```
{
 "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": "a05565aa-db43-4716-e87d-41c5c8a6f95e"
 },
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "2401\n"
   ]
  }
 ],
 "source": [
  "pow=7**4\n",
  "print(pow)"
 ]
},
{
 "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": 3,
   "metadata": {
    "id": "GD_Tls3H85j7"
   },
   "outputs": [
    {
     "name": "stdout",
     "output_type": "stream",
     "text": [
      "['Hi', 'there', 'Sam!']\n"
     ]
    }
   ],
   "source": [
    "s=\"Hi there Sam!\"\n",
    "String_list=s.split(\" \")\n",
    "print (String_list)"
```

```
 ]
},
{
 "cell_type": "code",
 "execution_count": 4,
 "metadata": {
  "id": "RRGOKoai85j8",
  "outputId": "cc52f0d8-2ed1-4b4d-e956-5bbeb332cdc2"
 },
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
   "text": [
    "['Hi', 'there', 'dad!']\n"
   ]
  }
 ],
 "source": [
  "s=\"Hi there dad!\"\n",
  "String_list=s.split(\" \")\n",
  "print (String_list)"
 ]
},
{
 "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": 5,
 "metadata": {
  "id": "2TrzmDcS85j-"
 },
 "outputs": [
  {
   "name": "stdout",
   "output_type": "stream",
    "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": "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": 6,
   "metadata": {
    "id": "6m5C0sTW85kE",
    "outputId": "c3417d1c-3081-4e24-8489-154cdce1b06b"
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "['hello']"
      ]
     },
     "execution_count": 6,
     "metadata": {},
     "output_type": "execute_result"
    }
```

```
 ],
 "source": [
  "lst = [1,2,[3,4],[5,[100,200,['hello']],23,11],1,7]\n",
  "lst[3][1][2]"
 ]
},
{
 "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": 7,
 "metadata": {
  "id": "FlILSdm485kH",
  "outputId": "4232540d-95c2-461d-c78d-24ea62398e08"
 },
 "outputs": [
  {
   "data": {
    "text/plain": [
     "'hello'"
    ]
   },
   "execution_count": 7,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
  "d = {'k1':[1,2,3,{'tricky':['oh','man','inception',{'target':[1,2,3,'hello']}]}]}\n",
  "d['k1'][3].get('tricky')[3].get('target')[3]"
 ]
},
{
 "cell_type": "markdown",
 "metadata": {
  "id": "FInV_FKB85kI"
 },
 "source": [
  "** What is the main difference between a tuple and a list? **"
 ]
},
{
 "cell_type": "code",
 "execution_count": 9,
 "metadata": {
  "id": "_VBWf00q85kJ"
 },
 "outputs": [],
 "source": [
  "# Tuple is immutable whereas List is mutable. Hence Tuple is faster than list.\n",
  "\n",
  "# Tuple can be represented as paranthesis() but List is represented in square brackets []"
 ]
},
{
 "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": 10,
   "metadata": {
    "id": "unvEAwjk85kL"
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "'domain.com'"
      ]
     },
     "execution_count": 10,
     "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "def domainGet(mail):\n",
   "    return mail.split('@')[1]\n",
   "\n",
   "domainGet('user@domain.com')"
  ]
 },
 {
  "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": 11,
  "metadata": {
   "id": "Q4ldLGV785kM"
  },
  "outputs": [
   {
    "data": {
     "text/plain": [
      "True"
     ]
    },
    "execution_count": 11,
    "metadata": {},
    "output_type": "execute_result"
   }
  ],
  "source": [
   "def dogcheck(input):\n",
   "    Chk=input.lower()\n",
   "    return 'dog' in Chk.split()\n",
   "dogcheck('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": 13,
   "metadata": {
    "id": "6hdc169585kO"
   },
   "outputs": [],
   "source": [
    "def countWord(string):\n",
    "  myList = string.split()\n",
    "  occurence = 0\n",
    "  for i in myList:\n",
    "    if(i in [\"Dog\", \"dog\", \"DOG\", \"dog\", \"dogs\", \"Dogs\"]):\n",
    "      occurence = occurence+1\n",
    "  return occurence"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": 14,
   "metadata": {
    "id": "igzsvHb385kO",
    "outputId": "0602a2b5-0b18-48d8-e2d4-fe644cbccf8a"
   },
   "outputs": [
    {
     "data": {
      "text/plain": [
       "2"
      ]
     },
     "execution_count": 14,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "countWord(\"Dog are so attractive as well as dogs are lovable\")"
   ]
  },
  {
   "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": 16,
   "metadata": {
    "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": 17,
  "metadata": {
   "id": "BU_UZcyk85kS",
   "outputId": "699de8ef-a18c-436b-fdd9-60dc44979906"
  },
  "outputs": [
   {
    "data": {
     "text/plain": [
      "'Big Ticket'"
     ]
    },
    "execution_count": 17,
    "metadata": {},
    "output_type": "execute_result"
   }
 ],
 "source": [
  "caught_speeding(85,False)"
 ]
},
{
 "cell_type": "code",
 "execution_count": 18,
 "metadata": {
  "id": "p1AGJ7DM85kR",
  "outputId": "ca80629f-5949-4926-8d27-1b61576669ac"
 },
 "outputs": [
  {
   "data": {
    "text/plain": [
     "'Small Ticket'"
    ]
   },
   "execution_count": 18,
   "metadata": {},
   "output_type": "execute_result"
  }
 ],
 "source": [
   "caught_speeding(85,True)"
  ]
 },
 {
  "cell_type": "markdown",
  "metadata": {
   "id": "Tie4rC7_kAOC"
  },
  "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. "
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 19,
  "metadata": {
   "id": "R5-CdXSKjacN"
  },
  "outputs": [
   {
    "name": "stdout",
    "output_type": "stream",
    "text": [
     "employees list:\n",
     "15000\n",
      "300456\n",
      "25000\n",
      "100000\n",
      "550000\n",
      "total salary expenditure:\n",
      "990456\n"
     ]
    }
   ],
   "source": [
    "employee=[15000,300456,25000,100000,550000]\n",
    "total=0\n",
    "print(\"employees list:\")\n",
    "for i in employee:\n",
    "  print(i)\n",
    "  total=total+i\n",
    "print('total salary expenditure:')\n",
    "print(total)"
   ]
  },
  {
   "cell_type": "markdown",
   "metadata": {
    "id": "-L1aiFqRkF5s"
   },
  "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. "
  ]
 },
 {
  "cell_type": "code",
  "execution_count": 20,
  "metadata": {
   "id": "8ugVoEe0kOsk"
  },
  "outputs": [
   {
    "data": {
     "text/plain": [
      "{'Empid': 1,\n",
      " 'Empname': 'Ram',\n",
      " 'Basicpay': '11 lpa',\n",
      " 'DeptName': 'CSE',\n",
      " 'DeptId': 123}"
      ]
     },
     "execution_count": 20,
     "metadata": {},
     "output_type": "execute_result"
    }
   ],
   "source": [
    "def Merge_dict(First, Second):\n",
    "    for i in Second.keys():\n",
    "        First[i]=Second[i]\n",
    "    return First\n",
    "First={'Empid':1, 'Empname':'Ram', 'Basicpay':'11 lpa'}\n",
    "Second={'DeptName':'CSE','DeptId':123}\n",
    "dict1= Merge_dict(First,Second)\n",
    "dict1"
   ]
  },
  {
   "cell_type": "code",
   "execution_count": null,
   "metadata": {},
   "outputs": [],
   "source": []
  }
```

```
],
"metadata": {
 "colab": {
  "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": "3.9.12"
 }
},
"nbformat": 4,
"nbformat_minor": 1
```

```
}
```