{

"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": [],

"source": [

"s = \"Hi there Sam!\""

]

},

{

"cell\_type": "code",

"execution\_count": 12,

"metadata": {

"id": "6mGVa3SQYLkb"

},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"Hi\n",

"there\n",

"Sam!\n"

]

}

],

"source": [

"arr=s.split(\" \")\n",

"for i in arr:\n",

" print(i)\n",

"#print(arr)"

]

},

{

"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": 14,

"metadata": {

"id": "\_ZHoml3kPqic"

},

"outputs": [],

"source": [

"planet = \"Earth\"\n",

"diameter = 12742"

]

},

{

"cell\_type": "code",

"execution\_count": 22,

"metadata": {

"id": "HyRyJv6CYPb4"

},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"The diameter of Earth is 12742 kilometers\n"

]

}

],

"source": [

"print(\"The diameter of Earth is {} kilometers\".format(diameter))"

]

},

{

"cell\_type": "markdown",

"metadata": {

"id": "KE74ZEwkRExZ"

},

"source": [

"## 3. In this nest dictionary grab the word \"hello\""

]

},

{

"cell\_type": "code",

"execution\_count": 24,

"metadata": {

"id": "fcVwbCc1QrQI"

},

"outputs": [],

"source": [

"d = {'k1':[1,2,3,{'tricky':['oh','man','inception',{'target':[1,2,3,'hello']}]}]}"

]

},

{

"cell\_type": "code",

"execution\_count": 31,

"metadata": {

"id": "MvbkMZpXYRaw"

},

"outputs": [

{

"data": {

"text/plain": [

"'hello'"

]

},

"execution\_count": 31,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"d[\"k1\"][3][\"tricky\"][3][\"target\"][3]"

]

},

{

"cell\_type": "markdown",

"metadata": {

"id": "bw0vVp-9ddjv"

},

"source": [

"# Numpy"

]

},

{

"cell\_type": "code",

"execution\_count": 32,

"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": 39,

"metadata": {

"id": "NHrirmgCYXvU"

},

"outputs": [

{

"data": {

"text/plain": [

"array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])"

]

},

"execution\_count": 39,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"np.zeros(10)"

]

},

{

"cell\_type": "code",

"execution\_count": 40,

"metadata": {

"id": "e4005lsTYXxx"

},

"outputs": [

{

"data": {

"text/plain": [

"array([5., 5., 5., 5., 5., 5., 5., 5., 5., 5.])"

]

},

"execution\_count": 40,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"np.ones(10)\*5"

]

},

{

"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": 43,

"metadata": {

"id": "oAI2tbU2Yag-"

},

"outputs": [

{

"data": {

"text/plain": [

"array([20, 22, 24, 26, 28, 30, 32, 34])"

]

},

"execution\_count": 43,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"np.arange(20,35,2)"

]

},

{

"cell\_type": "markdown",

"metadata": {

"id": "NaOM308NsRpZ"

},

"source": [

"## 6. Create a 3x3 matrix with values ranging from 0 to 8"

]

},

{

"cell\_type": "code",

"execution\_count": 46,

"metadata": {

"id": "tOlEVH7BYceE"

},

"outputs": [

{

"data": {

"text/plain": [

"array([[0, 1, 2],\n",

" [3, 4, 5],\n",

" [6, 7, 8]])"

]

},

"execution\_count": 46,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"np.arange(0,9).reshape(3,3)"

]

},

{

"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": 55,

"metadata": {

"id": "rAPSw97aYfE0"

},

"outputs": [

{

"data": {

"text/plain": [

"(array([[1, 2, 3, 4, 5, 6]]), array([1, 2, 3, 4, 5, 6]))"

]

},

"execution\_count": 55,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"a = np.array([1, 2, 3])\n",

"b = np.array([4, 5, 6])\n",

"np.array([a,b]).reshape(1,-1),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": 58,

"metadata": {

"id": "T5OxJRZ8uvR7"

},

"outputs": [],

"source": [

"import pandas as pd\n"

]

},

{

"cell\_type": "code",

"execution\_count": 62,

"metadata": {

"id": "xNpI\_XXoYhs0",

"scrolled": true

},

"outputs": [

{

"data": {

"text/plain": [

"( a b\n",

" 0 1 2\n",

" 1 3 4\n",

" 2 5 6,\n",

" a b\n",

" 0 1 4\n",

" 1 2 5\n",

" 2 3 6)"

]

},

"execution\_count": 62,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"pd.DataFrame(np.array([[1,2],[3,4],[5,6]]),columns=['a','b'])"

]

},

{

"cell\_type": "code",

"execution\_count": 63,

"metadata": {},

"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>a</th>\n",

" <th>b</th>\n",

" </tr>\n",

" </thead>\n",

" <tbody>\n",

" <tr>\n",

" <th>0</th>\n",

" <td>1</td>\n",

" <td>4</td>\n",

" </tr>\n",

" <tr>\n",

" <th>1</th>\n",

" <td>2</td>\n",

" <td>5</td>\n",

" </tr>\n",

" <tr>\n",

" <th>2</th>\n",

" <td>3</td>\n",

" <td>6</td>\n",

" </tr>\n",

" </tbody>\n",

"</table>\n",

"</div>"

],

"text/plain": [

" a b\n",

"0 1 4\n",

"1 2 5\n",

"2 3 6"

]

},

"execution\_count": 63,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"pd.DataFrame({\"a\":[1,2,3],\"b\":[4,5,6]})"

]

},

{

"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": 64,

"metadata": {

"id": "dgyC0JhVYl4F"

},

"outputs": [

{

"data": {

"text/plain": [

"DatetimeIndex(['2023-01-01', '2023-01-02', '2023-01-03', '2023-01-04',\n",

" '2023-01-05', '2023-01-06', '2023-01-07', '2023-01-08',\n",

" '2023-01-09', '2023-01-10', '2023-01-11', '2023-01-12',\n",

" '2023-01-13', '2023-01-14', '2023-01-15', '2023-01-16',\n",

" '2023-01-17', '2023-01-18', '2023-01-19', '2023-01-20',\n",

" '2023-01-21', '2023-01-22', '2023-01-23', '2023-01-24',\n",

" '2023-01-25', '2023-01-26', '2023-01-27', '2023-01-28',\n",

" '2023-01-29', '2023-01-30', '2023-01-31', '2023-02-01',\n",

" '2023-02-02', '2023-02-03', '2023-02-04', '2023-02-05',\n",

" '2023-02-06', '2023-02-07', '2023-02-08', '2023-02-09',\n",

" '2023-02-10'],\n",

" dtype='datetime64[ns]', freq='D')"

]

},

"execution\_count": 64,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"pd.date\_range(start=\"2023-01-01\",end=\"2023-02-10\")"

]

},

{

"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": 66,

"metadata": {

"id": "\_XMC8aEt0llB"

},

"outputs": [],

"source": [

"lists = [[1, 'aaa', 22], [2, 'bbb', 25], [3, 'ccc', 24]]"

]

},

{

"cell\_type": "code",

"execution\_count": 67,

"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": 67,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"pd.DataFrame(lists)"

]

},

{

"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": "3.9.7"

}

},

"nbformat": 4,

"nbformat\_minor": 1

}