# Numpy

Numpy - Bu kitabxana çoxölçülü massiv və matrislərdə işləməyimizdə bizə kömək edir. Daha az kodla daha effektiv əməliyyatlara imkan verir.
Numpy -ın obyekti (işçi elementi) ndarray-dır.
ndarray dedikdə burada burada n = 1, n = 2, n = 3 , n = 4 və s. ola bilər. Numpy - də olan massivlər əslində hər biri listdir. Standart pythonda massivlər dinamik olaraq böyüyə bilir.
Numpy modulu daxilində isə konkret ölçü verilməlidir. Əgər Numpyda biz ölçünü dəyişsək yeni bir massiv yaranar, köhnəsi isə silinər. Numpy əməliyyatlarının sürətli olması onun C və C++ ilə yazılması və dataların eyni RAM hücrə blok seximində saxlanmasıdır. Standart Pythonda isə massiv (matris) dataları fərqli yerlərdə saxlana bilir. Bu fərqi böyük əməllərdə daha yaxşı hiss etmək olur. Numpyda massivlərin sürəti Standart Pythonda olan massivlərin sürətindən **50 dəfə** çoxdur. Sadə nümunələrə baxaq:

```
import numpy as np
array_1 = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9])
print(f"array_1 = {array_1}")
```

```
array_1 = [1 2 3 4 5 6 7 8 9]
```

```
#version control
print(np.__version__)
```

```
1.21.5
```

Massivlər haqqında belə bir fikri demək olar ki, əlimizdə N - ölçülü matris (və ya massiv) varsa, bizim həmin N - ölçülü matris içərisində yerləşəcək matrislər (və ya massivlər) in ölçüsü (N-1) - ə bərabər və ya ondan kiçik olmalıdır. Misal üçün iki, üç, və ya dörd ölçülü matrisləri götürək .

İçərisində müxtəif sayda birölçülü (1D) matris saxlayan matrislərə ikiölçülü (2D) matrislər deyilir.
İçərisində müxtəif sayda ikiölçülü 2D matris saxlayan matrislərə üçölçülü (3D) matrislər deyilir.
İçərisində müxtəif sayda üçölçülü (3D) matris saxlayan matrislərə dördölçülü (4D) matrislər deyilir.

Arrayın yəni matrisin Dimensional ölçüsünün tapılması ***np.dim*** ilə həyata keçirilir.

Sıfır ölçülü (0D - zero Dimension) matris:

```
import numpy as np
array_2 = np.array(62)
print(f"Matrisin Dimensional ölçüsü: {array_2.ndim}")
```

```
Matrisin Dimensional ölçüsü: 0
```

Bir ölçülü ( 1D - one Dimension) matris:

```
import numpy as np
array_3 = np.array([5, 10, 35, 20, 19])
print(f"Matrisin Dimensional ölçüsü: {array_3.ndim}")
```

```
Matrisin Dimensional ölçüsü: 1
```

İki ölçülü ( 2D - two Dimension) matris:

```
import numpy as np
array_4 = np.array([[47, 32, 28, 10, 26], [44, 30, 38, 46, 17], [35, 37, 4, 41, 16]])
print(f"Matrisin Dimensional ölçüsü: {array_4.ndim}")
```

```
Matrisin Dimensional ölçüsü: 2
```

Üç ölçülü (3D - three Dimension ) matris :

```
import numpy as np
array_5 = np.array([[[22, 33, 20, 37, 47],
                     [35, 30, 26, 3, 21],
                     [16, 28, 34, 50, 39],
                     [46, 15, 18, 42, 12]],
                    [[15, 36, 9, 41, 44],
                     [9, 21, 45, 5, 17],
                     [46, 10, 45, 34, 39],
```

```
                    [12, 22, 47, 4, 24]]])
print(f"Matrisin Dimensional ölçüsü: {array_5.ndim}")
```

```
Matrisin Dimensional ölçüsü: 3
```

Dörd ölçülü (4D - four Dimension ) matris :

```
from numpy.core.fromnumeric import ndim
import numpy as np
array_6 = np.arange(1001,1521)
array_6 = array_6.reshape(2,2,10,13)
print(f"Matrisimiz:\n\n{array_6}")
#reshape sonra izah ediləcək
print(f"\n\nMatrisin Dimensional ölçüsü: {array_6.ndim}")
```

```
Matrisimiz:

[[[[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]]]]


Matrisin Dimensional ölçüsü: 4
```

Verilən ölçüdə massivin yaradılması **np.ndmin** ilə həyata keçirilir:

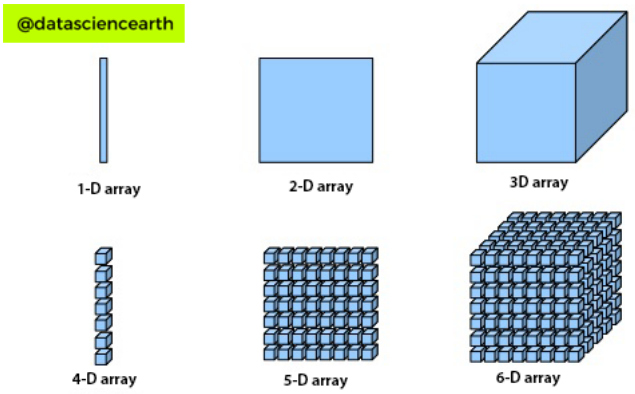
```
import numpy as np
array_7 = np.array([[43, 8, 27, 45, 26], [6, 23, 39, 41, 35], [32, 27, 2, 5, 36]],ndmin = 5)
print(f"\n\nMatrisin Dimensional ölçüsü: {array_7.ndim}\n")
print(array_7)
```

```
Matrisin Dimensional ölçüsü: 5

[[[[[43  8 27 45 26]
    [ 6 23 39 41 35]
    [32 27  2  5 36]]]]]
```

Müxtəlif ölçüdə olan matrislərin izahı:

Proqramlamada bir dəyişən sadəcə bir dəyər tuta bilər, əyər bu dəyişən matrisdirsə (array) bir və ya birdən çox dəyər ala bilmə xüsusiyyəti vardır.
Misal üçün
1-D matris bir vektor,
2-D matris bir düzbucaqlı
3-D matris bir kub kimidir.
4-D matrisi kubların vektoru olaraq ifadə edə bilərik. Eyni formada
5-D matrisi kubların düzbucaqlısı
6-D matrisi isə kubların kubu olaraq düşünə bilərik. Aşağıdakı şəkillərə baxaq:




Numpy matrisi elementlərinə onların indeksi vasitəsilə keçid edə bilirik;
Belə ki, aşağıdakı nümunədə **array_8** 2D - ölçülü matrisinin daxilindəki
2 ci sətir 4-cü sütun elementinə (1,3) indexi vasitəsi ilə keçid edə bilirik;
Nümunədə iki üsulla çağırma göstərilmişdir:

```
import numpy as np
array_8 = np.array([[22, 18, 7, 41, 4],
                    [23, 31, 24, 50, 47],
                    [20, 49, 28, 6, 26]])
print(f"Çağırma üsulu 1: {array_8[1,3]}\n")
print(f"Çağırma üsulu 2: {array_8[1][3]}")
```

```
Çağırma üsulu 1: 50

Çağırma üsulu 2: 50
```

Adi Toplama əməlinə baxaq:

```
print(array_8[1,3]+array_8[2,3])
```

```
56
```

Numpy matrislərində elementləri ifadə edərkən istifadə olunan indekslənmə metodunda son iki ədəd sətir və sütunu ifadə edər.
Misal olaraq,
**array_6** - nın quruluşu olan (2,2,10,13) - nı qarşıya qoyub nəzərdən keçirsək,
matrisin 4 ölçülü olduğunu ( **(2,2,10,13)** - mötərizəsi daxilindəki elementlərin sayı),
daxilində 2 ədəd üçölçülü matris saxladığını (mötərizə daxilindəki birinci ədəd)
ardınca həmin üçölçülü matrislərin də öz daxilində 2 ədəd (mötərizə daxilindəki ikinci ədəd) ikiölçülü matris saxladığını müşahidə edərik,
burada 10 rəqəmi sətirlərin sayını 13 isə sütunların sayını ifadə edər. Onu da xatırlatmaq lazımdır ki, matris indeksləri ilə iş zamanı həmişə axırıncı ədəd sütunların sayını , axırdan ikinci ədəd isə sətirlərin sayını ifadfə edər.

Standart python listlərində olduğu kimi neqativ yəni mənfi indekslərdən istifadə edilməsi burada da mümkündür.

```
import numpy as np
array_9 = np.array([
                    [[22, 18, 7, 41, 4],
                     [23, 31, 24, 50, 47],
                     [20, 49, 28, 6, 26]],
                    [[22, 18, 7, 41, 4],
                     [23, 31, 24, 50, 47],
                     [20, 49, 28, 6, 26]]
                    ])
array_9[1, 2, -2]
```

```
6
```

### Slicing

Listlərdə olduğu kimi burada da slicing yəni massivi parçalama və ya bölmə əməli həyata keçirmək mümkündür.

Birölçülü matrsilərdə baxaq:

```
import numpy as np
array_1D_slicing = np.array([5, 15, 40, 49, 22, 26, 14, 31, 48, 36])
print(f"1D array: {array_1D_slicing}")
```

```
1D array: [ 5 15 40 49 22 26 14 31 48 36]
```

```
print(f"array_1D_slicing[2:7] = {array_1D_slicing[2:7]}")
```

```
array_1D_slicing[2:7] = [40 49 22 26 14]
```

```
print(f"array_1D_slicing[3:] = {array_1D_slicing[3:]}")
```

```
array_1D_slicing[3:] = [49 22 26 14 31 48 36]
```

```
print(f"array_1D_slicing[:8] = {array_1D_slicing[:8]}")
```

```
array_1D_slicing[:8] = [ 5 15 40 49 22 26 14 31]
```

```
print(f"array_1D_slicing[-6:-2] = {array_1D_slicing[-6:-2]}")
```

```
array_1D_slicing[-6:-2] = [22 26 14 31]
```

```
print(f"array_1D_slicing[2:9:3] = {array_1D_slicing[2:9:3]}")
```

```
array_1D_slicing[2:9:3] = [40 26 48]
```

```
print(f"array_1D_slicing[::4] = {array_1D_slicing[::4]}")
```

```
array_1D_slicing[::4] = [ 5 22 48]
```

**İkiölçülü matrislərdə baxaq:**

```
import numpy as np
array_2D_slicing = np.array([[12, 7, 20, 45, 45, 5, 15, 41, 42, 14],
                             [41, 45, 4, 2, 43, 21, 47, 50, 44, 37],
                             [26, 45, 47, 13, 46, 47, 32, 30, 21, 27]])
print(f"2D array: \n\n{array_2D_slicing}")
```

```
2D array:

[[12  7 20 45 45  5 15 41 42 14]
 [41 45  4  2 43 21 47 50 44 37]
 [26 45 47 13 46 47 32 30 21 27]]
```

```
print(f"array_2D_slicing[1, 2:7] = {array_2D_slicing[1, 2:7]}")
```

```
array_2D_slicing[1, 2:7] = [ 4  2 43 21 47]
```

```
print(f"array_2D_slicing[2, 3:] = {array_2D_slicing[2, 3:]}")
```

```
array_2D_slicing[2, 3:] = [13 46 47 32 30 21 27]
```

```
print(f"array_2D_slicing[0:2, 2:7] = \n\n{array_2D_slicing[0:2, 2:7]}")
```

```
array_2D_slicing[0:2, 2:7] =

[[20 45 45  5 15]
 [ 4  2 43 21 47]]
```

NumPy Tensors, Slicing, and Images github mənbəsində, image -ların NumPy tensorları olaraq şərhi ve slicing əməliyyatı ilə əlaqədar detallı bir nümunə var. Bu nümunə NumPy, çoxölçülü massivlər və slicing kimi terminlərin zehninizdə konkretləşəcəyini düşünürəm.