{

"cells": [

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"<h1 style=\"color:Brown\"> Tuples </h1>\n",

"\n",

"- Tuples are an ordered sequnce of mixed data types.\n",

"- Tuples are written as comma-separated elements within parenthesis\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"t = (\"disco\", 12, 4.5)\n",

"t"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"type(t)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"t[0][3]= 't'"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"<h3> Tuples can be defined is various ways </h3>"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### A tuple can be defined without using parenthesis "

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"sample\_tuple = 1,2,3,4\n",

"\n",

"print(sample\_tuple)"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### single value tuple"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"sample\_tuple = 1, \n",

"print(sample\_tuple)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"sample\_tuple = (1,) \n",

"print(sample\_tuple)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"sample\_tuple1 = 1 #This is not a tuple"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"sample\_tuple2 = (1) # This is not a tuple"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Indexing in tuples"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"t = (\"Mumbai\", 84, \"Python\",)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# gives the element at index location 1\n",

"t[1]"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# gives the last element from tuple\n",

"t[-1]"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Slicing"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# Slicing first 3 elements from t\n",

"\n",

"t = (\"Mumbai\", 84, \"Python\", 5, 2, 1)\n",

"\n",

"t[0:3]"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# Slicing last 2 elements from t\n",

"\n",

"t = (\"Mumbai\", 84, \"Python\", 5, 2, 1)\n",

"\n",

"t[-2:]"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# no. of elements in tuple t\n",

"t = (\"Mumbai\", 84, \"Python\", 5, 2, 1)\n",

"len(t)"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Cancatenating tuples"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"tup1 = (\"This\", \"is\", \"Session\", 2)\n",

"tup2 = (\"on\", \"Tuples\")\n",

"\n",

"# Adding contents of tup2 to tup1 and storing in tup3\n",

"\n",

"tup3 = tup1 + tup2\n",

"\n",

"tup3"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### sum() - min() - max()"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"t = (2, 4, 3, 5, 7)\n",

"sum(t)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"t = (2, 4, 3, 5, 7)\n",

"min(t)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"t = (2, 4, 3, 5, 7)\n",

"max(t)"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Immutability of tuples"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"t = (\"USA\", 4, 3, \"Disco\", 7.5)\n",

"\n",

"t[3] = \"Hard Rock\""

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"new\_t = t[0:3] + (\"Hard Rock\",) + t[4:]\n",

"new\_t"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Sorting a tuple"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"t = (2,3,6,4,8,5)\n",

"\n",

"sorted(t)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# Nested Tuples\n",

"\n",

"t = (1,5,\"Disco\", (\"Python\", \"Java\"))\n",

"\n",

"# Access \"Java\" from the nested tuple\n",

"t[3][1]"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"d = {\"India\" : \"INR\", \"USA\" : \"Dollars\", \"France\" : \"Euros\", \"USA\" : \"USD\"}\n",

"d"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"### Packing and Unpacking In Tuples"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"t = (1,2,3,4) # Packing \n",

"(a,b,c,d) = t # Unpacking \n",

"print (a)\n"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### dir() - to view the attributes or methods of an object"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"t = ()\n",

"print(dir(t))"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"<h1 style=\"color:Brown\"> Lists </h1>\n",

"\n",

"- Lists are an ordered sequnce of mixed data types.\n",

"- Lists are written as comma-separated elements within square brackets"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"L = [\"India\", 23, 6, \"Mumbai\"]\n",

"\n",

"L"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# Nested List\n",

"\n",

"L = [\"Chemistry\", \"Biology\", [1989, 2004], (\"Oreily\", \"Pearson\")]\n",

"\n",

"L"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# Indexing\n",

"\n",

"L = [\"Chemistry\", \"Biology\", [1989, 2004], (\"Oreily\", \"Pearson\")]\n",

"\n",

"L[0]"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"L[-3]"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# Slicing\n",

"\n",

"L[0:3]"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# List Concatanetion\n",

"\n",

"L = [\"Chemistry\", \"Biology\", [1989, 2004], (\"Oreily\", \"Pearson\")]\n",

"new\_L = L + [5, 8]\n",

"new\_L"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# Replace \"Biology\" with \"Physics\"\n",

"\n",

"L = [\"Chemistry\", \"Biology\", [1989, 2004], (\"Oreily\", \"Pearson\")]\n",

"\n",

"L[1] = \"Physics\"\n",

"\n",

"L"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# extend()\n",

"\n",

"L = [\"Chemistry\", \"Biology\", [1989, 2004], (\"Oreily\", \"Pearson\")]\n",

"L.extend([5, 8])\n",

"L"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# append()\n",

"\n",

"L = [\"Chemistry\", \"Biology\", [1989, 2004], (\"Oreily\", \"Pearson\")]\n",

"L.append([5, 8])\n",

"L"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# del Command\n",

"\n",

"L = [\"Chemistry\", \"Biology\", [1989, 2004], (\"Oreily\", \"Pearson\")]\n",

"del L[0]\n",

"L"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# pop() \n",

"\n",

"L = [\"Chemistry\", \"Biology\", [1989, 2004], (\"Oreily\", \"Pearson\")]\n",

"\n",

"L.pop()\n",

"\n",

"L"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# remove()\n",

"\n",

"L = [\"Chemistry\", \"Biology\", [1989, 2004], (\"Oreily\", \"Pearson\")]\n",

"\n",

"L.remove(\"Chemistry\")\n",

"\n",

"L"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# Sorting Lists\n",

"\n",

"l = [32, 24, 65, 9]\n",

"l.sort()\n",

"l"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"l.sort(reverse= True)\n",

"l"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Difference between sort and sorted"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"A = [\"Orange\", \"Strawberry\", \"Mango\"]\n",

"B = A.sort()\n",

"\n",

"print(A)\n",

"print(B)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"A = [\"Orange\", \"Strawberry\", \"Mango\"]\n",

"C = sorted(A)\n",

"\n",

"print(A)\n",

"print(C)"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"#### Shallow Copying"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"A = [\"Orange\", \"Strawberry\", \"Mango\"]\n",

"B = A\n",

"\n",

"A[0] = \"Apple\""

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"A"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"B"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"A = [\"Orange\", \"Strawberry\", \"Mango\"]\n",

"B = A[:]\n",

"\n",

"A[0] = \"Apple\""

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"A"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"B"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"<h1 style=\"color:Brown\"> Sets </h1>\n",

"\n",

"- Sets are a type of collection like lists and tuples, storing mixed data.\n",

"- Sets are enclosed within curly brackets and elements are written as comma-separated.\n",

"- Sets are unordered\n",

"- Sets does not allow duplicates"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"l = [1,3,2,4,5,5]\n",

"\n",

"set\_l = set(l)\n",

"\n",

"set\_l"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"len(set\_l)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"a = {1, 2, 3, 4, 5}\n",

"\n",

"a.add(\"India\")\n",

"\n",

"a"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"a.add(\"India\")\n",

"a"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"a = {1, 2, 3, 4, 5, \"India\"}\n",

"a.remove(\"India\")\n",

"a"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# Set Operations\n",

"\n",

"A = {0, 2, 4, 6, 8}\n",

"B = {1, 2, 3, 4, 5}"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"print(A | B) # Union"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"print(A & B) # Intersection"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"print(A - B) # Difference"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"print(A ^ B) # Symmetric Difference"

]

},

{

"cell\_type": "markdown",

"metadata": {},

"source": [

"<h1 style=\"color:Brown\"> Dictionaries </h1>\n",

"\n",

"- A dictionary stores element as keys and values pairs.\n",

"- The key is like an index, its is always unique and immutable. \n",

"- The values are the objects that contain information.\n",

"- Values are accessed using their keys.\n",

"- Each key is followed by a value separated by a colon. \n",

"- The values can be immutable, mutable, and duplicates. \n",

"- Each key and value pair is separated by a comma enclosed inside curly brackets."

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# Creating a dictionary\n",

"\n",

"d = {\"India\" : \"INR\", \"USA\" : \"USD\", \"France\" : \"Euros\"}\n",

"\n",

"# Access value using keys\n",

"\n",

"d[\"India\"]"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# Replace the value for a key in a dictionary\n",

"\n",

"d[\"India\"] = \"Rs\"\n",

"\n",

"d"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# Insert new key value pair into a dictionary\n",

"\n",

"d[\"Japan\"] = \"Yen\"\n",

"\n",

"d"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# Deleting a key value pair\n",

"\n",

"del d[\"France\"]\n",

"\n",

"d"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": [

"# Sorting a dictionary\n",

"\n",

"sorted(d)"

]

},

{

"cell\_type": "code",

"execution\_count": 1,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"dict\_values(['INR', 'USD', 'Euros'])"

]

},

"execution\_count": 1,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"# Values() method\n",

"\n",

"d = {\"India\" : \"INR\", \"USA\" : \"USD\", \"France\" : \"Euros\"}\n",

"d.values()"

]

},

{

"cell\_type": "code",

"execution\_count": 2,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"dict\_keys(['India', 'USA', 'France'])"

]

},

"execution\_count": 2,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"# Keys() method\n",

"\n",

"d.keys()"

]

},

{

"cell\_type": "code",

"execution\_count": 3,

"metadata": {},

"outputs": [

{

"data": {

"text/plain": [

"{'India': 'Rupee', 'USA': 'USD', 'France': 'Euros'}"

]

},

"execution\_count": 3,

"metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"# Updat() method\n",

"\n",

"d.update({'India':'Rupee'})\n",

"d"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {},

"outputs": [],

"source": []

}

],

"metadata": {

"kernelspec": {

"display\_name": "Python 3",

"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": "3.7.3"

}

},

"nbformat": 4,

"nbformat\_minor": 2

}