Panduan Pembelajaran Pertemuan 1: Review Algoritma dan Pemrograman Python

Tujuan Pembelajaran

- Mengembangkan pengetahuan tentang berbagai paradigma pemrograman untuk diterapkan sesuai kebutuhan.
- Memahami konsep fungsi, kelas, modularisasi program, algoritma, dan paradigma pemrograman.
- Mengimplementasikan konsep algoritma dan OOP dalam Python.
- Meningkatkan keterampilan analisis, penyelesaian masalah, dan logika berpikir melalui pemrograman.

Topik Pembelajaran

1. Algoritma dan Paradigma Pemrograman:
- Pengertian dan karakteristik algoritma.
- Paradigma pemrograman: Imperatif, Deklaratif, Berorientasi Objek, dan Fungsional.
- Skema dasar algoritma: inisialisasi, proses, dan output.

2. Fungsi (Functions):
- Konsep dasar fungsi: definisi, parameter, dan pengembalian nilai.
- Manfaat fungsi untuk modularisasi kode dan meningkatkan efisiensi pemrograman.

3. Kelas (Class) dalam Pemrograman Berorientasi Objek (OOP):
- Konsep dasar kelas dan objek.
- Atribut (properties) dan metode (methods) dalam kelas.
- Implementasi enkapsulasi, pewarisan, dan polimorfisme.

4. Modularisasi Program:
- Pentingnya memecah program besar menjadi modul yang lebih kecil.
- Penggunaan pustaka Python dan cara membuat modul sendiri.

Referensi Praktikum

- Python Functions: https://www.w3schools.com/python/python_functions.asp
- Python Classes: https://www.w3schools.com/python/python_classes.asp
- Python Modules: https://www.w3schools.com/python/python_modules.asp

MATERI

Pengertian Algoritma

Algoritma adalah langkah-langkah sistematis dan logis yang digunakan untuk menyelesaikan suatu masalah atau mencapai tujuan tertentu. Algoritma terdiri dari serangkaian instruksi yang dirancang agar dapat dijalankan oleh komputer untuk menghasilkan output berdasarkan input tertentu.

Ciri-ciri algoritma yang baik:

1. Finiteness: Algoritma harus memiliki akhir.
2. Definiteness: Setiap langkah algoritma harus jelas dan tidak ambigu.
3. Input dan Output: Algoritma menerima input dan menghasilkan output.
4. Efisiensi: Algoritma dirancang untuk menyelesaikan masalah dengan penggunaan sumber daya minimal.

Paradigma Pemrograman

Paradigma pemrograman adalah pendekatan atau cara berpikir dalam menyusun dan mengorganisir kode program. Beberapa paradigma utama:

1. Pemrograman Imperatif: Berfokus pada langkah-langkah untuk mengubah keadaan sistem. Contoh: Python, C.
2. Pemrograman Deklaratif: Berfokus pada apa yang harus dilakukan tanpa mendeskripsikan cara melakukannya. Contoh: SQL, Prolog.
3. Pemrograman Berorientasi Objek (OOP): Mengorganisasi program ke dalam objek yang memiliki data (atribut) dan fungsi (metode). Contoh: Java, Python.
4. Pemrograman Fungsional: Berfokus pada fungsi matematika murni dan penghindaran perubahan keadaan. Contoh: Haskell, Lisp.

Skema Dasar Algoritma

Skema dasar algoritma biasanya terdiri dari tiga bagian utama:

1. Inisialisasi : Langkah awal untuk mendefinisikan nilai awal, variabel, atau input.
2. Proses : Serangkaian instruksi atau operasi yang dilakukan untuk mengolah data.
3. Output : Hasil dari proses yang telah dilakukan.

Contoh skema dasar algoritma:

1. Masukkan nilai A dan B.
2. Hitung jumlah dari A dan B.
3. Tampilkan hasil penjumlahan.

Dalam bentuk Python:

```
def penjumlahan(a, b):
  return a + b

print(penjumlahan(3, 4))  # Output: 7
```

Penugasan #1

Tugas Individu

1. Buat algoritma sederhana yang melibatkan perhitungan aritmetika, lebih **bagus** dari contoh dibawah ini.
2. Tuliskan algoritma tersebut dalam bentuk pseudocode.
3. Implementasikan algoritma tersebut dalam program Python.
4. Contoh pseudocode Menghitung Rataan Nilai Ujian:
    1. Masukkan jumlah data (n).
    2. Masukkan nilai-nilai data.
    3. Hitung jumlah total dari nilai-nilai tersebut.
    4. Hitung rata-rata dengan membagi total nilai dengan n.
    5. Tampilkan hasil rata-rata.

- Contoh implementasi Python:

```
def hitung_rata_rata(nilai):
  return sum(nilai) / len(nilai)

data = [80, 90, 70, 85]
print("Rata-rata:", hitung_rata_rata(data))  # Output: Rata-rata: 81.25
```

5. **Pelaporan:**
   Laporan disusun dalam dokumen format PDF, memuat:

    - Pseudocode (jika relevan).
    - Kode program Python.
    - Screenshot kode dan hasil eksekusi.
    - Narasi langkah pengerjaan dan penjelasan fungsi setiap bagian kode.

Fungsi

“Fungsi adalah blok kode yang dirancang untuk melakukan tugas tertentu “

Fungsi adalah blok kode yang dirancang untuk melakukan tugas tertentu dan dapat dipanggil berkali-kali dalam program. Fungsi membantu meningkatkan efisiensi dan keterbacaan kode dengan menghindari pengulangan logika yang sama di berbagai bagian program.

Fungsi biasanya menerima input berupa parameter, melakukan operasi berdasarkan logika yang telah ditentukan, dan dapat mengembalikan hasil sebagai output. Dengan memisahkan logika tertentu ke dalam fungsi, program menjadi lebih modular dan lebih mudah untuk di-debug, diperbaiki, atau diperluas.

Sebagai contoh, sebuah fungsi sederhana untuk menghitung luas persegi panjang:

```
def hitung_luas(panjang, lebar):
    return panjang * lebar
```

```
# Memanggil fungsi
luas = hitung_luas(5, 3)
print(f"Luas: {luas}")  # Output: Luas: 15
```

Fungsi ini dapat dipanggil kapan saja dengan berbagai nilai panjang dan lebar tanpa perlu menulis ulang logika perhitungan.

Struktur dasar fungsi di Python:

```
def nama_fungsi(parameter):
  # kode
  return nilai
```

Contoh:

```
def penjumlahan(a, b):
  return a + b
print(penjumlahan(3, 4))  # Output: 7
```

Kelas/Class dan Objek

"Kelas adalah cetak biru/blueprint untuk membuat objek, sedangkan objek adalah instance dari kelas. tuliskan disini saja".

Kelas adalah cetak biru (*blueprint*) untuk membuat objek, yang berfungsi sebagai template atau kerangka kerja dalam mendefinisikan atribut (properti) dan metode (fungsi) yang dimiliki oleh suatu entitas. Atribut digunakan untuk menyimpan data, sedangkan metode digunakan untuk mendefinisikan perilaku.

Objek, di sisi lain, adalah realisasi konkret dari kelas. Setiap objek yang dibuat dari sebuah kelas memiliki data dan perilaku yang telah didefinisikan dalam kelas tersebut, tetapi dengan nilai atribut yang bisa berbeda-beda.

Sebagai ilustrasi, kelas dapat diibaratkan sebagai rencana arsitektur sebuah rumah, sementara objek adalah rumah yang nyata dibangun berdasarkan rencana tersebut. Dengan menggunakan satu rencana, banyak rumah (objek) dapat dibangun dengan variasi tertentu, seperti warna cat atau tipe atap.

Contoh implementasi kelas:

```
class Kendaraan:
  def __init__(self, jenis, merk, warna):
    self.jenis = jenis
    self.merk = merk
    self.warna = warna

  def info(self):
    return f"{self.warna} {self.merk} {self.jenis}"
```

Objek dapat dibuat seperti ini:

```
mobil = Kendaraan("Mobil", "Toyota", "Merah")
print(mobil.info())  # Output: Merah Toyota Mobil
```

Modularisasi

"Modularisasi membantu memecah program besar menjadi bagian kecil yang lebih terorganisir "

Modularisasi membantu memecah program besar menjadi bagian-bagian kecil yang lebih terorganisir, yang disebut modul. Setiap modul berfungsi sebagai unit terpisah yang memiliki tugas atau fungsi spesifik. Pendekatan ini mempermudah pengelolaan, pemahaman, dan pengujian kode karena setiap bagian program dapat dikembangkan, diperbaiki, atau diperbarui secara mandiri tanpa memengaruhi modul lainnya.

Sebagai contoh, dalam proyek besar, seperti sistem e-commerce, modul dapat berupa sistem pembayaran, pengelolaan produk, atau autentikasi pengguna. Dengan modularisasi, pengembang dapat bekerja secara paralel pada modul yang berbeda, meningkatkan efisiensi tim. Selain itu, kode yang modular juga memungkinkan penggunaan ulang (reusability), di mana modul tertentu dapat diterapkan kembali dalam proyek lain tanpa perlu menulis ulang kode.

Contoh:

File modul (`modulku.py`):

```
def sapa(nama):
   return f"Halo, {nama}!"
```

Penggunaan modul:

```
import modulku
print(modulku.sapa("Alpro"))
```

Penugasan #2

Tugas Individu:

Penugasan #1: Materi Fungsi

1. **Tugas:**

- Buat fungsi Python yang lebih bagus dari contoh dibawah ini :\Contoh fungsi Menghitung rata-rata dari sekumpulan nilai dan Menentukan apakah suatu nilai adalah bilangan prima.

```
def rata_rata(nilai):
    return sum(nilai) / len(nilai)

def cek_prima(angka):
    if angka < 2:
        return False
    for i in range(2, int(angka**0.5) + 1):
```

```
        if angka % i == 0:
            return False
    return True
```

2. **Pelaporan:**
- Screenshot kode fungsi dan hasil eksekusi dengan beberapa nilai uji.
- Narasi menjelaskan langkah pengerjaan dan fungsi setiap bagian kode.

Penugasan #2: Materi Class

2. **Tugas:**
1. Buat dua kelas Python berbeda, masing-masing dengan:
2. Minimal **5 atribut** dan **3 metode**.
3. Contoh kelas:

```
class Mahasiswa:
    def __init__(self, nama, nim, prodi, angkatan, ipk):
        self.nama = nama
        self.nim = nim
        self.prodi = prodi
        self.angkatan = angkatan
        self.ipk = ipk

    def tampilkan_info(self):
        return f"{self.nama} ({self.nim}) dari {self.prodi},
angkatan {self.angkatan}."

    def ubah_ipk(self, ipk_baru):
        self.ipk = ipk_baru

    def cetak_ipk(self):
        return f"IPK saat ini: {self.ipk}"
```

3. T**ugas tambahan:**
- Buat objek dari setiap kelas.
- Jalankan semua metode dari objek yang dibuat.
4. **Pelaporan:**
- Screenshot kode kelas, objek, dan hasil eksekusi.
- Narasi menjelaskan proses pembuatan kelas, atribut, metode, dan implementasinya dalam objek.

Penugasan #3: Materi Modul

1. Tugas:
- Buat sebuah modul Python (modulku.py) yang berisi fungsi-fungsi lebih baik dari contoh berikut:
    - Menghitung luas lingkaran.
    - Menghitung luas persegi panjang.
    - Mengubah suhu dari Celsius ke Fahrenheit.

- Contoh isi file modulku.py:

```
def luas_lingkaran(jari_jari):
    return 3.14 * jari_jari**2

def luas_persegi_panjang(panjang, lebar):
    return panjang * lebar

def celsius_ke_fahrenheit(celsius):
    return (celsius * 9/5) + 32
```

- Beri nama modulNIM.py (NIM sesuaikan masing-masing)

2. **Tugas tambahan:**

- Buat file main.py untuk mengimpor modul dan menggunakan semua fungsi di dalamnya dengan beberapa nilai uji.
- Contoh penggunaan di main.py:

```
import modulku

print("Luas lingkaran:", modulku.luas_lingkaran(7))
print("Luas persegi panjang:", modulku.luas_persegi_panjang(5, 3))
print("Suhu dalam Fahrenheit:", modulku.celsius_ke_fahrenheit(25))
```

3. **Pelaporan:**

- Screenshot isi modulku.py dan main.py.
- Screenshot hasil eksekusi program main.py.
- Narasi menjelaskan cara membuat modul dan menggunakan fungsi-fungsi di dalamnya.

Format Laporan

- Laporan disusun dalam satu dokumen format PDF, memuat:
  - Pseudocode (jika relevan).
  - Kode program Python.
  - Screenshot kode dan hasil eksekusi.
  - Narasi langkah pengerjaan dan penjelasan fungsi setiap bagian kode.