0 | 10.0 | 10 | 32 | 18.0 | 50 | F |
| 33 | 022-14-19919 | BS(CS) | Muhammad Younus Baig | 20 | 4.0 | 2.0 | 4 | 8.0 | 10.0 | 10 | 34 | 17.0 | 51 | F |
| 41 | 022-14-110225 | BS(CS) | Shariqa Ahmad | 20 | 7.0 | 6.0 | 7 | 8.0 | 10.0 | 10 | 37 | 20.0 | 57 | F |
| 28 | 022-14-110387 | BS(CS) | Muhammad Saqib Intizar | 22 | 6.0 | 6.0 | 6 | 9.0 | 11.0 | 11 | 39 | 18.0 | 57 | F |
| 22 | 022-14-110398 | BS(CS) | Muhammad Abdullah | 20 | 7.0 | 6.0 | 7 | 7.0 | 0.0 | 7 | 34 | 23.0 | 57 | F |
| 21 | 022-14-110593 | BS(CS) | Muhammad Abdul Rehman Siddiqui | 20 | 9.0 | 6.0 | 9 | 9.0 | 11.0 | 11 | 40 | 20.0 | 60 | C |
| 18 | 022-14-110223 | BS(CS) | Madiha Jabeen | 16 | 5.0 | 2.0 | 5 | 8.0 | 10.0 | 10 | 31 | 29.0 | 60 | C |
| 8 | 022-14-110601 | BS(CS) | Areesha Sohail | 19 | 9.0 | 4.0 | 9 | 7.0 | 9.0 | 9 | 37 | 24.0 | 61 | C |
| 23 | 022-14-19983 | BS(CS) | Muhammad Ali Iqbal | 26 | 5.0 | 0.0 | 5 | 8.0 | 10.0 | 10 | 41 | 21.0 | 62 | C |
| 44 | 022-14-110589 | BS(CS) | Syed Sohaib | 25 | 7.0 | 5.0 | 7 | 9.0 | 0.0 | 9 | 41 | 22.0 | 63 | C |
| 2 | 022-14-110585 | BS(CS) | Afrah Zareen | 18 | 5.0 | 2.0 | 5 | 8.0 | 10.0 | 10 | 33 | 30.0 | 63 | C |
| 14 | 022-14-110389 | BS(CS) | Hareem Afshan | 21 | 7.0 | 4.0 | 7 | 6.0 | 8.0 | 8 | 36 | 29.0 | 65 | C |
| 46 | 022-14-19911 | BS(CS) | Usman Khan | 25 | 8.0 | 5.0 | 8 | 8.0 | 10.0 | 10 | 43 | 22.0 | 65 | C |
| 26 | 022-14-110370 | BS(CS) | Muhammad Ghazali Faridi | 27 | 6.0 | 0.0 | 6 | 6.0 | 8.0 | 8 | 41 | 24.0 | 65 | C |
| 35 | 022-14-19923 | BS(CS) | Rabi Ahmed | 20 | 8.0 | 5.0 | 8 | 0.0 | 10.0 | 10 | 38 | 28.0 | 66 | C |
| 13 | 022-14-110600 | BS(CS) | Hamza Abdul Jabbar | 24 | 8.0 | 4.0 | 8 | 8.0 | 10.0 | 10 | 42 | 25.0 | 67 | C |
| 42 | 022-14-110587 | BS(CS) | Sumbul Rehman | 28 | 5.0 | 6.0 | 6 | 8.0 | 10.0 | 10 | 44 | 23.0 | 67 | C |
| 39 | 022-14-110107 | BS(CS) | Sania Iqbal | 28 | 4.0 | 4.0 | 4 | 9.0 | 11.0 | 11 | 43 | 25.0 | 68 | C |
| 36 | 022-14-110582 | BS(CS) | Rida Nasim | 25 | 9.0 | 5.0 | 9 | 9.0 | 0.0 | 9 | 43 | 27.0 | 70 | B |
| 0 | 022-14-19987 | BS(CS) | Abdul Basit | 28 | 8.0 | 3.0 | 8 | 7.0 | 9.0 | 9 | 45 | 25.0 | 70 | B |
| 17 | 022-14-110396 | BS(CS) | Khalid Anwer | 20 | 8.0 | 5.0 | 8 | 9.0 | 11.0 | 11 | 39 | 31.0 | 70 | B |
| 20 | 022-14-110412 | BS(CS) | Muhammad Aamir | 24 | 7.0 | 6.0 | 7 | 7.0 | 9.0 | 9 | 40 | 31.0 | 71 | B |
| 27 | 022-14-110452 | BS(CS) | Muhammad Osama Khan | 27 | 8.0 | 6.0 | 8 | 8.0 | 10.0 | 10 | 45 | 26.0 | 71 | B |
| 5 | 022-14-110232 | BS(CS) | Ameer Hamza | 25 | 9.0 | 6.0 | 9 | 8.0 | 10.0 | 10 | 44 | 27.0 | 71 | B |
| 16 | 022-14-110596 | BS(CS) | Hassam Ahmed | 23 | 5.0 | 5.0 | 5 | 7.0 | 9.0 | 9 | 37 | 35.0 | 72 | B |
| 30 | 022-14-110401 | BS(CS) | Muhammad Shozab | 23 | 8.0 | 6.0 | 8 | 9.0 | 0.0 | 9 | 40 | 32.0 | 72 | B |
| 12 | 022-15-110994 | BS(CS) | Hafiza Tooba Akbani | 23 | 7.0 | 5.0 | 7 | 8.0 | 10.0 | 10 | 40 | 33.0 | 73 | B |
| 31 | 022-14-110231 | BS(CS) | Muhammad Taha Hasnain | 27 | 7.0 | 5.0 | 7 | 10.0 | 0.0 | 10 | 44 | 29.0 | 73 | B |
| 6 | 022-14-110588 | BS(CS) | Anas Ali Khan | 28 | 5.0 | 6.0 | 6 | 8.0 | 10.0 | 10 | 44 | 30.0 | 74 | B |
| 37 | 022-14-110230 | BS(CS) | Sadaf Nosheen | 27 | 9.0 | 5.0 | 9 | 9.0 | 0.0 | 9 | 45 | 30.0 | 75 | B |
| 11 | 022-14-110591 | BS(CS) | Habib Ullah | 28 | 8.0 | 5.0 | 8 | 5.0 | 0.0 | 5 | 41 | 35.0 | 76 | B |
| 4 | 022-14-110648 | BS(CS) | Ahsan Ali Vohra | 27 | 7.0 | 6.0 | 7 | 7.0 | 9.0 | 9 | 43 | 34.0 | 77 | B |
| 24 | 022-14-110215 | BS(CS) | Muhammad Bilal | 28 | 5.0 | 4.0 | 5 | 9.0 | 11.0 | 11 | 44 | 33.0 | 77 | B |
| 43 | 022-14-110451 | BS(CS) | Syed Faizan Uddin | 28 | 9.0 | 4.0 | 9 | 6.0 | 8.0 | 8 | 45 | 34.0 | 79 | B |
| 45 | 022-14-110400 | BS(CS) | Syeda Sabika Raza | 27 | 9.0 | 6.0 | 9 | 9.0 | 0.0 | 9 | 45 | 35.0 | 80 | A |
| 40 | 022-14-110584 | BS(CS) | Sharif Taqi | 27 | 9.0 | 6.0 | 9 | 8.0 | 10.0 | 10 | 46 | 35.0 | 81 | A |
| 9 | 022-14-110599 | BS(CS) | Arsalan | 28 | 8.0 | 6.0 | 8 | 8.0 | 0.0 | 8 | 44 | 40.0 | 84 | A |
| 7 | 022-14-110388 | BS(CS) | Aneebullah Niazi | 26 | 9.0 | 6.0 | 9 | 8.0 | 10.0 | 10 | 45 | 40.0 | 85 | A |
| 34 | 022-14-110413 | BS(CS) | Nazeer Bin Zafar | 25 | 9.0 | 6.0 | 9 | 8.0 | 10.0 | 10 | 44 | 42.0 | 86 | A |
| 25 | 022-14-110105 | BS(CS) | Muhammad Faraz | 26 | 9.0 | 6.0 | 9 | 8.0 | 10.0 | 10 | 45 | 41.0 | 86 | A |
| 19 | 022-14-110222 | BS(CS) | Mohammad Hunain | 27 | 9.0 | 6.0 | 9 | 8.0 | 10.0 | 10 | 46 | 45.0 | 91 | A |
| 10 | 022-14-110214 | BS(CS) | Fatima Haider Warsi | 30 | 8.0 | 7.0 | 8 | 8.0 | 0.0 | 8 | 46 | 45.0 | 91 | A |

### 6. Display the top 10 rows

6. Display the top 10 rows[¶](http://localhost:8888/notebooks/pandas%20(1).ipynb#6.-Display-the-top-10-rows)

r.head(10)

×

Out[34]:

|  | Student Code | Degree | Student Name | Mid | Quiz 1 | Quiz 2 | Best of Quizzes | Assignment 1 | Assignment 2 | Best of Assignments | Total Sessional (50) | Final (50) | Total (100) | Grade |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 022-14-19987 | BS(CS) | Abdul Basit | 28 | 8.0 | 3.0 | 8 | 7.0 | 9.0 | 9 | 45 | 25.0 | 70 | B |
| 1 | 022-14-110233 | BS(CS) | Adeel Ahmed | 17 | NaN | 5.0 | 5 | 8.0 | 10.0 | 10 | 32 | 18.0 | 50 | F |
| 2 | 022-14-110585 | BS(CS) | Afrah Zareen | 18 | 5.0 | 2.0 | 5 | 8.0 | 10.0 | 10 | 33 | 30.0 | 63 | C |
| 3 | 022-14-19718 | BS(CS) | Ahmed Ali Raza | 14 | 7.0 | 2.0 | 7 | NaN | 2.0 | 2 | 23 | 23.0 | 46 | F |
| 4 | 022-14-110648 | BS(CS) | Ahsan Ali Vohra | 27 | 7.0 | 6.0 | 7 | 7.0 | 9.0 | 9 | 43 | 34.0 | 77 | B |
| 5 | 022-14-110232 | BS(CS) | Ameer Hamza | 25 | 9.0 | 6.0 | 9 | 8.0 | 10.0 | 10 | 44 | 27.0 | 71 | B |
| 6 | 022-14-110588 | BS(CS) | Anas Ali Khan | 28 | 5.0 | 6.0 | 6 | 8.0 | 10.0 | 10 | 44 | 30.0 | 74 | B |
| 7 | 022-14-110388 | BS(CS) | Aneebullah Niazi | 26 | 9.0 | 6.0 | 9 | 8.0 | 10.0 | 10 | 45 | 40.0 | 85 | A |
| 8 | 022-14-110601 | BS(CS) | Areesha Sohail | 19 | 9.0 | 4.0 | 9 | 7.0 | 9.0 | 9 | 37 | 24.0 | 61 | C |
| 9 | 022-14-110599 | BS(CS) | Arsalan | 28 | 8.0 | 6.0 | 8 | 8.0 | NaN | 8 | 44 | 40.0 | 84 | A |

### 7. Display the last 10 rows

7. Display the last 10 rows[¶](http://localhost:8888/notebooks/pandas%20(1).ipynb#7.-Display-the-last-10-rows)

r.tail(10)

×

Out[35]:

|  | Student Code | Degree | Student Name | Mid | Quiz 1 | Quiz 2 | Best of Quizzes | Assignment 1 | Assignment 2 | Best of Assignments | Total Sessional (50) | Final (50) | Total (100) | Grade |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 38 | 022-14-110229 | BS(CS) | Sadaquat Rafique | 9 | 6.0 | 6.0 | 6 | 5.0 | 7.0 | 7 | 22 | 17.0 | 39 | F |
| 39 | 022-14-110107 | BS(CS) | Sania Iqbal | 28 | 4.0 | 4.0 | 4 | 9.0 | 11.0 | 11 | 43 | 25.0 | 68 | C |
| 40 | 022-14-110584 | BS(CS) | Sharif Taqi | 27 | 9.0 | 6.0 | 9 | 8.0 | 10.0 | 10 | 46 | 35.0 | 81 | A |
| 41 | 022-14-110225 | BS(CS) | Shariqa Ahmad | 20 | 7.0 | 6.0 | 7 | 8.0 | 10.0 | 10 | 37 | 20.0 | 57 | F |
| 42 | 022-14-110587 | BS(CS) | Sumbul Rehman | 28 | 5.0 | 6.0 | 6 | 8.0 | 10.0 | 10 | 44 | 23.0 | 67 | C |
| 43 | 022-14-110451 | BS(CS) | Syed Faizan Uddin | 28 | 9.0 | 4.0 | 9 | 6.0 | 8.0 | 8 | 45 | 34.0 | 79 | B |
| 44 | 022-14-110589 | BS(CS) | Syed Sohaib | 25 | 7.0 | 5.0 | 7 | 9.0 | NaN | 9 | 41 | 22.0 | 63 | C |
| 45 | 022-14-110400 | BS(CS) | Syeda Sabika Raza | 27 | 9.0 | 6.0 | 9 | 9.0 | NaN | 9 | 45 | 35.0 | 80 | A |
| 46 | 022-14-19911 | BS(CS) | Usman Khan | 25 | 8.0 | 5.0 | 8 | 8.0 | 10.0 | 10 | 43 | 22.0 | 65 | C |
| 47 | 022-14-110219 | BS(CS) | Waqar Ahmed | 11 | 9.0 | 5.0 | 9 | 5.0 | 7.0 | 7 | 27 | 19.0 | 46 | F |

### 8. Display only the odd rows

8. Display only the odd rows[¶](http://localhost:8888/notebooks/pandas%20(1).ipynb#8.-Display-only-the-odd-rows)

r[1::2]

×

Out[39]:

|  | Student Code | Degree | Student Name | Mid | Quiz 1 | Quiz 2 | Best of Quizzes | Assignment 1 | Assignment 2 | Best of Assignments | Total Sessional (50) | Final (50) | Total (100) | Grade |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 022-14-110233 | BS(CS) | Adeel Ahmed | 17 | NaN | 5.0 | 5 | 8.0 | 10.0 | 10 | 32 | 18.0 | 50 | F |
| 3 | 022-14-19718 | BS(CS) | Ahmed Ali Raza | 14 | 7.0 | 2.0 | 7 | NaN | 2.0 | 2 | 23 | 23.0 | 46 | F |
| 5 | 022-14-110232 | BS(CS) | Ameer Hamza | 25 | 9.0 | 6.0 | 9 | 8.0 | 10.0 | 10 | 44 | 27.0 | 71 | B |
| 7 | 022-14-110388 | BS(CS) | Aneebullah Niazi | 26 | 9.0 | 6.0 | 9 | 8.0 | 10.0 | 10 | 45 | 40.0 | 85 | A |
| 9 | 022-14-110599 | BS(CS) | Arsalan | 28 | 8.0 | 6.0 | 8 | 8.0 | NaN | 8 | 44 | 40.0 | 84 | A |
| 11 | 022-14-110591 | BS(CS) | Habib Ullah | 28 | 8.0 | 5.0 | 8 | 5.0 | NaN | 5 | 41 | 35.0 | 76 | B |
| 13 | 022-14-110600 | BS(CS) | Hamza Abdul Jabbar | 24 | 8.0 | 4.0 | 8 | 8.0 | 10.0 | 10 | 42 | 25.0 | 67 | C |
| 15 | 022-14-19916 | BS(CS) | Haseeb Sajid | 18 | NaN | NaN | 0 | NaN | NaN | 0 | 18 | NaN | 18 | F |
| 17 | 022-14-110396 | BS(CS) | Khalid Anwer | 20 | 8.0 | 5.0 | 8 | 9.0 | 11.0 | 11 | 39 | 31.0 | 70 | B |
| 19 | 022-14-110222 | BS(CS) | Mohammad Hunain | 27 | 9.0 | 6.0 | 9 | 8.0 | 10.0 | 10 | 46 | 45.0 | 91 | A |
| 21 | 022-14-110593 | BS(CS) | Muhammad Abdul Rehman Siddiqui | 20 | 9.0 | 6.0 | 9 | 9.0 | 11.0 | 11 | 40 | 20.0 | 60 | C |
| 23 | 022-14-19983 | BS(CS) | Muhammad Ali Iqbal | 26 | 5.0 | NaN | 5 | 8.0 | 10.0 | 10 | 41 | 21.0 | 62 | C |
| 25 | 022-14-110105 | BS(CS) | Muhammad Faraz | 26 | 9.0 | 6.0 | 9 | 8.0 | 10.0 | 10 | 45 | 41.0 | 86 | A |
| 27 | 022-14-110452 | BS(CS) | Muhammad Osama Khan | 27 | 8.0 | 6.0 | 8 | 8.0 | 10.0 | 10 | 45 | 26.0 | 71 | B |
| 29 | 022-14-110217 | BS(CS) | Muhammad Shahroz Khurshid | 17 | 7.0 | 1.0 | 7 | 6.0 | NaN | 6 | 30 | 9.0 | 39 | F |
| 31 | 022-14-110231 | BS(CS) | Muhammad Taha Hasnain | 27 | 7.0 | 5.0 | 7 | 10.0 | NaN | 10 | 44 | 29.0 | 73 | B |
| 33 | 022-14-19919 | BS(CS) | Muhammad Younus Baig | 20 | 4.0 | 2.0 | 4 | 8.0 | 10.0 | 10 | 34 | 17.0 | 51 | F |
| 35 | 022-14-19923 | BS(CS) | Rabi Ahmed | 20 | 8.0 | 5.0 | 8 | NaN | 10.0 | 10 | 38 | 28.0 | 66 | C |
| 37 | 022-14-110230 | BS(CS) | Sadaf Nosheen | 27 | 9.0 | 5.0 | 9 | 9.0 | NaN | 9 | 45 | 30.0 | 75 | B |
| 39 | 022-14-110107 | BS(CS) | Sania Iqbal | 28 | 4.0 | 4.0 | 4 | 9.0 | 11.0 | 11 | 43 | 25.0 | 68 | C |
| 41 | 022-14-110225 | BS(CS) | Shariqa Ahmad | 20 | 7.0 | 6.0 | 7 | 8.0 | 10.0 | 10 | 37 | 20.0 | 57 | F |
| 43 | 022-14-110451 | BS(CS) | Syed Faizan Uddin | 28 | 9.0 | 4.0 | 9 | 6.0 | 8.0 | 8 | 45 | 34.0 | 79 | B |
| 45 | 022-14-110400 | BS(CS) | Syeda Sabika Raza | 27 | 9.0 | 6.0 | 9 | 9.0 | NaN | 9 | 45 | 35.0 | 80 | A |
| 47 | 022-14-110219 | BS(CS) | Waqar Ahmed | 11 | 9.0 | 5.0 | 9 | 5.0 | 7.0 | 7 | 27 | 19.0 | 46 | F |

### 9. Display only those students who got failed in examination

9. Display only those students who got failed in examination[¶](http://localhost:8888/notebooks/pandas%20(1).ipynb#9.-Display-only-those-students-who-got-failed-in-examination)

fa=r[r.Grade == 'F']

print(fa)

×

Student Code Degree Student Name Mid Quiz 1 Quiz 2 \

1 022-14-110233 BS(CS) Adeel Ahmed 17 NaN 5.0

3 022-14-19718 BS(CS) Ahmed Ali Raza 14 7.0 2.0

15 022-14-19916 BS(CS) Haseeb Sajid 18 NaN NaN

22 022-14-110398 BS(CS) Muhammad Abdullah 20 7.0 6.0

28 022-14-110387 BS(CS) Muhammad Saqib Intizar 22 6.0 6.0

29 022-14-110217 BS(CS) Muhammad Shahroz Khurshid 17 7.0 1.0

32 022-14-110035 BS(CS) Muhammad Wajahat Khan 9 NaN 2.0

33 022-14-19919 BS(CS) Muhammad Younus Baig 20 4.0 2.0

38 022-14-110229 BS(CS) Sadaquat Rafique 9 6.0 6.0

41 022-14-110225 BS(CS) Shariqa Ahmad 20 7.0 6.0

47 022-14-110219 BS(CS) Waqar Ahmed 11 9.0 5.0

Best of Quizzes Assignment 1 Assignment 2 Best of Assignments \

1 5 8.0 10.0 10

3 7 NaN 2.0 2

15 0 NaN NaN 0

22 7 7.0 NaN 7

28 6 9.0 11.0 11

29 7 6.0 NaN 6

32 2 8.0 10.0 10

33 4 8.0 10.0 10

38 6 5.0 7.0 7

41 7 8.0 10.0 10

47 9 5.0 7.0 7

Total Sessional (50) Final (50) Total (100) Grade

1 32 18.0 50 F

3 23 23.0 46 F

15 18 NaN 18 F

22 34 23.0 57 F

28 39 18.0 57 F

29 30 9.0 39 F

32 21 8.0 29 F

33 34 17.0 51 F

38 22 17.0 39 F

41 37 20.0 57 F

47 27 19.0 46 F

### 10. Find out the basic statistical info about data

10. Find out the basic statistical info about data[¶](http://localhost:8888/notebooks/pandas%20(1).ipynb#10.-Find-out-the-basic-statistical-info-about-data)

r.describe()

×

C:\Users\7500.GC\AppData\Local\Continuum\Anaconda3\lib\site-packages\numpy\lib\function\_base.py:3834: RuntimeWarning: Invalid value encountered in percentile

RuntimeWarning)

Out[52]:

|  | Mid | Quiz 1 | Quiz 2 | Best of Quizzes | Assignment 1 | Assignment 2 | Best of Assignments | Total Sessional (50) | Final (50) | Total (100) |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| count | 48.000000 | 45.000000 | 45.000000 | 48.000000 | 45.000000 | 36.000000 | 48.000000 | 48.000000 | 47.000000 | 48.000000 |
| mean | 22.937500 | 7.288889 | 4.866667 | 7.020833 | 7.755556 | 9.444444 | 8.937500 | 38.895833 | 27.851064 | 66.166667 |
| std | 5.236558 | 1.561209 | 1.455397 | 1.973113 | 1.170772 | 1.629100 | 2.127892 | 7.179283 | 8.431253 | 15.227821 |
| min | 9.000000 | 4.000000 | 1.000000 | 0.000000 | 5.000000 | 2.000000 | 0.000000 | 18.000000 | 8.000000 | 18.000000 |
| 25% | 20.000000 | NaN | NaN | 6.000000 | NaN | NaN | 8.750000 | 36.750000 | NaN | 60.000000 |
| 50% | 25.000000 | NaN | NaN | 7.000000 | NaN | NaN | 10.000000 | 41.000000 | NaN | 69.000000 |
| 75% | 27.000000 | NaN | NaN | 9.000000 | NaN | NaN | 10.000000 | 44.000000 | NaN | 75.250000 |
| max | 30.000000 | 9.000000 | 7.000000 | 9.000000 | 10.000000 | 11.000000 | 11.000000 | 46.000000 | 45.000000 | 91.000000 |

### 11. How many students got A, B, C, F?

11. How many students got A, B, C, F?[¶](http://localhost:8888/notebooks/pandas%20(1).ipynb#11.-How-many-students-got-A,-B,-C,-F?)

r.groupby("Grade").count()

×

Out[56]:

|  | Student Code | Degree | Student Name | Mid | Quiz 1 | Quiz 2 | Best of Quizzes | Assignment 1 | Assignment 2 | Best of Assignments | Total Sessional (50) | Final (50) | Total (100) |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Grade |  |  |  |  |  |  |  |  |  |  |  |  |  |
| A | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 5 | 8 | 8 | 8 | 8 |
| B | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 16 | 11 | 16 | 16 | 16 | 16 |
| C | 13 | 13 | 13 | 13 | 13 | 11 | 13 | 12 | 12 | 13 | 13 | 13 | 13 |
| F | 11 | 11 | 11 | 11 | 8 | 10 | 11 | 9 | 8 | 11 | 11 | 10 | 11 |

### 12. What are the mean scores for students who got A, B, C, F?

12. What are the mean scores for students who got A, B, C, F?[¶](http://localhost:8888/notebooks/pandas%20(1).ipynb#12.-What-are-the-mean-scores-for-students-who-got-A,-B,-C,-F?)

r.groupby("Grade").mean()

×

Out[58]:

|  | Mid | Quiz 1 | Quiz 2 | Best of Quizzes | Assignment 1 | Assignment 2 | Best of Assignments | Total Sessional (50) | Final (50) | Total (100) |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Grade |  |  |  |  |  |  |  |  |  |  |
| A | 27.000000 | 8.750000 | 6.125000 | 8.750000 | 8.125000 | 10.000000 | 9.375000 | 45.125000 | 40.375000 | 85.500000 |
| B | 25.687500 | 7.437500 | 5.125000 | 7.500000 | 7.875000 | 9.636364 | 9.250000 | 42.437500 | 30.750000 | 73.187500 |
| C | 22.846154 | 6.615385 | 4.272727 | 6.692308 | 7.833333 | 9.750000 | 9.692308 | 39.230769 | 24.769231 | 64.000000 |
| F | 16.090909 | 6.625000 | 4.100000 | 5.454545 | 7.111111 | 8.375000 | 7.272727 | 28.818182 | 17.200000 | 44.454545 |

. . .

**Lab Task:**

**Lab 6: To study and implement Artificial Neural Network using Keras**

Keras is a powerful easy-to-use Python library for developing and evaluating deep learning models. It wraps the efficient numerical computation libraries Theano and TensorFlow and allows you to define and train neural network models in a few short lines of code. Install Keras by using the following command:

> pip install keras

**Lab Tasks:**

1. Initialize the random number generator

from keras.models import Sequential

from keras.layers import Dense

import numpy

# fix random seed for reproducibility

numpy.random.seed(7)

1. Load the data

# load pima indians dataset

dataset = numpy.loadtxt("pima-indians-diabetes.csv", delimiter=",")

# split into input (X) and output (Y) variables

X = dataset[:,0:8]

Y = dataset[:,8]

Now create a model:

# create model

model = Sequential()

model.add(Dense(12, input\_dim=8, activation='relu'))

model.add(Dense(8, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

1. Compile the model

model.compile(loss='binary\_crossentropy', optimizer='adam', metrics=['accuracy'])

1. Fit the model

model.fit(X, Y, epochs=150, batch\_size=10)

1. Evaluate the model

scores = model.evaluate(X, Y)

print("\n%s: %.2f%%" % (model.metrics\_names[1], scores[1]\*100))

1. Perform Predictions

predictions = model.predict(X)

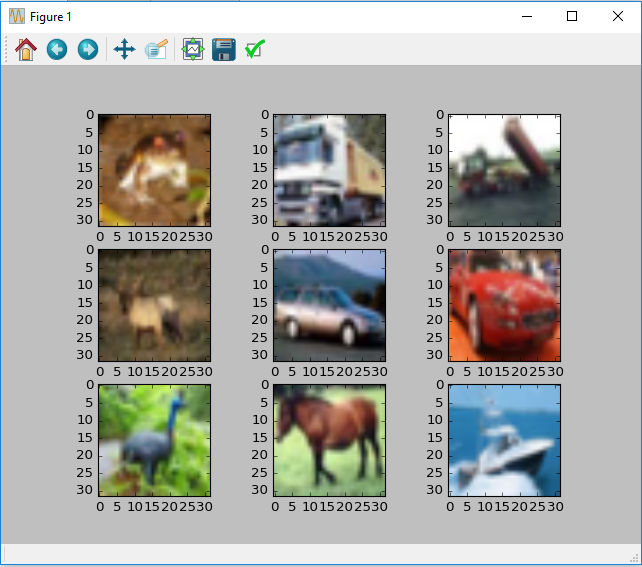
# round predictions

rounded = [round(x[0]) for x in predictions]

print(rounded)

**Lab7: To Study and implement Cnn on Keras**

**from** keras.datasets **import** cifar10  
**from** matplotlib **import** pyplot  
**from** scipy.misc **import** toimage  
(X\_train, y\_train), (X\_test, y\_test) = cifar10.load\_data()  
**for** i **in** range(0, 9):  
 pyplot.subplot(330 + 1 + i)  
 pyplot.imshow(toimage(X\_train[i]))  
pyplot.show()

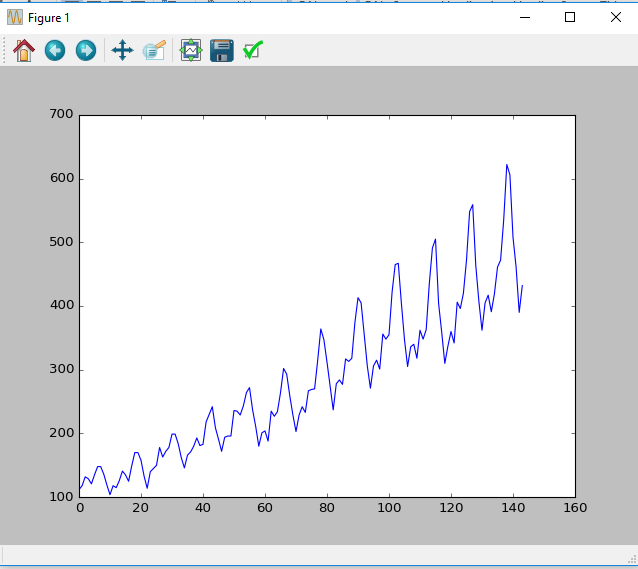
****

**import** numpy  
**from** keras.datasets **import** cifar10  
**from** keras.models **import** Sequential  
**from** keras.layers **import** Dense  
**from** keras.layers **import** Dropout  
**from** keras.layers **import** Flatten  
**from** keras.constraints **import** maxnorm  
**from** keras.optimizers **import** SGD  
**from** keras.layers.convolutional **import** Conv2D  
**from** keras.layers.convolutional **import** MaxPooling2D  
**from** keras.utils **import** np\_utils  
**from** keras **import** backend **as** K  
K.set\_image\_dim\_ordering(**'th'**)  
  
seed = 7  
numpy.random.seed(seed)  
seed = 7  
numpy.random.seed(seed)  
(X\_train, y\_train), (X\_test, y\_test) = cifar10.load\_data()  
X\_train = X\_train.astype(**'float32'**)  
X\_test = X\_test.astype(**'float32'**)  
X\_train = X\_train / 255.0  
X\_test = X\_test / 255.0  
y\_train = np\_utils.to\_categorical(y\_train)  
y\_test = np\_utils.to\_categorical(y\_test)  
num\_classes = y\_test.shape[1]  
model = Sequential()  
model.add(Conv2D(32, (3, 3), input\_shape=(3, 32, 32), padding=**'same'**, activation=**'relu'**, kernel\_constraint=maxnorm(3)))  
model.add(Dropout(0.2))  
model.add(Conv2D(32, (3, 3), activation=**'relu'**, padding=**'same'**, kernel\_constraint=maxnorm(3)))  
model.add(MaxPooling2D(pool\_size=(2, 2)))  
model.add(Flatten())  
model.add(Dense(512, activation=**'relu'**, kernel\_constraint=maxnorm(3)))  
model.add(Dropout(0.5))  
model.add(Dense(num\_classes, activation=**'softmax'**))  
epochs = 25  
lrate = 0.01  
decay = lrate/epochs  
sgd = SGD(lr=lrate, momentum=0.9, decay=decay, nesterov=**False**)  
model.compile(loss=**'categorical\_crossentropy'**, optimizer=sgd, metrics=[**'accuracy'**])  
print(model.summary())  
model.fit(X\_train, y\_train, validation\_data=(X\_test, y\_test), epochs=epochs, batch\_size=500)  
scores = model.evaluate(X\_test, y\_test, verbose=0)  
print(**"Accuracy: %.2f%%"** % (scores[1]\*100))

**Lab 8: To study and implement Rnn on keras**

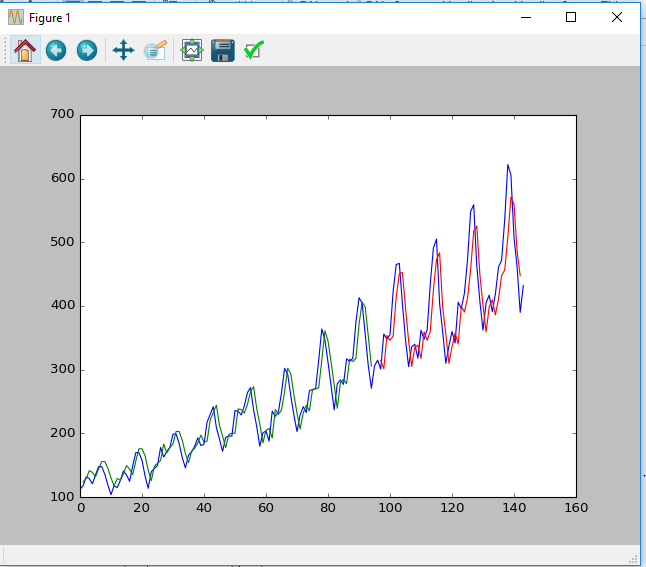
**Task1:**

**import** pandas  
**import** matplotlib.pyplot **as** plt  
dataset = pandas.read\_csv(**'international-airline-passengers.csv'**, usecols=[1], engine=**'python'**, skipfooter=3)  
plt.plot(dataset)  
plt.show()

****

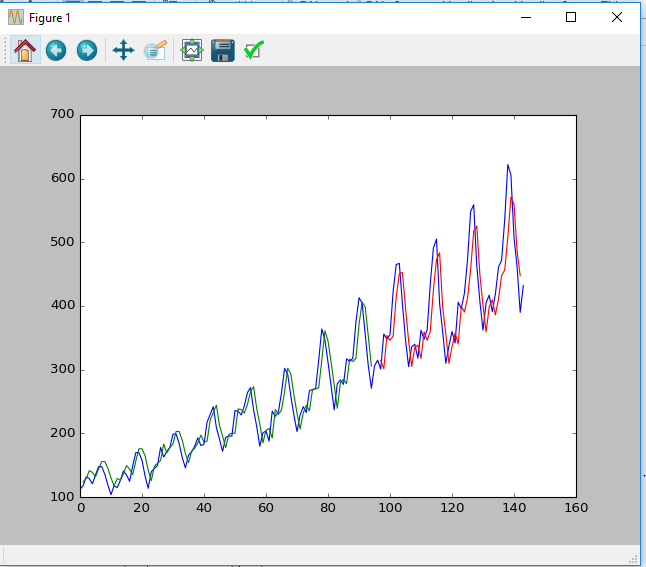
**Task2:**

**import** numpy  
**import** matplotlib.pyplot **as** plt  
**import** pandas  
**import** math  
**from** keras.models **import** Sequential  
**from** keras.layers **import** Dense  
**from** keras.layers **import** LSTM  
**from** sklearn.preprocessing **import** MinMaxScaler  
**from** sklearn.metrics **import** mean\_squared\_error  
  
*# fix random seed for reproducibility*numpy.random.seed(7)  
  
*# load the dataset*dataframe = pandas.read\_csv(**'international-airline-passengers.csv'**, usecols=[1], engine=**'python'**, skipfooter=3)  
dataset = dataframe.values  
dataset = dataset.astype(**'float32'**)  
  
*# normalize the dataset*scaler = MinMaxScaler(feature\_range=(0, 1))  
dataset = scaler.fit\_transform(dataset)  
  
*# split into train and test sets*train\_size = int(len(dataset) \* 0.67)  
test\_size = len(dataset) - train\_size  
train, test = dataset[0:train\_size,:], dataset[train\_size:len(dataset),:]  
print(len(train), len(test))  
  
*# convert an array of values into a dataset matrix***def** create\_dataset(dataset, look\_back=1):  
 dataX, dataY = [], []  
 **for** i **in** range(len(dataset)-look\_back-1):  
 a = dataset[i:(i+look\_back), 0]  
 dataX.append(a)  
 dataY.append(dataset[i + look\_back, 0])  
 **return** numpy.array(dataX), numpy.array(dataY)  
  
*# reshape into X=t and Y=t+1*look\_back = 1  
trainX, trainY = create\_dataset(train, look\_back)  
testX, testY = create\_dataset(test, look\_back)  
  
*# reshape input to be [samples, time steps, features]*trainX = numpy.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))  
testX = numpy.reshape(testX, (testX.shape[0], 1, testX.shape[1]))  
  
*# create and fit the LSTM network*model = Sequential()  
model.add(LSTM(4, input\_shape=(1, look\_back)))  
model.add(Dense(1))  
model.compile(loss=**'mean\_squared\_error'**, optimizer=**'adam'**)  
model.fit(trainX, trainY, epochs=100, batch\_size=1, verbose=2)  
  
  
*# make predictions*trainPredict = model.predict(trainX)  
testPredict = model.predict(testX)  
*# invert predictions*trainPredict = scaler.inverse\_transform(trainPredict)  
trainY = scaler.inverse\_transform([trainY])  
testPredict = scaler.inverse\_transform(testPredict)  
testY = scaler.inverse\_transform([testY])  
*# calculate root mean squared error*trainScore = math.sqrt(mean\_squared\_error(trainY[0], trainPredict[:,0]))  
print(**'Train Score: %.2f RMSE'** % (trainScore))  
testScore = math.sqrt(mean\_squared\_error(testY[0], testPredict[:,0]))  
print(**'Test Score: %.2f RMSE'** % (testScore))  
  
  
*# shift train predictions for plotting*trainPredictPlot = numpy.empty\_like(dataset)  
trainPredictPlot[:, :] = numpy.nan  
trainPredictPlot[look\_back:len(trainPredict)+look\_back, :] = trainPredict  
*# shift test predictions for plotting*testPredictPlot = numpy.empty\_like(dataset)  
testPredictPlot[:, :] = numpy.nan  
testPredictPlot[len(trainPredict)+(look\_back\*2)+1:len(dataset)-1, :] = testPredict  
*# plot baseline and predictions*plt.plot(scaler.inverse\_transform(dataset))  
plt.plot(trainPredictPlot)  
plt.plot(testPredictPlot)  
plt.show()

****

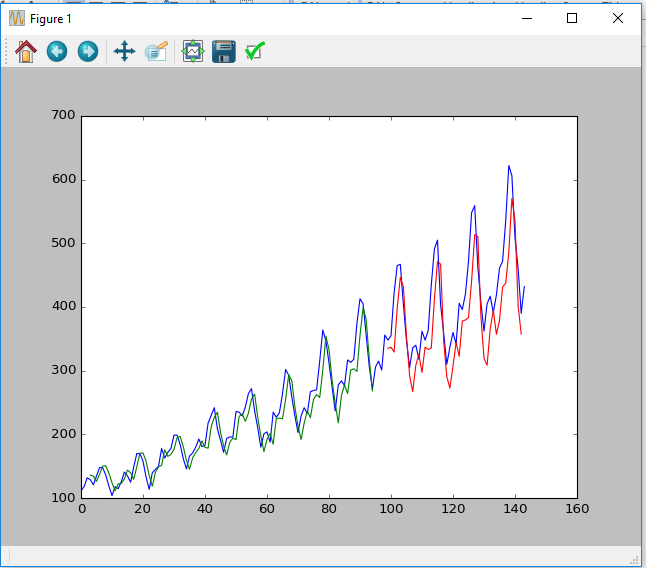
**Task 3:**

*# LSTM for international airline passengers problem with regression framing***import** numpy  
**import** matplotlib.pyplot **as** plt  
**from** pandas **import** read\_csv  
**import** math  
**from** keras.models **import** Sequential  
**from** keras.layers **import** Dense  
**from** keras.layers **import** LSTM  
**from** sklearn.preprocessing **import** MinMaxScaler  
**from** sklearn.metrics **import** mean\_squared\_error  
*# convert an array of values into a dataset matrix***def** create\_dataset(dataset, look\_back=1):  
 dataX, dataY = [], []  
 **for** i **in** range(len(dataset)-look\_back-1):  
 a = dataset[i:(i+look\_back), 0]  
 dataX.append(a)  
 dataY.append(dataset[i + look\_back, 0])  
 **return** numpy.array(dataX), numpy.array(dataY)  
*# fix random seed for reproducibility*numpy.random.seed(7)  
*# load the dataset*dataframe = read\_csv(**'international-airline-passengers.csv'**, usecols=[1], engine=**'python'**, skipfooter=3)  
dataset = dataframe.values  
dataset = dataset.astype(**'float32'**)  
*# normalize the dataset*scaler = MinMaxScaler(feature\_range=(0, 1))  
dataset = scaler.fit\_transform(dataset)  
*# split into train and test sets*train\_size = int(len(dataset) \* 0.67)  
test\_size = len(dataset) - train\_size  
train, test = dataset[0:train\_size,:], dataset[train\_size:len(dataset),:]  
*# reshape into X=t and Y=t+1*look\_back = 1  
trainX, trainY = create\_dataset(train, look\_back)  
testX, testY = create\_dataset(test, look\_back)  
*# reshape input to be [samples, time steps, features]*trainX = numpy.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))  
testX = numpy.reshape(testX, (testX.shape[0], 1, testX.shape[1]))  
*# create and fit the LSTM network*model = Sequential()  
model.add(LSTM(4, input\_shape=(1, look\_back)))  
model.add(Dense(1))  
model.compile(loss=**'mean\_squared\_error'**, optimizer=**'adam'**)  
model.fit(trainX, trainY, epochs=100, batch\_size=1, verbose=2)  
*# make predictions*trainPredict = model.predict(trainX)  
testPredict = model.predict(testX)  
*# invert predictions*trainPredict = scaler.inverse\_transform(trainPredict)  
trainY = scaler.inverse\_transform([trainY])  
testPredict = scaler.inverse\_transform(testPredict)  
testY = scaler.inverse\_transform([testY])  
*# calculate root mean squared error*trainScore = math.sqrt(mean\_squared\_error(trainY[0], trainPredict[:,0]))  
print(**'Train Score: %.2f RMSE'** % (trainScore))  
testScore = math.sqrt(mean\_squared\_error(testY[0], testPredict[:,0]))  
print(**'Test Score: %.2f RMSE'** % (testScore))  
*# shift train predictions for plotting*trainPredictPlot = numpy.empty\_like(dataset)  
trainPredictPlot[:, :] = numpy.nan  
trainPredictPlot[look\_back:len(trainPredict)+look\_back, :] = trainPredict  
*# shift test predictions for plotting*testPredictPlot = numpy.empty\_like(dataset)  
testPredictPlot[:, :] = numpy.nan  
testPredictPlot[len(trainPredict)+(look\_back\*2)+1:len(dataset)-1, :] = testPredict  
*# plot baseline and predictions*plt.plot(scaler.inverse\_transform(dataset))  
plt.plot(trainPredictPlot)  
plt.plot(testPredictPlot)  
plt.show()

****

**Task4:**

*# LSTM for international airline passengers problem with regression framing***import** numpy  
**import** matplotlib.pyplot **as** plt  
**from** pandas **import** read\_csv  
**import** math  
**from** keras.models **import** Sequential  
**from** keras.layers **import** Dense  
**from** keras.layers **import** LSTM  
**from** sklearn.preprocessing **import** MinMaxScaler  
**from** sklearn.metrics **import** mean\_squared\_error  
*# convert an array of values into a dataset matrix***def** create\_dataset(dataset, look\_back=1):  
 dataX, dataY = [], []  
 **for** i **in** range(len(dataset)-look\_back-1):  
 a = dataset[i:(i+look\_back), 0]  
 dataX.append(a)  
 dataY.append(dataset[i + look\_back, 0])  
 **return** numpy.array(dataX), numpy.array(dataY)  
*# fix random seed for reproducibility*numpy.random.seed(7)  
*# load the dataset*dataframe = read\_csv(**'international-airline-passengers.csv'**, usecols=[1], engine=**'python'**, skipfooter=3)  
dataset = dataframe.values  
dataset = dataset.astype(**'float32'**)  
*# normalize the dataset*scaler = MinMaxScaler(feature\_range=(0, 1))  
dataset = scaler.fit\_transform(dataset)  
*# split into train and test sets*train\_size = int(len(dataset) \* 0.67)  
test\_size = len(dataset) - train\_size  
train, test = dataset[0:train\_size,:], dataset[train\_size:len(dataset),:]  
*# reshape into X=t and Y=t+1*look\_back = 1  
trainX, trainY = create\_dataset(train, look\_back)  
testX, testY = create\_dataset(test, look\_back)  
*# reshape input to be [samples, time steps, features]*trainX = numpy.reshape(trainX, (trainX.shape[0], 1, trainX.shape[1]))  
testX = numpy.reshape(testX, (testX.shape[0], 1, testX.shape[1]))  
*# create and fit the LSTM network*model = Sequential()  
model.add(LSTM(4, input\_shape=(1, look\_back)))  
model.add(Dense(1))  
model.compile(loss=**'mean\_squared\_error'**, optimizer=**'adam'**)  
model.fit(trainX, trainY, epochs=100, batch\_size=1, verbose=2)  
*# make predictions*trainPredict = model.predict(trainX)  
testPredict = model.predict(testX)  
*# invert predictions*trainPredict = scaler.inverse\_transform(trainPredict)  
trainY = scaler.inverse\_transform([trainY])  
testPredict = scaler.inverse\_transform(testPredict)  
testY = scaler.inverse\_transform([testY])  
*# calculate root mean squared error*trainScore = math.sqrt(mean\_squared\_error(trainY[0], trainPredict[:,0]))  
print(**'Train Score: %.2f RMSE'** % (trainScore))  
testScore = math.sqrt(mean\_squared\_error(testY[0], testPredict[:,0]))  
print(**'Test Score: %.2f RMSE'** % (testScore))  
*# shift train predictions for plotting*trainPredictPlot = numpy.empty\_like(dataset)  
trainPredictPlot[:, :] = numpy.nan  
trainPredictPlot[look\_back:len(trainPredict)+look\_back, :] = trainPredict  
*# shift test predictions for plotting*testPredictPlot = numpy.empty\_like(dataset)  
testPredictPlot[:, :] = numpy.nan  
testPredictPlot[len(trainPredict)+(look\_back\*2)+1:len(dataset)-1, :] = testPredict  
*# plot baseline and predictions*plt.plot(scaler.inverse\_transform(dataset))  
plt.plot(trainPredictPlot)  
plt.plot(testPredictPlot)  
plt.show()

****