MAHENDRA ENGINEERING COLLEGE

NAME: Buvaneshwari.G CLASS: IV year ECE SUBJECT: IBM

REGISTER NO: 621519106014

{

"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": [ "print(7\*7\*7\*7)"

]

},

{

"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": [], "source": [

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

]

},

{

"cell\_type": "code", "execution\_count": 4, "metadata": {

"id": "RRGOKoai85j8",

"outputId": "cc52f0d8-2ed1-4b4d-e956-5bbeb332cdc2"

},

"outputs": [

{

"data": {

"text/plain": [

"['Hi', 'there', 'Sam!']"

]

},

"execution\_count": 4, "metadata": {},

"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": 5, "metadata": {

"id": "2TrzmDcS85j-"

},

"outputs": [], "source": [

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

"diameter=\"The diameter of {} is 12742 kilometers\".format(planet)\n"

]

},

{

"cell\_type": "code", "execution\_count": 6, "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(diameter)"

]

},

{

"cell\_type": "markdown", "metadata": {

"id": "QAKtN7Hh85kB"

},

"source": [

"\*\* Given this nested list, use indexing to grab the word \"hello\" \*\*"

]

},

{

"cell\_type": "code", "execution\_count": 7, "metadata": {

"id": "-7dzQDyK85kD"

},

"outputs": [], "source": [

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

]

},

{

"cell\_type": "code", "execution\_count": 8, "metadata": {

"id": "6m5C0sTW85kE",

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

},

"outputs": [

{

"data": { "text/plain": [

"['hello']"

]

},

"execution\_count": 8, "metadata": {},

"output\_type": "execute\_result"

}

],

"source": [ "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": 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": 10, "metadata": {

"id": "FlILSdm485kH",

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

},

"outputs": [

{

"data": { "text/plain": [

"'hello'"

]

},

"execution\_count": 10, "metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"\n", "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": "code", "execution\_count": null, "metadata": {},

"outputs": [], "source": [

"The main differences between lists and tuples are,Lists are enclosed in brackets([]) and their elements and size can be changed,while tuples are enclosed in parentheses() and cannot be updated.Tuples can be thought of as read only lists."

]

},

{

"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": 13, "metadata": {

"id": "unvEAwjk85kL"

},

"outputs": [

{

"name": "stdout", "output\_type": "stream", "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": "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": 14, "metadata": {

"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": 15, "metadata": {

"id": "EqH6b7yv85kN",

"outputId": "e7909af1-8df1-4534-fc8c-27b03d7369e5"

},

"outputs": [

{

"name": "stdout", "output\_type": "stream", "text": [

"True\n"

]

}

],

"source": [

"st = \"Is there a dog here?\"\n", "findDog(st)"

]

},

{

"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": 16, "metadata": {

"id": "6hdc169585kO"

},

"outputs": [],

"source": [

"value = 'This dog runs faster than the other dog dude!';"

]

},

{

"cell\_type": "code", "execution\_count": 17, "metadata": {

"id": "igzsvHb385kO",

"outputId": "0602a2b5-0b18-48d8-e2d4-fe644cbccf8a"

},

"outputs": [

{

"name": "stdout", "output\_type": "stream", "text": [

"1\n", "2\n"

]

}

],

"source": [

"def countdogs(value):\n", " count = 0\n",

" for word in value.lower().split():\n",

" if word == 'dog' or word == 'dogs':\n", " count = count + 1\n",

" print(count)\n", "\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": 18, "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": 19, "metadata": {

"id": "BU\_UZcyk85kS",

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

},

"outputs": [

{

"data": { "text/plain": [ "'Big Ticket'"

]

},

"execution\_count": 19, "metadata": {},

"output\_type": "execute\_result"

}

],

"source": [ "caught\_speeding(81,False)"

]

},

{

"cell\_type": "code", "execution\_count": 20, "metadata": {

"id": "p1AGJ7DM85kR",

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

},

"outputs": [

{

"data": { "text/plain": [ "'Small Ticket'"

]

},

"execution\_count": 20, "metadata": {},

"output\_type": "execute\_result"

}

],

"source": [

"speed=\"Your speed is more than 81\"\n", "caught\_speeding(81,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": 21, "metadata": {

"id": "R5-CdXSKjacN"

},

"outputs": [

{

"name": "stdout", "output\_type": "stream", "text": [

"Enter Gowsi's salary: 100000\n", "{'Gowsi': 100000}\n", "100000\n",

"Enter Durga's salary: 200000\n", "{'Gowsi': 100000, 'Durga': 200000}\n", "300000\n",

"Enter Suji's salary: 300000\n",

|  |  |  |
| --- | --- | --- |
| "{'Gowsi': 100000, 'Durga': 200000, | 'Suji': | 300000}\n", |
| "600000\n", |  |  |
| "Enter Kavya's salary: 400000\n", |  |  |
| "{'Gowsi': 100000, 'Durga': 200000, | 'Suji': | 300000, 'Kavya': |
| 400000}\n", |  |  |
| "1000000\n", |  |  |
| "Enter Manoj's salary: 500000\n", |  |  |
| "{'Gowsi': 100000, 'Durga': 200000, | 'Suji': | 300000, 'Kavya': 400000, |
| 'Manoj': 500000}\n", |  |  |
| "1500000\n", |  |  |
| "Enter Priyan's salary: 600000\n", |  |  |
| "{'Gowsi': 100000, 'Durga': 200000, | 'Suji': | 300000, 'Kavya': 400000, |
| 'Manoj': 500000, 'Priyan': 600000}\n", |  | |
| "2100000\n" |
| ] |
| } |
| ], |
| "source": [ |

"employee\_names = [\"Gowsi\", \"Durga\", \"Suji\", \"Kavya\", \"Manoj\",

\"Priyan\"]\n", "employee\_salaries = {}\n",

"for employee in employee\_names:\n",

" while True: # Input validation loop\n", " try:\n",

" employee\_salaries[employee] = int(input(f\"Enter

{employee}'s salary: \")) \n", " break\n",

" except ValueError:\n",

" print(\"Invalid input\")\n", " print(employee\_salaries)\n",

" total = sum(employee\_salaries.values())\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": 22, "metadata": {

"id": "8ugVoEe0kOsk"

},

"outputs": [], "source": [

"d1={\"Empid\":101,\n",

" \"Empname\":'Gowsi', \n",

" \"Basicpay\":5000}\n", "d2={\"DeptName\":'CSE', \n", " \"DeptId\":104}\n"

]

},

{

"cell\_type": "code", "execution\_count": 23, "metadata": {}, "outputs": [

{

"name": "stdout", "output\_type": "stream", "text": [

"{'Empid': 101, 'Empname': 'Gowsi', 'Basicpay': 5000, 'DeptName': 'CSE', 'DeptId': 104}\n"

]

}

],

"source": [

"print(d1|d2)"

]

}

],

"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

}