{

"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": "ffd59ff1-c876-44c1-ce28-83a041f4f518",

"colab": {

"base\_uri": "https://localhost:8080/"

}

},

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"2401\n"

]

}

],

"source": [

"print(7\*\*4)"

]

},

{

"cell\_type": "markdown",

"metadata": {

"id": "im59\_a5kfa0K"

},

"source": [

"\*\* Split this string:\*\*\n",

"\n",

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

" \n",

"\*\*into a list. \*\*"

]

},

{

"cell\_type": "code",

"execution\_count": 2,

"metadata": {

"collapsed": true,

"id": "GD\_Tls3H85j7",

"colab": {

"base\_uri": "https://localhost:8080/"

},

"outputId": "69cb07ed-b3a2-4a29-b400-2a9bdf77a671"

},

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

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

]

}

],

"source": [

"string = \"Hi there Sam!\"\n",

"print(string.split())"

]

},

{

"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": []

},

{

"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": 3,

"metadata": {

"collapsed": true,

"id": "2TrzmDcS85j-",

"colab": {

"base\_uri": "https://localhost:8080/"

},

"outputId": "18a0070d-7255-4aea-f610-6d4a34f0c9f2"

},

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

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

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

]

}

],

"source": [

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

"diameter = 12742\n",

"\n",

"print(\"The diameter of {} is {} kilometers.\".format(planet, diameter))\n",

"\n",

"#or, you can do it like that:\n",

"\n",

"print(f\"The diameter of {planet} is {diameter} kilometers.\")"

]

},

{

"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": []

},

{

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

},

"outputs": [

{

"data": {

"text/plain": [

"'hello'"

]

},

"execution\_count": 14,

"metadata": {

"tags": []

},

"output\_type": "execute\_result"

}

],

"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": 5,

"metadata": {

"id": "vrYAxSYN85kG",

"colab": {

"base\_uri": "https://localhost:8080/"

},

"outputId": "9162cca6-8661-4ecc-8b4b-0b1d74676ca3"

},

"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])"

]

},

{

"cell\_type": "markdown",

"metadata": {

"id": "FInV\_FKB85kI"

},

"source": [

"\*\* What is the main difference between a tuple and a list? \*\*"

]

},

{

"cell\_type": "code",

"execution\_count": 6,

"metadata": {

"collapsed": true,

"id": "\_VBWf00q85kJ",

"colab": {

"base\_uri": "https://localhost:8080/",

"height": 35

},

"outputId": "60429988-c3d9-4a88-e170-e35ad388d107"

},

"outputs": [

{

"output\_type": "execute\_result",

"data": {

"text/plain": [

"'\\ntuple is immutable, and list is muutable\\n'"

],

"application/vnd.google.colaboratory.intrinsic+json": {

"type": "string"

}

},

"metadata": {},

"execution\_count": 6

}

],

"source": [

"t = (1, 2, 3)\n",

"list = [1, 2, 3, 4, 5]\n",

"\n",

"'''\n",

"tuple is immutable, and list is muutable\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": 7,

"metadata": {

"collapsed": true,

"id": "unvEAwjk85kL",

"colab": {

"base\_uri": "https://localhost:8080/"

},

"outputId": "9382a89e-65b4-404c-c75c-054f198daaf6"

},

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Please enter your email: >ramaya20002@gmail.com\n",

"Your domain is: gmail.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": 8,

"metadata": {

"collapsed": true,

"id": "Q4ldLGV785kM",

"colab": {

"base\_uri": "https://localhost:8080/"

},

"outputId": "d9dd7027-07ff-4db4-f320-5433d536e5f6"

},

"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": "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": 9,

"metadata": {

"id": "6hdc169585kO",

"colab": {

"base\_uri": "https://localhost:8080/"

},

"outputId": "872d5f3c-7eed-4061-c2c5-bba3b9239296"

},

"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": "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": 10,

"metadata": {

"id": "BU\_UZcyk85kS",

"outputId": "3d7cc218-37a3-40d4-8f5e-45d19dd8a019",

"colab": {

"base\_uri": "https://localhost:8080/"

}

},

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Please enter the speed(km/h)(only number please): \n",

"\n",

"> 65\n",

"Please enter your birthday: (in DD/MM/YYYY format)\n",

"\n",

"> 29/5/2002\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": 12,

"metadata": {

"id": "p1AGJ7DM85kR",

"outputId": "826a9b0d-84ab-4473-cff7-805bacb7bcd4",

"colab": {

"base\_uri": "https://localhost:8080/"

}

},

"outputs": [

{

"output\_type": "stream",

"name": "stdout",

"text": [

"Please enter the speed(km/h)(only number please): \n",

"\n",

"> 66\n",

"Please enter your birthday: (in DD/MM/YYYY format)\n",

"\n",

"> 25/7/2006\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": "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": [],

"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": [],

"metadata": {

"id": "8ugVoEe0kOsk"

},

"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

}