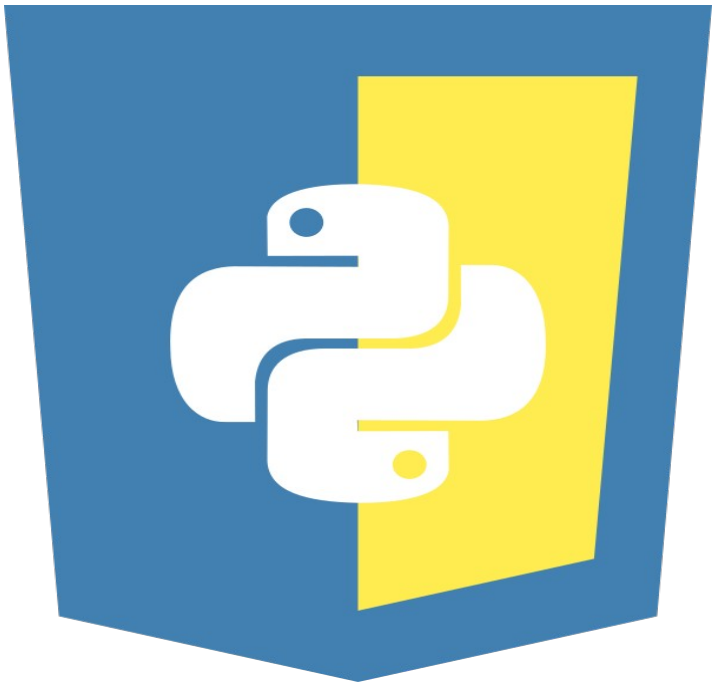
BUKU PANDUAN
PEMROGRAMAN PYTHON



Pemerintah Kabupaten Pemalang

Dinas Pemberdayaan Masyarakat Dan Pemerintahan Desa

Program Pusat Pemberdayaan Informatika Dan Desa

Pengantar

Terima kasih untuk Allah SWT atas semua Rahmat-Mu, Anugerah-Mu yang tak ternilai, hingga kami dapat menyelesaikan Buku “Pemrograman Komputer Dasar Python untuk Desa”. Buku ini merupakan panduan dasar yang membahas tentang pemrograman komputer python sebagai pedoman masyarakat desa untuk penerapan teknologi informasi di desa.

Terima kasih yang sebesar-besarnya disampaikan kepada H. Junaedi, SH,MM selaku Bupati Kabupaten Pemalang (Periode 2016-2021) dan Drs. H. Martono selaku Wakil Bupati Kabupaten Pemalang (Periode 2016-2021). Terima kasih juga disampaikan kepada Drs. Rifqi Jaya, MM selaku Kepala Dinas PERMASDES Kabupaten Pemalang (tahun 2017) beserta sekretaris dan staf. Terima kasih juga disampaikan untuk Bejo Setya Widodo, ST selaku KASI SID Dinas PERMASDES (tahun 2017) dan APD “Aparatur Pemerintah Desa” Kabupaten Pemalang yang tidak bisa saya sebutkan satu persatu, serta semua pihak yang telah ikut membantu dalam penyelesaian buku ini.

Kami menyadari masih terdapat kekurangan dalam buku ini untuk itu kritik dan saran terhadap penyempurnaan buku ini sangat diharapkan. Semoga buku ini dapat memberi manfaat bagi masyarakat khususnya dan bagi semua pihak yang membutuhkan.

Salam Hokya

Penulis

Daftar Isi

BAB 1 Pengenalan 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("Python sangat simpel")
```

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 ;

BAB 2 Instalasi Python

Sebelum Anda menggunakan Python, Anda harus menginstalnya terlebih dahulu di sistem operasi komputer Anda. Saat ini Python memiliki 2 versi yang berbeda, yaitu Python versi 3.4.3 dan Python versi 2.7.10. Disini kita akan belajar bahasa pemrograman Python menggunakan versi terbaru 3.4.3

Cara menginstal python sangat mudah, ikuti panduan dibawah ini. Dibawah adalah panduan cara instal python di platform Linux, Windows dan Mac OS.

2.1 Linux

- Buka browser, kunjungi http://www.python.org/downloads/source/
- Download versi terbaru Python berbentuk file zip untuk Unix/Linux
- Ekstrak file zip yang baru saja di download
- Edit file Modules/Setup jika Anda ingin kostumisasi Python
- Jalankan ./configure script
- make
- make install

  Langkah ini akan menginstal Python di lokasi standar /usr/local/bin dan library di /usr/local/lib/pythonXX dimana XX adalah versi terbaru Python yang anda gunakan.

2.2 Windows

- Buka browser, kunjungi http://www.python.org/downloads/windows/
- **ATAU**, klik direct link https://www.python.org/ftp/python/3.4.3/python-3.4.3.msi
- Buka (klik 2x) file installer python yang baru saja di download
- Ikuti langkah instalasi sampai selesai

2.3 Mac OS

- Buka browser, kunjungi http://www.python.org/download/mac/
- Download versi terbaru Python untuk Macintosh
- Buka file yang baru saja di download
- Ikuti langkah instalasi sampai selesai

BAB 3 Menjalankan Python

Untuk menjalankan Python ada banyak cara yang bisa dilakukan. Anda bisa menggunakan sheel, terminal atau menggunakan IDE (Integrated Development Environment). Di bawah ini adalah langkah-langkah menjalankan Python dengan cara yang paling mudah.

3.1 Linux

- Buka terminal **(Ctrl + Alt + T)**
- Ketik **python** maka Anda akan masuk ke sheel Python.
- Tuliskan script Python Anda, contoh: **print("Selamat datang di Python")**. jika sudah tekan tombol **Enter**, dan script Python akan dijalankan/eksekusi.
- Untuk keluar dari sheel Python ketik **exit()**
  atau
- Gunakan teks editor, misalnya gedit.
- Buat file baru, dan ketikan script python Anda, contoh: **print("Selamat datang di Python")**.
- Save As dengan ekstensi **.py** (contoh: cetak.py).
- Jalankan file dengan menggunakan Terminal.
- Buka terminal **(Ctrl + Alt + T)**.
- Masuk ke direktori dimana file Python Anda disimpan (contoh: **cd /Users/admin/ Desktop/**).
- Jalankan script Python dengan menggunakan **python** diikuti dengan nama file (contoh: **python cetak.py**).
- Script Python Anda akan dieksekusi/dijalankan.

3.2 Windows

- Buka Python sheel, Anda bisa mencarinya di tombol Start.
- Tuliskan script Python Anda, contoh: **print("Selamat datang di Python")**. jika sudah tekan tombol **Enter**, dan script Python akan dijalankan/eksekusi.
- Untuk keluar dari sheel Python ketik **exit()**

3.3 Macintosh

- Buka terminal.
- Ketik **python** maka Anda akan masuk ke sheel Python.
- Tuliskan script Python Anda, contoh: **print("Selamat datang di Python")**. jika sudah tekan tombol **Enter**, dan script Python akan dijalankan/eksekusi.
- Untuk keluar dari sheel Python ketik **exit()** atau
- Gunakan teks editor.

- Buat file baru, dan ketikan script python Anda, contoh: **print("Selamat datang di Python")**.
- Save As dengan ekstensi **.py** (contoh: cetak.py).
- Jalankan file dengan menggunakan Terminal.
- Buka terminal **(Ctrl + Alt + T)**.
- Masuk ke direktori dimana file Python Anda disimpan (contoh: **cd /Users/admin/Desktop/**).
- Jalankan script Python dengan menggunakan **python** diikuti dengan nama file (contoh: **python cetak.py**).
- Script Python Anda akan dieksekusi/dijalankan.

BAB 4 Integrated Development Environment (IDE) Python

IDE adalah sebuah software aplikasi yang memberikan Anda fasilitas bermanfaat ketika membuat program. Biasanya sebuah IDE terdiri dari source code editor build automation tools dan debugger.

Untuk menulis sebuah program, bisa menggunakan text editor atau IDE nya. Bagi yang sudah mahir, menulis program dengan text editor bukanlah menjadi masalah. Tetapi untuk pemula, akan lebih mudah menggunakan IDE.

IDE untuk Python sangatlah banyak, tersedia bermacam-macam IDE dengan kelebihan dan kekurangan masing-masing.

Beberapa IDE untuk Python yang cukup populer adalah :

- Komodo
- LiClipse
- NetBeans
- PyCharm
- Kdevelop
- PyDev
- Wing IDE
- dan masih banyak lainnya (http://wiki.python.org/moin/IntegratedDevelopmentEnvironments).

BAB 5 Hello World Python

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

5.1 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**

5.2 Python Case Sensitive

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 lainya.

5.3 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.

```
#Ini adalah komentar
# Tulisan ini tidak akan dieksekusi
#komentar dengan tanda pagar hanya bisa digunakan
#untuk
#satu
#baris
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.

5.4 Tipe Data pada Python

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

Python sendiri mempunyai tipe data yang cukup unik bila kita bandingkan dengan bahasa pemrograman yang lain.

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
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
```

5.5 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 natinya 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)
#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)
```

5.6 Operator

Operator adalah konstruksi yang dapan 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 Identisas (Identity Operators)

Mari kita membahasnya satu-persatu.

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

```
#file /python_dasar/operator_aritmatika.py
#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.

Assignment Operator

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.

Logical Operator

Operator	Contoh	Penjelasan
and	a, b = True, True # hasil akan True print a and b	Jika kedua operan bernilai True, maka kondisi akan bernilai True. Selain kondisi tadi maka akan bernilai False.
or	a, b = True, False # hasil akan True print a or b print b or a print a or a # hasil akan False print b or b	Jika salah satu atau kedua operan bernilai True maka kondisi akan bernilai True. Jika keduanya False maka kondisi akan bernilai False.
not	a, b = True, False # hasil akan True print not a print not b	Membalikkan nilai kebeneran pada operan misal jika asalnya True akan menjadi False dan begitupun sebaliknya.

Bitwise Operator

Operator	Contoh	Penjelasan

&	a, b = 13, 37
	# a akan bernilai '0000 1101'
	# b akan bernilai '0010 0101'
	c = a & b
	# c akan bernilai 5 = '0000 0101'
	print c
		Operator biner AND, memeriksa apakah operan di sebelah kiri dan operan sebelah kanan mempunyai angka biner 1 di setiap bit. Jika keduanya bernilai 1 maka bit hasil operasi akan bernilai 1.

|	a, b = 13, 37
	# a akan bernilai '0000 1101'
	# b akan bernilai '0010 0101'
	c = a | b
		Operator biner OR, memeriksa apakah operan di sebelah kiri dan operan sebelah kanan mempunyai angka biner 1 di setiap bit. Jika salah

Operator	Contoh	Penjelasan
	# c akan bernilai 45 = '0010 1101' print c	satunya bernilai 1 maka bit hasil operasi akan bernilai 1.

^	a, b = 13, 37
	# a akan bernilai '0000 1101'
	# b akan bernilai '0010 0101'
	c = a ^ b
	# c akan bernilai 40 = '0010 1000'
	print c
		Operator biner XOR, memeriksa apakah operan di sebelah kiri dan operan sebelah kanan mempunyai angka biner 1 di setiap bit. Jika keduanya bernilai 1 maka bit hasil operasi akan bernilai 0.

Kali sama dengan *=	`a *= 2`	Operator biner Negative, membalik nilai bit. Misal dari 1 menjadi 0, dari 0 menjadi 1.
~	a, b = 13, 37 # a akan bernilai '0000 1101' # b akan bernilai '0010 0101'	Memberikan nilai variabel dengan nilai variabel itu sendiri dibagi dengan nilai di sebelah kanan.
<<	a, b = 13, 37 # a akan bernilai '0000 1101' # b akan bernilai '0010 0101' # hasil bernilai 52 = "0011 0100' print a << 2 # hasil bernilai 148 = '1001 0100' print b << 2	Memberikan nilai variabel dengan nilai variabel itu sendiri dibagi dengan nilai di sebelah kanan. Yang diambil nantinya adalah sisa baginya.
>>	a, b = 13, 37 # a akan bernilai '0000 1101' # b akan bernilai '0010 0101' # hasil bernilai 3 = '0000 0011' print a >> 2 # hasil bernilai 9 = '0000 1001' print b >> 2	Operator penggeser biner ke kiri, deret bit akan digeser ke kiri sebanyak n kali.

Membership Operator

Operator	Contoh	Penjelasan
in	sebuah_list = [1, 2, 3,4 ,5] print 5 in sebuah_list	Memeriksa apakah nilai yang dicari berada pada list atau struktur data python lainnya. Jika nilai tersebut ada maka kondisi akan bernilai True.
not in	sebuah_list = [1, 2, 3,4 ,5] print 10 not in sebuah_list	Memeriksa apakah nilai yang dicari tidak ada pada list atau struktur data python lainnya. Jika nilai tersebut tidak ada maka kondisi akan bernilai True.

Identity Operator

Operator	Contoh	Penjelasan
is	a, b = 10, 10 # hasil akan True print a is b	Memeriksa apakah nilai di sebelah kiri operan memiliki identitas memori yang sama dengan nilai di sebelah kanan operan. Jika sama maka kondisi bernilai True.
is not	a, b = 10, 5 # hasil akan True print a is not b	Memeriksa apakah nilai di sebelah kiri operan memiliki identitas memori yang berbeda dengan nilai di sebelah kanan operan. Jika berbeda maka kondisi bernilai True.

5.7 Konfisi 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.

Dibawah ini adalah contoh penggunaan kondisi if pada Python

```
#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.

5.8 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:
    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

5.9 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

```
#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"):
    print("Saya akan kuliah")
elif(hari_ini == "Minggu"):
    print("Saya akan libur")
```

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

5.10 Pengulangan “Loop”

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

Pengulangan While

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
print ("Good bye!")
```

5.11 Pengulangan For

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

Dibawah ini adalah contoh penggunaan pengulangan While 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)
```

5.12 Pengulangan Bersarang (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!"
```

5.13 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
-0103	-92.	-.123+0J

Int	Float	Complex
-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.
- ong(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

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.
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

Nama	Penggunaan	Penjelasan
Fabs	`fabs(x)`	Nilai absolut dari 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
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

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.
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

Nama	Penggunaan	Penjelasan
		kurang dari atau sama dengan r dan r kurang dari y.

Fungsi Trigonometri

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

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

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

5.14 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])
print ("message[1:4]: ", messange[1:3])
```

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 Character

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
\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

Notasi Backslash	Karakter Hexadecimal	Penjelasan
\t	**0x09**	Tab
\v	**0x0b**	Vertical tab
\x		Character x
\xnn	**asdafsdfsdf**	Notasi Hexadecimal, dimana n berada di range 0.9, a.f, atau A.F

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

Triple Quote

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

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.
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.

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.

5.15 List

Dalam bahasa pemrograman Python, struktur data yang paling dasar adalah urutan atau lists. Setiap elemen-elemen berurutan akan diberi nomor posisi atau indeksnya. Indeks pertama dalam list adalah nol, indeks kedua adalah satu dan seterusnya.

Python memiliki enam jenis urutan built-in, namun yang paling umum adalah list dan tuple. Ada beberapa hal yang dapat Anda lakukan dengan semua jenis list. Operasi ini meliputi pengindeksan, pengiris, penambahan, perbanyak, dan pengecekan keanggotaan. Selain itu, Python memiliki fungsi built-in untuk menemukan panjang list dan untuk menemukan elemen terbesar dan terkecilnya.

Membuat List Python

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 ]
list3 = ["a", "b", "c", "d"]
```

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 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.

Python Function	Penjelasan
`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

5.16 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])
```

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

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
`cmp(tuple1, tuple2)`	# Tidak lagi tersedia dengan Python 3
`len(tuple)`	Memberikan total panjang tuple.
`max(tuple)`	Mengembalikan item dari tuple dengan nilai maks.
`min(tuple)`	Mengembalikan item dari tuple dengan nilai min.

Python Function	Penjelasan
`tuple(seq)`	Mengubah tuple menjadi tuple.

5.17 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'])
```

5.18 Tanggal dan Jam

Program Python yang dapat menangani tanggal dan waktu dalam beberapa cara. Mengkonversi antara format tanggal adalah tugas umum untuk komputer. Modul Python's waktu dan kalender membantu melacak tanggal dan waktu.

Interval waktu adalah angka floating-point dalam satuan detik. Instants tertentu dalam waktu dinyatakan dalam satu detik sejak 12:00 am, 1 Januari 1970(epoch).

Ini adalah waktu yang populer modul yang tersedia di Python yang menyediakan fungsi untuk bekerja dengan waktu, dan untuk mengkonversi antara pernyataan. Fungsi time.time() mengembalikan sistem saat ini waktu sejak 12:00 am, 1 Januari 1970.

```
import time;  # harus menginclude modul time

ticks = time.time()
print "Number of ticks since 12:00am, January 1, 1970:", ticks
```

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

```
Number of ticks since 12:00am, January 1, 1970: 7186862.73399
```

TimeTupple

Index	Penjelasan	Hasil
0	4-digit year	2008
1	Month	1 to 12
2	Day	1 to 31
3	Hour	0 to 23
4	Minute	0 to 59
5	Second	0 to 61 (60 or 61 are leap-seconds)
6	Day of Week	0 to 6 (0 is Monday)
7	Day of year	1 to 366 (Julian day)
8	Daylight savings	-1, 0, 1, -1 means library determines DST

5.19 Fungsi

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.

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

Contoh fungsi

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

5.20 Modul

Modul memungkinkan Anda mengatur kode Python secara logis. Mengelompokkan kode terkait ke dalam modul membuat kode lebih mudah dipahami dan digunakan. Modul adalah objek Python dengan atribut yang diberi nama yang bisa Anda bind dan dijadikan referensi.

Secara sederhana modul adalah file yang terdiri dari kode Python. Modul dapat mendefinisikan fungsi, kelas dan variabel. Modul juga bisa menyertakan kode yang bisa dijalankan "runable".

Berikut adalah contoh modul sederhana pada Python :

```
def print_func( par ):
    print "Halo : ", par
    return
```

Import Statement

Anda dapat menggunakan file sumber Python apapun sebagai modul dengan mengeksekusi pernyataan impor di file sumber Python lainnya. Impornya memiliki sintaks berikut.

Ketika interpreter menemukan sebuah pernyataan import, ia mengimpor modul jika modul tersebut ada di jalur pencarian. Jalur pencarian adalah daftar direktori yang

ditafsirkan juru bahasa sebelum mengimpor modul. Misalnya, untuk mengimpor modul hello.py, Anda perlu meletakkan perintah berikut di bagian atas script.

```
# Import module support
import support
# Anda bisa memanggil fungsi defined sebagai berikut
support.print_func("Bara")
```

5.21 Membaca Input Keyboard

Python 2 memiliki dua fungsi built-in untuk membaca data dari input standar, yang secara default berasal dari keyboard. Fungsi ini adalah **input()** dan **raw_input()**

Dengan Python 3, fungsi **raw_input()** tidak digunakan lagi. Selain itu,**input()** berfungsi membaca data dari keyboard sebagai string, terlepas dari apakah itu tertutup dengan tanda kutip (" atau "') atau tidak.

Input Python

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

5.22 Exception

Python menyediakan dua fitur yang sangat penting untuk menangani kesalahan tak terduga dalam program Python Anda dan menambahkan kemampuan debugging di dalamnya.

- **Exception Handling**
- **Assertions**

Exception adalah sebuah peristiwa, yang terjadi selama pelaksanaan program yang mengganggu aliran normal instruksi program. Secara umum, ketika skrip Python menemukan situasi yang tidak dapat diatasi, hal itu menimbulkan pengecualian. Exception adalah objek Python yang mewakili kesalahan.

Ketika skrip Python menimbulkan Exception, ia harus menangani Exception begitu saja sehingga berhenti dan berhenti.

Standard Exceptions

Nama	Penjelasan
Exception	Kelas dasar untuk semua pengecualian / exception
StopIteration	Dibesarkan ketika metode (iterator) berikutnya dari iterator tidak mengarah ke objek apa pun.
SystemExit	Dibesarkan oleh fungsi sys.exit ().
StandardError	Kelas dasar untuk semua pengecualian built-in kecuali StopIteration dan SystemExit.
ArithmeticError	Kelas dasar untuk semua kesalahan yang terjadi untuk perhitungan numerik.
OverflowError	Dibesarkan saat perhitungan melebihi batas maksimum untuk tipe numerik.
FloatingPointError	Dibesarkan saat perhitungan floating point gagal.
ZeroDivisonError	Dibesarkan saat pembagian atau modulo nol dilakukan untuk semua tipe numerik.
AssertionError	Dibesarkan jika terjadi kegagalan pernyataan Assert.
AttributeError	Dibesarkan jika terjadi kegagalan referensi atribut atau penugasan.
EOFError	Dibesarkan bila tidak ada input dari fungsi raw_input () atau input () dan akhir file tercapai.
ImportError	Dibesarkan saat sebuah pernyataan impor gagal.
KeyboardInterrupt	Dibesarkan saat pengguna menyela eksekusi program, biasanya dengan menekan Ctrl + c.
LookupError	Kelas dasar untuk semua kesalahan pencarian.
IndexError	Dibesarkan saat sebuah indeks tidak ditemukan secara berurutan.
KeyError	Dibesarkan saat kunci yang ditentukan tidak ditemukan dalam kamus.
NameError	Dibesarkan saat pengenal tidak ditemukan di namespace lokal atau global.
UnboundLocalError	Dibesarkan saat mencoba mengakses variabel lokal dalam suatu fungsi atau metode namun tidak ada nilai yang ditugaskan padanya.
EnvironmentError	Kelas dasar untuk semua pengecualian yang terjadi di luar lingkungan Python.
IOError	Dibesarkan saat operasi input / output gagal, seperti pernyataan cetak atau fungsi open () saat mencoba membuka file yang tidak ada.
OSError	Dibangkitkan untuk kesalahan terkait sistem operasi.
SyntaxError	Dibesarkan saat ada kesalahan dengan sintaks Python.
IndentationError	Dibesarkan saat indentasi tidak ditentukan dengan benar.

Nama	Penjelasan
SystemError	Dibesarkan saat penafsir menemukan masalah internal, namun bila kesalahan ini ditemui juru bahasa Python tidak keluar.
SystemExit	Dibesarkan saat juru bahasa Python berhenti dengan menggunakan fungsi sys.exit (). Jika tidak ditangani dalam kode, menyebabkan penafsir untuk keluar.
TypeError	Dibesarkan saat operasi atau fungsi dicoba yang tidak valid untuk tipe data yang ditentukan.
ValueError	Dibesarkan ketika fungsi bawaan untuk tipe data memiliki jenis argumen yang valid, namun argumen tersebut memiliki nilai yang tidak valid yang ditentukan.
RuntimeError	Dibesarkan saat kesalahan yang dihasilkan tidak termasuk dalam kategori apa pun.
NotImplementedError	Dibesarkan ketika metode abstrak yang perlu diimplementasikan di kelas warisan sebenarnya tidak dilaksanakan.