```
{
  "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": null,
      "metadata": {
        "id": "UhvE4PBC85j3",
        "outputId": "36745cc9-0492-45ed-ffe6-e70841478bee",
        "colab": {
          "base_uri": "https://localhost:8080/"
        }
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "2401\n"
          ]
        }
      ],
      "source": [
        "print(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": null,
  "metadata": {
    "collapsed": true,
    "id": "GD_Tls3H85j7"
  },
  "outputs": [],
  "source": [
    "string=\"Hi there Sam!\""
  ]
},
{
  "cell_type": "code",
  "execution_count": null,
  "metadata": {
    "id": "RRGOKoai85j8",
    "outputId": "cc52f0d8-2ed1-4b4d-e956-5bbeb332cdc2"
  },
  "outputs": [
    {
      "data": {
        "text/plain": [
          "['Hi', 'there', 'dad!']"
        ]
      },
      "execution_count": 3,
      "metadata": {
        "tags": []
      },
      "output_type": "execute_result"
    }
  ],
  "source": [
    "print(string.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": null,
  "metadata": {
```

```
        "collapsed": true,
        "id": "2TrzmDcS85j-",
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "outputId": "9280926c-a4c6-46f7-87a7-a77318b67183"
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "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": "s_dQ7_xc85j_",
        "outputId": "dd691fa4-e131-4b35-e0b1-b56ba260a0f3",
        "colab": {
          "base_uri": "https://localhost:8080/"
        }
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "The diameter of Earth is 12742 kilometers.\n"
          ]
        }
      ],
      "source": [
        "planet = \"Earth\"\n",
        "diameter = 12742\n",
        "print(f\"The diameter of {planet} is {diameter} kilometers.\")"
      ]
    },
    {
      "cell_type": "markdown",
      "metadata": {
        "id": "QAKtN7Hh85kB"
      },
      "source": [
        "** Given this nested list, use indexing to grab the word
\"hello\" **"
      ]
    },
```

```
    {
      "cell_type": "code",
      "execution_count": null,
      "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": null,
      "metadata": {
        "id": "6m5C0sTW85kE",
        "outputId": "4c86d9d6-3735-413d-cfb9-b5a3011556e4",
        "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",
        "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": null,
      "metadata": {
        "id": "vrYAxSYN85kG"
      },
      "outputs": [],
      "source": [
        "d =
{'k1':[1,2,3,{'tricky':['oh','man','inception',{'target':[1,2,3,'hello']}
]}]}"
      ]
```

```
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "FlILSdm485kH",
        "outputId": "5c10ef1e-f77f-40c3-d69a-c84c015a857e",
        "colab": {
          "base_uri": "https://localhost:8080/"
        }
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "hello\n"
          ]
        }
      ],
      "source": [
        "d =
{'k1':[1,2,3,{'tricky':['oh','man','inception',{'target':[1,2,3,'hello']}
]}]}\n",
        "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": null,
      "metadata": {
        "collapsed": true,
        "id": "_VBWf00q85kJ",
        "colab": {
          "base_uri": "https://localhost:8080/",
          "height": 35
        },
        "outputId": "17d74864-24fd-43e8-cdca-1f1b6e86ee17"
      },
      "outputs": [
        {
          "output_type": "execute_result",
          "data": {
            "text/plain": [
              "'\\ntuple is immutable,and list is mutable\\n'"
            ],
            "application/vnd.google.colaboratory.intrinsic+json": {
              "type": "string"
            }
```

```
        },
        "metadata": {},
        "execution_count": 12
      }
    ],
    "source": [
      "t=(1,2,3)\n",
      "list=[1,2,3,4,5]\n",
      "'''\n",
      "tuple is immutable,and list is mutable\n",
      "'''"
    ]
  },
  {
    "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": 1,
    "metadata": {
      "collapsed": true,
      "id": "unvEAwjk85kL",
      "colab": {
        "base_uri": "https://localhost:8080/"
      },
      "outputId": "f1fa3ce9-aa9f-4444-f8f8-03f8e608b22c"
    },
    "outputs": [
      {
        "output_type": "stream",
        "name": "stdout",
        "text": [
          "Please enter your email: >user@domain.com\n",
          "Your domain is: domain.com\n"
        ]
      }
    ],
    "source": [
      "def domainGet(email):\n",
      "    print(\"Your domain is: \" + email.split('@')[-1])\n",
      "\n",
      "email = input(\"Please enter your email: >\")\n",
      "domainGet(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": 2,
      "metadata": {
        "collapsed": true,
        "id": "Q4ldLGV785kM",
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "outputId": "73b85b79-b372-4ac7-b34d-c05f4aff3e51"
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Please key a string: >dog\n",
            "True\n"
          ]
        }
      ],
      "source": [
        "def findDog(st):\n",
        "    if 'dog' in st.lower():\n",
```

```
        "        print(\"True\")\n",
        "    else:\n",
        "        print(\"False\")\n",
        "\n",
        "st = input(\"Please key a string: >\")\n",
        "findDog(st)\n"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "EqH6b7yv85kN",
        "outputId": "e7909af1-8df1-4534-fc8c-27b03d7369e5"
      },
      "outputs": [
        {
          "data": {
            "text/plain": [
              "True"
            ]
          },
          "execution_count": 28,
          "metadata": {
            "tags": []
          },
          "output_type": "execute_result"
        }
      ],
      "source": []
    },
    {
      "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": 3,
      "metadata": {
        "id": "6hdc169585kO",
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
        "outputId": "d03f207e-4c55-46e6-c50e-e0afa7e7234e"
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Please enter your string: dog\n",
            "1\n"
```

```
          ]
        }
      ],
      "source": [
        "#-*- coding: utf-8 -*-\n",
        "#User/johnny.lu/Download/python3/PCCE/ex9.py\n",
        "\n",
        "'''\n",
        "**Create a function that counts the number of times the word \"dog\"\n",
        "occurs in a string. Again ignore edge cases.**\n",
        "'''\n",
        "\n",
        "string = input(\"Please enter your string: \")\n",
        "\n",
        "def countdogs(string):\n",
        "    count = 0\n",
        "    for word in string.lower().split():\n",
        "        if word == 'dog' or word == 'dogs':\n",
        "            count = count + 1\n",
        "            print(count)\n",
        "\n",
        "countdogs(string)"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": null,
      "metadata": {
        "id": "igzsvHb385kO",
        "outputId": "0602a2b5-0b18-48d8-e2d4-fe644cbccf8a"
      },
      "outputs": [
        {
          "data": {
            "text/plain": [
              "2"
            ]
          },
          "execution_count": 31,
          "metadata": {
            "tags": []
          },
          "output_type": "execute_result"
        }
      ],
      "source": []
    },
    {
      "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": null,
      "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": 4,
      "metadata": {
        "id": "BU_UZcyk85kS",
        "outputId": "f8200b2a-8969-43d4-ed46-6ba780d37e33",
        "colab": {
          "base_uri": "https://localhost:8080/"
        }
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Please enter the speed(km/h)(only number please): \n",
            "\n",
            "> 63\n",
            "Please enter your birthday: (in DD/MM/YYYY format)\n",
            "\n",
            "> 28/04/2002\n",
            "You get a small ticket\n"
          ]
```

```
        }
      ],
      "source": [
        "#-*- coding:utf-8 -*-\n",
        "#/User/johnny.lu/python3/PCCE/final_problem.py\n",
        "\n",
        "print(\"Please enter the speed(km/h)(only number please):
\\n\")\n",
        "speed = int(input(\"> \"))\n",
        "\n",
        "print(\"Please enter your birthday: (in DD/MM/YYYY
format)\\n\")\n",
        "birthday = str(input(\"> \"))\n",
        "\n",
        "def speeding(speed, birthday):\n",
        "    if birthday == '29/08/1989':\n",
        "        s = speed - 5\n",
        "    else:\n",
        "        s = speed\n",
        "\n",
        "    if s <= 60:\n",
        "        print(\"You pass.\")\n",
        "    elif s > 61 and s <= 80:\n",
        "        print(\"You get a small ticket\")\n",
        "    else:\n",
        "        print(\"You get a big ticket.\")\n",
        "\n",
        "speeding(speed, birthday)"
      ]
    },
    {
      "cell_type": "code",
      "execution_count": 6,
      "metadata": {
        "id": "p1AGJ7DM85kR",
        "outputId": "c5a306a0-85d5-45c7-88e0-faa017d1345d",
        "colab": {
          "base_uri": "https://localhost:8080/"
        }
      },
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "Please enter the speed(km/h)(only number please): \n",
            "\n",
            "> 82\n",
            "Please enter your birthday: (in DD/MM/YYYY format)\n",
            "\n",
            "> 03/6/2000\n",
            "You get a big ticket.\n"
          ]
        }
      ],
      "source": [
        "#-*- coding:utf-8 -*-\n",
        "#/User/johnny.lu/python3/PCCE/final_problem.py\n",
```

```
        "\n",
        "print(\"Please enter the speed(km/h)(only number please):
\\n\")\n",
        "speed = int(input(\"> \"))\n",
        "\n",
        "print(\"Please enter your birthday: (in DD/MM/YYYY
format)\\n\")\n",
        "birthday = str(input(\"> \"))\n",
        "\n",
        "def speeding(speed, birthday):\n",
        "    if birthday == '29/08/1989':\n",
        "        s = speed - 5\n",
        "    else:\n",
        "        s = speed\n",
        "\n",
        "    if s <= 60:\n",
        "        print(\"You pass.\")\n",
        "    elif s > 61 and s <= 80:\n",
        "        print(\"You get a small ticket\")\n",
        "    else:\n",
        "        print(\"You get a big ticket.\")\n",
        "\n",
        "speeding(speed, birthday)"
      ]
    },
    {
      "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": [
        "def weeklyPaid(hours_worked, wage):\n",
        "    if hours_worked > 40:\n",
        "        return 40 * wage + (hours_worked - 40) * wage * 1.5\n",
        "    else:\n",
        "        return hours_worked * wage\n",
        " \n",
        " \n",
        "hours_worked = 50\n",
        "wage = 100\n",
        " \n",
        "pay = weeklyPaid(hours_worked, wage)\n",
        " \n",
        "print(f\"Total gross pay: Rs.{pay:.2f} \")"
      ],
      "metadata": {
        "id": "R5-CdXSKjacN",
        "colab": {
          "base_uri": "https://localhost:8080/"
        },
```

```
    "outputId": "c03583fc-e249-4959-cee8-ae16a8da1a1c"
  },
  "execution_count": 8,
  "outputs": [
    {
      "output_type": "stream",
      "name": "stdout",
      "text": [
        "Total gross pay: Rs.5500.00 \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": [
    "# Python3 code to demonstrate working of \n",
    "# Assign similar index values in Dictionary\n",
    "# Using loop + keys()\n",
    "  \n",
    "# initializing dictionaries\n",
    "test_dict1 = {\"Gfg\" : 20, \"is\" : 36, \"best\" : 100}\n",
    "test_dict2 = {\"Gfg2\" : 26, \"is2\" : 19, \"best2\" : 70}\n",
    "  \n",
    "# printing original dictionaries\n",
    "print(\"The original dictionary 1 is : \" + str(test_dict1))\n",
    "print(\"The original dictionary 2 is : \" + str(test_dict2))\n",
    "  \n",
    "# extracting keys and values\n",
    "keys1 = list(test_dict1.keys())\n",
    "vals2 = list(test_dict2.values())\n",
    "  \n",
    "# assigning new values \n",
    "res = dict()\n",
    "for idx in range(len(keys1)):\n",
    "    res[keys1[idx]] = vals2[idx]\n",
    "      \n",
    "# printing result \n",
    "print(\"Mapped dictionary : \" + str(res)) "
  ],
  "metadata": {
    "id": "8ugVoEe0kOsk",
    "colab": {
```

```
          "base_uri": "https://localhost:8080/"
        },
        "outputId": "85cc1571-f642-471f-f2d8-f19fee8488ed"
      },
      "execution_count": 7,
      "outputs": [
        {
          "output_type": "stream",
          "name": "stdout",
          "text": [
            "The original dictionary 1 is : {'Gfg': 20, 'is': 36, 'best':
100}\n",
            "The original dictionary 2 is : {'Gfg2': 26, 'is2': 19,
'best2': 70}\n",
            "Mapped dictionary : {'Gfg': 26, 'is': 19, 'best': 70}\n"
          ]
        }
      ]
    },
    {
      "cell_type": "code",
      "source": [],
      "metadata": {
        "id": "PEs1JWPVDxMz"
      },
      "execution_count": null,
      "outputs": []
    }
  ],
  "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
}
```