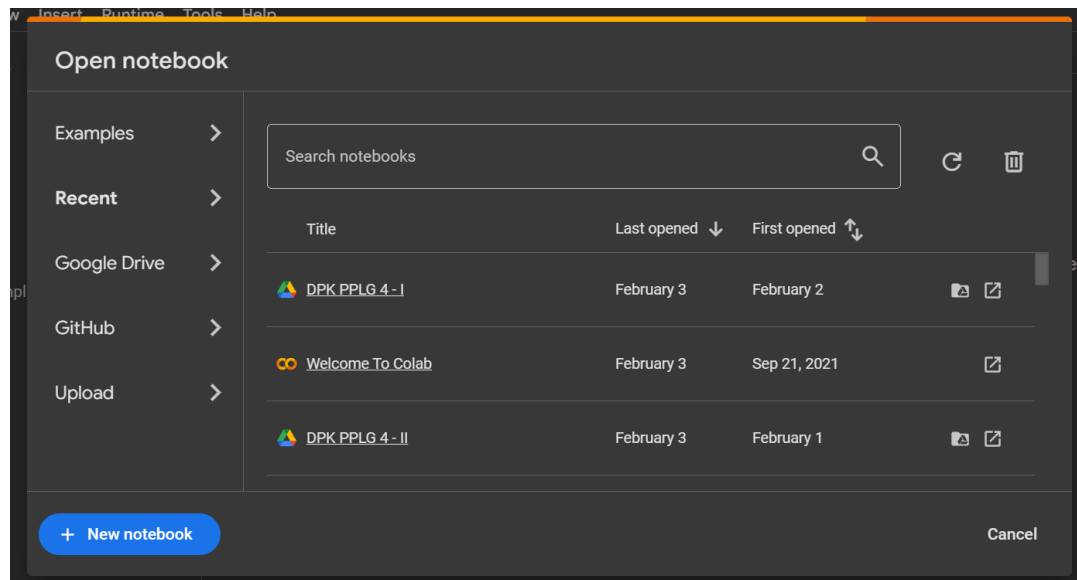
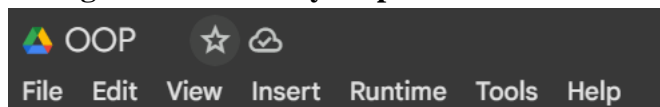


## Praktikum OOP / PBO Python

1. Buka website google colab (<https://colab.research.google.com/>)
2. Pilih New Notebook



3. Lalu ganti nama file nya seperti ini



4. Pilih Code untuk menulis kode, pilih Text untuk menambahkan catatan



5. Membuat Class dan Object

```
# Membuat Class dan Object
class Siswa:
    def __init__(self, nama, nsn): # Constructor (__init__)
        self.nama = nama # Public attribute
        self.nsn = nsn

    def tampilkan_info(self):
        return f>Nama: {self.nama}, NISN: {self.nsn}"

# Membuat objek
siswa1 = Siswa("Andi", "12345")
print(siswa1.tampilkan_info())
```

Output :

```
➦ Nama: Andi, NISN: 12345
```

## 6. Encapsulation (Enkapsulasi)

Klik lagi toolbar code

+ Code + Text

Lalu tuliskan kode ini didalamnya

```
[18] # Encapsulation (Enkapsulasi)
class AkunBank:
    def __init__(self, saldo):
        self.__saldo = saldo # Private attribute

    def tambah_saldo(self, jumlah):
        if jumlah > 0:
            self.__saldo += jumlah

    def lihat_saldo(self):
        return self.__saldo

akun = AkunBank(1000000)
akun.tambah_saldo(500000)
print("Saldo: ", akun.lihat_saldo())
```

Output:

```
➦ Saldo: 1500000
```

## 7. Inheritance (Pewarisan)

Klik lagi toolbar code

+ Code + Text

Lalu tuliskan kode ini didalamnya

```
# Inheritance (Pewarisan)
class Pegawai:
    def __init__(self, nama, gaji):
        self.nama = nama
        self.gaji = gaji

    def info(self):
        return f>Nama: {self.nama}, Gaji: {self.gaji}"

# Subclass (turunan)
class Manager(Pegawai):
    def __init__(self, nama, gaji, bonus):
        super().__init__(nama, gaji) # Memanggil konstruktor superclass
        self.bonus = bonus

    def info(self): # Override method
        return f>Nama: {self.nama}, Gaji: {self.gaji}, Bonus: {self.bonus}"

mgr = Manager("Budi", 10000000, 2000000)
print(mgr.info())
```

Output:

```
➤ Nama: Budi, Gaji: 10000000, Bonus: 2000000
```

## 8. Polymorphism (Polimorfisme)

Klik lagi toolbar code

+ Code + Text

Lalu tuliskan kode ini didalamnya

```
[16] # Polymorphism (Polimorfisme)
      def cetak_info(obj):
          print(obj.info())

      pegawai = Pegawai("Citra", 5000000)
      cetak_info(pegawai)
      cetak_info(mgr) # Manager juga bisa diproses dengan fungsi yang sama
```

Output:

```
➤ Nama: Citra, Gaji: 5000000
  Nama: Budi, Gaji: 10000000, Bonus: 2000000
```

## 9. Method Overloading menggunakan \*args dan \*\*kwargs

Klik lagi toolbar code

+ Code + Text

Lalu tuliskan kode program ini

```
▶ # Method Overloading menggunakan *args dan **kwargs
class Hitung:
    def tambah(self, *args):
        return sum(args)

hitung = Hitung()
print("Hasil Penjumlahan:", hitung.tambah(5, 10, 15))
```

Output:

```
➤ Hasil Penjumlahan: 30
```

## 10. Abstraction (Abstraksi)

Klik lagi toolbar code

+ Code + Text

Lalu tuliskan kode program ini


```
[15] # Abstraction (Abstraksi)
      from abc import ABC, abstractmethod

      class Kendaraan(ABC):
          @abstractmethod
          def jumlah_roda(self):
              pass

      class Mobil(Kendaraan):
          def jumlah_roda(self):
              return 4

      mobil = Mobil()
      print("Mobil memiliki", mobil.jumlah_roda(), "roda.")
```

Output :

 Mobil memiliki 4 roda.

## 11. \_\_init\_\_ dan \_\_str\_\_

Klik lagi toolbar code

+ Code + Text

Lalu tuliskan kode program ini

```
[14] # Menggunakan __init__ dan __str__
      class Produk:
          def __init__(self, nama, harga):
              self.nama = nama
              self.harga = harga

          def __str__(self):
              return f"Produk: {self.nama}, Harga: {self.harga}"

      produk = Produk("Laptop", 15000000)
      print(produk)
```

Output

 Produk: Laptop, Harga: 15000000