Pendahuluan Python

Python adalah bahasa pemrograman interpretatif multiguna. Tidak seperti bahasa lain yang susah untuk dibaca dan dipahami, python lebih menekankan pada keterbacaan kode agar lebih mudah untuk memahami sintaks. Hal ini membuat Python sangat mudah dipelajari baik untuk pemula maupun untuk yang sudah menguasai bahasa pemrograman lain.

Bahasa ini muncul pertama kali pada tahun 1991, dirancang oleh seorang bernama Guido van Rossum. Sampai saat ini Python masih dikembangkan oleh Python Software Foundation. Bahasa Python mendukung hampir semua sistem operasi, bahkan untuk sistem operasi Linux, hampir semua distronya sudah menyertakan Python di dalamnya.

Dengan kode yang simpel dan mudah diimplementasikan, seorang programmer dapat lebih mengutamakan pengembangan aplikasi yang dibuat, bukan malah sibuk mencari syntax error.

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

Hanya dengan menuliskan kode print seperti yang diatas, anda sudah bisa mencetak apapun yang anda inginkan di dalam tanda kurung `()`. Dibagian akhir kode pun, anda tidak harus mengakhirnya dengan tanda semicolon `;`

Syntax bahasa Python hampir sama dengan bahasa pemrograman pada umumnya seperti Java atau PHP.

Syntax Dasar

Dibawah ini adalah contoh fungsi Python yang digunakan untuk mencetak. Di Python untuk mencetak cukup gunakan fungsi `print()` , dimana sesuatu yang akan dicetak harus diletakkan diantara kurung buka dan kurung tutup, bahkan di Python versi 2.x Anda tidak harus menggunakan tanda kurung kurawal, cukup pisahkan dengan spasi.

Jika ingin mencetak tipe data String langsung, Anda harus memasukanya ke dalam tanda kutip terlebih dahulu.

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

Saat anda menjalankan script diatas, Anda akan melihat output berupa text `Hello World`

Python Case Sensitivity

Python bersifat case sensitif, ini artinya huruf besar dan huruf kecil memiliki perbedaan. Sebagai contoh jika Anda menggunakan fungsi print dengan huruf kecil `print()` akan berhasil. Lain hal jika anda menggunakan huruf kapital `Print()` atau `PRINT()` , akan muncul pesan error.

Aturan ini berlaku untuk nama variabel ataupun fungsi-fungsi lainnya.

Komentar Python

Komentar (comment) adalah kode di dalam script Python yang tidak dieksekusi atau tidak dijalankan mesin. Komentar hanya digunakan untuk menandai atau memberikan keterangan tertulis pada script.

Komentar biasa digunakan untuk membiarkan orang lain memahami apa yang dilakukan script. atau untuk mengingatkan kepada programmer sendiri jika suatu saat kembali mengedit script tersebut.

Untuk menggunakan komentar anda cukup menulis tanda pagar `#`, diikuti dengan komentar Anda.

Dibawah ini adalah contoh penggunaan komentar pada Python

```python
#Ini adalah komentar
# Tulisan ini tidak akan dieksekusi

#komentar dengan tanda pagar hanya bisa digunakan
#untuk
#satu
#baris

"""
Penulisan Komentar lebih dari satu baris yaitu
dengan menggunakan kutip dua 3 kali dan
ditutup dengan kutip dua 3 kali juga
"""

print("Hello World") #ini juga komentar

#print("Welcome")

# komentar bisa berisi spesial karakter !@#$%^&*(),./;'[]\

#mencetak nama
print("Budi")

#mencetak angka/integer
print(123)
```

Saat anda menjalankan script diatas, Anda akan melihat output berupa `Hello World`, `Budi` dan `123`, karena tulisan/komentar yang ditulis tidak dieksekusi.

Tipe Data Python

Tipe data adalah suatu media atau memori pada komputer yang digunakan untuk menampung informasi.

Berikut adalah tipe data dari bahasa pemrograman Python :

Tipe Data	Contoh	Penjelasan
Boolean	`True` atau `False`	Menyatakan benar `True` yang bernilai `1`, atau salah `False` yang bernilai `0`
String	`"Ayo belajar Python"`	Menyatakan karakter/kalimat bisa berupa huruf angka, dll (diapit tanda `"` atau `'`)
Integer	`25` atau `1209`	Menyatakan bilangan bulat
Float	`3.14` atau `0.99`	Menyatakan bilangan yang mempunyai koma
Hexadecimal	`9a` atau `1d3`	Menyatakan bilangan dalam format heksa (bilangan berbasis 16)
Complex	`1 + 5j`	Menyatakan pasangan angka real dan imajiner

Tipe Data	Contoh	Penjelasan
List	`['xyz', 786, 2.23]`	Data untaian yang menyimpan berbagai tipe data dan isinya bisa diubah-ubah
Tuple	`('xyz', 768, 2.23)`	Data untaian yang menyimpan berbagai tipe data tapi isinya tidak bisa diubah
Dictionary	`{'nama': 'adi','id':2}`	Data untaian yang menyimpan berbagai tipe data berupa pasangan penunjuk dan nilai

Untuk mencoba berbagai macam tipe data, silahkan coba script Python dibawah ini.

```
#tipe data Boolean
print(True)

#tipe data String
print("Ayo belajar Python")
print('Belajar Python Sangat Mudah')

#tipe data Integer
print(20)

#tipe data Float
print(3.14)

#tipe data Hexadecimal
print(9a)

#tipe data Complex
print(5j)

#tipe data List
print([1,2,3,4,5])
print(["satu", "dua", "tiga"])

#tipe data Tuple
print((1,2,3,4,5))
print(("satu", "dua", "tiga"))
```

```
#tipe data Dictionary
print({"nama":"Budi", 'umur':20})
#tipe data Dictionary dimasukan ke dalam variabel biodata
biodata = {"nama":"Andi", 'umur':21} #proses inisialisasi variabel biodata
print(biodata) #proses pencetakan variabel biodata yang berisi tipe data Dictionary
type(biodata) #fungsi untuk mengecek jenis tipe data. akan tampil <class 'dict'> yang berarti dict adalah tipe data dictionary
```

Variabel Python

Variabel adalah lokasi memori yang dicadangkan untuk menyimpan nilai-nilai. Ini berarti bahwa ketika Anda membuat sebuah variabel Anda memesan beberapa ruang di memori. Variabel menyimpan data yang dilakukan selama program dieksekusi, yang nantinya isi dari variabel tersebut dapat diubah oleh operasi - operasi tertentu pada program yang menggunakan variabel.

Variabel dapat menyimpan berbagai macam tipe data. Di dalam pemrograman Python, variabel mempunyai sifat yang dinamis, artinya variabel Python tidak perlu didekralasikan tipe data tertentu dan variabel Python dapat diubah saat program dijalankan.

Penulisan variabel Python sendiri juga memiliki aturan tertentu, yaitu :

1. Karakter pertama harus berupa huruf atau garis bawah/underscore `_`
2. Karakter selanjutnya dapat berupa huruf, garis bawah/underscore `_` atau angka
3. Karakter pada nama variabel bersifat sensitif (case-sensitif). Artinya huruf kecil dan huruf besar dibedakan. Sebagai contoh, variabel `namaDepan` dan `namadepan` adalah variabel yang berbeda.

Untuk mulai membuat variabel di Python caranya sangat mudah, Anda cukup menuliskan variabel lalu mengisinya dengan suatu nilai dengan cara menambahkan tanda sama dengan `=` diikuti dengan nilai yang ingin dimasukan.

Dibawah ini adalah contoh penggunaan variabel dalam bahasa pemrograman Python

```
#proses memasukan data ke dalam variabel
nama = "John Doe"
#proses mencetak variabel
print(nama)
```

```python
#nilai dan tipe data dalam variabel  dapat diubah
umur = 20               #nilai awal
print(umur)             #mencetak nilai umur
type(umur)              #mengecek tipe data umur
umur = "dua puluh satu" #nilai setelah diubah
print(umur)             #mencetak nilai umur
type(umur)              #mengecek tipe data umur

namaDepan = "Budi"
namaBelakang = "Susanto"
nama = namaDepan + " " + namaBelakang
umur = 22
hobi = "Berenang"
print("Biodata\n", nama, "\n", umur, "\n", hobi)

#contoh variabel lainya
inivariabel = "Halo"
ini_juga_variabel = "Hai"
_inivariabeljuga = "Hi"
inivariabel222 = "Bye"

panjang = 10
lebar = 5
luas = panjang * lebar
print(luas)
```

Operator Python

Operator adalah konstruksi yang dapat memanipulasi nilai dari operan.

Sebagai contoh operasi 3 + 2 = 5. Disini `3` dan `2` adalah operan dan `+` adalah operator.

Bahasa pemrograman Python mendukung berbagai macam operator, diantaranya :

- Operator Aritmatika (Arithmetic Operators)
- Operator Perbandingan (Comparison (Relational) Operators)
- Operator Penugasan (Assignment Operators)
- Operator Logika (Logical Operators)
- Operator Bitwise (Bitwise Operators)
- Operator Keanggotaan (Membership Operators)
- Operator Identitas (Identity Operators)

Operator Aritmatika

Operator	Contoh	Penjelasan
Penjumlahan `+`	`1 + 3 = 4`	Menjumlahkan nilai dari masing-masing operan atau bilangan
Pengurangan `-`	`4 - 1 = 3`	Mengurangi nilai operan di sebelah kiri menggunakan operan di sebelah kanan
Perkalian `*`	`2 * 4 = 8`	Mengalikan operan/bilangan
Pembagian `/`	`10 / 5 = 2`	Untuk membagi operan di sebelah kiri menggunakan operan di sebelah kanan
Sisa Bagi `%`	`11 % 2 = 1`	Mendapatkan sisa pembagian dari operan di sebelah kiri operator ketika dibagi oleh operan di sebelah kanan
Pangkat `**`	`8 ** 2 = 64`	Memangkatkan operan disebelah kiri operator dengan operan di sebelah kanan operator
Pembagian Bulat `//`	`10 // 3 = 3`	Sama seperti pembagian. Hanya saja angka dibelakang koma dihilangkan

Dibawah ini adalah contoh penggunaan Operator Aritmatika dalam bahasa pemrograman Python

```
#OPERATOR ARITMATIKA

#Penjumlahan
print(13 + 2)
apel = 7
jeruk = 9
buah = apel + jeruk #
print(buah)

#Pengurangan
hutang = 10000
bayar = 5000
sisaHutang = hutang - bayar
print("Sisa hutang Anda adalah ", sisaHutang)

#Perkalian
panjang = 15
lebar = 8
luas = panjang * lebar
print(luas)

#Pembagian
kue = 16
anak = 4
kuePerAnak = kue / anak
print("Setiap anak akan mendapatkan bagian kue sebanyak ", kuePerAnak)

#Sisa Bagi / Modulus
bilangan1 = 14
bilangan2 = 5
hasil = bilangan1 % bilangan2
print("Sisa bagi dari bilangan ", bilangan1, " dan ", bilangan2, " adalah ", hasil)

#Pangkat
bilangan3 = 8
bilangan4 = 2
hasilPangkat = bilangan3 ** bilangan4
print(hasilPangkat)

#Pembagian Bulat
print(10//3)
#10 dibagi 3 adalah 3.3333. Karena dibulatkan maka akan menghasilkan nilai 3
```

Operator Perbandingan

Operator perbandingan (comparison operators) digunakan untuk membandingkan suatu nilai dari masing-masing operan.

Operator	Contoh	Penjelasan
Sama dengan `==`	`1 == 1`	bernilai True Jika masing-masing operan memiliki nilai yang sama, maka kondisi bernilai benar atau True.
Tidak sama dengan `!=`	`2 != 2`	bernilai False Akan menghasilkan nilai kebalikan dari kondisi sebenarnya.
Tidak sama dengan `<>`	`2 <> 2`	bernilai False Akan menghasilkan nilai kebalikan dari kondisi sebenarnya.
Lebih besar dari `>`	`5 > 3`	bernilai True Jika nilai operan kiri lebih besar dari nilai operan kanan, maka kondisi menjadi benar.
Lebih kecil dari `<`	`5 < 3`	bernilai True Jika nilai operan kiri lebih kecil dari nilai operan kanan, maka kondisi menjadi benar.
Lebih besar atau sama dengan `>=`	`5 >= 3`	bernilai True Jika nilai operan kiri lebih besar dari nilai operan kanan, atau sama, maka kondisi menjadi benar.
Lebih kecil atau sama dengan `<=`	`5 <= 3`	bernilai True Jika nilai operan kiri lebih kecil dari nilai operan kanan, atau sama, maka kondisi menjadi benar.

Operator Penugasan

Operator penugasan digunakan untuk memberikan atau memodifikasi nilai ke dalam sebuah variabel.

Operator	Contoh	Penjelasan
Sama dengan `=`	`a = 1`	Memberikan nilai di kanan ke dalam variabel yang berada di sebelah kiri.
Tambah sama dengan `+=`	`a += 2`	Memberikan nilai variabel dengan nilai variabel itu sendiri ditambah dengan nilai di sebelah kanan.
Kurang sama dengan `-=`	`a -= 2`	Memberikan nilai variabel dengan nilai variabel itu sendiri dikurangi dengan nilai di sebelah kanan.
Kali sama dengan `*=`	`a *= 2`	Memberikan nilai variabel dengan nilai variabel itu sendiri dikali dengan nilai di sebelah kanan.
Bagi sama dengan `/=`	`a /= 4`	Memberikan nilai variabel dengan nilai variabel itu sendiri dibagi dengan nilai di sebelah kanan.
Sisa bagi sama dengan `%=`	`a %= 3`	Memberikan nilai variabel dengan nilai variabel itu sendiri dibagi dengan nilai di sebelah kanan. Yang diambil nantinya adalah sisa baginya.

Operator	Contoh	Penjelasan
Pangkat sama dengan `**=`	`a **= 3`	Memberikan nilai variabel dengan nilai variabel itu sendiri dipangkatkan dengan nilai di sebelah kanan.
Pembagian bulat sama dengan `//=`	`a //= 3`	Membagi bulat operan sebelah kiri operator dengan operan sebelah kanan operator kemudian hasilnya diisikan ke operan sebelah kiri.

Prioritas Eksekusi Operator di Python

Dari semua operator diatas, masing-masing mempunyai urutan prioritas yang nantinya prioritas pertama akan dilakukan paling pertama, begitu seterusnya sampai dengan prioritas terakhir.

Operator	Keterangan
`**`	Aritmatika
`~, +, -`	Bitwise

Operator	Keterangan
`*, /, %, //`	Aritmatika
`+, -`	Aritmatika
`>>, <<`	Bitwise
`&`	Bitwise
`^, |`	Bitwise
`<=, <, >, >=`	Perbandingan
`<> , ==, !=`	Perbandingan
`=, %=, /=, //=, -=, +=, *=, **=`	Penugasan
`is, is not`	Identitas
`in, not in`	Membership (Keanggotaan)

Operator	Keterangan
`not, or, and`	Logika

Kondisi Python

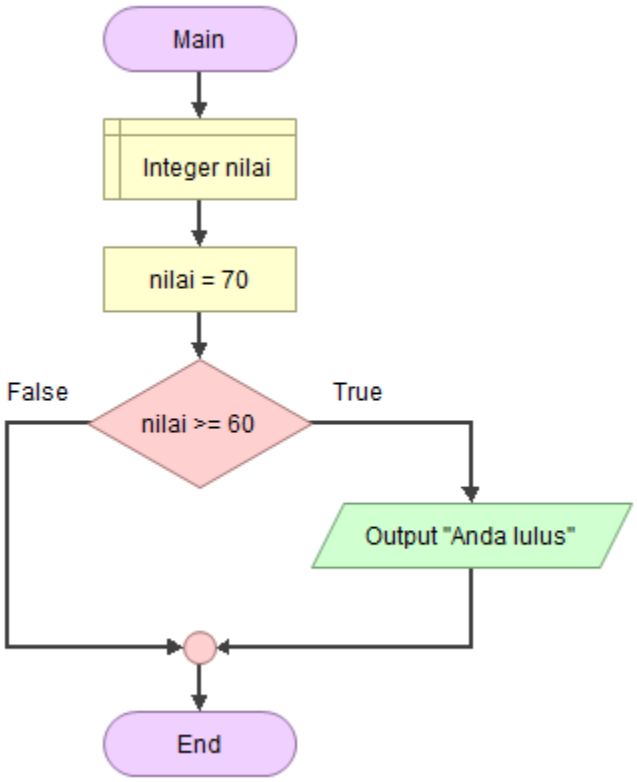
Kondisi If

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 `True`.

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

Dibawah ini adalah contoh penggunaan kondisi if pada Python



Kode dalam Bahasa Python

```
nilai = 70

if nilai >= 60:

    print("Anda lulus")
```

```
#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
if(nilai > 10):
    print("Selamat Anda Lulus")
```

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.

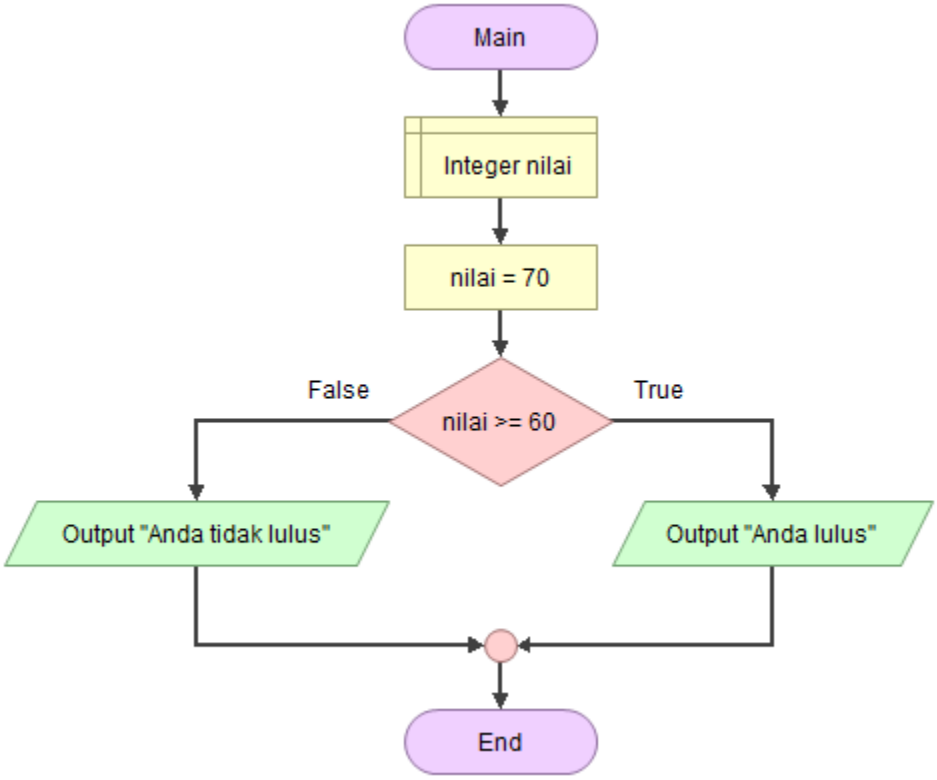
Kondisi If Else

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




```
nilai = 70
if nilai >= 60:
    print("Anda lulus")
else:
    print("Anda tidak lulus")
```

Source code dalam Bahasa Python :

```
nilai = 70
if nilai >= 60:
    print("Anda lulus")
else:
    print("Anda tidak lulus")
```

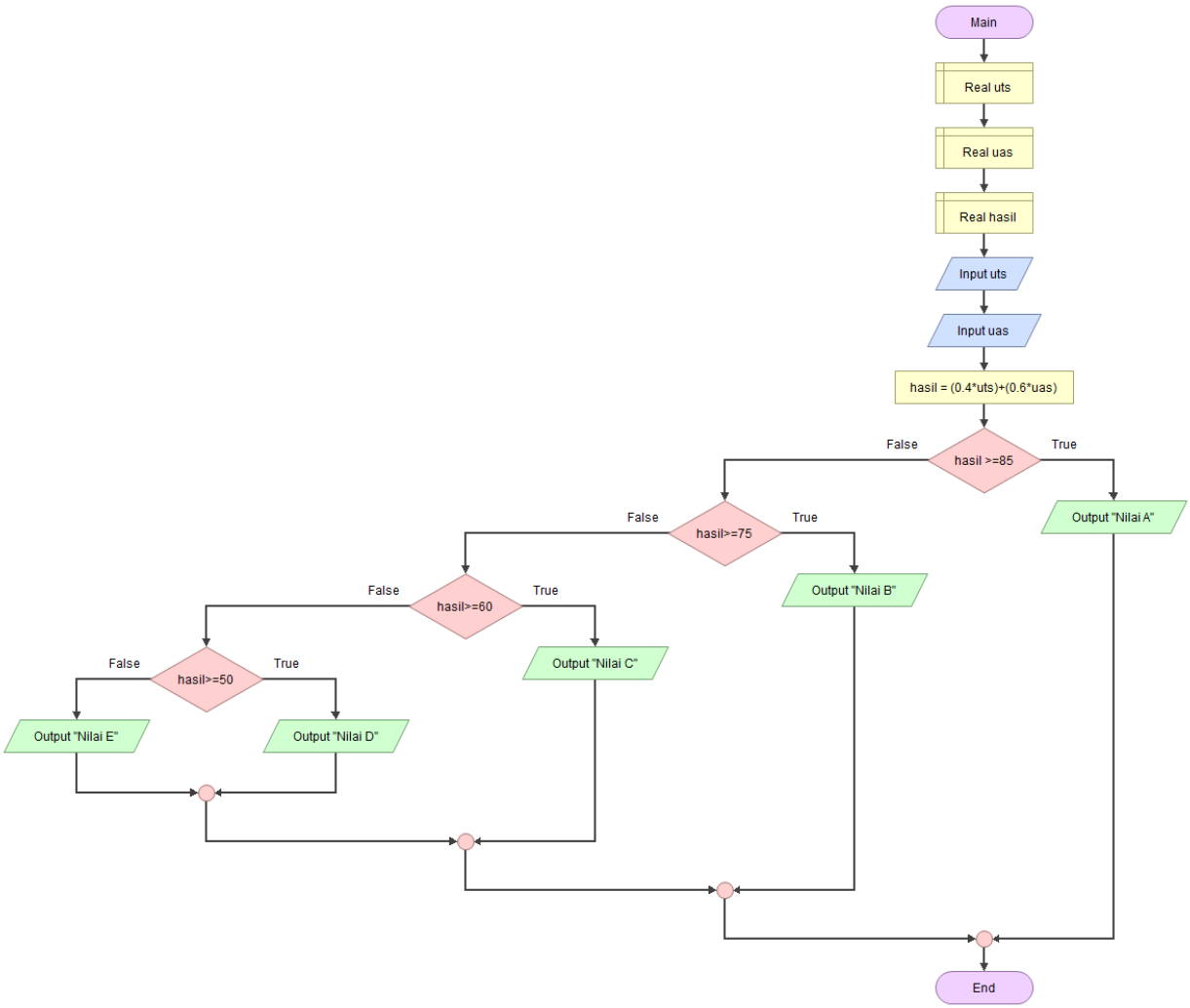
```
#Kondisi if else adalah jika kondisi bernilai TRUE maka akan dieksekusi pada if,
tetapi jika bernilai FALSE maka akan dieksekusi kode pada else

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:
    print("Maaf Anda Tidak Lulus")
```

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

Kondisi 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




```
uts = float(input())
uas = float(input())
hasil = 0.4 * uts + 0.6 * uas
if hasil >= 85:
    print("Nilai A")
else:
    if hasil >= 75:
        print("Nilai B")
    else:
        if hasil >= 60:
            print("Nilai C")
        else:
            if hasil >= 50:
                print("Nilai D")
            else:
                print("Nilai E")
```

Kode dalam Bahasa Python :

```
uts = float(input())
uas = float(input())
hasil = 0.4 * uts + 0.6 * uas
if hasil >= 85:
    print("Nilai A")
else:
    if hasil >= 75:
        print("Nilai B")
    else:
        if hasil >= 60:
            print("Nilai C")
        else:
            if hasil >= 50:
                print("Nilai D")
            else:
                print("Nilai E")
```

```
uts = float(input())

uas = float(input())

hasil = 0.4 * uts + 0.6 * uas

if hasil >= 85:

    print("Nilai A")

elif (hasil >= 75):

    print("Nilai B")

elif (hasil >= 60):

    print("Nilai C")

elif (hasil >= 50):
```

```
    print("Nilai D")
else:
    print("Nilai E")
```

Operator Python

Loop Python

Secara umum, pernyataan pada bahasa pemrograman akan dieksekusi secara berurutan. Pernyataan pertama dalam sebuah fungsi dijalankan pertama, diikuti oleh yang kedua, dan seterusnya. Tetapi akan ada situasi dimana Anda harus menulis banyak kode, dimana kode tersebut sangat banyak. Jika dilakukan secara manual maka Anda hanya akan membuang-buang tenaga dengan menulis beratus-ratus bahkan beribu-ribu kode. Untuk itu Anda perlu menggunakan pengulangan di dalam bahasa pemrograman Python.

Di dalam bahasa pemrograman Python pengulangan dibagi menjadi 3 bagian, yaitu :

- While Loop
- For Loop
- Nested 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
#Catatan: Penentuan ruang lingkup di Python bisa menggunakan tab alih-alih
menggunakan tanda kurung

count = 0
while (count < 9):
    print ("The count is: ", count)
    count = count + 1

print ("Good bye!")
```

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:
    print ("Saya suka makan", makanan)


contoh kasus menghitung RMSE
aktual=[1866.30,1866.30,1882.30,1903.20,1895.50]
prediksi=[1879.8,1879.8,1879.8,1962.5,1959.05]
rmse=0
jml=0
n=5
#proses menjumlah (sigma)
for i in range(5):
    jml=jml+((aktual[i]-prediksi[i])**2/n)
rmse=math.sqrt(jml)
print('RMSE=',rmse)




contoh kasus knn
rumah=['A','B','C','D','E','F','G','H','I','J']
lat=[11,15,19,18,16,23,25,21,23,29]
long=[26,29,28,30,26,25,22,24,25,24]
jarakkex=[]
jarakmin=1000
import math
for i in range(10):
    jarak=math.sqrt((19-lat[i])**2+(25-long[i])**2)

    jarakkex.append(jarak)
    print('data ke-',i)
    print('jarak ke x',jarakkex[i])
    if(jarakmin>jarak):
        jarakmin=jarak
print('jarak terpendek',jarakmin)


Contoh kasus regresi
X=[43,21,25,42,57,59]
Y=[99,65,79,75,87,81]


sigmay=0.0
```

```
sigmax=0.0
sigmaxx=0.0
sigmaxy=0.0
a=0.0
n=6
id=range(6)
for i in id:
  sigmax=sigmax+X[i]
  sigmaxx=sigmaxx+X[i]**2
  sigmay=sigmay+Y[i]
  sigmaxy=sigmaxy+X[i]*Y[i]
```

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
#Catatan: Penggunaan modulo pada kondisional mengasumsikan nilai selain nol sebagai
True(benar) dan nol sebagai False(salah)

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!")
```

Number Python

Number adalah tipe data Python yang menyimpan nilai numerik. Number adalah tipe data yang tidak berubah. Ini berarti, mengubah nilai dari sejumlah tipe data akan menghasilkan objek yang baru dialokasikan.

Objek Number dibuat saat Anda memberikan nilai pada-nya. Sebagai contoh : `angkaPertama = 1` `angkaKedua = 33`

Python mendukung beberapa tipe data Number diantaranya :

- Int
- Float
- Complex

Berikut ini adalah beberapa contoh dari Tipe data Number pada Python :

Int	Float	Complex
20	0.1	3.14j
300	1.20	35.j
-13	-41.2	3.12e-12j
020	32.23+e123	.873j

Int	Float	Complex
-0103	-92.	-.123+0J
-0x212	-32.52e10	3e+123J
0x56	60.2-E13	4.31e-4j

Konversi Tipe Data Number Python

Pada Python Anda bisa mengkonversi tipe data dengan menggunakan fungsi. Dibawah ini adalah beberapa fungsi untuk mengkonversi tipe data number Python.

- `int(x)` untuk meng-konversi x menjadi plain integer.
- `long(x)` untuk meng-konversi x menjadi long integer.
- `float(x)` untuk meng-konversi x menjadi floating point number.
- `complex(x)` untuk meng-konversi x menjadi complex number dengna real part x dan imaginary part zero.
- `complex(x, y)` untuk meng-konversi x dan y menjadi complex number dengan real part x dan imaginary part y. x dan numeric expressions y.

Fungsi Matematika Python

Pada bahasa pemrograman Python terdapat fungsi untuk melakukan perhitungan matematis, berikut adalah daftarnya :

Nama	Penggunaan	Penjelasan
Absolute	`abs(x)`	Nilai absolut dari x:(positive) jarak antara x and 0.

Nama	Penggunaan	Penjelasan
Ceiling	`ceil(x)`	Ceiling dari x: integer terkecil yang kurang dari x.
Cmp	`cmp(x, y)`	-1 if x < y, 0 if x == y, or 1 if x > y. Tidak berlaku lagi dengan Python 3. Sebaliknya gunakan return (x>y)-(x
Eksponen	`exp(x)`	Nilai eksponen dari x: ex
Fabs	`fabs(x)`	Nilai absolut dari x.
Floor	`floor(x)`	The floor of x: the largest integer not greater than x.
Floor	`floor(x)`	Nilai dasar dari x: internet terbesar tidak lebih besar dari x.
Log	`log(x)`	Logaritma dari x, untuk x > 0.
Log 10	`log10(x)`	Basis 10 logaritma dari x, untuk x > 0.
Max	`max(x1, x2,...)`	Argumen terbesar: Nilai terdekat dengan tak terhingga positif

Nama	Penggunaan	Penjelasan
Min	`min(x1, x2,...)`	Argumen terkecil: nilai yang paling mendekati tak berhingga negatif.
Modf	`modf(x)`	Bagian pecahan dan bilangan bulat dari x dalam tupel dua item. Kedua bagian memiliki tanda yang sama dengan x. Bagian integer dikembalikan sebagai float.
Pow	`pow(x, y)`	Nilai x ** y.
Round	`round(x [,n])`	X dibulatkan menjadi n digit dari titik desimal. Putaran Python jauh dari nol sebagai tie-breaker: round (0.5) adalah 1.0 dan round (-0.5) adalah -1.0.
Akar Kuadrat	`sqrt(x)`	Akar kuadrat x untuk x> 0.

Fungsi Nomor Acak Python

Nomor acak digunakan untuk aplikasi permainan, simulasi, pengujian, keamanan, dan privasi. Python mencakup fungsi berikut yang umum digunakan. Berikut adalah daftarnya :

Nama	Penggunaan	Penjelasan
Choice	`choice(seq)`	Item acak dari list, tuple, atau string.

Nama	Penggunaan	Penjelasan
RandRange	`randrange ([start,] stop [,step])`	Elemen yang dipilih secara acak dari jangkauan (start, stop, step).
Random	`random()`	A random float r, sehingga 0 kurang dari atau sama dengan r dan r kurang dari 1
Seed	`seed([x])`	Menetapkan nilai awal integer yang digunakan dalam menghasilkan bilangan acak. Panggil fungsi ini sebelum memanggil fungsi modul acak lainnya. Tidak ada pengembalian
Shuffle	`shuffle(lst)`	Mengacak daftar dari daftar di tempat. Tidak ada pengembalian
Floor	`floor(x)`	The floor of x: the largest integer not greater than x.
Uniform	`uniform(x, y)`	Sebuah float acak r, sedemikian rupa sehingga x kurang dari atau sama dengan r dan r kurang dari y.

Fungsi Trigonometri Python

Python mencakup fungsi berikut yang melakukan perhitungan trigonometri. Berikut adalah daftarnya :

Nama	Penggunaan Penjelasan	
Acos	`acos(x)`	Kembalikan kosinus x, di radian.
Asin	`asin(x)`	Kembalikan busur sinus x, dalam radian.
Atan	`atan(x)`	Kembalikan busur singgung x, di radian.
Atan 2	`atan2(y, x)`	Kembali atan (y / x), di radian.
Kosinus	`cos(x)`	Kembalikan kosinus x radian.
Hypot	`hypot(x, y)`	Kembalikan norma Euclidean, sqrt (x * x + y * y).
Sin	`sin(x)`	Kembalikan sinus dari x radian.
Tan	`tan(x)`	Kembalikan tangen x radian.
Derajat	`degrees(x)`	Mengonversi sudut x dari radian ke derajat.
Radian	`radians(x)`	Mengonversi sudut x dari derajat ke radian.

Konstanta Matematika Python

Modul ini juga mendefinisikan dua konstanta matematika. Berikut adalah daftarnya :

Nama	Penggunaan	Penjelasan
Pi	`pi`	Konstanta Pi matematika
e	`e`	Konstanta e matematika

String Python

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")
```

Mengakses Nilai dalam String

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])
print ("message[1:4]: ", message[1:4])
```

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
```

Escape Characters / Karakter Escape Python

Dibawah ini adalah tabel dari daftar karakter escape atau karakter non-printable yang dapat diwakili/ditulis dengan awalan notasi backslash.

Notasi Backslash	Karakter Hexadecimal	Penjelasan
\a	0x07	Bell atau alert
\b	0x08	Backspace
\cx		Control-x

Notasi Backslash	Karakter Hexadecimal	Penjelasan
\C-x		Control-x
\e	0x1b	Escape
\f	0x0c	Formfeed
\M-\C-x		Meta-Control-x
\n	0x0a	Newline
\nnn		Octal notation, dimana n berada di range 0.7
\r	0x0d	Carriage return
\s	0x20	Space
\t	0x09	Tab
\v	0x0b	Vertical tab

Notasi Backslash	Karakter Hexadecimal	Penjelasan
\x		Character x
\xnn		Notasi Hexadecimal, dimana n berada di range 0.9, a.f, atau A.F

Operator Spesial String Python

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

Operator	Contoh Penjelasan	
[:]	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 Python

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’)

Operator	Penjelasan
%f	Bilangan real floating point
%g	Yang lebih pendek dari% f dan% e
%G	Lebih pendek dari% f dan% E

Triple Quote Python

Python triple quotes digunakan dengan membiarkan string untuk ditulis dalam beberapa baris, termasuk kata kerja NEWLINEs, TABs, dan karakter khusus lainnya. Sintaks untuk triple quotes terdiri dari tiga tanda kutip tunggal atau ganda ditulis berturut-turut : Berikut adalah contohnya :

```
kutipantiga = """this is a long string that is made up of
several lines and non-printable characters such as
TAB ( \t ) and they will show up that way when displayed.
NEWLINEs within the string, whether explicitly given like
this within the brackets [ \n ], or just a NEWLINE within
the variable assignment will also show up.
"""
print (kutipantiga)
```

String Unicode Python

Pada Python 3, semua string diwakili dalam Unicode. Sedangkan pada Python 2 disimpan secara internal sebagai 8-bit ASCII, maka diperlukanlampiran ‘u’ untuk membuatnya menjadi Unicode. Tetapi hal ini tidak lagi diperlukan sekarang. :

Metode String Built-in

Python menyertakan metode built-in berikut untuk memanipulasi string

Metode	Penjelasan
`capitalize()`	Meng-kapitalkan huruf pertama string
`center(width, fillchar)`	Mengembalikan string yang dilapisi dengan fillchar dengan string asli yang dipusatkan pada total width kolom.
`count(str, beg = 0,end = len(string))`	Menghitung berapa kali str yang terjadi dalam string atau 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 pengkodean string 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 (jika memulai 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.

Metode	Penjelasan
`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 menimbulkan pengecualian jika str tidak ditemukan.
`isalnum()`	Mengembalikan true jika string memiliki minimal 1 karakter dan semua karakternya alfanumerik dan false sebaliknya.
`isalpha()`	Mengembalikan true jika string memiliki minimal 1 karakter dan semua karakter adalah abjad dan false sebaliknya.
`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.

Metode	Penjelasan
`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.

Metode	Penjelasan
`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).

Metode	Penjelasan
`isdecimal()`	Mengembalikan nilai true jika string unicode hanya berisi karakter desimal dan false sebaliknya.