Why we need efficient code and how to measure it

OPTIMIZING PYTHON CODE WITH PANDAS

Leonidas Souliotis
PhD Researcher

The poker dataset

	S1	R1	S2	R2	S3	R3	S4	R4	S5	R5
1	1	10	3	11	3	13	4	4	2	1
2	2	11	2	13	2	10	2	12	2	1
3	3	12	3	11	3	13	3	10	3	1

1. Hearts
2. Diamonds
3. Clubs
4. Spades

How do we measure time?

```
import time

start_time = time.time()
result = 5 + 2
print("Results from the first method calculated in %s
seconds" % (time.time() - start_time))
```

```
Results from the first method calculated
in 9.48905944824e-05 seconds
```

The time.time() function

```
start_time = time.time()
np.sum(poker['R2'])
print("Results from the first method calculated in %s \
seconds" % (time.time() - start_time))
```

```
Results from the first method calculated in 0.000539915466309 seconds
```

```
start_time = time.time()
poker['R2'].sum()
print("Results from the second method calculated in %s \
seconds" % (time.time() - start_time))
```

```
Results from the second method calculated in 0.000655038452148 seconds
```

```
Difference in speed: 29.1814946619%
```

Where time matters I

```python
def brute_force():
    res = 0
    for i in range(1,1000001):
        res+=i
    return res
```

```python
def formula():
    return 1000000*1000001/2
```

Where time matters II

```
start_time = time.time()
first_method = formula()
print("Results from the first method calculated in %s
seconds" %(time.time() - start_time))
```

```
Results from the first method calculated  in 0.000108957290649 seconds
```

```
start_time = time.time()
second_method = brute_force()
print("Results from the second method calculated in %s
seconds" %(time.time() - start_time))
```

```
Results from the second method calculated in 0.174870967865 seconds
```

```
Difference in speed: 160,394.967179%
```

Let's do it!

OPTIMIZING PYTHON CODE WITH PANDAS

Locate rows using the .iloc() and .loc() functions

OPTIMIZING PYTHON CODE WITH PANDAS

Leonidas Souliotis
PhD Candidate

Locate targeted rows

```python
rows = range(0, 500)

start_time = time.time()
data.loc[rows]
print("Results from the first method calculated in %s seconds" % (time.time() - start_time))
```

```
Results from the first method calculated in 0.001951932 seconds
```

```python
start_time = time.time()
data.iloc[rows]
print("Results from the first method calculated in %s seconds" % (time.time() - start_time))
```

```
Results from the second method calculated in 0.0007140636 seconds
```

```
Difference in speed: 173.355592654%
```

Locate targeted columns

```
start_time = time.time()
data.iloc[:,:3]
print("Results from the first method calculated in %s seconds" % (time.time() - start_time))
```

```
Results from the first method calculated in 0.00125193595886 seconds
```

```
start_time = time.time()
data[['S1', 'R1', 'S2']]
print("Results from the first method calculated in %s seconds" % (time.time() - start_time))
```

```
Results from the first method calculated in 0.000964879989624 seconds
```

```
Difference in speed: 29.7504324188%
```

Let's do it!

OPTIMIZING PYTHON CODE WITH PANDAS

Select random rows using .random()

OPTIMIZING PYTHON CODE WITH PANDAS

Leonidas Souliotis
PhD Candiadate

Sampling random rows

```python
start_time = time.time()
poker.sample(100, axis=0)
print("Results from the second method calculated in %s seconds" % (time.time() - start_time))
```

```
Results from the first method calculated in 0.000750064849854 seconds
```

Sampling random rows using .sample()

```
start_time = time.time()
poker.iloc[np.random.randint(low=0, high=poker.shape[0], size=100)]
print("Results from the second method calculated in %s
seconds" % (time.time() - start_time))
```

```
Results from the second method calculated in 0.00103211402893 seconds
```

```
Difference in speed: 37.6033057849%
```

Sampling random columns

```
start_time = time.time()
poker.sample(3, axis=1)
print("Results from the second method calculated in %s seconds" %(time.time() - start_time))
```

```
Results from the second method calculated in 0.000683069229126 seconds
```

```
N = poker.shape[1]
start_time = time.time()
poker.iloc[:,np.random.randint(low=0, high=N, size=3)]
print("Results from the first method calculated in %s seconds" %(time.time() - start_time))
```

```
Results from the first method calculated in 0.0010929107666 seconds
```

```
Difference in speed: 59.9999999998%
```

Let's do it!

OPTIMIZING PYTHON CODE WITH PANDAS