


<p><b>Nama:</b> Aini Rihhadatul Aisy</p> <p><b>NIM:</b> 064102400024</p>	 <p><b>Praktikum Algoritma &amp; Pemrograman</b></p>	<p><b>MODUL 3</b></p> <p><b>Nama Dosen:</b> Binti solihah, S.T, M.KOM</p>
<p><b>Hari/Tanggal:</b> Hari, Tanggal Bulan 2022</p>		<p><b>Nama Asisten Labratorium:</b></p> <ol style="list-style-type: none"> <li>1. Yustianas Rombon - 064002300015</li> <li>2. Vira Aditya Kurniawan - 065002300012</li> </ol>

## Struktur Kendali (Control Structure)

### 1. Teori Singkat

#### Ekspresi Boolean

Ekspresi Boolean merupakan ekspresi yang mengembalikan nilai True atau False, menggunakan operator relasional/operator perbandingan, dan juga operator logika. Selain itu Ekspresi Boolean juga dapat menggunakan operator keanggotaan (*membership operator*) dan juga operator identitas dalam beberapa kasus.

#### Operator Perbandingan

Operator Perbandingan adalah operator yang melakukan perbandingan antara dua buah nilai. Operator ini juga dikenal dengan operator relasional dan sering digunakan untuk membuat sebuah logika atau kondisi. Berikut ini adalah daftar Operator Aritmatika dalam Python:

Operator	Simbol
Lebih Besar	>
Lebih Kecil	<
Sama Dengan	==
Tidak Sama Dengan	!=

Lebih Besar Sama Dengan	$\geq$
Lebih Kecil Sama Dengan	$\leq$

## Operator Logika

Operator Logika merupakan sebuah operator yang digunakan untuk membuat logika dalam program yang kita buat. Operator logika juga sering disebut juga sebagai Operator Aljabar Boolean, biasanya operator logika ini digunakan untuk membuat operasi percabangan pada program. Operator Logika diantaranya seperti logika AND, OR, dan NOT.

Operator logika terdiri dari:

Operator	Simbol
Logika AND	and
Logika OR	or
Logika Negasi/Kebalikan	not

## Konstruksi Percabangan & Blok Program

Konstruksi Percabangan adalah sebuah program yang ketika dijalankan akan menimbulkan percabangan kedalam sub cabangnya yang berisi sebuah blok program sesuai dengan kondisi dan logika yang diminta. Umumnya konstruksi percabangan dalam Bahasa pemrograman Python sendiri dapat dibuat dengan memanggil keyword *if/elif/else*. Berikut tabelnya

Keterangan	Keyword
Terdapat 1 pilihan keputusan	if
Terdapat 2 pilihan keputusan	if/else
Terdapat lebih dari 2 pilihan keputusan	if/elif/else

Blok program berisi sekumpulan ekspresi dan statement untuk dikerjakan oleh komputer. Dalam Bahasa pemrograman Python blok program sendiri dapat diidentifikasi dengan tanda *colon* (":") setelah pendeklarasian konstruksi *if/elif/else*, *for*, *while* ataupun ketika melakukan definisi fungsi.

Blok program yang terdapat pada kondisi *if* sendiri akan dijalankan jika kondisi yang diminta bernilai *true*.

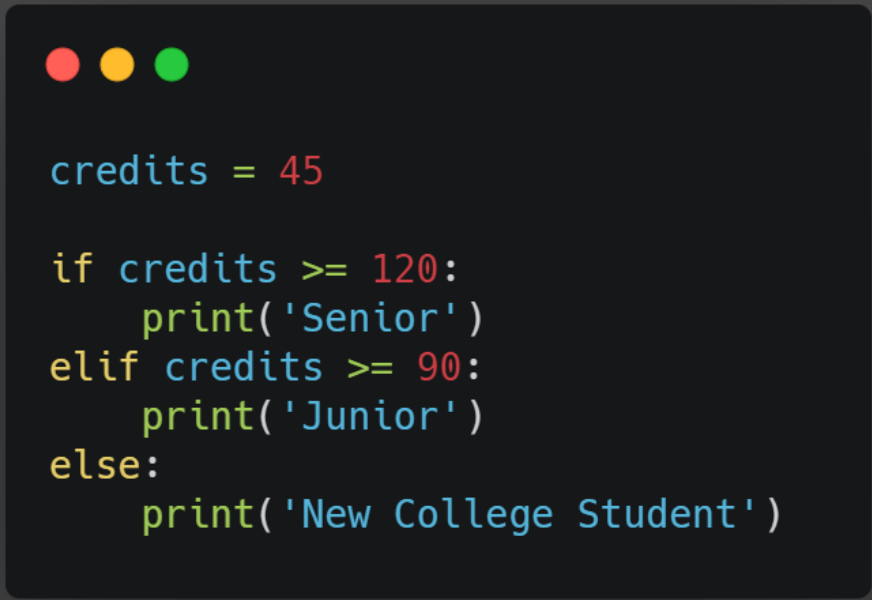


Blok program yang terdapat pada kondisi kondisi *elif* sendiri yang merupakan kepanjangan dari *else if* yang berarti jika tidak sesuai dengan kondisi sebelumnya maka akan disesuaikan dengan kondisi lainnya yang dapat bernilai *true*.

Blok program yang terdapat pada kondisi *else* akan dijalankan ketika nilai dari kondisi sebelumnya yaitu *if/elif* bernilai *false*.

Berikut ini adalah contoh sederhana program konstruksi percabangan yang menggunakan operator perbandingan:

#### Source Code

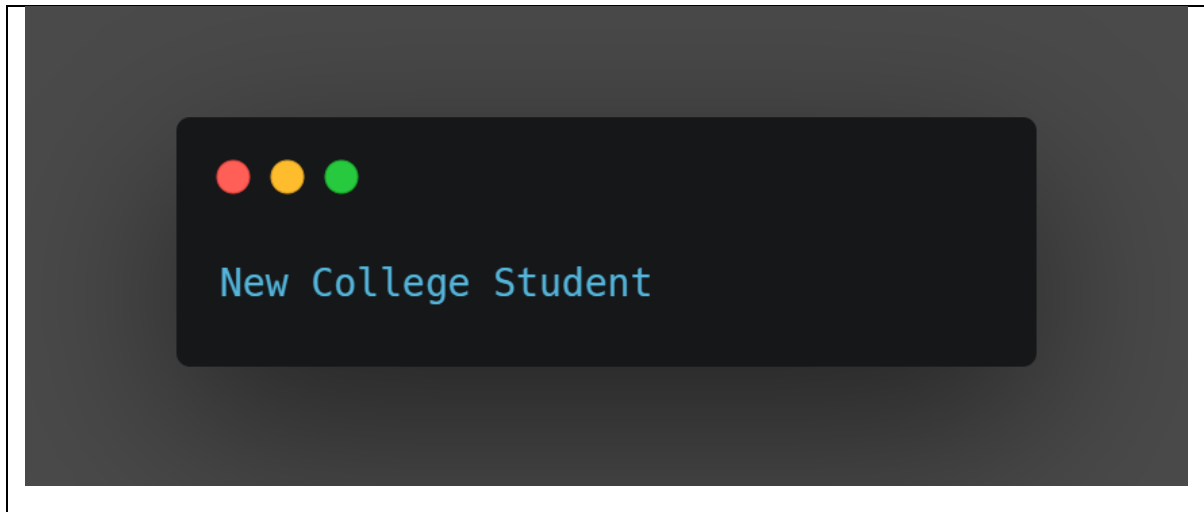


```
credits = 45

if credits >= 120:
    print('Senior')
elif credits >= 90:
    print('Junior')
else:
    print('New College Student')
```

#### Output





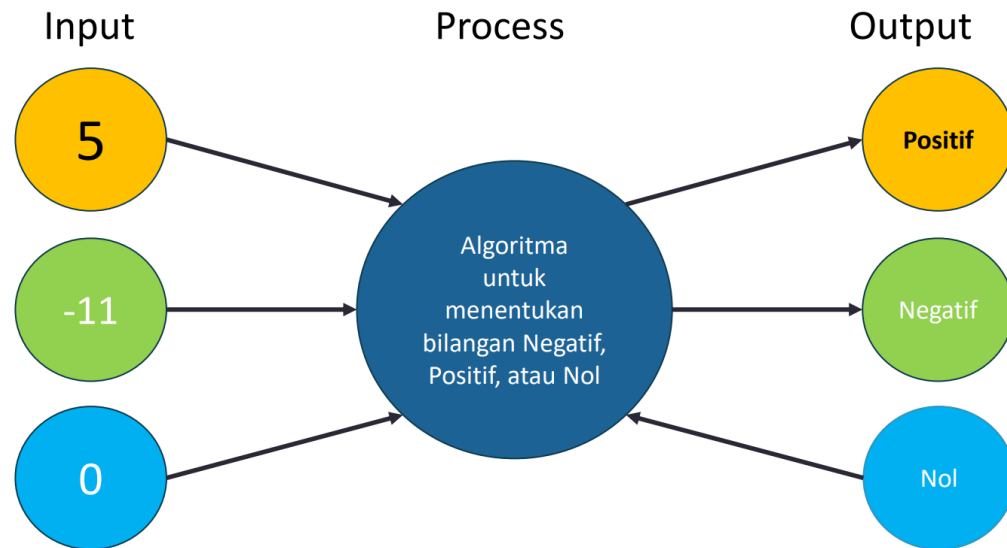
## IPO (Input Process Output)

Konsep Dasar Input, Process, dan Output (IPO)

- Konsep input, process, dan output adalah prinsip dasar dalam pemrograman dan pengembangan algoritma.
- Setiap algoritma melibatkan tiga tahap utama: mengambil data masukan (input), melakukan operasi atau pengolahan data (process), dan menghasilkan hasil akhir (output).
- Konsep ini menggambarkan bagaimana algoritma beroperasi untuk memproses informasi.



## Gambaran IPO (Menentukan Bilangan)



### Pseudocode

Pseudocode adalah suatu bentuk deskripsi informal yang mirip dengan bahasa manusia dan digunakan untuk menggambarkan algoritma atau proses secara naratif. Ini tidak terikat pada bahasa pemrograman tertentu, tetapi memberikan panduan tentang langkah-langkah yang harus diambil dalam suatu algoritma dengan bahasa yang lebih mudah dimengerti.



## Contoh PseudoCode

### Inisiasi Variabel:

```
N      = 0
total  = 0.0
```

### Pengulangan:

```
UNTUK i DARI 1 SAMPAI 10 LANGKAH 2
  CETAK i
END UNTUK
```

### Pengkondisional (Conditional):

```
JIKA nilai > 10
  CETAK "Nilai lebih dari 10"
SELAINNYA JIKA nilai = 10
  CETAK "Nilai sama dengan 10"
SELAINNYA
  CETAK "Nilai kurang dari 10"
AKHIR JIKA
```

### Fungsi atau Prosedur:

```
FUNGSI tambah(a, b)
  KEMBALIKAN a + b
AKHIR FUNGSI
```

### Contoh Lengkap:

```
DEKLARASI variabel n, bilangan, total, rata_rata FLOAT
MINTA "Masukkan jumlah bilangan: " SIMPAN
total = 0.0

UNTUK i DARI 1 SAMPAI n
  MINTA "Masukkan bilangan ke-" + i + ": " SIMPAN bilangan
  total = total + bilangan
END UNTUK

rata_rata = total / n
CETAK "Rata-rata adalah: " + rata_rata
```

## 2. Alat dan Bahan

Hardware : Laptop/PC

Software : Spyder (Anaconda Python)

## 3. Elemen Kompetensi

### a. Latihan pertama

Sebuah segitiga dibangun dari tiga garis lurus. Berdasarkan panjang dari sisi-sisinya, segitiga dapat dibedakan menjadi tiga jenis. Ada segitiga sama sisi, segitiga sama kaki, segitiga siku-siku, atau segitiga sembarang. Buatlah sebuah program yang menerima tiga bilangan yang merupakan panjang dari sisi-sisi sebuah segitiga. Berdasarkan panjang yang diberikan, program anda akan mencetak jenis segitiganya (sama sisi, sama kaki, atau sembarang). Hati-hati: Tidak semua kombinasi tiga bilangan dapat membentuk segitiga. Contoh: 1, 2, 3 tidak mungkin membentuk segitiga.

Pseudocode



```
Masukkan Panjang sisi A ke dalam Variabel sisiA
Masukkan Panjang sisi B ke dalam Variabel sisiB
Masukkan Panjang sisi C ke dalam Variabel sisiC
#Untuk Mengecheck apakah bisa membentuk segitiga atau tidak bisa membentuk
segitiga
Jika( sisiA + sisiB > sisiC) and (sisiB + sisiC > sisiA) and (sisiA + sisiC > sisiB)
Tampilkan "Bisa Membentuk Segitga"
Selain dari itu berarti "Tidak Bisa Membentuk Segitiga)
#Untuk Mengecheck Sebuah Segitiga
Jika sisiA = sisiB = sisiC Tampilkan "segitiga Sama sisi"
Selain itu Jika sisiA = sisiB atau sisiA = sisiC atau sisiB = sisiC
Tampilkan "segitiga sama kaki"
Selain dari itu Tampilkan segitiga Sembarang
#Jika tidak sesuai dengan yang diatas
Jika tidak memenuhi rumus if yang ada di atas, maka "Tidak Bisa Membentuk
Segitiga"
```

#### Input Process Output

Input =

- sisiA
- sisiB
- sisiC

Proses =

- Cek apakah ketiga sisi dapat membentuk segitiga:  
sisiA + sisiB > sisiC  
sisiB + sisiC > sisiA  
sisiA + sisiC > sisiB
- Jika bisa, Cek Jenis Segitiganya  
Segitiga Sama Sisi (Jika semua sisi sama)  
Segitiga Siku Siku (Jika dua sisi sama)  
Segitiga Sembarang (Jika Tidak ada sisi yang sama)

Output =

- Angka yang dimasukkan kedalam masing-masing variabel
- Hasil apakah bisa membentuk segitiga atau tidak



### Source Code

```
❏ sisiA = float(input("Masukkan panjang sisi A: "))
sisiB = float(input("Masukkan panjang sisi B: "))
sisiC = float(input("Masukkan panjang sisi C: "))

if (sisiA + sisiB > sisiC) and (sisiB + sisiC > sisiA) and (sisiA + sisiC > sisiB):
    print("Bisa membentuk segitiga")
    if sisiA == sisiB == sisiC:2
        print("Segitiga sama sisi")
    elif sisiA == sisiB or sisiA == sisiC or sisiB == sisiC:
        print("Segitiga sama kaki")
    else:
        print("Segitiga sembarang")
else:
    print("Tidak bisa membentuk segitiga")
```





Output

```
⇒ Masukkan panjang sisi A: 1  
Masukkan panjang sisi B: 2  
Masukkan panjang sisi C: 3  
Tidak bisa membentuk segitiga
```

b. Latihan Kedua

Buatlah program untuk mencari Akar Persamaan Kuadrat dan Determinan

Pseudocode



```
FUNGSI hitung_determinan(a, b, c):  
    RETURN  $b^2 - 4 * a * c$   
  
FUNGSI akar_persamaan_kuadrat(a, b, c):  
    D = hitung_determinan(a, b, c)  
  
    JIKA D > 0:  
         $x1 = (-b + \text{sqrt}(D)) / (2 * a)$   
         $x2 = (-b - \text{sqrt}(D)) / (2 * a)$   
        RETURN (x1, x2)  
  
    ELASE IF D == 0:  
         $x1 = -b / (2 * a)$   
        RETURN (x1)  
  
    ELASE:  
         $x1 = (-b + \text{sqrt}(D)) / (2 * a)$   
         $x2 = (-b - \text{sqrt}(D)) / (2 * a)$   
        RETURN (x1, x2)  
  
INPUT a DARI pengguna  
INPUT b DARI pengguna  
INPUT c DARI pengguna  
  
akar = akar_persamaan_kuadrat(a, b, c)  
determinan = hitung_determinan(a, b, c)  
  
TAMPILKAN "Determinan: ", determinan  
  
JIKA jumlah elemen dari akar == 2:  
    TAMPILKAN "Akar-akar persamaan kuadrat: x1 = ", akar[0], ", x2 = ", akar[1]  
  
    ELASE IF jumlah elemen dari akar == 1:  
        TAMPILKAN "Akar persamaan kuadrat: x = ", akar[0]  
  
    ELASE:  
        TAMPILKAN "Akar kompleks: x1 = ", akar[0], ", x2 = ", akar[1]
```

Input Output Process



Input :

- Koefisien a (float)
- Koefisien b (float)
- Koefisien c (float)

Proses :

- Hitung Determinan  $D = b^2 - 4ac$
- Tentukan jenis akar berdasarkan nilai determinan:  
Jika  $D > 0$ : Hitung dua akar real.  
Jika  $D = 0$ : Hitung satu akar real.  
Jika  $D < 0$ : Hitung dua akar kompleks.

Output :

- Hasil Determinan
- Hasil dari Akar Persamaan Kuadrat

### Source Code

```
import cmath
def hitung_determinan(a, b, c):
    return b**2 - 4*a*c
def akar_persamaan_kuadrat(a, b, c):
    D = hitung_determinan(a, b, c)
    if D > 0:
        x1 = (-b + cmath.sqrt(D)) / (2*a)
        x2 = (-b - cmath.sqrt(D)) / (2*a)
        return (x1, x2)
    elif D == 0:
        x1 = -b / (2*a)
        return (x1,)
    else:
        x1 = (-b + cmath.sqrt(D)) / (2*a)
        x2 = (-b - cmath.sqrt(D)) / (2*a)
        return (x1, x2)
a = float(input("Masukkan koefisien a: "))
b = float(input("Masukkan koefisien b: "))
c = float(input("Masukkan koefisien c: "))
akar = akar_persamaan_kuadrat(a, b, c)
determinan = hitung_determinan(a, b, c)
print(f"Determinan: {determinan}")
print(f"Akar persamaan kuadrat: x1 = {akar[0]}, x2 = {akar[1]}")
if len(akar) == 2:
    print(f"Akar-akar persamaan kuadrat: x1 = {akar[0]}, x2 = {akar[1]}")
elif len(akar) == 1:
    print(f"Akar persamaan kuadrat: x = {akar[0]}")
else:
    print(f"Akar kompleks: x1 = {akar[0]}, x2 = {akar[1]}")
```



### Output

```
➔ Masukkan koefisien a: 1  
    Masukkan koefisien b: 2  
    Masukkan koefisien c: 3  
    Determinan: -8.0  
    Akar-akar persamaan kuadrat: x1 = (-1+1.4142135623730951j), x2 = (-1-1.4142135623730951j)
```

#### 4. File Praktikum

Github Repository:

#### 5. Soal Latihan

Soal:

1. Dalam sebuah kasus program, terdapat sebuah kondisi percabangan *if/else*. Jika program yang dijalankan pada kondisi *if* tidak sesuai dengan kondisinya, maka itu akan menghasilkan status nilai *false* pada percabangan *if* tersebut, dan program tersebut akan masuk ke kondisi *else*, apakah status yang diberikan kondisi *else* tersebut? Jelaskan dan berikan alasannya serta deskripsikan kelanjutan dari program tersebut!



2. Deskripsikan serta narasikan jalannya alur source code program yang sebelumnya telah kalian buat pada Elemen Kompetensi Latihan Kedua!

Jawaban:

1. Jika kondisi dalam percabangan if tidak terpenuhi, maka status dari kondisi else adalah *true*. Artinya, program akan mengeksekusi blok kode yang ada di dalam else.

Untuk kelanjutan dari program ini yaitu Setelah blok else dieksekusi, program akan melanjutkan ke baris berikutnya setelah blok if/else. Program akan terus berjalan, menjalankan kode yang ada setelah kedua blok tersebut, tanpa mengulangi atau kembali ke blok if/else.

2. FUNGSI hitung\_determinan(a, b, c):

RETURN  $b^2 - 4 * a * c$

FUNGSI akar\_persamaan\_kuadrat(a, b, c):

D = hitung\_determinan(a, b, c)

JIKA  $D > 0$ :

$x1 = (-b + \text{sqrt}(D)) / (2 * a)$

$x2 = (-b - \text{sqrt}(D)) / (2 * a)$

RETURN (x1, x2)

ELASE IF  $D == 0$ :

$x1 = -b / (2 * a)$

RETURN (x1)

ELASE:

$x1 = (-b + \text{sqrt}(D)) / (2 * a)$

$x2 = (-b - \text{sqrt}(D)) / (2 * a)$

RETURN (x1, x2)

INPUT a DARI pengguna

INPUT b DARI pengguna

INPUT c DARI pengguna

akar = akar\_persamaan\_kuadrat(a, b, c)

determinan = hitung\_determinan(a, b, c)

TAMPILKAN "Determinan: ", determinan

JIKA jumlah elemen dari akar == 2:

TAMPILKAN "Akar-akar persamaan kuadrat: x1 = ", akar[0], ", x2 = ", akar[1]

ELASE IF jumlah elemen dari akar == 1:

TAMPILKAN "Akar persamaan kuadrat: x = ", akar[0]

ELASE:

TAMPILKAN "Akar kompleks: x1 = ", akar[0], ", x2 = ", akar[1]

## 6. Kesimpulan

- a. Dalam pengerjaan program dengan bahasa pemrograman Python, kita harus benar-benar teliti dalam menginputkan suatu fungsi untuk menampilkan suatu keluaran pada layar dengan sesuai.



- b. Kita dapat mengetahui cara untuk memakai keadaan if else, if elif, dan if elif elif dalam praktikum kali ini, dan menurut saya cukup sulit untuk menggunakan keadaan if else, if elif, dan if elif elif ini karena harus butuh ketelitian dalam mengerjakannya.

## 7. Cek List (✓)

No	Elemen Kompetensi	Penyelesaian	
		Selesai	Tidak Selesai
1.	Latihan Pertama	✓	
2.	Latihan Kedua	✓	

## 8. Formulir Umpan Balik

No	Elemen Kompetensi	Waktu Pengerjaan	Kriteria
1.	Latihan Pertama	60 Menit	Menarik
2.	Latihan Kedua	120 Menit	Menarik

Keterangan:

1. Menarik
2. Baik
3. Cukup
4. Kurang