Style Guide pada Python Code

Berikut adalah beberapa style guide menulis kode Python dengan baik dan benar. Panduan gaya penulisan kode ini mengacu pada PEP-008. Beberapa proyek mungkin memiliki style guide tersendiri. Sejumlah contoh kode yang ditulis di halaman ini berupa pseudocode, bertujuan hanya untuk memberikan gambaran tentang panduan gaya penulisan kode saja.

Guido, pembuat bahasa Python, merasakan bahwa kode lebih sering dibaca dibandingkan ditulis. Oleh sebab itu, panduan ini lebih ditekankan untuk kemudahan membaca kode dan membuatnya konsisten pada (hampir) setiap proyek Python yang ada.

Namun demikian pada kasus-kasus tertentu, keputusan adanya modifikasi tetap pada penulis kodenya. Mungkin sebuah kode dapat terbaca lebih jelas walaupun tidak mengikuti satu atau lebih panduan dalam modul ini.

Indentasi

Gunakan 4 spasi pada setiap tingkatan indentasi.

Python menggunakan indentasi untuk menulis kode bertingkat. Bahasa lain mungkin menggunakan statement tertentu (Begin, end - pascal), perbedaan baris atau kurung kurawal. Statement yang memiliki indentasi yang sama dan diletakkan secara berurutan dikenali sebagai blok statement oleh Python dan akan dijalankan secara berurutan.

```
1. Statement tingkat 1:
2.     Statement tingkat 2()
3.     Statement tingkat 2 yang kedua()
```

Baris Lanjutan

Seringkali, saat menulis kode, kita harus menggunakan baris lanjutan karena kode tidak cukup dituliskan dalam satu baris. Umumnya, kita dapat menggunakan tanda hubung, kurung, kurawal, atau seperti disarankan pada PEP-008, gunakan hanging indent. Beberapa panduan dalam menggunakan hanging indent dalam penulisan kode python adalah sebagai berikut:

Disarankan:

```
# Rata kiri dengan kurung atau pemisah dengan argumen utama
foo = long_function_name(var_one, var_two,
                         var_three, var_four)

# Opsi 2
# Tambahkan indentasi ekstra - (level indentasi baru) untuk memisahkan parameter/argument dari bagian lainnya
def long_function_name(
        var_one, var_two, var_three,
        var_four):
    print(var_one)

# Opsi 3
# Hanging indents dengan penambahan level indentasi saja
foo = long_function_name(
```

Tidak Disarankan:

```
# Contoh kesalahan 1
# Tidak rata kiri dengan bagian yang relevan
foo = long_function_name(var_one, var_two,
    var_three, var_four)

# Contoh kesalahan 2
# Sulit dibedakan antara baris lanjutan atau fungsi baru
def long_function_name(
    var_one, var_two, var_three,
    var_four):
    print(var_one)
```

> Catatan: 4 spasi bersifat opsional pada baris lanjutan, utamakan keterbacaan kode.

Anda juga dapat menggunakan jumlah spasi yang lain (misalnya 2) untuk baris lanjutan ini. Contohnya sepert ini:

```
# Hanging indents *boleh* menggunakan selain 4 spasi
foo = long_function_name(
  var_one, var_two,
  var_three, var_four)
```


Kondisional (If)

Saat menulis pernyataan kondisional, misalnya IF, kita juga menemukan penulisan yang terkadang tidak cukup dituliskan dalam satu baris, atau menggunakan beberapa operand yang akan menyulitkan apabila digabung berturut-turut.

Dalam kondisi ini, Python tidak memberikan panduan spesifik, mengingat kondisi yang dihadapi programmer mungkin berbeda. Contoh-contoh yang disarankan adalah sebagai berikut (meskipun dimungkinkan versi-versi lain selama keterbacaan kode tetap tinggi):

```
# Contoh kondisi visual yang tidak diubah/tanpa indentasi
if (sebuah kondisi dan
    kondisi yang lain):
    lakukanSesuatu()

# Tambahkan komentar
if (sebuah kondisi dan
    kondisi yang lain):
    #Mengingat Keduanya Benar, lakukan hal berikut
    lakukanSesuatu()

# Tambahkan ekstra indentasi pada baris lanjutan
if (sebuah kondisi dan
        kondisi yang lain):
    lakukanSesuatu()
```

Kurung/Siku Penutup

Penempatan kurung atau siku penutup juga dapat diletakkan pada baris lanjutan, dengan mengikuti posisi karakter pertama yang bukan whitespace (non-whitespace character) pada baris sebelumnya:

```
my_list = [
    1, 2, 3,
    4, 5, 6,
    ]

result = some_function_that_takes_arguments(
    'a', 'b', 'c',
    'd', 'e', 'f',
    )
```

```
my_list = [
    1, 2, 3,
    4, 5, 6,
]

result = some_function_that_takes_arguments(
    'a', 'b', 'c',
    'd', 'e', 'f',
)
```

Tab atau Spasi

Spasi adalah model yang disarankan PEP-008. Pengecualian pada kode yang sudah menggunakan tab/tabulasi sebelumnya. Python sejak versi 3 tidak memperbolehkan pencampuran antara Tab dan Spasi untuk indentasi. Anda disarankan untuk melakukan konversi kode untuk menggunakan spasi sepenuhnya.

Anda dapat menggunakan find-replace untuk mengganti tab, atau memanggil kode Anda yang berbasis Python 2 dengan opsi -t (warning) atau -tt (error) untuk mengetahui titik penggunaan tab dan spasi yang bercampur.

Panjang Baris Maksimum

Batasi panjang kode setiap baris hingga 79 karakter. Untuk komentar atau dokumentasi, usahakan untuk tidak melebihi 72 karakter.

Dengan membatasi panjang baris maksimum, Anda akan memudahkan pengguna lain membuka >1 window editor secara berdampingan, misalnya untuk melakukan review atau perbandingan. Panjang kode setiap baris yang dibatasi akan memudahkan Anda jika menggunakan *code review* tools yang menunjukkan dua versi berbeda secara berdampingan.

Mengapa 79? Hal ini dicontohkan pada editor-editor dengan window-width yang terset pada 80 karakter. 1 karakter tersisa bisa berupa marker glyph atau whitespace. Pembatasan 79 karakter ini membuat editor terkecil sekalipun tidak akan merusak struktur dan keterbacaan kode Anda. Jika Anda atau tim mengalami kesulitan (misalnya karena struktur penamaan variabel) yang telah disepakati, cenderung melebihi batasan panjang karakter, Anda dapat melakukan kesepakatan atau konvensi yang berlaku pada kode Anda sendiri. Umumnya hingga 99 karakter per baris.

> Catatan: Python Standard Library selalu dikembangkan secara konservatif dan mempertahankan standar 79 karakter pada kode, dan 72 pada komentar/dokumentasi.

Beberapa dari Anda mungkin mengenal pemisahan menggunakan backslash (\), namun tidak disarankan untuk digunakan, kecuali memang diharuskan. Contohnya adalah penggunaan backslash pada statement with atau assert yang tidak dapat menggunakan implicit continuation.

```
1. with open('/path/to/some/file/you/want/to/read') as file_1, \
2.      open('/path/to/some/file/being/written', 'w') as file_2:
3.     file_2.write(file_1.read())
```

Pastikan untuk memberikan indentasi yang sesuai pada baris-baris lanjutannya.

Sebelumnya

Selanjutnya