Style python

**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:

1. income = (gross\_wages +
2. taxable\_interest +
3. (dividends - qualified\_dividends) -
4. ira\_deduction -
5. 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.

1. income = (gross\_wages
2. + taxable\_interest
3. + (dividends - qualified\_dividends)
4. - ira\_deduction
5. - 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" \t "_blank) 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.

1. Yes: import os
2. import sys
4. No:  import sys, os

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

1. 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).

1. import mypkg.sibling
2. from mypkg import sibling
3. 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:

1. from myclass import MyClass
2. from foo.bar.yourclass import YourClass

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

1. import myclass
2. import foo.bar.yourclass

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

1. 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.

1. Yes: spam(ham[1], {eggs: 2})
2. No:  spam( ham[ 1 ], { eggs: 2 } )

Setelah koma, tanpa argumen lain setelahnya.

1. Yes: foo = (0,)
2. No:  bar = (0, )

Sebelum koma, titik dua, atau titik koma.

1. Yes: if x == 4: print x, y; x, y = y, x
2. 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.

1. **Yes:**
2. ham[1:9], ham[1:9:3], ham[:9:3], ham[1::3], ham[1:9:]
3. ham[lower:upper], ham[lower:upper:], ham[lower::step]
4. ham[lower+offset : upper+offset]
5. ham[: upper\_fn(x) : step\_fn(x)], ham[:: step\_fn(x)]
6. ham[lower + offset : upper + offset]
8. **No:**
9. ham[lower + offset:upper + offset]
10. ham[1: 9], ham[1 :9], ham[1:9 :3]
11. ham[lower : : upper]
12. ham[ : upper]

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

1. Yes: spam(1)
2. No:  spam (1)

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

1. Yes: dct['key'] = lst[index]
2. No:  dct ['key'] = lst [index]

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

1. Yes:
2. x = 1
3. y = 2
4. long\_variable = 3
6. No:
7. x             = 1
8. y             = 2
9. 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.

1. **Yes:**
2. i = i + 1
3. submitted += 1
4. x = x\*2 - 1
5. hypot2 = x\*x + y\*y
6. c = (a+b) \* (a-b)
8. **No:**
9. i=i+1
10. submitted +=1
11. x = x \* 2 - 1
12. hypot2 = x \* x + y \* y
13. 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:**

1. x = x + 1                 # Tambahkan x

**Disarankan (kontekstual):**

1. 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:

1. """Return a foobang
2. Optional plotz says to frobnicate the bizbaz first.
3. """

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*).

**Tipe Data pada Python - Numbers, String, dan Boolean**

Dasar dari mempelajari Bahasa Pemrograman yang baru adalah pemahaman terhadap tipe data. Di sini Anda akan diajarkan tentang tipe data bawaan yang ada di Python 3 beserta contoh penggunaannya.

**Numbers**

Tipe numerik pada Python dibagi menjadi 3: int, float, complex. Cobalah bermain-main dengan contoh berikut:

1. a = 10
2. print(a, "bertipe", type(a))
3. b = 1.7
4. print(a, "bertipe", type(b))
5. c = 1+3j
6. print(c, " Bertipe bilangan kompleks? ", isinstance(1+3j,complex))

Output seharusnya:

|  |
| --- |
| 10 bertipe <class 'int'> 1.7 bertipe <class 'float'> (1+2j) Bertipe bilangan kompleks? True |

Integer tidak dibatasi oleh angka atau panjang tertentu, namun dibatasi oleh memori yang tersedia. Sehingga Anda tidak perlu menggunakan variabel yang menampung big number misalnya long long (C/C++), biginteger, atau sejenisnya. Contoh kode untuk menunjukkan bahwa Python tidak membatasi output integer adalah pencarian bilangan ke-10.000 pada deret fibonacci (catatan: bilangan ke-10.000 pada deret fibonacci memiliki panjang 2.090 digit) sebagai berikut:

1. x=[0]\*10005; #inisialisasi array 0 sebanyak 10005; x[0]=0
2. x[1]=1; #x[1]=1
4. for j in range(2,10001):
5. x[j]=x[j-1]+x[j-2] # Fibonacci
6. print(x[10000])

Output:

1. 33644764876431783266621612005107543310302148460680063906564769974680081442166662368155595513633734025582065332680836159373734790483865268263040892463056431887354544369559827491606602099884183933864652731300088830269235673613135117579297437854413752130520504347701602264758318906527890855154366159582987279682987510631200575428783453215515103870818298969791613127856265033195487140214287532698187962046936097879900350962302291026368131493195275630227837628441540360584402572114334961180023091208287046088923962328835461505776583271252546093591128203925285393434620904245248929403901706233888991085841065183173360437470737908552631764325733993712871937587746897479926305837065742830161637408969178426378624212835258112820516370298089332099905707920064367426202389783111470054074998459250360633560933883831923386783056136435351892133279732908133732642652633989763922723407882928177953580570993691049175470808931841056146322338217465637321248226383092103297701648054726243842374862411453093812206564914032751086643394517512161526545361333111314042436854805106765843493523836959653428071768775328348234345557366719731392746273629108210679280784718035329131176778924659089938635459327894523777674406192240337638674004021330343297496902028328145933418826817683893072003634795623117103101291953169794607632737589253530772552375943788434504067715555779056450443016640119462580972216729758615026968443146952034614932291105970676243268515992834709891284706740862008587135016260312071903172086094081298321581077282076353186624611278245537208532365305775956430072517744315051539600905168603220349163222640885248852433158051534849622434848299380905070483482449327453732624567755879089187190803662058009594743150052402532709746995318770724376825907419939632265984147498193609285223945039707165443156421328157688908058783183404917434556270520223564846495196112460268313970975069382648706613264507665074611512677522748621598642530711298441182622661057163515069260029861704945425047491378115154139941550671256271197133252763631939606902895650288268608362241082050562430701794976171121233066073310059947366875

**Batasan akurasi variabel bertipe float**

Python melakukan pemotongan pada digit ke 16 pada variabel float. Float atau bilangan pecahan dibatasi akurasinya pada 15 desimal. Yang membedakan Integer dan Float adalah titik (decimal points). Misalnya dalam penulisan angka 1 jenisnya Integer, tapi jika dituliskan sebagai 1.0 artinya berjenis Float atau pecahan.

1. b = 0.1234567890123456789
2. print(b)

Output:

|  |
| --- |
| 0.12345678901234568 |

Contoh jika berupa integer:

1. a = 1234567890123456789
2. print(a)

Output:

|  |
| --- |
| 1234567890123456789 |

Karena Python banyak digunakan juga oleh matematikawan, tipe bilangan di Python juga mendukung bilangan imajiner dan bilangan kompleks. Nilai bilangan kompleks (complex) dituliskan dalam formulasi x + yj, yakni bagian x adalah bilangan real dan y adalah bilangan imajiner. Contohnya adalah sebagai berikut:

1. c = 1+5j
2. print(c)

Output:

|  |
| --- |
| (1+5j) |

**Strings**

String adalah urutan dari karakter unicode yang dideklarasikan dengan petik tunggal atau ganda. String >1baris dapat ditandai dengan tiga petik tunggal atau ganda ''' atau """.

1. s = "Ini adalah string baris tunggal"
2. s = '''Ini adalah string
3. yang memiliki baris pertama
4. dan selanjutnya baris kedua'''

**Bool/Boolean**

Tipe data bool atau Boolean merupakan turunan dari bilangan bulat (integer atau int) yang hanya punya dua nilai konstanta: True dan False.

**Nilai Boolean**

Nilai konstanta False dan True merepresentasikan nilai kebenaran (truth values), meskipun ada nilai-nilai lain yang juga dianggap benar atau salah. Di dalam konteks angka, misalnya digunakan sebagai argumen dari operator matematika aritmatika, kedua nilai ini berlaku seperti halnya bilangan bulat 0 dan 1, sesuai False dan True.

Ada fungsi bawaan bool() yang dapat mengubah nilai menjadi nilai Boolean, apabila nilai tersebut dapat direpresentasikan sebagai nilai kebenaran (truth values).

Nilai kebenaran adalah sebuah nilai yang dapat diuji sebagai benar atau salah, untuk digunakan di sintaksis kondisi if atau while atau sebagai operan dari operasi Boolean.

Berikut adalah objek bawaan yang didefinisikan bernilai salah dalam pengujian nilai kebenaran:

* Konstanta yang sudah didefinisikan bernilai salah: None dan False.
* Angka nol dari semua tipe numeric: 0, 0.0, 0j, Decimal(0), Fraction(0, 1).
* Urutan (sequence) dan koleksi (collection) yang kosong: '', (), {}, set(), range(0).

Untuk objek yang didefinisikan sendiri, representasi nilai Boolean akan bergantung dari definisi metode (method) khusus bernama \_\_bool\_\_(self). Jika metode ini mengembalikan True maka interpretasi nilai dari objeknya akan True, demikian juga sebaliknya.

**Operasi Boolean**

Operasi dan fungsi bawaan yang memiliki hasil Boolean akan selalu mengembalikan 0 atau False untuk yang bernilai salah, serta 1 atau True untuk yang bernilai benar, kecuali dinyatakan berbeda dalam dokumentasi.

Operasi untuk tipe Boolean akan dijelaskan lebih lanjut di modul Operator, Operands, dan Expressions.

**Tipe Data pada Python - List, Slicing, Tuple, Set, dan Dictionary**

**List**

List adalah jenis kumpulan data terurut (ordered sequence), dan merupakan salah satu variabel yang sering digunakan pada Python. Serupa, namun tak sama dengan array pada bahasa pemrograman lainnya. Bedanya, elemen List pada Python tidak harus memiliki tipe data yang sama. Mendeklarasikan List cukup mudah dengan kurung siku dan elemen yang dipisahkan dengan koma.

**Setiap data di dalamnya dapat diakses dengan indeks yang dimulai dari 0.**

1. a = [1, 2.2, 'python']

Python mengenal slicing operator [] yang dapat melakukan ekstraksi sebuah item atau beberapa item yang berada dalam range tertentu pada tipe data urutan (sequences), misalnya list, string dan tuple. Beberapa tipe urutan juga mendukung "extended slicing" dengan parameter ketiga berupa "step".

* **x[0]** artinya mengambil elemen paling awal, dengan index 0 dari List x.
* **x[5]** artinya mengambil elemen dengan index 5 dari List x.
* **x[-1]**artinya mengambil elemen dengan index paling belakang ke-1 dari List x.
* **x[3:5]** artinya membuat list dari anggota elemen List x dengan index 3 hingga sebelum index 5 (tidak termasuk elemen dengan index 5, dalam hal ini hanya index 3-4).
* **x[:5]** artinya membuat list dari anggota elemen List x paling awal hingga sebelum index 5 (tidak termasuk elemen dengan index 5, dalam hal ini hanya index 0-4).
* **x[-3:]** artinya membuat list dari anggota elemen List x mulai index ke-3 dari belakang hingga paling belakang.
* **x[1:7:2]** artinya membuat list dari anggota elemen List x dengan index 1 hingga sebelum index 7, dengan "step" 2 (dalam hal ini hanya index 1, 3, 5).

1. x = [5,10,15,20,25,30,35,40]
2. print(x[5])
3. print(x[-1])
4. print(x[3:5])
5. print(x[:5])
6. print(x[-3:])
7. print(x[1:7:2])

Output:

|  |
| --- |
| 30 40 [20, 25] [5, 10, 15, 20, 25] [30, 35, 40] [10, 20, 30] |

Elemen pada list dapat diubah atau ditambahkan. Misalnya untuk melakukan perubahan kemudian penambahan:

1. x = [1,2,3]
2. x[2]=4
3. print (x)

Output:

|  |
| --- |
| [1, 2, 4] |

1. x = [1,2,3]
2. x[2]=4
3. x.append(5)
4. print(x)

Output:

|  |
| --- |
| [1, 2, 4, 5] |

**Untuk menghapus item pada list, gunakan fungsi del.**Ingat bahwa Indeks Python dimulai dari 0:

1. binatang = ['kucing', 'rusa', 'badak', 'gajah']
2. del binatang[2]
3. print(binatang)

Output:

|  |
| --- |
| ['kucing', 'rusa', 'gajah'] |

Coba tambahkan kembali:

1. del bintang [2]
2. print(binatang)

 Pada baris terbawah kode di atas, maka output akan menjadi:

Output:

|  |
| --- |
| ['kucing', 'rusa'] |

**Slicing pada String**

Karena string mirip dengan list, maka slicing operator [ ]  juga dapat digunakan pada string untuk mengambil isinya atau bahkan substring. Sebuah string utuh bersifat mutable (bisa diubah), namun elemennya bersifat immutable (tidak bisa diubah).

1. s = "Hello World!"
2. print(s[4]) #ambil karakter kelima dari string s
3. print(s[6:11]) #ambil karakter ketujuh hingga sebelas dari string s
4. s[5]="d" #ubah karakter keenam dari string s menjadi "d", seharusnya gagal karena immutable
5. s = "Halo Dunia!" #ubah isi string s menjadi "Halo Dunia!", seharusnya berhasil karena mutable
6. print (s)

Output:

|  |
| --- |
| 'o' 'World' Traceback (most recent call last):   File "<stdin>", line 1, in <module> TypeError: 'str' object does not support item assignment 'Halo Dunia!' |

**Tuple**

Tuple adalah jenis dari list yang tidak dapat diubah elemennya. Umumnya tuple digunakan untuk data yang bersifat sekali tulis, dan dapat dieksekusi lebih cepat. Tuple didefinisikan dengan kurung dan elemen yang dipisahkan dengan koma.

1. t = (5,'program', 1+3j)

Seperti list, kita dapat melakukan slicing, namun pada tuple kita tidak dapat melakukan perubahan:

1. t = (5,'program', 1+3j)
2. print(t[1])
3. print(t[0:3])
4. print(t[0]=10)

Output:

|  |
| --- |
| 'program' (5, 'program', (1+3j)) Traceback (most recent call last): File "<stdin>", line 1, in <module> TypeError: 'tuple' object does not support item assignment |

**Set**

Set adalah kumpulan item bersifat unik dan tanpa urutan (unordered collection). Didefinisikan dengan kurawal dan elemennya dipisahkan dengan koma. Pada Set kita dapat melakukan union dan intersection, sekaligus otomatis melakukan penghapusan data duplikat.

1. a = {1,2,2,3,3,3}
2. print(a)

Output:

|  |
| --- |
| {1, 2, 3} |

Karena set bersifat unordered, maka kita tidak bisa mengambil sebagian data / elemen datanya menggunakan proses slicing.

1. a = {1,2,3}
2. print(a[1])

Output:

|  |
| --- |
| Traceback (most recent call last): File "<string>", line 301, in runcode File "<interactive input>", line 1, in <module> TypeError: 'set' object does not support indexing |

**Dictionary**

Dictionary pada Python adalah kumpulan pasangan kunci-nilai (pair of key-value) yang bersifat tidak berurutan. Dictionary dapat digunakan untuk menyimpan data kecil hingga besar. Untuk mengakses datanya, kita harus mengetahui kuncinya (key). Pada Python, dictionary didefinisikan dengan kurawal dan tambahan definisi berikut:

1. Setiap elemen pair key-value dipisahkan dengan koma (,).
2. Key dan Value dipisahkan dengan titik dua (:).
3. Key dan Value dapat berupa tipe variabel/obyek apapun.
4. d = {1:'value','key':2}
5. print(type(d))

Output:

|  |
| --- |
| <class 'dict'> |

1. d = {1:'value','key':2}
2. print(type(d))
3. print("d[1] = ", d[1]);
4. print("d['key'] = ", d['key']);

Output:

|  |
| --- |
| <class 'dict'> d[1] = value d['key'] =  2 |

Dictionary bukan termasuk dalam implementasi urutan (sequences), sehingga tidak bisa dipanggil dengan urutan indeks. Misalnya dalam contoh berikut dicoba dengan indeks 2, tetapi menghasilkan error (KeyError) karena tidak ada kunci (key) 2:

1. d = {1:'value','key':2}
2. print(type(d))
3. print("d[1] = ", d[1]);
4. print("d['key'] = ", d['key']);
6. # Generates error
7. print("d[2] = ", d[2]);

Output:

|  |
| --- |
| <class 'dict'> d[1] = value d['key'] = 2  --------------------------------------------------------------------------- KeyError Traceback (most recent call last) <ipython-input-7-4b566e677ca2> in <module>() 1 d = {1:'value','key':2} ----> 2 print("d[2] = ", d[2]);  KeyError: 2 |

**Konversi (conversion, cast) antar tipe data**

Kita dapat melakukan konversi tipe data bawaan dengan menggunakan fungsi konversi tipe bawaan (standard type) misalnya: int(), float(), str(), dll.

1. print(float(5))

Output:

|  |
| --- |
| 5.0 |

Konversi float ke int akan bersifat floor/truncating atau menghilangkan nilai di belakang koma.

1. print(int(10.6))

Output:

|  |
| --- |
| 10 |

1. print(int(-10.6))

Output:

|  |
| --- |
| -10 |

Konversi dari-dan-ke string akan melalui pengujian dan dipastikan validitasnya.

1. print(float('2.5'))

Output:

|  |
| --- |
| 2.5 |

1. print(str(25))

Output:

|  |
| --- |
| '25' |

1. print(int('1p'))

Output:

|  |
| --- |
| Traceback (most recent call last): File "<string>", line 301, in runcode File "<interactive input>", line 1, in <module> ValueError: invalid literal for int() with base 10: '1p' |

Anda juga dapat melakukan konversi kumpulan data (set, list, tuple).

1. print(set([1,2,3]))

Output:

|  |
| --- |
| {1, 2, 3} |

1. print(tuple({5,6,7}))

Output:

|  |
| --- |
| (5, 6, 7) |

1. print(list('hello'))

Output:

|  |
| --- |
| ['h', 'e', 'l', 'l', 'o'] |

Untuk konversi ke dictionary, data harus memenuhi persyaratan key-value. Berikut adalah dua contoh konversi:

List dari beberapa List yang isinya pasangan nilai menjadi Dictionary.

Serta konversi List dari beberapa Tuple yang isinya pasangan nilai menjadi Dictionary.

1. print(dict([[1,2],[3,4]]))

Output:

|  |
| --- |
| {1: 2, 3: 4} |

1. print(dict([(3,26),(4,44)]))

Output:

|  |
| --- |
| {3: 26, 4: 44} |