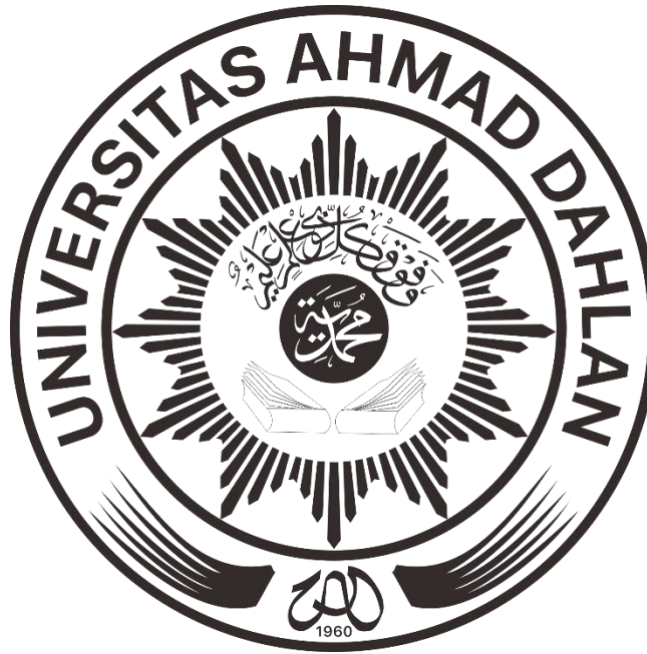


PETUNJUK PRAKTIKUM ALGORITMA PEMROGAMAN 2
SI/ALPRO2/II/R-2



LABORATORIUM PEMBELAJARAN
PROGRAM STUDI SISTEM INFORMASI
FAKULTAS SAINS DAN TEKNOLOGI TERAPAN
UNIVERSITAS AHMAD DAHLAN
2023

Keterangan Pengisian Kode Petunjuk Praktikum :

XXX = Singkatan program studi
YYY = Singkatan mata praktikum
ZZ = Semester ke-
N = Revisi ke-

DAFTAR SEJARAH REVISI PETUNJUK PRAKTIKUM

Laboratorium : Pembelajaran
 Program Studi : Sistem Informasi
 Semester : Genap

NO	NAMA PETUNJUK PRAKTIKUM	KODE PETUNJUK PRAKTIKUM	REVISI KE-	TANGGAL REVISI
1	Algortima Pemrograman 2	SI/ALPRO2/II/R-1	1	17 Maret 2021
2	Algortima Pemrograman 2	SI/ALPRO2/II/R-2	2	12 Maret 2023

SEJARAH REVISI PETUNJUK PRAKTIKUM

Nama Petunjuk Praktikum : Modul Praktikum Desain dan Pengelolaan Jaringan
 Semester : Genap
 Program Studi : Sistem Informasi
 Fakultas : Sains dan Teknologi Terapan

REVISI KE	TANGGAL REVISI	URAIAN REVISI
1	17 Maret 2021	Perubahan modul praktikum menggunakan C++ menjadi menggunakan Phyton
2	12 Maret 2023	Penambahan penjelasan pada dasar pengetahuan dan penambahan ilustrasi beserta contoh-contoh sederhana.

Daftar Isi

Halaman Sampul

Daftar Sejarah Revisi Petunjuk Praktikum

Sejarah Revisi Petunjuk Praktikum

Daftar Isi

Kata Pengantar

PERTEMUAN 1. FUNCTION	7
PERTEMUAN 2. SCOPE VARIABLE	15
PERTEMUAN 3. INSERTION SORT	23
PERTEMUAN 4. SELECTION SORT	26
PERTEMUAN 5. BUBBLE SORT	30
PERTEMUAN 6. PENCARIAN	33
PERTEMUAN 7. REKURSI	39
PERTEMUAN 8. MODULE	42
PERTEMUAN 9. CLASS / KELAS	48
PERTEMUAN 10. FILE I/O	54

Penutup

Daftar Pustaka

Kata Pengantar

Puji syukur kita panjatkan kehadiran Allah SWT yang telah memberikan rahmat dan hidayah-Nya, sehingga buku petunjuk praktikum algoritma pemrograman 2 mahasiswa Sistem informasi, Fakultas Sains dan Teknologi Terapan, Universitas Ahmad Dahlan ini dapat diselesaikan. Buku ini disusun untuk dapat digunakan sebagai acuan penyelenggaraan praktikum laboratorium komputer Strategi Bisnis Teknologi Informasi dan Pengembangan Sistem Informasi.

Buku Panduan ini ditujukan untuk memenuhi kebutuhan informasi yang diperlukan oleh para mahasiswa dan juga dosen yang akan terlibat dalam proses kegiatan praktikum agar pelaksanaan dan penyelenggaraannya dapat berjalan dengan lebih baik lagi.

Selanjutnya diucapkan terima kasih dan penghargaan kepada semua pihak yang telah memberi bantuan hingga selesainya Buku Panduan ini khususnya kepada Tim Penyusun yang terlibat dalam penyusunan buku ini. Semoga Bermanfaat. Medan, Agustus 2016 Tim Penyusun

Yogyakarta, 12 Maret 2023

Tim Penyusun

Pertemuan I. FUNCTION

Tujuan :

1. Mahasiswa memahami fungsi / sub program dalam python
2. Mahasiswa dapat membuat fungsi / sub program dalam python

Dasar Pengetahuan :

Function atau Fungsi adalah blok instruksi atau subprogram (*routine*) yang digunakan untuk menjalankan tugas pemrograman tertentu. Fungsi yang dibuat dapat dipanggil dari program utama atau subprogram lain dalam kata lain fungsi dapat digunakan ulang tanpa membuat ulang (cukup mendefinisikan sekali) pada kebutuhan yang sama. Kode program juga dapat lebih modular, mudah dibaca dan dipelihara.

Pada python sudah banyak sekali fungsi dengan berbagai macam kebutuhan pemrograman, baik secara built in pada python atau free source melalui media internet. Namun, pada kebutuhan yang lebih spesifik diperlukan pendefinisian fungsi sendiri.

Secara umum dalam pemrograman, terdapat dua jenis fungsi, yaitu.

1. Fungsi tanpa nilai balik.

Fungsi yang tidak menghasilkan nilai pengembalian, hanya digunakan untuk melakukan proses tertentu. Contoh : `print()` hanya untuk melakukan pencetakan text atau bilangan ke layar. Tidak ada nilai yang dihasilkan yang dapat di proses lebih lanjut lagi.

2. Fungsi dengan nilai balik.

Fungsi yang menghasilkan nilai jika dipanggil. Contoh: `sqrt()` untuk menghitung akar dari nilai tertentu, misalkan `sqrt(81)` menghasilkan nilai 9 dimana nilai 9 dapat dioperasikan lagi ke fungsi lain atau operator biasa.

Mendefinisikan fungsi :

Format penulisan fungsi adalah :

def nama_fungsi(parameters):	→	diawali dengan kata kunci def diikuti nama fungsi tanda () dan :
"function_docstring".	→	docstring (biasanya untuk memberikan keterangan).
function_suite.	→	function_suite tempat mendefinisikan fungsi berdasarkan algoritma
return [expression]	→	return : untuk mengembalikan nilai output dan keluar dari fungsi expression: argument sebagai output/ nilai pengembalian

Keterangan:

- Blok fungsi dimulai dengan def kata kunci diikuti oleh nama fungsi dan tanda kurung ().
- Parameter atau argumen input apa pun harus ditempatkan di dalam tanda kurung ini. Anda juga dapat menentukan parameter di dalam tanda kurung ini.
- Pernyataan pertama dari suatu fungsi dapat berupa pernyataan opsional - string dokumentasi dari fungsi atau docstring. (biasanya untuk memberikan keterangan) , sebagai komentar tidak dieksekusi oleh program
- Blok kode dalam setiap fungsi dimulai dengan titik dua (:) dan diindentasi.
- Pernyataan return [ekspresi] keluar dari suatu fungsi pengembalian, secara opsional meneruskan kembali ekspresi ke pemanggil. Pernyataan return tanpa argumen sama dengan tidak ada pengembalian.

Contoh 1. fungsi mengambil string sebagai input dan perintah mencetaknya

```
def printme( str ):
    " Ini mencetak string "
    print (str)
    return
```

Kode diatas untuk mendefinisikan fungsi dengan nama printme, dengan satu parameter input str. Fungsi merupakan fungsi tanpa nilai balik, yaitu untuk mencetak masukan berupa string atau numerik.

Contoh 2. Fungsi tambahan dua nilai

```
def tambah( a,b ):
    "fungsi untuk menjumlahkan dua bilangan"
    Hasil=a+b
    Return [Hasil]
```

Kode ini untuk mendefinisikan fungsi dengan nama tambah, yaitu untuk menjumlahkan dua buah bilangan (parameter) inputan a sebagai parameter 1 dan b sebagai parameter 2. Fungsi ini merupakan contoh fungsi dengan nilai balik, yaitu dengan mengembalikan nilai pada ekspresi Hasil.

Mendefinisikan fungsi :

Fungsi yang telah di definisikan dengan benar, dapat digunakan dengan memanggilnya dari bagian kode atau didalam program. Secara umum, pemanggilan fungsi dilakukan dengan menyebutkan nama fungsi beserta nilai variabel yang inputkan ke fungsi sebagai parameter fungsi. Untuk memanggil fungsi tanpa nilai balik strukturnya sebagai berikut

```
Nama_fungsi(parameter)
```

Sebagai contoh, akan memanggil fungsi printme pada contoh 1 dengan input parameter “membuat fungsi!”

```
printme("membuat fungsi!")
```

Akan menghasilkan cetakan di layar

```
membuat fungsi!
```


Sedangkan untuk fungsi dengan nilai balik, dapat di tambahkan variabel penyimpan nilai, sebagai berikut:

```
Var= Nama_fungsi(parameter)
```

Sebagai contoh, akan memanggil fungsi tambah pada contoh 2 dengan input parameter “a=5” dan “b=2”.

```
jumlah= tambah(5,2)
```

Hasil penjumlahan akan tersimpan pada variabel jumlah, dalam pemanggilan ini nilai balik baru tersimpan dan belum di tampilkan dilayar, namun variabel jumlah sudah ada nilainya dan dapat di operasikan lagi atau dipakai pada fungsi yang lain. Untuk menampilkan hasil jumlah dapat dituliskan dengan perintah cetak (print) berikut sebagai contoh

```
Jumlah= tambah(5,2)
print(“hasil penjumlahan 5 dan 2 adalah ” +
str(Jumlah))
```

Akan menghasilkan cetakan di layar

```
hasil penjumlahan 5 dan 2 adalah 7
```

Fungsi dengan nilai balik merupakan ekspresi (nilai), maka pada pemanggilan fungsi tersebut, nilai dapat di simpan hasilnya ke dalam suatu variabel, pada contoh di atas hasil disimpan pada variabel Jumlah. Sehingga pemanggilan fungsi tersebut dapat juga diproses layaknya sebuah variabel, seperti contoh berikut:

```
print(“hasil penjumlahan 5 dan 2 adalah ” + str(tambah(5,2)))
```

Akan menghasilkan cetakan yang sama di layar, yaitu:

```
hasil penjumlahan 5 dan 2 adalah 7
```

Parameter di dalam fungsi fungsi :

Parameter merupakan suatu nilai/variabel yang di kirimkan ke dalam fungsi yang akan di proses didalam fungsi tersebut. Dengan menggunakan parameter, nilai luaran dari fungsi akan berubah setiap pemanggilan fungsi dengan nilai parameter yang berbeda. Berikut contoh sebagai ilustrasi dalam fungsi matematika :

$$f(a) = a + 2$$

f merupakan suatu fungsi dengan a sebagai parameter dan $a + 2$ adalah operasi yang akan di proses Ketika fungsi f di panggil. Sehingga nilai yang dikembalikan fungsi f bergantung pada nilai a . Misal:

$$f(2) = 2 + 2 = 4$$

$$f(13) = 13 + 2 = 15$$

Ketika parameter a diberikan nilai 2 maka akan menghasilkan nilai 4, sedangkan jika nilai a diberikan 13, maka akan memberikan hasil 15. Itulah sebagai ilustrasi dari parameter. Cara kerja fungsi di dalam pemrograman juga sama seperti cara kerja fungsi dalam bentuk matematika di atas.

Terdapat dua jenis parameter pada pemrograman yaitu parameter formal dan parameter aktual. Parameter formal adalah parameter yang terdapat pada bagian define fungsi tersebut. Pada contoh 2 yaitu kode program penjumlahan, parameter a dan b merupakan contoh parameter formal, karena terdefinisi di dalam fungsi. Ini biasa disebut dengan *parameter* saja. Sedangkan parameter actual adalah parameter yang dilewatkan pada saat pemanggilan fungsi. Berikut sebagai contoh:

```
P= tambah(5,2)
print("P")
x=5
y=2
Z= tambah(x,y)
print("Z")
```

Dalam contoh tersebut, 5,2,x, dan y merupakan parameter aktual. Dalam pemrograman, parameter aktual biasa disebut dengan *argument*.

Percobaan 1. Menghitung luas persegi Panjang

Identifikasi :

Input : Panjang (p), lebar (l)

Proses : $L = p \times l$

Output : L

Flowchart kasus percobaan 1, yaitu sebagai berikut :



Fungsi luas persegi Panjang tanpa expresi

```
def LuasPP(p,l):
    "menghitung luas persegi panjang"
    L=p*l
    print(L)
    return
```

```
A=LuasPP(4,3)
print(A)
S=3*A
print(S)
```

Fungsi luas persegi Panjang dengan expresi

```
def LuasPP(p,l):
    "menghitung luas persegi panjang"
    L=p*l
    print(L)
    return L
```

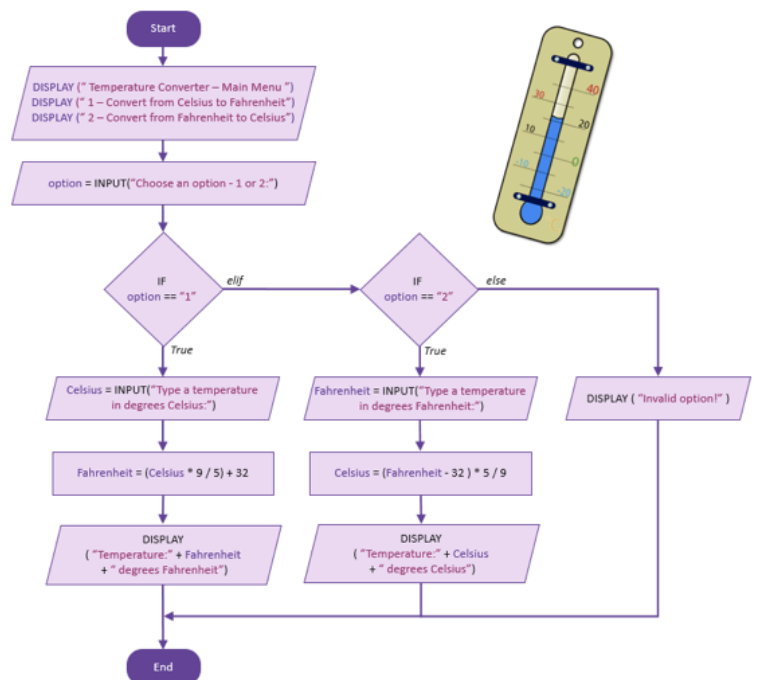
```
A=LuasPP(4,3)
print(A)
S=3*A
print(S)
```

Perhatikan perbedaanya.

Percobaan 2. Menkonversi temperature

$$^{\circ}\text{C} = (^{\circ}\text{F} - 32) \times 5 / 9$$

$$^{\circ}\text{F} = (^{\circ}\text{C} \times 9 / 5) + 32$$



```
def CF(c):
    "fungsi mengubah celcius ke fahrenheit"
    f=(c*9/5)+32
    return f
```

```
def FC(f):
    "fungsi mengubah F ke C"
    C=(f-32)*5/9
    return C
```

```
CF(30)
```

```
pil=1
suhu=50
if pil==1:
    sh=CF(suhu)
else:
    sh=FC(suhu)
print(sh)
```

122.0

Tugas.

1. Buatlah fungsi untuk menghitung luas lingkaran.
2. Buatlah fungsi untuk menghitung nilai minimum.
3. Buatlah fungsi untuk menghitung nilai rata-rata.

Pertemuan 2. SCOPE VARIABLE

Tujuan :

1. Mahasiswa dapat memahami scope variable
2. Mahasiswa dapat memahami variabel global dan variabel lokal
3. Mahasiswa dapat membedakan variabel global dan variabel lokal
4. Mahasiswa dapat menerapkan penggunaan variabel global dan variabel lokal

Dasar Pengetahuan:

Sebuah variabel di dalam sebuah program memiliki skop jangkauan variabel tertentu. Skop variabel terdiri dari: Variabel lokal dan Variabel global.

Variabel global adalah variabel yang dapat diakses di mana saja. Variabel lokal adalah kebalikannya, hanya dapat diakses di dalam frame/ blok/ fungsinya-nya. Perbedaannya adalah bahwa variabel global dapat diakses secara lokal, tetapi tidak dimodifikasi secara lokal. Sedangkan, Variabel lokal tidak dapat diakses secara global, secara inheren.

Variabel global

- Variabel yang dikenal diseluruh daerah di dalam program, di dalam dan luar fungsi.
- Dideklarasikan di luar suatu blok statemen atau di luar fungsi-fungsi yang menggunakannya.

Variabel lokal

- Variabel yang hanya dikenal di daerah yang lokal saja, misalnya di dalam sebuah fungsi/prosedur tertentu saja dan tidak dikenal di daerah lainnya.
- Harus dideklarasikan di dalam blok yang bersangkutan
- Variabel lokal akan dihapus dari memori bila proses sudah meninggalkan blok statemen letak variabel lokalnya

Contoh variable global.

Dalam Python, variabel yang dideklarasikan di luar fungsi atau dalam lingkup global dikenal sebagai variabel global. Ini berarti, variabel global dapat diakses di dalam atau di luar fungsi.

```
x = "global"

def contohG():
    print("x dalam :", x)

contohG()
print("x luar:", x)

x dalam : global
x luar: global
```

Fungsi diatas x di deklarasikan sebagai variabel global dan mendefinisikan fungsi contohG untuk mencetak variabel global tersebut. Bagaimana jika nilai x diubah di dalam fungsi tersebut.

```
: x = "global"

def contohG():
    x = x * 2

contohG()
print("x luar:", x)

-----
UnboundLocalError                                Traceback (most recent call last)
<ipython-input-5-b9ef904e8342> in <module>
      4     x = x * 2
      5
----> 6 contohG()
      7 print("x luar:", x)

<ipython-input-5-b9ef904e8342> in contohG()
      2
      3 def contohG():
----> 4     x = x * 2
      5
      6 contohG()

UnboundLocalError: local variable 'x' referenced before assignment
```

Output menunjukkan pesan error, karena x di dalam fungsi sebagai variabel local dan tidak di deklarasikan di dalam fungsi.

Contoh variable lokal.

Variabel yang dideklarasikan di dalam fungsi atau dalam lingkup lokal dikenal sebagai variabel lokal.

```
def contohL():
    y = "local"

contohL()
print(y)
```

```
-----
NameError                                Traceback (most recent call last)
<ipython-input-7-25f7e8db03ca> in <module>
      3
      4 contohL()
----> 5 print(y)

NameError: name 'y' is not defined
```

Outputnya menunjukkan kesalahan, karena variabel lokal y di panggil dalam lingkup global sedangkan variabel lokal hanya berfungsi di dalam fungsi contohL () atau cakupan lokal. Secara umum, variabel local di deklarasikan di dalam fungsi, seperti berikut :

```
def contohL():
    y = "local"
    print(y)
contohL()
```

```
local
```

Variabel Global and lokal

Bagaimana menggunakan variabel global dan local dalam kode yang sama.


```

x = "global"

def GL():
    global x
    y = "local"
    x = x * 2
    print(x)
    print(y)

GL()

globalglobal
local

```

Dalam kode di atas, x dideklarasikan sebagai global dan y sebagai variabel lokal di GL (). Kemudian, operator perkalian * digunakan untuk memodifikasi variabel global x dan mencetak x dan y. Setelah memanggil GL (), nilai x menjadi global global karena menggunakan x * 2 untuk mencetak dua kali global. Setelah itu, mencetak nilai variabel lokal y.

Variabel global dan lokal dengan nama yang sama.

```

x = 5

def GL2():
    x = 10
    print("local x:", x)

GL2()
print("global x:", x)

local x: 10
global x: 5

```

Di dalam kode di atas, nama x digunakan sebagai variabel local dan global secara bersamaan. Ketika di run, maka didapatkan hasil yang berbeda walaupun nama variabelnya sama, karena variabel tersebut dideklarasikan di dua cakupan yaitu lokal di dalam fungsi GL2() dan global di luar fungsi GL2().

Perintah cetak di dalam fungsi GL2() menghasilkan local x : 10, ini disebut lingkup lokal.

Perintah cetak di luar fungsi GL2() menghasilkan global x:5, ini disebut lingkup global.

Variabel NonLocal

Variabel nonlokal digunakan dalam fungsi bersarang yang cakupan lokalnya tidak ditentukan. Ini berarti, variabelnya tidak bisa dalam lingkup lokal maupun global.

Mari lihat contoh bagaimana variabel global dibuat dengan Python.

kata kunci ***nonlocal*** digunakan untuk membuat variabel nonlokal.

<pre>def outer(): x = "local" def inner(): nonlocal x x = "nonlocal" print("inner:", x) inner() print("outer:", x) outer()</pre> <p>inner: nonlocal outer: nonlocal</p>	<pre>def outer(): x = "local" def inner(): nonlocal x x = "test" print("inner:", x) inner() print("outer:", x) outer()</pre> <p>inner: test outer: test</p>	<pre>def outer(): x = "local" def inner(): #nonlocal x x = "nonlocal" print("inner:", x) inner() print("outer:", x) outer()</pre> <p>inner: nonlocal outer: local</p>
---	---	---

Dalam kode di atas ada fungsi bersarang inner (). Fungsi inner () didefinisikan dalam lingkup fungsi outer lainnya ().

Catatan: Jika nilai variabel nonlocal di ubah, perubahan akan muncul di variabel lokal.

Contoh Kasus :

Program untuk pengelolaan buku, dimana beberapa fungsi yang digunakan adalah :

1. Fungsi untuk menampilkan data list **buku**.
2. Fungsi untuk menambahkan ke list **buku**.
3. Fungsi untuk mengedit data di list **buku**.
4. Fungsi untuk menghapus data dari list **buku**.

Perhatikan bahwa semua fungsi memerlukan dan saling terkait dengan satu variabel yaitu variabel **buku**. Sehingga variabel **buku** harus bisa di akses oleh semua fungsi, maka variabel **buku** tersebut disebut dengan variabel **global**.

Tugas.

buatlah flowchart dari program di atas. Ikuti kode program berikut penyelesaian contoh kasus.

```
[1]: from IPython.display import clear_output
      # Variabel global untuk menyimpan data Buku
      buku = []
      kl=1;
```

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

```
[4]: show_data()

      BELUM ADA DATA
```

```
[5]: # fungsi untuk menambah data
      def insert_data():
          buku_baru = input("Judul Buku: ")
          buku.append(buku_baru)
```

```
[6]: insert_data()

      Judul Buku: dds
```

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

```
[9]: edit_data()
```

```
dds
Inputkan ID buku: 0
Judul baru: rew
```

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

```
[11]: delete_data()
```

```
rew
Inputkan ID buku: 0
```

```
[12]: # fungsi untuk menampilkan menu
def show_menu():
    Tr=1
    print ("\n")
    print ("----- MENU -----")
    print ("[1] Show Data")
    print ("[2] Insert Data")
    print ("[3] Edit Data")
    print ("[4] Delete Data")
    print ("[5] Exit")

    menu = int(input("PILIH MENU> "))
    print ("\n")
    clear_output(wait=True)
    if (menu == 1):
        show_data()
    elif (menu == 2):
        insert_data()
    elif (menu == 3):
        edit_data()
    elif (menu == 4):
        delete_data()
    elif (menu == 5):
        Tr=0;
    else:
        print ("Salah pilih!")
    return Tr
```

```
[13]: show_menu()
```

```
BELUM ADA DATA
```

Tugas : gabungkan semua fungsi di atas dalam satu blok

```
In [ ]:
```

Di akhir tambahkan sytax berikut :

```
: if __name__ == "__main__":  
    Tr=1  
    while(Tr==1):  
        Tr=show_menu()  
        clear_output(wait=True)
```

Pertemuan 3. Insertion Sort

Tujuan:

1. Mahasiswa mampu menerapkan perulangan bersarang pada kasus pengurutan penyisipan.
2. Mahasiswa mampu membuat kode program pengurutan penyisipan.

Dasar Pengetahuan:

Algoritma pengurutan insertion dapat di ilustrasikan seperti pengurutan pada kartu remi. Data di cek satu persatu mulai data ke dua sampai data terakhir. Jika di temukan data yang lebih kecil dari data sebelumnya , maka data tersebut di sisipkan (insert) pada posisi yang sesuai.

Ilustrasi

3	10	4	6	8	9	7	2	1	5
---	----	---	---	---	---	---	---	---	---

Dua bil pertama sdh terurut secara relatif

3	10	4	6	8	9	7	2	1	5
---	----	---	---	---	---	---	---	---	---

Sisipkan bilangan ketiga ke dalam bagian abu

3	10	4	6	8	9	7	2	1	5
---	----	---	---	---	---	---	---	---	---

1. Ambil bilangan ketiga (4)

3	10		6	8	9	7	2	1	5
---	----	--	---	---	---	---	---	---	---

2. Geser bilangan kedua (10)

3	4	10	6	8	9	7	2	1	5
---	---	----	---	---	---	---	---	---	---

3. Sisipkan bilangan 4 ke posisi yang tepat

tiga bilangan pertama sudah terurut secara relatif dan kita sisipkan bilangan keempat kepada tiga bilangan pertama tsb. Setelah penyisipan, empat bilangan pertama haruslah dalam keadaan terurut secara relatif.

3	4	6	10	8	9	7	2	1	5
---	---	---	----	---	---	---	---	---	---

Ulangi proses tsb sampai bilangan terakhir disisipkan

3	4	6	8	10	9	7	2	1	5
---	---	---	---	----	---	---	---	---	---

3	4	6	8	9	10	7	2	1	5
---	---	---	---	---	----	---	---	---	---

3	4	6					2	1	5
---	---	---	--	--	--	--	---	---	---

8	9	10
---	---	----

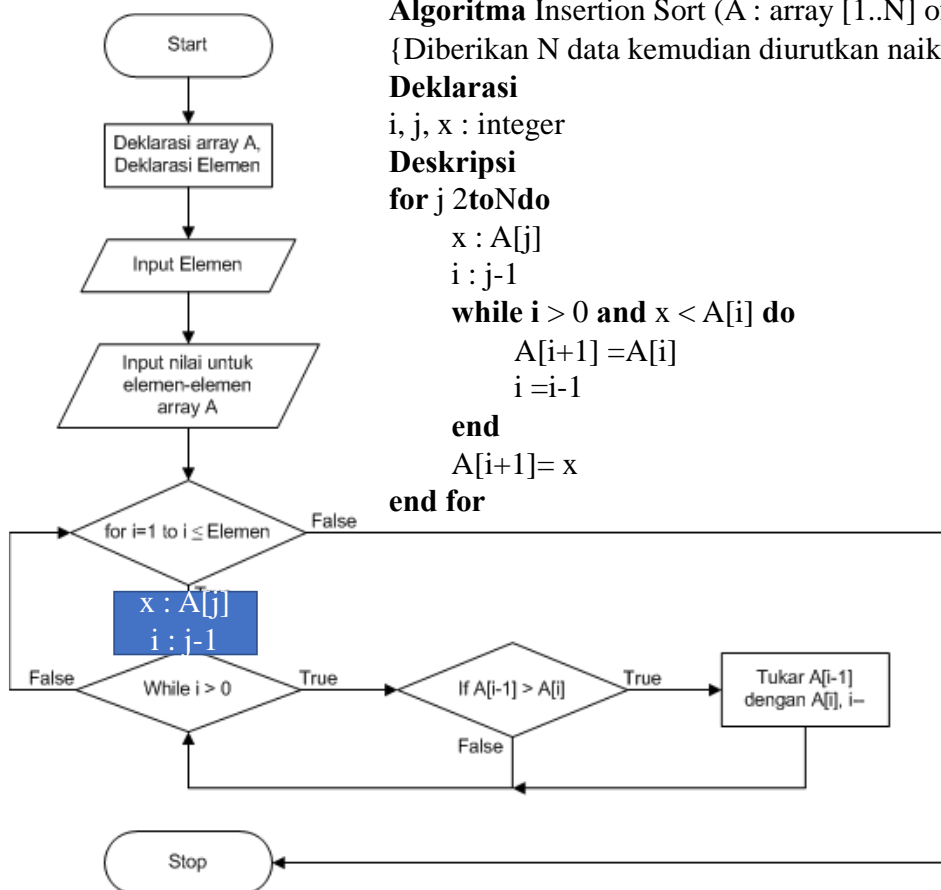
3	4	6	7	8	9	10	2	1	5
---	---	---	---	---	---	----	---	---	---

2	3	4	6	7	8	9	10	1	5
---	---	---	---	---	---	---	----	---	---

1	2	3	4	6	7	8	9	10	5
---	---	---	---	---	---	---	---	----	---

1	2	3	4	5	6	7	8	9	10
---	---	---	---	---	---	---	---	---	----

Flowchart dan Algoritma

**Kode Program (Percobaan) :**

```

def insertion_sort(sort_list):
    for i in range(1, len(sort_list)):
        key = sort_list[i]
        j = i - 1
        while j >= 0 and key < sort_list[j]:
            sort_list[j + 1] = sort_list[j]
            j -= 1
        sort_list[j + 1] = key
        print('iterasi', i-1, sort_list)

```

A=[4,3,5,6,2,78,98]

insertion_sort(A)

Latihan/Tugas :

1. Lakukan modifikasi fungsi pada percobaan untuk melakukan pengurutan data secara Descending.
2. Modifikasi fungsi tersebut sehingga menampilkan hasil tiap langkah pengurutan seperti contoh berikut :

```
A=[4,3,5,6,2,78,98]
insertion_sort(A)
```

```
iterasi 0 [3, 4, 5, 6, 2, 78, 98]
iterasi 1 [3, 4, 5, 6, 2, 78, 98]
iterasi 2 [3, 4, 5, 6, 2, 78, 98]
iterasi 3 [2, 3, 4, 5, 6, 78, 98]
iterasi 4 [2, 3, 4, 5, 6, 78, 98]
iterasi 5 [2, 3, 4, 5, 6, 78, 98]
```

Atau lebih lengkapnya (lebih baik)

```
A=[4,3,5,6,2,78,98]
insertion_sort(A)
```

```
pergeseran pada iterasi ke 0 j ke : 0 [4, 4, 5, 6, 2, 78, 98]

iterasi 0 [3, 4, 5, 6, 2, 78, 98]
iterasi 1 [3, 4, 5, 6, 2, 78, 98]
iterasi 2 [3, 4, 5, 6, 2, 78, 98]
pergeseran pada iterasi ke 3 j ke : 3 [3, 4, 5, 6, 6, 78, 98]

pergeseran pada iterasi ke 3 j ke : 2 [3, 4, 5, 5, 6, 78, 98]

pergeseran pada iterasi ke 3 j ke : 1 [3, 4, 4, 5, 6, 78, 98]

pergeseran pada iterasi ke 3 j ke : 0 [3, 3, 4, 5, 6, 78, 98]

iterasi 3 [2, 3, 4, 5, 6, 78, 98]
iterasi 4 [2, 3, 4, 5, 6, 78, 98]
iterasi 5 [2, 3, 4, 5, 6, 78, 98]
```


Pertemuan 4. SELECTION SORT

Tujuan :

1. Mahasiswa mampu menerapkan perulangan bersarang pada kasus pengurutan seleksi.
2. Mahasiswa mampu membuat kode program pengurutan seleksi.

Dasar Pengetahuan :

Metode pengurutan selection sort yaitu dengan cara mencari data terkecil (ascending) atau terbesar (descending) yang kemudian menukarnya dengan data acuan(pivot/key). Dikatakan selection sort karena algoritma ini mencoba memilih satu per satu elemen data dari posisi awal, untuk mencari data paling kecil dengan mencatat posisi index-nya saja, lalu dilakukan pertukaran hanya sekali pada akhir setiap tahapan.

Berikut proses pencariannya untuk kasus ascending :

1. Lakukan pengulangan dari data 1 ke N-1.
2. Pada setiap pengulangan dicari data terkecil dengan membandingkan data $i+1$ sampai data terakhir (N)
3. Data terkecil yang ditemukan di tukar dengan data acuan (pivot/key) yaitu data ke-i.

Ilustrasi

3	10	4	6	8	9	7	2	1	5
---	----	---	---	---	---	---	---	---	---

1 2 3 4 5 6 7 8 9 10

Cek seluruh elemen array,
temukan nilai terkecil (1)
tukarkan posisinya dengan posisi nilai yang
tersimpan pada posisi pertama dari array (3)

3	10	4	6	8	9	7	2	1	5
---	----	---	---	---	---	---	---	---	---

1	10	4	6	8	9	7	2	3	5
---	----	---	---	---	---	---	---	---	---

Temukan nilai terkecil kedua (2), dan tukarkan posisinya
dengan nilai yang berada pada posisi kedua (10).

1	10	4	6	8	9	7	2	3	5
---	----	---	---	---	---	---	---	---	---

1	2	4	6	8	9	7	10	3	5
---	---	---	---	---	---	---	----	---	---

Dua elemen biru pertama tidak akan berubah lagi sebab
mereka sudah merupakan nilai terkecil pertama dan kedua
dalam array tsb.

ulangi dengan cara/proses “pilih dan tukar”

1	2	4	6	8	9	7	10	3	5
1	2	3	6	8	9	7	10	4	5
1	2	3	6	8	9	7	10	4	5
1	2	3	4	8	9	7	10	6	5
1	2	3	4	8	9	7	10	6	5
1	2	3	4	8	9	7	10	6	5
1	2	3	4	5	9	7	10	6	8

1	2	3	4	5	9	7	10	6	8
1	2	3	4	5	6	7	10	9	8
1	2	3	4	5	6	7	10	9	8
1	2	3	4	5	6	7	10	9	8
1	2	3	4	5	6	7	8	9	10
1	2	3	4	5	6	7	8	9	10

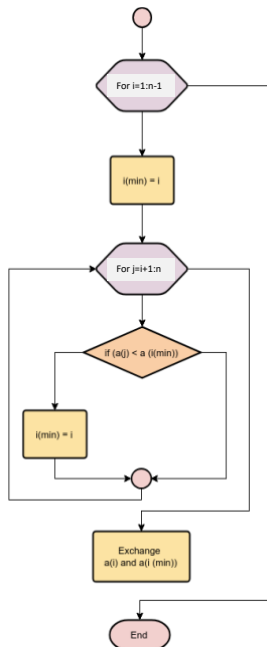
Indek key = 1:9

Pembanding : indeks setelah indeks key

Misal ada n data

indek key = i=1:n-1

Pembanding adalah j=i+1:n



1	Min 1			
2	ok	Min 2		
3	ok	ok 2	Min 3	
N-1				

Algoritma Selection Sort (A : array [1..N] of integer) {Diberikan
N data kemudian diurutkan naik }

Deklarasi

i, j, k : integer

Deskripsi

for i 1 to N-1 do

k=1

for j : i+1 to N do

if A[j] < A[k] then

k = j

end for

Tukar(A[i], A[k])

end for

Flowchart
dan
Algoritma

Ingat kembali proses penukaran

Kode Percobaan .

Kode Program :

```
def Selection_Sort(A):
    for i in range(len(A)-1):
        min_idx = i
        for j in range(i+1, len(A)):
            if A[min_idx] > A[j]:
                min_idx = j

        # penukaran
        temp=A[i]
        A[i]=A[min_idx]
        A[min_idx]=temp

    print('iterasi', i ,A)
```

```
def Selection_Sort(A):
    for i in range(len(A)-1):
        min_idx = i
        for j in range(i+1, len(A)):
            if A[j]<A[min_idx]:
                min_idx = j

        # penukaran
        temp=A[i]
        A[i]=A[min_idx]
        A[min_idx]=temp

    print('iterasi', i ,A)
```

Perhatikan yang di blok biru, apakah beda?

Bentuk lain kode python :

```
def Selection_Sort(A):
    for i in range(len(A)):
        min_idx = i
        for j in range(i+1, len(A)):
            if A[min_idx] > A[j]:
                min_idx = j
        A[i], A[min_idx] = A[min_idx], A[i] # pertukaran di python
```

perhatikan di pertukaran .

Tugas/Latihan.

1. Lakukan modifikasi fungsi pada percobaan untuk melakukan pengurutan data secara Descending.
2. Modifikasi fungsi tersebut sehingga menampilkan hasil tiap langkah pengurutan seperti contoh berikut :

```
A=[4,3,5,6,2,78,98]
Selection_Sort(A)
```

```
iterasi 0 [2, 3, 5, 6, 4, 78, 98]
iterasi 1 [2, 3, 5, 6, 4, 78, 98]
iterasi 2 [2, 3, 4, 6, 5, 78, 98]
iterasi 3 [2, 3, 4, 5, 6, 78, 98]
iterasi 4 [2, 3, 4, 5, 6, 78, 98]
iterasi 5 [2, 3, 4, 5, 6, 78, 98]
iterasi 6 [2, 3, 4, 5, 6, 78, 98]
```

Atau seperti ini lebih baik :

```
A=[4,3,5,6,2,78,98]
Selection_Sort(A)
```

```
indek key : 0 tukar : 4
    iterasi 0 [2, 3, 5, 6, 4, 78, 98]
indek key : 1 tukar : 1
    iterasi 1 [2, 3, 5, 6, 4, 78, 98]
indek key : 2 tukar : 4
    iterasi 2 [2, 3, 4, 6, 5, 78, 98]
indek key : 3 tukar : 4
    iterasi 3 [2, 3, 4, 5, 6, 78, 98]
indek key : 4 tukar : 4
    iterasi 4 [2, 3, 4, 5, 6, 78, 98]
indek key : 5 tukar : 5
    iterasi 5 [2, 3, 4, 5, 6, 78, 98]
indek key : 6 tukar : 6
    iterasi 6 [2, 3, 4, 5, 6, 78, 98]
```

Pertemuan 5. Bubble Sort

Tujuan :

1. Mahasiswa mampu menerapkan perulangan bersarang pada kasus pengurutan gelembung.
2. Mahasiswa mampu membuat kode program pengurutan gelembung.

Dasar Pengetahuan :

Algoritma Bubble Sort ini merupakan proses pengurutan yang secara berangsur-angsur berpindah ke posisi yang tepat karena itulah dinamakan Bubble yang artinya gelembung.

Secara sederhana, bisa didefinisikan algoritma Bubble Sort adalah pengurutan dengan cara pertukaran data dengan data disebelahnya secara terus menerus sampai dalam satu iterasi tertentu tidak ada lagi perubahan.

Ilustrasi (kasus ascending)

3 1 8 4 2

```
[1, 3, 8, 4, 2] bandingkan data ke 1 dengan data ke 2, bila data ke 1 lebih besar, maka tukar : (3-1) vs (1-3)
[1, 3, 8, 4, 2] bandingkan data ke 2 dengan data ke 3, bila data ke 2 lebih besar, maka tukar : tidak ada pertukaran
[1, 3, 4, 8, 2] bandingkan data ke 3 dengan data ke 4, bila data ke 3 lebih besar, maka tukar : (8-4) vs (4-8)
[1, 3, 4, 2, 8] bandingkan data ke 4 dengan data ke 5, bila data ke 4 lebih besar, maka tukar : (8-2) vs (2-8)
===== apakah sudah urut untuk semua langkah => tidak (masih ada proses pertukaran)
[1, 3, 4, 2, 8] bandingkan data ke 1 dengan data ke 2, bila data ke 1 lebih besar, maka tukar : tidak ada pertukaran
[1, 3, 4, 2, 8] bandingkan data ke 2 dengan data ke 3, bila data ke 2 lebih besar, maka tukar : tidak ada pertukaran
[1, 3, 2, 4, 8] bandingkan data ke 3 dengan data ke 4, bila data ke 3 lebih besar, maka tukar : (4-2) vs (2-4)
[1, 3, 2, 4, 8] bandingkan data ke 4 dengan data ke 5, bila data ke 4 lebih besar, maka tukar : tidak ada pertukaran
===== apakah sudah urut untuk semua langkah => tidak (masih ada proses pertukaran)
[1, 3, 2, 4, 8] bandingkan data ke 1 dengan data ke 2, bila data ke 1 lebih besar, maka tukar : tidak ada pertukaran
[1, 2, 3, 4, 8] bandingkan data ke 2 dengan data ke 3, bila data ke 2 lebih besar, maka tukar : (3-2) vs (2-3)
[1, 2, 3, 4, 8] bandingkan data ke 3 dengan data ke 4, bila data ke 3 lebih besar, maka tukar : tidak ada pertukaran
[1, 2, 3, 4, 8] bandingkan data ke 3 dengan data ke 4, bila data ke 3 lebih besar, maka tukar : tidak ada pertukaran
===== apakah sudah urut untuk semua langkah => tidak (masih ada proses pertukaran)
[1, 2, 3, 4, 8] bandingkan data ke 1 dengan data ke 2, bila data ke 1 lebih besar, maka tukar : tidak ada pertukaran
[1, 2, 3, 4, 8] bandingkan data ke 2 dengan data ke 3, bila data ke 2 lebih besar, maka tukar : (tidak ada pertukaran)
[1, 2, 3, 4, 8] bandingkan data ke 3 dengan data ke 4, bila data ke 3 lebih besar, maka tukar : tidak ada pertukaran
[1, 2, 3, 4, 8] bandingkan data ke 3 dengan data ke 4, bila data ke 3 lebih besar, maka tukar : tidak ada pertukaran
===== apakah sudah urut untuk semua langkah => Ya (tidak ada proses pertukaran)
```

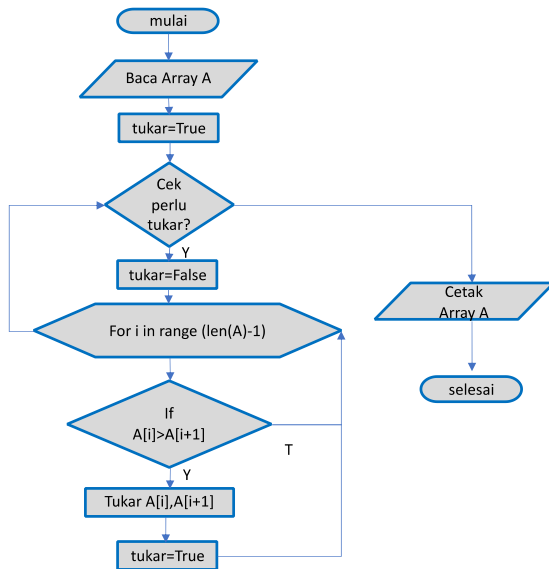
Pada langkah pertama
Jml element ada 5,
proses ada 4

Pada langkah kedua
Jml element ada 5,
proses ada 4

Misal jumlah
element array A
adalah **N (len(A))**
Maka jumlah iterasi
setiap langkah
adalah
N-1 atau len(A)-1

Jika kita perhatikan proses diatas, para proses **ketiga** data sudah terurut dengan benar. Tetapi algoritma Bubble Sort tetap berjalan hingga proses **ketiga** berakhir. Proses **keempat** masih terus berjalan karena pada algoritma Bubble Sort maksud terurut itu adalah tidak ada satupun penukaran pada suatu proses. Proses ketiga ini dilakukan untuk **verifikasi** data.

Flowchart (kasus Ascending)



Algoritma Bubble sort (A : array [1..N] of integer)
{Diberikan N data kemudian diurutkan naik }

Deklarasi

i, j, k : integer

Deskripsi

```

While tukar is true
  tukar=false
  for i : 1 to N-1 do
    if A[i] > A[i+1] then
      Tukar(A[i], A[i+1])
      tukar=True
    end if
  end for
end while
  
```

Kode Program (Percobaan) :

```

def bubble2(A):
    tukar=True
    while tukar:
        tukar=False
        for j in range(len(A)-1):
            if A[j]>A[j+1]:
                A[j], A[j+1] = A[j+1], A[j]
            tukar=True
        return A
  
```

Tugas / Latihan:

1. Lakukan modifikasi fungsi pada percobaan untuk melakukan pengurutan data secara Descending.
2. Modifikasi fungsi tersebut sehingga menampilkan hasil tiap langkah pengurutan seperti contoh berikut :

```
A = [3, 1, 8, 4, 2]  
bubble2(A)
```

```
tukar 1 dengan 3  
urutan menjadi [1, 3, 8, 4, 2]  
tukar 4 dengan 8  
urutan menjadi [1, 3, 4, 8, 2]  
tukar 2 dengan 8  
urutan menjadi [1, 3, 4, 2, 8]  
tukar 2 dengan 4  
urutan menjadi [1, 3, 2, 4, 8]  
tukar 2 dengan 3  
urutan menjadi [1, 2, 3, 4, 8]
```

Pertemuan 6. Pencarian

Tujuan :

1. Mahasiswa dapat menerapkan percabangan dan perulangan pada kasus pencarian
2. Mahasiswa memahami pencarian sequential dan biner
3. Mahasiswa dapat membuat kode program pencarian sequential dan biner

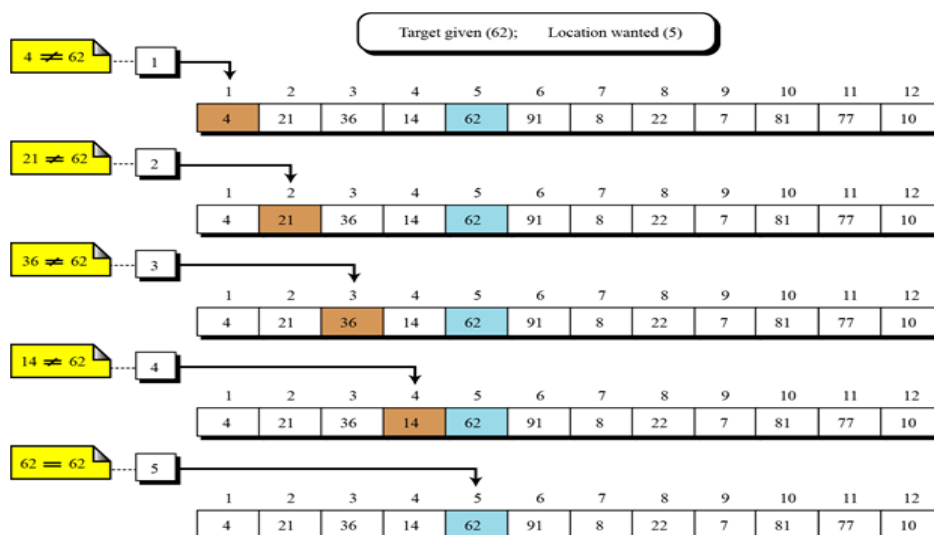
Dasar Pengetahuan :

Pencarian (Searching) yaitu proses menemukan suatu nilai tertentu pada kumpulan data. Hasil pencarian adalah salah satu dari tiga keadaan ini : (i) data ditemukan, (ii) data ditemukan lebih dari satu, atau (iii) data tidak ditemukan. Praktikum ini akan membahas dua cara pencarian yaitu Sequential Search dan Binary Search.

Sequential Search

Membandingkan data yang dicari dengan setiap elemen data satu-persatu secara beruntun, mulai dari elemen pertama sampai elemen yang dicari ditemukan atau seluruh elemen sudah dibandingkan.

Ilustrasi



Contoh Kasus

Diberikan data

2	3	3	4	5	6	3	4	7	8
---	---	---	---	---	---	---	---	---	---

Kasus 1. Temukan nilai 9



Data tidak di temukan

Kasus 2. Temukan nilai 5



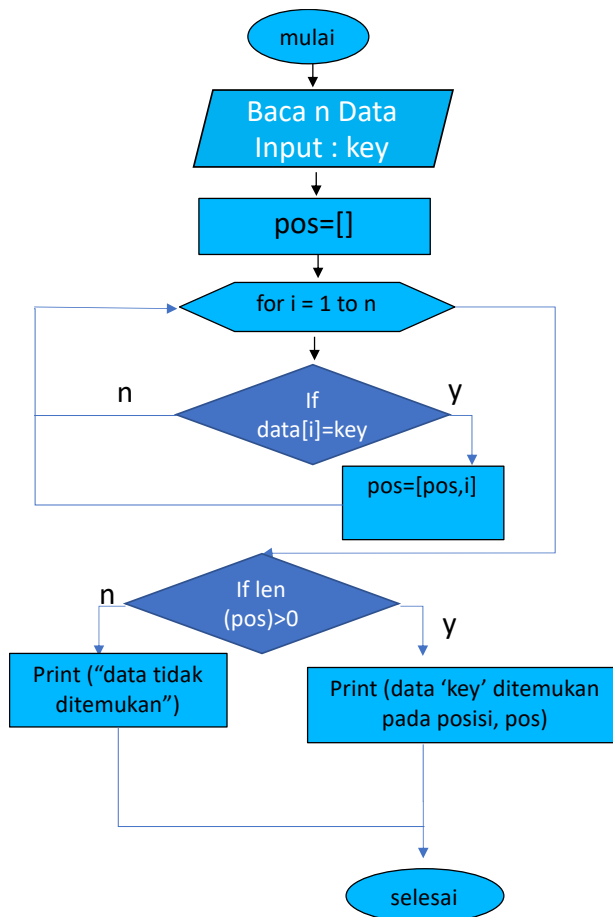
1 Data temuan di posisi ke : 5

Kasus 3. Temukan nilai 3



3 Data temuan di posisi ke : 2, 3 dan 7

Flowchart :

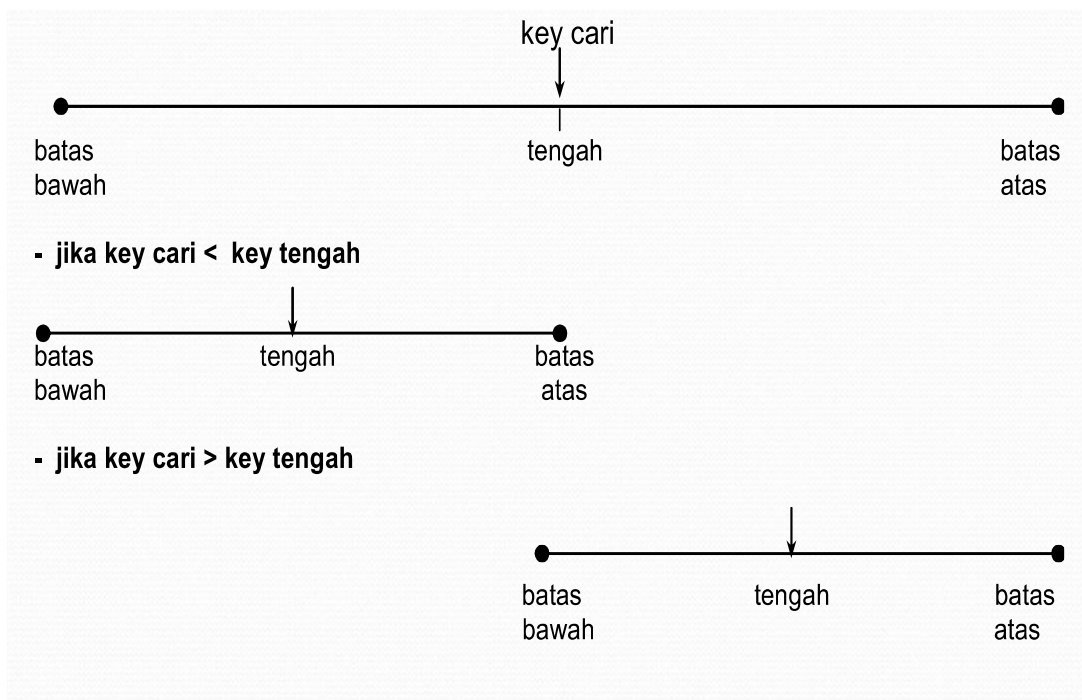


Kode Program (Percobaan) :

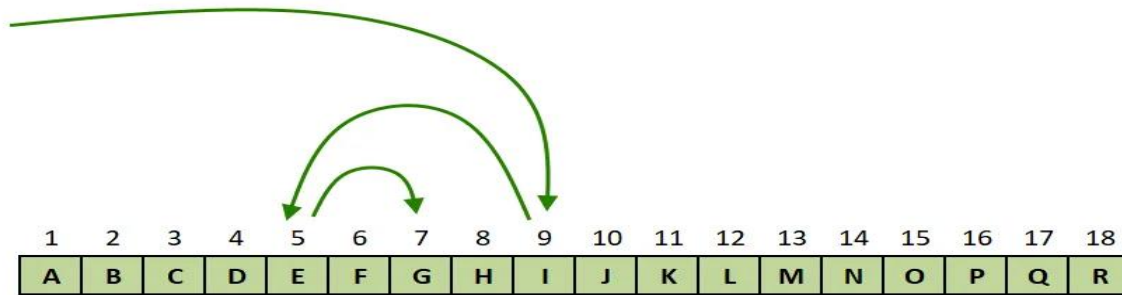
```
def sqsearch(data,key):
    pos=[]
    for i in range(len(data)):
        if data[i]==key:
            pos.append(i+1)
    if len(pos)>0:
        print('data', key, 'sebanyak ',len(pos),'ditemukan di posisi', pos)
    else:
        print('data tidak ditemukan')
    return pos
```

Binary Search

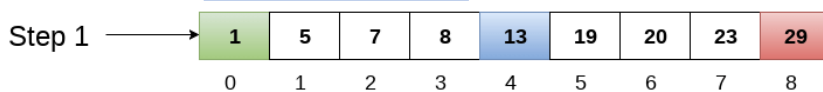
Dilakukan pada data yang elemennya telah terurut. Membandingkan elemen yang dicari dengan elemen data yang ada di posisi tengah, jika ditemukan maka proses akan berhenti, jika tidak maka proses dilanjutkan dengan mempersempit pencarian menjadi setengah dari jumlah elemennya. Proses ini diulang-ulang sampai data yang dicari ditemukan atau pencarian telah melampaui batas yang telah ditentukan.



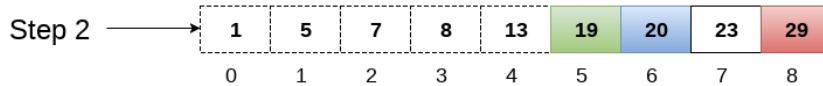
Ilustrasi :



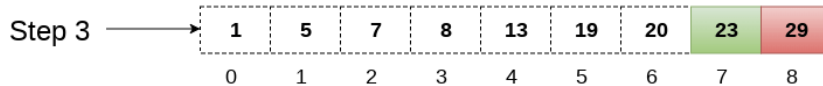
Key = 23



Batas bawah : 0
 Batas atas : 8
 $\text{mid} : (8+0)/2=4$
 $A[\text{mid}]=A[4]=13 < 23$



Batas bawah : $\text{mid}=5$
 Batas atas : 8
 $\text{mid} : (8+5)/2=13/2=6$
 $A[\text{mid}]=A[6]=20 < 23$



Batas bawah : $\text{mid}=7$
 Batas atas : 8
 $\text{mid} : (8+7)/2=14/2=7$
 $A[\text{mid}]=A[7]=23 == 23$

Algoritma Binary Search (A : array [1..N], Key) {Mencari Key dalam array A }

```

awal 1
akhir n
ketemu : false
while ( awal <= akhir ) and not ketemu do
  tengah ( awal + akhir ) / 2
  if key = A[tengah] then ketemu true
  else if key < A[tengah] then akhir else awal tengah+1
end if
end while
  
```

Percobaan :

```
def binsearch(data,key):
    awal=1
    akhir=len(data)+1
    ketemu= False
    while (awal<=akhir)and not ketemu:
        tengah=int((awal+akhir)/2)
        if key==data[tengah]:
            ketemu=True
            print('data', key, 'ditemukan di posisi', tengah)
        elif (key<data[tengah]):
            akhir=tengah-1
        else:
            awal=tengah+1
    if ketemu==False:
        print('data tidak ditemukan')
```

Tugas :

1. Modifikasi program sehingga dapat menampilkan jumlah iterasi yang diperlukan untuk mencari data yang dicari dengan menggunakan pencarian sequential dan binary.
2. cobalah kode anda untuk mencari angka 0 dalam NIM anda masing-masing, berapa iterasi yang diperlukan dengan menggunakan pencarian sequential dan binary
3. cobalah kode anda untuk mencari huruf i dalam kata "sistem informasi". Tentukan hasil dan jumlah iterasinya.
4. cobalah kode anda untuk mencari huruf a dalam "Nama anda masing-masing". Tentukan hasil dan jumlah iterasinya.

NOTE : untuk pengurutan string gunakan syntax :

```
".join(sorted(a))
```

```
: a='iwan tri riyadiyanto'
  res = ''.join(sorted(a))
  print(res)
```

```
aaadiiiinnorrttwyy
```

Contoh :

```
a='iwan tri riyadiyanto'  
res = ''.join(sorted(a))  
binsearch(res,i)
```

data i ditemukan di posisi 8 dari data setelah di urutkan
iterasi yang diperlukan 3

```
sqsearch(res,i)
```

data i sebanyak 4 ditemukan di posisi [7, 8, 9, 10]
iterasi yang diperlukan 20

Pertemuan 7. Rekursi

Tujuan :

1. Mahasiswa memahami fungsi rekursif
2. Mahasiswa mampu membuat kode program dengan menggunakan rekursif

Dasar Pengetahuan:

Rekursi : kemampuan suatu sub program / fungsi untuk memanggil dirinya sendiri.

Ciri fungsi rekursi:

- Kasus penyetop.
Dalam kasus ini terdapat nilai konstan (return value)
- Kasus pemanggilan rekursif.
Dalam kasus ini terdapat pemanggilan fungsi itu sendiri, tetapi harus mengarah kepada kasus penyetop.

Contoh 1:

Fungsi Faktorial.

$n! = n(n-1)(n-2)\dots(1)$.

Dapat dituliskan sebagai :

$n! = n(n-1)!$

dimana :

$n! = 1$ jika $n = 1$ atau $n = 0$

Analisis :

Kasus penyetop (= nilai awal) $n = 0$ atau $n = 1$ yaitu

bernilai konstan 1

Kasus rekursif :

faktorial (n) = n * faktorial (n-1)

```
def faktorial(n):
    if n == 0:
        return 1
    else:
        return n * faktorial(n - 1)
```

```
print(factorial(4))
```

24

Contoh 2.

Bilangan fibonacci ke n didefinisikan :

$$f_n = f_{n-1} + f_{n-2} \text{ dengan } f_0 = 0, f_1 = 1$$

Misalnya bilangan fibonacci ke 2 :

$$\begin{aligned} f_2 &= f_1 + f_0 \\ &= 1 + 0 \\ &= 1 \end{aligned}$$

Misalnya bilangan fibonacci ke 3 :

$$\begin{aligned} f_3 &= f_2 + f_1 \\ &= 1 + 1 \\ &= 2 \end{aligned}$$

Analisis :

Kasus penyetop $n = 0$ maka $f_0 = 0$, $n = 1$ maka $f_1 = 1$,

Kasus rekursif :

$$Fib(n) = Fib(n - 1) + Fib(n - 2)$$

```
def fib(n):
    if n==0:
        return 0
    elif n==1:
        return 1
    else:
        return (fib(n-1)+fib(n-2))
```

```
fib(7)
```

13

Tugas/Latihan :

1. Ubahlah fungsi fibonacci di atas ke dalam bentuk iterasi/perulangan menggunakan for atau while.
2. Diberikan fungsi untuk menghitung nilai dua pangkat x (2^x)
 - a. Buat analisis kasus penyetop dan kasih rekursinya
 - b. Buat program untuk menghitungnya dalam rekursi
 - c. Buat programnya dengan iterasi/perulangan.

Pertemuan 8. MODULE

Tujuan :

1. Mahasiswa memahami module dalam python
2. Mahasiswa mampu membuat module dalam python

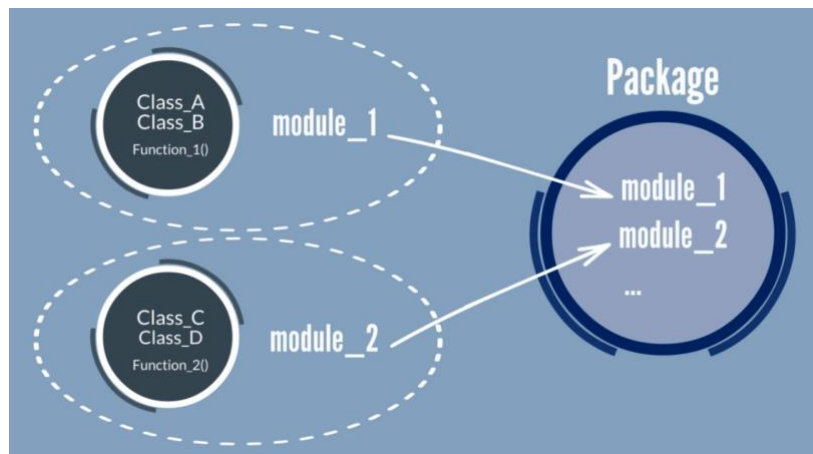
Dasar Pengetahuan :

Secara sederhana, modul adalah file yang terdiri dari kode. Modul dapat mendefinisikan fungsi, kelas, dan variabel. Modul juga dapat menyertakan kode “runnable”. program kami bertambah **besar** dalam ukuran, dimungkinkan untuk membaginya menjadi beberapa file untuk pemeliharaan yang lebih mudah serta kegunaan ulang kode.

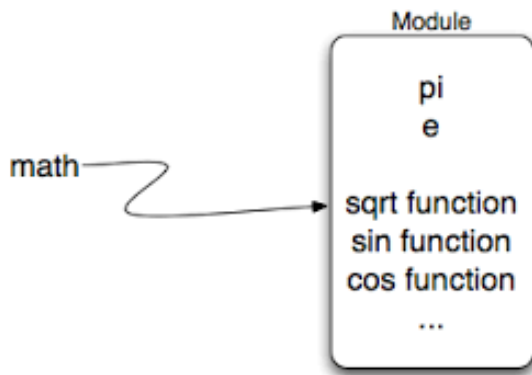
Solusi untuk ini adalah Modul.

Fungsi yang sering digunakan tidaklah perlu disalin definisinya ke dalam program yang berbeda. Modul dapat diimpor oleh program lain untuk memanfaatkan fungsinya.

Module => library fungsi



Modul memungkinkan untuk mengatur kode Python secara logis. Pengelompokan kode terkait ke dalam modul membuat kode lebih mudah dipahami dan digunakan.



File yang berisi kode Python, misalnya: math.py, disebut modul.

Modul vs. Fungsi

Fungsi: ini adalah blok kode yang dapat digunakan kembali dengan memanggilnya dengan kata kunci.

Modul: ini adalah file .py yang berisi daftar fungsi (juga dapat berisi variabel dan kelas).

Misalnya. Modul statistics : statistics.mean (a), mean adalah fungsi yang ditemukan dalam modul statistik. Statistic.modus(a),

Type Module

1. Module bawaan (built in) yang dapat diimpor dengan menggunakan kata kunci "import".

```
import math

content = dir(math)
print (content)

['__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']

math.sin(math.pi/2)

1.0
```

2. modul yang dibuat sendiri dan menggunakannya dalam program.

Membuat Module :

Kode Python untuk modul bernama NAME biasanya berada di file bernama NAME.py.

Berikut adalah contoh modul sederhana, contoh.py

Berikut contohnya :

```
*contoh.py - /Users/iwantririyadiyanto/contoh.py (3.7.0)*
def panggil_nama>Nama):
    print('Hello ', Nama)
    return

def NIM(nim):
    print('NIM :', nim)
    return
```

File fungsi saya simpan di folder iwantririyadiyanto dengan nama modul contoh.py yang berisi 2 fungsi :
Fungsi panggil nama
Fungsi NIM

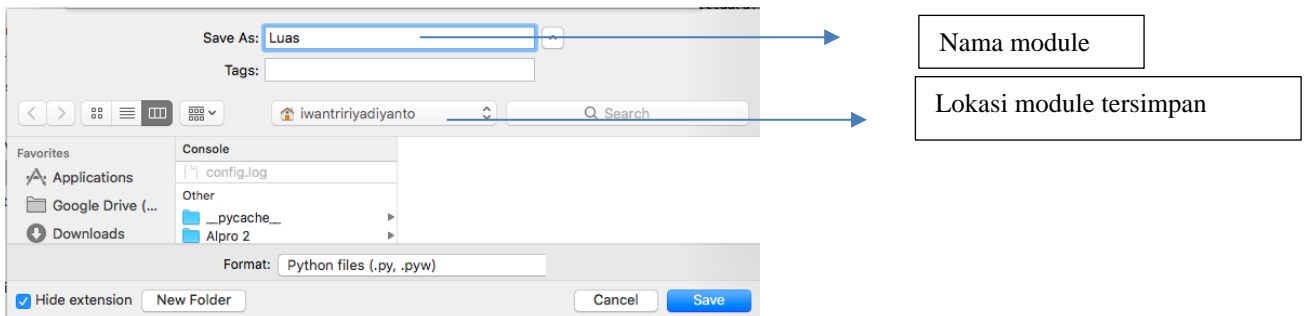
Buka idle phyton , buat new file module :

```
*contoh.py - /Users/iwantririyadiyanto/contoh.py (3.7.0)*
def panggil_nama>Nama):
    print('Hello ', Nama)
    return

def NIM(nim):
    print('NIM :', nim)
    return
```

Kumpulan beberapa fungsi

Simpan dalam dalam folder yang di inginkan :



Memanggil dan menggunakan module:

Import module

file sumber Python dapat digunakan sebagai modul dengan menjalankan pernyataan impor di beberapa file sumber Python lainnya. Impor memiliki sintaks berikut :

```
import module1[, module2[,... moduleN]
```

penerjemah mengimpor modul jika modul ada pada jalur pencarian (search path). Jalur pencarian adalah daftar direktori yang dicari oleh penerjemah sebelum mengimpor modul. Misalnya, untuk mengimpor modul contoh.py,

```
#import module contoh
import contoh
```

perintah diletakkan di bagian atas skrip, dan fungsi yang ada di dalam modul dapat di panggil sbb:

```
contoh.panggil_nama('iwan tri riyadi yanto')
```

```
Hello iwan tri riyadi yanto
```

```
contoh.NIM(45345)
```

```
NIM : 45345
```

```
contoh.panggil_nama('rofi')
contoh.NIM(34353)
```

```
Hello rofi
NIM : 34353
```

Form... import

dalam Python memungkinkan mengimpor atribut spesifik dari modul.

sintaks berikut:

```
from mod_name import name1[, name2[, ... nameN]]
```

Nama fungsi dalam modul dalam pemanggilan juga dapat di kostum sesuai keinginan

sintaks berikut:

```
from mod_name import name1 as Nama_baru
```

Contoh pemanggilan:

```

: from contoh import NIM
: NIM(934)
NIM : 934

: panggil_nama('iwan')
-----
NameError                                Traceback (most recent call last)
<ipython-input-4-27cblade81e5> in <module>
----> 1 panggil_nama('iwan')

NameError: name 'panggil_nama' is not defined

: from contoh import NIM as nomor
: nomor(435834)
NIM : 435834

: from contoh import panggil_nama as pn
: pn('iwan tri riyadi yanto')
Hello iwan tri riyadi yanto

```

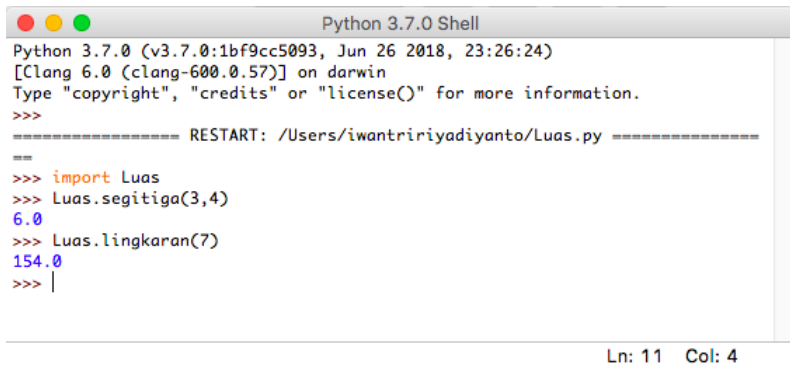
Lokasi modul:

Ketika mengimpor modul, interpreter Python mencari modul dalam urutan berikut:

1. Direktori saat ini
2. Jika modul tidak ditemukan, Python kemudian mencari setiap direktori dalam variabel shell PYTHONPATH.
3. Jika semuanya gagal, Python memeriksa path default. Pada UNIX, path default ini biasanya /usr/local/lib/python3/.

Jalur pencarian modul disimpan dalam sistem modul sistem sebagai variabel sys.path. Variabel sys.path berisi direktori saat ini, PYTHONPATH, dan default yang bergantung pada instalasi.

1. Berikut menggunakan module di idle phyton



```
Python 3.7.0 Shell
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 26 2018, 23:26:24)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /Users/iwantririyadiyanto/Luas.py =====
>>> import Luas
>>> Luas.segitiga(3,4)
6.0
>>> Luas.lingkaran(7)
154.0
>>> |
```

2. Berikut menggunakan module di jupyter

Import module

Import module

```
In [1]: import Luas
```

```
In [4]: Luas.segitiga(3,4)
```

```
Out[4]: 6.0
```

```
In [5]: Luas.lingkaran(7)
```

```
Out[5]: 154.0
```

```
In [ ]: |
```

Note : jika akan menambahkan fungsi, class atau variabel dalam module, supaya dapat di akses oleh phyton atau jupyter, run module terlebih dahulu.

Untuk yang jupyter, notebook yg running di shutdown , kemudian buka lagi.

Tugas /Latihan:

1. Membuat module yang berisi fungsi perhitungan
 - penjumlahan
 - pengurangan
 - perkalian
 - pembagian
2. buatlah module yang dapat menyelesaikan permasalahan di sekitar anda masing-masing.

Pertemuan 9. Class / Kelas

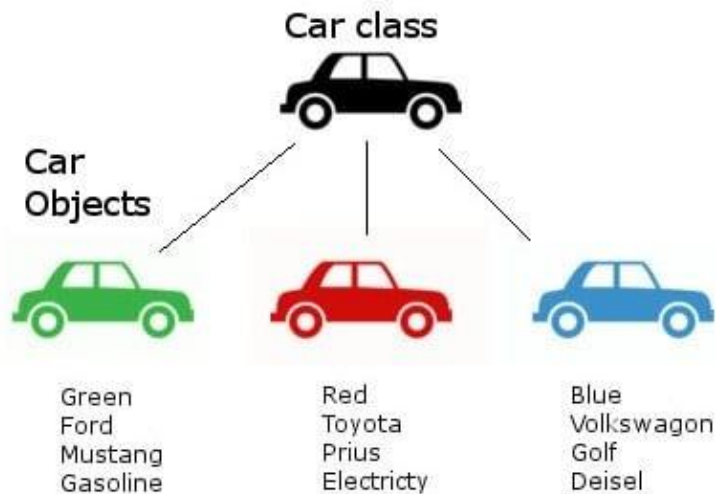
Tujuan :

1. Mahasiswa memahami class dalam python
2. Mahasiswa mampu membuat class dalam python

Dasar Pengetahuan :

Pemrograman berdasarkan konsep "objek", yang dapat berisi data, dalam bentuk *field* atau dikenal juga sebagai atribut; serta kode, dalam bentuk fungsi/prosedur atau dikenal juga sebagai *method*. Semua data dan fungsi di rangkai dalam *kelas-kelas* atau *objek-objek*. Setiap objek dapat menerima pesan, memproses data, dan mengirim pesan ke objek lainnya,

Ilustrasi



Tinjauan Umum tentang Terminologi OOP

Kelas (class) - Prototipe yang ditentukan pengguna untuk objek yang mendefinisikan sekumpulan atribut yang mengkarakterisasi objek apa pun dari kelas. Atribut adalah anggota data (variabel kelas dan variabel contoh) dan metode, diakses melalui notasi titik.

Variabel kelas (class variable) - Variabel yang dibagikan oleh semua instance kelas. Variabel kelas didefinisikan dalam kelas tetapi di luar metode kelas apa pun. Variabel kelas tidak digunakan sesering variabel instance.

Anggota data (data member) - Variabel kelas atau variabel instan yang menyimpan data yang terkait dengan kelas dan objeknya.

Function overloading - Penugasan lebih dari satu perilaku ke fungsi tertentu. Operasi yang dilakukan bervariasi berdasarkan jenis objek atau argumen yang terlibat.

Instance variable - Variabel yang didefinisikan di dalam suatu metode dan hanya dimiliki oleh instance kelas saat ini.

Inheritance - Pemindahan karakteristik suatu kelas ke kelas lain yang diturunkan darinya.

Instance - Objek individual dari kelas tertentu. Objek objek milik Lingkaran kelas, misalnya, adalah turunan dari Lingkaran kelas.

Instantiation - Penciptaan instance kelas.

Method - Jenis fungsi khusus yang didefinisikan dalam definisi kelas.

Objek (object)- Contoh unik dari struktur data yang ditentukan oleh kelasnya. Objek terdiri dari anggota data (variabel kelas dan variabel contoh) dan metode.

Operator overloading - Penugasan lebih dari satu fungsi ke sebuah part

Membuat kelas:

Untuk membuat kelas dalam python menggunakan keyword sebagai berikut :

```
class ClassName:
    'Optional class documentation string'
    class_suite
```

Class_suite semua komponen pernyataan yang mendefinisikan anggota kelas, data, attribute dan fungsi.

Contoh :

Membuat kelas mahasiswa :

- Objek : mahasiswa-mahasiswa yang terdaftar
- Variabel kelas : jumlah mahasiswa
- Instance variable : Nama, Prodi, NIM
- Method : fungsi tambah, fungsi tampil


```

class mahasiswa:
    jumlah=0
    def __init__(mhs,nama,prodi,NIM):
        mhs.prodi=prodi
        mhs.nama=nama
        mhs.nim=NIM
        mahasiswa.jumlah +=1

    def tampil_jumlah(mhs):
        print("jumlah mahasiswa %d : ", mahasiswa.jumlah)

    def tampil_mahasiswa(mhs):
        print("nama : ", mhs.nama, ",NIM : ", mhs.nim, "prodi : ", mhs.prodi)

```

Nama kelas : mahasiswa

Variabel jumlah merupakan variabel kelas yang nilainya dapat di akses/share ke semua instance pada kelas tersebut. Cara aksesnya :mahasiswa.jumlah

Fungsi __init__() merupakan fungsi khusus (fungsi inisiasi) ketika membuat instance baru pada kelas tersebut.

Selanjutnya dapat di deklarasikan fungsi/method yang lain mendefinisikan fungsi biasanya dengan argument pertama yaitu (dalam contoh) : mhs

Menambah instance

Menambahkan instance dengan memanggil nama kelas dan memberikan argument yang didefinisikan dalam fungsi __init__

```
class mahasiswa:
    jumlah=0
    def __init__(mhs,nama,prodi,NIM):
        mhs.prodi=prodi
        mhs.nama=nama
        mhs.nim=NIM
        mahasiswa.jumlah +=1

    def tampil_jumlah(mhs):
        print("jumlah mahasiswa %d : ", mahasiswa.jumlah)

    def tampil_mahasiswa(mhs):
        print("nama : ", mhs.nama, ",NIM : ", mhs.nim, "prodi : ", mhs.prodi)
```

```
# membuat objek pertama pada kelas mahasiswa
mhs1=mahasiswa("iwan","SI",150016001)
```

```
# membuat objek kedua pada kelas mahasiswa
mhs2=mahasiswa("riyadi","math",160016002)
```

```
# membuat objek ketiga pada kelas mahasiswa
mhs3=mahasiswa("yanto","fisika",140016003)
```

```
# membuat objek keempat pada kelas mahasiswa
mhs4=mahasiswa("tri","biologi",130016004)
```

Mengakses Attributes

Attribute dapat di akses dengan operator dot (.) pada objek Sedangkan variabel klas dapat di akses dengan nama kelasnya

```
mhs1.tampil_mahasiswa()
```

```
nama : iwan ,NIM : 150016001 prodi : SI
```

```
mhs2.tampil_mahasiswa()
```

```
nama : riyadi ,NIM : 160016002 prodi : math
```

```
mhs3.tampil_mahasiswa()
```

```
nama : yanto ,NIM : 140016003 prodi : fisika
```

```
mhs4.tampil_mahasiswa()
```

```
nama : tri ,NIM : 130016004 prodi : biologi
```

```
mahasiswa.jumlah
```

```
4
```

Fungsi untuk mengakses atribut

```
hasattr(mhs1, "nama")    #menghasilkan True jika ada "nama"
```

True

```
getattr(mhs1, "nama")    # menghasilkan nama pada atribut mhs1
```

'iwan'

```
setattr(mhs1, "nama", "ione")
```

```
getattr(mhs1, "nama")    # menghasilkan nama pada atribut mhs1
```

'ione'

- **setattr(obj,name,value)** – Untuk set attribute
- **delattr(obj, name)** – untuk menghapus atribut

- **getattr(obj, name[, default])**
– Untuk akses attribute objek
- **hasattr(obj,name)**
– Untuk cek terdapat attribute atau tidak

```
delattr(mhs1, 'nama')    # hapus atribut nama
```

```
mhs1.tampil_mahasiswa()
```

```
-----
AttributeError                                Traceback (most recent call last)
<ipython-input-59-fac1f8ebfa20> in <module>
----> 1 mhs1.tampil_mahasiswa()
```

```
<ipython-input-38-c38225c70e43> in tampil_mahasiswa(mhs)
```

```
11
12     def tampil_mahasiswa(mhs):
--> 13         print("nama : ", mhs.nama, ",NIM : ", mhs.nim, "prodi : ", mhs.prodi)
```

```
AttributeError: 'mahasiswa' object has no attribute 'nama'
```

```
setattr(mhs1, "nama", "iwan2")
```

```
mhs1.tampil_mahasiswa()
```

```
nama : iwan2 ,NIM : 150016001 prodi : SI
```

Kode keseluruhan :

```

class mahasiswa:
    jumlah=0
    def __init__(mhs,nama,prodi,NIM):
        mhs.prodi=prodi
        mhs.nama=nama
        mhs.nim=NIM
        mahasiswa.jumlah +=1

    def tampil_jumlah(mhs):
        print("jumlah mahasiswa %d : ", mahasiswa.jumlah)

    def tampil_mahasiswa(mhs):
        print("nama : ", mhs.nama, ",NIM : ", mhs.nim, "prodi : ", mhs.prodi)

# Menambah instance
# membuat objek pertama pada kelas mahasiswa
mhs1=mahasiswa("iwan","SI",150016001)
# membuat objek kedua pada kelas mahasiswa
mhs2=mahasiswa("riyadi","math",160016002)
# membuat objek ketiga pada kelas mahasiswa
mhs3=mahasiswa("yanto","fisika",140016003)
# membuat objek keempat pada kelas mahasiswa
mhs4=mahasiswa("tri","biologi",130016004)
#Mengakses attribute
mhs1.tampil_mahasiswa()
mhs2.tampil_mahasiswa()
mhs3.tampil_mahasiswa()
mhs4.tampil_mahasiswa()
#Menampilkan jumlah instance
mahasiswa.jumlah

nama : iwan ,NIM : 150016001 prodi : SI
nama : riyadi ,NIM : 160016002 prodi : math
nama : yanto ,NIM : 140016003 prodi : fisika
nama : tri ,NIM : 130016004 prodi : biologi

```

: 4

Tugas/Latihan : Buatlah sebuah kelas untuk

- Hewan
- Kendaraan
- Alat elektronik

Silahkan pilih salah satu , tentukan terlebih dahulu

Objek

Variabel kelas

Instance variable

Method

Pertemuan 10. File I/O

Tujuan :

1. Mahasiswa memahami pembuatan File pada python
2. Mahasiswa mampu membuat File pada python

Dasar Pengetahuan :

Printing to the Screen Cara paling sederhana untuk menghasilkan output adalah dengan menggunakan pernyataan cetak. Fungsi ini mengubah ekspresi masukkan ke bentuk string dan menulis hasilnya ke standar luaran sebagai berikut :

```
print('cetak sederhana')
```

cetak sederhana

Prosedur ini memberikan luaran standar dalam layar.

Pada Python 3, fungsi input () membaca data dari keyboard sebagai string, terlepas dari apakah itu dilampirkan dengan tanda kutip (' ' atau " ") atau tidak.

```
x=input("masukan sembarang nilai : ")
```

masukan sembarang nilai : 10

x

'10'

```
x=input("masukan sembarang nilai : ")
```

masukan sembarang nilai : '10'

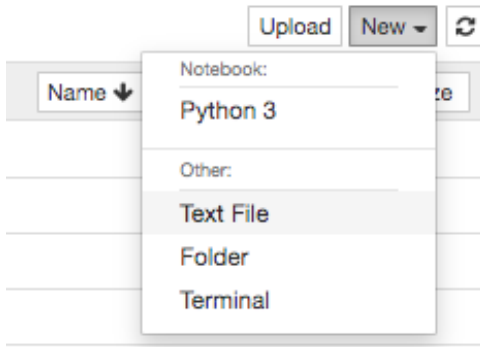
x

"'10'"

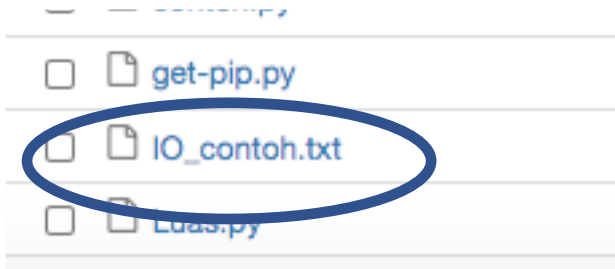
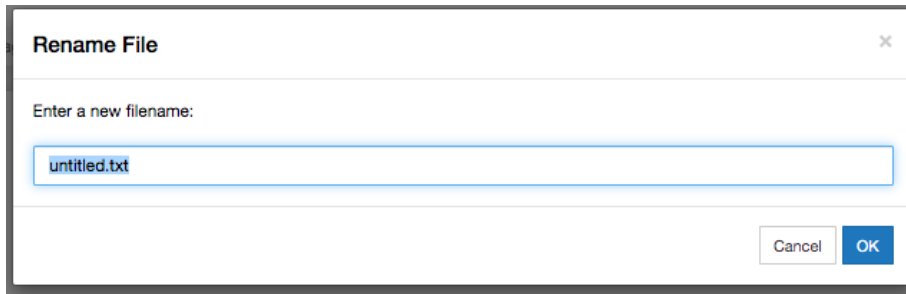
Hal ini untuk membaca dan menulis dengan input dan output standar. Sekarang, bagaimana menggunakan file data aktual. Python menyediakan fungsi dan metode dasar yang diperlukan

untuk memanipulasi file secara default. sebagian besar manipulasi file menggunakan objek file.

Membuat file txt dengan jupyter



Rename nama file : missal : IO_contoh



Python menyediakan fungsi dan metode dasar yang diperlukan untuk memanipulasi file secara default dan manipulasi file menggunakan objek file.

Open File

Sebelum membaca atau menulis file, File harus dibuka terlebih dahulu menggunakan fungsi `open ()` built-in Python.

Syntax

`file object = open(file_name [, access_mode][, buffering])`

`file_name` : nama file yang akan di akses

`access_mode` : menentukan mode di mana file harus dibuka, mis., baca, tulis, tambahkan, dll.

Buffering :

- Jika nilai buffering diatur ke 0, tidak ada buffering yang terjadi.
- Jika nilai buffering adalah 1, line buffering dilakukan saat mengakses file. Jika Anda menentukan nilai buffering sebagai bilangan bulat lebih besar dari 1, maka tindakan buffering dilakukan dengan ukuran buffer yang ditunjukkan.
- Jika negatif, ukuran buffer adalah default sistem (perilaku default).

Close File

Menutup objek file, setelah itu tidak ada lagi tulisan yang dapat dilakukan.

Syntax
`fileObject.close();`

Berikut contoh untuk open dan close file :

```
contoh_file = open("IO_contoh.txt", "wb", 0)
```

```
print("nama file : ", contoh_file.name)
print("tutup atau tidak ", contoh_file.closed)
print("mode buka : ", contoh_file.mode)
contoh_file.close()
```

```
nama file :  IO_contoh.txt
tutup atau tidak  False
mode buka :  wb
```

Menulis file

Untuk menulis file menggunakan fungsi **write ()**, fungsi ini tidak menambahkan karakter baris baru ('\n') ke akhir string

Syntax

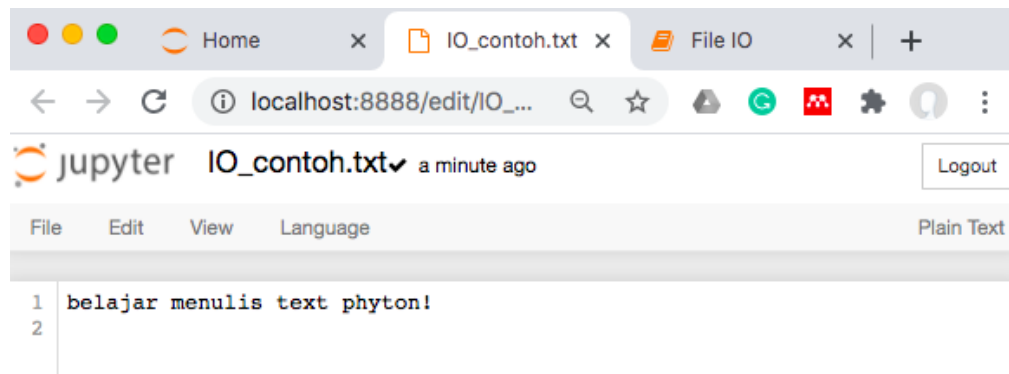
```
fileObject.write(string);
```

Membuka dan menutup Files

Berikut sebagai contoh:

```
contoh_file = open("IO_contoh.txt", "w")
contoh_file.write( "belajar menulis text phyton!\n")
contoh_file.close()
```

Jika dibuka file txt nya akan Nampak seperti berikut :



Membaca Files

Untuk membaca file menggunakan fungsi **read ()** untuk membaca string dari file yang terbuka.

Syntax

```
fileObject.read([count]);
```

Parameter **count** adalah jumlah byte yang dibaca dari file yang dibuka. fungsi ini mulai membaca dari awal file dan jika **count** tidak diberikan , maka file akan dibaca sebanyak mungkin, mungkin sampai akhir file.

Berikut sebagai contoh membaca file :


```

contoh_file = open("IO_contoh.txt", "r+")
str = contoh_file.read()
print(str)
posisi = contoh_file.tell()
print ("posisi : ", posisi)

```

belajar menulis text phyton!

posisi : 29

Fungsi **tell ()** memberi tahu posisi saat ini di dalam file

```

contoh_file = open("IO_contoh.txt", "r+")
str = contoh_file.read(15)
print(str)
posisi = contoh_file.tell()
print ("posisi : ", posisi)

```

belajar menulis

posisi : 15

Berikut percobaan membuat File :

Coba ketikan :

```

contoh_file = open("IO_contoh.txt", "w")
contoh_file.write( "belajar menulis text phyton!\n")
contoh_file.write( "Menyenangkan sekali !\n")
contoh_file.write( "Pusing Ga ya !\n")
contoh_file.close()

```

Kemudia refresh IO_contoh.txt

Percobaan 1 membaca file :

```

contoh_file = open("IO_contoh.txt", "r+")
str = contoh_file.read(7)
print(str)
posisi = contoh_file.tell()
print ("posisi : ", posisi)
position = contoh_file.seek(0,1)
str = contoh_file.read(7)
print(str)

```

Percobaan 2 membaca file:

```

contoh_file = open("IO_contoh.txt", "r+")
str = contoh_file.read()
print(str)
posisi = contoh_file.tell()
print ("posisi : ", posisi)
position = contoh_file.seek(0,0)
str = contoh_file.read(7)
print(str)

```

Tugas/Latihan :

1. tuliskan beberapa daftar berbagai mode membuka file
Cobalah dan tuliskan perbedaannya.
2. Cobalah beberapa fungsi untuk memproses dan memanipulasi file, tunjukkan perubahannya.

Fungsi untuk memanipulasi file

```

file.close()
file.flush()
file.fileno()
file.isatty()
next(file)
file.read([size])
file.readline([size])
file.readlines([sizehint])
file.seek(offset[, whence])
file.tell()
file.truncate([size])
file.write(str)
file.writelines(sequence)

```

Fungsi Untuk memproses file	
os.access(path, mode) os.chdir(path) os.chflags(path, flags) os.chmod(path, mode) os.chown(path, uid, gid) os.chroot(path) os.closerange(fd_low, fd_high) os.dup(fd) os.dup2(fd, fd2) os.fchdir(fd) os.fchmod(fd, mode) os.fchown(fd, uid, gid) os.fdatasync(fd) os.fdopen(fd[, mode[, bufsize]]) os.fpathconf(fd, name) os.fstat(fd) os.fstatvfs(fd) os.fsync(fd) os.ftruncate(fd, length) os.getcwd() os.getcwdu() os.isatty(fd) os.lchflags(path, flags) os.lchmod(path, mode) os.lchown(path, uid, gid) os.link(src, dst) os.listdir(path) os.lseek(fd, pos, how) os.lstat(path) os.major(device) os.makedev(major, minor) os.makedirs(path[, mode])	os.minor(device) os.mkdir(path[, mode]) os.mkfifo(path[, mode]) os.mknod(filename[, mode = 0600, device]) os.open(file, flags[, mode]) os.openpty() os.pathconf(path, name) os.pipe() os.popen(command[, mode[, bufsize]]) os.read(fd, n) os.readlink(path) os.remove(path) os.removedirs(path) os.rename(src, dst) os.renames(old, new) os.rmdir(path) os.stat(path) os.stat_float_times([newvalue]) os.statvfs(path) os.symlink(src, dst) os.tcgetpgrp(fd) os.tcsetpgrp(fd, pg) os.tempnam([dir[, prefix]]) os.tmpfile() os.tmpnam() os.ttyname(fd) os.utime(path, times) os.walk(top[, topdown = True[, onerror = None[, followlinks = False]]) os.write(fd, str)

DaftarPustaka

Pranata, A. (2000), Algoritma dan Pemrograman, J & J Learning Yogyakarta.

Abdul Kadir, (2000) Konsep Dasar Pemrograman Bahasa C, Andi Offset

Cormen T.H, Leiserson C.E (2001), Introduction to Algoritihms, MIT Press.

Budi Raharjo (2019), Mudah belajar phyton, Informatika