BUKU AJAR PEMROGRAMAN LANJUT
BAHASA PEMROGRAMAN PYTHON

Oleh:

ALFIAN MA'ARIF

PROGRAM STUDI TEKNIK ELEKTRO

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS AHMAD DAHLAN

YOGYAKARTA

2020

DAFTAR ISI

BAB PENGENALAN & MEMULAI PYTHON

Apa itu Python

Python adalah bahasa pemrograman yang populer saat ini. Bahasa Python dibuat oleh Guido van Rossum, dan dirilis pada tahun 1991. Python dapat digunakan untuk Pengembangan web (sisi server), Pengembangan perangkat lunak atau membuat aplikasi (software), Menyelesaikan persamaan Matematika, Pembuatan skrip sistem dan Pemrograman Mikrokontroller (Micro-Python)

Beberapa fungsi Bahasa Python adalah Python dapat digunakan di server untuk membuat aplikasi web, Python dapat digunakan bersama perangkat lunak untuk membuat alur kerja, Python dapat terhubung ke sistem database, Bahasa Python juga dapat membaca dan memodifikasi file, Python dapat digunakan untuk menangani data besar dan melakukan matematika yang kompleks, dan Python dapat digunakan untuk pembuatan prototipe dengan cepat, atau untuk pengembangan perangkat lunak siap produksi.

Alasan untuk menggunakan dan mempelajari Python adalah Python dapat bekerja pada platform yang berbeda (Windows, Mac, Linux, Raspberry Pi, dll), Python memiliki sintaks sederhana yang mirip dengan bahasa Inggris, Python memiliki sintaks yang memungkinkan pengembang untuk menulis program dengan lebih sedikit baris daripada beberapa bahasa pemrograman lainnya, Python berjalan pada sistem interpreter, artinya kode dapat dieksekusi segera setelah ditulis. Ini berarti pembuatan prototipe bisa sangat cepat, Python dapat diperlakukan dengan cara prosedural, cara berorientasi objek atau cara fungsional, dan Python memiliki banyak Pustaka

Python memiliki beberapa versi yaitu Python versi 2 dan Python versi 3. Versi utama terbaru dari Python adalah Python 3, yang akan kita gunakan dalam buku ini. Namun, Python 2, meskipun tidak diperbarui dengan apa pun selain pembaruan keamanan, masih cukup populer.

Dalam materi ini Python akan ditulis dalam editor teks. Programmer dapat menulis Python dalam Lingkungan Pengembangan Terintegrasi, seperti Jupyter, Thonny, Pycharm, Netbeans Code Visual, atau Eclipse yang sangat berguna saat mengelola koleksi file Python yang lebih besar.

Sintak Python dibandingkan dengan bahasa pemrograman lain memiliki beberapa kelebihan. Python dirancang agar mudah dibaca, dan memiliki beberapa kesamaan dengan bahasa Inggris dengan pengaruh dari matematika. Python menggunakan baris baru untuk menyelesaikan perintah, berbeda dengan bahasa pemrograman lain yang sering menggunakan titik koma atau

tanda kurung. Python mengandalkan indentasi, menggunakan spasi, untuk mendefinisikan ruang lingkup; seperti cakupan loop, fungsi, dan kelas. Bahasa pemrograman lain sering menggunakan tanda kurung kurawal untuk tujuan ini.

Contoh perbandingan Pemrograman C++, C#, Java dan Python untuk menampilkan tulisan "Hello World!". Terlihat bahwa bahasa pemrograman Python memiliki sintak terpendek, lebih sederhana, mudah, tidak perlu menggunakan titik koma sebagai akhiran program dan tidak perlu menggunakan kurung kurawal.

Pemrograman C# = 5 Baris Program

```
using System;
namespace HelloWorld {
  class Program {
    static void Main(string[] args) {
      Console.WriteLine("Hello World!");
    }
  }
}
```

Pemrograman C++ = 4 Baris

```
#include <iostream>
using namespace std;
int main() {
  cout << "Hello World!";
  return 0;
}
```

Pemrograman Java = 3 Baris

```
public class Main {
  public static void main(String[] args) {
    System.out.println("Hello World");
```

```
  }
}
```

Pemrograman Python = 1 Baris
`print("Hello, World!")`

Instalasi Python

Banyak komputer sudah menginstal python. Untuk memeriksa apakah Anda telah menginstal python pada PC Windows, cari di bilah mulai untuk Python atau jalankan perintah berikut di Command Line (cmd.exe):

```
C:\Users\Your Name>python --version
```

Untuk memeriksa apakah Anda telah menginstal python di Linux atau Mac, maka di linux buka baris perintah atau di Mac buka Terminal dan ketik:

```
python --version
```

Jika ternyata Anda tidak menginstal python di komputer Anda, Anda dapat mengunduhnya secara gratis dari situs web berikut: https://www.python.org/

Panduan Memulai Python

Python adalah bahasa pemrograman yang diinterpretasikan, artinya sebagai pengembang Anda menulis file Python (.py) di editor teks dan kemudian memasukkan file-file itu ke dalam interpreter python untuk dieksekusi. Cara menjalankan file python seperti ini pada baris perintah:

```
C:\Users\Your Name>python helloworld.py
```

Dengan "helloworld.py" adalah nama file python Anda.

Mari tulis file Python pertama kita, bernama helloworld.py, yang bisa dilakukan di editor teks manapun seperti notepad.

helloworld.py

```
print("Hello, World!")
```

Sederhana seperti itu. Simpan file Anda. Buka baris perintah Anda, arahkan ke direktori tempat Anda menyimpan file Anda, dan jalankan:

```
C:\Users\Your Name>python helloworld.py
```

Outputnya harusnya terbaca:

```
Hello, World!
```

Selamat, Anda telah menulis dan menjalankan program Python pertama Anda.

Baris Perintah Python

Untuk menguji sejumlah pendek kode di python terkadang cara tercepat dan termudah untuk tidak menulis kode dalam sebuah file. Ini dimungkinkan karena Python dapat dijalankan sebagai baris perintah itu sendiri.

Ketik perintah berikut di baris perintah Windows, Mac, atau Linux:

```
C:\Users\Your Name>python
```

Atau, jika perintah "python" tidak berfungsi, Anda dapat mencoba "py":

```
C:\Users\Your Name>py
```

Dari sana Anda dapat menulis python apa saja, termasuk contoh hello world kami dari tutorial sebelumnya:

```
>>> print("Hello, World!")
```

Yang akan menulis "Halo, Dunia!" di baris perintah:

```
Hello, World!
```

Setiap kali Anda selesai di baris perintah python, Anda cukup mengetik yang berikut ini untuk keluar dari antarmuka baris perintah python:

```
exit()
```

Mengeksekusi Bahasa Python

Seperti yang kita pelajari di halaman sebelumnya, sintak Python dapat dijalankan dengan menulis langsung di Command Line:

```
>>> print("Hello, World!")
Hello, World!
```

Atau dengan membuat file python di server, menggunakan ekstensi file .py, dan menjalankannya di Command Line:

```
C:\Users\Your Name>python myfile.py
```

Indentasi Python

Indentasi mengacu pada spasi di awal baris kode program. Dalam bahasa pemrograman lain paragraf yang menjorok dalam program hanya untuk dibaca, paragraf yang penjorok dalam Python sangat penting. Python menggunakan indentasi untuk menunjukkan blok kode.

```
if 5 > 2:
  print("Five is greater than two!")
```

Python akan error jika Anda melewati indentasi:

Contoh Sintak Error
`if 5 > 2:` `print("Five is greater than two!")`

Jumlah spasi terserah Anda sebagai programmer, tetapi setidaknya harus satu.

```
if 5 > 2:
 print("Five is greater than two!")
if 5 > 2:
        print("Five is greater than two!")
```

Anda harus menggunakan jumlah spasi yang sama dalam blok kode yang sama, jika tidak program Python akan error.

Sintak Error

```
if 5 > 2:
 print("Five is greater than two!")
        print("Five is greater than two!")
```

Variabel Python

Di Python, variabel dibuat saat Anda memberikan nilai pada variabel tersebut. Pada contoh berikut terdapat variabel x dan variabel y yang diberi nilai data angka dan data teks.

Variabel dalam Python:

```
x = 5
y = "Hello, World!"
```

Python tidak memiliki perintah untuk mendeklarasikan variabel.

Komentar Python

Python memiliki fitur komentar untuk tujuan dokumentasi dalam kode program. Komentar dimulai dengan tanda pagar (#), dan Python akan membuat sisa barisnya sebagai komentar. Kode komentar tidak akan dieksekusi oleh komputer, hanya berfungsi sebagai petunjuk, informasi atau memberi penjelasan tentang suatu bagian program.

Komentar pada Python

```
#Ini adalah komentar
print("Hello, World!")
```

Komentar dapat ditempatkan di akhir baris, dan Python akan mengabaikan sisa baris:

```
print("Hello, World!") #This is a comment
```

Komentar tidak harus berupa teks yang menjelaskan kode, itu juga dapat digunakan untuk mencegah Python mengeksekusi kode:

```
#print("Hello, World!")
print("Cheers, Mate!")
```

Komentar Multi Baris

Python tidak memiliki sintaks untuk komentar multi baris. Untuk menambahkan komentar multiline, Anda dapat memasukkan # untuk setiap baris:

```
#Ini adalah komentar
#ditulis dengan
#lebih dari satu baris
print("Hello, World!")
```

Atau, jika tidak seperti yang diharapkan, Anda dapat menggunakan string multiline. Python akan mengabaikan literal string yang tidak ditetapkan ke variabel, Anda dapat menambahkan string multiline (tanda kutip tiga) dalam kode Anda, dan menempatkan komentar Anda di dalamnya:

```
"""
Ini adalah komentar
ditulis dengan
lebih dari satu baris
"""
print("Hello, World!")
```

Selama string tidak ditetapkan ke variabel, Python akan membaca kodenya, tetapi kemudian mengabaikannya, dan Anda telah membuat komentar multiline.

BAB VARIABEL & TIPE DATA

Variabel

Variabel adalah wadah untuk menyimpan nilai data. Python tidak memiliki perintah untuk mendeklarasikan variabel. Variabel dibuat saat Anda pertama kali memberikan nilai padanya.

```
x = 5
y = "John"
print(x)
print(y)
```

Variabel tidak perlu dideklarasikan dengan tipe tertentu, dan bahkan bisa berubah tipe setelah disetel.now of type

```
x = 4       # x bertipe int
x = "Sally" # x sekarang bertipe str
print(x)
```

Casting

Jika Anda ingin menentukan tipe data variabel, ini dapat dilakukan dengan casting.

```
x = str(3)    # x akan bernilai '3'
y = int(3)    # y bernilai 3
z = float(3)  # z bernilai 3.0
```

Mendapatkan Tipe Data

Anda bisa mendapatkan tipe data variabel dengan fungsi type ().

```
x = 5
y = "John"
print(type(x))
print(type(y))
```

Kutipan Tunggal dan Ganda

Variabel string dapat dideklarasikan baik dengan menggunakan tanda kutip tunggal atau ganda:

```
x = "John"
# is the same as
x = 'John'
```

Case Sensitive

Nama variabel peka terhadap huruf besar dan kecil (case sensitive).

Perintah ini akan membuat dua variabel:

```
a = 4
A = "Sally"
#variabel A tidak akan mengganti variabel a
```

Nama Variabel

Variabel dapat memiliki nama pendek (seperti x dan y) atau nama yang lebih deskriptif (umur, namamobil, total_volume). Aturan untuk variabel Python adalah

1. Nama variabel harus dimulai dengan huruf atau garis bawah
2. Nama variabel tidak boleh dimulai dengan angka
3. Nama variabel hanya boleh berisi karakter alfanumerik dan garis bawah (A-z, 0-9, dan _)
4. Nama variabel peka huruf besar / kecil (usia, Usia, dan USIA adalah tiga variabel berbeda)

Contoh nama Variabel yang diperbolehkan:

```
myvar = "John"
my_var = "John"
_my_var = "John"
myVar = "John"
```

```
MYVAR = "John"
myvar2 = "John"
```

Contoh nama variabel yang tidak diperbolehkan

`2myvar = "John"`
`my-var = "John"`
`my var = "John"`

Ingatlah bahwa nama variabel sensitif terhadap huruf besar kecil.

Nama Variabel Multi Kata

Nama variabel dengan lebih dari satu kata bisa jadi sulit dibaca. Ada beberapa teknik yang bisa Anda gunakan untuk membuatnya lebih mudah dibaca:

Jenis Penamaan Variabel Camel

Setiap kata, kecuali yang pertama, dimulai dengan huruf kapital:

```
myVariableName = "John"
```

Jenis Penamaan Variabel Pascal

Setiap kata dimulai dengan huruf kapital:

```
MyVariableName = "John"
```

Jenis Penamaan Variabel Snake

Setiap kata dipisahkan oleh karakter garis bawah:

```
my_variable_name = "John"
```

Banyak Nilai ke Banyak Variabel

Python memungkinkan Anda untuk menetapkan nilai ke beberapa variabel dalam satu baris:

```
x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)
```

Catatan: Pastikan jumlah variabel sesuai dengan jumlah nilai, atau Anda akan mendapatkan error.

Satu Nilai untuk Beberapa Variabel

Dan Anda dapat menetapkan nilai yang sama ke beberapa variabel dalam satu baris:

```
x = y = z = "Orange"
print(x)
print(y)
print(z)
```

Membongkar Kumpulan Data

Jika Anda memiliki kumpulan nilai dalam daftar, tuple dll. Python memungkinkan Anda mengekstrak nilai ke dalam variabel. Ini disebut membongkar.

```
fruits = ["apple", "banana", "cherry"]
x, y, z = fruits
print(x)
print(y)
print(z)
```

Variabel Keluaran

Pernyataan print pada Python sering digunakan untuk mengeluarkan variabel. Untuk menggabungkan teks dan variabel, Python menggunakan karakter +:

```
x = "awesome"
print("Python is " + x)
```

Anda juga dapat menggunakan karakter + untuk menambahkan variabel ke variabel lain:

```
x = "Python is "
y = "awesome"
z =  x + y
print(z)
```

Untuk angka, karakter + berfungsi sebagai operator matematika:

```
x = 5
y = 10
print(x + y)
```

Jika Anda mencoba menggabungkan string dan angka, Python akan memberi Anda kesalahan:

```
x = 5
y = "John"
print(x + y)
```

Variabel Global

Variabel yang dibuat di luar fungsi (seperti dalam semua contoh sebelumnya) dikenal sebagai variabel global. Variabel global dapat digunakan oleh semua orang, baik di dalam fungsi maupun di luar.

Buat variabel di luar fungsi, dan gunakan di dalam fungsi

```
x = "awesome"
def myfunc():
  print("Python is " + x)
myfunc()
```

Jika Anda membuat variabel dengan nama yang sama di dalam fungsi, variabel ini akan menjadi variabel lokal, dan hanya dapat digunakan di dalam fungsi. Variabel global dengan nama yang sama akan tetap seperti sebelumnya, global dan dengan nilai aslinya.

Buat variabel di dalam fungsi, dengan nama yang sama dengan variabel global

```
x = "awesome"
def myfunc():
  x = "fantastic"
  print("Python is " + x)
myfunc()
print("Python is " + x)
```

Kata Kunci global

Biasanya, saat Anda membuat variabel di dalam fungsi, variabel itu bersifat lokal, dan hanya dapat digunakan di dalam fungsi itu. Untuk membuat variabel global di dalam suatu fungsi, Anda dapat menggunakan kata kunci global.

Jika Anda menggunakan kata kunci global, variabel tersebut termasuk dalam cakupan global:

```
def myfunc():
  global x
  x = "fantastic"
myfunc()
print("Python is " + x)
```

Selain itu, gunakan kata kunci global jika Anda ingin mengubah variabel global di dalam fungsi.

Untuk mengubah nilai variabel global di dalam fungsi, lihat variabel dengan menggunakan kata kunci global:

```
x = "awesome"
def myfunc():
  global x
  x = "fantastic"
myfunc()
print("Python is " + x)
```

Tipe Data Bawaan

Dalam pemrograman, tipe data merupakan konsep penting. Variabel dapat menyimpan data dari berbagai jenis, dan jenis yang berbeda dapat melakukan hal yang berbeda. Python memiliki tipe data bawaan berikut ini secara default, dalam kategori ini:

Tipe Teks: str

Jenis Numerik: int, float, complex

Jenis Urutan: list, tuple, range

Jenis Pemetaan: dict

Jenis Set: set, frozenset

Tipe Boolean: bool

Jenis Biner: byte, bytearray, memoryview

Mendapatkan Tipe Data

Anda bisa mendapatkan tipe data dari objek apa pun dengan menggunakan fungsi type ():

Cetak tipe data dari variabel x:
`x = 5` `print(type(x))`

Mengatur Jenis Data

Di Python, tipe data ditentukan saat Anda menetapkan nilai ke variabel:

Contoh	Data Type
x = "Hello World"	str
x = 20	int
x = 20.5	float
x = 1j	complex
x = ["apple", "banana", "cherry"]	list
x = ("apple", "banana", "cherry")	tuple
x = range(6)	range
x = {"name" : "John", "age" : 36}	dict
x = {"apple", "banana", "cherry"}	set
x = frozenset({"apple", "banana", "cherry"})	frozenset
x = True	bool
x = b"Hello"	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	memoryview

Mengatur Tipe Data Tertentu

Jika Anda ingin menentukan tipe data, Anda dapat menggunakan fungsi konstruktor berikut:

Contoh	Tipe Data
x = str("Hello World")	str
x = int(20)	int
x = float(20.5)	float
x = complex(1j)	complex
x = list(("apple", "banana", "cherry"))	list
x = tuple(("apple", "banana", "cherry"))	tuple
x = range(6)	range
x = dict(name="John", age=36)	dict
x = set(("apple", "banana", "cherry"))	set
x = frozenset(("apple", "banana", "cherry"))	frozenset
x = bool(5)	bool
x = bytes(5)	bytes
x = bytearray(5)	bytearray
x = memoryview(bytes(5))	memoryview

Angka Python

Ada tiga tipe numerik di Python:

int

float

kompleks

Variabel tipe numerik dibuat ketika Anda memberikan nilai padanya:

```
x = 1    # int
y = 2.8  # float
```

```
z = 1j   # complex
```

Untuk memverifikasi jenis objek apa pun dengan Python, gunakan fungsi type ():

```
print(type(x))
print(type(y))
print(type(z))
```

Int

Int, atau integer, adalah bilangan bulat, positif atau negatif, tanpa desimal, dengan panjang tidak terbatas.

```
x = 1
y = 35656222554887711
z = -3255522
print(type(x))
print(type(y))
print(type(z))
```

Float

Float, atau "floating point number" adalah angka, positif atau negatif, yang mengandung satu atau lebih desimal.

```
x = 1.10
y = 1.0
z = -35.59
print(type(x))
print(type(y))
print(type(z))
```

Float juga bisa berupa bilangan ilmiah dengan "e" untuk menunjukkan pangkat 10.

```
x = 35e3
y = 12E4
z = -87.7e100
print(type(x))
print(type(y))
print(type(z))
```

Kompleks

Bilangan kompleks ditulis dengan "j" sebagai bagian imajiner:

```
x = 3+5j
y = 5j
z = -5j
print(type(x))
print(type(y))
print(type(z))
```

Konversi Tipe Data

Anda bisa mengonversi dari satu tipe ke tipe lainnya dengan metode int (), float (), dan complex ():

```
x = 1    # int
y = 2.8  # float
z = 1j   # complex
#convert from int to float:
a = float(x)
```

```
#convert from float to int:
b = int(y)
#convert from int to complex:
c = complex(x)
print(a)
print(b)
print(c)
print(type(a))
print(type(b))
print(type(c))
```

Catatan: Anda tidak dapat mengonversi bilangan kompleks menjadi tipe bilangan lain.

Angka Acak

Python tidak memiliki fungsi random () untuk membuat angka acak, tetapi Python memiliki modul built-in bernama random yang dapat digunakan untuk membuat angka acak:

```
import random
print(random.randrange(1, 10))
```

Menentukan Jenis Variabel

Ada saat-saat ketika Anda ingin menentukan tipe pada variabel. Ini bisa dilakukan dengan casting. Python adalah bahasa berorientasi objek, dan karena itu ia menggunakan kelas untuk mendefinisikan tipe data, termasuk tipe primitifnya. Karena itu, casting dalam python dilakukan menggunakan fungsi konstruktor:

- int () - membangun bilangan bulat dari literal integer, literal float (dengan menghapus semua desimal), atau literal string (memberikan string mewakili bilangan bulat)
- float () - membangun angka float dari literal integer, literal float atau literal string (asalkan string mewakili float atau integer)

- str () - membangun string dari berbagai tipe data, termasuk string, literal integer, dan literal float

Int

```
x = int(1)   # x will be 1
y = int(2.8) # y will be 2
z = int("3") # z will be 3
```

Float

```
x = float(1)     # x will be 1.0
y = float(2.8)   # y will be 2.8
z = float("3")   # z will be 3.0
w = float("4.2") # w will be 4.2
```

String

```
x = str("s1") # x will be 's1'
y = str(2)    # y will be '2'
z = str(3.0)  # z will be '3.0'
```

BAB STRING

String

Variabel string dalam python dikurung oleh tanda kutip tunggal, atau tanda kutip ganda, sebagai contoh 'halo' sama dengan "halo". Anda dapat menampilkan literal string dengan fungsi print ():

```
print("Hello")
print('Hello')
```

Memasukkan String ke Variabel

Memasukkan nilai string ke variabel dilakukan dengan nama variabel diikuti dengan tanda sama dengan dan string:

```
a = "Hello"
print(a)
```

String Multibaris

Anda dapat menetapkan string multibaris ke variabel dengan menggunakan tiga tanda kutip:

Anda dapat menggunakan tiga tanda kutip ganda:

```
a = """ Erfahren Sie Python-Programmierung,
Einfach, leicht zu lernen,
Spaß, profitabel und
Nützlich für die Zukunft."""
print(a)
```

Atau tiga tanda kutip tunggal:

```
a = Erfahren Sie Python-Programmierung,
```

```
Einfach, leicht zu lernen,
Spaß, profitabel und
Nützlich für die Zukunft.'''
print(a)
```

String adalah Array

Seperti banyak bahasa pemrograman populer lainnya, string dalam Python adalah array byte yang mewakili karakter unicode. Namun, Python tidak memiliki tipe data karakter, satu karakter hanyalah string dengan panjang 1. Tanda kurung siku dapat digunakan untuk mengakses elemen string.

Dapatkan karakter di posisi 1 (ingat bahwa karakter pertama memiliki posisi 0):

```
a = "Hello, World!"
print(a[1])
```

Perulangan pada String

Karena string adalah array, kita dapat melakukan perulangan melalui karakter dalam string, dengan for loop.

Ulangi huruf pada kata "banana":

```
for x in "banana":
  print(x)
```

Panjang String

Untuk mendapatkan panjang string, gunakan fungsi len ().

Fungsi len () mengembalikan panjang string:

```
a = "Hello, World!"
```

```
print(len(a))
```

Memeriksa String

Untuk memeriksa apakah frase atau karakter tertentu ada dalam sebuah string, kita dapat menggunakan kata kunci in.

Periksa apakah kata "free" ada dalam teks berikut:

```
txt = "The best things in life are free!"
print("free" in txt)
```

Atau menggunakan pernyataan if:

Cetak hanya jika kata "free" ada:

```
txt = "The best things in life are free!"
if "free" in txt:
  print("Yes, 'free' is present.")
```

Memeriksa Jika Tidak

Untuk memeriksa apakah frase atau karakter tertentu TIDAK ada dalam string, kita dapat menggunakan kata kunci not in.

Periksa apakah kata "expensive" TIDAK ada dalam teks berikut:

```
txt = "The best things in life are free!"
print("expensive" not in txt)
```

Atau menggunakan pernyataan if:

cetak hanya jika kata "expensive" TIDAK ada:

```
txt = "The best things in life are free!"
if "expensive" not in txt:
  print("Yes, 'expensive' is NOT present.")
```

Mengiris Kata

Anda dapat mengembalikan berbagai karakter dengan menggunakan sintaksis irisan. Tentukan nilai indeks awal dan nilai indeks akhir, lalu pisahkan oleh titik dua, untuk mengembalikan bagian dari string.

Dapatkan karakter dari posisi 2 ke posisi 5 (tidak termasuk):

```
b = "Hello, World!"
print(b[2:5])
```

Irisan dari Awal

Dengan meninggalkan indeks awal, rentang akan dimulai dari karakter pertama:

Dapatkan karakter dari awal hingga posisi 5 (tidak termasuk):

```
b = "Hello, World!"
print(b[:5])
```

Irisan Sampai Akhir

Dengan meninggalkan indeks akhir, rentang akan menuju ke akhir:

Dapatkan karakter dari posisi 2 sampai terus hingga akhir:

```
b = "Hello, World!"
print(b[2:])
```

Pengindeksan Negatif

Gunakan indeks negatif untuk memulai potongan dari akhir string:

Dapatkan karakter dari: "o" di "World!" (posisi -5)

Untuk, tetapi tidak termasuk: "d" dalam "World!" (posisi -2):

```
b = "Hello, World!"
print(b[-5:-2])
```

Modifikasi String

Python memiliki seperangkat metode bawaan yang dapat Anda gunakan pada string.

Huruf Besar

Metode upper () mengembalikan string dalam huruf besar:

```
a = "Hello, World!"
print(a.upper())
```

Huruf Kecil

Metode lower () mengembalikan string dalam huruf kecil:

```
a = "Hello, World!"
print(a.lower())
```

Menghapus Spasi Putih

Spasi putih adalah spasi sebelum dan / atau setelah teks sebenarnya, dan seringkali Anda ingin menghapus spasi ini. Metode strip () menghapus spasi apa pun dari awal atau akhir:

```
a = " Hello, World! "
print(a.strip()) # Mengembalikan "Hello, World!"
```

Mengganti String

Metode replace () menggantikan string dengan string lain:

```
a = "Hello, World!"
```

```
print(a.replace("H", "J"))
```

Memisahkan String

Metode split () mengembalikan daftar di mana teks di antara pemisah yang ditentukan menjadi daftar item. Metode split () membagi string menjadi beberapa substring jika menemukan item pemisah:

```
a = "Hello, World!"

print(a.split(",")) # mengembalikan ['Hello', ' World!']
```

Menggabungkan String

Untuk menggabungkan, atau menggabungkan, dua string Anda dapat menggunakan operator +. Gabungkan variabel a dengan variabel b menjadi variabel c:

```
a = "Hello"

b = "World"

c = a + b

print(c)
```

Untuk menambahkan spasi di antara keduanya, tambahkan "":

```
a = "Hello"

b = "World"

c = a + " " + b

print(c)
```

Format String

Seperti yang kita pelajari di bab Variabel Python, kita tidak bisa menggabungkan string dan angka seperti ini:

```
age = 36

txt = "My name is John, I am " + age

print(txt)
```

Tapi kita bisa menggabungkan string dan angka dengan menggunakan metode format ()! Metode format () mengambil argumen yang diteruskan, memformatnya, dan menempatkannya dalam string di mana placeholder {} berada. Gunakan metode format () untuk memasukkan angka ke dalam string:

```
age = 36

txt = "My name is John, and I am {}"

print(txt.format(age))
```

Metode format () menggunakan jumlah argumen yang tidak terbatas, dan ditempatkan ke dalam placeholder masing-masing:

```
quantity = 3

itemno = 567

price = 49.95

myorder = "I want {} pieces of item {} for {} dollars."

print(myorder.format(quantity, itemno, price))
```

Anda bisa menggunakan nomor indeks {0} untuk memastikan argumen ditempatkan di placeholder yang benar:

```
quantity = 3

itemno = 567

price = 49.95

myorder = "I want to pay {2} dollars for {0} pieces of item {1}."

print(myorder.format(quantity, itemno, price))
```

Karakter Escape

Untuk menyisipkan karakter yang ilegal dalam string, gunakan karakter escape. Karakter escape adalah garis miring terbalik \ diikuti dengan karakter yang ingin Anda sisipkan. Contoh karakter ilegal adalah tanda kutip ganda di dalam string yang diapit tanda kutip ganda. Anda akan mendapatkan pesan kesalahan jika Anda menggunakan tanda kutip ganda di dalam string yang dikelilingi oleh tanda kutip ganda:

```
txt = "We are the so-called "Vikings" from the north."
```

Untuk memperbaiki masalah ini, gunakan karakter escape \ ". Karakter escape memungkinkan Anda untuk menggunakan tanda kutip ganda saat Anda biasanya tidak diizinkan:

```
txt = "We are the so-called \"Vikings\" from the north."
```

Karakter Escape

Karakter escape lain yang digunakan di Python adalah

Kode	Hasil
\'	Single Quote
\\	Backslash
\n	New Line
\r	Carriage Return
\t	Tab
\b	Backspace
\f	Form Feed
\ooo	Octal value
\xhh	Hex value

Metode String

Python memiliki seperangkat metode bawaan yang dapat Anda gunakan pada string.

capitalize () Mengonversi karakter pertama menjadi huruf besar

casefold () Mengubah string menjadi huruf kecil

center () Mengembalikan string yang berada di tengah

count () Mengembalikan berapa kali nilai yang ditentukan terjadi dalam string

encode () Mengembalikan versi string yang dikodekan

endswith () Mengembalikan nilai true jika string diakhiri dengan nilai yang ditentukan

expandtabs () Menyetel ukuran tab dari string

find () Mencari string untuk nilai tertentu dan mengembalikan posisi di mana ia ditemukan

format () Memformat nilai yang ditentukan dalam sebuah string

format_map () Memformat nilai yang ditentukan dalam sebuah string

index () Mencari string untuk nilai tertentu dan mengembalikan posisi di mana ia ditemukan

isalnum () Mengembalikan True jika semua karakter dalam string adalah alfanumerik

isalpha () Mengembalikan True jika semua karakter dalam string ada dalam alfabet

isdecimal () Mengembalikan True jika semua karakter dalam string adalah desimal

isdigit () Mengembalikan True jika semua karakter dalam string adalah digit

isidentifier () Mengembalikan True jika string adalah pengenal

islower () Mengembalikan True jika semua karakter dalam string adalah huruf kecil

isnumeric () Mengembalikan True jika semua karakter dalam string adalah numerik

isprintable () Mengembalikan True jika semua karakter dalam string dapat dicetak

isspace () Mengembalikan True jika semua karakter dalam string adalah spasi putih

istitle () Mengembalikan True jika string mengikuti aturan judul

isupper () Mengembalikan True jika semua karakter dalam string adalah huruf besar

join () Menggabungkan elemen dari sebuah iterable ke akhir string

ljust () Mengembalikan versi rata kiri dari string

lower () Mengubah string menjadi huruf kecil

lstrip () Mengembalikan versi trim kiri string

maketrans () Mengembalikan tabel terjemahan untuk digunakan dalam terjemahan

partisi () Mengembalikan tupel di mana string dibagi menjadi tiga bagian

replace () Mengembalikan string di mana nilai yang ditentukan diganti dengan nilai yang ditentukan

rfind () Mencari string untuk nilai tertentu dan mengembalikan posisi terakhir di mana ia ditemukan

rindex () Mencari string untuk nilai tertentu dan mengembalikan posisi terakhir di mana ia ditemukan

rjust () Mengembalikan versi rata kanan dari string

rpartition () Mengembalikan tupel di mana string dibagi menjadi tiga bagian

rsplit () Memisahkan string pada pemisah yang ditentukan, dan mengembalikan daftar

rstrip () Mengembalikan versi trim kanan dari string

split () Memisahkan string pada pemisah yang ditentukan, dan mengembalikan daftar

splitlines () Memisahkan string pada jeda baris dan mengembalikan daftar

startswith () Mengembalikan nilai true jika string dimulai dengan nilai yang ditentukan

strip () Mengembalikan versi string yang dipangkas

swapcase () Swaps case, huruf kecil menjadi huruf besar dan sebaliknya

title () Mengonversi karakter pertama dari setiap kata menjadi huruf besar

translate () Mengembalikan string yang diterjemahkan

upper () Mengubah string menjadi huruf besar

zfill () Mengisi string dengan sejumlah nilai 0 di awal

BAB BOOLEAN DAN OPERATOR

Nilai Boolean

Boolean mewakili salah satu dari dua nilai: True (Benar) atau False (Salah). Dalam pemrograman Anda sering perlu mengetahui apakah ekspresi itu Benar atau Salah. Anda dapat mengevaluasi ekspresi apa pun dengan Python, dan mendapatkan salah satu dari dua jawaban, Benar atau Salah. Saat Anda membandingkan dua nilai, ekspresi dievaluasi dan Python mengembalikan jawaban Boolean:

```
print(10 > 9)
print(10 == 9)
print(10 < 9)
```

Saat Anda menjalankan kondisi dalam pernyataan if, Python mengembalikan True atau False. Cetak pesan berdasarkan apakah kondisinya True atau False:

```
a = 200
b = 33
if b > a:
  print("b is greater than a")
else:
  print("b is not greater than a")
```

Mengevaluasi Nilai dan Variabel

Fungsi bool () memungkinkan Anda mengevaluasi nilai apa pun, dan memberi Anda True atau False sebagai gantinya. Evaluasi string dan angka:

```
print(bool("Hello"))
print(bool(15))
```

Evaluasi dua variabel:

```
x = "Hello"
y = 15
print(bool(x))
print(bool(y))
```

Kebanyakan Nilai adalah Benar

Hampir semua nilai dievaluasi ke True jika memiliki semacam konten. Semua string adalah True, kecuali string kosong. Angka apa pun Benar, kecuali 0. Setiap daftar, tupel, set, dan kamus adalah True, kecuali yang kosong. Kode berikut ini akan mengembalikan True:

```
bool("abc")
bool(123)
bool(["apple", "cherry", "banana"])
```

Beberapa Nilai Salah

Nyatanya, tidak banyak nilai yang dievaluasi ke False, kecuali nilai kosong, seperti (), [], {}, "", angka 0, dan nilai None. Dan tentu saja nilai False terevaluasi menjadi False. Kode berikut ini akan mengembalikan False:

```
bool(False)
bool(None)
bool(0)
bool("")
bool(())
bool([])
bool({})
```

Satu lagi, atau objek dalam kasus ini, mengevaluasi ke False, dan itu jika Anda memiliki objek yang dibuat dari kelas dengan fungsi __len__ akan mengembalikan 0 atau False:

```
class myclass():
  def __len__(self):
    return 0
myobj = myclass()
print(bool(myobj))
```

Fungsi dapat Mengembalikan Boolean

Anda dapat membuat fungsi yang mengembalikan Nilai Boolean. Cetak jawaban dari suatu fungsi:

```
def myFunction() :
  return True
print(myFunction())
```

Anda dapat mengeksekusi kode berdasarkan jawaban Boolean dari suatu fungsi. Cetak "YA!" jika fungsi mengembalikan True, jika tidak cetak "NO!":

```
def myFunction() :
  return True
if myFunction():
  print("YES!")
else:
  print("NO!")
```

Python juga memiliki banyak fungsi bawaan yang mengembalikan nilai boolean, seperti fungsi isinstance (), yang dapat digunakan untuk menentukan apakah suatu objek berjenis data tertentu. Periksa apakah suatu objek adalah bilangan bulat atau bukan:

```
x = 200
```

```
print(isinstance(x, int))
```

Operator

Operator digunakan untuk melakukan operasi pada variabel dan nilai. Pada contoh di bawah ini, kita menggunakan operator + untuk menjumlahkan dua nilai:

```
print(10 + 5)
```

Python membagi operator dalam grup berikut: Operator aritmatika, Operator penugasan, Operator perbandingan, Operator logika, Operator identitas, Operator keanggotaan, Operator bitwise.

Operator Aritmatika Python

Operator aritmatika digunakan dengan nilai numerik untuk melakukan operasi matematika umum:

Contoh	Nama	Operator
+	Tambahan	`x + y`
-	Pengurangan	`x - y`
*	Perkalian	`x * y`
/	Divisi	`x / y`
%	Modulus	`x % y`
**	Eksponen	`x ** y`
//	Floor division	`x // y`

Operator Penugasan Python

Operator penugasan digunakan untuk menetapkan nilai ke variabel:

Contoh	Operator	Sama Seperti
=	`x = 5`	`x = 5`
+=	`x += 3`	`x = x + 3`

-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3
<<=	x <<= 3	x = x << 3

Operator Perbandingan Python

Operator perbandingan digunakan untuk membandingkan dua nilai:

Contoh	Nama	Operator
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Operator Logika Python

Operator logika digunakan untuk menggabungkan pernyataan kondisional:

Contoh	Deskripsi	Operator
and	Mengembalikan True jika kedua pernyataan benar	x < 5 and x < 10
or	Mengembalikan True jika salah satu pernyataan benar	x < 5 or x < 4
not	Membalikkan hasilnya, mengembalikan False jika hasilnya benar	not(x < 5 and x < 10)

Operator Identitas Python

Operator identitas digunakan untuk membandingkan objek, bukan jika mereka sama, tetapi jika mereka sebenarnya adalah objek yang sama, dengan lokasi memori yang sama:

is	Mengembalikan True jika kedua variabel adalah objek yang sama	x is y
is not	Mengembalikan True jika kedua variabel bukan objek yang sama	x is not y

Operator Keanggotaan Python

Operator keanggotaan digunakan untuk menguji apakah urutan disajikan dalam suatu objek:

Contoh	Deskripsi	Operator
in	Mengembalikan True jika urutan dengan nilai yang ditentukan ada di objek	x in y

not in	Mengembalikan True jika urutan dengan nilai yang ditentukan tidak ada dalam objek	x not in y

Operator Bitwise Python

Operator bitwise digunakan untuk membandingkan angka (biner):

Deskripsi	Nama	Operator
&	AND	Set setiap bit ke 1 jika kedua bit adalah 1
|	OR	Set setiap bit ke 1 jika salah satu dari dua bit adalah 1
^	XOR	Setel setiap bit menjadi 1 jika hanya satu dari dua bit yang bernilai 1
~	NOT	Membalikkan semua bit
<<	Zero fill left shift	Geser ke kiri dengan menekan angka nol dari kanan dan biarkan bit paling kiri jatuh
>>	Signed right shift	Geser ke kanan dengan mendorong salinan bit paling kiri ke dalam dari kiri, dan biarkan bit paling kanan jatuh

BAB LIST, TUPLE, SET, DAN DICTIONARY

List

List digunakan untuk menyimpan beberapa item dalam satu variabel. List adalah salah satu dari 4 tipe data bawaan dalam Python yang digunakan untuk menyimpan kumpulan data, 3 lainnya adalah Tuple, Set, dan Dictionary, semuanya dengan kualitas dan penggunaan yang berbeda.

List dibuat menggunakan tanda kurung siku:

Contoh Membuat List:

```
listbuah = ["apel", "pisang", "ceri"]
print(listbuah)
```

Tuple

Tuple digunakan untuk menyimpan beberapa item dalam satu variabel. Tuple adalah salah satu dari 4 tipe data bawaan dalam Python yang digunakan untuk menyimpan kumpulan data, 3 lainnya adalah List, Set, dan Kamus, semuanya dengan kualitas dan penggunaan yang berbeda.

Tuple adalah kumpulan data dan **tidak dapat diubah**. Tuple ditulis dengan tanda kurung bulat.

Contoh Membuat Tuple:

```
tupleBuah = ("apel", "pisang", "ceri")
print(tupleBuah)
```

Set

Set digunakan untuk menyimpan beberapa item dalam satu variabel. Set adalah salah satu dari 4 tipe data bawaan dalam Python yang digunakan untuk menyimpan kumpulan data, 3 lainnya adalah List, Tuple, dan Dictionary, semuanya dengan kualitas dan penggunaan yang berbeda.

Himpunan adalah kumpulan yang tidak terurut dan tidak terindeks. Himpunan ditulis dengan kurung kurawal.

Contoh Membuat Set:

```
setBuah = {"apel", "pisang", "ceri"}
print(setBuah)
```

Dictionary

Dictionary digunakan untuk menyimpan nilai data dalam pasangan kunci:nilai. Dictionary adalah kumpulan yang tersusun, dapat diubah dan tidak memungkinkan duplikat.

Contoh Membuat Dictionary:

```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
print(thisdict)
```

Kumpalan Data Python (Array)

Ada empat tipe data kumplan data dalam bahasa pemrograman Python:

List adalah kumpulan data yang tersusun dan dapat diubah. Memungkinkan data duplikat.

Tuple adalah kumpulan yang tersusun dan **tidak dapat diubah**. Memungkinkan data duplikat.

Set adalah kumpulan yang tidak berurutan dan tidak terindeks. Tidak ada data duplikat.

Dictionary adalah kumpulan yang tersusun dan dapat diubah. Tidak ada data duplikat.

BAB IF… ELSE

Kondisi Python dan Pernyataan If

Python mendukung kondisi logis yang biasa dari matematika:

Sama dengan: a == b

Tidak Sama dengan: a != b

Kurang dari: a < b

Kurang dari atau sama dengan: a <= b

Lebih besar dari: a > b

Lebih besar dari atau sama dengan: a >= b

Kondisi ini dapat digunakan dalam beberapa cara, paling umum dalam "pernyataan if" dan loop.

Sebuah "pernyataan if" ditulis dengan menggunakan kata kunci if.

```
a = 33
b = 200
if b > a:
  print("b lebih besar dari a")
```

Dalam contoh ini menggunakan dua variabel, a dan b, yang digunakan sebagai bagian dari pernyataan if untuk menguji apakah b lebih besar dari a. Karena a adalah 33, dan b adalah 200, Dapat diketahui bahwa 200 lebih besar dari 33, jadi tercetak ke layar bahwa "b lebih besar dari a".

Identation

Python bergantung pada identation (spasi di awal baris) untuk menentukan ruang lingkup dalam kode. Bahasa pemrograman lain sering menggunakan kurung kurawal untuk tujuan ini.

Contoh: Jika pernyataan, tanpa identation (akan menimbulkan kesalahan):

```
a = 33
b = 200
if b > a:
```

```
print("b lebih besar dari a") # kita akan menemui error program
```

Elif

Kata kunci elif adalah cara python untuk mengatakan "jika kondisi sebelumnya tidak benar, maka coba kondisi ini".

```
a = 33
b = 33
if b > a:
  print("b lebih besar dari a")
elif a == b:
  print("a dan b bernilai sama")
```

Dalam contoh ini a sama dengan b, jadi kondisi pertama tidak benar, tetapi kondisi elif benar, jadi kami mencetak ke layar bahwa "a dan b sama".

Else

Kata kunci else memproses apa pun yang tidak diproses oleh kondisi sebelumnya.

```
a = 200
b = 33
if b > a:
  print("b lebih besar dari a")
elif a == b:
  print("a dan b bernilai sama")
else:
  print("a lebih besar dari b")
```

Dalam contoh ini a lebih besar dari b, jadi kondisi pertama tidak benar, juga kondisi elif tidak benar, jadi kita pergi ke kondisi lain dan mencetak ke layar bahwa "a lebih besar dari b".

Programmer juga dapat memiliki yang lain tanpa elif:

```
a = 200
b = 33
if b > a:
  print("b lebih besar dari a")
else:
  print("b tidak lebih besar dari a")
```

If Pendek

Jika programmer hanya memiliki satu pernyataan untuk dieksekusi, programmer dapat meletakkannya di baris yang sama dengan pernyataan if.

Contoh Satu baris pernyataan if:

```
a = 200
b = 33
if a > b: print("a lebih besar dari b")
```

If… Else Pendek

Jika Programmer hanya memiliki satu pernyataan untuk dieksekusi, satu untuk jika, dan satu untuk yang lain, Programmer dapat meletakkan semuanya di baris yang sama:

Contoh Pernyataan if… else satu baris:

```
a = 2
b = 330
print("A") if a > b else print("B")
```

Teknik ini dikenal sebagai Operator Ternary, atau Ekspresi Bersyarat. Programmer juga dapat memiliki beberapa pernyataan lain pada baris yang sama:

Contoh Pernyataan if… else satu baris dengan 3 kondisi:

```
a = 330

b = 330

print("A") if a > b else print("=") if a == b else print("B")
```

And

Kata kunci and adalah operator logika, dan digunakan untuk menggabungkan pernyataan bersyarat:

```
a = 200

b = 33

c = 500

if a > b and c > a:

  print("Kedua nya bernilai benar")
```

Or

Kata kunci or adalah operator logika, dan digunakan untuk menggabungkan pernyataan bersyarat:

```
a = 200

b = 33

c = 500

if a > b or a > c:

  print("Sedikitnya satu kondisi terpenuhi")
```

If Bersarang (Nested If)

Programmer dapat memiliki pernyataan if di dalam pernyataan if, ini disebut pernyataan if bersarang.

```
x = 41

if x > 10:

  print("Nilai di atas 10,")
```

```
  if x > 20:
    print("Dan juga di atas 20!")
  else:
    print("Tapi tidak di atas 20.")
```

Pernyataan Pass

Pernyataan if tidak boleh kosong, tetapi jika Programmer dikarenakan alasan tertentu memiliki pernyataan if tanpa konten, programmer dapat memasukkan pernyataan pass untuk menghindari kesalahan program.

```
a = 33
b = 200
if b > a:
  pass
```

BAB PERULANGAN WHILE & FOR

Python memiliki dua perintah perulangan primitive yaitu Perulangan while dan Perulangan for.

Dengan perulangan while kita dapat mengeksekusi satu set pernyataan selama kondisinya benar.

Contoh Cetak i selama i kurang dari 6:

```
i = 1
while i < 6:
  print(i)
  i += 1
```

Catatan: Ingat untuk menambah i, atau loop akan berjalan selamanya.

Perulangan while membutuhkan variabel yang relevan dan tersedia, dalam contoh ini programmer perlu mendefinisikan variabel pengindeksan, i, yang ditentukan sebagai nilai ke 1.

Pernyataan Break

Dengan pernyataan break kita dapat menghentikan perulangan meskipun kondisi while benar:

Contoh Keluar dari loop ketika i adalah 3:

```
i = 1
while i < 6:
  print(i)
  if i == 3:
    break
  i += 1
```

Pernyataan Continue

Dengan pernyataan continue programmer dapat menghentikan iterasi saat ini, dan melanjutkan dengan yang berikutnya:

Contoh Lanjutkan ke iterasi berikutnya jika i adalah 3:

```
i = 0
while i < 6:
  i += 1
  if i == 3:
    continue
  print(i)
```

Pernyataan Else

Dengan pernyataan else programmer dapat menjalankan blok kode satu kali ketika kondisinya tidak lagi benar:

Contoh Cetak pesan setelah kondisinya salah:

```
i = 1
while i < 6:
  print(i)
  i += 1
else:
  print("i tidak lagi kurang dari 6")
```

Perulangan For

Perulangan for digunakan untuk mengulangi urutan (yaitu daftar, tupel, kamus, set, atau string).

Kurang lebih ini seperti perintah perulangan for dalam bahasa pemrograman lain, dan bekerja lebih seperti metode iterator (iterasi) seperti yang ditemukan dalam bahasa pemrograman berorientasi objek lainnya.

Dengan perulangan for programmer dapat mengeksekusi satu set pernyataan, sekali untuk setiap item dalam list, tuple, set dll.

Contoh Cetak setiap buah dalam daftar buah:

```
fruits = ["apel", "pisang", "ceri"]
for x in fruits:
  print(x)
```

Perulangan for tidak memerlukan variabel pengindeksan untuk disetel sebelumnya.

Perulangan Melalui String

Bahkan string adalah objek yang dapat diubah, mereka berisi urutan karakter:

```
for x in "banana":
  print(x)
```

Pernyataan Break

Dengan pernyataan break kita dapat menghentikan loop sebelum loop melewati semua item:

Contoh Keluar dari loop ketika x adalah "pisang":

```
buah = ["apel", "pisang", "ceri"]
for x in buah:
  print(x)
  if x == "pisang":
    break
```

Contoh Keluar dari loop ketika x adalah "pisang", tetapi kali ini jeda muncul sebelum cetakan:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    break
  print(x)
```

Pernyataan Continue

Dengan pernyataan continue kita dapat menghentikan iterasi loop saat ini, dan melanjutkan dengan yang berikutnya:

Contoh Jangan mencetak pisang:

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
  if x == "banana":
    continue
  print(x)
```

Fungsi range()

Untuk mengulang satu set kode beberapa kali, kita dapat menggunakan fungsi range(), Fungsi range() mengembalikan urutan angka, mulai dari 0 secara default, dan bertambah 1 (secara default), dan berakhir pada angka yang ditentukan.

Contoh Menggunakan fungsi range():

```
for x in range(6):
  print(x)
```

Perhatikan bahwa range(6) bukanlah nilai 0 hingga 6, tetapi nilai 0 hingga 5.

Fungsi range() default ke 0 sebagai nilai awal, namun dimungkinkan untuk menentukan nilai awal dengan menambahkan parameter: range(2, 6), yang berarti nilai dari 2 hingga 6 (tetapi tidak termasuk 6):

Contoh Menggunakan parameter mulai:

```
for x in range(2, 6):
  print(x)
```

Fungsi range() default untuk menambah urutan sebesar 1, namun dimungkinkan untuk menentukan nilai kenaikan dengan menambahkan parameter ketiga: range(2, 30, 3):

Contoh Tingkatkan urutan dengan 3 (default adalah 1):

```
for x in range(2, 30, 3):
  print(x)
```

Else di Perulangan For

Kata kunci else dalam perulangan for menentukan blok kode yang akan dieksekusi ketika perulangan selesai:

Contoh Cetak semua angka dari 0 hingga 5, dan cetak pesan saat loop telah berakhir:

```
for x in range(6):
  print(x)
else:
  print("Finally finished!")
```

Catatan: Blok else TIDAK akan dieksekusi jika loop dihentikan oleh pernyataan break.

Contoh Putuskan loop ketika x adalah 3, dan lihat apa yang terjadi dengan blok else:

```
for x in range(6):
  if x == 3: break
  print(x)
```

```
else:
  print("Finally finished!")
```

Perulangan Bersarang

Perulangan bersarang adalah perulangan di dalam perulangan. "Perulangan dalam" akan dieksekusi satu kali untuk setiap iterasi dari "perulangan luar":

Contoh Cetak setiap kata sifat untuk setiap buah:

```
adj = ["red", "big", "tasty"]
fruits = ["apple", "banana", "cherry"]


for x in adj:
  for y in fruits:
    print(x, y)
```

Pernyataan Pass

Perulangan loop tidak boleh kosong, tetapi jika programmer karena alasan tertentu memiliki perulangan for tanpa konten, masukkan pernyataan pass untuk menghindari kesalahan.

Contoh

```
for x in [0, 1, 2]:
  pass
```

BAB FUNGSI

Fungsi adalah blok kode yang hanya berjalan ketika dipanggil. Programmer dapat meneruskan data, yang dikenal sebagai parameter, ke dalam suatu fungsi. Sebuah fungsi dapat mengembalikan data sebagai hasilnya.

Membuat Fungsi

Dalam Python, sebuah fungsi didefinisikan menggunakan kata kunci def:

Contoh

```
def my_function():
  print("Halo Python dari Fungsi")
```

Memanggil Fungsi

Untuk memanggil fungsi, gunakan nama fungsi diikuti dengan tanda kurung:

Contoh

```
def my_function():
  print("Halo Python dari Fungsi")


my_function()
```

Argumen

Informasi dapat diteruskan ke fungsi sebagai argumen. Argumen ditentukan setelah nama fungsi, di dalam tanda kurung. Anda dapat menambahkan argumen sebanyak yang Anda inginkan, cukup pisahkan dengan koma. Contoh berikut memiliki fungsi dengan satu argumen (fname). Saat fungsi dipanggil, kami memberikan nama depan, yang digunakan di dalam fungsi untuk mencetak nama lengkap:

Contoh:

```
def my_function(fname):
  print(fname + " Refsnes")


my_function("Emil")
my_function("Tobias")
my_function("Linus")
```

Argumen sering disingkat menjadi args dalam dokumentasi Python.

Parameter atau Argumen?

Istilah parameter dan argumen dapat digunakan untuk hal yang sama: informasi yang diteruskan ke suatu fungsi.

Dari perspektif fungsi:

Parameter adalah variabel yang terdaftar di dalam tanda kurung dalam definisi fungsi. Argumen adalah nilai yang dikirim ke fungsi saat dipanggil.

Jumlah Argumen

Secara default, suatu fungsi harus dipanggil dengan jumlah argumen yang benar. Artinya jika fungsi Anda mengharapkan 2 argumen, Anda harus memanggil fungsi dengan 2 argumen, tidak lebih, dan tidak kurang.

Contoh: Fungsi ini mengharapkan 2 argumen, dan mendapat 2 argumen:

```
def my_function(fname, lname):
  print(fname + " " + lname)


my_function("Alfian", "Ma’arif")
```

Jika Programmer mencoba memanggil fungsi dengan 1 atau 3 argumen, programmer akan mendapatkan kesalahan:

Contoh: Fungsi ini mengharapkan 2 argumen, tetapi hanya mendapatkan 1:

```
def my_function(fname, lname):
  print(fname + " " + lname)


my_function("Alfian")
```

Argumen Berubah-Ubah, *args

Jika Programmer tidak tahu berapa banyak argumen yang akan diteruskan ke fungsi, tambahkan * sebelum nama parameter dalam definisi fungsi. Dengan cara ini fungsi akan menerima tupel argumen, dan dapat mengakses item yang sesuai:

Contoh Jika jumlah argumen tidak diketahui, tambahkan * sebelum nama parameter:

```
def my_function(*kids):
  print("Anak bungsu adalah " + kids[2])


my_function("Emil", "Tobias", "Linus")
```

Argumen berubah-ubah sering disingkat menjadi args* dalam dokumentasi Python.

Argumen Kata Kunci

Programmer juga dapat mengirim argumen dengan sintaks kunci = nilai. Dengan cara ini urutan argumen tidak menjadi masalah.

Contoh

```
def my_function(anak3, anak2, anak1):
  print("Anak bungsu adalah " + anak3)


my_function(anak1 = "Emil", anak2 = "Tobias", anak3 = "Linus")
```

DAFTAR PUSTAKA

https://www.w3schools.com/python/