Timing

Help Center

Often in Computer Science we are interesting in finding the fastest way to compute a solution. However, "fastest" can have relative meaning. Do we mean theoretically fastest? Or perhaps "wall clock" fast (as measured by a wristwatch). Or perhaps processor fast (consuming the fewest processor resources...though this is measured differently for different operating systems). Python allows us to measure the latter using the `time` function in the `time` module. `time` returns a floating point number representing the number of seconds of processor time used so far by your program. What this means varies from OS to OS, but you can think of it in general terms as the wall clock time of your program minus the time the executing computer spent on other processes (e.g. other programs running at the same time as your program). Unfortunately Windows machines don't report this level of detail to Python and as such on Windows this function just returns wall clock time.

We can use this function to evaluate execution times of different implementations of an algorithm. Consider the two implementations below for finding the sum of the numbers *1...n*:

Sum using range

```
import time

def sum_range(x):
    sum = 0
    for i in range(x+1):
        sum = sum + i
    return sum

def time_sum_range(x):
    start = time.time()
    result = sum_range(x)
    stop = time.time()
    elapsed = stop - start
    return elapsed
```

Sum using xrange

```
import time

def sum_xrange(x):
    sum = 0
    for i in xrange(x+1):
        sum = sum + i
    return sum

def time_sum_xrange(x):
    start = time.time()
    result = sum_xrange(x)
    stop = time.time()
    elapsed = stop - start
    return elapsed
```

We can now write a function `time_trial` to call both `sum_range` and `sum_xrange` to discover which is empirically faster for a specific test case:

```
def time_trial(sum_to, num_trials):

    for i in xrange(num_trials):
        range_result = time_sum_range(sum_to)
        xrange_result = time_sum_xrange(sum_to)
        delta = range_result - xrange_result
        print 'range =  ' + str(range_result)
```

```
        print 'xrange = ' + str(xrange_result)
        print 'xrange faster by: ' + str(delta) + ' seconds.'
```

Here is the output when this program is run in IDLE on a HP EliteBook 8560w (Intel Core i7-2820QM @ 2.30 GHz, 8GB RAM, Win7 Enterprise 64-bit):

```
>>> time_trial(100000000, 5)
range =  14.1627201429
xrange = 12.6366764776
xrange faster by: 1.52604366536 seconds.
range =  13.9327558941
xrange = 12.5941677183
xrange faster by: 1.33858817587 seconds.
range =  13.9334787715
xrange = 12.5433699651
xrange faster by: 1.39010880643 seconds.
range =  13.9190993126
xrange = 12.5750873254
xrange faster by: 1.34401198727 seconds.
range =  13.8270921945
xrange = 12.6479877236
xrange faster by: 1.17910447095 seconds.
```

Created Sun 24 Aug 2014 12:41 AM BST

Last Modified Sun 24 Aug 2014 6:01 AM BST