ASSIGNMENT-3

**PYTHON CODING WITH GOOGLE COLABORATORY**

{

"cells": [

{

"cell\_type": "markdown",

"metadata": {

"id": "view-in-github",

"colab\_type": "text"

},

"source": [

"<a href=\"https://colab.research.google.com/github/IBM-EPBL/IBM-Project-18081-1659678980/blob/main/C\_Magimai\_Assignment\_3\_Python.ipynb\" target=\"\_parent\"><img src=\"https://colab.research.google.com/assets/colab-badge.svg\" alt=\"Open In Colab\"/></a>"

]

},

{

"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": "a05565aa-db43-4716-e87d-41c5c8a6f95e"

},

"outputs": [

{

"data": {

"text/plain": [

"2401"

]

},

"execution\_count": 1,

"metadata": {

"tags": []

},

"output\_type": "execute\_result"

}

],

"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": null,

"metadata": {

"collapsed": true,

"id": "GD\_Tls3H85j7"

},

"outputs": [],

"source": [

"s = \"Hi there Sam!\"\n"

]

},

{

"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": [

"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": null,

"metadata": {

"collapsed": true,

"id": "2TrzmDcS85j-"

},

"outputs": [],

"source": [

"planet = \"Earth\"\n",

"diameter = 12742\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "s\_dQ7\_xc85j\_",

"outputId": "4235fdfb-5591-4dd9-f9d2-77f311977633"

},

"outputs": [

{

"name": "stdout",

"output\_type": "stream",

"text": [

"The diameter of Earth is 12742 kilometers.\n"

]

}

],

"source": [

"print(\"The diameter of {} is {} kilometres.\".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": [

"\n",

"lst = [1,2,[3,4],[5,[100,200,['hello']],23,11],1,7]"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "6m5C0sTW85kE",

"outputId": "c3417d1c-3081-4e24-8489-154cdce1b06b"

},

"outputs": [

{

"data": {

"text/plain": [

"'hello'"

]

},

"execution\_count": 14,

"metadata": {

"tags": []

},

"output\_type": "execute\_result"

}

],

"source": [

"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": "4232540d-95c2-461d-c78d-24ea62398e08"

},

"outputs": [

{

"data": {

"text/plain": [

"'hello'"

]

},

"execution\_count": 16,

"metadata": {

"tags": []

},

"output\_type": "execute\_result"

}

],

"source": [

"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"

},

"outputs": [],

"source": [

"#Tuple is immutable"

]

},

{

"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": null,

"metadata": {

"collapsed": true,

"id": "unvEAwjk85kL"

},

"outputs": [],

"source": [

"def domainGet(email):\n",

" return email.split('@')[-1]"

]

},

{

"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": [

"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": null,

"metadata": {

"collapsed": true,

"id": "Q4ldLGV785kM"

},

"outputs": [],

"source": [

"def findDog(st):\n",

" return 'dog in st.lower().split()"

]

},

{

"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": [

"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": null,

"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": 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": [

"countDog('This dog runs faster than the other dog dude!')"

]

},

{

"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": null,

"metadata": {

"id": "BU\_UZcyk85kS",

"outputId": "699de8ef-a18c-436b-fdd9-60dc44979906"

},

"outputs": [

{

"data": {

"text/plain": [

"'Big Ticket'"

]

},

"execution\_count": 6,

"metadata": {

"tags": []

},

"output\_type": "execute\_result"

}

],

"source": [

"caught\_speeding(81,False)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"metadata": {

"id": "p1AGJ7DM85kR",

"outputId": "ca80629f-5949-4926-8d27-1b61576669ac"

},

"outputs": [

{

"data": {

"text/plain": [

"'Small Ticket'"

]

},

"execution\_count": 5,

"metadata": {

"tags": []

},

"output\_type": "execute\_result"

}

],

"source": [

"caught\_speedin(81,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": [

"for(i=0;i<=5;i++)\n",

"sample\_dict = {\n",

" \"name\":\"Kelly\",\n",

" \"age\":28,\n",

" \"salary\":20000,\n",

" \"city\":\"New york\"\n",

" \"emp id\":'62345' }\n",

"key=[\"name\",\"salary\"]"

],

"metadata": {

"id": "R5-CdXSKjacN"

},

"execution\_count": null,

"outputs": []

},

{

"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": [

"def Merge(dict\_1,dict\_2):\n",

" result = dict\_1| dict\_2\n",

" return result\n",

"dict\_1 ={'Empid': 76543, 'Empname': muthu, 'Basicpay': 9000}\n",

"dict\_1 ={'DeptName': computer science engineering, 'DeptName': 98761,}\n",

"dict\_3 = Merge(dict\_1,dict\_2)\n",

"print(dict\_3)\n",

"\n"

],

"metadata": {

"id": "8ugVoEe0kOsk"

},

"execution\_count": null,

"outputs": []

}

],

"metadata": {

"colab": {

"provenance": [],

"include\_colab\_link": true

},

"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

}