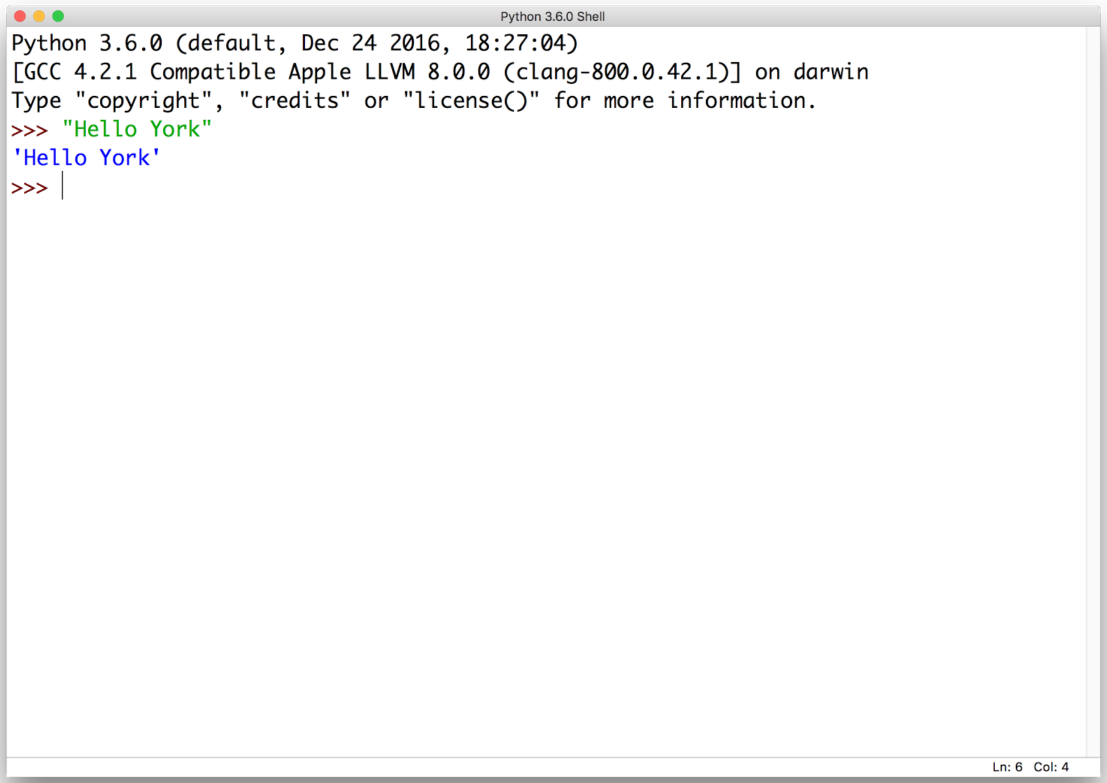
#CodeYork

Solutions 1A

Summer 2017

Task 1 - Hello York

This task was designed to make sure everybody can download the exercises off the website, and know how to use IDLE.



```
Python 3.6.0 (default, Dec 24 2016, 18:27:04)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.42.1)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> "Hello York"
'Hello York'
>>> 
```

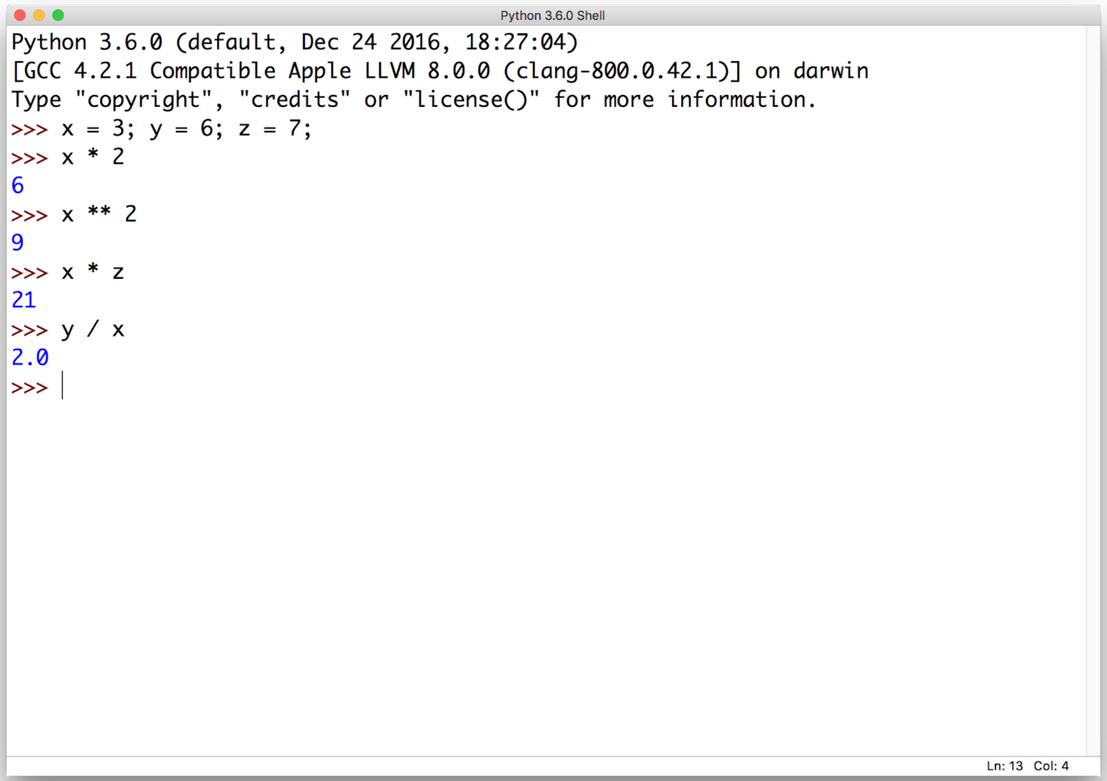
Task 2 - Results May Vary

First we have `x * 2` which evaluates to x * 2, where x = 3, so we have 3 * 2 = 6.

Next we have we have `x ** 2` which evaluates to x^2, where x = 3, so we have 3^2 = 9.

Now we have `x * z` which evaluates to x * y, where x = 3 and z = 7, so we have 3 * 7 = 21.

Finally we have `y / x` which evaluates to y / x using floating point division. Since y = 6, x = 3, we have 6 / 3 = 2.0. If we had wanted integer division, we could have instead used a double slash, and got back the integer 2.

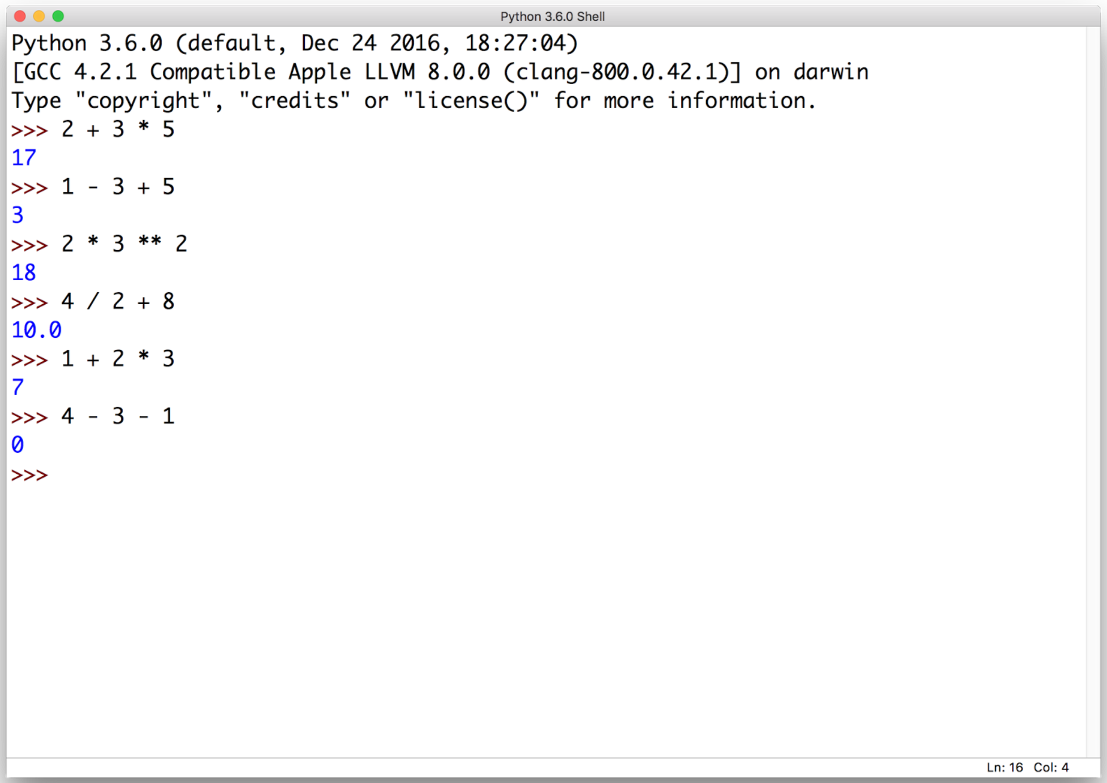


```
Python 3.6.0 (default, Dec 24 2016, 18:27:04)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.42.1)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> x = 3; y = 6; z = 7;
>>> x * 2
6
>>> x ** 2
9
>>> x * z
21
>>> y / x
2.0
>>> 
```

Task 3 - Order of Operations

Python gives addition and subtraction the same precedence, unlike in BODMAS, so we rely on the fact that it evaluates them using left associativity to correctly determine the result. This means that 1 - 3 + 5 is evaluated as (1 - 3) + 5, rather than the BODMAS convention.

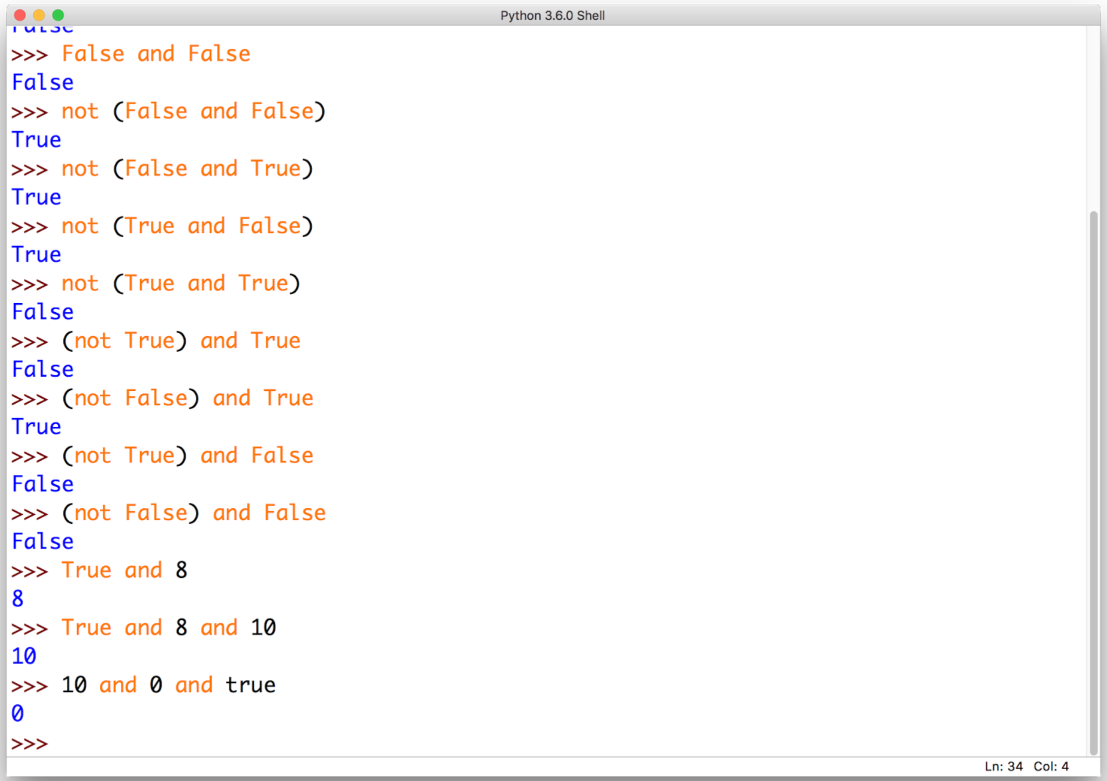
1. 2 + 3 * 5 = 2 + (3 * 5) = 2 + 15 = 17
2. 1 - 3 + 5 = (1 - 3) + 5 = -2 + 5 = 3
3. 2 * 3 ** 2 = 2 * (3 ** 2) = 2 * 9 = 18
4. 4 / 2 + 8 = 2.0 + 8 = 10.0
5. 1 + 2 * 3 = 1 + (2 * 3) = 1 + 6 = 7
6. 4 - 3 - 1 = (4 - 3) - 1 = 1 - 1 = 0



```
Python 3.6.0 (default, Dec 24 2016, 18:27:04)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.42.1)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> 2 + 3 * 5
17
>>> 1 - 3 + 5
3
>>> 2 * 3 ** 2
18
>>> 4 / 2 + 8
10.0
>>> 1 + 2 * 3
7
>>> 4 - 3 - 1
0
>>>
```

Task 4 - Boolean Expressions

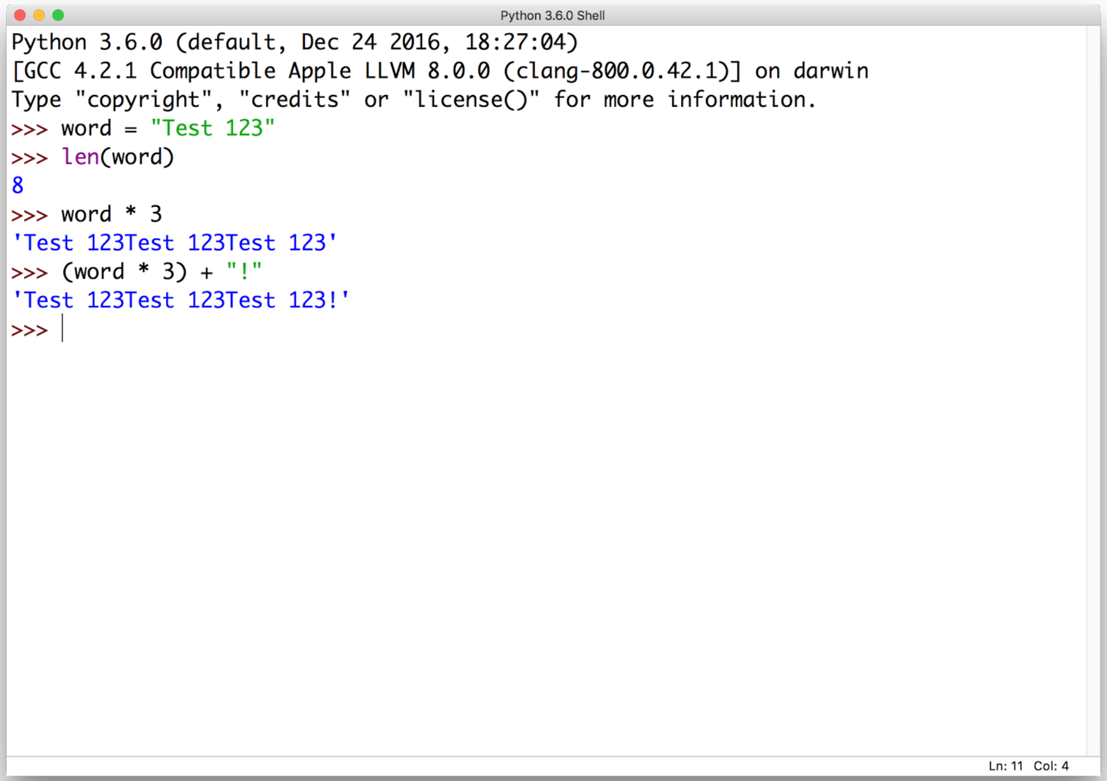
Python evaluates these expressions using short circuiting. It also doesn't technically evaluate them to a boolean value as such. In the case of conjunction (ands), this either returns the first falsey term it finds, or the last truthy term it finds. The behaviour is similarly reversed for disjunction (ors).



```
>>> False and False
False
>>> not (False and False)
True
>>> not (False and True)
True
>>> not (True and False)
True
>>> not (True and True)
False
>>> (not True) and True
False
>>> (not False) and True
True
>>> (not True) and False
False
>>> (not False) and False
False
>>> True and 8
8
>>> True and 8 and 10
10
>>> 10 and 0 and true
0
>>>
```

Task 5 - Strings

This final task checks assignment and basic string operations are understood.



```
Python 3.6.0 (default, Dec 24 2016, 18:27:04)
[GCC 4.2.1 Compatible Apple LLVM 8.0.0 (clang-800.0.42.1)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> word = "Test 123"
>>> len(word)
8
>>> word * 3
'Test 123Test 123Test 123'
>>> (word * 3) + "!"
'Test 123Test 123Test 123!'
>>> 
```