Lab 1

Today you will gain some hands-on experience in programming by working with Python. At your machine, run Python IDLE by going to Start→All Programs→Python 2.7→IDLE (Python GUI). From this interface, you can evaluate expressions.

If you are working on your own laptop, you should download Python 2.7 at `http://www.python.org`. Also, the lecture notes and labs are on the website `http://www.usvicoder.com`

Exercise 1: Imagine evaluating the expressions below in the order given. What would they evaluate to (some of them will give errors)? Come up with an answer for each expression before typing it into IDLE and see if you get it right.

```
>>> 5 + 4*3
```

17

```
>>> 5+4 * 3
```

17

```
>>> (5 + 4)*3
```

27

```
>>> (5 + 4.0)*3
```

27.0

```
>>> (5 + 4.0)/3
```

3.0

```
>>> 5 + 4.0/3
```

6.333333333333333

```
>>> 5 + 4.0//3
```

6.0

```
>>> 5+‘4’ * 3
```

Error, trying to add `int` 5 to `str` ’444’

```
>>> ‘5’+‘4’ * 3
```

‘5444’

```
>>> (‘5’+‘4’) * 3
```

‘545454’

```
>>> 2**3*3
```

24

```
>>> (2**3)*3
```

24

```
>>> 2**(3*3)
```

512

```
>>> [5]+'4' * 3
```

Error, trying to add `list` [5] to `str` '444'

```
>>> [5]+[4] * 3
```

[5,4,4,4]

```
>>> ['a',5,2.0][2]
```

2.0 (remember, indexing starts at 0)

```
>>> x = ['a',5,2.0]
>>> x[0]
```

'a'

```
>>> x[1]
```

5

```
>>> x[1:2]
```

[5]

```
>>> x[0:3]
```

['a',5,2.0]

```
>>> x[0]*x[1]
```

'aaaaa'

```
>>> x[0]*x[2]
```

Error, trying to multiply `str` 'a' by `float` 2.0 (can only multiply `str` by `int` or `long`)

```
>>> x[1]+x[2]
```

7.0

```
>>> x[1]**x[2]
```

25.0

```
>>> x += x[1]*x
>>> x
```

['a', 5, 2.0, 'a', 5, 2.0, 'a', 5, 2.0, 'a', 5, 2.0, 'a', 5, 2.0, 'a', 5, 2.0]

```
>>> y = x[2:4]
>>> y
```

[2.0, 'a']

```
>>> x = x[0:2]
>>> x + y
```

['a', 5, 2.0, 'a']

```
>>> x = '123456789'
>>> x + y
```

Error, trying to add `str` '123456789' to `list` [2.0, 'a']

```
>>> x[2:5]
```

'345'

```
>>> x[2:5]*x[2]
```

Error, trying to multiply `str` '345' by `str` '3'

```
>>> y = [3,2,1]
>>> x[2:5] + x[y[2]]
```

'3452'

```
>>> y = [1,2,3,4,5,6]
>>> y
```

[1,2,3,4,5,6]

```
>>> y[0:4] = [7]
>>> y
```

[7,5,6]

```
>>> not True or True
```

True

```
>>> not (True or True)
```

False

```
>>> x = 10
>>> x>20 or x%5=1
```

Error, trying to assign value to non-variable x%5

```
>>> x>20 or x%5==1
```

False

```
>>> x<20 or x%5==0
```

True

Exercise 2: Write a function `inchesToCentimeters(x)` which takes as input a numeric value x corresponding to a number of inches, then outputs a float which is the corresponding number of centimeters. Note: 1 inch is 2.54 centimeters.

For example: evaluating `inchesToCentimeters(4)` should return 10.16. What do you think `inchesToCentimeters(1/2.54)` should return? Try it out (due to the way in which Python stores `float` values, you may not get exactly what you expect, but it will be close).

Example solution:

```
def inchesToCentimeters(x):
    return x*2.54
```

Exercise 3: Write a function `doubleIt(x)` which takes a `list` or `string` x and returns x concatenated with itself.

Example solution:

```
def doubleIt(x):
    return 2*x
```

or

```
def doubleIt(x):
    return x+x
```

Exercise 4: Write a function `timeFromSeconds(x)` which takes an `int` x representing some number of seconds from the start of the day, then returns the time in hours:minute:seconds format, using "military time" (so 1pm is 13:0:0, and 1am is 1:0:0). For example, `timeFromSeconds(0)` should return `'0:0:0'`. `timeFromSeconds(60)` should return `'0:1:0'`. `timeFromSeconds(60*60)` should return `'1:0:0'`. `timeFromSeconds(24*60*60 - 1)` should return `'23:59:59'`. You might want to use the fact that given an `int` variable `x`, `str(x)` outputs a `string` version of `x`.

Example solution:

```
def timeFromSeconds(x):
    seconds = x%60
    minutes = x/60%60
    hour = x/60/60
    return str(hour) + ':' + str(minutes) + ':' + str(seconds)
```