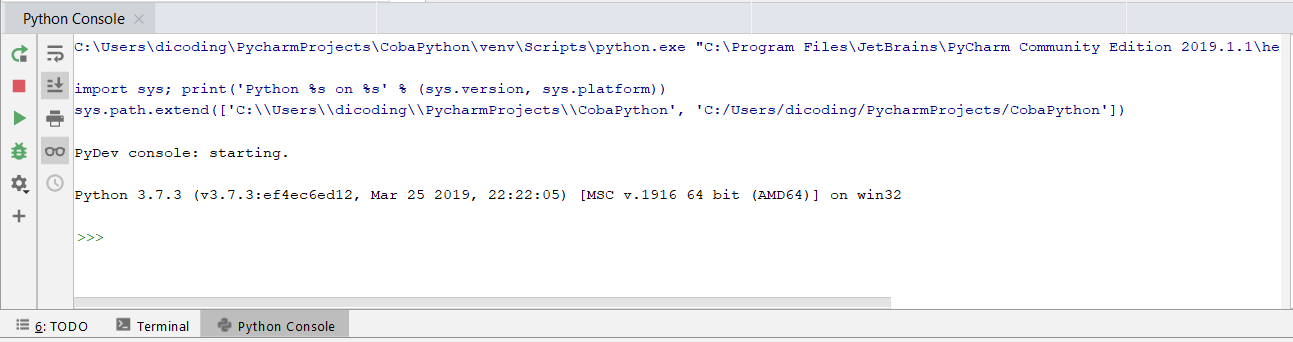
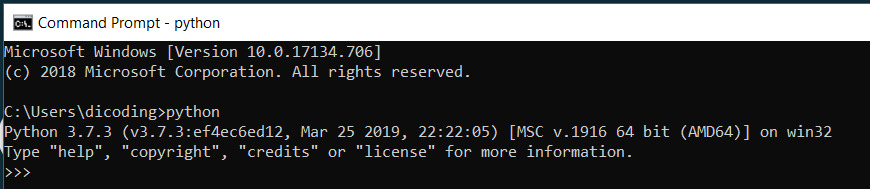
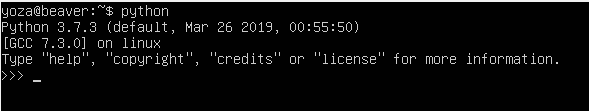
**Mode pada Python**

Pada Python dikenal beberapa mode operasi: Interactive, Script (scripting), dan Notebook.

**Interactive**

Berbeda dengan bahasa pemrograman lainnya, bahasa Python yang berbasis interpreter memungkinkan kita untuk menjalankan perintah secara interaktif. Mode ini dapat diakses di bagian bawah PyCharm atau dengan memanggil perintah python di command prompt/terminal.

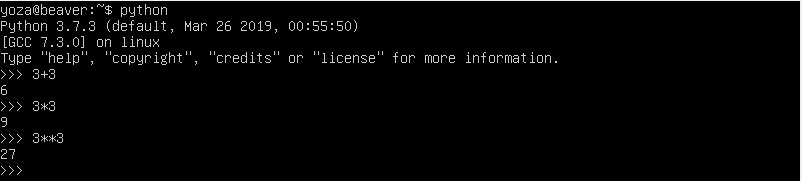
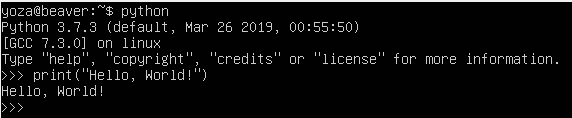
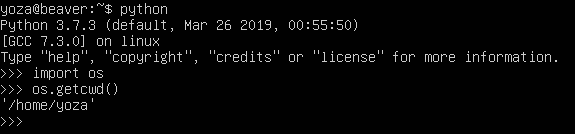
[](https://www.dicoding.com/academies/86/tutorials/4751)  
[](https://www.dicoding.com/academies/86/tutorials/4751)

[](https://www.dicoding.com/academies/86/tutorials/4751)

Catatan: pastikan Anda menjalankan versi yang tepat apabila tersedia >1 instalasi python di komputer Anda.

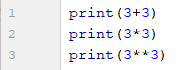
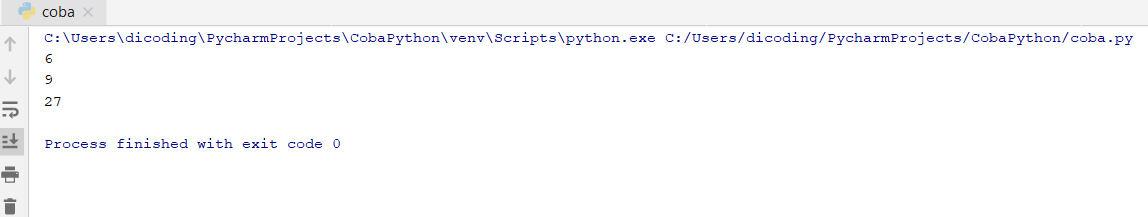
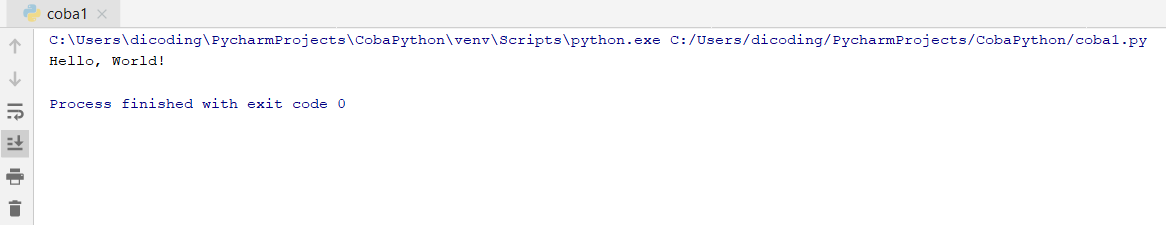
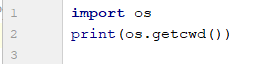
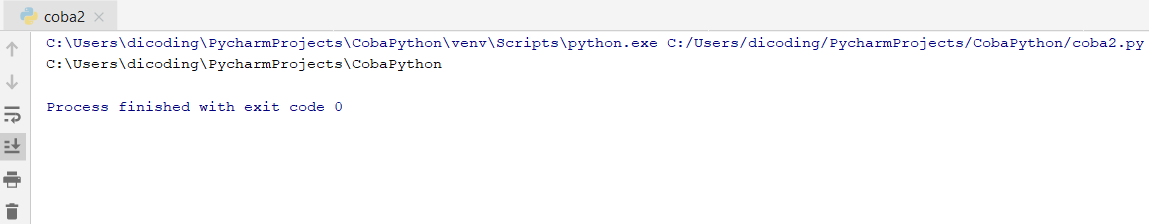
**Apa saja yang dapat Anda lakukan pada python interactive ini?**

Seluruh kode python dapat Anda jalankan secara berurutan pada sesi interaktif ini. Variabel juga akan tetap disimpan. Anda juga dapat memanggil (*import*) library. Sehingga salah satu penggunaan utama pada sesi interaktif ini adalah untuk rapid-prototyping.

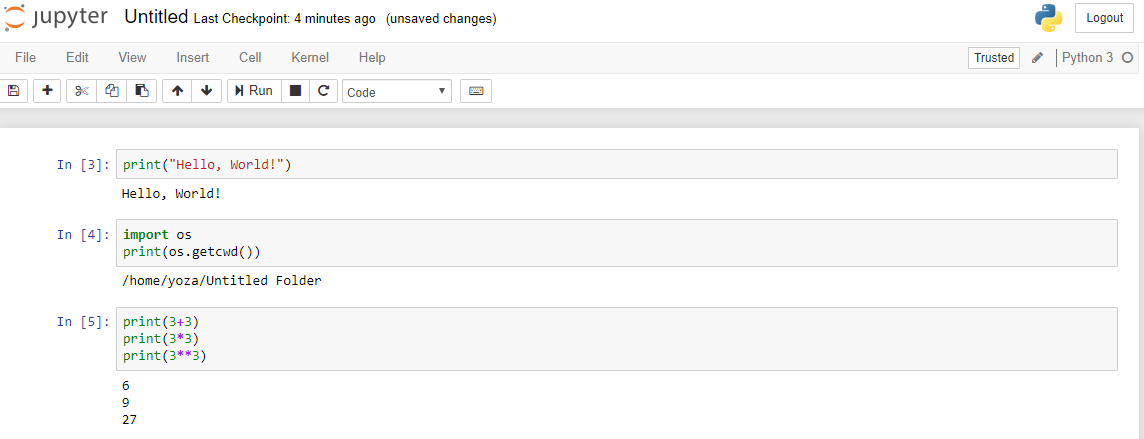
* **InteractiveExample: Calculator**  
  **[](https://www.dicoding.com/academies/86/tutorials/4751)**Anda dapat menggunakan python pada sesi interaktif ini menjadi kalkulator untuk perhitungan matematika.
* **InteractiveExample: Print**  
  **[](https://www.dicoding.com/academies/86/tutorials/4751)**
* **InteractiveExample:  import library dan mendapatkan Current working directory**  
  [](https://www.dicoding.com/academies/86/tutorials/4751)

**Script**

Mode yang lain dan sering dipergunakan pada python adalah script (scripting). Pada mode ini kita menggunakan sebuah berkas teks (umumnya berekstensi .py) dan kemudian akan dieksekusi oleh compiler/interpreter. Contoh yang sama untuk ketiga hal yang kita coba pada mode interactive adalah sebagai berikut:

* **RunnableExample: Calculator**  
  **[](https://www.dicoding.com/academies/86/tutorials/4751)**[](https://www.dicoding.com/academies/86/tutorials/4751)
* **RunnableExample: Print**  
  **[](https://www.dicoding.com/academies/86/tutorials/4751)**
* **RunnableExample: import library dan mendapatkan Current working directory**  
  [](https://www.dicoding.com/academies/86/tutorials/4751)**[](https://www.dicoding.com/academies/86/tutorials/4751)**

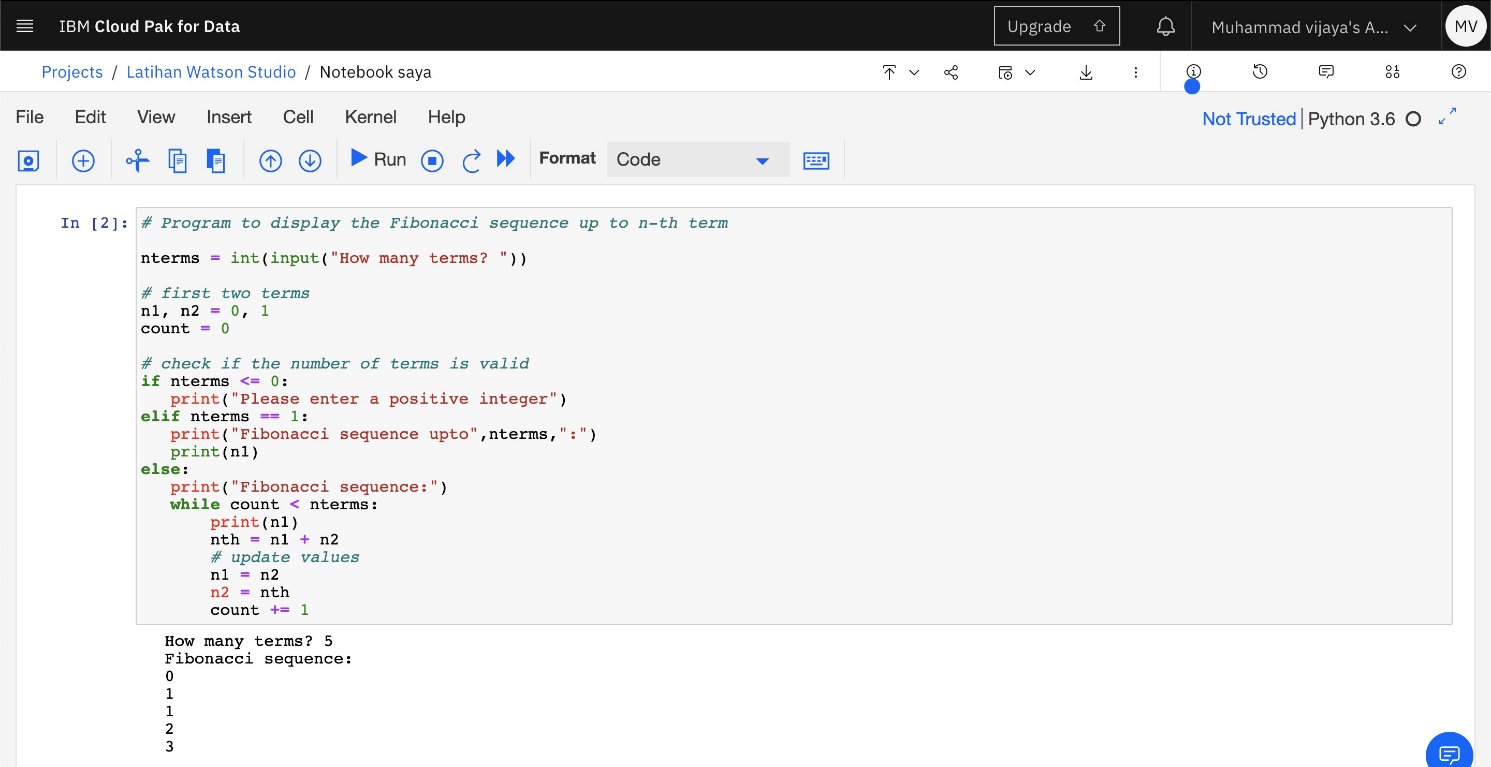
**Notebook**

[](https://www.dicoding.com/academies/86/tutorials/4751)

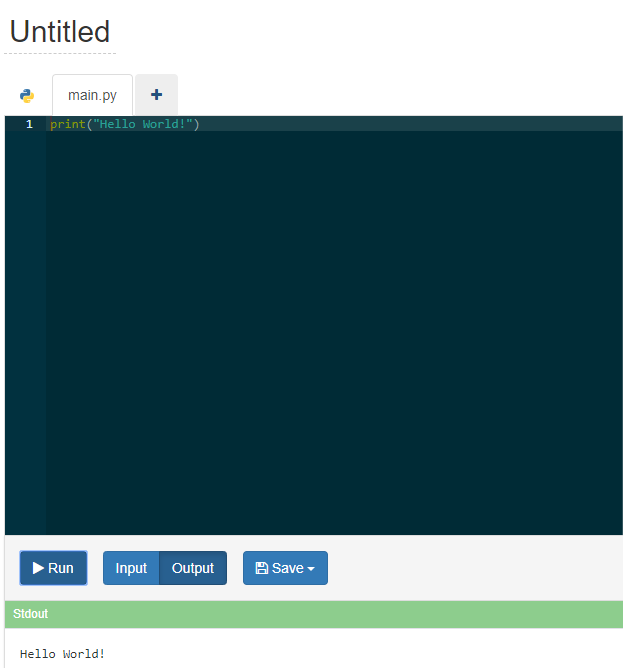
**Alternatif**

Alternatif yang lain, Anda dapat menggunakan tools online untuk menjalankan kode-kode Python Anda. beberapa yang umum digunakan di industri antara lain IBM Watson Studio, Google Colab, glot.io, REPL.it, dan ideone.com.

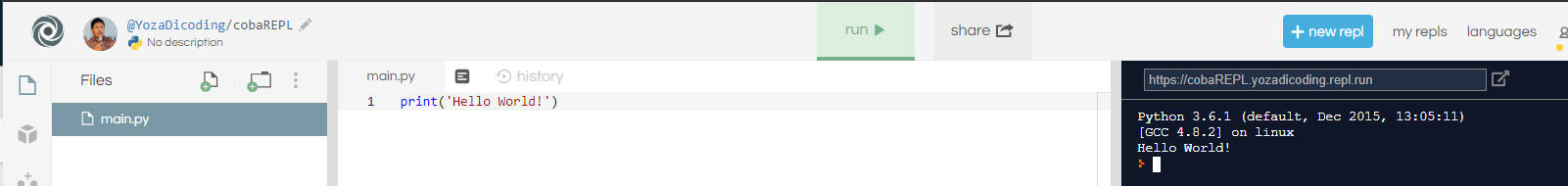
**Watson Studio**

**[](https://www.dicoding.com/academies/86/tutorials/4751)**

**Glot.io**

[](https://www.dicoding.com/academies/86/tutorials/4751)

**Repl.it**

[](https://www.dicoding.com/academies/86/tutorials/4751)

Anda dapat menggunakan yang mana saja dalam proses pembelajaran ini. Di industri, teknik yang dipilih adalah yang paling efisien (misalnya untuk research, kebanyakan menggunakan notebook), sementara untuk *backend* atau *infrastructure management*, dipilih script.

## 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](https://www.python.org/dev/peps/pep-0008/). 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:

1. # Opsi 1
2. # Rata kiri dengan kurung atau pemisah dengan argumen utama
3. foo = long\_function\_name(var\_one, var\_two,
4. var\_three, var\_four)
6. # Opsi 2
7. # Tambahkan indentasi ekstra - (level indentasi baru) untuk memisahkan parameter/argument dari bagian lainnya
8. def long\_function\_name(
9. var\_one, var\_two, var\_three,
10. var\_four):
11. print(var\_one)
13. # Opsi 3
14. # Hanging indents dengan penambahan level indentasi saja
15. foo = long\_function\_name(
16. var\_one, var\_two,
17. var\_three, var\_four)

#### Tidak Disarankan:

1. # Contoh kesalahan 1
2. # Tidak rata kiri dengan bagian yang relevan
3. foo = long\_function\_name(var\_one, var\_two,
4. var\_three, var\_four)
6. # Contoh kesalahan 2
7. # Sulit dibedakan antara baris lanjutan atau fungsi baru
8. def long\_function\_name(
9. var\_one, var\_two, var\_three,
10. var\_four):
11. 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:

1. # Hanging indents \*boleh\* menggunakan selain 4 spasi
2. foo = long\_function\_name(
3. var\_one, var\_two,
4. var\_three, var\_four)

### Kondisional (If)

Bagian ini hanya memberikan gambaran mengenai standar penulisan, pembahasan mengenai kondisional ada di modul Percabangan.

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):

1. # Contoh kondisi visual yang tidak diubah/tanpa indentasi
2. if (sebuah kondisi **dan**
3. kondisi yang lain):
4. lakukanSesuatu()
6. # Tambahkan komentar
7. if (sebuah kondisi **dan**
8. kondisi yang lain):
9. #Mengingat Keduanya Benar, lakukan hal berikut
10. lakukanSesuatu()
12. # Tambahkan ekstra indentasi pada baris lanjutan
13. if (sebuah kondisi **dan**
14. kondisi yang lain):
15. 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:

1. my\_list = [
2. 1, 2, 3,
3. 4, 5, 6,
4. ]
6. result = some\_function\_that\_takes\_arguments(
7. 'a', 'b', 'c',
8. 'd', 'e', 'f',
9. )

Atau dapat diletakkan sejajar dengan statemen utama, contoh:

1. my\_list = [
2. 1, 2, 3,
3. 4, 5, 6,
4. ]
6. result = some\_function\_that\_takes\_arguments(
7. 'a', 'b', 'c',
8. 'd', 'e', 'f',
9. )

### 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.

Seperti telah dibahas sebelumnya, Anda direkomendasikan untuk menggunakan baris lanjutan dengan kurung, kurawal, siku, maupun hanging indents. Baris yang cukup panjang dapat dipisahkan menjadi beberapa baris. 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.

## Python Basic Style Guide - Penggantian Baris, Komentar, dan Dokumentasi

### Mengganti baris : Sebelum atau Sesudah Operator Binary

Bagian ini hanya memberikan gambaran mengenai standar penulisan, pembahasan mengenai kondisional dibahas di modul Operator, Operands, dan Expressions.

Penggantian baris setelah operator binary memang pernah menjadi rekomendasi. Namun ternyata penggunaan metode ini membuat mata cepat lelah dan Anda perlu melakukan pengecekan ulang pada baris berbeda. Contohnya:

income = (gross\_wages +

          taxable\_interest +

          (dividends - qualified\_dividends) -

          ira\_deduction -

          student\_loan\_interest)

Untuk menyelesaikan masalah ini, dipilih pendekatan baris baru sebelum operator binary. Hal ini untuk mempermudah pembaca kode mengerti operasi yang dilakukan terhadap variabel berikutnya.

income = (gross\_wages

          + taxable\_interest

          + (dividends - qualified\_dividends)

          - ira\_deduction

          - student\_loan\_interest)

Kedua pendekatan ini dimungkinkan di Python. Anda direkomendasikan untuk menggunakan pendekatan kedua (baris baru sebelum operator) untuk menulis kode baru.

### Baris Kosong

Anda disarankan untuk menambahkan dua baris kosong pada top level function dan class definitions. Kemudian untuk setiap deklarasi method, dipisahkan dengan satu baris kosong.

Anda juga dapat menambahkan baris kosong ini apabila dibutuhkan, misalnya untuk memisahkan gabungan beberapa fungsi yang memiliki fungsi terkait atau untuk meningkatkan keterbacaan kode. Pemisahan baris kosong tidak diperlukan jika deklarasi fungsi/method Anda bersifat satu baris (one-liner), umumnya untuk fungsi/method yang belum diimplementasikan secara penuh.

### File Encoding

Kode dalam inti Python, selalu menggunakan [encoding](https://id.wikipedia.org/wiki/Pengodean_karakter) UTF-8 (Python 3) atau ASCII (Python 2). Dalam hal ini, apabila dalam sebuah berkas tidak ditulis deklarasi encoding, maka berkas tersebut menggunakan encoding ASCII (Python 2) atau UTF-8 (Python 3). Dalam standard library, non-default encoding hanya digunakan untuk pengujian atau memberikan komentar/dokumentasi, misalnya nama penulis yang tidak menggunakan karakter ASCII.

Untuk Python 3 dan seterusnya, pada standard library hanya menggunakan karakter ASCII dan sebisa mungkin menggunakan kata-kata dalam Bahasa Inggris. Proyek yang menggunakan python 3 didorong untuk menggunakan standar yang sama. Lihat [PEP 3131](https://www.python.org/dev/peps/pep-3131).

### Import

Saat melakukan import library, lakukan import setiap library pada baris berbeda.

Yes: import os

     import sys

No:  import sys, os

Kecuali, jika anda memerlukan lebih dari satu sub-library dari library yang sama.

from subprocess import Popen, PIPE

Import umumnya diletakkan pada bagian awal berkas. Setelah komentar dan dokumentasi tentang berkas tersebut (misalnya definisi kelas, dll), sebelum variabel global dan konstanta. Jika memungkinkan, kelompokkan import dalam urutan berikut:

1. Standard Library
2. Library Pihak Ketiga
3. Local/Library spesifik

Setiap grup baiknya dipisahkan oleh sebuah baris kosong.

Pada Python 2, dikenal explicit relative import, yakni proses import yang menggunakan path relatif yang digunakan. Pada Python 3, seluruh import yang dilakukan bersifat absolute (beserta seluruh path secara penuh).

import mypkg.sibling

from mypkg import sibling

from mypkg.sibling import example

Kode pada Standard library umumnya dapat menggunakan absolute import. Anda juga dapat mengimpor kelas/sub-library, Anda tentu saja dapat menggunakan pemanggilan berikut:

from myclass import MyClass

from foo.bar.yourclass import YourClass

Jika ada penamaan kelas yang sama, gunakan pemanggilan secara eksplisit:

import myclass

import foo.bar.yourclass

saat memanggil, gunakan "myclass.MyClass" dan "foo.bar.yourclass.YourClass".

from <module> import \*

Wildcard imports seperti tertulis, sedapat mungkin dihindari untuk mengatasi ambiguitas dan ketidaktahuan tentang modul apa yang di-import.

### Tanda Petik

Petik tunggal (‘) dan petik ganda (“) dianggap sama oleh Python, dan tidak memiliki preferensi khusus untuk penggunaannya. Hal ini dikarenakan ada kemungkinan string yang memuat salah satunya. Anda disarankan untuk menggunakan salah satunya secara konsisten.

Docstring (dokumentasi kode/fungsi/method) pada Python didefinisikan dengan tiga tanda petik, disarankan tanda petik ganda (”””) pada awal dan akhir statement docstring.

### Whitespace pada Expressions dan Statements

Wajib dihindari penambahan whitespace yang tidak perlu.

Antara kurung, kurawal, kurung siku.

Yes: spam(ham[1], {eggs: 2})

No:  spam( ham[ 1 ], { eggs: 2 } )

Setelah koma, tanpa argumen lain setelahnya.

Yes: foo = (0,)

No:  bar = (0, )

Sebelum koma, titik dua, atau titik koma.

Yes: if x == 4: print x, y; x, y = y, x

No:  if x == 4 : print x , y ; x , y = y , x

Namun, jika Anda menggunakan titik dua/colon sebagai slice (sub-list), pastikan ia memiliki spasi/whitespace yang sama pada kedua sisinya.

**Yes:**

ham[1:9], ham[1:9:3], ham[:9:3], ham[1::3], ham[1:9:]

ham[lower:upper], ham[lower:upper:], ham[lower::step]

ham[lower+offset : upper+offset]

ham[: upper\_fn(x) : step\_fn(x)], ham[:: step\_fn(x)]

ham[lower + offset : upper + offset]

**No:**

ham[lower + offset:upper + offset]

ham[1: 9], ham[1 :9], ham[1:9 :3]

ham[lower : : upper]

ham[ : upper]

Saat memberikan parameter pada fungsi, sebelum kurung tidak boleh ada spasi.

Yes: spam(1)

No:  spam (1)

Saat memberikan parameter/index pada list, sebelum kurung siku tidak boleh ada spasi.

Yes: dct['key'] = lst[index]

No:  dct ['key'] = lst [index]

Saat membuat assignment pada variabel, sebaiknya tidak menambahkan whitespace yang tidak perlu.

Yes:

x = 1

y = 2

long\_variable = 3

No:

x             = 1

y             = 2

long\_variable = 3

### Rekomendasi Lainnya

Hindari menambahkan whitespace di belakang statement apapun, utamanya di statement akhir dalam sebuah baris, karena whitespace tersebut tidak mudah dilihat.

Biasakan untuk menambahkan satu spasi baik di kiri maupun kanan untuk operasi berikut:

1. Assignment (=),
2. Augmented assignment (+=, -=etc.),
3. Comparisons (==, <, >, !=, <>, <=, >=, in, not in, is, is not),
4. Booleans (and, or, not).

Jika operator dengan berbagai tingkatan prioritas digunakan, letakkan whitespace pada operator-operator dengan prioritas terendah. Namun Anda juga dapat menyesuaikannya sendiri.

Catatan: jangan pernah menggunakan >1 spasi dan gunakan spasi yang sama baik di sebelah kiri maupun kanan dari operator-operator binary Anda.

**Yes:**

i = i + 1

submitted += 1

x = x\*2 - 1

hypot2 = x\*x + y\*y

c = (a+b) \* (a-b)

**No:**

i=i+1

submitted +=1

x = x \* 2 - 1

hypot2 = x \* x + y \* y

c = (a + b) \* (a - b)

## Komentar

Dalam sebuah kode Python, Anda diajak untuk memastikan kode Anda terbaca oleh programmer lain. Salah satu caranya adalah dengan menggunakan fitur komentar untuk memberitahu fungsi atau informasi lain terkait kode Anda. Pastikan komentar Anda ter-update dan tidak mengalami kontradiksi dengan kode yang ada.

Umumnya, komentar dituliskan dalam kalimat utuh dengan memperhatikan penulisan (huruf besar di awal kalimat, huruf kecil saat diawali dengan identifier atau variabel, dan diakhiri titik di akhir kalimat). Anda juga bisa menggabungkan beberapa kalimat menjadi blok komentar dengan menambah dua spasi saat berganti kalimat dalam satu paragraf, kecuali pada kalimat terakhir.

Jika memungkinkan, tuliskan komentar dalam bahasa Inggris, kecuali Anda yakin bahwa pembaca komentar ini dipastikan mengerti bahasa Anda.

#### Blok Komentar

Blok komentar umumnya digunakan untuk menjelaskan fungsi utuh atau sub-fungsi yang mengikuti/berada di bawahnya. Blok komentar diindentasi setara dengan kode yang dijelaskan. Setiap barisnya diawali dengan # dan sebuah spasi serta setiap paragrafnya dimulai pada baris baru.

#### Komentar Inline

Komentar Inline pada Python umumnya diletakkan pada baris yang sama dengan kode. Umumnya dipisahkan dan dirapikan dengan jarak dua spasi dari kode yang dimaksud, diawali # dan sebuah spasi. Komentar inline dapat juga digunakan di atas baris yang ingin diberikan komentar, agar tidak mengurangi jumlah karakter yang dapat dituliskan dalam sebuah baris. Untuk semua jenis komentar, jangan menuliskan komentar untuk hal yang sudah langsung dapat dibaca dari kodenya, seperti contoh berikut:

**Tidak disarankan:**

x = x + 1                 # Tambahkan x

**Disarankan (kontekstual):**

x = x + 1                 # Mengakomodasi layar ukuran Z

### Dokumentasi

Guideline untuk menuliskan dokumentasi (docstring) yang baik tersedia di [PEP 257](https://www.python.org/dev/peps/pep-0257). Kuncinya:

* Buatlah dokumentasi untuk semua modul, fungsi, kelas, dan method yang bersifat public atau akan diakses publik.
* Docstring tidak diwajibkan pada method yang tidak bersifat public, namun Anda disarankan menambahkan komentar tentang Apa saja yang dilakukan fungsi/modul ini beserta informasi lainnya yang mungkin diperlukan. Komentar ini diletakkan setelah baris def.

[PEP 257](https://www.python.org/dev/peps/pep-0257) memberikan panduan detil yang dapat digunakan. Seperti yang sudah-sudah, Anda disarankan untuk menutup sebuah docstring yang lebih dari satu baris, pada baris baru berikutnya:

"""Return a foobang

Optional plotz says to frobnicate the bizbaz first.

"""

Untuk docstring satu baris, Anda disarankan untuk meletakkan penutup """ - nya pada baris yang sama.

Meskipun secara sintaksis Anda dapat menggantikan 3-tanda-kutip-dua """ dengan 3-tanda-kutip-satu ''', untuk penulisan komentar multi-baris, tetapi [PEP 257](https://www.python.org/dev/peps/pep-0257/) memberikan panduan gunakan 3-tanda-kutip-dua untuk dokumentasi (docstring).