```
{
  "cells": [
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "fwU2iooz85jt"
      },
      "source": [
        "## Exercises\n",
        "\n",
        "Answer the questions or complete the tasks outlined in bold
below, use the specific method described if applicable."
      ]
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "SzBQQ_ml85j1"
      },
      "source": [
        "** What is 7 to the power of 4?**"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 2,
      "metadata": {
        "id": "UhvE4PBC85j3",
        "outputId": "4e5ce783-6066-4843-95a5-ed03a1821c49",
        "colab": {
          "base_uri": "https://localhost:8080/"
        }
      },
      "outputs": [
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "2401"
            ]
          },
          "metadata": {},
          "execution_count": 2
        }
      ],
      "source": [
        "7**4"
      ]
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "ds8G9S8j85j6"
      },
      "source": [
        "** Split this string:**\n",
        "\n",
        "    s = \"Hi there Sam!\"\n",
```

```
    "    \n",
    "**into a list. **"
  ]
},
{
  "cell_type": "code",
  "execution_count": 4,
  "metadata": {
    "collapsed": true,
    "id": "GD_Tls3H85j7"
  },
  "outputs": [],
  "source": [
    "S='Hi there Sam!'"
  ]
},
{
  "cell_type": "code",
  "execution_count": 6,
  "metadata": {
    "id": "RRGOKoai85j8",
    "outputId": "29d1f9a5-4c1c-4f43-8a44-f8ba3fc8b999",
    "colab": {
      "base_uri": "https://localhost:8080/"
    }
  },
  "outputs": [
    {
      "output_type": "execute_result",
      "data": {
        "text/plain": [
          "['Hi', 'there', 'Sam!']"
        ]
      },
      "metadata": {},
      "execution_count": 6
    }
  ],
  "source": [
    "S.split()"
  ]
},
{
  "cell_type": "markdown",
  "metadata": {
    "id": "_bBNOu-785j9"
  },
  "source": [
    "** Given the variables:**\n",
    "\n",
    "    planet = \"Earth\"\n",
    "    diameter = 12742\n",
    "\n",
    "** Use .format() to print the following string: **\n",
    "\n",
    "    The diameter of Earth is 12742 kilometers."
  ]
},
```

```
    {
      "cell_type": "code",
      "execution_count": 7,
      "metadata": {
        "collapsed": true,
        "id": "2TrzmDcS85j-"
      },
      "outputs": [],
      "source": [
        "planet=\"Earth\""
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 8,
      "metadata": {
        "id": "s_dQ7_xc85j_",
        "outputId": "32b168f1-da6b-45e1-f9cc-ce8fb759ad90",
        "colab": {
          "base_uri": "https://localhost:8080/"
        }
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "The diameter of Earth is 12742 kilometers.\n"
          ]
        }
      ],
      "source": [
        "diameter=12742\n",
        "print('The diameter of {} is {} kilometers.'
.format(planet,diameter));"
      ]
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "QAKtN7Hh85kB"
      },
      "source": [
        "** Given this nested list, use indexing to grab the word
\"hello\" **"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 11,
      "metadata": {
        "collapsed": true,
        "id": "-7dzQDyK85kD"
      },
      "outputs": [],
      "source": [
        "lst = [1,2,[3,4],[5,[100,200,['hello']],23,11],1,7]"
      ]
```

```
    },
    {
      "cell_type": "code",
      "execution_count": 12,
      "metadata": {
        "id": "6m5C0sTW85kE",
        "outputId": "17ba953e-1102-4e49-bf5c-0595e5726c14",
        "colab": {
          "base_uri": "https://localhost:8080/"
        }
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "['hello']\n"
          ]
        }
      ],
      "source": [
        "lst = [1,2,[3,4],[5,[100,200,['hello']],23,11],1,7]\n",
        "a=lst[3][1][2]\n",
        "print(a)"
      ]
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "9Ma7M4a185kF"
      },
      "source": [
        "** Given this nest dictionary grab the word \"hello\". Be
prepared, this will be annoying/tricky **"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 9,
      "metadata": {
        "id": "vrYAxSYN85kG"
      },
      "outputs": [],
      "source": [
        "d =
{'k1':[1,2,3,{'tricky':['oh','man','inception',{'target':[1,2,3,'hello']}
]}]}"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 13,
      "metadata": {
        "id": "FlILSdm485kH",
        "outputId": "28ee05b3-019b-415d-e548-512b3bc3f7e7",
        "colab": {
          "base_uri": "https://localhost:8080/"
        }
      }
```

```
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "hello\n"
          ]
        }
      ],
      "source": [
        "print(d['k1'][3][\"tricky\"][3]['target'][3])"
      ]
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "FInV_FKB85kI"
      },
      "source": [
        "** What is the main difference between a tuple and a list? **"
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "Tuples are immutable whereas Lists are mutable. Tuples consumes less memory whereas Lists consume more memory. Tuples does not have many built-in methods whereas Lists have several built-in methods."
      ],
      "metadata": {
        "id": "8lSRTmhfAUe-"
      }
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "zP-j0HZj85kK"
      },
      "source": [
        "** Create a function that grabs the email website domain from a string in the form: **\n",
        "\n",
        "    user@domain.com\n",
        "    \n",
        "**So for example, passing \"user@domain.com\" would return: domain.com**"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 30,
      "metadata": {
        "id": "unvEAwjk85kL",
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "outputId": "8cac6e2b-6e9d-40b6-a08a-a925ce6df1af"
```

```
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Please enter your emailmanovelammal@hotmail.com\n",
            "Your domain is: hotmail.com\n"
          ]
        }
      ],
      "source": [
        "def domainGet(email):\n",
        "  print(\"Your domain is: \"+email.split('@')[-1])\n",
        "email=input(\"Please enter your email\")\n",
        "domainGet(email) "
      ]
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "gYydb-y085kM"
      },
      "source": [
        "** Create a basic function that returns True if the word 'dog' is contained in the input string. Don't worry about edge cases like a punctuation being attached to the word dog, but do account for capitalization. **"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 15,
      "metadata": {
        "collapsed": true,
        "id": "Q4ldLGV785kM",
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "outputId": "7d221d3a-236c-414c-d172-de4fd62d5048"
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "True\n"
          ]
        }
      ],
      "source": [
        "def finddog(st):\n",
        "  if 'dog' in st.lower():\n",
        "    print(\"True\")\n",
        "  else:\n",
        "    print(\"False\")  \n",
        "st=\"is there a dog here?\"\n",
        "finddog(st)  "
```

```
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 16,
    "metadata": {
     "id": "EqH6b7yv85kN",
     "outputId": "4b65eabe-fcc3-4425-9e6e-a730a272d065",
     "colab": {
      "base_uri": "https://localhost:8080/"
     }
    },
    "outputs": [
     {
      "output_type": "stream",
      "name": "stdout",
      "text": [
       "True\n"
      ]
     }
    ],
    "source": [
     "finddog(\"is there a dog here?\")"
    ]
   },
   {
    "cell_type": "markdown",
    "metadata": {
     "id": "AyHQFALC85kO"
    },
    "source": [
     "** Create a function that counts the number of times the word \"dog\" occurs in a string. Again ignore edge cases. **"
    ]
   },
   {
    "cell_type": "code",
    "execution_count": 17,
    "metadata": {
     "id": "6hdc169585kO",
     "colab": {
      "base_uri": "https://localhost:8080/"
     },
     "outputId": "aff258eb-4555-483d-e600-26247a40921f"
    },
    "outputs": [
     {
      "output_type": "stream",
      "name": "stdout",
      "text": [
       "2\n"
      ]
     }
    ],
    "source": [
     "def countdogs(value):\n",
     "  cnt=0;\n",
     "  for word in value.lower().split():\n",
```

```
        "    if word == 'dog' or word == 'dogs':\n",
        "      cnt+=1\n",
        "  print(cnt)\n",
        "value='this dog run faster than the other dogs'      \n",
        "countdogs(value) "
      ]
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "3n7jJt4k85kP"
      },
      "source": [
        "### Problem\n",
        "**You are driving a little too fast, and a police officer stops you. Write a function\n",
        "  to return one of 3 possible results: \"No ticket\", \"Small ticket\", or \"Big Ticket\". \n",
        "  If your speed is 60 or less, the result is \"No Ticket\". If speed is between 61 \n",
        "  and 80 inclusive, the result is \"Small Ticket\". If speed is 81 or more, the result is \"Big    Ticket\". Unless it is your birthday (encoded as a boolean value in the parameters of the function) -- on your birthday, your speed can be 5 higher in all \n",
        "  cases. **"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 20,
      "metadata": {
        "collapsed": true,
        "id": "nvXMkvWk85kQ"
      },
      "outputs": [],
      "source": [
        "def caught_speeding(speed, is_birthday):\n",
        "    \n",
        "    if is_birthday:\n",
        "        speeding = speed - 5\n",
        "    else:\n",
        "        speeding = speed\n",
        "    \n",
        "    if speeding > 80:\n",
        "        return 'Big Ticket'\n",
        "    elif speeding > 60:\n",
        "        return 'Small Ticket'\n",
        "    else:\n",
        "        return 'No Ticket'"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 21,
      "metadata": {
        "id": "BU_UZcyk85kS",
        "outputId": "d918942f-4895-45a9-b6d6-129007c84023",
        "colab": {
```

```
      "base_uri": "https://localhost:8080/",
      "height": 36
    }
  },
  "outputs": [
    {
      "output_type": "execute_result",
      "data": {
        "text/plain": [
          "'Big Ticket'"
        ],
        "application/vnd.google.colaboratory.intrinsic+json": {
          "type": "string"
        }
      },
      "metadata": {},
      "execution_count": 21
    }
  ],
  "source": [
    "caught_speeding(90,True)"
  ]
},
{
  "cell_type": "code",
  "execution_count": 22,
  "metadata": {
    "id": "p1AGJ7DM85kR",
    "outputId": "e3654585-5b20-45d1-ee6f-085f72bc00b5",
    "colab": {
      "base_uri": "https://localhost:8080/",
      "height": 36
    }
  },
  "outputs": [
    {
      "output_type": "execute_result",
      "data": {
        "text/plain": [
          "'Small Ticket'"
        ],
        "application/vnd.google.colaboratory.intrinsic+json": {
          "type": "string"
        }
      },
      "metadata": {},
      "execution_count": 22
    }
  ],
  "source": [
    "caught_speeding(61,False)"
  ]
},
{
  "cell_type": "markdown",
  "source": [
```

```
        "Create an employee list with basic salary values(at least 5 values for 5 employees)  and using a for loop retreive each employee salary and calculate total salary expenditure. "
      ],
      "metadata": {
        "id": "Tie4rC7_kAOC"
      }
    },
    {
      "cell_type": "code",
      "source": [
        "employee = [15000,20000,25000,30000,40000]\n",
        "sum=0\n",
        "for i in employee:\n",
        "  sum+=i\n",
        "  print(i)\n",
        "print(sum)"
      ],
      "metadata": {
        "id": "R5-CdXSKjacN",
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "outputId": "0c664c68-eb0d-42a6-b845-3612794e72c9"
      },
      "execution_count": 23,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "15000\n",
            "20000\n",
            "25000\n",
            "30000\n",
            "40000\n",
            "130000\n"
          ]
        }
      ]
    },
    {
      "cell_type": "markdown",
      "source": [
        "Create two dictionaries in Python:\n",
        "\n",
        "First one to contain fields as Empid,  Empname,  Basicpay\n",
        "\n",
        "Second dictionary to contain fields as DeptName,  DeptId.\n",
        "\n",
        "Combine both dictionaries. "
      ],
      "metadata": {
        "id": "-L1aiFqRkF5s"
      }
    },
    {
      "cell_type": "code",
```

```
      "source": [

"dict_1={\"Empid\":\"1\",\"Empname\":\"mano\",\"Basicpay\":\"55000\"}\n",
        "dict_2={\"DeptName\":\"DevOps\",\"DeptId\":\"1018\"}\n",
        "dict_3={**dict_1, **dict_2}\n",
        "print(dict_3)"
      ],
      "metadata": {
        "id": "8ugVoEe0kOsk",
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "outputId": "7d1d5437-f473-4463-b51e-6ca98be1e249"
      },
      "execution_count": 29,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "{'Empid': '1', 'Empname': 'mano', 'Basicpay': '55000',
'DeptName': 'DevOps', 'DeptId': '1018'}\n"
          ]
        }
      ]
    }
  ],
  "metadata": {
    "colab": {
      "provenance": []
    },
    "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": "3.8.5"
    }
  },
  "nbformat": 4,
  "nbformat_minor": 0
}
```