

<p>Nama: MUHAMMAD RIZKI</p> <p>NIM: 064102400020</p>	 <p>Praktikum Algoritma & Pemrograman</p>	<h1>MODUL 8</h1> <p>Nama Dosen: Binti solihah, S.T, M.KOM</p> <p>Nama Asisten Laboratorium:</p> <ol style="list-style-type: none">Yustianas Rombon - 064002300015Vira Aditya Kurniawan - 065002300012
----------------------------------------------------------------------------	---------------------------------------------------------------------------------------------------------------------------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Fungsi Rekursif (Recursive Function)

1. Teori Singkat

Fungsi Rekursif

Fungsi rekursif adalah fungsi yang memanggil dirinya sendiri secara berulang. Jadi di dalam tubuh fungsi yang dideklarasikan kita memanggil fungsi itu sendiri. Rekursif ini sebenarnya merupakan sebuah perulangan di dalam sebuah program. Namun, perulangan rekursif ini sangat berbeda dengan perulangan pada umumnya, seperti while dan for. Walaupun fungsinya sama yaitu untuk melakukan perulangan atau looping. Letak perbedaannya adalah dari cara kerjanya. Jika ‘for’ dan ‘while’ merupakan sebuah perulangan yang menggunakan sebuah kondisi atau Boolean (true/false), maka pada rekursif ini terjadi pada sebuah fungsi atau metode yang memanggil dirinya sendiri. Dari penjelasan tersebut dapat kita katakan sebagai perulangan yang memanggil dirinya sendiri untuk melakukan sebuah perulangan.

Fungsi rekursif dapat menyelesaikan beberapa persoalan seperti perhitungan bilangan fibbonaci dan faktorial.

$$\text{faktorial}(5) = 5 * \text{faktorial}(4)$$

$$\text{faktorial}(4) = 4 * \text{faktorial}(3)$$

$$\text{faktorial}(3) = 3 * \text{faktorial}(2)$$

$$\text{faktorial}(2) = 2 * \text{faktorial}(1)$$

$$\text{faktorial}(1) = 1$$



Maka faktorial(5) = $5 * 4 * 3 * 2 * 1$, akan menghasilkan 120

Source Code

```
#FUnksi Rekursif
def rekursif(angka):
    if angka > 0:
        print (angka)
        angka = angka - 1
        rekursif(angka)
    else:
        print(angka)

masukkan = int(input("masukkan angka: "))
rekursif(masukkan)
```

Output





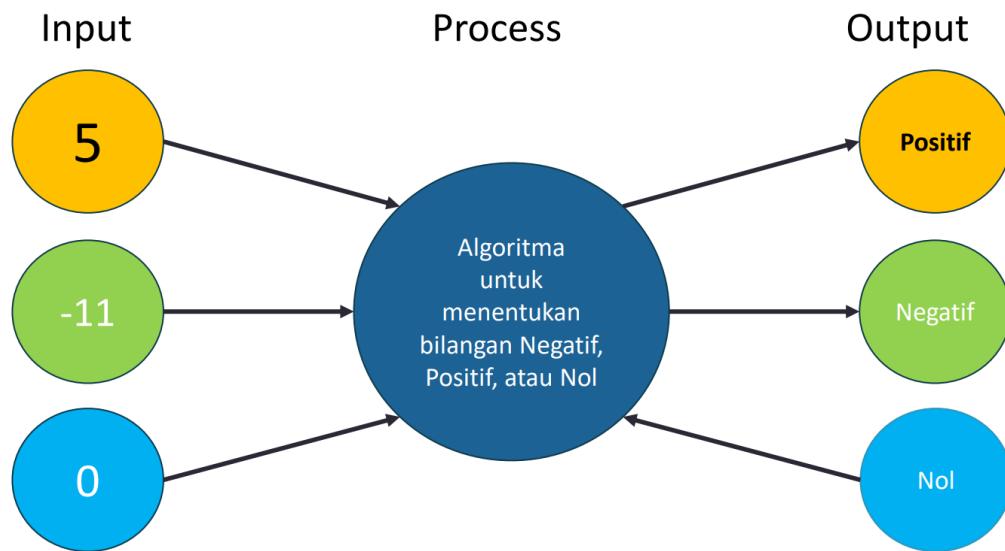
IPO (Input Process Output)

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)



Notasi Algoritma Flowchart

1. Flowchart adalah representasi visual atau diagram alir yang digunakan untuk menggambarkan langkah-langkah dan urutan proses suatu algoritma atau program.
2. Flowchart menyajikan langkah-langkah dalam bentuk simbol-simbol grafis yang saling terhubung, membantu dalam memvisualisasikan bagaimana informasi mengalir dan bagaimana proses dilakukan.
3. Dalam kaitannya dengan notasi deskriptif, notasi algoritma yang menggunakan flowchart dapat lebih cepat dibaca dan dilihat alur dan hubungannya.

Simbol-simbol pada Flowchart

1. Setiap elemen flowchart dihubungkan oleh garis aliran bertanda panah
2. Garis aliran dimulai dari atas symbol dan keluar dari bagian bawah, kecuali symbol keputusan yang alirannya keluar dari bawah atau samping
3. Aliran bergerak dari atas ke bawah
4. Proses awal dan akhir menggunakan symbol terminal.



... Simbol-simbol pada Flowchart



Terminator yang menandakan *Start* (awal) atau *End* (akhir) program.



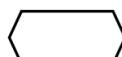
Flow line yang digunakan untuk menunjukkan arah aliran pada program.



Process menunjukkan proses yang dilakukan pada masukan.



Input atau output untuk menunjukkan masukan dan keluaran.



Preparation digunakan untuk membuat deklarasi nilai awal.



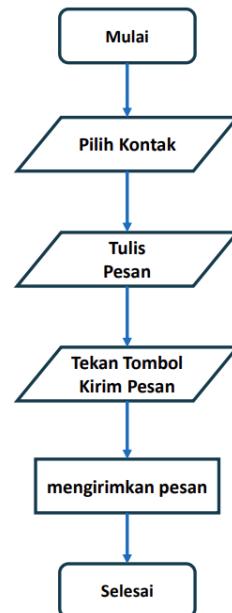
On Page Connector digunakan untuk menghubungkan antar *flowchart*.



Decision menunjukkan keputusan atau kondisi untuk memilih keputusan.

**Contoh sederhana
Penggunaan *flowchart*
untuk menunjukkan algoritma**

**Kasus/Aliran:
Mengirim pesan WhatsApp**



2. Alat dan Bahan

Hardware : Laptop/PC

Software : Spyder (Anaconda Python)

3. Elemen Kompetensi

a. Latihan pertama



Buatlah sebuah fungsi penjumlahan berurut menggunakan konsep rekursif (tanpa for/while). User perlu memasukkan angka awal dan bilangan-bilangan yang ingin ditambahkan seperti pada berikut:

Masukkan Jumlah: 3 (inputan user)

Masukkan angka ke-1: 1 (inputan user)

Masukkan angka ke-2: 2 (inputan user)

Masukkan angka ke-3: 3 (inputan user)

IPO (Input Output Process)

```
def penjumlahan_berurut(jumlah, index=1, total=0):
    if index > jumlah:
        return total
    else:
        angka = int(input(f"Masukkan angka ke-{index}: "))
        return penjumlahan_berurut(jumlah, index + 1, total + angka)

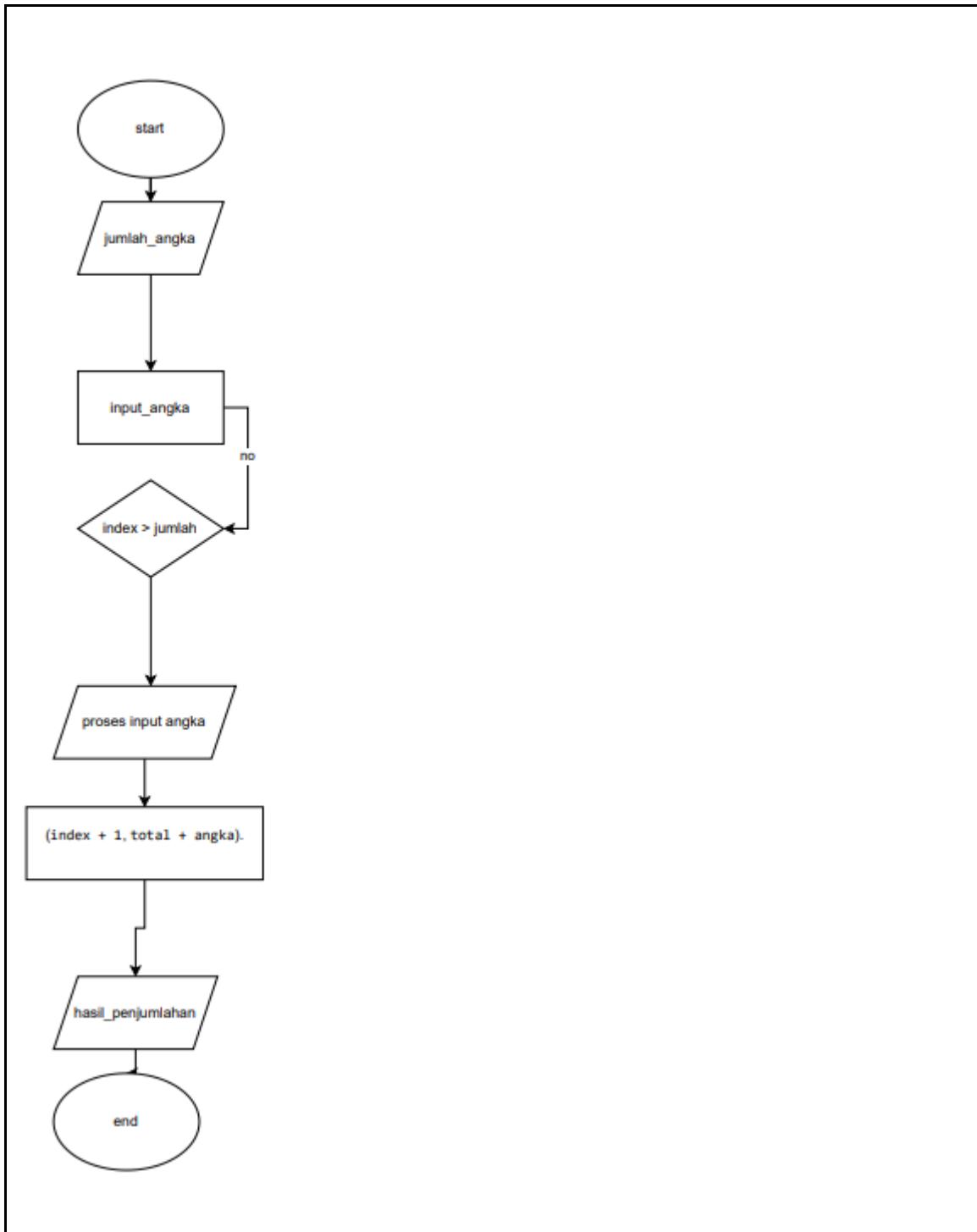
# Input dari user
jumlah_angka = int(input("Masukkan Jumlah: "))

# Memanggil fungsi rekursif dan menampilkan hasil
hasil = penjumlahan_berurut(jumlah_angka)
print("Hasil penjumlahan:", hasil)
```

→ Masukkan Jumlah: 3
Masukkan angka ke-1: 21
Masukkan angka ke-2: 11
Masukkan angka ke-3: 10
Hasil penjumlahan: 42

Flowchart





Source Code



```
def penjumlahan_berurut(jumlah, index=1, total=0):
    if index > jumlah:
        return total
    else:
        angka = int(input(f"Masukkan angka ke-{index}: "))
        return penjumlahan_berurut(jumlah, index + 1, total + angka)

# Input dari user
jumlah_angka = int(input("Masukkan Jumlah: "))

# Memanggil fungsi rekursif dan menampilkan hasil
hasil = penjumlahan_berurut(jumlah_angka)
print("Hasil penjumlahan:", hasil)
```

Output

```
→ Masukkan Jumlah: 3
Masukkan angka ke-1: 21
Masukkan angka ke-2: 11
Masukkan angka ke-3: 10
Hasil penjumlahan: 42
```

b. Latihan Kedua

Buatlah sebuah fungsi perpangkatan menggunakan konsep rekursif (tanpa for/while). User hanya perlu memasukkan base number (angka awal) dan power (pangkatnya). Hasil akhir berupa perhitungan perpangkatannya.

IPO (Input Output Process)



```
✓ ➜ # Fungsi perpangkatan menggunakan rekursi
      def perpangkatan(base, power):
          # Basis kasus: jika power adalah 0, maka hasilnya 1 (karena base^0 = 1)
          if power == 0:
              return 1
          # Rekurens: kalikan base dengan hasil pemanggilan fungsi dengan power - 1
          else:
              return base * perpangkatan(base, power - 1)

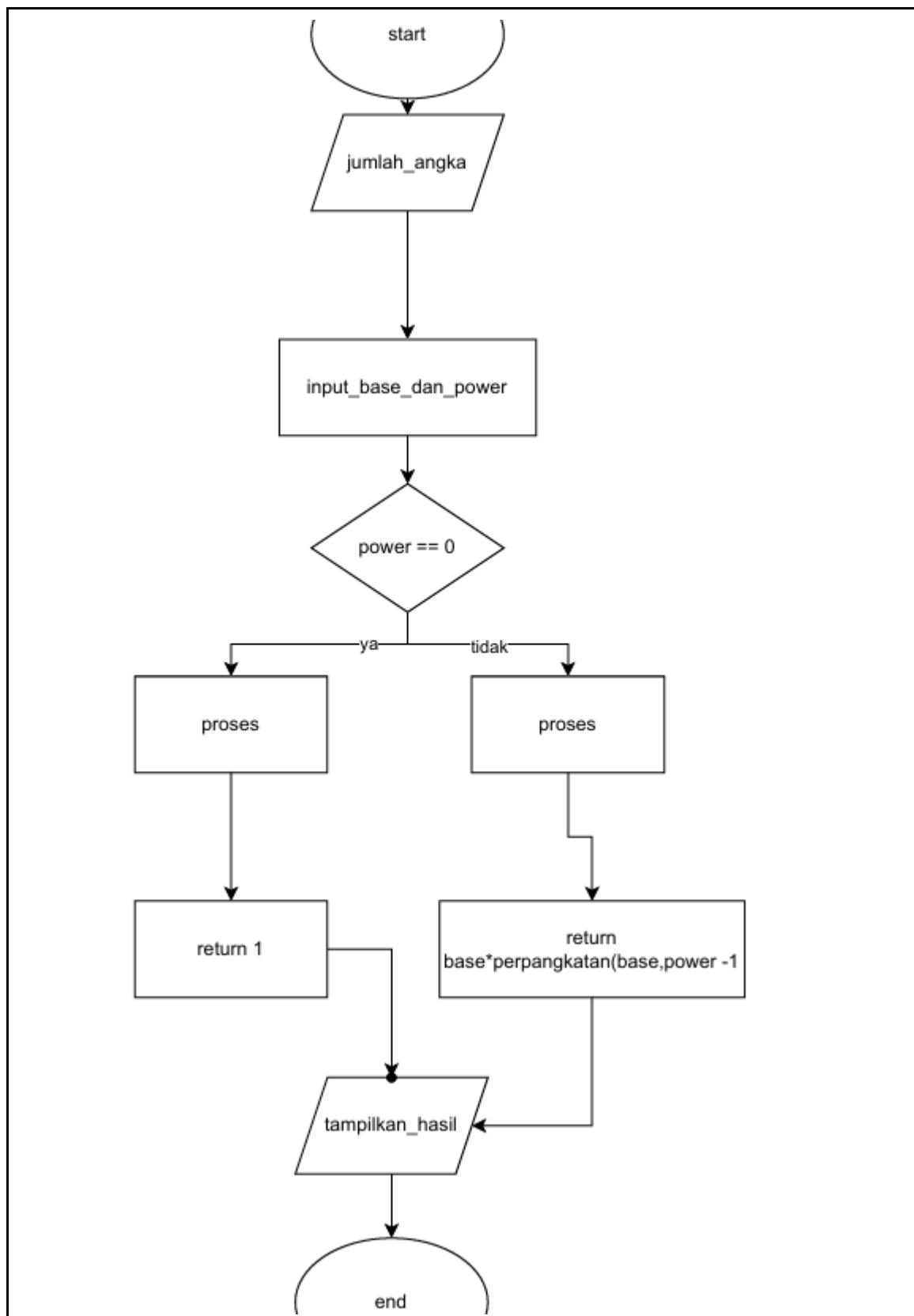
      # Input dari user
      base = int(input("Masukkan base number: "))
      power = int(input("Masukkan power: "))

      # Memanggil fungsi dan menampilkan hasil
      hasil = perpangkatan(base, power)
      print("Hasil perpangkatan:", hasil)

➜ Masukkan base number: 22
Masukkan power: 133
Hasil perpangkatan: 34851104745202970491329713375947821250245367244962618108032101513140780342372601348210418178888836634
```

Flowchart





Source Code

```
# Fungsi perpangkatan menggunakan rekursi
def perpangkatan(base, power):
    # Basis kasus: jika power adalah 0, maka hasilnya 1 (karena base^0 = 1)
    if power == 0:
        return 1
    # Rekurens: kalikan base dengan hasil pemanggilan fungsi dengan power - 1
    else:
        return base * perpangkatan(base, power - 1)

# Input dari user
base = int(input("Masukkan base number: "))
power = int(input("Masukkan power: "))

# Memanggil fungsi dan menampilkan hasil
hasil = perpangkatan(base, power)
print("Hasil perpangkatan:", hasil)
```

Output

```
↳ Masukkan base number: 22
Masukkan power: 133
Hasil perpangkatan: 3485110474520297049132971337594782125024536724496261810803210151314078034237260134821041817888836634
```



c. Latihan Ketiga (Tidak Wajib)

Buatlah 2 buah fungsi konversi ‘hexadecimal ke decimal’ yang mengkonversi string berbentuk hexadecimal ke integer dan ‘decimal ke hexadecimal’ yang mengkonversi integer berbentuk decimal kedalam bentuk hexadecimal. Setiap fungsi akan mengambil 1 nilai untuk dikonversi (sebagai satu-satunya parameter yang dimilikinya) dan me-return nilai yang sudah terkonversi sebagai satu-satunya hasil akhir. Berikan error message jika inputan yang diberikan invalid. Gunakan konsep rekursi untuk mendapat nilai maksimal di soal bonus ini. (lihat Exercise 98)

Source Code

```
# Fungsi rekursif untuk konversi hexadecimal ke decimal
def hex_to_dec(hex_str):
    # Basis kasus: jika panjang string hanya 1 karakter
    if len(hex_str) == 1:
        return int(hex_str, 16) if hex_str.isalnum() else "Input tidak valid"

    # Rekurens: konversi karakter pertama ke decimal dan lanjutkan dengan sisa string
    try:
        return int(hex_str[0], 16) * (16 ** (len(hex_str) - 1)) + hex_to_dec(hex_str[1:])
    except ValueError:
        return "Input tidak valid"

# Contoh penggunaan
hex_str = input("Masukkan string hexadecimal: ")
hasil_dec = hex_to_dec(hex_str)
print("Hasil konversi ke decimal:", hasil_dec)
```

Output



Masukkan string hexadecimal: 233
Hasil konversi ke decimal: 563

4. File Praktikum

Github Repository:

<https://github.com/muhrizki26/praktikum-ke-8.git>

5. Soal Latihan

Soal:

1. Sebutkan dan jelaskan perbedaan secara teknis antara perulangan for/while dengan perulangan fungsi rekursif?
2. Deskripsikan serta narasikan jalannya alur source code program yang sebelumnya telah kalian buat pada Elemen Kompetensi Latihan Kedua!

Jawaban:

1. **for/while:** Perulangan iteratif yang lebih efisien dalam penggunaan memori dan lebih cepat. Cocok untuk tugas berulang sederhana.
Rekursi: Fungsi memanggil dirinya sendiri, memecah masalah jadi sub-masalah. Memerlukan lebih banyak memori (call stack) dan lebih lambat. Cocok untuk masalah seperti pohon biner dan perhitungan factorial.
2. **Input:** Program meminta pengguna untuk memasukkan `base` `number` (angka awal) dan `power` (pangkat).
Pemanggilan Fungsi Rekursif:
Basis Kasus: Jika `power` bernilai 0, fungsi mengembalikan 1, karena aturan matematika menyatakan bahwa $x^0 = 1$.
Rekurens: Jika `power` lebih besar dari 0, fungsi memanggil dirinya sendiri dengan `power - 1` dan mengalikan hasilnya dengan `base`. Ini menghitung `base^power` dengan mengurangi pangkat hingga mencapai 0.



Proses Penghitungan:

Setiap pemanggilan fungsi memecah masalah menjadi sub-masalah yang lebih kecil, misalnya menghitung $\text{base} * (\text{base} * (\text{base} * \dots))$ hingga power menjadi 0.

Output: Setelah proses rekursif selesai, hasil perpangkatan dikembalikan dan ditampilkan ke layar.

Kesimpulan

- Dalam pengerjaan 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 car akita menyelesaikan suatu tugas yaitu pemfaktorialan dan juga fungsi perpangkatan selain itu juga kita mengerjakan tugas hexa decimal dimana kita mengubah sesuatu fungsi dari hexadecimal ke decimal laporan ini banyak pelajaran baru dimana kita harus koding tanpa menggunakan for atau while dalam menjalankan atau memmbuat sebuah program

6. Cek List (✓)

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

7. Formulir Umpam Balik

No	Elemen Kompetensi	Waktu Pengerjaan	Kriteria
1.	Latihan Pertama	20 Menit	baik
2.	Latihan Kedua	25 Menit	Cukup

Keterangan:

- Menarik
- Baik
- Cukup
- Kurang

