Modul Praktikum
Algoritma

Rolly Maulana Awangga
0410118609

Applied Bachelor of Informatics Engineering
Program Studi D4 Teknik Informatika

Applied Bachelor Program of Informatics Engineering
Politeknik Pos Indonesia
Bandung 2019

‘Jika Kamu tidak dapat menahan lelahnya belajar,
Maka kamu harus sanggup menahan perihnya Kebodohan.’
Imam Syafi’i

Acknowledgements

Pertama-tama kami panjatkan puji dan syukur kepada Allah SWT yang telah memberikan rahmat dan hidayah-Nya sehingga Buku Pedoman Tingkat Akhir ini dapat diselesaikan.

Abstract

Buku Pedoman ini dibuat dengan tujuan memberikan acuan, bagi mahasiswa Tingkat Akhir dan dosen Pembimbing. Pada intinya buku ini menjelaskan secara lengkap tentang Standar pengerjaan Intership dan Tugas Akhir di Program Studi D4 Teknik Informatika, dan juga mengatur mekanisme, teknik penulisan, serta penilaiannya.Dengan demikian diharapkan semua pihak yang terlibat dalam aktivitas Bimbingan Mahasiswa Tingkat Akhir berjalan lancar dan sesuai dengan standar.

Contents

List of Figures

Chapter 1

Python

Python merupakan bahasa pemrograman tingkat tinggi yang diracik oleh Guido van Rossum.

Python banyak digunakan untuk membuat berbagai macam program, seperti: program CLI, Program GUI (desktop), Aplikasi Mobile, Web, IoT, Game, Program untuk Hacking, dsb.

Python juga dikenal dengan bahasa pemrograman yang mudah dipelajari, karena struktur sintaknya rapi dan mudah dipahami.

1.1 Hello World Python

1.1.1 Teori

Python memang sangat sederhana dibandingkan bahasa yang lainnya. Tidak perlu ini dan itu untuk membuat program Hello World!. Bahkan tagline di websitenya menjelaskan, kalau python akan membuatmu bekerja lebih cepat dan efektif. Berikut salah satu contoh dasar perbedaan python dengan bahasa pemograman lain:

C++ "Hello World"

```
#include <iostream.h>
main()
{
count << "hello world!";
}
retun 0
```

Listing 1.1: koding

Java "Hello World"

```
class HelloWorldApp
{
public static void main(String[] args)
    {
```

```
5       System.out.println("Hello World!");
6       }
7   }
```

Listing 1.2: koding

Python

```
1 print "Hello World"
```

Listing 1.3: koding

Chapter 2

Fungsi Pada Python

Pada pembuatan program yang kompleks dan memiliki banyak fitur, kita diharuskan menggunakan fungsi. hal tersebut karena bisa jadi kita akan kerepotan menulis kode programnya, karena banyak yang harus ditulis dan kode akan menjadi sulit dibaca dan dirawat (maintenance).

Dengan fungsi, kita dapat memecah program besar menjadi sub program yang lebih sederhana. Masing-masing fitur pada program dapat kita buat dalam satu fungsi. Pada saat kita membutuhkan fitur tersebut, kita tinggal panggil fungsinya saja.Hal ini akan kita coba pada contoh program yang sudah saya sediakan di bawah.Namun, terlebih dahulu. Kita harus memahami teori dasar dan hal apa saja yang harus kita ketahui tentang fungsi di Python.

2.1 Cara Membuat Fungsi Python

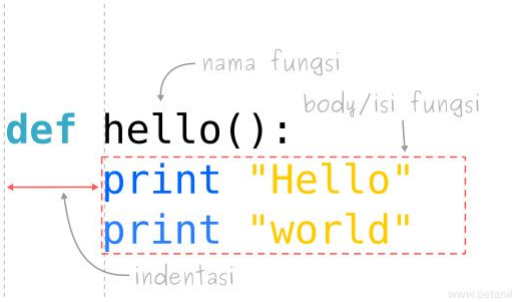
Fungsi pada Python, dibuat dengan kata kunci def kemudian diikuti dengan nama fungsinya.

Contoh:

```
1 def nama_fungsi():
2     print "Hello ini Fungsi"
```

Listing 2.1: koding

Sama seperti blok kode yang lain, kita juga harus memberikan identasi (tab atau spasi 2x) untuk menuliskan isi fungsi.



Setelah kita buat, kita bisa memanggilnya seperti ini:

```
nama_fungsi()
```

Listing 2.2: koding

Sebagai contoh, coba tulis kode program berikut:

```
# Membuat Fungsi
def salam():
    print "Hello World"

## Pemanggilan Fungsi
salam()
salam()
salam()
```

Listing 2.3: koding

Hasilnya:

```
Hello World
Hello World
Hello World
```

Listing 2.4: koding

Intinya adalah apapun yang ada di dalam fungsi, ketika dipanggil itulah yang akan dilakukan.

FYI: fungsi juga dapat dipanggil pada fungsi lain, bahkan bisa memanggil dirinya sendiri. Fungsi yang memanggil dirinya sendiri, disebut fungsi rekursif.

2.2 Fungsi Dengan Parameter

Fungsi parameter merupakan variabel yang menampung nilai untuk diproses di dalam fungsi.

Contoh:

```
def salam(ucapan):
    print(ucapan)
```

Listing 2.5: koding

Pada contoh diatas, kita membuat fungsi dengan parameter ucapan.Kemudian cara untuk pemanggilan fungsi yang memiliki parameter tersebut adalah seperti berikut:

```
salam("Selamat siang")
```

Listing 2.6: koding

"Selamat siang" adalah nilai parameter yang kita berikan.

Untuk parameter yang lebih dari satu kita bisa menggunakan tanda koma (,) untuk memisahnya.

Contoh:

```
# Membuat fungsi dengan parameter
def luas_segitiga(alas, tinggi):
    luas = (alas * tinggi) / 2
    print "Luas segitiga: %f" % luas;

# Pemanggilan fungsi
luas_segitiga(4, 6)
```

Listing 2.7: koding

Hasilnya:

```
Luas segitiga: 12.000000
```

Listing 2.8: koding

2.3 Fungsi Yang Mengembalikan Nilai

Fungsi yang tidak mengembalikan nilai biasanya disebut dengan prosedur. Tetapi, suatu waktu kita butuh hasil proses dari fungsi untuk digunakan pada proses berikutnya.Maka fungsi harus mengembalikan nilai dari hasil pemrosesannya.Cara mengembalikan nilai adalah menggunakan kata kunci *return* lalu diikuti dengan nilai atau *variabel* yang akan dikembalikan.

Contoh:

```
def luas_persegi(sisi):
    luas = sisi * sisi
    return luas

# pemanggilan fungsi
print "Luas persegi: %d" % luas_persegi(6)
```

Listing 2.9: koding

Hasilnya:

```
Luas persegi: 36
```

Listing 2.10: koding

Perbedaan dengan fungsi luas segitiga() yang sebelumnya adalah pada fungsi luas segitiga() kita melakukan print dari hasil pemrosesan secara langsung di dalam fungsinya. Sedangkan fungsi luas persegi(), kita melakukan print pada saat pemanggilannya.Jadi, fungsi luas persegi() akan bernilai sesuai dengan hasil yang dikembalikan. Sehingga kita dapat memanfaatkannya untuk pemerosesan berikutnya.

Seperti pada contoh berikut:

```
# rumus: sisi x sisi
def luas_persegi(sisi):
    luas = sisi * sisi
    return luas


# rumus: sisi x sisi x sisi
def volume_persegi(sisi):
    volume = luas_persegi(sisi) * sisi
```

Listing 2.11: koding

Pada contoh di atas, kita melakukan pemanggilan fungsi luas persegi() untuk menghitung volume persegi.

2.4 Variabel Global dan Lokal pada Python

Variabel Global(berbeda file) adalah variabel yang bisa diakses dari semua fungsi, sedangkan variabel lokal(satu file) hanya bisa diakses di dalam fungsi tempat ia berada saja.Pada Python, urutan pengaksesan variabel (scope) dikenal dengan sebutan LGB (Local, Global, dan Build-in).Jadi program python mulai mencari vairabel lokal terlebih dahulu, kalau ada maka itu yang digunakan.Tetapi kalau tidak ada, pencarian terus ke Global, dan Build-in. Variabel Build-in adalah variabel yang sudah ada di dalam Python.

Contoh program:

```
# membuat variabel global
nama = "Petanikode"
versi = "1.0.0"

def help():
    # ini variabel lokal
    nama = "Programku"
    versi = "1.0.2"
    # mengakses variabel lokal
    print "Nama: %s" % nama
    print "Versi: %s" % versi


# mengakses variabel global
```

```
15 print "Nama: %s" % nama
16 print "Versi: %s" % versi
17
18 # memanggil fungsi help()
19 help()
```

Listing 2.12: koding

Hasilnya:

```
1 Nama: Petanikode
2 Versi: 1.0.0
3 Nama: Programku
4 Versi: 1.0.2
```

Listing 2.13: koding

Perhatikanlah variabel nama yang berada di dalam fungsi help() dan diluar fungsi ‘help(). Variabel nama yang berada di dalam fungsi help() adalah variabel lokal. Jadi, saat kita memanggil fungsi help() maka nilai yang akan tampil adalah nilai yang ada di dalam fungsi help(). Mengapa yang tampil tidak global di karenakan Python mulai mencari dari lokal, ke global, dan build-in. Jika di tiga tempat itu tidak ditemukan, maka biasanya akan terjadi NameError atau variabel tidak ditemukan.

2.5 Contoh Program Dengan fungsi

Berikut adalah langkah membuat program nya. Silahkan buat file baru bernama programfungsi.py. Lalu kita mulai tulis kodenya. Pertama kita buat sebuah variabel global berupa list untuk menampung judul-judul buku.

```
# Variabel global untuk menyimpan data Buku
buku = []
```

Listing 2.14: koding

Kemudian program ini akan mampu melakukan operasi CRUD (Create, Read, Update, dan Delete). Maka kita membutuhkan fungsi-fungsi berikut:

```
    \item show\_data() untuk menampilkan data dari list buku;
    \item insert\_data() untuk menambahkan data ke list buku;
    \item edit\_data() untuk mengedit data di list buku;
    \item delete\_data() untuk untuk menghapus data dari list buku.
```

Listing 2.15: koding

Langkah Berikutnya adalah dimulai dari fungsi show_data():

```
# fungsi untuk menampilkan semua data
def show_data():
    if len(buku) <= 0:
        print "BELUM ADA DATA"
    else:
        for indeks in range(len(buku)):
            print "[%d] %s" % (indeks, buku[indeks])
```

Listing 2.16: koding

Fungsi di atas akan mengecek isi dari list buku. Jika isinya kosong (len(buku) ¡= 0) maka tampilkan "BELUM ADA DATA". Namun, apabila ada isinya, maka tampilkan semua isinya dengan perulangan.

Selanjutnya membuat fungsi insert_data():

```
# fungsi untuk menambah data
def insert_data():
    buku_baru = raw_input("Judul Buku: ")
    buku.append(buku_baru)
```

Listing 2.17: koding

Selanjutnya membuat fungsi edit_data():

```
# fungsi untuk edit data
def edit_data():
    show_data()
    indeks = input("Inputkan ID buku: ")
    if(indeks > len(buku)):
        print "ID salah"
    else:
        judul_baru = raw_input("Judul baru: ")
        buku[indeks] = judul_baru
```

Listing 2.18: koding

Fungsi di atas akan menampilkan isi dari list buku dengan memanggil fungsi show_data() di dalamnya. Setelah itu, kita meminta user untuk menginputkan ID atau nomer indeks buku yang akan diedit. Lalu kita lakukan pengecekan, jika ID yang diinputkan melebihi dari isi list buku (indeks ¿ len(buku)), maka tampilkan pesan "ID salah". Namun, apabila tidak melebihi dari isi buku, maka ambil input untuk judul baru dan simpan sesuai ID-nya.

Selanjutnya membuat fungsi delete_data():

```
# fungsi untuk menhapus data
def delete_data():
    show_data()
    indeks = input("Inputkan ID buku: ")
    if(indeks > len(buku)):
        print "ID salah"
    else:
        buku.remove(buku[indeks])
```

Listing 2.19: koding

Hampir sama dengan fungsi edit_data(). Fungsi delete_data() juga harus menampilkan isi list buku dan mengambil ID yang akan dihapus. Kita dapat menghapus item pada list dengan fungsi remove().

Setelah langkah di atas, ada 2 fungsi lagi yang di butuhkan:

1. Fungsi untuk menampilkan menu
2. Fungsi untuk keluar (sudah ada di python: exit())

```
# fungsi untuk menampilkan menu
def show_menu():
    print "\n"
    print "---------- MENU ----------"
    print "[1] Show Data"
    print "[2] Insert Data"
    print "[3] Edit Data"
    print "[4] Delete Data"
```

```
    print "[5] Exit"

    menu = input("PILIH MENU> ")
    print "\n"

    if menu == 1:
        show_data()
    elif menu == 2:
        insert_data()
    elif menu == 3:
        edit_data()
    elif menu == 4:
        delete_data()
    elif menu == 5:
        exit()
    else:
        print "Salah pilih!"
```

Listing 2.20: koding

Fungsi di atas akan menampilkan menu dari 15, lalu memanggil fungsi-fungsi yang sudah dibuat berdasarkan menu yang dipilih.

Terakhir, kita harus membuat main loop programnya.

```

if __name__ == "__main__":

    while(True):
        show_menu()
```

Listing 2.21: koding

Program akan mengulang terus-menerus sampai fungsi exit() dieksekusi.

if __name__ == "__main__": adalah blok main di Python. Sebenarnya tanpa ini, programnya sudah bisa dijalankan namun agar lebih bagus kita baiknya menambahkannya.

Sehingga kode lengkapnya akan seperti ini:

```
# Variabel global untuk menyimpan data Buku
buku = []


# fungsi untuk menampilkan semua data
def show_data():
    if len(buku) <= 0:
        print "BELUM ADA DATA"
    else:
        for indeks in range(len(buku)):
            print "[%d] %s" % (indeks, buku[indeks])


# fungsi untuk menambah data
def insert_data():
```

```
16      buku_baru = raw_input("Judul Buku: ")
17      buku.append(buku_baru)
18
19  # fungsi untuk edit data
20  def edit_data():
21      show_data()
22      indeks = input("Inputkan ID buku: ")
23      if(indeks > len(buku)):
24          print "ID salah"
25      else:
26          judul_baru = raw_input("Judul baru: ")
27          buku[indeks] = judul_baru
28
29  # fungsi untuk menhapus data
30  def delete_data():
31      show_data()
32      indeks = input("Inputkan ID buku: ")
33      if(indeks > len(buku)):
34          print "ID salah"
35      else:
36          buku.remove(buku[indeks])
37
38  # fungsi untuk menampilkan menu
39  def show_menu():
40      print "\n"
41      print "---------- MENU ----------"
42      print "[1] Show Data"
43      print "[2] Insert Data"
44      print "[3] Edit Data"
45      print "[4] Delete Data"
46      print "[5] Exit"
47
48      menu = input("PILIH MENU> ")
49      print "\n"
50
51      if menu == 1:
52          show_data()
53      elif menu == 2:
54          insert_data()
55      elif menu == 3:
56          edit_data()
57      elif menu == 4:
58          delete_data()
59      elif menu == 5:
60          exit()
61      else:
62          print "Salah pilih!"
63
64
65  if __name__ == "__main__":
66
67      while(True):
```

```
68            show_menu()
```

Listing 2.22: koding

Chapter 3

Class Pada Phyton

Class merupakan sebuah objek yang di dalam nya biasanya terdapat beberapa metode yang memang merupakan isi dari sebuah class ini. Class dan metode ini biasa di sembut sebagai OOP atau object oriented programing. Dan OOP ini memang fungsinya untuk memudahkan proses atau kegiatan programing kita. Class ini merupakan sebuah objek yang lebih complex dengan di dalamnya berisi beberapa metode.

3.1 Cara Membuat Sebuah Class Pada Python

Untuk membuat sebuah class ini, harus kita awali dengan sebuah kata kunci. Yaitu class yang kemudian di ikuti dengan nama class nya. Dan yang terakhir adalah tanda kurung buka dan tutup serta tanda titik dua () dan :. untuk lebih mudahnya kita bisa lihat atau simak contohnya di bawah ini:

```
1 class namaClass () :
2     def metode 1 (self) :
3         Isi metode
4     def metode 2 (self) :
5         Isi metode
```

Listing 3.1: koding

3.2 Cara memanggil sebuah class dan metode didalamnya.

Untuk memanggil sebuah class, sama saja seperti layak nya memanggil metode.. Kita cukup menyebutkan nama classnya dengan di akhiri dengan tanda kurung buka dan tutup seperti di bawah ini.

```
1 namaClass()
```

Listing 3.2: koding

Untuk memanggil metodenya, kita cukup menggunakan memanggil class yang kemudian di ikuti dengan pemanggilan nama metode yang tersedia di dalam class tersebut dengan di pisahkan oleh tanda titik. Seperti di bawah ini.

```
1 namaClass().namaMetode()
```

Listing 3.3: koding

Untuk memudahkan pemanggilan metode ini, kita bisa menampung class nya ke dalam sebuah variabel terlebih dahulu. Yang kemudian kita panggil metodenya seperti di bawah ini.

```
1 penampung = namaClass()
2 penampung.namaMetode()
```

Listing 3.4: koding

Di dalam sebuah class,biasanya terdapat sebuah metode yang namanya sudah di sediakan oleh python. Namanya adalah __init__. Dan jika contoh di atas kita tambahkan __init__ maka kurang lebih akan seperti berikut ini.

```
1 class namaClass () :
2     def __init__() :
3         Isi yang ingin kalian masukkkan
4     def metode 1 (self) :
5         Isi metode
6     def metode 2 (self) :
7         Isi metode
```

Listing 3.5: koding

Dan sama seperti metode, kita bisa menggunakan atau mengirim sebuah nilai di dalamnya atau tidak. Untuk mengirimnya sama saja. Kita cukup memasukkan sebuah variabel di dalam tanda Kurung pada metode __init__. Dan ingat, bukan pada tanda kurung milik clannya ya. Seperti dibawah ini:

```
1 class namaClass () :
2     def __init__(self, parameter) :
3         Code program yang akan kalian eksekusi pertama kali.
4     def metode 1 (self, parameter) :
```

```
        Isi metode
    def metode 2 (self) :
        Isi metode
```

Listing 3.6: koding

Untuk memanggil sebuah class yang memiliki parameter, tentu kita harus memasukkan sebuah nilai saat pemanggilannya. Seperti yang ada di bawah ini.

```
namaClass(isiNilai)
```

Listing 3.7: koding

Penjelasan mengenai __init__, metode ini merupakan metode yang akan langsung dijalankan ketika class kita di panggil nantinya. Jadi kita tidak perlu memanggil metodenya secara manual seperti metode - metode yang lain seperti yang sudah saya jelaskan di atas.

3.3 Contoh dan pemanfaatan sebuah class pada python.

Setelah kita mengetahui cara membuat dan cara memanggilnya, maka sekarang saya akan mencoba untuk melihat contoh dan pemanfaatan dari sebuah class ini. Hal ini tentu agar membuat lebih paham mengenai apa yang dimaksud dengan class pada python ini. Contoh programnya di seperti bawah ini:

```
#merupakan sebuah class dengan metode untuk mencari jumlah huruf vokal
class pencariHuruf() :
    def __init__(self, teks):
        self.kata = teks
        self.a = 0
        self.i = 0
        self.u = 0
        self.e = 0
        self.o = 0
    def A (self) :
        for i in range(len(self.kata)):
            if self.kata[i] == 'a' or self.kata[i] == 'A' :
                self.a = self.a + 1
        return self.a
    def I (self) :
        for i in range(len(self.kata)):
            if self.kata[i] == 'i' or self.kata[i] == 'i' :
                self.i = self.i + 1
        return self.i
    def U (self) :
        for i in range(len(self.kata)):
            if self.kata[i] == 'u' or self.kata[i] == 'u' :
                self.u = self.u + 1
```

```
        return self.u
    def E (self) :
        for i in range(len(self.kata)):
            if self.kata[i] == 'e' or self.kata[i] == 'e' :
                self.e = self.e + 1
        return self.e
    def O (self) :
        for i in range(len(self.kata)):
            if self.kata[i] == 'o' or self.kata[i] == 'o' :
                self.o = self.o + 1
        return self.o

#proses pemanggilan dan penampungan class "pencariHuruf"
teks = 'ma mi mu me mo'
penampung = pencariHuruf(teks)

#pemanggilan metode yang ada di class "pencariHuruf"
jumlahA = penampung.A()
jumlahI = penampung.I()
jumlahU = penampung.U()
jumlahE = penampung.E()
jumlahO = penampung.O()

#mencetak hasil proses yang di tampung di dalam variabel di atas.
print(jumlahA)
print(jumlahI)
print(jumlahU)
print(jumlahE)
print(jumlahO)
```

Listing 3.8: koding

Chapter 4

Import

4.1 Import File dalam folder

Bagi yang sudah berpengalaman terjun di dunia computer programming, mereka sudah biasa mengelola banyak file kode. Tentunya banyak file program yang kita temui tidak hanya memiliki satu file script saja. Mereka memisahkan script fungsi ataupun class untuk mempermudah debug, refactor code dan memperjelas struktur kode mereka. Cara ini disebut dengan Modularisasi. Hal yang sangat saya rasakan dari keuntungan modularisasi ini adalah kita menjadi tidak perlu mengubah banyak kode untuk hal pengembangan atau perubahan sistem program kita. Di python, cara meng-import file nya terdapat banyak cara. Namun kita akan menggunakan cara yang lebih sederhana dan mudah.

4.1.1 Cara import dalam direktori yang sama.

Misalkan kita sedang berada di direktori kerja bernama project. Kita memiliki suatu program utama bernama Main.py dan file lain bernama KumpulanFungsi.py. Susunannya seperti ini:

```
1 - project
2   |---- Main.py
3   |---- KumpulanFungsi.py
```

Listing 4.1: koding

Di dalam script KumpulanFungsi.py berisi:

```
1
2 def perkalian(x,y):
3     return x * y
4
5 def pembagian(x,y):
6     return x / y
7
```

```
8  def pertambahan(x,y):
9      return x + y
10
11 def pengurangan(x,y):
12     return x - y
```

Listing 4.2: koding

Untuk menggunakan fungsi-fungsi di Main.py, maka cara importnya adalah seperti ini:

```
1 #Cara import seluruh isi file
2 from KumpulanFungsi import *
3
4 print pertambahan(10,12)
5 print pengurangan(20,9)
6 print pembagian(21,3)
7 print perkalian(6,6)
```

Listing 4.3: koding

Atau mengimport function/class yang dibutuhkan saja, untuk tujuan menghemat memori atau membatasi proses yang tidak dibutuhkan. Cara nya seperti ini:

```
1 #Cara import satuan function/class
2 from KumpulanFungsi import pertambahan
3
4 print pertambahan(21,24)
5
6 #Cara import beberapa function/class
7 from KumpulanFungsi import perkalian, pembagian
8
9 print pertambahan(perkalian(2,2),pembagian(12,3))
```

Listing 4.4: koding

4.1.2 Cara import di direktori yang berbeda

Untuk import dari direktori yang berbeda, python membutuhkan suatu file kosong bernama __init__.py di dalam direktori yang berisi file yang akan digunakan. Masih di kasus sebelumnya, kita menambahkan suatu folder baru bernama KumpulanKelas, yang berisi file Kucing.py dan Ayam.py. Maka susunannya akan seperti ini:

```
1 - project
2   |---- KumpulanKelas
3         |---- Kucing.py
4         |---- Ayam.py
5         |---- __init__.py
6   |---- Main.py
7   |---- KumpulanFungsi.py
```

Listing 4.5: koding

File Kucing.py berisi:

```
class Kucing(object):
    def suara(self):
        print "Meow!"
    def jenis(self):
        print "Mamalia"
```

Listing 4.6: koding

File Ayam.py berisi :

```
class Ayam(object):
    def suara(self):
        print "Kukuruyuk!"
    def jenis(self):
        print "Unggas"
```

Listing 4.7: koding

Maka cara importnya seperti berikut:

```
#Cara import semua function/kelas dari file di direktori lain
from KumpulanKelas.Kucing import *

Morganisa = Kucing()
Morganisa.suara()
Morganisa.jenis()

#Cara import satu function/kelas dari file di direktori lain
from KumpulanKelas.Ayam import Ayam

Pelung = Ayam()
Pelung.suara()
Pelung.jenis()
```

Listing 4.8: koding

Dari kode di atas, from KumpulanKelas.Kucing dimaksud dengan susunan direktori dari terluar sampai nama file yang ingin digunakan.

4.2 Import Library

Berikut 5 library data scientist:

1. **Scipy**

   Kegunaanya adalah untuk menangani operasi aljabar dan matriks serta operasi matematika lainya. Disini kamu dapat menangani sejumlah operasi matematika yang lebih kompleks daripada menggunakan library math bawaan Python. Ada juga beberapa fungsi statistika dasar yang dimiliki oleh Scipy.

```
>>> import numpy as np
>>> from scipy import linalg
>>> A = np.array([[1,2],[3,4]])
>>> A
array([[1, 2],
       [3, 4]])
>>> linalg.inv(A)
array([[-2. ,  1. ],
       [ 1.5, -0.5]])
>>> b = np.array([[5,6]]) #2D array
>>> b
array([[5, 6]])
>>> b.T
array([[5],
       [6]])
>>> A*b #not matrix multiplication!
array([[ 5, 12],
       [15, 24]])
>>> A.dot(b.T) #matrix multiplication
array([[17],
       [39]])
>>> b = np.array([5,6]) #1D array
>>> b
array([5, 6])
>>> b.T  #not matrix transpose!
array([5, 6])
>>> A.dot(b)  #does not matter for multiplication
array([17, 39])
2. Numpy
```

Listing 4.9: koding

2. **Numpy**

   Numpy memiliki kegunaan untuk operasi vektor dan matriks. Fiturnya hampir sama dengan MATLAB dalam mengelola array dan array multidimensi. Numpy merupakan salah satu library yang digunakan oleh library lain seperti Scikit-Learn untuk keperluan analisis data.

```
>>> x = np.array([[1, 2, 3], [4, 5, 6]], np.int32)
>>> type(x)
<type 'numpy.ndarray'>
>>> x.shape
(2, 3)
>>> x.dtype
dtype('int32')
```

Listing 4.10: koding

3. **Pandas**

   Dengan menggunakan sistem dataframe, kamu dapat memuat sebuah file ke dalam tabel virtual ala spreadsheet dengan menggunakan Pandas. Dengan menggunakan Pandas, kamu dapat mengolah suatu data dan mengolahnya seperti join, distinct, group by, agregasi, dan teknik seperti pada SQL. Hanya saja dilakukan pada tabel yang dimuat dari file ke RAM.

   Pandas juga dapat membaca file dari berbagai format seperti .txt, .csv, .tsv, dan lainnya. Anggap saja Pandas adalah spreadsheet namun tidak memiliki GUI dan punya fitur seperti SQL.

```
import pandas as pd
pd.set_option('display.mpl_style', 'default') # Make the graphs a
    bit prettier
figsize(15, 5)
broken_df = pd.read_csv('../data/bikes.csv')
# Look at the first 3 rows
broken_df[:3]
```

Listing 4.11: koding

4. **Matplotlib**

   Data yang kita olah tentu tidak elok apabila ditampilkan begitu saja dengan tabel hitam saja kepada investor atau manajemen. Bila ditampilkan dengan sejumlah grafik berwarna pasti mereka akan lebih tertarik melihatnya. Matplotlib membantu kamu untuk memvisualisasikan data dengan lebih indah dan rapi.

   Ada plot untuk menampilkan data secara 2D atau 3D. Sehingga kamu dapat menampilkan data yang telah kamu olah sesuai kebutuhan. Matplotlib pun terintegrasi dengan iPython Notebook atau Jupyter dimana kamu dapat membuat sebuah buku interaktif yang dapat diberi penjelasan dan kode yang disisipkan begitupun hasil plottingnya.

Matplotlib adalah library paling banyak digunakan oleh data science untuk menyajikan datanya ke dalam visual yang lebih baik.

```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4])
plt.ylabel('some numbers')
plt.show()
```

Listing 4.12: koding

5. **Scikit-Learn**

Machine learning ada yang berbasis statistika ada juga yang tidak. Salah satunya adalah support vector machine dan regresi linier. Mungkin bagi sebagian orang sudah biasa menulis sendiri library untuk implementasi kedua algoritma tadi. Tapi untuk membuatnya dalam waktu singkat tentu butuh waktu yang tidak sedikit pula.

Scikit-Learn memberikan sejumlah fitur untuk keperluan data science seperti:

Algoritma Regresi Algoritma Naive Bayes Algoritma Clustering Algoritma Decision Tree Parameter Tuning Data Preprocessing Tool Export / Import Model Machine learning pipeline dan lainnya Scikit-Learn sudah teruji dan memiliki dokumentasi yang super lengkap. Bahkan kontributornya pun banyak. Scikit-Learn pun menyediakan ekstensi untuk fuzzy logic dan computer vision.

```
$ python
>>> from sklearn import datasets
>>> iris = datasets.load_iris()
>>> digits = datasets.load_digits()
>>> print(digits.data)
[[  0.   0.   5. ...,   0.   0.   0.]
 [  0.   0.   0. ...,  10.   0.   0.]
 [  0.   0.   0. ...,  16.   9.   0.]
 ...,
 [  0.   0.   1. ...,   6.   0.   0.]
 [  0.   0.   2. ...,  12.   0.   0.]
 [  0.   0.  10. ...,  12.   1.   0.]]
>>> digits.target
array([0, 1, 2, ..., 8, 9, 8])
>>> digits.images[0]
array([[  0.,   0.,   5.,  13.,   9.,   1.,   0.,   0.],
       [  0.,   0.,  13.,  15.,  10.,  15.,   5.,   0.],
       [  0.,   3.,  15.,   2.,   0.,  11.,   8.,   0.],
       [  0.,   4.,  12.,   0.,   0.,   8.,   8.,   0.],
       [  0.,   5.,   8.,   0.,   0.,   9.,   8.,   0.],
       [  0.,   4.,  11.,   0.,   1.,  12.,   7.,   0.],
       [  0.,   2.,  14.,   5.,  10.,  12.,   0.,   0.],
       [  0.,   0.,   6.,  13.,  10.,   0.,   0.,   0.]])
>>> from sklearn import svm
```

```
>>> clf = svm.SVC(gamma=0.001, C=100.)
>>> clf.fit(digits.data[:-1], digits.target[:-1])
SVC(C=100.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma=0.001, kernel='rbf
    ',
  max_iter=-1, probability=False, random_state=None, shrinking=True
    ,
  tol=0.001, verbose=False)
>>> clf.predict(digits.data[-1:])
array([8])
```

Listing 4.13: koding

Appendix A

Form Penilaian Jurnal

gambar A.1 dan A.2 merupakan contoh bagaimana reviewer menilai jurnal kita.

NO	UNSUR	KETERANGAN	MAKS	KETERANGAN
1	Keefektifan Judul Artikel	Maksimal 12 (dua belas) kata dalam Bahasa Indonesia atau 10 (sepuluh) kata dalam Bahasa Inggris	2	a. Tidak lugas dan tidak ringkas (0) b. Kurang lugas dan kurang ringkas (1) c. Ringkas dan lugas (2)
2	Pencantuman Nama Penulis dan Lembaga Penulis		1	a. Tidak lengkap dan tidak konsisten (0) b. Lengkap tetapi tidak konsisten (0,5) c. Lengkap dan konsisten (1)
3	Abstrak	Dalam Bahasa Indonesia dan Bahasa Inggris yang baik, jumlah 150-200 kata. Isi terdiri dari latar belakang, metode, hasil, dan kesimpulan. Isi tertuang dengan kalimat yang jelas.	2	a. Tidak dalam Bahasa Indonesia dan Bahasa Inggris (0) b. Abstrak kurang jelas dan ringkas, atau hanya dalam Bahasa Inggris, atau dalam Bahasa Indonesia saja (1) c. Abstrak yang jelas dan ringkas dalam Bahasa Indonesia dan Bahasa Inggris (2)
4	Kata Kunci	Maksimal 5 kata kunci terpenting dalam paper	1	a. Tidak ada (0) b. Ada tetapi kurang mencerminkan konsep penting dalam artikel (0,5) c. Ada dan mencerminkan konsep penting dalam artikel (1)
5	Sistematika Pembaban	Terdiri dari pendahuluan, tinjauan pustaka, metode penelitian, hasil dan pembahasan, kesimpulan dan saran, daftar pustaka	1	a. Tidak lengkap (0) b. Lengkap tetapi tidak sesuai sisetm (0,5) c. Lengkap dan bersistem (1)
6	Pemanfaatan Instrumen Pendukung	Pemanfaatan Instrumen Pendukung seperti gambar dan tabel	1	a. Tak termanfaatkan (0) b. Kurang informatif atau komplementer (0,5) c. Informatif dan komplementer (1)
7	Cara Pengacuan dan Pengutipan		1	a. Tidak baku (0) b. Kurang baku (0,5) c. Baku (1)
8	Penyusunan Daftar Pustaka	Penyusunan Daftar Pustaka	1	a. Tidak baku (0) b. Kurang baku (0,5) c. Baku (1)
9	Peristilahan dan Kebahasaan		2	a. Buruk (0) b. Baik (1) c. Cukup (2)
10	Makna Sumbangan bagi Kemajuan		4	a. Tidak ada (0) b. Kurang (1) c. Sedang (2) d. Cukup (3) e. Tinggi (4)

Figure A.1: Form nilai bagian 1.

11	Dampak Ilmiah		7	a. Tidak ada (0) b. Kurang (1) c. Sedang (3) d. Cukup (5) e. Besar (7)
12	Nisbah Sumber Acuan Primer berbanding Sumber lainnya	Sumber acuan yang langsung merujuk pada bidang ilmiah tertentu, sesuai topik penelitian dan sudah teruji.	3	a. < 40% (1) b. 40-80% (2) c. > 80% (3)
13	Derajat Kemutakhiran Pustaka Acuan	Derajat Kemutakhiran Pustaka Acuan	3	a. < 40% (1) b. 40-80% (2) c. > 80% (3)
14	Analisis dan Sintesis	Analisis dan Sintesis	4	a. Sedang (2) b. Cukup (3) c. Baik (4)
15	Penyimpulan	Sangat jelas relevasinya dengan latar belakang dan pembahasan, dirumuskan dengan singkat	3	a. Kurang (1) b. Cukup (2) c. Baik (3)
16	Unsur Plagiat		0	a. Tidak mengandung plagiat (0) b. Terdapat bagian-bagian yang merupakan plagiat (-5) c. Keseluruhannya merupakan plagiat (-20)
TOTAL			**36**	
	Catatan : Nilai minimal untuk diterima **25**			

Figure A.2: form nilai bagian 2.

Appendix B

FAQ

M : Kalo Intership II atau TA harus buat aplikasi ? D : Ga harus buat aplikasi tapi harus ngoding

M : Pa saya bingung mau ngapain, saya juga bingung mau presentasi apa? D : Makanya baca de, buka jurnal topik 'ganteng' nah kamu baca dulu sehari 5 kali ya, 4 hari udah 20 tuh. Bingung itu tanda kurang wawasan alias kurang baca.

M : Pa saya sudah cari jurnal terindeks scopus tapi ga nemu. D : Kamu punya mata de? coba dicolok dulu. Kamu udah lakuin apa aja? tolong di list laporkan ke grup Tingkat Akhir. Tinggal buka google scholar klik dari tahun 2014, cek nama jurnalnya di scimagojr.com beres.

M : Pa saya belum dapat tempat intership, jadi ga tau mau presentasi apa? D : kamu kok ga nyambung, yang dipresentasikan itu yang kamu baca bukan yang akan kamu lakukan.

M : Pa ini jurnal harus yang terindex scopus ga bisa yang lain ? D : Index scopus menandakan artikel tersebut dalam standar semantik yang mudah dipahami dan dibaca serta bukan artikel asal jadi. Jika diluar scopus biasanya lebih sukar untuk dibaca dan dipahami karena tidak adanya proses review yang baik dan benar terhadap artikel.

M : Pa saya tidak mengerti D : Coba lihat standar alasan

M : Pa saya bingung D : Coba lihat standar alasan

M : Pa saya sibuk D : Mbahmu....

M : Pa saya ganteng D : Ndasmu....

M : Pa saya kece D : wes karepmu lah....

Biasanya anda memiliki alasan tertentu jika menghadapi kendala saat proses bimbingan, disini saya akan melakukan standar alasan agar persepsi yang diterima sama dan tidak salah kaprah. Penggunaan kata alasan tersebut antara lain :

1. Tidak Mengerti : anda boleh menggunakan alasan ini jika anda sudah melakukan tahapan membaca dan meresumekan 15 jurnal. Sudah mencoba dan mempraktekkan teorinya dengan mencari di youtube dan google minimal 6 jam sehari selama 3 hari berturut-turut.

2. Bingung : anda boleh mengatakan alasan bingung setelah maksimal dalam berusaha menyelesaikan tugas bimbingan dari dosen(sudah dilakukan semua). Anda belum bisa mengatakan alasan bingung jika anda masih belum menyelesaikan tugas bimbingan dan poin nomor 1 diatas. Setelah anda menyelesaikan tugas bimbingan secara maksimal dan tahap 1 poin diatas, tapi anda masih tetap bingung maka anda boleh memakai alasan ini.

Bibliography