Menulis Module dan Kelas pada Python

Module Python adalah berkas teks berekstensi .py yang berisikan kode Python. Anda dapat mereferensi berkas .py apa pun sebagai modul. Modul-modul umum yang disediakan oleh Python Standard Library dan mungkin sudah terinstal secara default pada instalasi Python Anda. PIP juga dapat dimanfaatkan untuk menginstal modul atau library berikut dengan dependensi yang dibutuhkannya. Anda pun dapat membuat dan menghasilkan modul Python Anda sendiri.

Menulis Modul

Menuliskan modul pada bahasa Python dapat dimulai dengan menuliskan definisi fungsi, kelas, dan variabel yang dapat digunakan kembali pada program lainnya. Misalkan saja kita membuat berkas hello.py yang akan kita panggil di berkas lain.

hello.py

```
1. # Define a function
2. def world():
3.     print("Hello, World!")
```

Jika hello.py dijalankan, maka program tidak akan menjalankan apapun karena world() hanya berupa definisi fungsi, kita belum memanggilnya. Jika kita biasanya memanggil sebuah fungsi dari berkas yang sama di bagian main, kali ini kita akan membuat berkas lain main.py yang seolah mengimpor hello.py. Pastikan hello.py dan main.py berada dalam satu direktori agar dapat diimpor dan dipanggil.

main.py

```
1. #impor
2. import hello
3.
4. #panggil
5. hello.world()
```

Saat memanggil sebuah fungsi dari modul yang kita impor, jangan lupa untuk menambahkan nama modulnya diikuti tanda titik, baru fungsi yang akan kita panggil. Dalam hal ini karena kita mengimpor hello.py, maka cukup kita tulis import hello, dan saat memanggilnya dengan hello.world(). Selain itu, kita juga dapat menggunakan from ... import ..., dalam hal ini adalah from hello import world dan memanggil fungsinya langsung

Sekarang, saat memanggil main.py, maka akan menghasilkan:

Menambahkan variabel

Menambahkan variabel pada modul hello, tambahkan variabel nama, misalnya Dicoding.

hello.py

```
1. def world():
2.     print("Hello, World!")
3. 
4. nama = "Dicoding"
```

Berikutnya, kita coba cetak variabel nama.

main.py

```
1. #impor
2. import hello
3. 
4. #panggil
5. hello.world()
6. 
7. #cetak
8. print(hello.nama)
```

Saat Dijalankan:

```
Hello, World!
Dicoding
```

Menambahkan kelas

Contoh yang lain, mari tambahkan kelas di modul hello. Kita akan membuat kelas Reviewer dengan atribut nama dan kelas, serta fungsi review() yang akan mencetak atribut yang telah didefinisikan.

hello.py

```
    print("Hello, World!")

nama = "Dicoding"

class Reviewer:
    def __init__(self, nama, kelas):
        self.nama = nama
        self.kelas = kelas

    def review(self):
        print("Reviewer " + self.nama + " bertanggung jawab di kelas " + self.kelas)
```

Tambahkan kelas pada main.py.

main.py

```
#impor
import hello

#panggil
hello.world()

#cetak
print(hello.nama)

#review
diko = hello.Reviewer("Diko", "Python")
diko.review()
```

Seperti umumnya kelas pada bahasa pemrograman lainnya, Fungsi dan Atributnya dapat diakses setelah kita melakukan instansiasi. Fungsi Review adalah fungsi yang melekat pada kelas Reviewer. Kita juga dapat memanggil **diko.Nama** atau **diko.Kelas** sebagai atribut yang melekat di kelas tersebut.

Output:

```
Hello, World!
Dicoding
Reviewer Diko bertanggung jawab di kelas Python
```

Lihat kedua variabel "nama" yang dapat menghasilkan dua nilai berbeda, karena nama yang pertama (hello.nama) melekat pada modul, sementara **diko.Nama** adalah atribut nama pada kelas Reviewe cukup berhati-hati dalam memastikan variabel seperti pada pembahasan fungsi yang lalu.

Implementasi Kode

Seringkali, modul dimanfaatkan untuk dapat memisahkan antara definisi dan implementasi kode. Namun modul juga dapat berfungsi selayaknya program pada umumnya, yang juga langsung mengeksekusi dalam modul itu sendiri. Contohnya, kita buat hello2.py seperti berikut:

hello2.py

```
# Definisi
def world():
    print("Hello, World!")

# Panggil disini
world()
```

Kemudian bersihkan main.py hingga menyisakan import hello2 saja.

main.py

```
import hello2
```

Saat main_program dijalankan, langsung muncul:

```
Hello, World!
```

Sehingga modul dapat digunakan dengan berbagai metode pemanggilan, bergantung pada definisi, maupun implementasi.

Mengakses Modul dari Folder Lain

Jika Anda bekerja dengan beberapa proyek secara paralel, berikut adalah opsi untuk mengakses modul dari folder lain:

Menambahkan path folder

Opsi ini dipilih umumnya di tahap awal pengembangan, sebagai solusi temporer. Untuk mengetahui path pemanggilan utama, Anda perlu bantuan dari modul sys yang sudah tersedia. Impor modul sys di main program dan gunakan fungsi sys.path.append. Misalnya berkas hello.py kita berada di direktori /home/dico main.py di direktori lainnya. Anda cukup menambahkan path /home/dicoding pada main.py seperti

```
sys.path.append('/home/dicoding')

import hello
…
```

Menambahkan modul pada Python Path

Alternatif ini dapat dipilih saat Anda melakukan pemanggilan modul >1x. Pada intinya pilihan ini akan menambahkan modul yang Anda buat pada Path yang diperiksa oleh Python sebagai modul dan paket-paket bawaan. Anda dapat memanfaatkan sys.path kembali untuk mengetahui posisi Anda saat ini.

```
print(sys.path)
```

Anda mungkin akan menerima output seperti berikut, sangat bergantung dengan jenis environment Anda, tapi pilihlah (atau cobalah satu per satu) jika ada beberapa output.

```
'/home/dicoding/my_env/lib/python3.5/site-packages'
```

Pindahkan hello.py pada direktori di atas. Maka Ia akan dikenali sebagai sebuah modul yang dapat diimpor oleh siapa saja dalam environment tersebut.

Pada main_program.py cukup impor.

```
import hello
```

Pastikan path yang Anda assign tepat untuk menghasilkan pemanggilan yang tepat. Modul yang tersebar pada beberapa folder mungkin akan menghasilkan galat. Usahakan peletakan yang se-sederhana mungkin.

Sebelumnya

Selanjutnya

Penghargaan

image
click bila
belum muncul