

MODUL 2

PYTHON FUNGTIONS & MODULES

Python:

https://www.w3schools.com/Python/python_functions.asp

<https://www.w3schools.in/python/functions.asp>

Python

<https://www.w3schools.in/python-tutorial/modules/>

https://www.w3schools.com/Python/python_modules.asp

1. Fungtion

Di Python, fungsi (atau "function") adalah blok kode yang dirancang untuk melakukan tugas tertentu dan dapat dipanggil di berbagai bagian program. Fungsi membantu dalam membuat kode lebih modular, dapat digunakan kembali, dan lebih mudah untuk dipelihara. Berikut adalah penjelasan dasar tentang bagaimana mendefinisikan dan menggunakan fungsi di Python.

Mendefinisikan Fungsi

Untuk mendefinisikan fungsi di Python, Anda menggunakan kata kunci `def`, diikuti dengan nama fungsi dan tanda kurung yang mungkin berisi parameter. Setelah itu, Anda menuliskan blok kode di dalam indentasi.

```
def nama_fungsi(parameter1, parameter2):  
    # Blok kode  
    hasil = parameter1 + parameter2  
    return hasil
```

Contoh Fungsi Sederhana

Berikut adalah contoh fungsi sederhana yang menjumlahkan dua angka:

```
def tambah(x, y):  
    hasil = x + y  
    return hasil
```

Untuk memanggil fungsi ini dan mendapatkan hasilnya:

```
hasil_tambah = tambah(5, 3)  
print(hasil_tambah) # Output: 8
```

Contoh Fungsi dengan Parameter Default

Anda juga bisa memberikan nilai default pada parameter fungsi. Jika argumen tidak diberikan saat pemanggilan, nilai default akan digunakan.

```
def sapa(nama="Dunia"):
    print(f"Halo, {nama}!")

sapa()          # Output: Halo, Dunia!
sapa("Alice")   # Output: Halo, Alice!
```

Contoh Fungsi dengan Banyak Nilai Kembali

Fungsi di Python dapat mengembalikan lebih dari satu nilai dengan menggunakan tuple.

Dalam Python, tuple adalah tipe data yang digunakan untuk menyimpan beberapa nilai dalam satu variabel. Tuple sangat mirip dengan list, tetapi dengan perbedaan utama yaitu tuple bersifat **immutable**, yang berarti sekali diciptakan, elemennya tidak bisa diubah.

Tuple dibuat dengan menggunakan tanda kurung () dan memisahkan elemen-elemen dengan koma. Berikut adalah beberapa contoh:

```
# Tuple dengan beberapa elemen
my_tuple = (1, 2, 3, 'hello', 5.6)

# Tuple dengan satu elemen (harus diakhiri dengan koma)
single_element_tuple = (42,)
```

Contoh Mengakses Elemen Tuple

Elemen dalam tuple dapat diakses dengan menggunakan indeks, yang dimulai dari 0:

```
print(my_tuple[0]) # Output: 1
print(my_tuple[3]) # Output: hello
```

Contoh Menggunakan Tuple dalam Fungsi

Tuple sering digunakan untuk mengembalikan beberapa nilai dari sebuah fungsi:

```
def get_user_info():  
    name = "Alice"  
    age = 30  
    return (name, age)  
  
user_info = get_user_info()  
print(user_info) # Output: ('Alice', 30)
```

2. Modules

Dalam pemrograman Python, **modul** adalah file yang berisi kode Python, termasuk definisi fungsi, kelas, dan variabel, serta dapat juga menyimpan kode yang dapat dieksekusi. Modul membantu dalam mengorganisir kode dan memungkinkan penggunaan kembali.

Berikut adalah penjelasan tentang modul Python, cara membuatnya, dan cara menggunakannya:

1. Membuat Modul

Modul sederhana yang bernama `math_operations.py`. Modul ini akan berisi beberapa fungsi matematika dasar seperti penjumlahan, pengurangan, perkalian, dan pembagian.

Langkah 1: Buat file `math_operations.py`

Buka editor teks atau IDE favorit Anda, lalu buat file baru bernama `math_operations.py` dan tambahkan kode berikut:

```
# math_operations.py

def add(a, b):
    """Mengembalikan hasil penjumlahan a dan b."""
    return a + b

def subtract(a, b):
    """Mengembalikan hasil pengurangan a dengan b."""
    return a - b

def multiply(a, b):
    """Mengembalikan hasil perkalian a dan b."""
    return a * b

def divide(a, b):
    """Mengembalikan hasil pembagian a dengan b. Menangani pembagian dengan nol.
    if b == 0:
        return "Error: Pembagian dengan nol!"
    return a / b
```

2. Menggunakan Modul

Sekarang kita akan menggunakan modul `math_operations` yang telah kita buat di program lain.

Langkah 2: Buat file `main.py`

Buat file baru bernama `main.py` di direktori yang sama dengan `math_operations.py`, lalu tambahkan kode berikut:

```
# main.py

# Mengimpor modul yang telah dibuat
import math_operations

# Menggunakan fungsi dari modul
a = 10
b = 5

print(f"{a} + {b} = {math_operations.add(a, b)}")
print(f"{a} - {b} = {math_operations.subtract(a, b)}")
print(f"{a} * {b} = {math_operations.multiply(a, b)}")
print(f"{a} / {b} = {math_operations.divide(a, b)}")

# Contoh pembagian dengan nol
print(f"{a} / 0 = {math_operations.divide(a, 0)}")
```

3. Menjalankan Program

Langkah 3: Jalankan program `main.py`

Pastikan Anda berada di direktori yang sama dengan kedua file, lalu jalankan `main.py` menggunakan terminal atau command prompt:

```
[user@usernoMacBook-Pro program modules % python3 main.py
10 + 5 = 15
10 - 5 = 5
10 * 5 = 50
10 / 5 = 2.0
10 / 0 = Error: Pembagian dengan nol!
```

Studi Kasus: Modul Manajemen Pelanggan

Tujuan

Membuat modul bernama `customer_manager.py` yang dapat digunakan untuk:

1. Menambahkan pelanggan baru.
2. Menghapus pelanggan berdasarkan ID.
3. Mencari pelanggan berdasarkan nama.
4. Menampilkan semua pelanggan.

Langkah 1: Membuat Modul `customer_manager.py`

1. **Buat file bernama `customer_manager.py`.**

Di dalam file ini, kita akan mendefinisikan fungsi-fungsi yang diperlukan untuk mengelola data pelanggan.

```
# customer_manager.py

customers = []

def add_customer(name, age):
    """Menambahkan pelanggan baru ke daftar."""
    customer_id = len(customers) + 1
    customer = {'id': customer_id, 'name': name, 'age': age}
    customers.append(customer)
    return customer

def remove_customer(customer_id):
    """Menghapus pelanggan dari daftar berdasarkan ID."""
    global customers
    customers = [c for c in customers if c['id'] != customer_id]

def find_customer(name):
    """Mencari pelanggan berdasarkan nama."""
    return [c for c in customers if name.lower() in c['name'].lower()]

def list_customers():
    """Menampilkan semua pelanggan."""
    return customers
```

Langkah 2: Menggunakan Modul dalam `main.py`

Buat file bernama `main.py`.

Di dalam file ini, kita akan mengimpor modul `customer_manager` dan memberikan antarmuka sederhana untuk berinteraksi dengan pengguna.

```

import customer_manager

def main():
    while True:
        print("\nMenu:")
        print("1. Tambah Pelanggan")
        print("2. Hapus Pelanggan")
        print("3. Cari Pelanggan")
        print("4. Tampilkan Semua Pelanggan")
        print("5. Keluar")

        choice = input("Pilih opsi (1-5): ")

        if choice == '1':
            name = input("Masukkan nama pelanggan: ")
            age = int(input("Masukkan usia pelanggan: "))
            customer = customer_manager.add_customer(name, age)
            print(f"Pelanggan ditambahkan: {customer}")

        elif choice == '2':
            customer_id = int(input("Masukkan ID pelanggan yang ingin dihapus: "))
            customer_manager.remove_customer(customer_id)
            print(f"Pelanggan dengan ID {customer_id} telah dihapus.")

        elif choice == '3':
            name = input("Masukkan nama pelanggan yang dicari: ")
            results = customer_manager.find_customer(name)
            if results:
                print("Pelanggan ditemukan:")
                for c in results:
                    print(c)
            else:
                print("Pelanggan tidak ditemukan.")

        elif choice == '4':
            customers = customer_manager.list_customers()
            if customers:
                print("Daftar Pelanggan:")
                for c in customers:
                    print(c)
            else:
                print("Tidak ada pelanggan.")

        elif choice == '5':
            print("Keluar dari program.")
            break

        else:
            print("Opsi tidak valid. Silakan coba lagi.")

if __name__ == "__main__":
    main()

```

Langkah 3: Menjalankan Program

3. Jalankan `main.py`.

Pastikan kedua file (`customer_manager.py` dan `main.py`) berada dalam direktori yang sama. Kemudian buka terminal atau command prompt, dan jalankan perintah berikut:

Kesimpulan

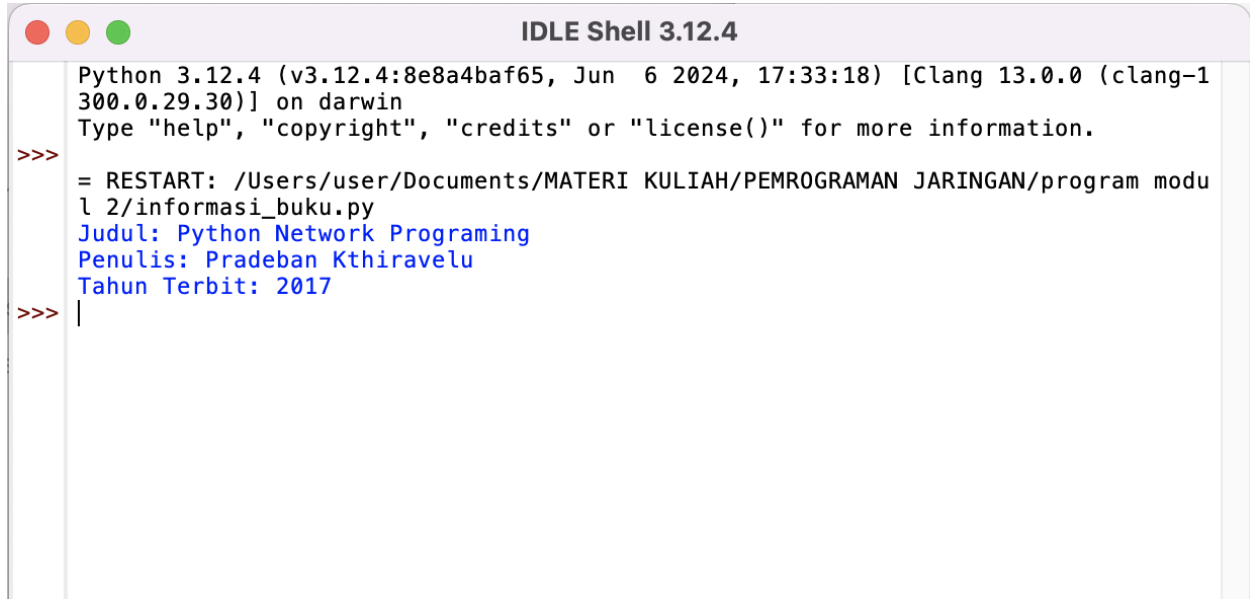
Tuple adalah struktur data yang berguna ketika Anda membutuhkan kumpulan nilai yang tidak akan berubah. Mereka menyediakan cara yang efisien untuk mengelompokkan data yang bersifat tetap dan dapat digunakan dalam berbagai konteks di Python.

Fungsi di Python adalah alat yang sangat penting dalam pemrograman, memungkinkan Anda untuk menulis kode yang lebih bersih dan terorganisir. Anda dapat memanfaatkan berbagai fitur fungsi untuk membuat kode yang lebih efisien dan mudah dipahami.

Dengan membuat dapat mengorganisir kode Anda ke dalam modul-modul terpisah yang membuatnya lebih mudah untuk dikelola dan digunakan kembali.

SOAL MODUL 2

1. Menggunakan Tuple untuk mengelompokkan data, Buatlah Fungsi yang mengelompokkan informasi tentang buku seperti berikut



```
Python 3.12.4 (v3.12.4:8e8a4baf65, Jun 6 2024, 17:33:18) [Clang 13.0.0 (clang-1300.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: /Users/user/Documents/MATERI KULIAH/PEMROGRAMAN JARINGAN/program modul 2/informasi_buku.py
Judul: Python Network Programing
Penulis: Pradeban Kthiravelu
Tahun Terbit: 2017
>>> |
```

2. Buat program untuk manajemen data mahasiswa menggunakan modul.