{

"cells": [

{

"cell\_type": "code",

"execution\_count": 3,

"id": "92f7c1c5",

"metadata": {},

"outputs": [

{

"ename": "SyntaxError",

"evalue": "invalid syntax (1815908867.py, line 1)",

"output\_type": "error",

"traceback": [

"\u001b[1;36m Input \u001b[1;32mIn [3]\u001b[1;36m\u001b[0m\n\u001b[1;33m 1.[]\u001b[0m\n\u001b[1;37m ^\u001b[0m\n\u001b[1;31mSyntaxError\u001b[0m\u001b[1;31m:\u001b[0m invalid syntax\n"

]

}

],

"source": [

"1.[]"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "04ebbefa",

"metadata": {},

"outputs": [],

"source": [

"2. please provide the answer for this"

]

},

{

"cell\_type": "code",

"execution\_count": 5,

"id": "d5d9f3f5",

"metadata": {},

"outputs": [

{

"name": "stderr",

"output\_type": "stream",

"text": [

"<>:1: SyntaxWarning: 'float' object is not subscriptable; perhaps you missed a comma?\n",

"<>:1: SyntaxWarning: 'float' object is not subscriptable; perhaps you missed a comma?\n",

"C:\\Users\\lenovo\\AppData\\Local\\Temp\\ipykernel\_16312\\295450905.py:1: SyntaxWarning: 'float' object is not subscriptable; perhaps you missed a comma?\n",

" 3.[int(int('3'\*2)/11)]\n",

"C:\\Users\\lenovo\\AppData\\Local\\Temp\\ipykernel\_16312\\295450905.py:1: SyntaxWarning: 'float' object is not subscriptable; perhaps you missed a comma?\n",

" 3.[int(int('3'\*2)/11)]\n",

"C:\\Users\\lenovo\\AppData\\Local\\Temp\\ipykernel\_16312\\295450905.py:1: SyntaxWarning: 'float' object is not subscriptable; perhaps you missed a comma?\n",

" 3.[int(int('3'\*2)/11)]\n"

]

},

{

"ename": "TypeError",

"evalue": "'float' object is not subscriptable",

"output\_type": "error",

"traceback": [

"\u001b[1;31m---------------------------------------------------------------------------\u001b[0m",

"\u001b[1;31mTypeError\u001b[0m Traceback (most recent call last)",

"Input \u001b[1;32mIn [5]\u001b[0m, in \u001b[0;36m<cell line: 1>\u001b[1;34m()\u001b[0m\n\u001b[1;32m----> 1\u001b[0m \u001b[38;5;241;43m3.\u001b[39;49m\u001b[43m[\u001b[49m\u001b[38;5;28;43mint\u001b[39;49m\u001b[43m(\u001b[49m\u001b[38;5;28;43mint\u001b[39;49m\u001b[43m(\u001b[49m\u001b[38;5;124;43m'\u001b[39;49m\u001b[38;5;124;43m3\u001b[39;49m\u001b[38;5;124;43m'\u001b[39;49m\u001b[38;5;241;43m\*\u001b[39;49m\u001b[38;5;241;43m2\u001b[39;49m\u001b[43m)\u001b[49m\u001b[38;5;241;43m/\u001b[39;49m\u001b[38;5;241;43m11\u001b[39;49m\u001b[43m)\u001b[49m\u001b[43m]\u001b[49m\n",

"\u001b[1;31mTypeError\u001b[0m: 'float' object is not subscriptable"

]

}

],

"source": [

"3.[int(int('3'\*2)/11)]"

]

},

{

"cell\_type": "code",

"execution\_count": 6,

"id": "592ad0c4",

"metadata": {},

"outputs": [

{

"ename": "SyntaxError",

"evalue": "invalid syntax (2728238118.py, line 1)",

"output\_type": "error",

"traceback": [

"\u001b[1;36m Input \u001b[1;32mIn [6]\u001b[1;36m\u001b[0m\n\u001b[1;33m 4.spam[-1]\u001b[0m\n\u001b[1;37m ^\u001b[0m\n\u001b[1;31mSyntaxError\u001b[0m\u001b[1;31m:\u001b[0m invalid syntax\n"

]

}

],

"source": [

"4.spam[-1]"

]

},

{

"cell\_type": "code",

"execution\_count": 7,

"id": "1cde7b68",

"metadata": {},

"outputs": [

{

"ename": "SyntaxError",

"evalue": "invalid syntax (1205937565.py, line 1)",

"output\_type": "error",

"traceback": [

"\u001b[1;36m Input \u001b[1;32mIn [7]\u001b[1;36m\u001b[0m\n\u001b[1;33m 5. spam[:2]\u001b[0m\n\u001b[1;37m ^\u001b[0m\n\u001b[1;31mSyntaxError\u001b[0m\u001b[1;31m:\u001b[0m invalid syntax\n"

]

}

],

"source": [

"5. spam[:2]"

]

},

{

"cell\_type": "code",

"execution\_count": 8,

"id": "b8cd9b2b",

"metadata": {},

"outputs": [

{

"ename": "SyntaxError",

"evalue": "invalid syntax (3279133679.py, line 1)",

"output\_type": "error",

"traceback": [

"\u001b[1;36m Input \u001b[1;32mIn [8]\u001b[1;36m\u001b[0m\n\u001b[1;33m 6.bacon.index('cat')\u001b[0m\n\u001b[1;37m ^\u001b[0m\n\u001b[1;31mSyntaxError\u001b[0m\u001b[1;31m:\u001b[0m invalid syntax\n"

]

}

],

"source": [

"6.bacon.index('cat')"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "2b7c0ef7",

"metadata": {},

"outputs": [],

"source": [

"7.bacon contains the list [3.14, 'cat', 11, 'cat', True]. What does bacon. append(99) make the list value in bacon look like? bacon contains the list [3.14, 'cat', 11, 'cat', True]."

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "ae90db75",

"metadata": {},

"outputs": [],

"source": [

"8.please provide answer for this"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "33ecc336",

"metadata": {},

"outputs": [],

"source": [

"9.What are the operators for list concatenation and list replication? The operator for list concatenation is +, while the operator for replication is \*. (This is the same as for strings.)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "16bb3b86",

"metadata": {},

"outputs": [],

"source": [

"10.The only difference between append() and insert() is that insert function allows us to add a specific element at a specified index of the list unlike append() where we can add the element only at end of the list."

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "77f1cf01",

"metadata": {},

"outputs": [],

"source": [

"11.The methods are remove(), pop() and clear() . Besides the list methods, you can also use a del keyword to remove items from a list"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "60e85921",

"metadata": {},

"outputs": [],

"source": [

"12.The similarity between Lists and Strings in Python is that both are sequences. The differences between them are that firstly, Lists are mutable but Strings are immutable. Secondly, elements of a list can be of different types whereas a String only contains characters that are all of String type.\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "9683dcb0",

"metadata": {},

"outputs": [],

"source": [

"13.The key difference between the tuples and lists is that while the tuples are immutable objects the lists are mutable . This means that tuples cannot be changed while the lists can be modified"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "108ec0bd",

"metadata": {},

"outputs": [],

"source": [

"14.Also, tuples are written using parentheses, ( and ), while lists use the square brackets, [ and ]. How do you type the tuple value that has just the integer value 42 in it? (42,) (The trailing comma is mandatory)"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "81a98cf2",

"metadata": {},

"outputs": [],

"source": [

"15.please provide answer for this"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "ee5f6601",

"metadata": {},

"outputs": [],

"source": [

"16.Variables will contain references to list values rather than list values themselves. But for strings and integer values, variables simply contain the string or integer value.\n"

]

},

{

"cell\_type": "code",

"execution\_count": null,

"id": "6a4d6337",

"metadata": {},

"outputs": [],

"source": [

"17.copy() create reference to original object. If you change copied object - you change the original object. . deepcopy() creates new object and does real copying of original object to new one"

]

}

],

"metadata": {

"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": 5

}