**Objects and Data Structures Assessment Test**

**Test your knowledge.**

**Answer the following questions**

Write a brief description of all the following Object Types and Data Structures we've learned about:

Numbers: ‘Int’ data types that can be whole numbers, floating point (decimal) numbers, or other kinds like complex numbers and fractions.

Strings: ‘String’ data types that are immutable/unchangeable and composed of individual characters, including letters, numbers, and sometimes symbols

Lists: A collection of items that are ordered and mutable/changeable; a list is denoted by square brackets

Tuples: A collection of items that are ordered and unchangeable; a tuple is denoted by parenthesis

Dictionaries: Data type that contains keys associated with values (or key/value pairs). The entries in dictionaries are mutable/changeable and unordered

**Numbers**

Write an equation that uses multiplication, division, an exponent, addition, and subtraction that is equal to 100.25. **PEMDAS**

Hint: This is just to test your memory of the basic arithmetic commands, work backwards from 100.25

In [ ]: **(0.5)\*\*2 + ((10\*100)/5) - 100**

Answer these 3 questions without typing code. Then type code to check your answer. (PEMDAS)

What is the value of the expression 4 \* (6 + 5) **44**

What is the value of the expression 4 \* 6 + 5 **29**

What is the value of the expression 4 + 6 \* 5 **34**

In [ ]:

What is the *type* of the result of the expression 3 + 1.5 + 4? **Floating type**

What would you use to find a number’s square root, as well as its square?

In [ ]:

# Square root: **4\*\*(1/2) or 4\*\*(0.5) [output: 2]**

In [ ]:

# Square: **2\*\*2 [output: 4]**

**Strings**

Given the string 'hello' give an index command that returns 'e'. Enter your code in the cell below:

In [ ]:

s = 'hello'

# Print out 'e' using indexing

**print(s[1]) or s[1]**

Reverse the string 'hello' using slicing:

In [ ]:

s ='hello'

# Reverse the string using slicing

**print(s[::-1]) or s[::-1]**

Given the string hello, give two methods of producing the letter 'o' using indexing.

In [ ]:

s ='hello'

# Print out the 'o'

# Method 1: **print(s[-1]) or s[-1]**

In [ ]:

# Method 2: **print(s[4]) or s[4]**

**Lists**

Build this list [0,0,0] two separate ways.

In [ ]:

# Method 1: **list.append(0) three times or [0]\*3**

In [ ]:

# Method 2: **list0 = [0,0,0] => list.append(list0)**

*Reassign 'hello' in this nested list to say 'goodbye' instead:*

In [ ]:

list3 = [1,2,[3,4,'hello']]

**list[2][2] = 'goodbye'**

Sort the list below:

In [ ]:

list4 = [5,3,4,6,1]

**list4.sort() OR sorted(list4)**

**Dictionaries**

Using keys and indexing, grab the 'hello' from the following dictionaries:

In [ ]:

d = {'simple\_key':'hello'}

# Grab 'hello'

**d.get(‘simple\_key’) OR d[‘simple key’]**

In [ ]:

d = {'k1':{'k2':'hello'}}

# Grab 'hello'

**d['k1']['k2']**

In [ ]:

# Getting a little tricker

d = {'k1':[{'nest\_key':['this is deep',['hello']]}]}

**d['k1'][0]['nest\_key'][1][0]**

#Grab hello

In [ ]:

# This will be hard and annoying!

d = {'k1':[1,2,{'k2':['this is tricky',{'tough':[1,2,['hello']]}]}]}

**d['k1'][2]['k2'][1]['tough'][2][0]**

**Work:**

***d['k1'] -> [1,2,{'k2':['this is tricky',{'tough':[1,2,['hello']]}]}]***

***d[2] -> {'k2':['this is tricky',{'tough':[1,2,['hello']]}]}***

***d['k2'] -> ['this is tricky',{'tough':[1,2,['hello']]}]***

***d[1] -> {'tough':[1,2,['hello']]}***

***d['tough'] -> [1,2,['hello']]***

***d[2] -> ['hello']***

***d[0] -> 'hello'***

Can you sort a dictionary? Why or why not? **No, because a dictionary is an unordered collection of elements which are *mappings*, not sequences.**

**Tuples**

What is the major difference between tuples and lists? **Tuples are immutable, their elements can’t be changed, while lists are mutable.**

How do you create a tuple? **With parenthesis; example: tuple1 = (0, 1, ‘hello’)**

**Sets**

What is unique about a set? **A set is a collection of unordered and unique elements that are mutable. A set is also iterable.**

Use a set to find the unique values of the list below:

In [ ]:

list5 = [1,2,2,33,4,4,11,22,3,3,2]

**set(list5)**

**Output: {1, 2, 3, 4, 11, 22, 33}**

**Booleans**

For the following quiz questions, we will get a preview of comparison operators. In the table below, a=3 and b=4.

|  |  |  |
| --- | --- | --- |
| **Operator** | **Description** | **Example** |
| == | If the values of two operands are equal, then the condition becomes true. | (a == b) is not true. |
| != | If values of two operands are not equal, then condition becomes true. | (a != b) is true. |
| > | If the value of left operand is greater than the value of right operand, then condition becomes true. | (a > b) is not true. |
| < | If the value of left operand is less than the value of right operand, then condition becomes true. | (a < b) is true. |
| >= | If the value of left operand is greater than or equal to the value of right operand, then condition becomes true. | (a >= b) is not true. |
| <= | If the value of left operand is less than or equal to the value of right operand, then condition becomes true. | (a <= b) is true. |

What will be the resulting Boolean of the following pieces of code (answer fist then check by typing it in!)

In [ ]:

# Answer before running cell

2 > 3 **False**

In [ ]:

# Answer before running cell

3 <= 2 **False**

In [ ]:

# Answer before running cell

3 == 2.0 **False**

In [ ]:

# Answer before running cell

3.0 == 3 **True**

In [ ]:

# Answer before running cell

4\*\*0.5 != 2 **False**

Final Question: What is the boolean output of the cell block below?

In [ ]:

# two nested lists

l\_one = [1,2,[3,4]]

l\_two = [1,2,{'k1':4}]

# True or False?

l\_one[2][0] >= l\_two[2]['k1'] **=> Is 3 >= 4? False**

**Great Job on your first assessment!**[**¶**](https://render.githubusercontent.com/view/ipynb?commit=c2407e5f71f48d23e029eb66d0beb366bcb0e3c9&enc_url=68747470733a2f2f7261772e67697468756275736572636f6e74656e742e636f6d2f5069657269616e2d446174612f436f6d706c6574652d507974686f6e2d332d426f6f7463616d702f633234303765356637316634386432336530323965623636643062656233363662636230653363392f30302d507974686f6e2532304f626a656374253230616e64253230446174612532305374727563747572652532304261736963732f30392d4f626a65637473253230616e6425323044617461253230537472756374757265732532304173736573736d656e74253230546573742e6970796e62&nwo=Pierian-Data%2FComplete-Python-3-Bootcamp&path=00-Python+Object+and+Data+Structure+Basics%2F09-Objects+and+Data+Structures+Assessment+Test.ipynb&repository_id=121291758&repository_type=Repository#Great-Job-on-your-first-assessment!)