MODUL PYTHON

IF – ELSE CONDITION

IF

Pengambilan keputusan (kondisi if) digunakan untuk mengantisipasi kondisi yang terjadi saat jalanya program dan menentukan tindakan apa yang akan diambil sesuai dengan kondisi.

Pada python ada beberapa statement/kondisi diantaranya adalah if, else dan elif Kondisi if digunakan untuk mengeksekusi kode jika kondisi bernilai benar.

Jika kondisi bernilai salah maka statement/kondisi if tidak akan di-eksekusi.

```
#Kondisi if adalah kondisi yang akan dieksekusi oleh program jika bernilai benar atau TRUE

nilai = 9

#jika kondisi benar/TRUE maka program akan mengeksekusi perintah dibawahnya
if(nilai > 7):

    print("Selamat Anda Lulus")

#jika kondisi salah/FALSE maka program tidak akan mengeksekusi perintah dibawahnya
```

Dari contoh diatas, jika program dijalankan maka akan mencetak string "Selamat Anda

Lulus Ujian" sebanyak 1 kali yaitu pada if pertama. Di if kedua statement bernilai salah, jadi perintah print("Selamat Anda Lulus") tidak akan dieksekusi.

IF ELSE

Pengambilan keputusan (kondisi if else) tidak hanya digunakan untuk menentukan tindakan apa yang akan diambil sesuai dengan kondisi, tetapi juga digunakan untuk menentukan tindakan apa yang akan diambil/dijalankan jika kondisi tidak sesuai.

Pada python ada beberapa statement/kondisi diantaranya adalah if, else dan elif Kondisi if digunakan untuk mengeksekusi kode jika kondisi bernilai benar.

Kondisi if else adalah kondisi dimana jika pernyataan benar (true) maka kode dalam if akan dieksekusi, tetapi jika bernilai salah (false) maka akan mengeksekusi kode di dalam else.

Dibawah ini adalah contoh penggunaan kondisi if else pada Python

```
#Kondisi if else adalah jika kondisi bernilai TRUE maka akan dieksekusi pada
if, tetapi jika bernilai FALSE maka akan dieksekusi kode pada else

nilai = 3

#Jika pernyataan pada if bernilai TRUE maka if akan dieksekusi, tetapi jika
FALSE kode pada else yang akan dieksekusi.

if(nilai > 7):

    print("Selamat Anda Lulus")
else:
```

Pada contoh diatas, jika program dijalankan maka akan mencetak string "Maaf Anda Tidak Lulus" karena pernyataan pada if bernilai FALSE

ELIF

Pengambilan keputusan (kondisi if elif) merupakan lanjutan/percabangan logika dari "kondisi if". Dengan elif kita bisa membuat kode program yang akan menyeleksi beberapa kemungkinan yang bisa terjadi. Hampir sama dengan kondisi "else", bedanya kondisi "elif" bisa banyak dan tidak hanya satu.

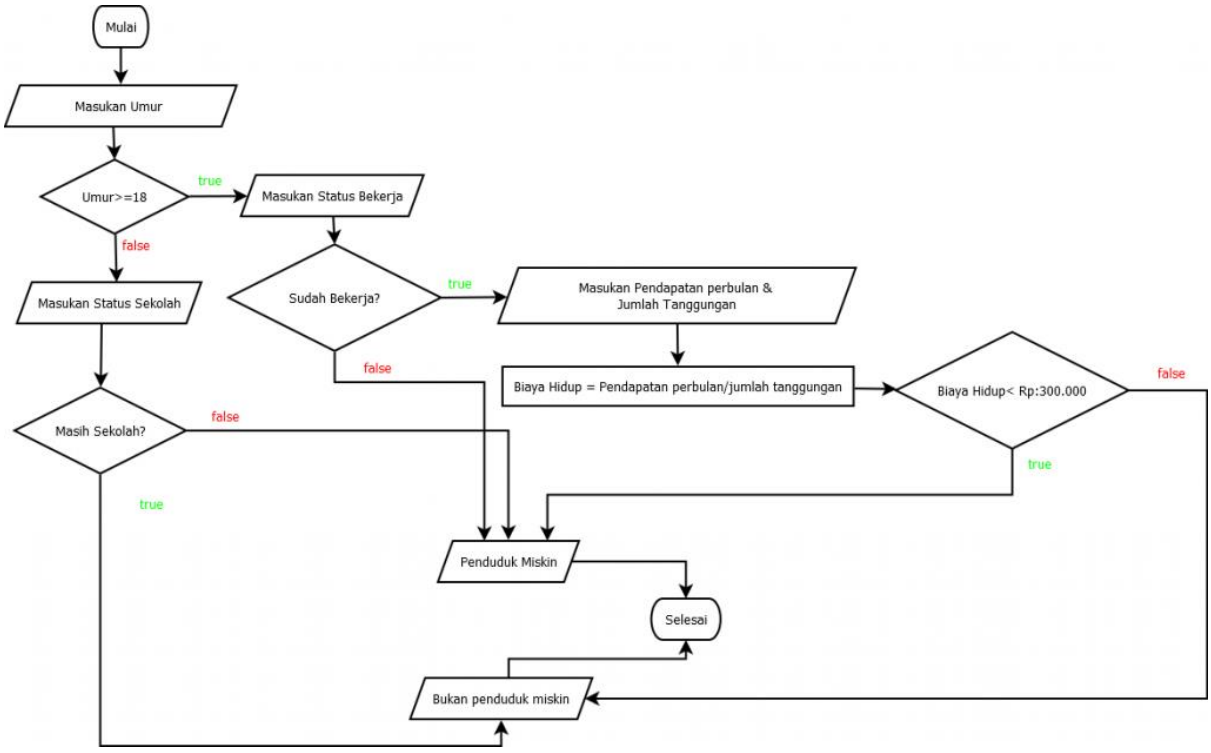
Dibawah ini adalah contoh penggunaan kondisi elif pada Python

```
#Contoh penggunaan kondisi elif
hari_ini  =  "Minggu"
if(hari_ini == "Senin"):

    print("Saya akan kuliah")
elif(hari_ini == "Selasa"):

    print("Saya akan kuliah")
elif(hari_ini == "Rabu"):

    print("Saya akan kuliah")
elif(hari_ini == "Kamis"):

    print("Saya akan kuliah")
elif(hari_ini == "Jumat"):

    print("Saya akan kuliah")
elif(hari_ini == "Sabtu"):
```

Pada contoh diatas, jika program dijalankan maka akan mencetak string "Saya akan libur".

LATIHAN IF ELSE

- Buatlah Program berdasarkan flowchart berikut :



Output 1 :

```
Masukan Usia Kamu = 45
Apakah kamu sudah bekerja = Y
Masukan pendapatan perbulan = 2000000
Masukan jumlah tanggungan = 7
-----------------------------------------------
Anda masuk kategori Penduduk Miskin
```

Output 2 :

```
Masukan Usia Kamu = 14
Apakah kamu masih sekolah = Y
-----------------------------------------------
Anda masuk kategori Bukan penduduk miskin
```

PERULANGAN LOOP

WHILE LOOP

Pengulangan While Loop di dalam bahasa pemrograman Python dieksesusi statement berkali-kali selama kondisi bernilai benar atau True.

Dibawah ini adalah contoh penggunaan pengulangan While Loop.

```
#Contoh penggunaan While Loop

count = 0

while (count < 9):

    print ('The count is:',
    count)count = count + 1
```

FOR LOOP

Pengulangan For pada Python memiliki kemampuan untuk mengulangi item dari urutan apapun, seperti list atau string.

Dibawah ini adalah contoh penggunaan pengulangan For Loop.

```
#Contoh pengulangan for sederhana
angka = [1,2,3,4,5]

for x in angka:
    print(x)

#Contoh pengulangan for

buah = ["nanas", "apel", "jeruk"]
for makanan in buah:
```

NESTED LOOP

Bahasa pemrograman Python memungkinkan penggunaan satu lingkaran di dalam loop lain. Bagian berikut menunjukkan beberapa contoh untuk menggambarkan konsep tersebut.

Dibawah ini adalah contoh penggunaan Nested Loop.

```
#Contoh penggunaan Nested Loop
i = 2

while(i < 100):

    j = 2

    while(j <= (i/j)):

        if not(i%j): break
        j = j + 1

    if (j > i/j) : print i, " is prime"
    i = i + 1

print "Good bye!"
```

LATIHAN LOOP

- Buatlah program yang dapat menghasilkan tampilan seperti gambar dibawah ini :

```
0 1
0 1 1
0 1 1 2
0 1 1 2 3
0 1 1 2 3 5
0 1 1 2 3 5 8
0 1 1 2 3 5 8 13
0 1 1 2 3 5 8 13 21
0 1 1 2 3 5 8 13 21 34
```

STRING

String adalah jenis yang paling populer di bahasa pemrograman. Kita bisa membuatnya hanya dengan melampirkan karakter dalam tanda kutip. Python memperlakukan tanda kutip tunggal sama dengan tanda kutip ganda. Membuat string semudah memberi nilai pada sebuah variabel.

Dibawah ini adalah contoh sederhana dari sebuah string pada bahasa pemrograman Python.

```
print("Hello World")
```

Python tidak menggunakan tipe karakter titik koma ; Ini diperlakukan sebagai string dengan panjang satu, sehingga juga dianggap sebagai substring.

Untuk mengakses substring, gunakan tanda kurung siku untuk mengiris beserta indeks atau indeks untuk mendapatkan substring Anda. Sebagai contoh :

```
name = 'John Doe' message = "John Doe belajar bahasa python di
Belajarpython"

print ("name[0]: ", name[0])
```

Bila kode diatas dieksekusi, maka akan menghasilkan hasil sebagai berikut :

name[0]: J message[1:4]: ohn

Mengupdate STRING

Anda dapat "memperbarui" string yang ada dengan (kembali) menugaskan variabel ke string lain. Nilai baru dapat dikaitkan dengan nilai sebelumnya atau ke string yang sama sekali berbeda sama sekali. Sebagai contoh

```
message = 'Hello World'
print ("Updated String :- ", message[:6] + 'Python')
```

Bila kode diatas dieksekusi, maka akan menghasilkan hasil sebagai berikut :

Updated String :- Hello Python

Operator Special String

Asumsikan variabel string adalah 'Belajar' dan variabel b adalah 'Python', lalu dibawah ini adalah operator yang bisa dipakai pada kedua string di variabel tersebut.

a = "Belajar" b = "Python"

Berikut adalah daftar operator spesial string pada Python :

Operator	Contoh	Penjelasan
+	a + b akan menghasilkan BelajarPython	Concatenation - Menambahkan nilai pada kedua sisi operator
*****	a*2 akan menghasilkan BelajarBelajar	Pengulangan - Membuat string baru, menggabungkan beberapa salinan dari string yang sama

[]	a[1] akan menghasilkan e	Slice - Memberikan karakter dari indeks yang diberikan
[:]	a[1:4] akan menghasilkan ela	Range Slice - Memberikan karakter dari kisaran yang diberikan
in	B in a akan menghasilkan 1	Keanggotaan - Mengembalikan nilai true jika ada karakter dalam string yang diberikan
not in	Z not in a akan menghasilkan 1	Keanggotaan - Mengembalikan nilai true jika karakter tidak ada dalam string yang diberikan
r/R	print r'\n' prints \n dan print R'\ n'prints \n	Raw String - Menekan arti aktual karakter Escape. Sintaks untuk string mentah sama persis dengan string biasa kecuali operator string mentah, huruf "r", yang mendahului tanda petik. "R" bisa berupa huruf kecil (r) atau huruf besar (R) dan harus ditempatkan tepat sebelum tanda kutip pertama.
%		Format - Melakukan format String

Operator Format String

Salah satu fitur Python yang paling keren adalah format string operator %. Operator ini unik untuk string dan membuat paket memiliki fungsi dari keluarga printf C () C. berikut adalah contoh sederhananya :

print ("My name is %s and weight is %d kg!" % ('Zara', 21))

Berikut adalah daftar lengkap simbol yang bisa digunakan bersamaan dengan % :

Operator	Penjelasan
%c	character
%s	Konversi string melalui str () sebelum memformat
%i	Dianggap sebagai bilangan bulat desimal
%d	Dianggap sebagai bilangan bulat desimal
%u	Unsigned decimal integer
%o	Bilangan bulat oktal
%x	Bilangan bulat heksadesimal (huruf kecil)
%X	Bilangan bulat heksadesimal (huruf besar)
%e	Notasi eksponensial (dengan huruf kecil 'e')
%E	Notasi eksponensial (dengan huruf besar 'E')
%f	Bilangan real floating point
%g	Yang lebih pendek dari% f dan% e
%G	Lebih pendek dari% f dan% E

Metode String Built-in

Metode	Penjelasan
`capitalize()`	Meng-kapitalkan huruf pertama string
`center(width, fillchar)`	Mengembalikan string yang dilapisi dengan fillchardengan string asli yang dipusatkan pada total widthkolom.
`count(str, beg = 0,end = len(string))`	Menghitung berapa kali str yang terjadi dalam stringatau dalam substring string jika memulai indeks **beg**dan **end** index end diberikan.
`decode(encoding = 'UTF-8',errors = 'strict')`	Dekode string menggunakan codec yang terdaftar untuk pengkodean. Encoding default ke pengkodeanstring default.
`encode(encoding = 'UTF-8',errors = 'strict')`	Mengembalikan versi string yang dikodekan string;Pada kesalahan, default adalah menaikkan ValueError kecuali jika kesalahan diberikan dengan'ignore' atau 'replace'.
`endswith(suffix, beg = 0, end = len(string))`	Menentukan apakah string atau substring string (jikamemulai indeks memohon dan mengakhiri akhir indeks diberikan) berakhir dengan akhiran; Mengembalikan nilai true jika benar dan salah.
`expandtabs(tabsize = 8)`	Memperluas tab dalam string ke banyak ruang; Default ke 8 spasi per tab jika tabsize tidak tersedia.
`find(str, beg = 0 end =len(string))`	Tentukan jika str terjadi dalam string atau dalam substring string jika memulai indeks beg dan end index end diberikan return index jika ditemukan dan -1 sebaliknya.
`index(str, beg = 0, end = len(string))`	Sama seperti find (), namun menimbulkanpengecualian jika str tidak ditemukan.
`isalnum()`	Mengembalikan true jika string memiliki minimal 1karakter dan semua karakternya alfanumerik dan false sebaliknya.
`isalpha()`	Mengembalikan true jika string memiliki minimal 1karakter dan semua karakter adalah abjad dan false sebaliknya.

Metode	Penjelasan
isdigit()	Mengembalikan true jika string hanya berisi digit dan false sebaliknya.
islower()	Mengembalikan true jika string memiliki setidaknya 1 karakter casing dan semua karakter casing dalam huruf kecil dan false sebaliknya.
isnumeric()	Mengembalikan true jika string unicode hanya berisi karakter numerik dan false sebaliknya.
isspace()	Mengembalikan true jika string hanya berisi karakter spasi dan false sebaliknya.
istitle()	Mengembalikan true jika string benar "titlecased" dan false sebaliknya.
isupper()	Mengembalikan true jika string memiliki setidaknya satu karakter casing dan semua karakter casing ada dalam huruf besar dan false sebaliknya.
join(seq)	Merges (concatenates) representasi string elemen dalam urutan seq menjadi string, dengan string pemisah.
len(string)	Mengembalikan panjang string
ljust(width[, fillchar])	Mengembalikan string berlapis ruang dengan string asli dibiarkan dibenarkan ke kolom lebar total.
lower()	Mengonversi semua huruf besar dalam bentuk string menjadi huruf kecil.
lstrip()	Menghapus semua spasi utama dalam string.
maketrans()	Mengembalikan tabel terjemahan untuk digunakan dalam fungsi terjemahan.
max(str)	Mengembalikan karakter alfabetik dari string str.
min(str)	Mengembalikan min karakter abjad dari string str.
replace(old, new [, max])	Menggantikan semua kemunculan lama dalam string dengan kejadian baru atau paling maksimal jika max diberikan.
rfind(str, beg = 0,end = len(string))	Sama seperti find (), tapi cari mundur dalam string.
rindex(str, beg = 0, end = len(string))	Sama seperti index (), tapi cari mundur dalam string.
rjust(width,[, fillchar])	Mengembalikan string berlapis ruang dengan senar asli benar-dibenarkan untuk total kolom lebar.
rstrip()	Menghapus semua spasi spasi string.
split(str="", num=string.count(str))	Membagi string sesuai dengan pemisah str (ruang jika tidak disediakan) dan mengembalikan daftar substring; Terpecah menjadi paling banyak substring jika diberikan.

Metode	Penjelasan
splitlines(num=string.count('\n'))	Membagi string sama sekali (atau num) NEWLINEs dan mengembalikan daftar setiap baris dengan NEWLINEs dihapus.
startswith(str, beg=0,end=len(string)	Determines if string or a substring of string (if starting index beg and ending index end are given) starts with substring str; returns true if so and false otherwise.
strip([chars])	Lakukan kedua lstrip () dan rstrip () pada string
swapcase()	Kasus invers untuk semua huruf dalam string.
title()	Mengembalikan versi string "titlecased", yaitu, semua kata diawali dengan huruf besar dan sisanya huruf kecil.
translate(table, deletechars="")	Menerjemahkan string sesuai dengan tabel terjemahan str (256 karakter), menghapus string del.
upper()	Mengonversi huruf kecil dalam bentuk string ke huruf besar.
zfill (width)	Mengembalikan string asli yang tertinggal dengan angka nol ke total karakter lebar; Dimaksudkan untuk angka, zfill () mempertahankan tanda apapun yang diberikan (kurang satu nol).
isdecimal()	Mengembalikan nilai true jika string unicode hanya berisi karakter desimal dan false sebaliknya.

Metode	Penjelasan
splitlines(num=string.count('\n'))	Membagi string sama sekali (atau num) NEWLINEs dan mengembalikan daftar setiap baris dengan NEWLINEs dihapus.
startswith(str, beg=0,end=len(string)	Determines if string or a substring of string (if starting index beg and ending index end are given) starts with substring str; returns true if so and false otherwise.
strip([chars])	Lakukan kedua lstrip () dan rstrip () pada string
swapcase()	Kasus invers untuk semua huruf dalam string.
title()	Mengembalikan versi string "titlecased", yaitu, semua kata diawali dengan huruf besar dan sisanya huruf kecil.
translate(table, deletechars="")	Menerjemahkan string sesuai dengan tabel terjemahan str (256 karakter), menghapus string del.
upper()	Mengonversi huruf kecil dalam bentuk string ke huruf besar.
zfill (width)	Mengembalikan string asli yang tertinggal dengan angka nol ke total karakter lebar; Dimaksudkan untuk angka, zfill () mempertahankan tanda apapun yang diberikan (kurang satu nol).
isdecimal()	Mengembalikan nilai true jika string unicode hanya berisi karakter desimal dan false sebaliknya.

LATIHAN STRING

Buatlah program yang dapat menyelesaikan permasalahan di bawah ini :

- Hitung berapa jumlah huruf dari nama lengkap mu masing-masing !
- Hitung berapa jumlah huruf vokal dari nama lengkap mu !
- Hitung berapa jumlah huruf konsonan dari nama lengkap mu !

List

List adalah tipe data yang paling serbaguna yang tersedia dalam bahasa Python, yang dapat ditulis sebagai daftar nilai yang dipisahkan koma (item) antara tanda kurung siku. Hal penting tentang daftar adalah item dalam list tidak boleh sama jenisnya.

Membuat list sangat sederhana, tinggal memasukkan berbagai nilai yang dipisahkan koma di antara tanda kurung siku. Dibawah ini adalah contoh sederhana pembuatan list dalam bahasa Python.

```
#Contoh sederhana pembuatan list pada bahasa pemrograman python
list1 = ['kimia', 'fisika', 1993, 2017]

list2 = [1, 2, 3, 4, 5 ]
```

Akses Nilai Dalam List

Untuk mengakses nilai dalam list python, gunakan tanda kurung siku untuk mengiris beserta indeks atau indeks untuk mendapatkan nilai yang tersedia pada indeks tersebut.

Berikut adalah contoh cara mengakses nilai di dalam list python :

```
#Cara mengakses nilai di dalam list Python

list1 = ['fisika', 'kimia', 1993, 2017]

list2 = [1, 2, 3, 4, 5, 6, 7 ]

print ("list1[0]: ", list1[0])

print ("list2[1:5]: ", list2[1:5])
```

Setelah Anda mengeksekusi kode diatas, hasilnya akan seperti dibawah ini :

list1[0]: fisika

list2[1:5]: [2, 3, 4, 5]

Update Nilai dalam List

Anda dapat memperbarui satu atau beberapa nilai di dalam list dengan memberikan potongan di sisi kiri operator penugasan, dan Anda dapat menambahkan nilai ke dalam list dengan metode append (). Sebagai contoh :

```
list = ['fisika', 'kimia', 1993, 2017]
print ("Nilai ada pada index 2 : ", list[2])
list[2] = 2001

print ("Nilai baru ada pada index 2 : ", list[2])
```

Hapus Nilai dalam List

Untuk menghapus nilai di dalam list python, Anda dapat menggunakan salah satu pernyataan del jika Anda tahu persis elemen yang Anda hapus. Anda dapat menggunakan metode remove() jika Anda tidak tahu persis item mana yang akan dihapus. Sebagai contoh :

```
#Contoh cara menghapus nilai pada list python
list = ['fisika', 'kimia', 1993, 2017]

print (list)
del list[2]

print ("Setelah dihapus nilai pada index 2 : ", list)
```

Operasi Dasar List

List Python merespons operator + dan * seperti string; Itu artinya penggabungan dan pengulangan di sini juga berlaku, kecuali hasilnya adalah list baru, bukan sebuah String.

Sebenarnya, list merespons semua operasi urutan umum yang kami gunakan pada String di bab sebelumnya. Dibawah ini adalah tabel daftar operasi dasar pada list python.

Python Expression	Hasil	Penjelasan
`len([1, 2, 3, 4])`	`4`	Length
`[1, 2, 3] + [4, 5, 6]`	`[1, 2, 3, 4, 5, 6]`	Concatenation
`['Halo!'] * 4`	`['Halo!', 'Halo!', 'Halo!', 'Halo!']`	Repetition
`2 in [1, 2, 3]`	`True`	Membership
`for x in [1,2,3] : print (x,end = ' ')`	`1 2 3`	Iteration

Indexing, Slicing dan Matrix pada List Python

Karena list adalah urutan, pengindeksan dan pengiris bekerja dengan cara yang sama untuk list seperti yang mereka lakukan untuk String.

Dengan asumsi input berikut :

L = ['C++", 'Java', 'Python']

Python Expression	Hasil	Penjelasan
`L[2]`	`'Python'`	Offset mulai dari nol
`L[-2]`	`'Java'`	Negatif: hitung dari kanan
`L[1:]`	`['Java', 'Python']`	Slicing mengambil bagian

Method dan Fungsi Build-in pada List Python

Python menyertakan fungsi built-in sebagai berikut

Python Function	Penjelasan
`cmp(list1, list2)`	# Tidak lagi tersedia dengan Python 3
`len(list)`	Memberikan total panjang list.
`max(list)`	Mengembalikan item dari list dengan nilai maks.
`min(list)`	Mengembalikan item dari list dengan nilai min.
`list(seq)`	Mengubah tuple menjadi list.

Python menyertakan methods built-in sebagai berikut

Python Method	Penjelasan
`list.append(obj)`	Menambahkan objek obj ke list
`list.count(obj)`	Jumlah pengembalian berapa kali obj terjadi dalam list
`list.extend(seq)`	Tambahkan isi seq ke list
`list.index(obj)`	Mengembalikan indeks terendah dalam list yang muncul obj
`list.insert(index, obj)`	Sisipkan objek obj ke dalam list di indeks offset
`list.pop(obj = list[-1])`	Menghapus dan mengembalikan objek atau obj terakhir dari list
`list.remove(obj)`	Removes object obj from list
`list.reverse()`	Membalik list objek di tempat
`list.sort([func])`	Urutkan objek list, gunakan compare func jika diberikan

LATIHAN LIST

TUPLE

Sebuah tupel adalah urutan objek Python yang tidak berubah. Tupel adalah urutan, seperti daftar. Perbedaan utama antara tupel dan daftarnya adalah bahwa tupel tidak dapat diubah tidak seperti List Python. Tupel menggunakan tanda kurung, sedangkan List Python menggunakan tanda kurung siku.

Membuat tuple semudah memasukkan nilai-nilai yang dipisahkan koma. Secara opsional, Anda dapat memasukkan nilai-nilai yang dipisahkan koma ini di antara tanda kurung juga. Sebagai contoh :

```
#Contoh sederhana pembuatan tuple pada bahasa pemrograman python
tup1 = ('fisika', 'kimia', 1993, 2017)

tup2 = (1, 2, 3, 4, 5 )

tup3 = "a", "b", "c", "d"
```

Tupel kosong ditulis sebagai dua tanda kurung yang tidak berisi apa-apa, contohnya :

tup1 = ();

Untuk menulis tupel yang berisi satu nilai, Anda harus memasukkan koma, meskipun hanya ada satu nilai, contohnya :

tup1 = (50,)

Seperti indeks String, indeks tuple mulai dari 0, dan mereka dapat diiris, digabungkan, dan seterusnya.

Akses Nilai dalam Tuple

Untuk mengakses nilai dalam tupel, gunakan tanda kurung siku untuk mengiris beserta indeks atau indeks untuk mendapatkan nilai yang tersedia pada indeks tersebut. Sebagai contoh

```
#Cara mengakses nilai tuple
tup1 = ('fisika', 'kimia', 1993, 2017)

tup2 = (1, 2, 3, 4, 5, 6, 7 )

print ("tup1[0]: ", tup1[0])

print ("tup2[1:5]: ", tup2[1:5])
```

Hasil nya :

tup1[0] : fisika

tup2[1:5] : (2,3,4,5)

Update Nilai dalam Tuple

Tupel tidak berubah, yang berarti Anda tidak dapat memperbarui atau mengubah nilai elemen tupel. Anda dapat mengambil bagian dari tupel yang ada untuk membuat tupel baru seperti ditunjukkan oleh contoh berikut.

```
tup1 = (12, 34.56)
tup2 = ('abc', 'xyz')

# Aksi seperti dibawah ini tidak bisa dilakukan pada tuple python
# Karena memang nilai pada tuple python tidak bisa diubah

# tup1[0] = 100;

# Jadi, buatlah tuple baru sebagai berikut
tup3 = tup1 + tup2

print (tup3)
```

Menghapus Nilai dalam Tuple

Menghapus elemen tuple individual tidak mungkin dilakukan. Tentu saja, tidak ada yang salah dengan menggabungkan tupel lain dengan unsur-unsur yang tidak diinginkan dibuang.

Untuk secara eksplisit menghapus keseluruhan tuple, cukup gunakan del statement. Sebagai contoh

```
tup = ('fisika', 'kimia', 1993, 2017);
print (tup)

del tup;

print "Setelah menghapus tuple : "
print tup
```

Operasi Dasar pada List Tuple

Tupel merespons operator + dan * sama seperti String; Mereka berarti penggabungan dan pengulangan di sini juga berlaku, kecuali hasilnya adalah tupel baru, bukan string.

Sebenarnya, list merespons semua operasi urutan umum yang kami gunakan pada String di bab sebelumnya. Dibawah ini adalah tabel daftar operasi dasar pada list python

Python Expression	Hasil	Penjelasan
`len((1, 2, 3))`	`3`	Length
`(1, 2, 3) + (4, 5, 6)`	`(1, 2, 3, 4, 5, 6)`	Concatenation
`('Halo!',) * 4`	`('Halo!', 'Halo!', 'Halo!', 'Halo!')`	Repetition
`3 in (1, 2, 3)`	`True`	Membership
`for x in (1,2,3) : print (x, end = ' ')`	`1 2 3`	Iteration

Indexing, Slicing, dan Matrix

Karena tupel adalah urutan, pengindeksan dan pengiris bekerja dengan cara yang sama untuk tupel seperti pada String, dengan asumsi masukan berikut

Dengan asumsi input berikut :

T = ('C++', 'Java', 'Python')

Python Expression	Hasil	Penjelasan
`T[2]`	`'Python'`	Offset mulai dari nol
`T[-2]`	`'Java'`	Negatif: hitung dari kanan
`T[1:]`	`('Java', 'Python')`	Slicing mengambil bagian

Fungsi Build-in

Python menyertakan fungsi built-in sebagai berikut :

Python Function	Penjelasan
`len(tuple)`	Memberikan total panjang tuple.
`max(tuple)`	Mengembalikan item dari tuple dengan nilai maks.
`min(tuple)`	Mengembalikan item dari tuple dengan nilai min.
`Tuple(seq)`	Mengubah tuple menjadi tuple.

LATIHAN TUPLE

Dictionary Python

Dictionary Python berbeda dengan List ataupun Tuple. Karena setiap urutanya berisi key dan value. Setiap key dipisahkan dari value-nya oleh titik dua (:), item dipisahkan oleh koma, dan semuanya tertutup dalam kurung kurawal. Dictionary kosong tanpa barang ditulis hanya dengan dua kurung kurawal, seperti ini: {}.

Nilai kamus bisa berupa tipe apa pun, namun key harus berupa tipe data yang tidak berubah seperti string, angka, atau tupel.

Akses Nilai

Untuk mengakses elemen Dictionary, Anda dapat menggunakan tanda kurung siku yang sudah dikenal bersama dengan key untuk mendapatkan nilainya. Berikut adalah contoh sederhananya :

```
#Contoh cara membuat Dictionary pada Python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
print ("dict['Name']: ", dict['Name'])
print ("dict['Age']: ", dict['Age'])
```

Update Nilai

Anda dapat memperbarui Dictionary dengan menambahkan entri baru atau pasangan nilai kunci, memodifikasi entri yang ada, atau menghapus entri yang ada seperti ditunjukkan pada contoh sederhana yang diberikan di bawah ini.

```
#Update dictionary python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
dict['Age'] = 8; # Mengubah entri yang sudah ada
dict['School'] = "DPS School" # Menambah entri baru
print ("dict['Age']: ", dict['Age'])
print ("dict['School']: ", dict['School'])
```

Hapus Nilai

Anda dapat menghapus elemen Dictionary individual atau menghapus keseluruhan isi Dictionary. Anda juga dapat menghapus seluruh Dictionary dalam satu operasi.

Untuk menghapus seluruh Dictionary secara eksplisit, cukup gunakan del statement. Berikut adalah contoh sederhana :

```
#Contoh cara menghapus pada Dictionary Python
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}
del dict['Name'] # hapus entri dengan key 'Name'
dict.clear()     # hapus semua entri di dict
del dict         # hapus dictionary yang sudah ada
print ("dict['Age']: ", dict['Age'])
print ("dict['School']: ", dict['School'])
```

LATIHAN DICTIONARY

FUNCTION

Fungsi adalah blok kode terorganisir dan dapat digunakan kembali yang digunakan untuk melakukan sebuah tindakan/action. Fungsi memberikan modularitas yang lebih baik untuk aplikasi Anda dan tingkat penggunaan kode yang tinggi.

Mendefinisikan Fungsi Python

Anda dapat menentukan fungsi untuk menyediakan fungsionalitas yang dibutuhkan. Berikut adalah aturan sederhana untuk mendefinisikan fungsi dengan Python.

1. Fungsi blok dimulai dengan def kata kunci diikuti oleh nama fungsi dan tanda kurung (()).
2. Setiap parameter masukan atau argumen harus ditempatkan di dalam tanda kurung ini. Anda juga dapat menentukan parameter di dalam tanda kurung ini.
3. Pernyataan pertama dari sebuah fungsi dapat berupa pernyataan opsional - string dokumentasi fungsi atau docstring.
4. Blok kode dalam setiap fungsi dimulai dengan titik dua (:) dan indentasi.
5. Pernyataan kembali [ekspresi] keluar dari sebuah fungsi, secara opsional menyampaikan kembali ekspresi ke pemanggil. Pernyataan pengembalian tanpa
6. argumen sama dengan return None.

Contoh :

```
def cetak(str):
   "This prints a passed string into this
   function"
   print (str)
   return
```

Latihan Function

Input Keyboard

Fungsi input([prompt]) setara dengan raw_input, kecuali mengasumsikan bahwa input adalah ekspresi Python yang valid dan mengembalikan hasil yang dievaluasi ke Anda.

```
>>> x = input("nilai : ")
nilai : 20
>>> x
20
>>> x = input("nilai : ")
nilai  : '20'
>>> x
'20'
```