Quintor

Performant Python Programming

Van snel Python code naar snelle Python code

G.J. de Jong

02-07-2020

Inhoudsopgave

Python in de praktijk

Performance meten

- timeit
- sys.getsizeof

Performant programmeren

- Types
- Data structuren
- Vermijd libraries
- Vermijd loops
- Laziness

In het kort

Python in de praktijk

- Data science buiten de IT
- Veel gebruik van libraries

> *Professionals working with data science applications don't want to be bogged down with complicated programming requirements. They want to use programming languages like Python and Ruby to perform tasks hassle-free.*

Performance meten

- Tijd
- Geheugengebruik

timeit

- Tool om snelheid te meten
- In Python Standard Library

```
>>> numbers_list = list(range(9999))
>>> timeit.timeit(lambda: 5000 in numbers_list,
    number=1000)
0.06302383900037967
```

```
[denni@Arch ~]$ python −m timeit −s ”numbers_list
    =list(range(9999))” ”5000␣in␣numbers_list”
10000 loops, best of 5: 31.2 usec per loop
```

```
numbers_list = list(range(9999))

%timeit 5000 in numbers_list
32.4 µs ± 310 ns per loop (mean ± std. dev. of 7
    runs, 10000 loops each)
```

sys.getsizeof

- Tool om geheugengebruik te meten
- In Python Standard Library

```
>>> sys.getsizeof(list(range(999)))
8048
>>> sys.getsizeof(array.array('i', list(range(999))))
4060
```

Performant programmeren

- Technieken vergelijken

Types

- Weet waar je mee werkt
- Dynamically typed
- Strongly typed

```
>>> a = 5
>>> a = str(5)
>>> a
'5'
```

```
>>> print("9" + 9)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (not
    "int") to str
```

Data structuren

- List
- Tuple
- Set
- Dict
- Array

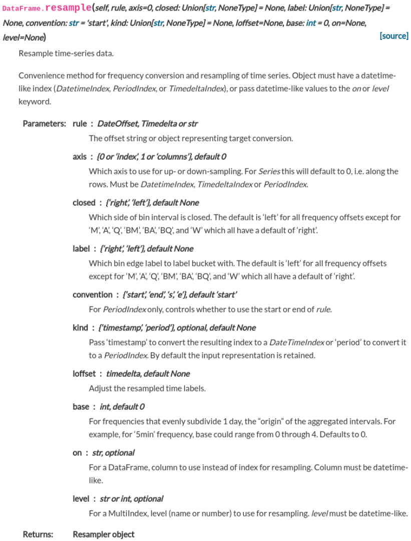
	in	append
List	O(n)	O(1)
Tuple	O(n)	/
Set	O(1)	O(1)
Dict	O(1)	O(1)
Array	O(n)	O(1)

```
numbers_list = list(range(9999))
numbers_tuple = tuple(range(9999))
numbers_set = set(range(9999))
numbers_dict = dict([(x, x) for x in range(9999)])
numbers_array = array.array('i', list(range(9999)))

%timeit 5000 in numbers_list
  32.4 µs ± 310 ns per loop
%timeit 5000 in numbers_tuple
  28.2 µs ± 384 ns per loop
%timeit 5000 in numbers_set
  24.2 ns ± 0.108 ns per loop
%timeit 5000 in numbers_dict
  28.3 ns ± 0.553 ns per loop
%timeit 5000 in numbers_array
  76.5 µs ± 529 ns per loop
```

Vermijd libraries

- Toegankelijkheid maakt traag



DataFrame.**resample**(*self, rule, axis=0, closed: Union[str, NoneType] = None, label: Union[str, NoneType] = None, convention: str = 'start', kind: Union[str, NoneType] = None, loffset=None, base: int = 0, on=None, level=None*) [source]

Resample time-series data.

Convenience method for frequency conversion and resampling of time series. Object must have a datetime-like index (*DatetimeIndex*, *PeriodIndex*, or *TimedeltaIndex*), or pass datetime-like values to the *on* or *level* keyword.

Parameters: **rule** : *DateOffset, Timedelta or str*

The offset string or object representing target conversion.

axis : *{0 or 'index', 1 or 'columns'}, default 0*

Which axis to use for up- or down-sampling. For *Series* this will default to 0, i.e. along the rows. Must be *DatetimeIndex*, *TimedeltaIndex* or *PeriodIndex*.

closed : *{'right', 'left'}, default None*

Which side of bin interval is closed. The default is 'left' for all frequency offsets except for 'M', 'A', 'Q', 'BM', 'BA', 'BQ', and 'W' which all have a default of 'right'.

label : *{'right', 'left'}, default None*

Which bin edge label to label bucket with. The default is 'left' for all frequency offsets except for 'M', 'A', 'Q', 'BM', 'BA', 'BQ', and 'W' which all have a default of 'right'.

convention : *{'start', 'end', 's', 'e'}, default 'start'*

For *PeriodIndex* only, controls whether to use the start or end of *rule*.

kind : *{'timestamp', 'period'}, optional, default None*

Pass 'timestamp' to convert the resulting index to a *DateTimeIndex* or 'period' to convert it to a *PeriodIndex*. By default the input representation is retained.

loffset : *timedelta, default None*

Adjust the resampled time labels.

base : *int, default 0*

For frequencies that evenly subdivide 1 day, the "origin" of the aggregated intervals. For example, for '5min' frequency, base could range from 0 through 4. Defaults to 0.

on : *str, optional*

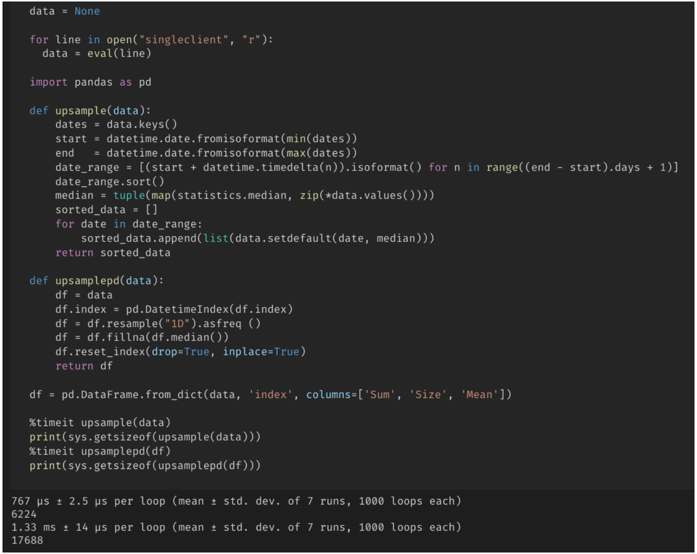
For a DataFrame, column to use instead of index for resampling. Column must be datetime-like.

level : *str or int, optional*

For a MultiIndex, level (name or number) to use for resampling. *level* must be datetime-like.

Returns: **Resampler object**

Vermijd libraries



```
data = None

for line in open("singleclient", "r"):
  data = eval(line)

import pandas as pd

def upsample(data):
    dates = data.keys()
    start = datetime.date.fromisoformat(min(dates))
    end   = datetime.date.fromisoformat(max(dates))
    date_range = [(start + datetime.timedelta(n)).isoformat() for n in range((end - start).days + 1)]
    date_range.sort()
    median = tuple(map(statistics.median, zip(*data.values())))
    sorted_data = []
    for date in date_range:
        sorted_data.append(list(data.setdefault(date, median)))
    return sorted_data

def upsamplepd(data):
    df = data
    df.index = pd.DatetimeIndex(df.index)
    df = df.resample("1D").asfreq ()
    df = df.fillna(df.median())
    df.reset_index(drop=True, inplace=True)
    return df

df = pd.DataFrame.from_dict(data, 'index', columns=['Sum', 'Size', 'Mean'])

%timeit upsample(data)
print(sys.getsizeof(upsample(data)))
%timeit upsamplepd(df)
print(sys.getsizeof(upsamplepd(df)))
```

```
767 µs ± 2.5 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
6224
1.33 ms ± 14 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
17688
```

Vermijd loops

- Loops in Python zijn niet specifiek traag
- Andere methodes zijn sneller

```
def plusone(numbers):
  ret = []
  for n in numbers:
      ret.append(n + 1)
  return ret

numbers = range(9999)

%timeit plusone(numbers)
  487 μs ± 277 ns per loop
%timeit [n + 1 for n in numbers]
  252 μs ± 1.19 μs per loop
```

Laziness

- Developers zijn lui, waarom niet ook het programma?

```
numbers = list(range(9999))

%timeit [n + 1 for n in numbers]
   250 µs ± 324 ns per loop
   87616
%timeit (n + 1 for n in numbers)
   188 ns ± 2.69 ns per loop
   112
%timeit map(lambda n: n + 1, numbers)
   142 ns ± 0.735 ns per loop
   48
```

In het kort

- Weet waar je mee werkt
- Vermijd libraries
- Vermijd loops
- Pas laziness toe