```
{
 "cells": [
  {
   "cell_type": "markdown",
   "id": "d2847674-40a9-443a-bf4f-99574a4ebdee",
   "metadata": {},
   "source": [
    "# "
   ]
  },
  {
   "cell_type": "markdown",
   "id": "315b04d0-568a-40e4-8e7d-0574c62ae8c1",
   "metadata": {},
   "source": [
    "Lec 1 of this course. \n",
    "In this file, we will study about the different functions in numpy"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 1,
   "id": "f60bf848-310a-44ae-a919-fe6afcf83ee7",
   "metadata": {},
   "outputs": [],
   "source": [
    "import numpy as np"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "d780d6d5-0245-466e-8a8b-a07571e241cb",
   "metadata": {},
   "source": [
    "Below, we will declare an array and a matrix. \n",
    "Let's see if math works here: $x = 0$"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 7,
   "id": "2b70be70-6158-43e5-8006-1ce09ce809f5",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "[1 2 3]\n"
     ]
    }
   ],
   "source": [
    "data1 = np.array([1, 2, 3])\n",
```

```
  "print(data1)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 19,
 "id": "2911340b-aaec-43af-aa95-e6e0f84558e5",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "[[1. 2. 3.]\n",
    " [4. 5. 7.]]\n"
   ]
  }
 ],
 "source": [
  "data2 = np.array([[1, 2, 3], [4, 5, 7.0]])\n",
  "print(data2)"
 ]
},
{
 "cell_type": "markdown",
 "id": "7dc35a30-7480-407f-8716-6f5cb1c3da3e",
 "metadata": {},
 "source": [
  "1. Queries about the array:\n",
  "   shape\n",
  "   size\n",
  "   ndim\n",
  "   nbytes\n",
  "   dtype"
 ]
},
{
 "cell_type": "code",
 "execution_count": 22,
 "id": "505f791e-b7a1-483a-a820-d7385942a678",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "2\n",
    "3\n"
   ]
  },
  {
   "data": {
    "text/plain": [
     "dtype('float64')"
```

```
    ]
   },
   "execution_count": 22,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "data2.shape\n",
  "# We can extract only the rows and columns as well\n",
  "n_r = data2.shape[0]\n",
  "n_c = data2.shape[1]\n",
  "print(n_r); print(n_c)\n",
  "data2.dtype"
 ]
},
{
 "cell_type": "code",
 "execution_count": 29,
 "id": "ab9c5685-0db1-460f-b817-37761629e23b",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "complex64\n",
    "[1.+0.j 2.+0.j 3.+0.j]\n"
   ]
  }
 ],
 "source": [
  "data = np.array([1, 2, 3], dtype=np.complex64)\n",
  "print(data.dtype)\n",
  "print(data)"
 ]
},
{
 "cell_type": "markdown",
 "id": "e665deb7-846a-43c9-a91f-28fb271e3392",
 "metadata": {},
 "source": [
  "Another way of typecating data is using: astype"
 ]
},
{
 "cell_type": "code",
 "execution_count": 31,
 "id": "28d71020-1c7a-4c2e-8f01-33f58dd64509",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
```

```
   "text": [
    "[1.+0.j 2.+0.j 3.+0.j]\n"
   ]
  }
 ],
 "source": [
  "data = np.array([1, 2, 3])\n",
  "data = data.astype(np.complex64)\n",
  "print(data)"
 ]
},
{
 "cell_type": "markdown",
 "id": "f4f319d9-c548-446a-8931-af373ca4ab47",
 "metadata": {},
 "source": [
  "2. Array initialziation methods\n",
  "   One should avoid dynamic arrays\n",
  "   C -> malloc, "
 ]
},
{
 "cell_type": "code",
 "execution_count": 48,
 "id": "cd6cb976-b2db-48fc-a7da-125930c6c0e2",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "[[0. 0. 0. 0.]\n",
    " [0. 0. 0. 0.]\n",
    " [0. 0. 0. 0.]\n",
    " [0. 0. 0. 0.]]\n",
    "[[1. 0. 0. 0. 0.]\n",
    " [0. 1. 0. 0. 0.]\n",
    " [0. 0. 1. 0. 0.]\n",
    " [0. 0. 0. 1. 0.]\n",
    " [0. 0. 0. 0. 1.]]\n",
    "[1 2 3]\n",
    "[[1 0 0]\n",
    " [0 2 0]\n",
    " [0 0 3]]\n"
   ]
  }
 ],
 "source": [
  "a = np.zeros([4,4])\n",
  "print(a)\n",
  "b = np.ones([5,5,5])\n",
  "#print(b)\n",
  "c = np.eye(5)\n",
  "print(c)\n",
```

```
  "print(data1)\n",
  "d = np.diag(data1)\n",
  "print(d)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 52,
 "id": "768e8783-513d-4fbd-8a8b-ba85b252219d",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "[1.  1.1 1.2 1.3 1.4 1.5 1.6 1.7 1.8 1.9 2. ]\n",
    "(11,)\n"
   ]
  }
 ],
 "source": [
  "e = np.linspace(1, 2, 11)\n",
  "print(e)\n",
  "f = np.logspace(2, 3, 11)\n",
  "print(f.shape)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 56,
 "id": "788e73d2-3b30-475e-b388-aafa6124df8d",
 "metadata": {},
 "outputs": [],
 "source": [
  "g = lambda i,j: i*j # Lambda functions"
 ]
},
{
 "cell_type": "code",
 "execution_count": 55,
 "id": "cca96216-094d-4774-a03f-6f1a97bd3a4b",
 "metadata": {},
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "10\n"
   ]
  }
 ],
 "source": [
  "print(g(5,2))"
 ]
```

```
  },
  {
   "cell_type": "code",
   "execution_count": 62,
   "id": "e5468309-a679-4756-a8a5-912cfd436538",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "[[ 0.          0.         -0.         -0.         -0.          0.
]\n",
      " [ 0.84147098  0.45464871 -0.35017549 -0.83304996 -0.55002214
0.2386935 ]\n",
      " [ 0.90929743  0.4912955  -0.37840125 -0.90019763 -0.59435646
0.2579333 ]\n",
      " [ 0.14112001  0.07624747 -0.05872664 -0.13970775 -0.09224219
0.04003041]\n",
      " [-0.7568025  -0.40890213  0.31494096  0.74922879  0.49467912 -
0.21467625]\n",
      " [-0.95892427 -0.518109    0.3990533   0.94932784  0.62679474 -
0.27201056]\n",
      " [-0.2794155  -0.15096884  0.11627788  0.27661925  0.18263816 -
0.07925961]\n",
      " [ 0.6569866   0.35497137 -0.27340289 -0.6504118  -0.4294351
0.18636225]\n",
      " [ 0.98935825  0.53455254 -0.4117183  -0.97945724 -0.64668771
0.28064352]\n",
      " [ 0.41211849  0.22266857 -0.1715018  -0.40799421 -0.26937862
0.11690243]\n",
      " [-0.54402111 -0.29393586  0.22639266  0.53857682  0.35559593 -
0.15431822]]\n"
     ]
    }
   ],
   "source": [
    "## x1 = np.linspace(0, 5, 6)\n",
    "y1 = np.linspace(0, 10, 11)\n",
    "[X, Y] = np.meshgrid(x1, y1)\n",
    "\n",
    "Z = np.cos(X)*np.sin(Y)\n",
    "print(Z)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 66,
   "id": "98a2b031-4435-4468-b65e-ff41967f7d0c",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
```

```
     "text": [
      "[0.         0.45454545 0.90909091 1.36363636 1.81818182
2.27272727\n",
      " 2.72727273 3.18181818 3.63636364 4.09090909 4.54545455 5.
]\n",
      "[0.         0.45454545 0.90909091 1.36363636 1.81818182
2.27272727\n",
      " 2.72727273 3.18181818 3.63636364 4.09090909 4.54545455 5.
]\n",
      "[-1.          0.45454545  0.90909091  1.36363636  1.81818182
2.27272727\n",
      "  2.72727273  3.18181818  3.63636364  4.09090909  4.54545455  5.
]\n",
      "[-1.          0.45454545  0.90909091  1.36363636  1.81818182
2.27272727\n",
      "  2.72727273  3.18181818  3.63636364  4.09090909  4.54545455  5.
]\n"
     ]
    }
   ],
   "source": [
    "x2 = np.linspace(0,5, 12)\n",
    "y2 = x2\n",
    "print(x2)\n",
    "print(y2)\n",
    "y2[0] = -1\n",
    "print(y2)\n",
    "print(x2)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 72,
   "id": "132d4193-5b4c-46ee-baed-025aa504a895",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "[0.         0.45454545 0.90909091 1.36363636 1.81818182
2.27272727\n",
      " 2.72727273 3.18181818 3.63636364 4.09090909 4.54545455 5.
]\n",
      "[-1.          0.45454545  0.90909091  1.36363636  1.81818182
2.27272727\n",
      "  2.72727273  3.18181818  3.63636364  4.09090909  4.54545455  5.
]\n",
      "[[1. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]\n",
      " [0. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]\n",
      " [0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]\n",
      " [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0.]\n",
      " [0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]\n",
      " [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]\n",
```

```
      " [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]\n",
      " [0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]\n",
      " [0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]\n",
      " [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]\n",
      " [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]\n",
      " [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]]\n"
     ]
    }
   ],
   "source": [
    "x2 = np.linspace(0,5, 12)\n",
    "y2 = np.copy(x2)\n",
    "y2[0] = -1\n",
    "print(x2)\n",
    "print(y2)\n",
    "y3 = np.diag(np.ones(np.shape(x2)))\n",
    "print(y3)"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "f652fa03-3536-4b9b-a572-2628c95ac6ee",
   "metadata": {},
   "source": [
    "Fancy indexing"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 81,
   "id": "9b03a239-124a-4e3a-82f1-bd92fd9f997a",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "[ 0.00000000e+00  3.24699469e-01  6.14212713e-01  8.37166478e-
01\n",
      "  9.69400266e-01  9.96584493e-01  9.15773327e-01  7.35723911e-
01\n",
      "  4.75947393e-01  1.64594590e-01 -1.64594590e-01 -4.75947393e-
01\n",
      " -7.35723911e-01 -9.15773327e-01 -9.96584493e-01 -9.69400266e-
01\n",
      " -8.37166478e-01 -6.14212713e-01 -3.24699469e-01 -2.44929360e-
16]\n",
      "[ True  True  True  True  True  True  True  True  True  True False
False\n",
      " False False False False False False False False]\n",
      "[0.32469947 0.61421271 0.83716648 0.96940027 0.99658449
0.91577333\n",
      " 0.73572391 0.47594739 0.16459459]\n",
```

```
      "[0.         0.05263158 0.10526316 0.15789474 0.21052632
0.26315789\n",
      " 0.31578947 0.36842105 0.42105263 0.47368421 0.52631579
0.57894737\n",
      " 0.63157895 0.68421053 0.73684211 0.78947368 0.84210526
0.89473684\n",
      " 0.94736842 1.        ]\n",
      "[0.05263158 0.10526316 0.15789474 0.21052632 0.26315789
0.31578947\n",
      " 0.36842105 0.42105263 0.47368421]\n"
     ]
    }
   ],
   "source": [
    "A = np.sin(np.linspace(0, 2*np.pi, 20))\n",
    "print(A)\n",
    "B = A[A>0]\n",
    "print(A>=0)\n",
    "print(B)\n",
    "C = np.linspace(0,1,20);\n",
    "D = C[A>0]\n",
    "print(C)\n",
    "print(D)"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 86,
   "id": "3be25ba9-2b7f-4a05-86ed-30cd8320b308",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "[0 1 2 3 4 5 6 7 8 9]\n",
      "[20  1  3  4  5  7]\n",
      "[0 1 2 3 4 5 6 7 8 9]\n"
     ]
    }
   ],
   "source": [
    "A = np.arange(10)\n",
    "print(A)\n",
    "indices = np.array([True, True, False, True, True, True, False,
True, False, False])\n",
    "B = A[indices] # not a view, its a copy\n",
    "B[0] = 20\n",
    "print(B)\n",
    "print(A)"
   ]
  },
  {
   "cell_type": "code",
```

```
   "execution_count": 91,
   "id": "2296c1a9-b78e-4f5a-b9fd-27cfaa5c9ac6",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "[[-0.35017549 -0.83304996]\n",
      " [-0.37840125 -0.90019763]]\n",
      "[-0.54402111 -0.29393586  0.22639266  0.53857682  0.35559593 -
0.15431822]\n"
     ]
    }
   ],
   "source": [
    "#print(Z)\n",
    "print(Z[1:3, 2:4])\n",
    "print(Z[-1,:])\n"
   ]
  },
  {
   "cell_type": "markdown",
   "id": "6cad2ef8-ee4d-45fa-9e5d-4c438fd60f06",
   "metadata": {},
   "source": [
    "Palindrome checker"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 112,
   "id": "a9cd367b-2f7a-448f-82ee-d002fc34e979",
   "metadata": {},
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "361\n",
      "163\n",
      "342\n",
      "243\n",
      "324\n",
      "423\n",
      "323\n",
      "323\n",
      "306\n",
      "603\n",
      "304\n",
      "403\n",
      "289\n",
      "982\n",
      "288\n",
```

```
"882\n",
"285\n",
"582\n",
"272\n",
"272\n",
"270\n",
"072\n",
"266\n",
"662\n",
"256\n",
"652\n",
"255\n",
"552\n",
"252\n",
"252\n",
"247\n",
"742\n",
"240\n",
"042\n",
"238\n",
"832\n",
"234\n",
"432\n",
"228\n",
"822\n",
"225\n",
"522\n",
"224\n",
"422\n",
"221\n",
"122\n",
"216\n",
"612\n",
"210\n",
"012\n",
"209\n",
"902\n",
"208\n",
"802\n",
"204\n",
"402\n",
"198\n",
"891\n",
"196\n",
"691\n",
"195\n",
"591\n",
"192\n",
"291\n",
"190\n",
"091\n",
"187\n",
"781\n",
"182\n",
```

```
    "281\n",
    "180\n",
    "081\n",
    "176\n",
    "671\n",
    "170\n",
    "071\n",
    "169\n",
    "961\n",
    "168\n",
    "861\n",
    "165\n",
    "561\n",
    "160\n",
    "061\n",
    "156\n",
    "651\n",
    "154\n",
    "451\n",
    "150\n",
    "051\n",
    "144\n",
    "441\n",
    "143\n",
    "341\n",
    "140\n",
    "041\n",
    "132\n",
    "231\n",
    "130\n",
    "031\n",
    "121\n",
    "121\n",
    "120\n",
    "021\n",
    "110\n",
    "011\n",
    "100\n",
    "001\n"
   ]
  }
 ],
 "source": [
  "import numpy as np\n",
  "\n",
  "a = np.arange(10,20)\n",
  "b = np.outer(a,a)\n",
  "c = np.ravel(b)\n",
  "c.sort()\n",
  "d = np.unique(c)\n",
  "\n",
  "\n",
  "for elem in d[::-1]:\n",
  "    s = str(elem)\n",
```

```
   "    print(s)\n",
   "    print(s[::-1])\n",
   "    break"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 109,
  "id": "4342c8b1-79e3-4a24-919b-f26256fbf75e",
  "metadata": {},
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "[10 11 12 13 14 15 16 17 18 19]\n",
     "(10, 10)\n",
     "(100,)\n",
     "[19 18 17 16 15 14 13 12 11 10]\n"
    ]
   }
  ],
  "source": [
   "print(a)\n",
   "print(b.shape)\n",
   "print(c.shape)\n",
   "print(a[::-1])"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 113,
  "id": "a34678b6-a998-4a89-b9b8-4042380a363d",
  "metadata": {},
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "[False False False False False False False False False False]\n"
    ]
   }
  ],
  "source": [
   "print(a == a[::-1])"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 118,
  "id": "71e5ab3b-bb1e-49b1-bf34-e558bb18ea09",
  "metadata": {},
  "outputs": [
   {
```

```
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "[10 11 12 13 14 15 16 17 18 19 20]\n",
      "[20 19 18 17 16 15 14 13 12 11 10]\n",
      "UniqueCountsResult(values=array([False,  True]), counts=array([10,
1]))\n"
     ]
    }
   ],
   "source": [
    "mm = np.arange(10,21)\n",
    "print(mm)\n",
    "print(mm[::-1])\n",
    "print(np.unique_counts(mm == mm[::-1]))"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "id": "83524463-7093-4ad2-afc7-b562c2d37e5a",
   "metadata": {},
   "outputs": [],
   "source": []
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3 (ipykernel)",
   "language": "python",
   "name": "python3"
  },
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.12.4"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 5
}
```