example5

May 7, 2018

1 Some common python 3 features that may be unfamiliar

Below are a few common features introduced in python 3 that may be unfamiliar (note they all work in the latest versions of python 3)

1.1 The print function

This is a very basic change. Python 3 will no longer accept the print statement without parantheses.

For example:

```
In [1]: # python 2
        print 'string'
```

```
string
```

```
In [2]: # python 3
```

```
In [3]: print('string')
```

```
string
```

1.2 Integer division

Integer division is now not the default (even for integers) i.e. x/y will always give a float even if x and y are intergers.

```
In [4]: # python 2
        x = 13
        y = 5
        print x/y
```

```
2
```

```
In [6]: # this is needed in python 2 to make it the same as python 3
        from __future__ import division

        # python 3
        x = 13
        y = 5
        print(x/y)
```

```
2.6
```

```
In [7]: # integer division is done as follows:
        x = 13
        y = 5
        print(x//y)
```

```
2
```

1.3 String formatting

This is by far the most useful change. In python 2 pushing ints, floats, strings into another string was done using the % operator. This has now been replaced with the use of {} and a .format method.

i.e.

```
In [11]: # python 2
         x = 3.14159265359
         y = 100
         kind = 'multiplied'
         symbol = '*'

         statement = 'x %s y = %d %s %d = %d' % (kind, x, symbol, y, x*y)

         print(statement)
```

```
x multiplied y = 3 * 100 = 314
```

```
In [18]: # python 3
         x = 3.14159265359
         y = 100
         kind = 'multiplied'
         symbol = '*'

         statement = 'x {} y = {} {} {} = {}'.format(kind, x, symbol, y, x*y)

         print(statement)
```

```
x multiplied y = 3.14159265359 * 100 = 314.159265359
```

Note that number formatting is similar:

```
In [15]: # python 2
         statement = 'x %s y = %.2f %s %.2f = %.2f' % (kind, x, symbol, y, x*y)

         print(statement)
```

```
x multiplied y = 3.14 * 100.00 = 314.16
```

```
In [19]: # python 3
         statement = 'x {} y = {:.2f} {} {:.2f} = {:.2f}'.format(kind, x, symbol, y, x*y)

         print(statement)
```

```
x multiplied y = 3.14 * 100.00 = 314.16
```

However unlike in python 2, we can order inputs, repeat inputs, and even access elements from dictionarys:

```
In [37]: # python 3 string formatting with list
         mylist = [x, y, x*y, x*x, kind, symbol]
         statement = 'List: x {4} y = {0:.2f} {5} {1:.2f} = {2:.2f}'.format(*mylist)
         print(statement)
```

```
List: x multiplied y = 3.14 * 100.00 = 314.16
```

```
In [36]: # python 3 string formatting with dictionary
         p = dict()
         p['x'] = 3.14159265359
         p['y'] = 100
         p['xy'] = x * y
         p['xx'] = x * x
         p['mult'] = 'multiplied'
         p['m_s'] = '*'

         statement = 'Dict: x {mult} y = {x:.2f} {m_s} {y:.2f} = {xy:.2f}'.format(**p)
         print(statement)
```

```
Dict: x multiplied y = 3.14 * 100.00 = 314.16
```

```
In [35]: # python 3 string formatting with repetition
         statement1 = 'List: x {5} x = {0:.2f} {5} {0:.2f} = {3:.2f}'.format(*mylist)
         print(statement1)

         statement2 = 'Dict: x {m_s} x = {x:.2f} {m_s} {x:.2f} = {xx:.2f}'.format(**p)
         print(statement2)
```

```
List: x * x = 3.14 * 3.14 = 9.87
Dict: x * x = 3.14 * 3.14 = 9.87
```