# 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:

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]:
```
(100.5-0.25)*(50/50)
```
Out[1]:
```
100.25
```

```
Explain what the cell below will produce and why. Can you change it so the
answer is correct?
```

In [3]:
```
2/3
```
Out[3]:
```
0
```

In [2]:
```
2.0/3
```
Out[2]:
```
0.6666666666666666
```

In [3]:
```
from __future__ import division
```

```
In [4]: 2/3
```

```
Out[4]: 0.6666666666666666
```

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

```
Out[6]: 44
```

```
In [7]: 4 * 6 + 5
```

```
Out[7]: 29
```

```
In [8]: 4 + 6 * 5
```

```
Out[8]: 34
```

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

Type *Markdown* and LaTeX: $\alpha^2$

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

```
In [11]: (56**0.5)**2
```

```
Out[11]: 56.0
```

```
In [12]: 56**(0.5**2)
```

```
Out[12]: 2.735564799734761
```

## Strings

Given the string 'hello' give an index command that returns 'e'. Use the code below:

In [14]:

```
s = 'hello'
# Print out 'e' using indexing

s[1]

# Code here

print s[1]
```

```
e
```

Reverse the string 'hello' using indexing:

In [15]:

```
s ='hello'

# Reverse the string using indexing

# Code here

s[::-1]
```

Out[15]:

```
'olleh'
```

In [15]:

```
s='hello'

print s[::-1]
print s[::-2]
print s[::2]
```

```
olleh
olh
hlo
```

In [18]:

```
s= 'hello'

print s[4]
print s[-1]
```

```
o
o
```

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

In [16]:

```
s ='hello'

# Print out the

# Code here
s[4]
s[-1]
```

Out[16]:

```
'o'
```

## Lists

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

```
In [17]: s= [0,0,0]
```

```
In [19]: s = [0,0]
         s

Out[19]: [0, 0]
```

```
In [21]: s = s+ [0]
         s

Out[21]: [0, 0, 0, 0]
```

```
In [20]: s = [0]*3
         s

Out[20]: [0, 0, 0]
```

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

```
In [22]: l = [1,2,[3,4,'hello']]

         l[2][2]='goodbye'
         l

Out[22]: [1, 2, [3, 4, 'goodbye']]
```

Sort the list below:

```
In [24]: l = [3,4,5,5,6]
         l.sort()
         l

Out[24]: [3, 4, 5, 5, 6]
```

## Dictionaries

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

```
In [28]: d = {'simple_key':'hello'}
         # Grab 'hello'
         d['simple_key']

Out[28]: 'hello'
```

In [29]:

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

Out[29]:

```
'hello'
```

In [33]:

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

#Grab hello
d['k1'][0]['nest_key'][1][0]
```

Out[33]:

```
'hello'
```

In [23]:

```
d = {'k1':[{'nest_key':['this is deep',['hello']]}]}

print d['k1'][0]['nest_key'][1][0]
```

```
hello
```

In [39]:

```
# 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]
```

Out[39]:

```
'hello'
```

In [24]:

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

print d['k1'][2]['k2'][1]['tough'][2][0]
```

```
hello
```

In [40]:

```
Markdown and LaTex:
```

```
  File "<ipython-input-40-ee01c433bdb5>", line 1
    Markdown and LaTex:
                     ^
SyntaxError: invalid syntax
```

In [ ]:

```


Can you sort a dictionary? Why or why not?  #no, dictionary is a hash table and n
```

Type *Markdown* and LaTeX: $\alpha^2$

## Tuples

What is the major difference between tuples and lists?

Type *Markdown* and LaTeX: $\alpha^2$

How do you create a tuple?

Type *Markdown* and LaTeX: $\alpha^2$

## Sets

What is unique about a set?

Type *Markdown* and LaTeX: $\alpha^2$

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

```
In [41]: l = [1,2,2,33,4,4,11,22,3,3,2]
         set(l)

Out[41]: {1, 2, 3, 4, 11, 22, 33}
```

## Booleans

For the following quiz questions, we will get a preview of comparison operators:

| 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. | |
| <> | If values of two operands are not equal, then condition becomes true. | (a <> b) is true. This is similar to != operator. |
| > | 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 [42]:
```
# Answer before running cell
2 > 3
```

Out[42]:
```
False
```

In [43]:
```
# Answer before running cell
3 <= 2
```

Out[43]:
```
False
```

In [44]:
```
# Answer before running cell
3 == 2.0
```

Out[44]:
```
False
```

In [25]:
```
# Answer before running cell
3.0 == 3
```

Out[25]:
```
True
```

In [46]:
```
# Answer before running cell
4**0.5 != 2
```

Out[46]:
```
False
```

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

In [47]:
```
# 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']

#3 >=4 ?
```

Out[47]:
```
False
```

## Great Job on your first assessment!