Python Basic Style Guide - Prinsip Penamaan pada Python

Penamaan pada Python

Penamaan pada pustaka (library) Python agak sulit dibuat konsisten, mengingat jumlah paketnya sudah banyak dan terdapat beberapa library eksternal yang sudah tidak lagi dikelola. Namun, berikut adalah beberapa rekomendasi untuk penamaan. Modul dan Paket-paket yang baru (termasuk framework) sebaiknya ditulis dengan standar baru ini. Namun Anda juga dapat memilih mempertahankan styling pada pustaka lama yang sudah Anda gunakan sebelumnya. Sekali lagi, konsistensi internal lebih diutamakan.

Prinsip Overriding

Nama yang dilihat oleh user publik (misalnya API) sebaiknya merefleksikan penggunaan/fungsinya, tidak merefleksikan implementasinya. Misal nama fungsi berikut.

```
1. cariJalan()
```

akan lebih mudah dipahami dibanding,

```
1. aStarSearch()
```

Algoritma yang digunakan dapat dijelaskan dalam docstring ataupun komentar.

Penamaan Deskriptif

Terdapat berbagai cara penamaan. Akan sangat membantu jika Anda telah memilih sebuah cara penamaan, terlepas bagaimana cara tersebut digunakan. Beberapa cara penamaan yang umum, antara lain:

- b (satu karakter huruf kecil).
- B (satu karakter huruf besar).
- hurufkecil.
- huruf_kecil_dengan_pemisah_kata_garis_bawah.
- HURUFBESAR.
- HURUF_BESAR_DENGAN_PEMISAH_GARIS_BAWAH.

- hurufCampuran (mirip dengan CapWords, hanya berbeda di karakter paling awal).
- Huruf_Besar_Di_Awal_Kata_Dengan_Garis_Bawah.

Pada beberapa pustaka juga digunakan frase awalan pendek untuk mengelompokkan beberapa fungsi atau variabel yang berelasi atau berasal dari satu induk. Misalnya pada fungsi os.stat(), seluruh parameter menggunakan awalan st karena pada prinsipnya, fungsi tersebut akan memanggil properti pada struct (st_size, st_mode, st_mtime, dll). Atau pustaka X11 yang menggunakan awalan X pada seluruh fungsi publiknya.

Penggunaan frase atau huruf pada awal fungsi ini tidak disarankan pada Python, atau lebih tepatnya tidak dibutuhkan, karena struktur pada Python:

- Atribut dan method name bersifat pre-fixed dengan objek,
- Function name selalu diawali dengan module name.

Beberapa bentuk khusus yang umum ditemukan (dapat digabungkan dengan case convention):

1. _diawali_sebuah_garis_bawah: weak "internal use" indicator. impor fungsi
   from M import * tidak akan mengimpor objek dengan awalan garis bawah.

2. diakhiri_sebuah_garis bawah_: digunakan untuk mengatasi redundant dengan keyword / reserved words di Python, misal:
   Tkinter.Toplevel(master, class_='ClassName').

3. __diawali_dua_garis bawah: menegaskan bahwa sebuah objek merupakan bagian dari kelas tertentu (pada kelas FooBar, fungsi __boo menjadi _FooBar__boo).

4. __diawali_dan_diakhiri_dua_garis bawah__: Objek atau atribut tertentu yang diciptakan Python untuk digunakan dalam program: __init__, __import__ or __file__. Jangan membuatnya sendiri, hanya gunakan yang telah didokumentasikan.

Yang perlu diperhatikan dalam penamaan

Nama yang dihindari

Hindari karakter l (huruf L kecil), O (huruf o besar) atau I (huruf i besar) sebagai nama variabel 1 karakter karena mereka sulit dibedakan dengan angka satu dan nol. Saat Anda ingin menggunakan l (huruf l kecil), akan sangat membantu jika Anda menggunakan L besar.

ASCII

Identifiers yang digunakan pada library standar harus ASCII-Compatible - lihat PEP 3131.

Nama Paket dan Nama Modul

Nama Modul sebaiknya pendek/singkat, menggunakan huruf kecil dan opsional garis bawah (_) untuk meningkatkan keterbacaan. Nama Paket juga sebaiknya singkat, menggunakan huruf kecil, dan hindari garis bawah(_). Jika sebuah modul ekstensi yang ditulis dengan C/C++ memiliki modul Python yang menyediakan antar muka (interface), modul C/C++ tersebut ditulis dengan diawali garis bawah (misalnya _socket).

Nama Kelas

Gunakan CamelCase atau CapWords convention. Pastikan semua akronim (misal HTTP) ditulis keseluruhan dengan huruf besar.

Penulisan Tipe Variable

Umumnya menggunakan CamelCase atau CapWords, lebih pendek lebih baik:

```
1. T, AnyStr, Num
```

Jika terdapat covariant atau contravariant dari sebuah variabel, tambahkan di akhir variabel untuk mempermudah pembacaan. Covariant memungkinkan Anda menggunakan tipe turunan (lebih spesifik) dari yang telah ditentukan sebelumnya. Sedangkan, contravariant merupakan istilah yang merujuk pada kemampuan untuk menggunakan tipe yang lebih umum dari sebelumnya.

```
1. from typing import TypeVar
2. VT_co = TypeVar('VT_co', covariant=True)
3. KT_contra = TypeVar('KT_contra', contravariant=True)
```

Nama Exception

Karena exception seharusnya bertipe kelas, Anda juga menerapkan konvensi penamaan kelas pada exception. Bedanya, tambahkan “Error” atau nama deskriptif lain pada nama exception Anda.

Nama Variabel Global

Nama Variabel Global mengikuti fungsi/modul yang bersifat publik. Anda bisa menggunakan garis bawah untuk menghindari variabel tersebut di-import jika ia merupakan modul non-publik.

Nama Fungsi, Parameter, dan Variabel

dengan style tertentu.

Instansiasi Fungsi dan Method

Gunakan self sebagai argument pertama instansiasi method.

Gunakan cls sebagai argumen pertama pada class methods.

Jika nama argument fungsi merupakan reserved keyword, tambahkan garis bawah di akhir nama argument. Jangan mengorbankan keterbacaan nama dengan menyingkatnya. Mengganti argumen bernama class dengan class_ atau kelas, lebih baik daripada clss, misalnya.

Nama Method dan Nama Variabel dalam Method

Gunakan standar penamaan Fungsi: huruf kecil dengan pemisah kata garis bawah untuk meningkatkan keterbacaan. Tambahkan garis bawah sebagai awalan untuk method non-publik dan variabel internal pada fungsi.

Untuk menghindari kesamaan dengan subkelas, gunakan __dimulai_dua_garis_nama_method untuk memanggil proses yang tepat. Python menggabungkan nama modul dengan nama kelas. Jika kelas Foo memiliki atribut __a, maka kita tidak dapat mengaksesnya melalui Foo.__a, melainkan Foo._Foo__a. Mulai dengan dua garis bawah hanya digunakan jika terjadi konflik dengan atribut di kelas atau subkelas lainnya.

Konstanta

Konstanta umumnya didefinisikan pada bagian atas modul dengan huruf besar, misalnya MAX_OVERFLOW dan TOTAL.

Selalu Persiapkan untuk Inheritance

Saat sebuah metode dan variabel dalam sebuah kelas didefinisikan, sebaiknya Anda dapat langsung mengetahui atribut pada metode dan variabel tersebut, apakah publik atau non-publik. Jika Anda ragu, jadikan atributnya non-publik. **Karena lebih mudah menjadikan sebuah variabel/method bersifat non-publik menjadi publik, dibandingkan sebaliknya.**

Method/Variabel Publik dipersiapkan untuk pihak eksternal menggunakan kelas Anda. Anda juga otomatis berkomitmen untuk menghindari adanya *incompatible backward changes*. Sebaliknya, Method/Variabel dengan atribut non-publik hanya digunakan oleh Anda sebagai developer, dan tidak memberikan garansi kepada siapapun bahwa Anda tidak akan mengubah atau menghapusnya. Di sini kita tidak menggunakan a karena di Python tidak ada atribut yang benar-benar **Private**.

perilaku (behavior) kelas. Dalam mendesain kelas-kelas sejenis, pastikan untuk membuat keputusan eksplisit, mana Variabel/Method yang memiliki atribut publik, bagian dari subclass API, dan mana yang hanya anda gunakan secara internal.

Saat mendeklarasikan variabel/method tersebut, ikuti panduan Pythonic berikut:

- Atribut publik tidak menggunakan awalan garis bawah.
- Jika nama sebuah Method/Variabel publik sama dengan reserved keyword, tambahkan akhiran garis bawah. Hindari menyingkat atau mengurangi huruf.
- Pada Data publik bersifat simple, hindari nama yang terlalu panjang. Cukup dengan nama atribut sependek mungkin. Ingatlah bahwa di masa depan Anda akan mungkin mengembangkan skema atau data ini, sehingga nama sependek mungkin akan menguntungkan Anda.
- Jika Anda berniat untuk mewariskan atau membuat subclass dari kelas, dan menginginkan sebuah variabel hanya digunakan di kelas utama saja, tambahkan awalan dua garis bawah. Ini akan memudahkan Anda karena Python mengenalinya sebagai konvensi kelas, menghindari kemungkinan kesamaan nama atau implementasi.

Interface

Umumnya, garansi backward compatibility hanya diaplikasikan pada interface publik. Untuk itu pengguna harus dapat membedakan dengan jelas, interface publik dan internal/non-publik. Interface yang didokumentasikan umumnya dianggap sebagai interface publik, kecuali dijelaskan secara eksplisit. Sebaliknya, setiap interface yang tidak terdokumentasi, umumnya dianggap bersifat internal.

Untuk keterbacaan, modul sebaiknya mendeklarasikan nama interface/API melalui atribut __all__. Jika __all__ kosong, artinya modul tersebut tidak memiliki interface/API Publik. Selain __all__ yang diisi dengan sesuai, internal interface (paket, modul, kelas, fungsi, atribut, atau nama lainnya), sebaiknya tetap dituliskan dengan diawali garis bawah.

Sebuah interface otomatis dianggap internal, jika namespace (paket, modul, atau kelasnya) bersifat internal.

Nama yang di-import harap selalu dianggap sebagai detail implementasi. Modul lainnya tidak diperbolehkan untuk melakukan akses tidak langsung untuk nama-nama tersebut, kecuali jika sudah didokumentasikan, misalnya os.path atau modul __init__ yang mengekspos fungsionalitas dari submodul.

Sebelumnya

Selanjutnya

Jawa Barat 40123

Tentang Kami

Penghargaan

image
click bila
belum muncul