

 43b8ab8f1c ▾



[Complete-Python-3-Bootcamp](#) / [00-Python Object and Data Structure Basics](#) / [10-Objects and Data Structures Assessment Test-Solution.ipynb](#)



Pierian-Data PYTHON 3 UPDATES



 1 contributor

951 lines (951 sloc) | 17.4 KB



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:

For the full answers, review the Jupyter notebook introductions of each topic!

[Numbers](#)

[Strings](#)

[Lists](#)

[Tuples](#)

[Dictionaries](#)

Numbers

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

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

```
In [1]: # Your answer is probably different
(60 + (10 ** 2) / 4 * 7) - 134.75
```

```
Out[1]: 100.25
```

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

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

What is the value of the expression `4 * 6 + 5`

What is the value of the expression `4 + 6 * 5`

```
In [2]: 4 * (6 + 5)
```

```
Out[2]: 44
```

Out[2]: 17

In [3]: `4 * 6 + 5`

Out[3]: 29

In [4]: `4 + 6 * 5`

Out[4]: 34

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

Answer: Floating Point Number

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

In [5]: `# Square root:
100 ** 0.5`

Out[5]: 10.0

In [6]: `# Square:
10 ** 2`

Out[6]: 100

Strings

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

In [7]: `s = 'hello'
Print out 'e' using indexing

s[1]`

Out[7]: 'e'

Reverse the string 'hello' using slicing:

In [8]: `s = 'hello'
Reverse the string using slicing

s[::-1]`

Out[8]: 'olleh'

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

In [9]: `s = 'hello'`

```
s = "hello"
# Print out the 'o'

# Method 1:

s[-1]
```

Out[9]: 'o'

```
In [10]: # Method 2:

s[4]
```

Out[10]: 'o'

Lists

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

```
In [11]: # Method 1:

[0]*3
```

Out[11]: [0, 0, 0]

```
In [12]: # Method 2:
list2 = [0,0,0]
list2
```

Out[12]: [0, 0, 0]

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

```
In [13]: list3 = [1,2,[3,4,'hello']]
```

```
In [14]: list3[2][2] = 'goodbye'
```

```
In [15]: list3
```

Out[15]: [1, 2, [3, 4, 'goodbye']]

Sort the list below:

```
In [16]: list4 = [5,3,4,6,1]
```

```
In [17]: # Method 1:
sorted(list4)
```

Out[17]: [1, 3, 4, 5, 6]

```
In [18]: # Method 2:
```

```
# Method 2.
list4.sort()
list4
```

Out[18]: [1, 3, 4, 5, 6]

Dictionaries

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

```
In [19]: d = {'simple_key': 'hello'}
          # Grab 'hello'

          d['simple_key']
```

Out[19]: 'hello'

```
In [20]: d = {'k1': {'k2': 'hello'}}
          # Grab 'hello'

          d['k1']['k2']
```

Out[20]: 'hello'

```
In [21]: # Getting a little trickier
          d = {'k1': [{'nest_key': ['this is deep', ['hello']]]}]
```

```
In [22]: # This was harder than I expected...
          d['k1'][0]['nest_key'][1][0]
```

Out[22]: 'hello'

```
In [23]: # This will be hard and annoying!
          d = {'k1': [1,2,{'k2': ['this is tricky',{'tough':[1,2,['hello']]}]}]}
```

```
In [24]: # Phew!
          d['k1'][2]['k2'][1]['tough'][2][0]
```

Out[24]: 'hello'

Can you sort a dictionary? Why or why not?

Answer: No! Because normal dictionaries are *mappings* not a sequence.

Tuples

What is the major difference between tuples and lists?

Tuples are immutable!

How do you create a tuple?

How do you create a tuple?

```
In [25]: t = (1,2,3)
```

Sets

What is unique about a set?

Answer: They don't allow for duplicate items!

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

```
In [26]: list5 = [1,2,2,33,4,4,11,22,3,3,2]
```

```
In [27]: set(list5)
```

```
Out[27]: {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 first then check by typing it in!)

```
In [28]: # Answer before running cell
         2 > 3
```

```
Out[28]: False
```

In [29]: `# Answer before running cell`
`3 <= 2`

Out[29]: False

In [30]: `# Answer before running cell`
`3 == 2.0`

Out[30]: False

In [31]: `# Answer before running cell`
`3.0 == 3`

Out[31]: True

In [32]: `# Answer before running cell`
`4**0.5 != 2`