PEMROGRAMAN JARINGAN KOMPUTER DENGAN PYTHON

Justinus Andjarwirawan

Pemrograman Jaringan Komputer dengan Python

Disusun oleh: Justinus Andjarwirawan

Penerbit:

Lembaga Penelitian dan Pengabdian kepada Masyarakat
Universitas Kristen Petra

Pemrograman Jaringan Komputer dengan Python

Justinus Andjarwirawan

Surabaya, Bagian Penerbitan Lembaga Penelitian dan Pengabdian kepada Masyarakat, Universitas Kristen Petra, 2023

ISBN: 978-623-5457-09-3 (PDF)

Pemrograman Jaringan Komputer dengan Python
Cetakan Pertama, Januari 2023

Penulis:
Justinus Andjarwirawan

Penerbit:

Lembaga Penelitian dan Pengabdian kepada Masyarakat
Universitas Kristen Petra
Jl. Siwalankerto 121-131, Surabaya 60236
Telp. 031-2983139, 2983147; Fax. 031-2983111

DAFTAR ISI

Kata Pengantar

Buku Pemrograman Jaringan Komputer dengan Python ini disusun oleh Justinus Andjarwirawan selaku dosen Informatika di Universitas Kristen Petra, Surabaya. Meskipun buku ini dirancang untuk pemrograman jaringan komputer, namun dapat pula digunakan untuk mata kuliah:

- Teknologi Open Source
- Service Oriented Architecture
- Dasar Pemrograman
- Struktur Data
- mata kuliah-mata kuliah lain yang menggunakan Python sebagai bahasa pemrogramannya, seperti Machine Learning / Artificial Intelligence dan Data Science.

Dengan adanya buku ini diharapkan mahasiswa dan juga pengajar dapat memahami Python dengan cepat dimana mahasiswa dan dosen sudah memahami bahasa pemrograman yang lain, terutama yang berbasis object oriented. Karena buku ini dirancang sedemikian rupa dimana pembaca telah memahami bahasa pemrograman lainnya.

Pendahuluan

Bahasa pemrograman Python merupakan bahasa interpreter yang idenya mulai dikembangkan sejak tahun 1980an oleh Guido van Rossum, seorang warga negara Belanda. Di awal tahun 2000an Python muncul dengan versi 2 dan versi 3 di tahun 2008 yang sampai saat ini dikembangkan terus. Perkembangannya diikuti dengan perbaikan bug, performa dan keamanannya.

Pengembangan menggunakan Python dapat dijumpai pada banyak system seperti aplikasi web, mobile, machine learning hingga peralatan IoT.

Hingga buku ini ditulis, Python masih menyediakan versi 2 meskipun versi 3 sudah tersedia, hal ini dikarenakan masih banyaknya system berjalan yang menggunakan versi 2 dan library-library yang hanya tersedia di versi 2. Sedangkan di versi 3 sudah semakin banyak library tambahan yang disediakan secara internal, dimana saat di versi 2 masih harus menambahkannya secara manual dengan library pihak ke-3.

Download Python

Pemrograman Python dapat dilakukan dengan banyak cara, mulai dari instalasi di komputer sendiri maupun di cloud. Karena Python merupakan bahasa pemrograman interpreter, maka eksekusinya bisa langsung dari script yang dieksekusi oleh interpreter Python tanpa perlu melalui proses compile. Untuk menggunakan Python secara lokal dapat dijalankan di 3 platform popular yaitu Microsoft Windows, Apple macOS dan Linux. Untuk menentukan platform yang digunakan dapat dilihat dari tujuan pembuatannya. Misalnya untuk aplikasi web maka disarankan untuk menggunakan Linux karena instalasi modul-modul tambahannya lebih mudah dan banyak tersedia di repository distribusi Linux yang popular. Berikut beberapa cara mendapatkan Python:

Cara pertama: download Python dari website resminya: https://python.org

Disarankan untuk menggunakan versi 3 terbaru kecuali ada suatu ketergantungan pada aplikasi yang sudah berjalan. Untuk mengawali pengembangan dan mulai belajar tetaplah menggunakan versi 3. Hingga penulisan buku ini versi terakhir adalah 3.11.0

Cara kedua: download Python dalam package Anaconda Distribution: https://anaconda.com

Dalam Anaconda tidak hanya menyediakan Python saja namun dilengkapi dengan berbagai macam pilihan IDE, library-library yang siap pakai, dan system packagingnya sendiri, untuk memudahkan pengembangan yang telah disiapkan dalam sebuah ekosistem.

Cara ketiga: menggunakan Python di cloud: https://colab.research.google.com

Cara ketiga ini menggunakan akun Google kita sebagai identitas penggunanya, dimana script yang kita buat akan selalu tersimpan di Google. Setiap code yang kita buat dapat disertai dengan dokumentasinya, ini sama seperti yang dijumpai di Anaconda dengan Jupyter Notebook nya.

Cara keempat: menggunakan REPL (Read-Eval-Print Loop) pada website yaitu sebuah aplikasi interaktif dimana kita dapat menuliskan script Python pada sebuah website

untuk kemudian dieksekusi disana, cara keempat ini banyak sekali tersedia di internet namun cara ini sangat terbatas dengan library yang tersedia saja, sehingga cocoknya untuk digunakan belajar dasar-dasar perintah Python, tidak cocok untuk pembuatan aplikasi web dan aplikasi berat lainnya. Contoh salah satunya adalah: https://repl.it

Apabila pembuatan program dengan Python dilakukan secara lokal, disarankan menggunakan Visual Studio Code, karena Visual Studio Code memiliki banyak fitur dan extensions yang sangat membantu dalam pemrograman banyak bahasa pemrograman termasuk Python. Fitur seperti auto-complete method atau fungsi yang bisa digunakan pada perintah-perintah yang kita ketik, dan juga peringatan kesalahan dalam penulisan perintah-perintah maupun variabel yang sudah kita buat. Visual Studio Code tersedia untuk platform Microsoft Windows, Apple macOS dan Linux, dapat di download di alamat:

https://code.visualstudio.com/

Environment Variable

Hal penting lainnya dalam penggunaan Python adalah letak file interpreter Python. Pastikan pada PATH kita sudah ada lokasi dimana file python berada (python.exe pada Windows), agar interpreter Python dapat dieksekusi dari posisi directory mana pun, terutama apabila Python dipanggil dari aplikasi lain seperti beberapa IDE dan Visual Studio Code.

Dengan PATH kita juga dapat mengatur penggunaan interpreter Python yang mana sebagai default saat kita menginstall beberapa versi Python di komputer yang sama. Perlu diperhatikan juga dalam instalasi library-library baru agar mengikuti versi Python yang sedang digunakan.

Untuk Python di macOS dan Linux umumnya berupa link yang mengarah ke file Python dengan imbuhan nomor versi Python yang terinstall. Link tersebut ada yang diletakkan di /usr/bin/python ada pula yang di /usr/local/bin/python tergantung kita menggunakan installer yang mana.

Mengeksekusi script Python

Setelah PATH ditambahkan, maka interpreter Python dapat dipanggil dari posisi directory mana pun, sehingga kita dapat fokus bekerja pada sebuah directory pekerjaan. Contoh eksekusi script Python:

```
python myfile.py
```

Apabila PATH belum ditambahkan, maka akibatnya lokasi Python harus diketik dengan lengkap, contohnya:

```
/usr/local/bin/python myfile.py
```

Lokasi interpreter Python itu sendiri dapat ditulis di baris pertama script yang kita buat, terutama bila kita membuat script tersebut di Linux atau macOS, contoh scriptnya adalah sebagai berikut:

```
#!/usr/local/bin/python
print("Hello")
```

Diawali dengan #! lalu kemudian diikuti dengan lokasi file Python lengkap dengan pathnya. Dengan demikian file tersebut dapat dieksekusi langsung tanpa menyebutkan perintah python didepannya, namun file script tersebut harus memiliki permission execute:

```
chmod u+x myfile.py
```

Maka file tersebut dapat dieksekusi langsung:

```
./myfile.py
```

Penggunaan IDE dapat membantu eksekusi script tanpa harus ke command line, dan tentunya harus dengan konfigurasi yang tepat terutama letak file python yang digunakan. IDE akan mengeksekusi script pada window yang disediakan, biasanya menggunakan istilah terminal atau console.

Bab 1 - Standard Output dan Standard Input

Dalam pemrograman selalu ada proses dimana diawali dengan data yang diinputkan, begitu pula output untuk menampilkan hasil dari sebuah proses. Dalam Python dikenal dua perintah yaitu **print()** untuk STDOUT dan **input()** untuk STDIN.

Perintah print() terdapat kurung yang artinya print() merupakan sebuah function, begitu pula input(). Untuk perintah print() bisa diisi dengan berbagai parameter selain teks yang akan ditampilkan ke output.

```
print("Selamat pagi")
```

Penggunaan double quotes (") lebih umum untuk menampilkan string secara langsung, namun tidak menutup kemungkinan menggunakan single quote (') juga. Single quote lebih sering digunakan di regular expression dan key dari type data dict.

```
print("Selamat pagi", end="")
```

Print memiliki parameter tambahan yaitu **end**, kegunaannya untuk menentukan karakter terakhir setelah seluruh output ditampilkan, defaultnya adalah newline (**\n**), maka untuk menghilangkan newline dapat ditulis **end=""**. Karena perintah print() itu sendiri sudah menampilkan newline. End dapat didefinisikan ke karakter apapun dan karakter tersebut akan muncul di akhir baris.

```
print("Selamat", "pagi", "semuanya")
```

output:

Selamat pagi semuanya

Print dengan menampilkan beberapa string dipisahkan dengan tanda koma akan secara otomatis ditampilkan seluruhnya dengan batasan berupa satu spasi. Satu buah spasi ini merupakan sebuah separator, jadi karakter separator ini dapat diganti dengan tambahan parameter **sep**, misalnya antar kata tidak menggunakan spasi namun tanda # maka:

```
print("Selamat", "pagi", "semuanya", sep="#")
```

output:

Selamat#pagi#semuanya

print("Selamat", "pagi", "semuanya", sep="#", end="$")

output:

Selamat#pagi#semuanya$

Yang perlu diperhatikan pada perintah print adalah apabila terdapat penggabungan dengan tanda '+' maka seluruh elemen yang digabung harus satu type, misal string maka harus string semua

print("Selamat " + "pagi " + "semuanya")

output:

Selamat pagi semuanya

print("Selamat " + "kepada " + "nomor " + 5)

output:

TypeError: can only concatenate str (not "int") to str

Apabila ingin ditampilkan dengan benar, maka perlu dipastikan seluruh elemen dalam satu type dan bila diperlukan harus dikonversi ke type data yang sama:

print("Selamat " + "kepada " + "nomor " + str(5))

output:

Selamat kepada nomor 5

Untuk standar input, Python menggunakan perintah **input**

isian = input()

maka isian akan diisi sesuai apa yang diinputkan melalui keyboard, dan bertype string secara default. Perlu dikonversi ke type data yang sesuai apabila ingin diproses lebih lanjut. Misalnya yang diharapkan pada input adalah sebuah bilangan float maka:

isian = float(input())

atau

isian = input()

```
isian = float(isian)
```

Untuk memastikan yang diinput memungkinkan untuk dikonversi ke type data yang diharapkan maka input harus berada dalam sebuah try dan exception:

```
try:
  isian = float(input())
except:
  print("input tidak sesuai dengan type data yang diinginkan")
```

Bisa diperhatikan penulisan diatas, bahwa dalam Python tidak menggunakan tanda kurung untuk mengawali dan menutup sebuah fungsi, loop dan lain sebagainya. Sebagai penggantinya adalah dengan memberikan indent ke kanan. Jumlah spasi pada indent harus konsisten. Bila penulisan baris berikutnya ditempatkan di awal baris maka sudah dianggap keluar dari lingkupnya. Jumlah spasi harus konsisten, apabila tidak, maka akan dianggap sudah berada di luar lingkupnya. Apabila menggunakan tab, maka sebaiknya tab terus agar konsisten, karena Python kadang membedakan antara tab dengan spasi meskipun secara jarak indent sudah sama.

Apabila menggunakan Python secara interaktif, maka untuk mengakhiri sebuah indent bisa dilakukan dengan penekanan Enter, lalu kemudian sesuaikan indent yang berikutnya.

Input melalui argument command line

Input dapat diterima oleh script python yang dieksekusi melalui command line, yaitu dengan menuliskan argument yang ada setelah pengetikan nama file script, contoh:

```
$ python3 script.py hello 12
```

Untuk menangkap argument hello dan 12 adalah dengan cara:

```
import sys
arg1 = sys.argv[1]
arg2 = sys.argv[2]
```

Artinya sys.argv adalah berupa list yang isinya argument yang dituliskan, untuk sys.argv[0] akan berisi nama file script yang digunakan saat eksekusi. Apabila jumlah argument bersifat dinamis, maka kita bisa dapatkan seluruhnya dengan sys.argv[1:] dimana akan berupa list berisi seluruh argument yang ada.

Komentar pada Python

Menulis komentar pada Python bisa dengan dua cara:

 # menggunakan tanda ini di awal tulisan, akan membuat seluruh teks setelahnya sebagai komentar atau catatan

```
"""
    Untuk lebih dari satu baris komentar
    dapat menggunakan petik 3 kali
    dan diakhiri dengan petik 3 kali juga
"""
```

Bab 2 - Variable, data type, loop dan if condition

Variable di Python tidak diinisialisasi dengan data type di awal pembuatannya, namun otomatis data type ditentukan dari value yang di-assign terhadap variable tersebut.

A = "Hello"

maka otomatis A ber-type string, yang memiliki value = "Hello"

B = []

maka B adalah variable type list yang belum memiliki elemen apapun.

Type-type data yang umum digunakan di Python:

str

float

int

bool

list

tuple

set

dict

NoneType

Untuk mengetahui type sebuah nilai, dapat menggunakan perintah: type, contoh:

a = "5"

print(type(a))

maka akan muncul: <class 'str'> karena a berisi sebuah string "5".

Untuk mengkonversi type data suatu variable bisa dilakukan dengan cara menuliskan type data tujuan kemudian di dalam kurung adalah variable asal, contoh:

a = "6"

a = int(a)

print(type(a))

maka akan muncul: <class 'int'> sesuai type data barunya. Perhatikan bahwa konversi type data hanya bisa berhasil dilakukan apabila konversinya memungkinkan untuk dilakukan, jadi huruf abjad tidak mungkin bisa dikonversi ke int maupun float.

Variabel string apabile dikonversikan ke list maka akan menjadi sebuah list yang elemennya terdiri dari huruf tiap karakter yang ada pada string tersebut:

a = "Hello"

a = list(a)

maka a akan menjadi ['H', 'e', 'l', 'l', 'o']

Type data bool memiliki dua kemungkinan nilai yaitu True dan False. Sedangkan type data None Type hanya memiliki satu nilai yaitu None, bukan seperti bool. None Type bisa digunakan untuk mendefinisikan sebuah nilai null, atau sesuatu yang tidak memiliki nilai.

Hal-hal penting dalam melakukan perhitungan dengan type-type data tertentu:

- Sebuah nilai int bila dibagi dengan int maka akan menghasilkan nilai bertype float, meskipun habis dibagi.
- Penjumlahan dan pengurangan antara int dengan float akan menghasilkan float.

List

List adalah sebuah type data di Python yang berupa array, namun isi array ini dapat terdiri dari berbagai macam type data bahkan array di dalam array. Penulisannya menggunakan kurung [ ]. Beberapa contoh:

myArray = [0, 87, 23, 1, 34, 19]

myArray = ['hello', 3.2, [9, 'hi']]

print(myArray[2][1])

akan menampilkan: 'hi'

for i in myArray:

```
    print(i)
```

akan menampilkan seluruh isi myArray

Untuk menambahkan elemen pada list dapat menggunakan method .append(), contoh:

```
myArray.append('7.6')
```

maka isi myArray saat ini adalah ['hello', 3.2, [9, 'hi'] , '7.6']

Untuk menghapus sebuah elemen dapat menggunakan method .remove(), contoh:

```
myArray.remove('hello')
```

apabila ada nilai elemen yang sama, perintah remove hanya menghapus satu saja, yang pertama. Untuk menghapus sebuah elemen list pada nomor index tertentu bisa dengan cara:

```
del(myArray[3])
```

Untuk menghapus elemen suatu posisi dan seluruh elemen sisanya maka bisa menggunakan cara:

```
del(myArray[3:])
```

seluruh elemen posisi index 3 (elemen ke-4) dan sisanya akan dihapus semua.

```
del(myArray[1:3])
```

akan menghapus elemen myArray[1] dan myArray[2]

```
del(myArray[:3])
```

akan menghapus elemen myArray[0], myArray[1] dan myArray[2]

Fungsi .pop() dapat juga digunakan untuk mengeluarkan elemen dari list, dimana argumen diisi dengan angka index dari array yang akan dikeluarkan.

```
t = [ 'a', 'b', 'c' ]
x = t.pop(1)
```

maka t akan menjadi ['a', 'c'] dan variable x memiliki nilai 'b'.

Gunakan fungsi len untuk mengetahui jumlah elemen sebuah list:

```
len(myArray)
```

Data type string sangat erat hubungannya dengan type list, selain setiap karakter pada string dapat dijadikan elemen-elemen pada list, bisa juga kata-kata pada kalimat dipisah ke elemen-elemen list dengan cara:

```
s = "Selamat datang di kelas"
t = s.split()
```

maka t akan menjadi: ['Selamat', 'datang', 'di', 'kelas']

parameter split dapat diisi dengan karakter pemisah kata, contoh diatas adalah defaultnya yaitu spasi.

```
s = "Selamat-datang-di-kelas"
t = s.split("-")
```

maka t akan menjadi: ['Selamat', 'datang', 'di', 'kelas']

Untuk menggabungkan kembali menjadi satu kalimat bisa gunakan method .join() didepan karakter pemisahnya, misalnya spasi:

```
pemisah = " "
pemisah.join(t)
```

maka t akan kembali menjadi: "Selamat datang di kelas"

Tuple

Tuple mirip dengan list namun sifatnya immutable, artinya elemen yang sudah ada tidak dapat dimodifikasi. Penulisan tuple menggunakan kurung ( ). Contoh:

```
myTuple = ( 8, 34, 'hi', 3.5 )
```

Hal yang penting dalam tuple adalah apabila elemen yang dimiliki hanya ada satu, maka harus diakhiri dengan tanda koma, contoh:

```
myTuple = ('hello',)
print(myTuple[0])
```

akan menampilkan 'hello'

Karena apabila tidak menggunakan koma maka akan dianggap isi elemen tuple tersebut adalah h, e, l, l dan o.

Untuk menambahkan elemen tuple bisa dengan cara konversi ke list terlebih dahulu, tambahkan elemen baru dengan append, lalu dikembalikan lagi ke tuple:

```
myTuple = ('hello', 23, 5.1)
myTuple = list(myTuple)
myTuple.append('The End')
myTuple = tuple(myTuple)
```

Set

Set merupakan kumpulan elemen yang unik, dalam sebuah set tidak ada elemen yang sama. Penulisan menggunakan kurung { }.

```
mySet = { 5, 9, 12, 7, 8 }
```

Gunakan method .add() untuk menambah elemen set:

```
mySet.add(4)
```

Apabila elemen yang ditambahkan sudah ada dalam set maka akan diabaikan.

Gunakan .remove() atau .discard() untuk menghapus sebuah elemen set. Penggunaan .discard() tidak akan memberi pesan error apabila elemen yang berusaha dihapus tidak ditemukan dalam set.

Dict

Dict merupakan kumpulan elemen yang terdiri dari dua nilai yaitu key dan value, atau pasangan key sebagai identitas unik, dan value sebagai nilainya. Mirip seperti associative array. Penulisan dict menggunakan kurung { } seperti pada set namun tiap elemennya terdiri dari dua bagian yang dipisah dengan tanda ':'

```
myDict = { 'pertama' : 13, 'kedua': 89, 'ketiga': 14 }
```

```
for k, v in myDict.items():
  print("Key:", k, "Value:", v)
```

akan mencetak semua key dengan valuenya. Apabila dibutuhkan hanya mencetak keynya saja atau valuenya saja maka bisa gunakan method .keys() dan .values():

```
for k in myDict.keys():
  print(k)

for v in myDict.values():
  print(v)
```

Zip function

Fungsi zip dapat menggabungkan dua buah list menjadi pasangan key-value dict maupun elemen list baru atau tuple. Contoh:

```
s = "abc"
t = [ 1, 2, 3 ]
u = list(zip(s, t))
```

maka u menjadi: [('a', 1), ('b', 2), ('c', 3)]

Bisa dicoba u = dict(zip(s, t))

```
list(zip("Budi", "Eva"))
```

akan menjadi [('B', 'E'), ('u', 'v'), ('d', 'a')] dan 'i' akan diabaikan karena tidak memiliki pasangan.

Loop dengan for

Loop dengan for telah dijumpai pada contoh-contoh sebelumnya namun untuk sebuah pembacaan seluruh nilai dalam list atau dict. Apabila menggunakan loop dengan for untuk sebuah range nilai maka dapat menggunakan cara:

```
for i in range(0,10,1):
  print(i)
```

akan mencetak 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 dan angka 1 di paling kanan adalah step, bisa diabaikan karena defaultnya adalah 1. Untuk angka awal defaultnya 0, maka 0 bisa tidak ditulis juga.

Maka for loop yang arahnya mundur bisa menggunakan cara:

```
for i in range(5, 0, -1):
  print(i)
```

Hasilnya adalah:

5

4

3

2

1

Loop dengan while

Loop dengan menggunakan while membutuhkan suatu kondisi true agar loop tersebut berjalan:

```
j = 1
while j <= 5:
    print(j)
    j += 1
```

Maka akan mencetak:

1

2

3

4

5

karena setelah j bernilai 6 kondisi while menjadi false sehingga loop tidak dilanjutkan.

If dan Else

Mengeksekusi sesuatu berdasarkan suatu kondisi maka dibutuhkan perintah if untuk mengevaluasi kondisi tersebut:

```
c = 5
if c > 0:
    print("c lebih besar dari 0")
else:
    print("c tidak lebih besar dari 0")
```

Kondisi c > 0 bernilai true maka lingkup didalamnya akan dieksekusi, yaitu perintah cetak "c lebih besar dari 0", sedangkan lingkup dalam else dieksekusi apabila kondisi pada if bernilai false, misal c = -2.

Perintah elif dapat digunakan apabila dalam else haru melakukan pengecekan lagi menggunakan if:

```
nilai = 75
if nilai >= 86:
    print("Nilai: A")
elif nilai >= 76:
    print("Nilai: B+")
elif nilai >= 69:
    print("Nilai: B")
elif nilai >= 61:
    print("Nilai: C+")
elif nilai >= 56:
    print("Nilai: C")
elif nilai >= 41:
    print("Nilai: D")
else:
    print("Nilai: E")
```

Maka akan menampilkan Nilai: B karena dua kondisi awal tidak terpenuhi atau false.

Bab 3 - Fungsi Matematika

Python secara internal sudah dilengkapi dengan fungsi-fungsi matematika. Untuk menggunakan fungsi ini diharuskan melakukan import math. Contoh penggunaan:

```
import math
print(math.pi)
```

akan menampilkan nilai pi yaitu 3.141592653589793

```
import math
print(math.sqrt(9))
```

akan menampilkan angka 3.0 dimana sqrt adalah akar.

```
import math
print(math.pow(5, 3))
```

akan menampilkan angka 125.0 dimana pow adalah pangkat, atau 5 pangkat 3 untuk contoh diatas.

Dalam jaringan komputer banyak menggunakan bilangan berbasis 2, 8 dan 16 atau disebut dengan binary, octal dan hexadecimal. Secara internal Python sudah menyediakan fungsi untuk seluruh basis bilangan tersebut, dengan demikian konversinya dari dan ke desimal dapat dilakukan. Contoh:

```
a = 54
bilangan = bin(a)
print(bilangan)
```

akan tercetak: 0b110110

Bilangan binary akan diawali dengan '0b' untuk menandakan bahwa angka-angka setelahnya adalah angka binary. secara keseluruhan variable bilangan diatas ber-type string. Untuk dikonversikan kembali ke desimal:

```
desimal = int(bilangan, 2)
```

maka desimal akan menjadi integer dengan nilai 54. Parameter kedua di dalam int menandakan basis bilangan, 2 untuk binary.

Contoh untuk octal:

a = 54

okt = oct(a)

maka a akan bernilai: 0o66, karena bilangan octal pada Python diawali dengan '0o', dan untuk konversi kembali ke integer dilakukan hal serupa seperti diatas dengan nilai parameter kedua = 8:

desimal = int(okt, 8)

print(desimal)

akan tercetak kembali: 54 dan ber-type integer

Contoh hexadecimal:

a = 54

heksa = hex(a)

print(heksa)

akan tercetak 0x36, karena hexadecimal di Python diawali dengan '0x' dan juga secara keseluruhan ber-type string, untuk konversi kembali ke desimal:

desimal = int(heksa, 16)

Fungsi lain yang berhubungan dengan character set adah chr() dan kebalikannya ke integer adalah ord(). Contoh untuk karakter huruf 'A' memiliki angka ASCII 68, untuk mendapatkannya:

huruf = chr(68)

print(huruf)

maka huruf akan menampilkan 'A'

Kebalikannya:

angkaASCII = ord(huruf)

maka angkaASCII bernilai 68

Fungsi Random

Untuk menghasilkan angka random di Python perlu menggunakan library random, dimana library ini tidak berhubungan dengan math, namun angka random seringkali digunakan dengan operasional yang berhubungan dengan matematika atau perhitungan.

```
import random
print(random.randrange(2, 8))
```

maka akan muncul angka acak antara 2 dan 8, termasuk angka 2 dan 8

untuk method randrange ada 3 parameter, yang ke tiga adalah step, defaultnya adalah 1, maka contoh diatas bisa tidak dituliskan (2, 8, 1), artinya step adalah default yaitu 1.

Method shuffle bisa digunakan untuk mengacak urutan dari sebuah list, contoh:

```
import random
items = [ 2, 1, 5, 7, 11, 21, 9, 4 ]
random.shuffle(items)
print(items)
```

Bab 4 - Function

Membuat fungsi di Python diawali dengan kata def lalu kemudian nama fungsi dan parameter-parameter bila dibutuhkan:

```
def tambah(a, b):
    return a + b

print(tambah(2, 5))    # akan menampilkan 7
k = tambah(6, 9)       # k memiliki nilai 15
```

Dengan adanya return maka fungsi tersebut akan selalu mengembalikan sebuah nilai. Nilai tersebut bisa diarahkan ke sebuah variable atau langsung ke output dengan perintah print.

Apabila ada sebuah fungsi tanpa parameter maka pemanggilannya tetap menggunakan tanda kurung meskipun tidak ada yang ditulis didalamnya:

```
def fun():
    print("Fun!")

fun()
```

Function juga dapat memanggil function yang lain maupun function itu sendiri. Perlu diperhatikan bahwa recursive dapat membebani resource komputer yang digunakan, jadi perlu diperhatikan codingnya agar tidak ada perhitungan yang berlebihan, misalnya melebihi kapasitas data type yang digunakan.

Apabila jumlah parameter fungsi bersifat dinamis, maka satu cara bisa menggunakan data type list untuk argumen dari parameter tersebut:

```
def fun(mylist):
    for i in mylist:
        print(i)

thelist = [ "tree", "house", "plane" ]
fun(thelist)
```

Maka fungsi tersebut akan mencetak:

tree

house

plane

Namun penggunaan list seperti itu mengharuskan parameter yang diisi harus memiliki type list. Apabila jumlah parameter yang benar-benar dinamis maka bisa menggunakan cara:

```
def fun(*mylist):
    for i in mylist:
        print(i)

fun(7, 9.0, "hello")
fun("Hi")
fun()
```

Maka outpunya adalah:

7

9.0

hello

Hi

Dengan cara tersebut maka jumlah parameter menjadi bebas bahkan tanpa parameter sekalipun, dan tidak akan mengakibatkan error.

Bisa juga fungsi dengan parameter yang bersifat deskriptif seperti dictionary, sehingga setiap nilainya berpasangan, ada key dan value:

```
def fun(**mylist):
    for i, j in mylist.items():
        print(i + " isinya: " + j)

fun(kamar="meja", dapur="kompor", taman="pot")
```

Maka akan menampilkan:

kamar isinya meja

dapur isinya kompor

taman isinya pot

Bab 5 - Membaca dan menulis file

Dengan Python memungkinkan untuk membaca maupun menulis file pada sistem operasi dimana interpreter Python tersebut berada. Perlu diperhatikan bahwa Python dapat mengakses ke file system secara langsung tanpa persetujuan user, maka perlu dilakukan secara hati-hati agar tidak merusak file-file yang sudah ada.

```
namafile = "teks.txt"
h = open(namafile, "r")
t = h.read()
w = t.split()
```

Contoh diatas akan mengisi variable w dengan kata-kata yang ada pada file teks.txt.

Apabila ingin membaca baris per baris pada sebuah file teks maka bisa dilakukan:

```
namafile = "teks.txt"
h = open(namafile, "r")
t = h.readlines()
for i in t:
    print(i, end="")
```

Loop diatas akan menampilkan isi file tersebut baris per baris, cara ini cocok untuk pembacaan data dari suatu file teks yang bisa disimpan ke list atau struktur data yang lain.

Catatan mode akses file:

mode "r" adalah read, "w" adalah write, "a" adalah append, "rb" adalah read binary file.

```
namafile = "teks.txt"
n = open(namafile, "w")
n.write("Saya suka Python")
n.close()
```

Karena contoh diatas menggunakan "w", maka yang ditulis oleh method write akan menimpa apapun yang sudah ada di file teks.txt, atau akan membuat file baru apabila

belum ada. Bila menggunakan "a" maka write akan menambahkannya di bagian akhir file tersebut, tanpa menimpa yang ada sebelumnya.

Bab 6 - Regular Expression

Python dilengkapi kemampuan regular expression yang sangat membantu dalam manipulasi dan pencarian teks berdasarkan pola-pola. Python menggunakan regular expression seperti yang ada di pemrograman lain misalnya Perl.

Untuk bekerja dengan regular expression dibutuhkan import re. Contoh pertama ini akan melaporkan bahwa format penulisan NRP benar atau salah. Benar apabila diawali dengan sebuah huruf dan diikuti dengan 8 angka, selain itu akan salah:

```
import re
nrp = input("Masukkan NRP: ")
match = re.search(r'^[a-zA-Z]\d{8}$', nrp)
try:
    print("Format NRP benar:", match.group())
except:
    print("Format NRP salah")
```

Contoh berikutnya adalah pengecekan format nomor telepon dimana syaratnya adalah diawali dengan +62 lalu diikuti dengan 9 sampai dengan 11 angka:

```
import re
n = input("Masukkan Nomor hp: ")
match = re.search(r'\+62\d{9,11}$', n)
try:
    print("Format Nomor hp benar:", match.group())
except:
    print("Format Nomor hp salah")
```

Pola-pola regular expression

Pola-pola dibawah ini merupakan pola regular expression yang banyak digunakan, selain itu masih ada banyak lagi namun penggunaannya tidak sesering yang disebutkan dibawah ini:

. (titik) akan match 1 karakter apa pun kecuali newline '\n'

+ akan match 1 atau lebih pola di kirinya, contoh k+ maka akan match apabila ada 1 k atau lebih

* akan match 0 atau lebih pola di kirinya

? akan match 0 atau 1 pola di kirinya

\d akan match angka desimal, atau sama dengan pola [0-9], arti kurung kotak (brackets) adalah karakter-karakter didalamnya bila ditemukan maka akan match, tanda - merupakan tanda range

\w akan match word character, yaitu karakter [a-zA-Z0-9_], huruf a sampai z huruf kecil maupun besar, angka 0 sampai 9, dan karakter garis bawah (underscore)

^ awal baris

$ akhir baris

\n newline

\r return

\t tab

\s akan match whitespace character, yaitu spasi, tab, return, newline

\S akan match non-whitespace character

\ special character, digunakan untuk menyatakan sebuah karakter yang sesuai tanpa dianggap sebuah pola regular expression, contoh: \$ maka akan match ke symbol dolar $, bukan akhir baris

Untuk pencarian sebuah pola ke dalam sebuah string yang cukup panjang, maka akan ada kemungkinan match yang ditemukan ada lebih dari satu. Agar bisa menangkap seluruh pola yang ditemukan dalam sebuah string maka dapat menggunakan method findall(), contoh:

```
import re
kalimat = "hai jay@email.net dan juga kane@kirim.com mohon kemari"
temuan = re.findall(r'[\w\.-]+@[\w\.-]+', kalimat)
for i in temuan:
    print(i)
```

Variable temuan diatas akan berupa list yang berisi beberapa pola matching yaitu: ['jay@email.net', 'kane@kirim.com'] karena regular expression memiliki pola word character, tanda titik dan minus lalu diikuti '@' dan kembali lagi word character, tanda titik dan minus dimana akan match sebuah username lalu @ kemudian diikuti dengan nama domain sebuah alamat email. Kemudian ada tanda + yang artinya group matching tersebut bisa muncul 1 atau lebih karakter.

Apabila antara username dan domainnya ingin dipisah, maka regular expressionnya menjadi:

```
temuan = re.findall(r'([\w\.-]+)@([\w\.-]+)', kalimat)
```

Dimana temuan akan berisi [('jay', 'email.net'), ('kane', 'kirim.com')]

Bila dilihat bedanya dengan sebelumnya adalah penambahan tanda kurung dimana pola tersebut akan dikelompokkan (grouping), maka untuk memecahkan datanya menggunakan loop:

```
for t in temuan:
    print(t[0])
    print(t[1])
```

Dimana t[0] akan mencetak username, sedangkan t[1] adalah nama domain email. Setiap elemen dalam list temuan akan disimpan dalam sebuah tuple.

Pencarian pola di sebuah file teks juga dapat dilakukan, contoh:

```
import re
f = open('file.txt', 'r')
s = re.findall(r'pola yang dicari', f.read())
```

Untuk substitusi atau find and replace pada regular expression Python menggunakan method sub. Misalnya pada contoh diatas pada kalimat terdapat dua alamat email, apabila domain email tersebut diganti dengan sesuatu yang baru maka:

```
import re
kalimat = "hai jay@email.net dan juga kane@kirim.com mohon kemari"
print(re.sub(r'([\w\.-]+)@([\w\.-]+)', r'\1@domainbaru.com', kalimat))
```

Maka akan menjadi: "hai jay@domainbaru.com dan juga kane@domainbaru.com mohon kemari" karena \1 adalah group pertama yaitu yang ada dalam kurung yang pertama dimana letak username jay dan kane berada. Kemudian \2 adalah nama domainnya. \1 dicantumkan pada bagian substitusinya, ditambah dengan string @domainbaru.com, maka dari itu seluruh alamat email yang match akan berubah domainnya.

Untuk mengabaikan huruf kecil dan besar, dapat ditambahkan parameter flags=re.I:

```
print(re.sub(r'([\w\.-]+)@([\w\.-]+)', r'\1@domainbaru.com', kalimat, flags=re.I))
```

Flag re yang lain dapat dilihat di dokumentasi Python:

https://docs.python.org/3/library/re.html#contents-of-module-re

Bab 7 - Object Oriented

Python merupakan bahasa pemrograman yang sangat fleksibel karena dapat dituliskan dalam bentuk object oriented maupun tidak. Tentunya object oriented pada Python harus digunakan untuk aplikasi-aplikasi yang banyak bekerja pada data karena wujud informasi yang sangat beragam strukturnya. Selain itu juga banyak library maupun framework yang tersedia pemakaiannya harus dengan object oriented, contohnya seperti Flask dan Django.

Dalam pemrograman berorientasi objek (object oriented programming) selalu dikenal yang namanya class. Dengan sebuah class kita bisa membuat sebuah kerangka objek dimana didalamnya terdiri dari property atau variable, dan method atau fungsi. Contoh sebuah class di Python:

```
class Karyawan:
    "Ini sebuah class Karyawan"
    hitungKaryawan = 0

    def __init__(self, nama, gaji):
        self.nama = nama
        self.gaji = gaji
        Karyawan.hitungKaryawan += 1

    def tampilJumlah(self):
        print("Jumlah karyawan: %d" % Karyawan.hitungKaryawan)

    def tampilKaryawan(self):
        print("Nama:", self.nama, "\b, Gaji:", self.gaji)

k1 = Karyawan("Abi", 4000)
k2 = Karyawan("Dia", 5000)
k1.tampilKaryawan()
k2.tampilKaryawan()
k1.tampilJumlah()     # akan cetak: 2 karena ada k1 dan k2
k2.tampilJumlah()     # akan tetap cetak: 2 karena jumlah objek yand dibuat ada 2
print ("Jumlah karyawan: %d" % Karyawan.hitungKaryawan)
```

hitungKaryawan merupakan sebuah static variable yang digunakan sebagai counter untuk menjumlahkan jumlah objek yang sudah dibuat menggunakan class tersebut. Penulisan lengkapnya diawali dengan nama class, maka ditulis Karyawan.hitungKaryawan.

String dibawah nama class adalah dokumentasi keterangan mengenai class tersebut, bisa diisi dengan informasi penggunaan class dan lain sebagainya.

def __init__ merupakan sebuah konstruktor, bagian dari overloading methods, yaitu sebuah method atau function yang akan dieksekusi pada saat sebuah objek baru dibuat. Contoh overloading methods lainnya:

__del__(self): destructor, akan menghapus objek, contoh pemakaian: del namaobjek

__repr__(self): menampilkan isi objek yang mungkin tidak dalam string

__str__ (self): menampilkan isi objek dalam bentuk string

__repr__ maupun __str__ dapat menampilkan isi yang sama, tergantung objek.

Apabila dibutuhkan sebuah atribut atau variable pada sebuah object, bisa menggunakan cara:

```
k1.umur = 31
k2.umur = 29
```

Untuk mengecek sebuah objek memiliki atribut tersebut dan beberapa aksi lainnya:

```
hasattr(k1, 'umur')     # akan memberikan nilai true apabila k1 memiliki nilai umur
getattr(k1, 'umur')     # akan memberi return nilai umur dari objek k1
delattr(k1, 'umur')     # menghapus atribut umur dari objek k1
setattr(k1, 'umur', 25)         # mengganti nilai atribut umur k1 menjadi 25
```

Overriding methods

Overriding merupakan cara menumpuk method yang ada pada class atasnya (parent), sehingga apabila ada sebuah object child, apabila method yg dieksekusi memiliki nama yang sama dengan yang ada di parent, yang dieksekusi adalah method yang ada di childnya:

```
class Parent:
    def methodKu(self):
        print("Memanggil method di parent")

class Child(Parent):
```

```
    def methodKu(self):
        print("Memanggil method di child")


c = Child()
c.methodKu()
```

maka method ini akan mengeksekusi methodKu() yang ada di dalam class Child

Untuk child class yang masih menggunakan constructor parentnya, bisa dilihat pada contoh ini dimana parent classnya adalah mobil, childnya adalah mobil listrik, dimana parent mobil merupakan bentuk mobil secara umum sedangkan childnya ada satu perbedaan yaitu baterai:

```
class Car(object):
    kondisi = "new"


    def __init__(self, model, warna, kpl):
        self.model = model
        self.warna = warna
        self.kpl   = kpl

class ElectricCar(Car):
    def __init__(self, battery_type, model, warna, kpl):
        self.battery_type=battery_type
        super(ElectricCar, self).__init__(model, warna, kpl)

car = ElectricCar('battery', 'ford', 'golden', 10)
oldcar = Car('toyota', 'red', 12)
print(car.__dict__)
print(oldcar.__dict__)
```

Maka outputnya:

```
{'battery_type': 'battery', 'model': 'ford', 'warna': 'golden', 'kpl': 10}
{'model': 'toyota', 'warna': 'red', 'kpl': 12}
```

Jadi mobil secara umum ada variable model, warna dan kpl (kilometer per liter), sedangkan ElectricCar ada variable battery_type selain variable lain seperti yang ada di parent.

__dict__ adalah bagian dari built-in class attributes, yaitu untuk menampilkan isi sebuah objek dalam bentuk dictionary, contoh lainnya adalah:

__doc__ : untuk menampilkan dokumentasi, string yang ditulis dibawah class

__name__ : menampilkan nama class

__module__ : nama modul class, akan muncul __main__ karena class utama/awal

Bab 8 - Pemrograman Jaringan

Python tidak hanya dibuat untuk pemrosesan teks saja, namun juga dapat membuat aplikasi yang berhubungan dengan jaringan komputer.

Socket Programming

Pemrograman socket merupakan dasar pemrograman untuk client dan server. Socket adalah istilah dalam jaringan komputer yang melibatkan alamat IP dan port yang digunakan, begitu pula jenis transportnya yaitu TCP atau UDP.

Sebuah socket dapat diinisialisasi dengan socket method untuk menentukan transport TCP atau UDP:

```
import socket
s = socket.socket(socket_family, socket_type)
```

dimana socket_family bisa: AF_UNIX atau AF_INET

socket_type bisa: SOCK_STREAM untuk TCP, atau SOCK_DGRAM untuk UDP.

Maka untuk TCP dapat dituliskan:

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

Contoh sebuah client dan server TCP dengan menggunakan Python:

Server TCP:

```
import socket
s = socket.socket()
host = socket.gethostname()
port = 12345
s.bind((host, port))
s.listen(5)
while True:
    c, addr = s.accept()
    print("Mendapatkan koneksi dari: ", addr)
    c.send("Selamat terkoneksi".encode())
    c.close()
```

Client TCP:

```
import socket
s = socket.socket()
host = socket.gethostname()
port = 12345
s.connect((host, port))
print(s.recv(1024).decode())
s.close()
```

Port dapat ditentukan sendiri, asalkan antara client dan server sama, dan diatas 1024 karena kita menjalankan client dan server ini dari user non-root. Apabila address family tidak disebutkan, maka secara default socket tersebut menggunakan TCP.

Cara menjalankan server: python3 server.py

Setelah server aktif, maka Python pada terminal tersebut akan selalu aktif dan menunggu adanya client yang melakukan koneksi. Untuk menjalankan client perlu dieksekusi dari terminal lain, maka buatlah sesi terminal baru untuk menjalankan client.

Contoh UDP:

Server UDP:

```
import socket
import sys

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

server_address = ('localhost', 10000)
print('aktif di {} port {}'.format(*server_address))
sock.bind(server_address)

while True:
    print('\nmenunggu pesan masuk')
    data, address = sock.recvfrom(4096)

    print('received {} bytes from {}'.format(
        len(data), address))
    print(data)
```

```
    if data:
        sent = sock.sendto(data, address)
        print('mengirim {} bytes kembali ke {}'.format(
            sent, address))
```

Client UDP:

```
import socket
import sys

# Create a UDP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

server_address = ('localhost', 10000)
message = 'Ini adalah sebuah pesan.  Akan diulang.'.encode()

try:

    # Send data
    print('sending {!r}'.format(message))
    sent = sock.sendto(message, server_address)

    # Receive response
    print('waiting to receive')
    data, server = sock.recvfrom(4096)
    print('received {!r}'.format(data.decode()))

finally:
    print('closing socket')
    sock.close()
```

Library ipaddress

Python menyediakan library yang sangat membantu kita untuk melakukan kalkulasi dan penulisan IPv4 maupun IPv6, yaitu ipaddress. Library yang harus diimport adalah ipaddress:

```
import ipaddress
print(ipaddress.ip_network('192.168.0.0/24'))
```

Method ip_network akan memberikan return IPv4Network('192.168.0.0/24') agar dikenal sebagai alamat network IP, namun bila di print masih bisa menampilkan IP network tersebut.

Method hosts bisa memberikan list berisi seluruh alamat IP pada sebuah network, contoh:

```
list(ip_network('192.168.2.0/29').hosts())
```

akan memberikan nilai:

```
[IPv4Address('192.168.2.1'), IPv4Address('192.168.2.2'),
 IPv4Address('192.168.2.3'), IPv4Address('192.168.2.4'),
 IPv4Address('192.168.2.5'), IPv4Address('192.168.2.6')]
```

Method .subnets() akan memberi list berupa subnet yang bisa dipecah lagi, bila tanpa parameter maka akan dibagi dua, misal asalnya adalah network /24 maka list akan berisi 2 buah network /25. Apabila dari /24 akan dibagi menjadi 4 buah /26 parameternya adalah: prefixlen_diff=2.

```
list(ip_network('192.0.2.0/24').subnets())
```

Hasilnya:

```
[IPv4Network('192.0.2.0/25'), IPv4Network('192.0.2.128/25')]
```

```
list(ip_network('192.0.2.0/24').subnets(prefixlen_diff=2))
```

Hasilnya:

```
[IPv4Network('192.0.2.0/26'), IPv4Network('192.0.2.64/26'),
 IPv4Network('192.0.2.128/26'), IPv4Network('192.0.2.192/26')]
```

Untuk mengetahui bagian network yang lebih besar, yaitu kebalikannya subnet, menggunakan .supernet():

ip_network('192.0.2.0/24').supernet()

Outputnya:

IPv4Network('192.0.2.0/23')

Dengan parameter:

ip_network('192.0.2.0/24').supernet(prefixlen_diff=2)

Outputnya:

IPv4Network('192.0.0.0/22')

Bisa juga kita cek apakah suatu network masuk dalam bagian network lain:

a = ip_network('192.168.1.0/24')

b = ip_network('192.168.1.128/30')

b.subnet_of(a)

Maka jawabannya adalah True, karena network 192.168.1.128/30 ada sebagai bagian dari network 192.168.1.0/24.

Dokumentasi library ipaddress dapat dilihat di:

https://docs.python.org/3/library/ipaddress.html

Urllib dan JSON

Untuk berinteraksi dengan sebuah web service dapat menggunakan library urllib, dimana service yang ada berjalan dengan protocol http atau https. Sedangkan JSON (Javascript Object Notation) adalah format data yang sering digunakan dan dijumpai pada API (Application Programming Interface) atau web service publik maupun privat.

Contoh sebuah pembacaan data dari sebuah API fixer.io, mohon bisa mendaftar sendiri untuk mendapatkan API key:

import urllib, json

import urllib.request

url = "http://data.fixer.io/api/latest?access_key=*tulis API key disini*"

response = urllib.request.urlopen(url)

data = json.loads(response.read())

print (data)

print (data["rates"])

```
print (data["rates"]["IDR"])
```

Apabila ada data berupa array, maka isi dari array dapat diambil dengan loop, contoh:

```
response = urllib.request.urlopen(url)
data = json.loads(response.read())
for a in data["konten"]:
  print(a["nama"]+" "+a["email"])
```

CGI (Common Gateway Interface)

Apabila kita ingin menggunakan Python sebagai back-end sebuah aplikasi web, maka salah satu caranya adalah menempelkan Python pada sebuah web server, misalnya Apache, sebagai CGI. Maka apabila ada request dari web client yang menuju ke sebuah script Python di URL nya, web server yang akan mengeksekusi, dan pada umumnya output dari script Python tersebut akan diteruskan ke web client.

Agar web server, contoh disini menggunakan Apache, dapat mengizinkan script Python untuk dieksekusi, maka perlu diberikan sebuah konfigurasi, contohnya adalah directory public_html untuk seluruh user, maka konfigurasinya adalah:

```
<Directory "/home/*/public_html/cgi-bin">
  AddHandler cgi-script .py .cgi
  AllowOverride None
  Options ExecCGI
  Order allow, deny
  Allow from all
</Directory>
```

Restart Apache web server agar konfigurasi tersebut berlaku. Setelah itu, di dalam directory user, di dalam public_html dapat dibuat directory baru dengan nama cgi-bin, dan seluruh file script Python disini dapat dieksekusi melalui web browser melalui alamat URL. Apabila cgi-bin ada di Linux atau macOS, maka perlu ada informasi lokasi interpreter Python dan juga file script tersebut harus bisa dieksekusi, yaitu memiliki permission untuk execute. Perhatikan permission yang diberikan dan script yang dibuat agar tidak berakibat ke masalah keamanan, karena Python bisa memiliki hak akses sampai ke system dan file. Contoh script CGI:

```
#!/usr/bin/python3
print("Content-type:text/html\n")
```

```
print("<html>")
print("<head><title>Python CGI</title></head>"
print("<body>")
print("<p>test</p>")
print("</body>")
print("</html>")
```

Gunakan Content-type:text/html agar seluruh output dengan print setelahnya akan dianggap sebagai konten web. Apabila CGI ini digunakan sebagai web service maka sesuaikan Content-type dengan data yang akan dioutputkan, misalnya: Content-type:application/json.

Python CGI dapat menerima input melalui get dan post method dan cara mengambilnya adalah dengan cara:

```
#!/usr/bin/python3
import cgi, cgitb

form = cgi.FieldStorage()

first_name = form.getvalue('first')
last_name  = form.getvalue('last')

print ("Content-Type: text/html\n")
print ("<!doctype html")
print ("<html>")
print ("<head>")
print ("<title>Hello - Second CGI Program</title>")
print ("</head>")
print ("<body>")
print ("<h2>Hello %s %s</h2>" % (first_name, last_name))
print ("</body>")
print ("</html>")
```

Method getvalue() diisi dengan nama input form dari file html yang memanggil CGI tersebut, bisa berasal dari text input, radio button, checkbox, dan lain sebagainya.

Untuk menerima file dari form, maka form yang dibuat harus memiliki atribut enctype, contohnya:

```
<form enctype="multipart/form-data" action="cgi-bin/upload.py" method="post">
  <input type="file" name="filename">
  <inpuy type="submit">
</form>
```

Dan file CGInya:

```
#!/usr/bin/python3
import cgi, os
import cgitb

cgitb.enable()

form = cgi.Fieldstorage()
fileitem = form["filename"]

if fileitem.filename:
  fn = os.path.basename(fileitem.fileitem)
  open("/home/user/files/" + fn, "wb").write(fileitem.file.read())
  message = 'The file "' + fn + '" was uploaded successfully'
else:
  message = "No file was uploaded"

print ("Content-Type: text/html\n")
print("<html><head></head><body>"+message+"</body></html>\n")
```

Perlu dipastikan bahwa directory tersebut dapat diisi dengan file hasil upload, perhatikan file dan folder permissions. Agar aman, dapat menggunakan suexec atau menambahkan module mpm-itk pada Apache agar tiap home directory hanya bisa diakses oleh user yang bersangkutan.

DNS

DNS atau Domain Name System adalah sebuah sistem di Internet untuk menterjemahkan nama domain ke alamat IP dan sebaliknya. Sebuah server DNS juga dapat menjadi sebuah tempat dimana konfigurasi sebuah domain berada, yang bisa berisi seluruh subdomain yang ada dan record lainnya seperti alamat MX, CNAME untuk alias, AAAA untuk alamat IPv6 sebuah host, dan lain sebagainya.

Untuk menggunakan DNS dengan bantuan library di Python dapat menggunakan library dnspython, dengan install pip3 install dnspython, atau apabila menggunakan Linux distribusi Ubuntu bisa dengan sudo apt install python-dnspython.

Contoh melakukan query MX suatu domain, MX adalah penunjukan alamat tujuan untuk email:

```
import dns.resolver

answers = dns.resolver.resolve('domain.co.id', 'MX')
for rdata in answers:
    print('Host', rdata.exchange, 'has preference', rdata.preference)
```

Untuk entry record DNS lainnya bisa menggunakan RRset (Resource Record Set):

```
import dns.resolver

HOST = "domain.co.id"
for qtype in ['A', 'AAAA', 'CNAME', 'MX', 'NS']:
  answers = dns.resolver.resolve(HOST, qtype, raise_on_no_answer=False)
  print(answers.rrset)
```

Maka output akan menunjukkan record A, AAAA, CNAME, MX dan NS untuk host yang diinputkan.

Record PTR adalah sebuah informasi nama domain berdasarkan IP yang diinputkan, atau istilahnya reverse, bisa didapatkan dengan cara:

```
import dns.resolver
import dns.reversename

REV = "202.123.20.8"          # IP hanya sebagai contoh, gunakan public IP
```

```
hrev = dns.reversename.from_address(REV)
domain = dns.resolver.resolve(hrev, "PTR")[0]
print(domain)
```

SMTP

SMTP atau Simple Mail Transfer Protocol digunakan untuk mengirim pesan email ke sebuah tujuan SMTP server yang lain. Python menggunakan library smtplib untuk bisa menjadi sebuah SMTP sender, berguna untuk mengirim email ke sebuah alamat:

```
import smtplib

sender = 'from@mydomain.com'
receivers = ['to@todomain.com']

message = """From: From Person <from@mydomain.com>
To: To Person <to@todomain.com>
Subject: SMTP e-mail test

This is a test e-mail message.
"""

try:
   smtpObj = smtplib.SMTP('smtp.mydomain.com')
   smtpObj.sendmail(sender, receivers, message)
   print ("Successfully sent email")
except smtplib.SMTPException as e:
   print (str(e))
```

Pesan yang ditulis dengan multiple line harus berisi header email seperti From:, To: dan Subject: untuk mengurangi kemungkinan dianggap sebagai email spam.

POP3 dan IMAP

Protocol POP3 dan IMAP sudah jarang digunakan setelah email berbasis web menjadi popular meninggalkan mail client berbasis desktop. Namun kedua protokol ini dapat dimanfaatkan proses authenticationnya sebagai alat authentication untuk aplikasi apa

pun. Setelah authentication berhasil tidak perlu melihat inbox yang ada, karena hanya membutuhkan proses authentication saja dimana hanya dibutuhkan benar atau salahnya password yang dimasukkan.

Contoh POP3, server harus ada server POP3 pada port 110, contohnya bisa menggunakan Dovecot, untuk script Pythonnya:

```
import getpass, poplib

M = poplib.POP3('pop3.domain.id')
M.user(getpass.getuser())
M.pass_(getpass.getpass())
numMessages = len(M.list()[1])
for i in range(numMessages):
    for j in M.retr(i+1)[1]:
        print (j)
```

getpass adalah sebuah fungsi untuk menyembunyikan input yang diketikkan melalui keyboard, atau istilahnya tidak di-echo-kan kemabli ke layar. Proses diatas akan mengakses hingga ke mailbox, apabila kita hanya membutuhkan authenticationnya saja:

```
import getpass, poplib

M = poplib.POP3('pop3.domain.id')
U = input("Masukkan username: ")
M.user(U)
try:
    M.pass_(getpass.getpass())
    print("Password benar")
except:
    print("Password salah")
```

Contoh IMAP:

```
import getpass, imaplib
```

```
M = imaplib.IMAP4('imap.domain.id')
M.login(getpass.getuser(), getpass.getpass())
M.select()
typ, data = M.search(None, 'ALL')
for num in data[0].split():
    typ, data = M.fetch(num, '(RFC822)')
    print ('Message %s\n%s\n' % (num, data[0][1]))
M.close()
M.logout()
```

IMAP untuk authentication saja:

```
import getpass, imaplib, sys

server = input("Server: ")
try:
  M = imaplib.IMAP4(server)
except Exception as e:
  print("Error because:", str(e))
  sys.exit()
user = input("Login: ")
try:
  M.login(user, getpass.getpass())
  print("Password Benar!")
except:
  print("Password Salah!")
```

SNMP

SNMP atau Simple Network Management Protocol digunakan untuk memonitor kondisi perangkat jaringan, pemakaian interface perangkat jaringan, dan informasi lainnya yang bisa ditambahkan dengan konfigurasi.

Instalasi library: pip3 install pysnmp

Contoh membaca informasi dari SNMP agent:

```
from pysnmp.hlapi import *

errorIndication, errorStatus, errorIndex, varBinds = next(
    getCmd(SnmpEngine(),
           CommunityData('public'),
           UdpTransportTarget(('10.2.2.3', 161)),
           ContextData(),
           ObjectType(ObjectIdentity('1.3.6.1.2.1.1.1.0')),
           ObjectType(ObjectIdentity('1.3.6.1.2.1.1.6.0')))
)

if errorIndication:
    print(errorIndication)
elif errorStatus:
    print('%s at %s' % (errorStatus.prettyPrint(),
                        errorIndex and varBinds[int(errorIndex) - 1][0] or '?'))
else:
    for varBind in varBinds:
        print(' = '.join([x.prettyPrint() for x in varBind]))
```

IP 10.2.2.3 adalah contoh sebuah target SNMP device yang akan dibaca informasinya, sedangkan 161 adalah port untuk protokol SNMP.

Angka 1.3.6.1.2.1.1.1.0 dan 1.3.6.1.2.1.1.6.0 adalah contoh dua nomor MIB atau Management Information Base yaitu susunan informasi secara hirarki dari sebuah SNMP agent yang biasanya terpasang pada router dan switch, namun bisa juga pada perangkat jaringan lainnya seperti server, firewall, dan lain-lain. Yang perlu diperhatikan adalah versi SNMP yang digunakan antara agent dengan client harus sama. Contoh diatas adalah SNMPv2. Untuk SNMPv3 bedanya ada di system keamanannya karena bisa dilengkapi dengan authentication dan encryption. Ganti angka-angka object identity tersebut untuk melihat informasi lainnya.

FTP Client

FTP atau File Transfer Protocol adalah salah satu cara untuk mengirim dan menerima file di jaringan TCP/IP. Umum digunakan untuk type file teks dan binary. Mode yang aman digunakan adalah binary karena meskipun file yang ditransfer adalah teks masih dapat diterima dengan baik. Contoh sebuah client FTP menggunakan Python:

```
import sys,getpass
host = sys.argv[1]      #nama host target diinputkan melalui argument
user = sys.argv[2]      #username diinputkan melalui argument kedua
filename = sys.argv[3]          #nama file melalui argument ketiga
print(host)
print(user)
print(filename)
from ftplib import FTP          #menggunakan library ftplib

ftp = FTP(host)
ftp.login(user,getpass.getpass())      #password diinputkan tanpa echo ke layar
ftp.storbinary("STOR "+filename, open(filename, "rb"))
```

Contoh diatas menunjukkan cara meng-upload sebuah file ke sebuah host, terlihat disitu mode file yang di-upload adalah binary "rb".

Akses ke database MySQL/MariaDB

Python dapat juga akses ke database yang banyak digunakan seperti MySQL atau MariaDB, salah satu library yang bisa digunakan adalah mysql.connector dengan menginstall: python3 apt install python3-mysql.connector. Contoh melakukan query select pada sebuah database:

```
import mysql.connector
db = mysql.connector.connect(user = "namauser", password = "12345", host = "localhost", database = "namadb")
cursor = db.cursor()
cursor.execute("select * from tableku")
results = cursor.fetchall()
for i in results:
  print(i[0],i[1],i[2],i[3])
db.close()
```

Method .fetchall() akan mengambil seluruh hasil query dengan field-field yang dimasukkan ke sebuah array atau list, sehingga untuk mengambil seluruh isinya dapat dilakukan dengan for loop seperti contoh diatas.

Yang perlu diperhatikan pada koneksi MySQL ini adalah untuk query yang sifatnya membuat perubahan pada database yaitu insert, delete dan update, harus dilakukan **.commit()** setelah eksekusi query .execute():

```
import mysql.connector
db = mysql.connector.connect(user = "namauser", password = "12345", host = "localhost", database = "namadb")
cursor = db.cursor()
nama = input("Masukkan nama: ")
age = input("Masukkan umur: ")
salary = input("Masukkan gaji: ")
sql = "insert into employee values (NULL, '"+nama+"', "+age+", "+salary+")"
cursor.execute(sql)
db.commit()
db.close()
```

Bab 9 - Otomasi Jaringan

Sebagai administrator jaringan tentunya akan kewalahan apabila harus berhadapan dengan perangkat jaringan komputer yang jumlahnya sangat banyak. Otomasi pada jaringan akan sangat membantu apabila dibutuhkan suatu konfigurasi yang memiliki pola yang dapat diatur. Merangkum dari sebuah blog di https://zakkymuhammad.com/blog/network-automation/, inilah beberapa cara untuk melakukan otomasi jaringan dengan Python:

Telnetlib

Salah satu cara untuk mengendalikan perangkat jaringan adalah melalui protokol telnet, meski protokol ini tidak terenkripsi namun masih digunakan karena pemakaiannya yang mudah dan tersedia di perangkat-perangkat jaringan yang lama. Dipastikan dahulu pada perangkat jaringan untuk service telnet yang aktif beserta dengan username dan passwordnya. Library telnetlib sudah tersedia dalam paket Python 3 sehingga tidak perlu download secara terpisah. Contoh penggunaannya:

```
import telnetlib

tn = telnetlib.Telnet("10.2.2.3")

tn.read_until(b"Username: ")
tn.write(b"admin\n")

tn.read_until(b"Password: ")
tn.write(b"secretPass81\n")

tn.write(b"enable\n")

tn.read_until(b"Password: ")
tn.write(b"cisco\n")

tn.write(b"conf term\n")
tn.write(b"hostname R1\n")
tn.write(b"exit\n")
tn.write(b"copy running-config startup-config\n")
```

```
print(tn.read_all().decode('ascii'))
```

Contoh diatas dilakukan untuk remote ke sebuah router Cisco untuk memberi konfigurasi hostname menjadi R1 lalu save ke memory. Method .read_until() mendeteksi teks yang diterima dari perangkat Cisco agar bisa kemudian melakukan entry teks yang sesuai. Apabila username dan password diambil dari variable maka perlu melakukan encoding ke ASCII:

```
import telnetlib

user = "admin"
passw = "secretPass81"

tn = telnetlib.Telnet("10.2.2.3")

tn.read_until(b"Username: ")
tn.write(user.encode('ascii') + b"\n")

tn.read_until(b"Password: ")
tn.write(passw.encode('ascii') + b"\n")

tn.write(b"enable\n")

tn.read_until(b"Password: ")
tn.write(b"cisco\n")

tn.write(b"conf term\n")
tn.write(b"hostname R1\n")
tn.write(b"exit\n")
tn.write(b"copy running-config startup-config\n")

print(tn.read_all().decode('ascii'))
```

Apabila dibutuhkan entry manual untuk passwordnya maka dapat menggunakan library getpass seperti pada contoh POP3 dan IMAP di bab sebelumnya, agar ketikan password tidak ditampilkan di layar.

Paramiko

Apabila dibutuhkan koneksi yang lebih aman selain telnet, dapat menggunakan SSH, pastikan perangkat jaringan sudah mengaktifkan service SSH dan menyediakan username dengan passwordnya. Salah satu cara untuk koneksi menggunakan SSH adalah dengan library Paramiko, instalasinya:

```
pip3 install paramiko
```

Lalu contoh cara penggunaannya:

```
import paramiko
import time

ssh_client = paramiko.SSHClient()
ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())
ssh_client.connect(hostname="10.2.2.3",username="admin",password="secret81")

conn = ssh_client.invoke_shell()

conn.send("enable\n")
conn.send("cisco123\n")
conn.send("conf term\n")
conn.send("hostname R1\n")
#time.sleep(1)          #kadang dibutuhkan delay agar outputnya terlihat

output = conn.recv(65535)
print(output.decode("ascii"))

ssh_client.close()
```

Netmiko

Alternatif lain adalah Netmiko dimana library ini mendukung koneksi untuk beberapa type perangkat jaringan komputer yang popular seperti Cisco, Juniper, Aruba, MikroTik dan lain sebagainya. Cara instalasi library Netmiko:

```
pip3 install netmiko
```

Contoh penggunaanya ke perangkat Cisco:

```
from netmiko import ConnectHandler

router = {
    'device_type':'cisco_ios',
    'ip':'10.2.2.3',
    'username':'admin',
    'password':'secretPass81',
    'secret':'cisco123'
}

conn = ConnectHandler(**router)

#contoh pengetikan perintah di prompt user exec mode
print(conn.send_command('sh ip int b'))

#masuk ke privilege exec mode
print(conn.enable())

#dalam global config mode
cmd = [
        'interface G0/0/1',
        'ip address 192.168.0.1 255.255.255.0',
        'no shutdown'
]
print(conn.send_config_set(cmd))
```

Napalm

Seperti Netmiko, Napalm (Network Automation and Programmability Abstraction Layer with Multivendor support) juga mendukung perangkat-perangkat jaringan yang popular, cara instalasi librarynya:

pip3 install napalm

Apabila ada dependencies yang belum terinstall, bisa dilakukan instalasinya:

apt install libssl-dev libffi-dev python-dev python-cffi

Contoh penggunaannya untuk membaca konfigurasi perangkat Cisco:

```
from napalm import get_network_driver

driver = get_network_driver('ios')
enpass = {'secret':'cisco123'}
device = driver(hostname='10.2.2.3', username='admin', password='secretPass81', optional_args=enpass)
device.open()

devshow = device.get_facts()
print(devshow)
#atau bisa juga output dalam format JSON:
print(json.dumps(devshow, indent=3))

device.close()
```

Dalam hal otomasi, melakukan akses ke banyak perangkat sekaligus merupakan tujuan utamanya agar menghemat waktu dan tenaga, serta menjaga akurasinya karena menggantikan pekerjaan manual yang harus dilakukan ke setiap perangkat satu per satu. Contoh dibawah ini adalah untuk mengambil backup konfigurasi di seluruh device dalam list:

```
from napalm import get_network_driver

hostlist = ['10.2.2.3', '10.2.2.4', '10.2.2.5', '10.2.2.6']

for host in hostlist:
    driver = get_network_driver('ios')
    other_args = {'secret':'cisco123','port':22}
    device = driver(hostname=host, username='admin', password='secretPass81', optional_args=other_args)
```

```
    device.open()

    # mendapatkan konfigurasi (all)
    device_config = device.get_config()

    # mendapatkan running config
    device_config_running = device_config['running']

    # mendapatkan startup config
    device_config_startup = device_config['startup']

    # mendapatkan candidate config
    device_config_candidate = device_config['candidate']

    # menulis running config ke file
    file = open('config_running_{}'.format(host), 'w')
    file.write(device_config_running)
    file.close()

    # menulis startup config ke file
    file = open('config_startup_{}'.format(host), 'w')
    file.write(device_config_startup)
    file.close()

device.close()
```

Mikrotik RouterOS

Kita lihat sebelumnya bahwa perangkat jaringan dengan service SSH dapat dikendalikan dengan library Paramiko, maka berlaku juga untuk perangkat MikroTik karena support SSH. Selain SSH MikroTik juga dapat diakses melalui web dan aplikasi Webfig dari MikroTik sendiri. Untuk MikroTik sendiri sebetulnya ada library khusus yaitu routeros_api yang dapat diinstall dengan cara:

```
pip3 install routeros_api
```

Contoh penggunaannya untuk melihat list IP address dari sebuah router MikroTik:

```
import routeros_api
import json

host = '10.2.2.10'

connection = routeros_api.RouterOsApiPool(host, username='root', password='123456',
plaintext_login=True)
api = connection.get_api()

list_ipadd = api.get_resource('ip/address/')
show_ipadd = list_ipadd.get()

print(json.dumps(show_ipadd, indent=3))

# bila ingin menampilkan dari key address saja:
for i in show_ipadd:
    print(i['address'])

connection.disconnect()
```

Dokumentasi API ini dapat dilihat di: https://github.com/socialwifi/RouterOS-api

Bab 10 - Keamanan Jaringan

Python juga populer di dunia cybersecurity karena banyak sekali tools yang tersedia dan library-library yang berhubungan dengan network scanning dan penetration test. Di luar Python apabila ingin menguji sebuah target umumnya menggunakan tool bernama nmap. Dengan nmap dapat melakukan scan ke sebuah target alamat IP untuk kemudian diketahui service mana saja yang aktif maupun terfilter oleh firewall. Dengan Python dapat dilakukan hal serupa dengan beberapa cara, salah satunya adalah dengan script sebagai berikut:

```
from socket import *
import time
startTime = time.time()

if __name__ == '__main__':
    target = input('Masukkan host untuk discan: ')
    t_IP = gethostbyname(target)
    print ('Starting scan on host: ', t_IP)

    for i in range(50, 500):
        s = socket(AF_INET, SOCK_STREAM)

        conn = s.connect_ex((t_IP, i))
        if(conn == 0) :
            print ('Port %d: OPEN' % (i,))
        s.close()
print('Time taken:', time.time() - startTime)
```

Cara penggunaannya: python3 scanner.py 202.43.124.9

Contoh adalah IP target, outputnya akan memunculkan laporan seperti nmap:

Port 53: OPEN

Port 80: OPEN

Port 106: OPEN

Port 110: OPEN

Port 111: OPEN

Port 143: OPEN

Port 389: OPEN

Port 443: OPEN

Time taken: 4.661398410797119

Seluruh port yang terbuka akan ditampilkan, dan juga waktu yang dibutuhkan untuk melakukan scanning.

Sumber source:

https://www.tutorialspoint.com/python_penetration_testing/python_penetration_testing_network_scanner.htm

Hal lain yang dilakukan oleh potensi penyerang adalah melakukan ping sweep, yaitu cara untuk mengetahui host-host yang aktif dalam sebuah range network:

```
import os
import platform

from datetime import datetime
net = input("Masukkan Network Address: ")
net1= net.split('.')
a = '.'

net2 = net1[0] + a + net1[1] + a + net1[2] + a
st1 = int(input("Masukkan Angka Awal (mis: 1): "))
en1 = int(input("Masukkan Angka Akhir (mis: 100): "))
en1 = en1 + 1
oper = platform.system()

if (oper == "Windows"):
   ping1 = "ping -n 1 "
elif (oper == "Linux"):
   ping1 = "ping -c 1 "
else :
   ping1 = "ping -c 1 "
t1 = datetime.now()
print ("Scanning in Progress:")
```

```
for ip in range(st1,en1):
   addr = net2 + str(ip)
   comm = ping1 + addr
   response = os.popen(comm)

   for line in response.readlines():
      if(line.count("TTL")):
         break
      if (line.count("TTL")):
         print (addr, "--> Live")

t2 = datetime.now()
total = t2 - t1
print ("Scanning completed in: ",total)
```

Kadang-kadang dengan cara diatas tidak ditemukan satu host pun karena beberapa alasan, salah satunya adalah protokol ICMP yang difilter oleh firewall, sehingga tidak dapat dilakukan ping pada target. Maka dari itu dapat digunakan cara lain yaitu dengan cara TCP scan, karena apabila terjadi 3-way handshake pada sebuah target, maka bisa diartikan target IP tersebut live:

```
import socket
from datetime import datetime
net = input("Enter the IP address: ")
net1 = net.split('.')
a = '.'

net2 = net1[0] + a + net1[1] + a + net1[2] + a
st1 = int(input("Masukkan Angka Awal (mis: 1): "))
en1 = int(input("Masukkan Angka Akhir (mis: 100): "))
en1 = en1 + 1
t1 = datetime.now()

def scan(addr):
   s = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
```

```
    socket.setdefaulttimeout(1)
    result = s.connect_ex((addr,135))
    if result == 0:
       return 1
    else :
       return 0


def run1():
    for ip in range(st1,en1):
       addr = net2 + str(ip)
       if (scan(addr)):
          print (addr , "is live")


run1()
t2 = datetime.now()
total = t2 - t1
print ("Scanning completed in: " , total)
```

Dengan TCP scan tingkat keberhasilan sweepingnya lebih tinggi karena mayoritas server-server dengan IP publik menjalankan beberapa service, meskipun tidak seluruhnya bisa diakses dari luar karena melewati firewall.

Daftar Pustaka

1. Network Automation. https://zakkymuhammad.com/blog/network-automation/ (diakses 1 Oktober 2023)
2. Python 3 documentation. https://docs.python.org/3/ (diakses 1 Oktober 2023)
3. Python Penetration Testing. https://www.tutorialspoint.com/python_penetration_testing/index.htm (diakses 1 Oktober 2023)

PEMROGRAMAN JARINGAN KOMPUTER DENGAN PYTHON

Buku ini berisikan bahan ajar mata pelajaran Pemrograman Jaringan Komputer dengan pengantar bahasa pemrograman Python versi 3. Dalam dunia jaringan komputer lebih mengenal konfigurasi perangkat jaringan seperti router dan switch tanpa adanya pemrograman. Dengan berkembangnya waktu dan kebutuhan, pengelola jaringan komputer mengalami kesulitan dalam mengatur jumlah perangkat yang semakin banyak, sehingga membutuhkan banyak waktu dan tenaga untuk mengatasinya. Dengan otomasi jaringan, script Python dapat membantu menyelesaikan tugas seorang administrator jaringan untuk melakukan monitoring hingga konfigurasi beberapa peralatan sekaligus, selain membantu dalam hal kalkulasi IP subnetting.

Lembaga Penelitian dan Pengabdian kepada Masyarakat (LPPM)
Universitas Kristen Petra
Jl. Siwalankerto 121-131, Surabaya 60236, Indonesia
Telp. 031-2983111
https://lppm.petra.ac.id

ISBN 978-623-5457-09-3 (PDF)