LAPORAN RESMI
MODUL IV
POLA DAN STRUKTUR PERULANGAN
ALGORITMA PEMROGRAMAN

NAMA	**: FAHIRA ANNISA**
N.R.P	**: 250441100096**
DOSEN	**: FITRI DAMAYANTI, S.KOM., M.KOM**
ASISTEN	**: MUHAMMAD MAULANA KHANIF**
TGL PRAKTIKUM	**: 24 OKTOBER 2025**

Disetujui : 28 0KTOBER 2025
Asisten

MUHAMMAD MAULANA KHANIF
23.04.411.00047

LABORATORIUM BISNIS INTELIJEN SISTEM
PRODI SISTEM INFORMASI
JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS TRUNOJOYO MADURA

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perulangan merupakan fondasi pemrograman yang memungkinkan eksekusi kode berulang berdasarkan kondisi spesifik, mengatasi tugas repetitif seperti pemrosesan data atau simulasi. Struktur perulangan meliputi tiga komponen kunci: inisialisasi untuk menetapkan nilai awal, kondisi boolean yang mengatur penghentian, dan iterasi untuk pembaruan. Jenis utama mencakup for loop untuk iterasi terbatas (contoh: for i in range(5): print(i)), while loop untuk kondisi fleksibel, do-while untuk eksekusi minimal sekali, serta nested loops untuk struktur multidimensi seperti matriks.

Pola perulangan mengacu pada skema umum untuk menyelesaikan masalah algoritmik, meminimalkan redundansi kode. Pola iteratif sederhana digunakan untuk tugas seperti penjumlahan array, sementara pola pencarian (misalnya, linear search) memanfaatkan loop while dengan kondisi berhenti saat elemen ditemukan. Pola akumulasi mengumpulkan hasil, seperti perhitungan rata-rata, melalui inisialisasi variabel dan penambahan bertahap. Dibandingkan rekursi, perulangan iteratif lebih efisien untuk skala besar, menghindari risiko stack overflow. Pola paralel, seperti distribusi loop ke thread ganda, meningkatkan performa dalam pemrograman paralel.

Perulangan berevolusi dari instruksi mesin primitif menjadi konstruksi tingkat tinggi, dipengaruhi paradigma structured programming oleh Dijkstra pada 1960-an. Optimasi seperti loop unrolling oleh compiler mempercepat eksekusi, sementara kontrol tambahan seperti break dan continue mencegah kesalahan umum seperti infinite loop. Pemahaman pola ini krusial untuk algoritma efisien, termasuk sorting (bubble sort) atau traversal graf, memastikan kode yang ringkas dan andal.

1.2 Tujuan

- Mahasiswa mampu memahami bagaimana pola dan struktur dibangun menggunakan perulangan dalam python.
- Mahasiswa dapat menyajikan contoh penggunaan perulangan for dan while untuk menyusun kombinasi, perhitungan matematis, dan pola grafis.

BAB II
DASAR TEORI

2.1 Dasar Teori

Dalam pemrograman, salah satu konsep penting yang sering digunakan adalah perulangan (loops). Perulangan memungkinkan eksekusi sebuah blok kode berulang kali berdasarkan kondisi tertentu, sehingga memudahkan pengulangan tugas yang sama tanpa perlu menulis kode yang berulang. Bahasa pemrograman Python menyediakan beberapa jenis perulangan seperti for loop dan while loop, yang digunakan dalam berbagai skenario. Salah satu aplikasi utama dari perulangan adalah pembentukan pola, kombinasi angka, atau urutan tertentu yang sering digunakan dalam masalah matematika, fisika, atau dalam berbagai algoritma. Dengan memanfaatkan perulangan bersarang (nested loops), kita dapat menyusun struktur data atau pola yang lebih kompleks seperti kombinasi biner, pola piramida, atau bahkan deret angka seperti Fibonacci. Dalam pembahasan ini, kita akan mempelajari bagaimana pola dan struktur tertentu dapat dibentuk menggunakan perulangan for dan while di Python. Pembahasan ini juga mencakup teknik kombinasi nested loops yang digunakan untuk menghasilkan output yang lebih rumit, seperti kombinasi angka biner, perhitungan FPB, bilangan Fibonacci, serta pola bintang dan angka.

2.2 Perulangan Bersarang (Nested Loops)

Nested loop adalah perulangan di dalam perulangan lainnya. Ini sering digunakan saat kita ingin mengulangi beberapa operasi untuk setiap elemen dari loop luar. Misalnya, dalam pembentukan pola dua dimensi seperti matriks atau tabel, perulangan bersarang menjadi sangat penting untuk mengakses elemen dalam dua dimensi tersebut.

Contoh :

```
# Loop luar (outer loop)
for i in range(1, 4):  # Loop ini akan berjalan 3 kali (i = 1, 2, 3)
    print(f"Loop luar i = {i}")

    # Loop dalam (inner loop)
```

```
    for j in range(1, 4): # Loop ini juga akan berjalan 3 kali
untuk setiap iterasi dari loop luar
        print (f" Loop dalam j = {j}")
```

Penjelasan :

1 Perulangan i pertama (i = 1) :

Mencetak "Loop luar i = 1"

a. Perulangan j pertama (j = 1) :

Mencetak "Loop dalam j = 1"

b. Perulangan j kedua (j = 2) :

Mencetak "Loop dalam j = 2"

c. Perulangan j ketiga (j = 3) :

Mencetak "Loop dalam j = 3"

2 Perulangan i pertama (i = 2) :

Mencetak "Loop luar i = 2"

a. Perulangan j pertama (j = 1)

Mencetak "Loop dalam j = 1"

b. Perulangan j kedua (j = 2)

Mencetak "Loop dalam j = 2"

c. Perulangan j ketiga (j = 3)

Mencetak "Loop dalam j = 3"

3 Perulangan i pertama (i = 3) :

Mencetak "Loop luar i = 3"

a. Perulangan j pertama (j = 1)

Mencetak "Loop dalam j = 1"

b. Perulangan j kedua (j = 2)

Mencetak "Loop dalam j = 2"

c. Perulangan j ketiga (j = 3)

Mencetak "Loop dalam j = 3"

Output :

```
 Loop luar i = 1
 Loop dalam j = 1
 Loop dalam j = 2
 Loop dalam j = 3
Loop luar i = 2
```

```
 Loop dalam j = 1
 Loop dalam j = 2
 Loop dalam j = 3
Loop luar i = 3
 Loop dalam j = 1
 Loop dalam j = 2
 Loop dalam j = 3
```

2.3 Struktur pengulangan dan kombinasi

Struktur berulang seperti deret biner, angka Fibonacci, atau perhitungan FPB dapat diprogram seara efisien menggunakan perulangan. Setiap struktur ini memiliki pola pengulangan yang khas, yang dapat dioptimalkan dengan menggunakan teknik pengulangan yang tepat.

Contoh:

```python
a = 24
b = 36

while b != 0:
    a, b = b, a % b

print(f"FPB-nya adalah: {a}")
```

Penjelasan:

a = 24, b = 3

1. **Perulangan pertama:**
   a. Kondisi b != 0 adalah benar (b = 36).
   b. Nilai a diperbarui menjadi 3 (nilai b).
   c. Nilai b diperbarui menjadi 36% 24, yang menghasilkan 12.
   d. Setelah perulangan a = 36, b = 24.2.
2. **Perulangan kedua:**
   a. Kondisi b != 0 masih benar (b = 24).
   b. Nilai a diperbarui menjadi 24 (nilai b).
   c. Nilai b diperbarui menjadi 36 % 24, yang menghasilkan 12.
   d. Setelah perulangan: a = 24, b = 12. 3.
3. **Perulangan ketiga:**
   a. Kondisi b != 0 masih benar (b = 12).
   b. Nilai a diperbarui menjadi 12 (nilai b).
   c. Nilai b diperbarui menjadi 24 % 12, yang menghasilkan 0.
   d. Setelah perulangan: a = 12, b = 0. 4.

4. **Perulangan selesai:**

   a. Kondisi b != 0 tidak lagi benar (b = 0), sehingga perulangan berhenti.

5. **Hasil:**

a. Nilai a yang tersisa adalah 12, yang merupakan FPB dari 24 dan 36 Setiap iterasi perulangan, nilai b akan di-update dengan sisa bagi (a % b), dan iterasi terus berlanjut hingga b menjadi nol. Nilai a terakhir saat b = 0 adalah FPB-nya.

Output:

```
2025/modul-4
FPB-nya adalah: 12
PS C:\Users\SONI
```

2.4 Pola Matematika dalam Perulangan

Banyak pola matematika, seperti deret Fibonacci, perhitungan FPB, dan lainnya dapat diselesaikan menggunakan pendekatan berbasis loop. Misalnya, urutan bilangan Fibonacci dapat dihitung dengan memperbarui nilai dua bilangan sebelumnya pada setiap iterasi, sedangkan FPB (Faktor Persekutuan Terbesar) dapat ditemukan menggunakan algoritma Euclidean berbasis modulus, yang terus menerus mengulang hingga mencapai hasil yang diinginkan.

Contoh Deret Fibonacci:

```python
n = 100  #batas angka
a, b = 0,1

print("Bilangan Fibonacci hingga", n)
while a <= n:
    print(a, end=" ")
    a, b = b, a + b
```

Penjelasan :

1. **Inisialisasi :**

   a. n diatur ke 100, yang merupakan batas atas untuk bilangan Fibonacci yang akan dicetak.

   b. a diinisialisasi dengan 0 (bilangan Fibonacci pertama) dan b diinisialisasi dengan 1 (bilangan Fibonacci kedua).'

2. **Perulangan pertama :**

   a. Kondisi a <=n adalah benar (a = 0 <= 100).

   b. Bilangan a (0) dicetak ke layar.

c. Nilai a diupdate menjadi nilai b (1), dan b diupdate menjadi hasil penjumlahan a + b (0 + 1), sehingga setelah perulangan a = 1, b = 1.

3. **Perulangan kedua :**
   a. Kondisi a <= n masih benar (a = 1 <= 100).
   b. Bilangan a (1) dicetak ke layar.
   c. Nilai a diupdate menjadi nilai b (1), dan b diupdate menjadi hasil penjumlahan a + b (1+1), sehingga setelah perulangan: a = 1, b = 2.
4. **Perulangan ketiga :**
   a. Kondisi a <= n masih benar (a = 1 <= 100).
   b. Bilangan a (1) dicetak ke layar.
   c. Nilai a diupdate menjadi nilai b (2), dan bilangan b diupdate menjadi hasil penjumlahan a + b (2 + 3), sehingga setelah perulangan: a = 2, b = 3.
5. **Perulangan keempat :**
   a. Kondisi a <= n masih benar (a = 2 <= 100).
   b. Bilangan a (2) dicetak ke layar.
   c. Nilai a diupdate menjadi nilai b (3), dan b diupdate menjadi hasil penjumlahan a+b (2+3), sehingga setelah perulangan a = 3, b = 5.
6. **Proses berlanjut :**
   a. Proses ini berulang, dimana setiap iterasi mencetak nilai a dan memperbarui nilai a dan b hingga a menjadi lebih besar dari 100. Beberapa nilai yang dicetak antara lain 3, 5, 8, 13, 21, 34, 55, 89, dan terakhir 144, dimana pada iterasi ini kondisi a <= n menjadi salah.
7. **Akhir perulangam :**
   a. Ketika a mencapai 144, kondisi a <= n tidak lagi benar (144 > 100), sehingga perulangan berhenti.

Output :

```
0441100096_Fanira Annisa/tugas.
Bilangan Fibonacci hingga 100
0 1 1 2 3 5 8 13 21 34 55 89
PS C:\Users\SONIC MASTER\Docum
ritma-pemrograman-1B-2025>
```

2.5 Pola Grafis dalam Perulangan

Dengan memanfaatkan nested loops, kita bisa mencetak pola visual seperti piramida bintang atau angka. Struktur seperti ini membantu kita dalam memahami logika kontrol aliran program, serta memberikan pemahaman tentang bagaimana sebuah program dapat digunakan untuk menyusun representasi visual atau gravis.

Contoh :

```
n = 5

for i in range(1, n+1):

    for j in range(n-i):
        print(' ', end= ' ')

    for k in range(1, i+1):
        print(k, end=' ')
    print()
```

Penjelasan:

1. **Perulangan i Pertama (i = 1):**
   a. **Perulangan j (j = 0 hingga 3) :**
      Karena n - i = 5 - 1 = 4, maka perulangan j berjalan 4 kali dan mencetak 4 spasi. Ini menggeser angka yang akan dicetak pada baris ini.
   b. **Perulangan k (k = 1 hingga 1) :**
      Perulangan k berjalan 1 kali dan mencetak angka 1.
   c. Setelah kedua perulangan selesai, program pindah ke baris baru.
2. **Perulangan i Kedua (i = 2):**
   a. **Perulangan j (j = 0 hingga 2) :**
      Karena n - i = 5 - 2 = 3, maka perulangan j berjalan 3 kali dan mencetak 3 spasi untuk menggeser angka.
   b. Perulangan k (k = 1 hingga 2) :
      Perulangan k berjalan 2 kali, mencetak angka 1 2.
   c. **Setelah selesai, pindah ke baris baru.**
3. **Perulangan i Ketiga (i = 3):**
   a. **Perulangan j (j = 0 hingga 1) :**

Karena n - i = 5 - 3 = 2, maka perulangan j berjalan 2 kali, mencetak spasi.

b. Perulangan k (k = 1 hingga 3) :

Perulangan k berjalan 3 kali, mencetak angka 1 2 3.

c. Pindah ke baris berikutnya.

4. Perulangan i Keempat (i = 4):

a. Perulangan j (j = 0 hingga 0) :
Karena n - i = 5 - 4 = 1, maka perulangan j berjalan 1 kali, mencetak 1 spasi.

b. Perulangan k (k = 1 hingga 4) :
Perulangan k berjalan 4 kali, mencetak angka 1 2 3 4.

c. Pindah ke baris berikutnya.

5. Perulangan i Kelima (i = 5):

a. Perulangan j (tidak berjalan karena n – i = 0) :

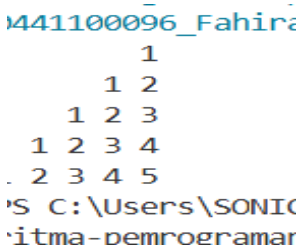
Tidak ada spasi yang dicetak, karena n – i = 0.

b. Perulangan k (k = 1 hingga 5) :

Perulangan k berjalan 5 kali, mencetak angka 1 2 3 4 5.

c. Setelah selesai, program tidak lagi pindah baris karena ini adalah perulangan terakhir.

Output :



BAB III
TUGAS PENDAHULUAN

3.1 Soal

1. Mengapa penggunaan perulangan sangat penting dalam pemrograman? Berikan beberapa contoh keuntungannya!
2. Jelaskan tentang bagaimana perulangan bersarang (nested loop) bekerja, dan berikan contoh implementasinya!
3. Apa perbedaan utama antara perulangan for dan while dalam Python? Jelaskan kapan lebih baik menggunakan for dibandingkan while, dan sebaliknya!
4. Apa potensi masalah yang bisa muncul saat menggunakan nested loops dalam pemrograman? Jelaskan disertai contohnya!
5. Apa yang dimaksud dengan deret fibonacci? Jelaskan cara menghitung angka fibonacci menggunakan struktur pengulangan dan berikan contohnya!
6. Buatlah program dengan studi kasus sebagai berikut:
   Seorang siswa ingin membuat program sederhana untuk menampilkan pola piramida angka secara otomatis menggunakan Python. Program harus meminta pengguna memasukkan jumlah baris pola yang diinginkan. Setiap baris menampilkan deretan angka naik secara berurutan sehingga membentuk pola seperti piramida.
   Dengan ketentuan program sebagai berikut:
   - Gunakan struktur nested loop (perulangan bersarang).
   - Pola harus menampilkan angka 1 hingga baris ke-n.
   - Program harus tetap rapi, dengan jarak spasi yang sesuai agar berbentuk segitiga.

3.2 Jawaban

1. Karena perulangan memungkinkan kode untuk menjalankan instruksi berulang tanpa duplikasi, membuat program lebih efisien dan ringkas.

   Keuntungan :

   - Mengurangi duplikasi
   - Menangani data variabel
   - perubahan di satu tempat

2. Cara kerja perulangan bersarang :

   - Loop luar : Adalah loop utama yang mengontrol jumlah iterasi keseluruhan.
   - Loop Dalam : Adalah loop yang berada di dalam loop luar. Setiap kali loop luar berjalan sekali, loop dalam akan di eksekusi dari awal sampai akhir.
   - Urutan Eksekusi : Jika loop luar memiliki 3 iterasi dan loop dalam memiliki 4 iterasi, maka total eksekusi kode di dalam loop adalah 3x4 : 12
   - Variabel Loop : Loop luar menggunakan variabel seperti 'i', dan loop dalam menggunakan 'j', untuk menghindari konflik.

   contoh implementasinya :

```
for i in range(1, 4): # Loop luar : 3 baris
    for j in range (i): # Loop dalam : cetak buku
                                sebanyak i
        Print (" buku", end = " ")
    Print () # Pindah baris
```

3. - for : Iterasi melalui Iterable (list, range) dengan jumlah iterasi otomatis. Biasanya digunakan untuk data yang sudah diketahui panjangnya
   - while : Akan berjalan selama kondisi 'True', jumlah iterasi yang tidak pasti. Digunakan untuk kondisi dinamis.
   - Gunakan for saat iterasi melalui koleksi atau jumlah loop diketahui, agar lebih aman hindari infinite loop.

- Gunakan while saat kondisi berhenti bergantung pada perubahan (input , flag)

4. Potensi masalah pada Nested Loops
   - Kompleksitas waktu yang tinggi
     Loop bersarang akan meningkatkan waktu eksekusi secara eksponensial yang menyebabkan kurang cepat untuk input besar
   - Loop tak terbatas
     Risiko bug dalam kondisi atau indeks, seperti infinite loop
   - Penggunaan memori berlebih
     Penyimpanan data dalam loop bisa menyebabkan memory leak atau overflow
     contoh :

```
for i in range (100):
    for j in range (100):
        print (i,j)
```

5. Deret fibonacci adalah urutan bilangan yang dimana setiap angka adalah jumlah dari dua angka sebelumnya, dimulai dari 0 dan 1 atau 1 dan 1.
   untuk menghitung angka Fibonacci kita dapat menggunakan loop seperti for atau while untuk menghasilkan deret sampai posisi yang diinginkan.
   contoh :

```
a,b : 0,1
for i in range (12):
    Print (a, end: '')
    a,b : b, a+b
```

6.

```
n : int (input ("Masukkan jumlah baris :")
for i in range ( 1, n+1 ):
    Print (" " * (n-i), end = " ")
    for j in range ( 1 , i + 1 ):
        Print (j, end = " ")
    Print ()
```

Outputnya :

```
    1
   1 2
  1 2 3
 1 2 3 4
1 2 3 4 5
```

BAB IV
IMPLEMENTASI

4.1 Tugas Praktikum

4.1.1 Tugas Praktikum Soal No. 1

Mas rusdi kini mengawasi beberapa baris lampu di taman kota, dan setiap barismemiliki jumlah lampu yang berbeda. Untuk membantu temannya, mas narji ingin membuat program Python yang dapat menampilkan kondisi setiap lampu. Program harus meminta input jumlah baris lampu, lalu jumlah lampu di setiap baris. Setiap lampu diberi nomor urut, dan program menampilkan pesan "Lampu ke-[x] pada baris [y] menyala." Jika nomor lampu merupakan kelipatan 3, tampilkan "Lampu ke-[x] pada baris [y] rusak." Selain itu, jika baris lampu adalah baris terakhir, tambahkan pesan "Periksa sambungan daya utama." Program ini membantu mas rusdi dan temannya mengetahui kondisi setiap lampu dengan cepat dan efisien tanpa harus memeriksa secara manual satu per satu.

4.1.2 Tugas Praktikum Soal No. 2

Pak wowo kini bekerja di perusahaan besar dengan dua shift, pagi dan malam. Ia ingin menghitung total gaji mingguan dengan memperhitungkan lembur dan bonus shift malam. Buatlah program Python yang menggunakan perulangan for selama 7 hari. Program harus meminta input jumlah jam lembur dan apakah hari itu termasuk shift malam (y/n). Setiap hari, gaji pokok Pak wowo adalah Rp100.000, dengan tambahan lembur: 1–3 jam = Rp25.000 per jam, 4 jam = Rp100.000, 6 jam = Rp200.000, 8 jam = Rp300.000. Jika lembur lebih dari 8 jam, tampilkan "Lembur melebihi batas, tidak dihitung." Jika hari tersebut shift malam, tambahkan bonus Rp50.000. Setelah 7 hari, program menampilkan total jam lembur, total bonus shift malam, dan total gaji seminggu. Program ini harus menangani input tidak valid dan tetap berjalan hingga semua data lengkap dimasukkan dengan benar.

4.1.3 Tugas Praktikum Soal No. 3

Bjirka senang membuat pola angka untuk menenangkan pikirannya. Kali ini, ia ingin membuat dua piramida angka yang saling berhadapan seperti cermin. Jika pengguna memasukkan angka n = 5. Program harus menggunakan tiga perulangan bersarang (nested loop) – satu kiri piramida, satu untuk ruang kosong di tengah, dan satu untuk sisi kanan piramida. Gunakan logika agar jumlah angka

dan spasi berubah sesuai barisnya sehingga pola tetap simetris untuk nilai n berapa pun.

4.2 Source Code

4.2.1 Tugas Praktikum Soal No. 1

```
# Kondisi setiap lampu
jumlah_baris = int(input("Masukkan jumlah baris lampu: "))

for baris in range(1, jumlah_baris + 1):
    jumlah_lampu = int(input(f"Masukkan jumlah lampu pada baris
{baris}: "))

    for lampu in range(1, jumlah_lampu +1):
        if lampu % 3 == 0:
            print(f"Lampu ke-{lampu} pada baris {baris} rusak")
        else:
            print(f"Lampu ke-{lampu} pada baris {baris} menyala")

    if baris == jumlah_baris:
        print("Memeriksa sambungan data utama")
```

4.2.2 Tugas Praktikum Soal No. 2

```
total_lembur = 0
total_bonus = 0
total_gaji = 0

for i in range(7):
    print(f"\n hari ke-{i+1}")
    jam = int(input("Masukkan jam lembur: "))
    shift = input("shift malam? (iya/tidak):").lower()

    gaji = 100000 # Rp.100.000
    if 1 <= jam <= 3:
        lembur = jam * 25000 # Rp.25.000
    elif jam == 4:
        lembur = 100000 # Rp.100.000
    elif jam == 6:
        lembur = 200000 # Rp.200.000
    elif jam == 8:
        lembur = 300000 # Rp.300.000
    elif jam > 8:
        print("Lembur melebihi batas")
        lembur = 0
    else:
        lembur = 0

    bonus = 50000  if shift == "iya"else 0
```

```
    total_gaji += gaji + lembur + bonus
    total_lembur += jam
    total_bonus += bonus

print("\n=== Total Mingguan===")
print("Total jam lembur:", total_lembur)
print("Total bonus: Rp", total_bonus)
print("Total gaji: Rp", total_gaji)
```

4.2.3 Tugas Praktikum Soal No. 3

```
n = 5

for i in range(1, n + 1):
    # Sisi kiri piramida: cetak angka dari 1 sampai i
    for j in range(1, i + 1):
        print(j, end="")

    # Ruang kosong di tengah: cetak spasi sebanyak 2 * (n - i)
    for k in range(2 * (n - i)):
        print(" ", end="")

    # Sisi kanan piramida: cetak angka dari i sampai 1
    for l in range(i, 0, -1):
        print(l, end="")

    # Pindah ke baris berikutnya
    print()
```

4.3 Hasil

4.3.1 Tugas Praktikum Soal No. 1

```
0441100096_Fahira Annisa/tugas1.py"
Masukkan jumlah baris lampu: 2
Masukkan jumlah lampu pada baris 1: 4
Lampu ke-1 pada baris 1 menyala
Lampu ke-2 pada baris 1 menyala
Lampu ke-3 pada baris 1 rusak
Lampu ke-4 pada baris 1 menyala
Masukkan jumlah lampu pada baris 2: 3
Lampu ke-1 pada baris 2 menyala
Lampu ke-2 pada baris 2 menyala
Lampu ke-3 pada baris 2 rusak
Memeriksa sambungan data utama
PS C:\Users\SONIC MASTER\Documents\gith
```

4.3.2 Tugas Praktikum Soal No. 2

```
 hari ke-1
Masukkan jam lembur: 3
shift malam? (iya/tidak):iya

 hari ke-2
Masukkan jam lembur: 4
shift malam? (iya/tidak):tidak

 hari ke-3
Masukkan jam lembur: 6
shift malam? (iya/tidak):iya

 hari ke-4
Masukkan jam lembur: 8
shift malam? (iya/tidak):tidak

 hari ke-5
Masukkan jam lembur: 4
shift malam? (iya/tidak):iya

 hari ke-6
Masukkan jam lembur: 3
shift malam? (iya/tidak):tidak

 hari ke-7
Masukkan jam lembur: 8
shift malam? (iya/tidak):iya

=== Total Mingguan===
Total jam lembur: 36
Total bonus: Rp 200000
Total gaji: Rp 2050000
PS C:\Users\SONIC MASTER\Documents
```

4.3.3 Tugas Praktikum Soal No. 3

```
/Users/SONIC MA
1         1
12       21
123     321
1234   4321
1234554321
PS C:\Users\SON
```

4.4 Penjelasan

4.4.1 Tugas Praktikum Soal No. 1

Di soal No 1, kita disuruh menginputkan jumlah baris yang berfungsi untuk menyimpan jumlah baris lampu, kemudian masukkan pengulangan for baris in range(1, jumlah_baris + 1) yang akan menghasilkan urutan angka dari 1 sampai jumlah barisnya, dan masukkan variabel jumlah_lampu yang menginputkan jumlah lampu pada baris, baris di dalam tanda kurung kurawal digunakan agar nomor baris

bisa ditampilkan di teks. Dan masukkan perulangan for lampu in range(1, jumlah_lampu + 1) perulangan ini digunakan untuk mengecek setiap lampu di dalam baris tertentu, jika jumlah lampu = 5 maka lampu akan bernilai 1, 2, 3, 4, dan 5. Selanjutnya masukkan if lampu % 3 == 0 yang digunakan untuk memeriksa apakah nomor lampu termasuk kelipatan 3, lampu % 3 fungsinya untuk mengetahui hasil sisa bagi lampu dengan 3, jika bukan kelipatan 3 maka lampu akan rusak. Setelah selesai memeriksa lampu di setiap baris, program akan mengecek apakah baris ini termasuk baris terakhir jika benar maka akan menampilkan pesan "periksa sambungan daya utama" dan pesan ini akan muncul ketika di akhir progra.

4.4.2 Tugas Praktikum Soal No. 2

Di soal No 2, langkah pertama yaitu membuat variabel total_lembur, total_bonus dan total_gaji, variabel ini digunakan untuk menjumlahkan nilai-nilai dari setiap hari. Kemudian masukkan perulangan for i in range (7) untuk memungkinkan program meminta input dan menghitung data setiap hari secara berulang, selanjutnya print(f"\n hari ke-{i+1}") yang akan mencetak pesan dari hari ke 1 sampai hari ke 7 dengan baris baru di depan untuk pemisah i+1 digunakan karena dimulai dari 0, sehingga hari pertama adalah hari ke 1, setelah itu inputkan jam lembur untuk memberikan konteks hari dan mendapatkan data jam lembur dari pengguna, dan menginputkan shift malam (iya/tidak) untuk menentukan apakah bonus diberikan berdasarkan shift malam. Kemudian masukkan variabel gaji = 100.000 gaji ini akan ditambahkan ke total gaji setiap harinya, masukkan if 1 <= jam <= 3:, lembur = jam * 25.000, elif jam == 4:, lembur = 100.000, elif jam 6:, lembur = 200.000, elif jam == 8:, lembur = 300.000, elif jam > 8 :, print("Lembur melebihi batas"), lembur = 0 , else:, lembur = 0. Tujuannya untuk menghitung bayaran lembur sesuai aturan yang di tentukan, jika jam tidak valid lembur di anggap 0. Kemudian masukkan bonus = 50.000 if shift == "iya" else 0 jika kamu shift malam maka akan mendapatkan bonus 50.000 jika tidak maka tidak akan mendapatkan bonus. Setelah itu masukkan total_gaji += gaji+lembur+bonus, total_lembur += jam, dan total_bonus += bonus untuk mengakumulasikan nilai-nilai harian ke dalam total mingguan lalu print semua total lembur,bonus, dan total gaji untuk memberikan hasil akhir perhitungan kepada pengguna.

4.4.3 Tugas Praktikum Soal No. 3

Pada soal no 3, langkah pertama yaitu masukkan variabel n = 5 yang akan mendeklarasikan variabel n dengan nilai 5. Kemudian masukkan for i in range(1, n+1) : perulangan ini akan berjalan dari i =1 hingga i = 5 karena range (1, 6) menghasilkan 1, 2, 3, 4, 5 dan di setiap iterasi mewakili satu baris piramida, lalu masukkan for j in range (1, i + 1): yang artinya yaitu j = 1 hingga j = i , print (j, end=" ") untuk mencetak nilai j tanpa baris baru untuk end =" " fungsinya untuk membuat output tetap di baris yang sama. Selanjutnya masukkan for k in ramge(2 * (n-i)) untuk menghitung jumlah spasi, lalu print (" ", end= " ") berfungsi untuk mencetak spasi tanpa baris baru dan memastikan sisi kanan dan kiri terpisah oleh ruang yang menurun seiring garis bertambah. Setelah itu masukkan perulangan for 1 in range (i, 0, -1): yang akan mencetak angka dari i turun ke 1 pada sisi kanan piramida, lalu print(1, end=" ") digunakan untuk mencetak nilai i tanpa baris baru dan print() untuk memindahkan output ke baris berikutnya setelah semua elemen pada piramida selesai di cetak.

BAB V
PENUTUP

5.1 Analisa

Dari hasi praktikum, praktikan menganalisa bahwa, perulangan merupakan elemen fundamental dalam pemrograman yang memungkinkan eksekusi kode berulang berdasarkan kondisi, dengan komponen utama seperti inisialisasi, kondisi, badan loop, dan pembaruan. Struktur for loop cocok untuk iterasi dengan jumlah yang diketahui, seperti `for i in range(5): print(i)`, yang efisien untuk array atau range. While loop lebih fleksibel untuk kondisi dinamis, misalnya `while i < 5: print(i); i++`, namun berisiko infinite loop jika tidak diperbarui.

Pola perulangan mencakup traversal untuk mengunjungi elemen struktur data dengan kompleksitas O(n), accumulation untuk mengumpulkan nilai seperti sum array, dan nested loops untuk data multidimensi seperti matriks dengan kompleksitas $O(n^2)$. Pola infinite loop dengan break berguna untuk event-driven sistem, sementara rekursi mensimulasikan loop hierarki meski berisiko stack overflow. Pola ini didukung oleh praktik algoritma, seperti linear search atau bubble sort, yang mengoptimalkan readability dan performa.

5.2 Kesimpulan

Pola dan struktur pengulangan merupakan fondasi penting dalam pemrograman yang memungkinkan eksekusi instruksi secara berulang tanpa penulisan kode yang berulang. Praktikum ini memberikan pemahaman mendalam tentang bagaimana perulangan, dapat dimanfaatkan untuk membentuk pola dan menyelesaikan berbagai permasalahan algoritma. Dari hasil praktikum, praktikan menyimpulkan bahwa:

- Perulangan membuat kode berjalan berulang-ulang dengan bagian awalnya inisialisasi, cek kondisi, dan pembaruan.
- Jenisnya ada for**,** while**,** do-while, dan foreach**,** di pilih sesuai dengan kebutuhan.
- Digunakan supaya tidak perlu menulis kode yang berulang dan meningkatkan efisiensi.