```
{
  "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": 1,
      "metadata": {
        "id": "UhvE4PBC85j3",
        "outputId": "7ff1a343-c024-4832-a25e-873fdb3c483a",
        "colab": {
          "base_uri": "https://localhost:8080/"
        }
      },
      "outputs": [
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "2401"
            ]
          },
          "metadata": {},
          "execution_count": 1
        }
      ],
      "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": 5,
   "metadata": {
    "collapsed": true,
    "id": "GD_Tls3H85j7"
   },
   "outputs": [],
   "source": [
    "s = \"Hi there Sam!\""
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 6,
   "metadata": {
    "id": "RRGOKoai85j8",
    "outputId": "d2d1209e-3f85-44f9-b7f9-8fe29277e456",
    "colab": {
     "base_uri": "https://localhost:8080/"
    }
   },
   "outputs": [
    {
     "output_type": "stream",
     "name": "stdout",
     "text": [
      "['Hi', 'there', 'Sam!']\n"
     ]
    }
```

```
    ],
    "source": [
      "print(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": [
      "txt = \"The diameter of {planet} is {diameter}
kilometers.\".format(planet = \"Earth\", diameter=12742)"
    ]
  },
  {
    "cell_type": "code",
    "execution_count": 8,
    "metadata": {
      "id": "s_dQ7_xc85j_",
      "outputId": "5fc0baec-b1fb-4045-ba36-dfe4960e3490",
      "colab": {
        "base_uri": "https://localhost:8080/"
      }
    },
    "outputs": [
      {
```

```
      "output_type": "stream",
      "name": "stdout",
      "text": [
        "The diameter of Earth is 12742 kilometers.\n"
      ]
    }
  ],
  "source": [
    "print(txt)"
  ]
},
{
  "cell_type": "markdown",
  "metadata": {
    "id": "QAKtN7Hh85kB"
  },
  "source": [
    "** Given this nested list, use indexing to grab the word \"hello\" **"
  ]
},
{
  "cell_type": "code",
  "execution_count": 10,
  "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": 11,
  "metadata": {
    "id": "6m5C0sTW85kE",
    "outputId": "20a79e6a-998a-4540-e5f7-2fa6a83bdac3",
    "colab": {
      "base_uri": "https://localhost:8080/"
    }
  },
  "outputs": [
    {
      "output_type": "stream",
      "name": "stdout",
```

```
          "text": [
            "hello\n"
          ]
        }
      ],
      "source": [
        "print(lst[3][1][2][0])"
      ]
    },
    {
      "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": 13,
      "metadata": {
        "id": "vrYAxSYN85kG"
      },
      "outputs": [],
      "source": [
        "d =
{'k1':[1,2,3,{'tricky':['oh','man','inception',{'target':[1,2,3,'hello']}]}]}"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 14,
      "metadata": {
        "id": "FlILSdm485kH",
        "outputId": "12afe734-d119-48c8-a542-14707ab310a0",
        "colab": {
          "base_uri": "https://localhost:8080/"
        }
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
```

```
            "hello\n"
           ]
         }
       ],
       "source": [
         "print(d['k1'][3][\"tricky\"][3]['target'][3])\n"
       ]
     },
     {
       "cell_type": "markdown",
       "metadata": {
         "id": "FInV_FKB85kI"
       },
       "source": [
         "** What is the main difference between a tuple and a list? **"
       ]
     },
     {
       "cell_type": "code",
       "execution_count": 17,
       "metadata": {
         "collapsed": true,
         "id": "_VBWf00q85kJ"
       },
       "outputs": [],
       "source": [
         "t = (1, 2, 3)\n",
         "list = [1, 2, 3, 4, 5]\n",
         "\n",
         "#tuple is immutable, and list is muutable"
       ]
     },
     {
       "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": 42,
  "metadata": {
    "collapsed": true,
    "id": "unvEAwjk85kL",
    "colab": {
      "base_uri": "https://localhost:8080/"
    },
    "outputId": "3e3a9c81-515d-4f4b-fba8-e2085308b77b"
  },
  "outputs": [
    {
      "output_type": "stream",
      "name": "stdout",
      "text": [
        "Emailbavyaajoghee@gmail.com\n",
        "Your domain is: gmail.com\n"
      ]
    }
  ],
  "source": [
    "def domain(email):\n",
    "    print(\"Your domain is: \" + email.split('@')[-1])\n",
    "\n",
    "email = input(\"Email\")\n",
    "domain(email)"
  ]
},
{
  "cell_type": "code",
  "execution_count": null,
  "metadata": {
    "id": "Gb9dspLC85kL",
    "outputId": "4216116b-da08-45a2-9545-d6b13bcefaeb"
  },
  "outputs": [
    {
      "data": {
        "text/plain": [
          "'domain.com'"
        ]
      },
      "execution_count": 26,
      "metadata": {
```

```
            "tags": []
          },
          "output_type": "execute_result"
        }
      ],
      "source": []
    },
    {
      "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": 25,
      "metadata": {
        "collapsed": true,
        "id": "Q4ldLGV785kM"
      },
      "outputs": [],
      "source": [
        "def findDog(st):\n",
        "    if 'dog' in st.lower():\n",
        "        print(\"True\")\n",
        "    else:\n",
        "        print(\"False\")"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 26,
      "metadata": {
        "id": "EqH6b7yv85kN",
        "outputId": "825e7461-9e96-4b35-c915-1fe99c6c781c",
        "colab": {
          "base_uri": "https://localhost:8080/"
        }
      },
      "outputs": [
        {
```

```
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "True\n"
          ]
        }
      ],
      "source": [
        "findDog(\"I like dog\")"
      ]
    },
    {
      "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": 23,
      "metadata": {
        "id": "6hdc169585kO"
      },
      "outputs": [],
      "source": [
        "def countDog(st):\n",
        "    count = 0\n",
        "    for word in st.lower().split():\n",
        "        if word == 'dog':\n",
        "            count += 1\n",
        "    return count"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 24,
      "metadata": {
        "id": "igzsvHb385kO",
        "outputId": "186c6941-c507-450a-fd1d-711f363531b3",
        "colab": {
          "base_uri": "https://localhost:8080/"
        }
```

```
      },
      "outputs": [
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "2"
            ]
          },
          "metadata": {},
          "execution_count": 24
        }
      ],
      "source": [
        "countDog(\"I like having a dog as a dog is cute\")"
      ]
    },
    {
      "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": 29,
      "metadata": {
        "collapsed": true,
        "id": "nvXMkvWk85kQ"
      },
      "outputs": [],
      "source": [
```

```
    "def caught_speeding(speed, is_birthday):\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": 30,
  "metadata": {
    "id": "BU_UZcyk85kS",
    "outputId": "f4b8f222-3435-466c-9704-f09b354f3a76",
    "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": 30
    }
  ],
  "source": [
    "caught_speeding(100, True)"
  ]
},
{
  "cell_type": "code",
```

```
      "execution_count": 34,
      "metadata": {
        "id": "p1AGJ7DM85kR",
        "outputId": "c68750df-cc5d-4385-ef91-6d2167f9fbec",
        "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": 34
        }
      ],
      "source": [
        "caught_speeding(80, True)"
      ]
    },
    {
      "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": [
        "j=0\n",
        "print(\"Employees Salary\")\n",
        "emp=[10000, 15000, 30000, 14500, 20000]\n",
        "for i in emp:\n",
```

```
    "  print(i)\n",
    "  j=j+i\n",
    "print(\"Total Salary Expenditure:\",j)"
   ],
   "metadata": {
    "id": "R5-CdXSKjacN",
    "colab": {
     "base_uri": "https://localhost:8080/"
    },
    "outputId": "c51d5839-0231-41c5-b959-5b35dd5e0626"
   },
   "execution_count": 41,
   "outputs": [
    {
     "output_type": "stream",
     "name": "stdout",
     "text": [
      "Employees Salary\n",
      "10000\n",
      "15000\n",
      "30000\n",
      "14500\n",
      "20000\n",
      "Total Salary Expenditure: 89500\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": [
```

```
      "d1 = {'Empid', 'Empname', 'Basicpay'}\n",
      "d2 = {'DeptName', 'DeptId'}\n",
      "d2.update(d1)\n",
      "print(d2)"
    ],
    "metadata": {
      "id": "8ugVoEe0kOsk",
      "colab": {
        "base_uri": "https://localhost:8080/"
      },
      "outputId": "cc79981a-fa8d-4300-8515-1f4d0c3e6b80"
    },
    "execution_count": 37,
    "outputs": [
      {
        "output_type": "stream",
        "name": "stdout",
        "text": [
          "{'Basicpay', 'Empid', 'DeptName', 'DeptId', 'Empname'}\n"
        ]
      }
    ]
  }
],
"metadata": {
  "colab": {
    "provenance": [],
    "collapsed_sections": []
  },
  "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
}
```