

<b>Nama:</b> <b>(MUHAMMAD RIZKI)</b>  <b>NIM:</b> <b>(064102400020)</b>	 <b>Praktikum Algoritma &amp; Pemrograman</b>	<b>MODUL 3</b>  <b>Nama Dosen:</b> <b>Binti solihah, S.T, M.KOM</b>
<b>Hari/Tanggal:</b> <b>Hari, Tanggal Bulan 2022</b>		<b>Nama Asisten Laboratorium:</b> 1. <b>Yustianas Rombon - 064002300015</b> 2. <b>Vira Aditya Kurniawan - 065002300012</b>

### 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	>=
Lebih Kecil Sama Dengan	<=



## 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 *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

```
New College Student
```

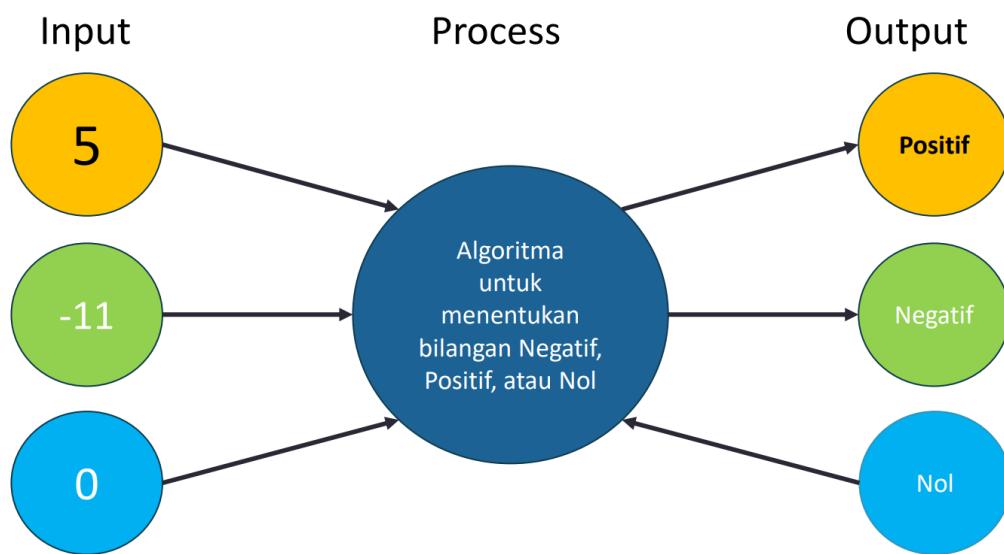


## IPO (Input Process Output)

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



START

Input sisi segitiga

Tampilkan "Masukkan panjang sisi pertama (a): "

INPUT a Simpan input ke variabel a

Tampilkan "Masukkan panjang sisi kedua (b): "

INPUT b Simpan input ke variabel b

Tampilkan "Masukkan panjang sisi ketiga (c): "

INPUT c Simpan input ke variabel c

Cek apakah sisi dapat membentuk segitiga

IF  $a + b > c$  AND  $a + c > b$  AND  $b + c > a$  THEN

Cek segitiga sama sisi

IF  $a == b$  AND  $b == c$  THEN

Tampilkan "Segitiga Sama Sisi"

Cek segitiga sama kaki

ELSE IF  $a == b$  OR  $a == c$  OR  $b == c$  THEN

Tampilkan "Segitiga Sama Kaki"

Cek segitiga siku-siku

ELSE IF  $a^2 + b^2 == c^2$  OR  $a^2 + c^2 == b^2$  OR  $b^2 + c^2 == a^2$  THEN

Tampilkan "Segitiga Siku-Siku"

Segitiga sembarang jika tidak ada kondisi lain

ELSE

Tampilkan "Segitiga Sembarang"

ELSE

Jika sisi tidak memenuhi syarat untuk membentuk segitiga

Tampilkan "Bukan Segitiga"

END

Input Process Output



**Input:**

- Panjang sisi pertama segitiga (a): Bilangan pecahan (float).
- Panjang sisi kedua segitiga (b): Bilangan pecahan (float).
- Panjang sisi ketiga segitiga (c): Bilangan pecahan (float).

**Proses:**

1. Cek apakah jumlah dua sisi lebih besar dari sisi yang tersisa (validasi segitiga).
2. Jika valid, identifikasi jenis segitiga berdasarkan kondisinya:
  - Segitiga Sama Sisi:  $a == b == c$ .
  - Segitiga Sama Kaki:  $a == b$ ,  $a == c$ , atau  $b == c$ .
  - Segitiga Siku-Siku:  $a^2 + b^2 == c^2$ , atau variasi lainnya.
  - Segitiga Sembarang: Jika tidak ada kondisi lain yang terpenuhi.
3. Jika tidak valid sebagai segitiga, tampilkan bahwa itu **bukan segitiga**.

**Output:**

- "Segitiga Sama Sisi" jika semua sisi sama panjang.
- "Segitiga Sama Kaki" jika dua sisi segitiga sama panjang.
- "Segitiga Siku-Siku" jika memenuhi kondisi Pythagoras.
- "Segitiga Sembarang" jika tidak ada kondisi lainnya yang terpenuhi.
- "Bukan Segitiga" jika sisi yang dimasukkan tidak memenuhi syarat segitiga.

[Source Code](#)



```
a = float(input("Masukkan panjang sisi pertama: "))
b = float(input("Masukkan panjang sisi kedua: "))
c = float(input("Masukkan panjang sisi ketiga: "))
#untuk cek segitiga
if a + b > c and a + c > b and b + c > a:
    # UNTUK SEGITIGA SAMA SISI
    if a == b == c:
        print("Segitiga Sama Sisi")
    # UNTUK SEGITIGA SAMA KAKI
    elif a == b or a == c or b == c:
        print("Segitiga Sama Kaki")
    # UNTUK SEGITIGA SIKU SIKU
    elif a**2 + b**2 == c**2 or a**2 + c**2 == b**2 or b**2 + c**2 == a**2:
        print("Segitiga Siku-Siku")
    # SEGITIGA SEMBARANG JIKA TIDAK AA KONDISI LAIN
    else:
        print("Segitiga Sembarang")
else:
    print("Bukan Segitiga")
```

### Output



```

if a + b > c and a + c > b and b + c > a:
    # UNTUK SEGITIGA SAMA SISI
    if a == b == c:
        print("Segitiga Sama Sisi")
    # UNTUK SEGITIGA SAMA KAKI
    elif a == b or a == c or b == c:
        print("Segitiga Sama Kaki")
    # UNTUK SEGITIGA SIKU SIKU
    elif a**2 + b**2 == c**2 or a**2 + c**2 == b**2 or b**2 + c**2 == a**2:
        print("Segitiga Siku-Siku")
    # SEGITIGA SEMBARANG JIKA TIDAK AA KONDISI LAIN
    else:
        print("Segitiga Sembarang")
else:
    print("Bukan Segitiga")

Masukkan panjang sisi pertama: 34
Masukkan panjang sisi kedua: 44
Masukkan panjang sisi ketiga: 45
Segitiga Sembarang

```

b. Latihan Kedua

Buatlah program untuk mencari Akar Persamaan Kuadrat dan Determinan

Pseudocode

START

Input koefisien persamaan kuadrat

Tampilkan "Masukkan nilai a: "

INPUT a Simpan input ke variabel a

Tampilkan "Masukkan nilai b: "

INPUT b Simpan input ke variabel b

Tampilkan "Masukkan nilai c: "

INPUT c Simpan input ke variabel c

Hitung determinan

$D = b^2 - 4 * a * c$

Tampilkan "Determinant (D) = " + D

Cek jenis akar berdasarkan nilai determinan

IF D > 0 THEN

    Hitung dua akar nyata

        akar1 =  $(-b + \sqrt{D}) / (2 * a)$



```
    akar2 = (-b - sqrt(D)) / (2 * a)
    Tampilkan "Akar-akar nyata dan berbeda: " + akar1 + ", " + akar2
ELSE IF D == 0 THEN
    Hitung satu akar kembar
    akar = -b / (2 * a)
    Tampilkan "Akar nyata dan kembar: " + akar
ELSE
    Hitung dua akar kompleks
    real_part = -b / (2 * a)
    imag_part = sqrt(-D) / (2 * a)
    akar1 = real_part + imag_part * i
    akar2 = real_part - imag_part * i
    Tampilkan "Akar-akar kompleks: " + akar1 + ", " + akar2
END
```

Input Output Process



**Input:**

1. Nilai aaa: Koefisien kuadrat (float).
2. Nilai bbb: Koefisien linear (float).
3. Nilai ccc: Konstanta (float).

**Proses:**

1. Menghitung **determinan**  $D=b^2-4ac$
2. Memeriksa nilai determinan untuk menentukan jenis akar:
  - o Jika  $D>0$ : Dua akar nyata berbeda.
    - $\text{akar1} = -\frac{b + \sqrt{D}}{2a}$
    - $\text{akar2} = -\frac{b - \sqrt{D}}{2a}$
  - o Jika  $D=0$ : Satu akar nyata kembar.
    - $\text{akar} = -\frac{b}{2a}$
  - o Jika  $D<0$ : Dua akar kompleks.
    - Real part:  $-\frac{b}{2a}$
    - Imaginary part:  $-\frac{\sqrt{-D}}{2a}$
    - Akar kompleks:  $\text{akar1} = \text{real} + i\text{imaginary}$  dan  $\text{akar2} = \text{real} - i\text{imaginary}$

**Output:**

1. Jika  $D>0$ : Menampilkan dua akar nyata berbeda.
2. Jika  $D=0$ : Menampilkan satu akar nyata kembar.
3. Jika  $D<0$ : Menampilkan dua akar kompleks.

[Source Code](#)



```
import math

def akar_persamaan_kuadrat(a, b, c):
    # Hitung determinan
    D = b**2 - 4*a*c

    print(f"Determinant (D) = {D}")

    # Jika determinan positif, dua akar nyata
    if D > 0:
        akar1 = (-b + math.sqrt(D)) / (2*a)
        akar2 = (-b - math.sqrt(D)) / (2*a)
        print(f"Akar-akar nyata dan berbeda: {akar1}, {akar2}")

    # Jika determinan nol, satu akar nyata
    elif D == 0:
        akar = -b / (2*a)
        print(f"Akar nyata dan kembar: {akar}")

    # Jika determinan negatif, dua akar kompleks
    else:
        real_part = -b / (2*a)
        imag_part = math.sqrt(-D) / (2*a)
        akar1 = complex(real_part, imag_part)
        akar2 = complex(real_part, -imag_part)
        print(f"Akar-akar kompleks: {akar1}, {akar2}")

# Input koefisien
a = float(input("Masukkan nilai a: "))
b = float(input("Masukkan nilai b: "))
c = float(input("Masukkan nilai c: "))

# Hitung akar-akar persamaan kuadrat
akar_persamaan_kuadrat(a, b, c)
```



Output

The screenshot shows a Jupyter Notebook cell with the following Python code:

```
# Jika determinan positif, dua akar nyata
if D > 0:
    akar1 = (-b + math.sqrt(D)) / (2*a)
    akar2 = (-b - math.sqrt(D)) / (2*a)
    print(f"Akar-akar nyata dan berbeda: {akar1}, {akar2}")
# Jika determinan nol, satu akar nyata
elif D == 0:
    akar = -b / (2*a)
    print(f"Akar nyata dan kembar: {akar}")
# Jika determinan negatif, dua akar kompleks
else:
    real_part = -b / (2*a)
    imag_part = math.sqrt(-D) / (2*a)
    akar1 = complex(real_part, imag_part)
    akar2 = complex(real_part, -imag_part)
    print(f"Akar-akar kompleks: {akar1}, {akar2}")

# Input koefisien
a = float(input("Masukkan nilai a: "))
b = float(input("Masukkan nilai b: "))
c = float(input("Masukkan nilai c: "))

# Hitung akar-akar persamaan kuadrat
akar_persamaan_kuadrat(a, b, c)
```

The output of the code is:

```
Masukkan nilai a: 4
Masukkan nilai b: 6
Masukkan nilai c: 8
Determinant (D) = -92.0
Akar-akar kompleks: (-0.75+1.1989578808281798j), (-0.75-1.1989578808281798j)
```

Double-click (or enter) to edit

**4. File Praktikum**

Github Repository:

<https://github.com/muhrizki26/praktek-algo-3.git>

**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. Ketika sebuah program dengan percabangan if/else menghasilkan nilai false, kondisi else akan dieksekusi. Ini berarti bahwa kondisi if tidak terpenuhi, dan program akan masuk ke blok else.

Penjelasan dan Alasan: if: Blok percabangan pertama memeriksa apakah suatu kondisi terpenuhi atau tidak. Jika kondisi benar (bernilai true), program akan menjalankan instruksi di dalam blok if. Jika kondisi di blok if tidak terpenuhi (bernilai false), program akan langsung masuk ke blok else dan menjalankan instruksi yang ada di sana.

Status yang diberikan kondisi else: Blok else dieksekusi tanpa kondisi tambahan diperiksa. Oleh karena itu, ketika kondisi if gagal, statusnya adalah eksekusi default. Kondisi lain di sini menjamin bahwa program terus berjalan meskipun kondisi utama tidak terpenuhi.

2. 1. Program menerima input untuk koefisien \*\*a\*\*, \*\*b\*\*, dan \*\*c\*\*.

2. Program menghitung \*\*determinan (D)\*\*.

- 3 Akar-akar dihitung menggunakan \*\*cmath.sqrt()\*\* untuk menangani akar kuadrat dari bilangan negatif.

4.#Program mencetak kedua akar, baik real maupun kompleks.

Program untuk mencari akar persamaan kuadrat dimulai dengan mengimpor modul cmath untuk menangani akar kuadrat dari bilangan kompleks. Kemudian, program meminta pengguna untuk memasukkan koefisien a, b, dan c sebagai bilangan desimal. Program menghitung determinan dengan menggunakan rumus  $D = b^2 - 4ac$  setelah input diterima. Nilai D menunjukkan jenis akar persamaan kuadrat: jika D lebih besar dari nol, persamaan memiliki dua akar nyata yang berbeda; jika D sama dengan nol, persamaan memiliki satu akar nyata (akar ganda); dan jika D kurang dari nol, persamaan memiliki dua akar kompleks. Program ini membantu dalam menyelesaikan persamaan kuadrat secara efektif karena akar-akar tersebut dihitung menggunakan rumus kuadrat  $(-b \pm \sqrt{D}) / 2a$ , dan hasilnya dicetak untuk ditunjukkan kepada pengguna.

## 6. Kesimpulan

- Dalam penggeraan program dengan bahasa pemrograman Python, kita harus benar-benar teliti dalam menginputkan suatu fungsi untuk menampilkan suatu keluaran pada layar dengan sesuai.
- Kita dapat mengetahui bagaimana cara untuk mengetahui suatu program dimana untuk mencari sisi pada rumus segitiga dan mengidentifikasi bentuk segitiga sama sisi,kaki,maupun siku".latihan kedua mempelajari untuk mencari Akar Persamaan Kuadrat dan Determinan dengan rumus rumus atau Bahasa program untuk mengetahui suatu nilai terebut

## 7. Cek List (✓)



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

## 8. Formulir Umpam Balik

No	Elemen Kompetensi	Waktu Pengerjaan	Kriteria
1.	Latihan Pertama	35 Menit	menarik
2.	Latihan Kedua	60 Menit	menarik

Keterangan:

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

