

Pendahuluan

Object dan **Class** merupakan bagian dari **OOP**. **Class** digunakan untuk mendefinisikan atribut dan metode yang akan dimiliki oleh **Object** yang akan dibuat dari **Class** tersebut. **Object**, di sisi lain, adalah instance dari kelas, ia merupakan entitas yang diciptakan berdasarkan struktur yang didefinisikan dalam **Class**.

Tujuan dan Manfaat

Tujuan

1. **Memahami Konsep Dasar OOP:** Memahami konsep dasar pemrograman berorientasi objek (OOP), termasuk **Object** dan **Class**, serta bagaimana konsep-konsep ini diimplementasikan dalam Python.
2. **Menguasai Pembuatan Class dan Object:** Menguasai cara membuat dan menggunakan kelas dan objek dalam Python untuk mengembangkan program yang modular dan terstruktur.

Manfaat

1. **Penggunaan Struktur Kode yang Lebih Baik:** Membantu dalam menciptakan kode yang lebih terstruktur dan terorganisir, sehingga lebih mudah untuk dipelihara dan diperbarui.
2. **Kode yang Dapat Digunakan Kembali:** Dengan menggunakan konsep **Class**, kode dapat digunakan kembali di berbagai bagian program atau proyek lain, meningkatkan efisiensi pengembangan.
3. **Pemecahan Masalah yang Efisien:** Meningkatkan kemampuan dalam menganalisis dan memecahkan masalah dengan pendekatan OOP, yang sering kali lebih efisien dibandingkan metode pemrograman prosedural.
4. **Meningkatkan Keterampilan Pemrograman:** Mengembangkan keterampilan pemrograman yang lebih mendalam, yang bermanfaat untuk karier di bidang pengembangan perangkat lunak.
5. **Memahami Lebih Mendalam tentang Python:** Memperdalam pemahaman tentang bahasa Python dan cara kerjanya, khususnya dalam konteks OOP, yang merupakan salah satu paradigma pemrograman utama yang didukung oleh Python.

Program

1.

The screenshot shows a Python IDE with two windows. The left window displays the source code for a program. The right window shows the output of the program after execution.

```
class Employee:
    'Common base class For all employees'
    empCount = 0
    def __init__(self, name, salary):
        self.name = name
        self.salary = salary
        Employee.empCount += 1
    def displayCount(self):
        print("Total Employee %d" % Employee.empCount)
    def displayEmployee(self):
        print("Name :", self.name, ", Salary :", self.salary)

#This would create first object Employee class
emp1 = Employee("Zara", 2000)
#This would create second object Employee class
emp2 = Employee("Manni", 5000)
emp1.displayEmployee()
emp2.displayEmployee()
print("Total Employee %d" % Employee.empCount)
print("")

class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

    def myfunc(self):
        print("Hello nama saya :", self.name)
        print("Umur saya :", self.age)

p1 = Person("Budi", 19)
p1.myfunc()
```

The output window shows the following results:

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: D:\pemula\Python\Tugas Prak 11 no 1.py =====
Name : Zara , Salary : 2000
Name : Manni , Salary : 5000
Total Employee 2

Hello nama saya : Budi
Umur saya : 19
>>>
```

2A.

The screenshot shows a Python IDE with two windows. The left window displays the source code for a program. The right window shows the output of the program after execution.

```
class Person:
    person = 'Person'

    def __init__(self, rambut, warna):
        self.rambut = rambut
        self.warna = warna

Budi = Person("Ikal", "Hitam")
Michael = Person("Lurus", "Pirang")

print('Budi details:')
print('Rambut:', Budi.rambut)
print('Warna:', Budi.warna)

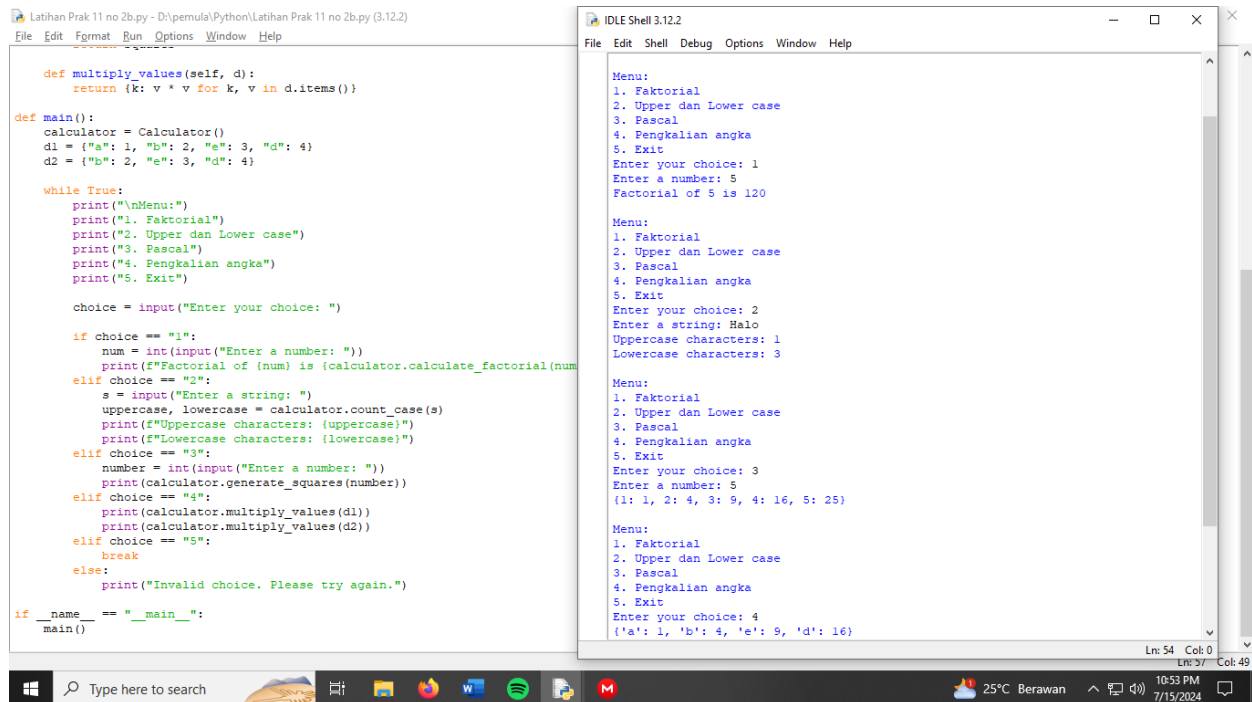
print('\nMichael details:')
print('Rambut:', Michael.rambut)
print('Warna:', Michael.warna)
```

The output window shows the following results:

```
Python 3.12.2 (tags/v3.12.2:6abddd9, Feb 6 2024, 21:26:36) [MSC v.1937 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
= RESTART: D:\pemula\Python\Tugas Prak 11 no 2.py
Budi details:
Rambut: Ikal
Warna: Hitam

Michael details:
Rambut:urus
Warna: Pirang
>>>
```

2B.



The screenshot displays a Python IDE with two windows. The left window, titled 'Latihan Prak 11 no 2b.py', contains the source code for a calculator program. The right window, titled 'IDLE Shell 3.12.2', shows the program's execution output.

```
def multiply_values(self, d):
    return {k: v * v for k, v in d.items()}

def main():
    calculator = Calculator()
    d1 = {"a": 1, "b": 2, "e": 3, "d": 4}
    d2 = {"b": 2, "e": 3, "d": 4}

    while True:
        print("\nMenu:")
        print("1. Faktorial")
        print("2. Upper dan Lower case")
        print("3. Pascal")
        print("4. Pengkalian angka")
        print("5. Exit")

        choice = input("Enter your choice: ")

        if choice == "1":
            num = int(input("Enter a number: "))
            print(f"Factorial of (num) is {calculator.calculate_factorial(num)}")
        elif choice == "2":
            s = input("Enter a string: ")
            uppercase, lowercase = calculator.count_case(s)
            print(f"Uppercase characters: {uppercase}")
            print(f"Lowercase characters: {lowercase}")
        elif choice == "3":
            number = int(input("Enter a number: "))
            print(calculator.generate_squares(number))
        elif choice == "4":
            print(calculator.multiply_values(d1))
            print(calculator.multiply_values(d2))
        elif choice == "5":
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    main()
```

The shell output shows the program's execution flow:

```
Menu:
1. Faktorial
2. Upper dan Lower case
3. Pascal
4. Pengkalian angka
5. Exit
Enter your choice: 1
Enter a number: 5
Factorial of 5 is 120

Menu:
1. Faktorial
2. Upper dan Lower case
3. Pascal
4. Pengkalian angka
5. Exit
Enter your choice: 2
Enter a string: Halo
Uppercase characters: 1
Lowercase characters: 3

Menu:
1. Faktorial
2. Upper dan Lower case
3. Pascal
4. Pengkalian angka
5. Exit
Enter your choice: 3
Enter a number: 5
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

Menu:
1. Faktorial
2. Upper dan Lower case
3. Pascal
4. Pengkalian angka
5. Exit
Enter your choice: 4
{'a': 1, 'b': 4, 'e': 9, 'd': 16}
```

Analisis program

Program ini menunjukkan bagaimana konsep **Object** dan **Class** dapat digunakan untuk membangun sistem yang modular dan terorganisir. Dengan menggunakan OOP, program menjadi lebih mudah dipelihara dan diperluas. Penggunaan enkapsulasi, dan potensi untuk pewarisan dan polymorphism, menunjukkan fleksibilitas dan kekuatan pendekatan OOP dalam penggunaan di python.

Referensi

<https://www.w3schools.com/python/>