```
{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {
    "id": "McSxJAwcOdZ1"
   },
   "source": [
    "# Basic Python"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {
    "id": "CU48hgo4Owz5"
   },
   "source": [
    "## 1. Split this string"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 3,
   "metadata": {
    "id": "s07c7JK7Oqt-"
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "['Hi', 'there', 'Sam!']"
      ]
     },
     "execution_count": 3,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "s = \"Hi there Sam!\"\n",
    "s.split()"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {
    "id": "6mGVa3SQYLkb"
  },
  "outputs": [],
  "source": []
 },
 {
  "cell_type": "markdown",
  "metadata": {
   "id": "GH1QBn8HP375"
  },
  "source": [
   "*`italicized text`*## 2. Use .format() to print the following string. \n",
   "\n",
   "### Output should be: The diameter of Earth is 12742 kilometers."
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 4,
  "metadata": {
   "id": "_ZHoml3kPqic"
  },
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "The diameter of Earth is 12742 kilometers\n"
    ]
   }
  ],
  "source": [
   "planet = \"Earth\"\n",
   "diameter = 12742\n",
   "print(\"The diameter of {} is {} kilometers\".format(planet,diameter))"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {
   "id": "HyRyJv6CYPb4"
  },
  "outputs": [],
  "source": []
 },
 {
  "cell_type": "markdown",
  "metadata": {
   "id": "KE74ZEwkRExZ"
  },
  "source": [
   "## 3. In this nest dictionary grab the word \"hello\""
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 5,
  "metadata": {
   "id": "fcVwbCc1QrQI"
  },
  "outputs": [
   {
    "data": {
     "text/plain": [
      "'hello'"
     ]
    },
    "execution_count": 5,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "d = {'k1':[1,2,3,{'tricky':['oh','man','inception',{'target':[1,2,3,'hello']}]}]}\n",
   "d['k1'][3]['tricky'][3]['target'][3]"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": []
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {
   "id": "MvbkMZpXYRaw"
  },
  "outputs": [],
  "source": []
 },
 {
  "cell_type": "markdown",
  "metadata": {
   "id": "bw0vVp-9ddjv"
  },
  "source": [
   "# Numpy"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 7,
  "metadata": {
   "id": "LLiE_TYrhA1O"
  },
  "outputs": [],
  "source": [
   "import numpy as np"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {
   "id": "wOg8hinbgx30"
  },
  "source": [
   "## 4.1 Create an array of 10 zeros? \n",
   "## 4.2 Create an array of 10 fives?"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 13,
  "metadata": {
   "id": "NHrirmgCYXvU"
  },
  "outputs": [
   {
    "data": {
     "text/plain": [
      "array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])"
     ]
    },
    "execution_count": 13,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "x=np.zeros(10)\n",
   "x"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 14,
  "metadata": {
   "id": "e4005lsTYXxx"
  },
  "outputs": [
   {
    "data": {
     "text/plain": [
      "array([5., 5., 5., 5., 5., 5., 5., 5., 5., 5.])"
     ]
    },
    "execution_count": 14,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "y=np.ones(10)*5\n",
   "y"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": null,
  "metadata": {},
  "outputs": [],
  "source": []
 },
 {
  "cell_type": "markdown",
  "metadata": {
   "id": "gZHHDUBvrMX4"
  },
  "source": [
   "## 5. Create an array of all the even integers from 20 to 35"
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 12,
  "metadata": {
   "id": "oAI2tbU2Yag-"
  },
  "outputs": [
     {
      "data": {
       "text/plain": [
        "array([20, 22, 24, 26, 28, 30, 32, 34])"
       ]
      },
      "execution_count": 12,
      "metadata": {},
      "output_type": "execute_result"
     }
    ],
    "source": [
     "A=np.arange(20,35,2)\n",
     "A"
    ]
   },
   {
    "cell_type": "markdown",
    "metadata": {
     "id": "NaOM308NsRpZ"
    },
    "source": [
     "## 6. Create a 3x3 matrix with values ranging from 0 to 8"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 15,
    "metadata": {
     "id": "tOlEVH7BYceE"
    },
    "outputs": [
     {
      "data": {
       "text/plain": [
        "array([[0, 1, 2],\n",
        "       [3, 4, 5],\n",
        "       [6, 7, 8]])"
       ]
      },
      "execution_count": 15,
      "metadata": {},
      "output_type": "execute_result"
     }
    ],
    "source": [
     "matrix=np.arange(0,9).reshape(3,3)\n",
     "matrix"
```

```
 ]
},
{
 "cell_type": "markdown",
 "metadata": {
  "id": "hQ0dnhAQuU_p"
 },
 "source": [
  "## 7. Concatinate a and b \n",
  "## a = np.array([1, 2, 3]), b = np.array([4, 5, 6])"
 ]
},
{
 "cell_type": "code",
 "execution_count": 17,
 "metadata": {
  "id": "rAPSw97aYfE0"
 },
 "outputs": [
  {
   "data": {
    "text/plain": [
     "array([1, 2, 3, 4, 5, 6])"
    ]
   },
   "execution_count": 17,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "a=np.array([1,2,3])\n",
  "b=np.array([4,5,6])\n",
  "np.concatenate((a,b),axis=0)"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {
  "id": "dlPEY9DRwZga"
 },
 "source": [
  "# Pandas"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {
    "id": "ijoYW51zwr87"
   },
   "source": [
    "## 8. Create a dataframe with 3 rows and 2 columns"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 18,
   "metadata": {
    "id": "T5OxJRZ8uvR7"
   },
   "outputs": [],
   "source": [
    "import pandas as pd\n"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 20,
   "metadata": {
    "id": "xNpI_XXoYhs0"
   },
   "outputs": [
    {
     "data": {
      "text/html": [
       "<div>\n",
       "<style scoped>\n",
       "    .dataframe tbody tr th:only-of-type {\n",
       "        vertical-align: middle;\n",
       "    }\n",
       "\n",
       "    .dataframe tbody tr th {\n",
       "        vertical-align: top;\n",
       "    }\n",
       "\n",
       "    .dataframe thead th {\n",
       "        text-align: right;\n",
       "    }\n",
       "</style>\n",
       "<table border=\"1\" class=\"dataframe\">\n",
       "  <thead>\n",
       "    <tr style=\"text-align: right;\">\n",
       "      <th></th>\n",
       "      <th>subjects</th>\n",
       "      <th>marks</th>\n",
       "    </tr>\n",
       "  </thead>\n",
       "  <tbody>\n",
       "    <tr>\n",
       "      <th>0</th>\n",
       "      <td>maths</td>\n",
       "      <td>99</td>\n",
       "    </tr>\n",
       "    <tr>\n",
       "      <th>1</th>\n",
       "      <td>science</td>\n",
       "      <td>97</td>\n",
       "    </tr>\n",
       "    <tr>\n",
       "      <th>2</th>\n",
       "      <td>social</td>\n",
       "      <td>95</td>\n",
       "    </tr>\n",
       "  </tbody>\n",
       "</table>\n",
       "</div>"
      ],
      "text/plain": [
       "  subjects  marks\n",
       "0    maths     99\n",
       "1  science     97\n",
       "2   social     95"
      ]
     },
     "execution_count": 20,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "M={\"subjects\":[\"maths\",\"science\",\"social\"],\"marks\":[99,97,95]}\n",
    "marklist=pd.DataFrame(M)\n",
    "marklist\n",
    "\n"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {
    "id": "UXSmdNclyJQD"
   },
   "source": [
    "## 9. Generate the series of dates from 1st Jan, 2023 to 10th Feb, 2023"
   ]
```

```
},
{
 "cell_type": "code",
 "execution_count": 26,
 "metadata": {
  "id": "dgyC0JhVYl4F"
 },
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "2023-01-01 00:00:00\n",
    "2023-01-02 00:00:00\n",
    "2023-01-03 00:00:00\n",
    "2023-01-04 00:00:00\n",
    "2023-01-05 00:00:00\n",
    "2023-01-06 00:00:00\n",
    "2023-01-07 00:00:00\n",
    "2023-01-08 00:00:00\n",
    "2023-01-09 00:00:00\n",
    "2023-01-10 00:00:00\n",
    "2023-01-11 00:00:00\n",
    "2023-01-12 00:00:00\n",
    "2023-01-13 00:00:00\n",
    "2023-01-14 00:00:00\n",
    "2023-01-15 00:00:00\n",
    "2023-01-16 00:00:00\n",
    "2023-01-17 00:00:00\n",
    "2023-01-18 00:00:00\n",
    "2023-01-19 00:00:00\n",
    "2023-01-20 00:00:00\n",
    "2023-01-21 00:00:00\n",
    "2023-01-22 00:00:00\n",
    "2023-01-23 00:00:00\n",
    "2023-01-24 00:00:00\n",
    "2023-01-25 00:00:00\n",
    "2023-01-26 00:00:00\n",
    "2023-01-27 00:00:00\n",
    "2023-01-28 00:00:00\n",
    "2023-01-29 00:00:00\n",
    "2023-01-30 00:00:00\n",
    "2023-01-31 00:00:00\n",
    "2023-02-01 00:00:00\n",
    "2023-02-02 00:00:00\n",
    "2023-02-03 00:00:00\n",
    "2023-02-04 00:00:00\n",
    "2023-02-05 00:00:00\n",
    "2023-02-06 00:00:00\n",
    "2023-02-07 00:00:00\n",
    "2023-02-08 00:00:00\n",
    "2023-02-09 00:00:00\n",
    "2023-02-10 00:00:00\n"
   ]
  }
 ],
 "source": [
  "p = pd.date_range(start='2023-01-01',end='2023-02-10')\n",
  "for dates in p:\n",
  "  print(dates)"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {
  "id": "ZizSetD-y5az"
 },
 "source": [
  "## 10. Create 2D list to DataFrame\n",
  "\n",
  "lists = [[1, 'aaa', 22],\n",
  "         [2, 'bbb', 25],\n",
  "         [3, 'ccc', 24]]"
 ]
},
{
 "cell_type": "code",
 "execution_count": 22,
 "metadata": {
  "id": "_XMC8aEt0llB"
 },
 "outputs": [],
 "source": [
  "lists = [[1, 'aaa', 22], [2, 'bbb', 25], [3, 'ccc', 24]]"
 ]
},
{
 "cell_type": "code",
 "execution_count": 24,
 "metadata": {
  "id": "knH76sDKYsVX"
 },
 "outputs": [
  {
   "data": {
    "text/html": [
 "<div>\n",
 "<style scoped>\n",
 "    .dataframe tbody tr th:only-of-type {\n",
 "        vertical-align: middle;\n",
 "    }\n",
 "\n",
 "    .dataframe tbody tr th {\n",
 "        vertical-align: top;\n",
 "    }\n",
 "\n",
 "    .dataframe thead th {\n",
 "        text-align: right;\n",
 "    }\n",
 "</style>\n",
 "<table border=\"1\" class=\"dataframe\">\n",
 "  <thead>\n",
 "    <tr style=\"text-align: right;\">\n",
 "      <th></th>\n",
 "      <th>0</th>\n",
 "      <th>1</th>\n",
 "      <th>2</th>\n",
 "    </tr>\n",
 "  </thead>\n",
 "  <tbody>\n",
 "    <tr>\n",
 "      <th>0</th>\n",
 "      <td>1</td>\n",
 "      <td>aaa</td>\n",
 "      <td>22</td>\n",
 "    </tr>\n",
 "    <tr>\n",
 "      <th>1</th>\n",
 "      <td>2</td>\n",
 "      <td>bbb</td>\n",
 "      <td>25</td>\n",
 "    </tr>\n",
 "    <tr>\n",
 "      <th>2</th>\n",
 "      <td>3</td>\n",
 "      <td>ccc</td>\n",
 "      <td>24</td>\n",
 "    </tr>\n",
 "  </tbody>\n",
 "</table>\n",
 "</div>"
],
"text/plain": [
 "   0    1   2\n",
       "0  1  aaa  22\n",
       "1  2  bbb  25\n",
       "2  3  ccc  24"
      ]
     },
     "execution_count": 24,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "Df=pd.DataFrame(lists)\n",
    "Df"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  }
 ],
 "metadata": {
  "colab": {
   "collapsed_sections": [],
   "provenance": []
  },
  "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": "3.9.12"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 1
```

```
}
```