Bab 11. Bekerja dengan Datetime

Di dalam Python terdapat pula module `datetime` yang bisa digunakan apabila dalam program kita membutuhkan pengolahan terhadap tanggal dan waktu. Misalnya cara untuk mendapatkan tanggal dan waktu sekarang, atau mendapatkan tanggal n hari setelah hari ini atau n hari sebelum hari ini, dan sebagainya.

Untuk menggunakan function-function pengolah tanggal dan waktu di Python, terlebih dahulu module datetime dipanggil dengan cara memberikan perintah:

```
import datetime
```

atau bisa juga kita mengimportnya dengan cara memberikan perintah

```
from datetime import *
```

Function **now()**

Function `now()` digunakan untuk membaca tanggal dan waktu saat program dijalankan. Dalam hal ini tanggal dan waktu yang terbaca diambil dari waktu yang ada di komputer. Contoh penggunaannya adalah:

```
import datetime
skrg = datetime.datetime.now()
print('Saat ini adalah : ', skrg)
```

```
Saat ini adalah :  2018-12-17 06:52:17.154641
```

atau bisa juga ditulis seperti berikut

```
from datetime import *
skrg = datetime.now()
print('Saat ini adalah : ', skrg)
```

```
Saat ini adalah :  2018-12-17 06:53:35.380748
```

Sebagai catatan, waktu dan tanggal yang tampil adalah pada saat program dijalankan.

Apabil diperhatikan, format output dari function `now()` adalah `tahun-bulan-tanggal jam:menit:detik`. Dalam hal ini format tersebut tidak bisa kita ubah-ubah sesuai yang diinginkan. Selanjutnya bagaimana jika ingin mengubah format tersebut? Untuk mengubah formatnya kita bisa lakukan namun menggunakan trik khusus. Sebagai contoh, misalkan kita ingin menampilkan tanggal sekarang menjadi format tanggal/bulan/tahun. Berikut ini adalah program yang dapat digunakan untuk menampilkan tanggal dalam format tersebut.

```
from datetime import *

skrg = datetime.now()
tglskrg = str(skrg.day) + '/' + str(skrg.month) + '/' + str(skrg.year)

print('Tanggal sekarang : ', tglskrg)
```

```
Tanggal sekarang :  17/12/2018
```

Perhatikan contoh program yang diberikan. Di dalam program tersebut, perintah `skrg.day` digunakan untuk membaca properti tanggal dari variabel `skrg`. Untuk membaca bulan dari variabel `skrg`, digunakan `skrg.month`. Sedangkan untuk membaca tahunnya dengan `skrg.year`.

Selain properti `day`, `month`, dan `year`, properti lainnya yang bisa kita gunakan untuk mengambil dari properti datetime adalah `hour` (untuk membaca properti jam), `minute` (untuk membaca properti menit), dan `second` (untuk membaca properti detik). Perhatikan contoh berikut ini:

```
from datetime import *

skrg = datetime.now()

print('saat ini adalah : ', skrg)

print('tahun sekarang  : ', skrg.year)
print('bulan sekarang  : ', skrg.month)
print('tanggal sekarang: ', skrg.day)

print('jam sekarang    : ', skrg.hour)
print('menit sekarang  : ', skrg.minute)
print('detik sekarang  : ', skrg.second)
```

Output dari program tersebut adalah

```
saat ini adalah :  2018-12-17 06:56:39.794694
tahun sekarang  :  2018
bulan sekarang  :  12
tanggal sekarang:  17
jam sekarang    :  6
menit sekarang  :  56
detik sekarang  :  39
```

Selain menggunakan `now()`, kita juga bisa menggunakan `today()` untuk mendapatkan tanggal dan jam saat ini.

```
from datetime import *
skrg = datetime.today()
print('Saat ini adalah : ', skrg)
```

```
Saat ini adalah :  2018-12-17 06:57:54.950420
```

Function `date()`

Function `date()` digunakan untuk mengambil bagian tanggal (YYYY-MM-DD) dari serangkaian tanggal dan waktu. Misalkan dari function `now()`, kita hanya ingin ambil bagian tanggalnya saja maka bisa menggunakan date(). Contoh penggunaannya:

```
from datetime import *

# mengambil tanggal dari now()
skrg = datetime.date(datetime.now())
print('Tanggal hari ini: ', skrg)
```

```
Tanggal hari ini:  2018-12-17
```

Function `time()`

Selanjutnya bagaimana cara mengambil bagian waktunya (jam:menit:detik) dari format tanggal datetime yang dihasilkan, misalkan dari hasil `now()`? Caranya adalah menggunakan function `time()`. Contoh implementasinya:

```
from datetime import *

# mengambil waktu dari now()
skrg = datetime.time(datetime.now())
print('Waktu saat ini: ', skrg)
```

```
Waktu saat ini:  06:59:11.215409
```

Function `timedelta()`

Permasalahan berikutnya, misalkan hari ini adalah tanggal 10 Pebruari 2018. Bagaimana caranya kita mendapatkan tanggal 45 hari ke depan atau 10 hari sebelumnya dengan menggunakan Python? Tidak perlu khawatir, karena di dalam Python terdapat fitur timedelta() yang bisa memudahkan kita menentukan waktu ke depan atau sebelumnya.

Perhatikan contoh program berikut ini untuk menentukan tanggal berapa setelah 15 hari dari tanggal hari ini.

```
from datetime import *

#membaca tanggal sekarang
x = datetime.date(datetime.now())
print('tanggal sekarang     : ', x)

#mendapatkan tanggal 15 hari ke depan
#dari tanggal sekarang
y = x + timedelta(days=15)
print('15 hari lagi tanggal : ', y)
```

```
tanggal sekarang     :  2018-12-17
15 hari lagi tanggal :  2019-01-01
```

Dalam program di atas, ekspresi `y = x + timedelta(days=15)` digunakan untuk menentukan tanggal 15 hari setelah hari ini. Dalam hal ini, `x` adalah tanggal hari ini sedangkan `y` adalah tanggal 15 hari berikutnya.

Selanjutnya bagaimana jika mau menentukan 20 hari sebelum hari ini? Caranya cukup mudah, yaitu dengan memberikan operator −. Misalkan `x` adalah tanggal hari ini, dan `y` adalah 20 hari sebelumnya maka `y` dapat diperoleh dengan ekspresi `y = x - timedelta(days=20)` atau bisa juga menggunakan `y = x + timedelta(days=-20)`. Contoh implementasinya, perhatikan program berikut ini

```
from datetime import *

#membaca tanggal sekarang
x = datetime.date(datetime.now())
print('tanggal sekarang        : ', x)

#mendapatkan tanggal 20 hari sebelumnya
#dari tanggal sekarang
y = x + timedelta(days=-20)
print('20 hari sblmnya tanggal : ', y)
```

```
tanggal sekarang        :  2018-12-17
20 hari sblmnya tanggal :  2018-11-27
```

Selain parameter days dalam timedelta(), parameter lainnya yang bisa digunakan sebagaimana tersaji pada Tabel 11.1

Tabel 11.1 Parameter function timedelta()

Parameter	**Kegunaan**	**Contoh**
seconds	Menentukan waktu dalam selisih n detik	`x = y + timedelta(seconds=-10)` menentukan waktu 10 detik sebelumnya

microseconds	Menentukan waktu dalam selisih n mikro detik	`x = y + timedelta(microseconds=20)` menentukan waktu 20 mikrodetik setelahnya
milliseconds	Menentukan waktu dalam selisih n mili detik	`x = y + timedelta(miliseconds=-30)` menentukan waktu 30 mili detik sebelumnya
minutes	Menentukan waktu dalam selisih n menit	`x = y + timedelta(minutes=40)` menentukan waktu 40 menit setelahnya
hours	Menentukan waktu dalam selisih n jam	`x = y + timedelta(hours=-50)` menentukan waktu 50 jam sebelumnya
weeks	Menentukan waktu dalam selisih n minggu	`x = y + timedelta(weeks=60)` menentukan waktu 60 minggu setelahnya

Bab 12. Pemrograman Berorientasi Obyek di Python

Kelebihan dari Python adalah memungkinkan membuat kode program menggunakan style berorientasi obyek (PBO). Tidak seperti bahasa pemrograman berorientasi obyek lainnya, misalnya Java, struktur PBO dalam Python lebih sederhana sehingga lebih mudah dipahami dibandingkan Java. Terdapat tiga kemampuan utama ada dalam PBO yang penting untuk diketahui, yaitu inheritance, encapsulation, dan polymorphisme.

Membuat Class

Di dalam PBO, class merupakan blue print atau rancangan dari suatu obyek. Pada Python, sebuah class dibuat dengan struktur sintaks sebagai berikut:

```
class namaClass:
      ...
      ...
```

Selanjutnya, di dalam class bisa ditambahkan atribut atau properti dengan sintaks:

```
class namaClass:
      def __init__(self, value1, value2, …):
            self.atribut1 = value1
            self.atribut2 = value2
            .
            .
```

Perintah ‘self’ bermakna penunjukan ke class itu sendiri. Perintah ini sama dengan penggunaan kata ‘this’ apabila dalam Java atau yang lainnya.

Sebagai contoh, misalkan akan dibuat class dengan nama Manusia. Class manusia memiliki beberapa atribut, yaitu nama dan umur.

```
class Manusia:
    def __init__(self, value1, value2):
        self.nama = value1
        self.umur = value2
```

Setelah class dibuat, selanjutnya bisa dibuat sebuah obyek dari class tersebut. Obyek merupakan wujud implementasi dari sebuah class. Misalnya akan dibuat dua buah obyek dari class Manusia, yaitu amir dan budi. Apabila amir dan budi diprint, maka akan menunjukkan sifat obyeknya yaitu termasuk obyek Manusia.